

---

# **User Interfaces for Accessing Information in Digital Repositories**

---

**Jon Olav Hauglid**

Department of Computer and Information Science  
Norwegian University of Science and Technology  
Trondheim, 2004



---

# Abstract

---

Technological advances have made more information available to a larger part of the population than ever before. At the same time, the ability to locate and retrieve relevant information has become much more central. Combined, these trends form the motivation for the work presented in this thesis.

My research focus has been to study how user interfaces for accessing information in large repositories can be designed to provide assistance to users without impairing usability. To this end, existing solutions have been examined and key design challenges identified. Most important among these were supporting a wider variety of users and handling complex data models and large repositories. In order to study these challenges, a methodology of repeated design-implementation-evaluation was used. To focus the research, the implemented solutions have all been based on four fundamental design ideas as to how a simple, usable interface for large information repositories can be made.

The first presented system, SESAM, was designed for textual metadata databases. It was later extended to handle images. The second system, Savanta, targeted temporal video annotation databases. Both systems integrated various user interface techniques to create a rich and powerful environment for search, retrieval, browsing, exploration and analysis, while not sacrificing ease of use.

Several usability evaluations highlighted two main contributions. They demonstrated the power of an iterative interaction model which integrates several methods for accessing information. It was also showed that using dynamic analysis to derive high-level information about properties of a collection of information objects, is a powerful technique that makes it much easier to get an overview of and navigate in such collections.



---

# Preface

---

This thesis is submitted to the Norwegian University of Science and Technology in partial fulfilment of the requirements for the degree *Doktor Ingeniør*. The work has been carried out at the Database Systems Group which is a part of the Department of Computer and Information Science (IDI). The doctoral study was funded by Telenor.

## **Acknowledgements**

First, I would like to thank my advisor Associate Professor Roger Midtstraum for his continuous guidance and for many helpful discussions throughout my work.

For most of the time I worked on this thesis, I shared office with Jon Hegland. I would like to thank him for valuable feedback and for good cooperation on the Savanta project. I would also like to thank the remainder of the Database Systems group in general, and Associate Professor Kjetil Nørvåg and Olav Sandstå in particular.

The Primus database used as a test case throughout much of this thesis was kindly provided by the Norwegian Folk Museum. Without this database, my work would have been much more difficult. I would especially like to thank Trond Bjorli, Lene Walle, Steinar Bjørneset, Stein Langørgen and Hege Holmen for their assistance.

I would like to thank Professor Mads Nygård for many valuable comments on the thesis. I also thank graduated students Jan Obrestad, Rune Rystad, Magnus Grøtan and Anders Langmyr for their contributions.

Finally, I thank my friends and family for constant understanding and encouragement. In particular, I would like to thank my father for proof-reading the thesis.



---

# Contents

---

<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Motivation .....	1
1.2 Research question .....	5
1.2.1 Fundamental design ideas .....	6
1.2.2 Approach .....	8
1.3 Research methodology .....	9
1.4 Thesis organization .....	10
<b>Chapter 2: An introduction to user interfaces</b>	<b>13</b>
2.1 The role of user interfaces .....	13
2.2 Characteristics of user interfaces .....	15
2.3 A general classification of user interfaces .....	23
2.3.1 Command entry .....	24
2.3.2 Menus and navigation .....	25
2.3.3 Form-fills and spreadsheets .....	26
2.3.4 Natural language dialogue .....	28
2.3.5 Direct manipulation .....	29
2.3.6 Concluding remarks .....	31
<b>Chapter 3: Evaluating user interfaces</b>	<b>33</b>
3.1 Why evaluate? .....	33
3.2 Gathering information .....	35
3.2.1 Quantitative data .....	35
3.2.2 Qualitative data .....	36
3.3 Selecting test subjects .....	37
3.4 Methods of evaluation .....	40
3.4.1 Performance measurement .....	41
3.4.2 Questionnaires .....	42
3.4.3 Logging .....	43

---

3.4.4 Inspection .....	44
3.4.5 Interviews .....	45
3.4.6 Observation .....	46
3.4.7 Concluding remarks .....	47
3.5 Potential pitfalls .....	48
3.5.1 Reliability .....	48
3.5.2 Validity .....	49
3.6 Summary .....	50
<b>Chapter 4: Accessing information</b>	<b>51</b>

---

4.1 Different strategies for information access .....	51
4.2 Overview of existing interfaces .....	54
4.2.1 Textual query languages .....	54
4.2.2 Query-by-Example .....	56
4.2.3 Forms .....	57
4.2.4 Content-based multimedia queries .....	59
4.2.5 Information Retrieval .....	62
4.2.6 Dynamic queries and query preview .....	65
4.2.7 Natural language queries .....	68
4.2.8 Category browsing .....	71
4.3 Key challenges .....	74
4.3.1 Large information repositories .....	74
4.3.2 New classes of users .....	75
4.3.3 Complex data models .....	76
4.3.4 Different information access strategies .....	77
4.3.5 Multimedia data types .....	78
4.4 Summary .....	79
<b>Chapter 5: Four fundamental design ideas</b>	<b>81</b>

---

5.1 Revised interaction model .....	82
5.2 Intra-result analysis .....	86
5.3 Active user interfaces .....	88
5.4 Dynamic user interfaces .....	90
5.5 The road ahead .....	92
<b>Chapter 6: Searching supported by analysis of metadata</b>	<b>95</b>

---

6.1 Primus .....	96
6.2 Applying design ideas to textual metadata databases .....	100
6.2.1 Intra-result analysis .....	100
6.2.2 Revised interaction model .....	101
6.2.3 Active user interface .....	102
6.2.4 Dynamic user interface .....	102
6.3 The SESAM approach .....	103

---



---

6.3.1 Applying SESAM to a database .....	103
6.3.2 Properties of useful questions .....	104
6.3.3 Comparing the quality of questions .....	105
6.4 Related approaches .....	112
6.5 Summary .....	115
<b>Chapter 7: The first SESAM prototype</b>	<b>117</b>

---

7.1 The prototype .....	118
7.1.1 Interaction model .....	118
7.1.2 Overview of a sample search .....	119
7.1.3 User interface .....	121
7.2 Usability evaluation .....	122
7.2.1 An interface based on dynamic query .....	123
7.2.2 A forms-based interface .....	124
7.2.3 Evaluation methods .....	126
7.2.4 Evaluation tasks .....	128
7.3 Usability evaluation results .....	129
7.3.1 User performance measurements .....	129
7.3.2 Results from observation .....	130
7.3.3 Results from questionnaire .....	132
7.3.4 Discussion .....	135
7.4 System performance evaluation .....	137
7.5 Summary .....	139
<b>Chapter 8: The second SESAM prototype</b>	<b>141</b>

---

8.1 Design .....	142
8.1.1 Different strategies for information access .....	143
8.1.2 Finding suitable filters .....	144
8.2 Implementation .....	147
8.2.1 The initial query .....	147
8.2.2 Presenting results .....	148
8.2.3 From questions to filters .....	151
8.2.4 Improved looks .....	154
8.3 Usability evaluation .....	155
8.3.1 Setting .....	156
8.3.2 Results .....	161
8.3.3 Discussion of evaluation results .....	168
8.4 Discussion .....	170
<b>Chapter 9: Extending SESAM to image databases</b>	<b>173</b>

---

9.1 Extracting image features .....	174
9.1.1 Segmentation .....	174
9.1.2 Shape .....	176
9.1.3 Colour .....	180

---

---

9.1.4 Texture .....	181
<b>9.2 Extending SESAM .....</b>	<b>182</b>
<b>9.3 Experiences .....</b>	<b>183</b>
9.3.1 Feature extraction .....	184
9.3.2 User interface .....	185
9.3.3 The four design ideas .....	186
<b>Chapter 10: Savanta .....</b>	<b>189</b>
<hr/>	
<b>10.1 Introduction .....</b>	<b>189</b>
10.1.1 Related work .....	190
10.1.2 Applying the design ideas to temporal multimedia annotation databases .....	192
<b>10.2 Design and implementation .....</b>	<b>195</b>
10.2.1 Stored metadata .....	195
10.2.2 Derived metadata .....	198
10.2.3 Visualization .....	202
10.2.4 Navigation .....	204
10.2.5 Filtering .....	206
10.2.6 Searching .....	207
<b>10.3 Usability evaluation .....</b>	<b>208</b>
10.3.1 Setting .....	208
10.3.2 Results of usability evaluation .....	216
10.3.3 Discussion of evaluation results .....	220
<b>10.4 Discussion .....</b>	<b>221</b>
<b>Chapter 11: Discussion .....</b>	<b>225</b>
<hr/>	
<b>11.1 Research background .....</b>	<b>225</b>
<b>11.2 A discussion of the four design ideas .....</b>	<b>226</b>
11.2.1 Revised interaction model .....	226
11.2.2 Intra-result analysis .....	227
11.2.3 Active user interfaces .....	227
11.2.4 Dynamic user interfaces .....	228
11.2.5 Design ideas related to type of repository .....	228
<b>11.3 Post-project reflections .....</b>	<b>229</b>
<b>Chapter 12: Contributions and future work .....</b>	<b>231</b>
<hr/>	
12.1 Contributions .....	231
12.2 Future work .....	232
<b>Appendix A: Detailed evaluation results .....</b>	<b>235</b>
<hr/>	
A.1 Evaluation of first SESAM prototype .....	235
A.2 Evaluation of second SESAM prototype .....	239
A.3 Evaluation of Savanta .....	242

---

---

<b>Appendix B: About Savanta</b>	<b>245</b>
<hr/>	
<b>Appendix C: Evaluation handouts</b>	<b>247</b>
<hr/>	
C.1 Evaluation of first SESAM prototype .....	247
C.2 Evaluation of second SESAM prototype .....	249
C.3 Evaluation of Savanta .....	252
<b>References</b>	<b>255</b>

---



---

# List of Figures

---

2.1	The Interaction Cycle (Abowd and Beale, 1991). .....	14
2.2	Command entry using Microsoft Windows XP CMD.EXE. ....	25
2.3	Drop down menus from Adobe FrameMaker. ....	26
2.4	Registration form for Oracle's Technology Network. ....	27
2.5	Natural language interface from www.askjeeves.com. ....	28
2.6	Equipping a character in Divine Divinity using direct manipulation. ....	30
3.1	Three dimensional classification of users (Nielsen, 1993). ....	38
3.2	Example of a closed evaluation question from QUIS. ....	43
4.1	Oracle's SQL*Plus. ....	55
4.2	Query-by-example from Microsoft Access. ....	56
4.3	Forms-based query interface from www.bibsys.no. ....	58
4.4	QBIC Layout search – from The State Hermitage Museum. ....	60
4.5	QBIC Colour search. ....	61
4.6	Google. ....	63
4.7	Google – Advanced search. ....	63
4.8	Google – Result presentation. ....	64
4.9	Dynamic HomeFinder. ....	66
4.10	An example of Query Preview. ....	68
4.11	The START Natural Language System. ....	69
4.12	Example of a result from a START query. ....	70
4.13	Yahoo! Directory. ....	72
5.1	Standard model of interaction (Baeza-Yates and Ribeiro-Neto, 1999). ....	82
5.2	Revised model of interaction. ....	84
5.3	The Interaction Cycle (Abowd and Beale, 1991). ....	90
6.1	Subset of the Primus data model. ....	97
6.2	PrimusWeb. ....	98
6.3	Primus query interface. ....	99

---

6.4	Colour-distribution of cars. ....	101
6.5	Database-specific decisions. ....	104
6.6	Utility function for yes/no-questions for categorical data. ....	107
6.7	Different frequency distributions. ....	108
6.8	Sample distributions for categorical data. ....	110
6.9	The Eureka interface. ....	113
6.10	The OBIWAN system. Items suggest after a query for “wireless data systems”. ....	114
7.1	SESAM’s interaction model. ....	118
7.2	Phases in a typical search. ....	120
7.3	The interface of the first prototype. ....	121
7.4	Dynamic query interface. ....	123
7.5	Forms based interface. ....	125
7.6	Results from questionnaires. ....	134
7.7	Time spent on initial search and analysis as a function of load. ....	138
8.1	Initial query interface. ....	148
8.2	Result display from first prototype. ....	148
8.3	Result display from second prototype. ....	149
8.4	Detailed information from second prototype. ....	150
8.5	Display of questions from first prototype. ....	151
8.6	Constructing filters for categorical data. ....	152
8.7	Constructing filters for numeric data. ....	152
8.8	Display of “my filters”. ....	153
8.9	The user interface of the second SESAM prototype. ....	155
8.10	Updated version of dynamic query interface. ....	156
8.11	Updated version of the forms based interface. ....	158
8.12	Results from questionnaires - Computer professionals. ....	165
8.13	Results from questionnaires - Museum conservators. ....	167
9.1	Sample image from the Primus database. ....	175
9.2	Two examples of image segmentation. ....	176
9.3	Stages in the process of extracting shape information. ....	177
9.4	Standardized shapes used for classification. ....	179
9.5	Partitioning of the CIELab colour space. ....	181
9.6	Shape filter interface. ....	182
9.7	Colour filter interface. ....	183
9.8	Examples of problematic images to segment. ....	184
9.9	Rotating a 3D object alters shape of 2D projection. ....	186
10.1	Conceptual model of temporal annotation databases. ....	193
10.2	Interaction model for proposed system. ....	194
10.3	The OntoLog data model. ....	196
10.4	Simplified annotation model. ....	197
10.5	Stored metadata in Savanta. ....	197
10.6	The War in Iraq described by The Middle East; related to	

---

---

George W. Bush; differs from Afghanistan. ....	200
10.7 Terms, timeline and intervals. ....	202
10.8 Collapsed term and aggregated intervals. ....	203
10.9 Savanta. ....	203
10.10 Media navigation controls. ....	205
10.11 Hypertext navigation panel. ....	205
10.12 Filters. ....	207
10.13 Savantoogle. ....	210
10.14 Savantapplet, Savantoogle's interface for presenting a single media "document". ....	211
10.15 Forms interface. ....	213
10.16 Results from questionnaires. ....	219

---





---

# Chapter 1

# Introduction

---

The topic of this thesis is user interfaces for accessing information in digital repositories. The focus is on usability combined with special support for handling large repositories.

In this chapter, I present the motivation for this topic and the research methodology I have used. I also describe the organization of the thesis.

## 1.1 Motivation

---

Throughout history, collecting and managing information have always been important tasks for the human race. These tasks were relatively easy in the beginning, as information carriers such as cave paintings and oral communication limited the amount of information that could be transferred and archived. This all changed with the emergence of written and printed media. These new media made it possible to have information repositories that were simply too large to be managed in an unstructured manner. Organizing content by topic, title and author, therefore became a necessity – just imagine finding a specific book in a library where the books are not organized in a recognizable manner.

As we have entered the information age, the characteristics of information repositories have changed dramatically. Not only is physical size all but removed as a limiting factor, but modern communication technology has made more information accessible to a larger part of the world's population than ever before. While having access to enough information available used to be the problem, today we usually have so much information available that the problem is rather to separate what is relevant from what is not. Typically, organization alone is no longer enough to

make today's huge repositories usable in practice. This means that there is great need for ways of providing support to the process of accessing information.

The main topic of this thesis is to study computerized tools for information access in digital information repositories with focus on the role of user interfaces. In order to gain insight into this field, the work includes an overview of existing tools for information access, an identification of possible avenues for improvements; as well as design, implementation and evaluation of my own solutions.

The motivation behind this topic comes from the significant technological advances that have been made in information and communication technology during the last decades. Three distinct areas come to mind, namely advances in digital communication, processing power and storage technologies. Combined, they have not only been instrumental with regards to advances in software technology in general, but also fundamentally changed the environment where computerized information access tools can play a role.

As each of these areas has far-reaching consequences for the topic of this thesis, they are described below along with a brief discussion of how they pose new challenges for the development of tools for information access.

### **Advances in communication technology**

Digital communication technology allows us to transfer huge amounts of information, blazingly fast over large distances. With the increased implementation of the required infrastructure, information is now more accessible than ever before. In my view, the single most important development in this respect, is the rise of the Internet as the global communication network. By tying together computers all around the world, huge information resources are now available from computers everywhere.

In comparison, traditional information systems have above all been characterised by a small and homogeneous user group. Before the Internet emerged, there was really no easy way for the mass public to directly access an information repository electronically. Thus, most systems were only used within an organization. As a consequence, the number of potential users was fairly low and it was feasible to design user interfaces that required user training. Also, one could be fairly certain that the users had reasonably comparable requirements, background knowledge and experience. All this made it reasonable to design user interfaces

---

---

tailor-made to a very uniform group of users. The fact that this could make them almost inaccessible to other groups of users, was of little or no concern.

However, with the increased communication capabilities, it became possible to reach a much wider audience. This poses an important challenge as different classes of users often have conflicting requirements and qualifications. What might be an optimal solution for an experienced user, might be completely incomprehensible to a novice. Designing one general interface to “rule them all” might be possible, but it certainly will not be easy.

### **Advances in processing power**

The world’s first electronic digital computer, ENIAC, was developed in the latter half of World War II by the U.S. military for calculating ballistic firing tables. The 167 square meter monstrosity was able to perform 5000 additions per second (Bellis, 2000). By comparison, a modern workstation processor can do in excess of 500 million additions per second.

This tremendous increase in processing power has naturally had its effects on information repositories. For one, it has allowed ever larger repositories to be handled without prohibitive consequences for response times. Also, more processing power allows more time-consuming algorithms and techniques to be used to offer new functionality.

An example of such new functionality, is the handling of other data types than text and numbers. More complex media types such as images, audio and video can now be processed, displayed and stored. This has changed the computer from being an electronic word processor or calculator to a multimedia workstation. A natural consequence of this development is the emergence of digital multimedia repositories. As data types used in such repositories are fundamentally different from text and numbers (Subrahmanian, 1998), it is often necessary to develop new techniques for information access. For example, the meaning of textual data is inherent in the data itself. Stored video, on the other hand, is just a sequence of bitmap images – coloured dots placed in a grid. The meaning of such data is highly dependent on the viewer and is the result of a high-level cognitive process difficult to recreate in a computer and thus difficult to support in information access tools. It is therefore difficult to have a dialogue between user and system in a language that is natural to both. Further, designing a user interface for multimedia navigation is

---

difficult due to the spatial and/or temporal nature of images, audio and video. While text can be summarized and rapidly browsed, this is much harder to do for temporal media. Assisting the user in gaining an overview of a collection of multimedia objects, is therefore a important challenge for multimedia access tools.

### **Advances in storage technology**

In 1965, Gordon Moore of Intel Corporation predicted that the number of transistors on a single chip would quadruple every third year (Stallings, 2003). This prediction, later dubbed Moore's law, has proven to be more than reasonably accurate. Together with advances in magnetic and optical storage media, it has given us tremendous increases in storage capacity. As an example, the most powerful version of the original IBM Personal Computer from 1980 had 256 KB primary storage and 320 KB secondary storage. Today (2004), we can easily find 1 GB memory modules and 250 GB hard disk drives.

This development has certainly been a necessary prerequisite for the emergence of multimedia repositories (as described above), but it also has had a clear impact on text repositories. With increased storage capacity and larger repositories, users are often forced to search huge amounts of data to find the information they are looking for. This might be no problem if it is easy for the user to separate relevant information from irrelevant data. Unfortunately, this is often not the case. Frequently the average relevance of retrieved objects, for example items in a query result, can be quite low.

This problem can be greatly reduced if the user interface is able to provide meaningful assistance in separating relevant and irrelevant information. As ever more fine grained control is required to do this successfully, the design and capabilities of the interface become more important. Unfortunately, the expressivity needed is often difficult to achieve without negatively affecting interface simplicity and usability. This can easily lead to a choice between complex interfaces or unmanageable amounts of data.



Ever since the dawn of man, collecting and managing information have been one of our most important challenges. As information repositories have grown larger, simply organizing the content in a logical way is no longer enough. Computerized repositories need to offer ways to help users in locating what they are looking for. Much work has already been

---

done in this field – especially with regards to the handling of textual data. However, technological advances in several key areas have made several new possibilities and challenges apparent. These form the motivation for the work I will present in this thesis.

## 1.2 Research question

---

Based on the discussion of current developments and challenges in the previous section, the main research question of this thesis is:

*How can user interfaces for accessing information in large repositories be designed to provide assistance to users without impairing usability?*

This is a quite broad research question. In order to narrow the scope, I have decided to focus my research on five important challenges related to the research question. These five key challenges are presented in detail in Section 4.3, but an overview is given below:

◆ **Large information repositories**

As repositories grow larger, which consequences will this have to their usefulness and the associated interfaces' usability?

◆ **New classes of users**

Users accessing a program via the Internet, cannot be expected to read manuals and receive training. How can interface design take this into account?

◆ **Complex data models**

In order to make use of data stored in a database, it is important to understand how the data is structured (the data model). As model complexity grows, this task becomes more difficult – especially for users with limited training and background knowledge. Can the interface be used to mask this complexity?

◆ **Different information access strategies**

Writing an exact query is only one way in which a repository can be accessed. Does it make sense to support different strategies and can this be done without sacrificing usability?

◆ **Multimedia data types**

With increased storage and processing capability comes the possibility of handling complex data types such as images, sound and video. How can large multimedia repositories be accessed in a usable and efficient manner?

---

Even if these five challenges are more focused than the overall research question, they are still too broad and encompassing for the in-depth study a thesis is meant to be. For this reason, I will use four fundamental design ideas to address the key challenges. These represent my suggestions as to how simple, usable interfaces for large information repositories could be designed. By designing, implementing and evaluating several interfaces based on these four ideas, I expect to gain insight into the research question and the key challenges.

### **1.2.1 Fundamental design ideas**

The four ideas are presented in detail in Chapter 5, but a brief introduction is given in this section in order to better explain my research approach.

#### **1: Revised interaction model**

Traditionally, interfaces for information access have users issue a textual query, forward this to a query processing system, and present the resulting answer to the user (Baeza-Yates and Ribeiro-Neto, 1999). If the answer does not satisfy the information need of the user, the process is repeated.

I find this interaction model lacking in two respects. For one, a result is final – if it is not satisfactory, one has no choice but to restart the whole process. Secondly, queries are only one of several possible methods for accessing information repositories. By providing more options to users, they get more ways to reach their goal and are less likely to get stuck. I am therefore interested in studying the performance of a revised iterative model where results can form the basis for further interaction and where several, integrated access methods are available.

#### **2: Intra-result analysis**

Typically, systems for accessing information are client-server-based with the client more or less just presenting information retrieved from the server. Thus, processing capabilities on the client are largely left unused. In my opinion, these resources could be utilized to enhance the capabilities of the system as a whole.

The avenue I am interested in exploring, is if metadata, perhaps otherwise left on the server and thus unavailable to the user, could be analysed in order to derive new information. For example, automatic categorization, summarization etc. can provide high-level information that

---

---

otherwise might be unavailable to the user and might be helpful in gaining an overview of the presented information.

### **3: Active user interfaces**

In order to construct usable interfaces, one must take the capabilities of human beings into account. One of the most important areas in this respect is the human memory system (Miller, 1956). Two central memory concepts are *recognition* (e.g. recognizing a person you have met before) and *recall* (e.g. recalling from memory how someone looks). Not surprisingly, recognition has been found to be much easier than recall (McCracken and Wolfe, 2004).

One way to apply this finding to the world of user interfaces, is to have users as often as possible respond to information provided by the system. In this way, users can recognize options that might be helpful to them rather than to be forced to recall which actions the user interface supports and what their effects are. In this way, the interface becomes more active by often (where applicable) initiating interaction rather than simply waiting to be used.

### **4: Dynamic user interfaces**

Interaction between a human and a computer can be seen as a cycle where the user expresses actions using input devices, is presented with results using output devices and uses these as a basis for new actions (Abowd and Beale, 1991). Presenting system output and interaction controls are the domain of the user interface. Usually, the user interface design for a given system is constant – only the content changes.

If the presentation format and controls are constant, they will not necessarily always be optimum for a given content. A better solution might be to have a dynamic user interface where the presentation design and the controls depended on previous interaction and the content to be displayed. This would amount to a context sensitive interface.



These four design ideas do not necessarily represent something brand new when it comes to interface design. Even when restricted to interfaces for information access, previous work exists in various degrees. My contribution will rather be in clarifying the concepts, integrating them, in the specific solutions I come up with and in evaluating any positive synergy effects that might arise when all four are combined.

---

## **1.2.2 Approach**

The way in which I intend to approach the research question is to make designs and implementations based on the four design ideas above. These implementations will then be subjected to evaluations – mostly concerning usability. The results from these evaluations are expected to provide insight into the applicability of the design ideas with respect to the research question.

In order to provide a wider platform for evaluation, I have singled out three different types of information repositories. By implementing information access tools for each of these types of repositories, a broader understanding of the relevance of the design ideas can be gained. It will also allow me to study the characteristics of different types of repositories.

The three types of information repositories to be studied are:

### **1: Textual metadata databases**

Text is the most fundamental information carrier, and techniques applied to other media often originate from the textual domain. This makes it natural to start by studying information access in textual repositories and to have this part be the main part of the thesis.

Rather than examining all kinds of textual repositories, I have chosen to focus on textual metadata databases. This is in part due to my previous experience with databases and the availability of a large real-world textual metadata database, but mostly due to the unique possibilities offered by database schemas. A database schema (or data model) is a collection of concepts used to describe the static structure of a database (Garcia-Molina et al., 2002). According to the schema, every object will have a number of properties, which typically includes both attributes and relationships to other objects. One of the main ideas presented in this thesis, is that the metadata represented by attributes and relations could be used to improve information access.

As this is the first setting in which the design ideas will be evaluated, a two-phased approach will be used. Based on the results of a usability evaluation of an initial prototype, a second prototype is implemented and evaluated. This not only makes it possible to improve the reliability and validity of my findings, but also serves as an evaluation of the usability evaluation procedure in itself.

---



## **2: Image databases**

Complex media, such as images, audio and video, are perceived in fundamentally different ways by humans and computers. While we use object recognition, spatial placement and previous knowledge to make sense of an image, it is difficult to make a computer see more than just pixels. As a common language is a fundamental requirement for successful communication, this semantic gap between humans and computer “vocabulary” makes information access challenging and an interesting topic for study.

I aim to design and implement a solution based on my four design ideas to investigate if they can help to close the semantic gap. This will also serve to study the characteristics of image databases with respect to the research question.

## **3: Temporal multimedia annotation databases**

Temporal data is data that varies with time – for example information about which characters are on screen in a movie or the salaries for persons in a company. As including time adds an extra dimension, it increases the complexity of the repository and makes it more difficult to manually locate information as well as to identify patterns and trends.

In this thesis, I use temporal multimedia annotations as an example of temporal data. Temporal annotations are time-dependant descriptions – in this case of video clips. For example, annotations could be used to describe which actors are present in which parts of the clip.

The example data comes from the OntoLog project for Ontology-based annotation of video and audio and was carried out by Jon Heggland, a colleague of mine, simultaneously with the work presented in this thesis. By designing and implementing a tool for accessing temporal multimedia annotations based on the four design ideas presented in Section 1.2.1, I will investigate if patterns in the data can be made more accessible to the user and therefore help increase the usefulness of such a database.

## **1.3 Research methodology**

---

In (Denning et al., 1989) three different paradigms for the discipline of computing (encompassing both computer science and computer engineering) are described:

- ◆ **Theory**

- Establishing a theorem describing the objects of study; proving the theorem; interpreting the results.

---

**◆ Abstraction**

Forming a hypothesis; collecting data from an experiment; analyse the results.

**◆ Design**

Stating the requirements; designing and implementing a system; testing the system.

As this thesis is focused on user interface design, the latter paradigm is a natural choice – in essence designing and implementing several systems to investigate the research question. The three different areas to be studied were presented in the previous section.

The testing phase is of fundamental importance to the outcome of my research. As the research question is directed towards usability, it is natural that most of the testing will be usability evaluations. Chapter 3 is therefore in its entirety geared towards how to perform usability evaluations. In order to understand the requirements for information access applications, a study of the state of art in this field is presented in Chapter 4.

## **1.4 Thesis organization**

---

This thesis is divided into three parts:

### **Part 1: Introduction and background**

The first part contains a general introduction to user interfaces and usability evaluation as well as an overview of current interfaces for accessing information. It concludes with a detailed presentation of the four design ideas already mentioned in Section 1.2.2.

- ◆ *Chapter 1* contains this introduction.
  - ◆ *Chapter 2* is a brief and general introduction to user interfaces.
  - ◆ *Chapter 3* is a general introduction to user interface evaluation.
  - ◆ *Chapter 4* presents interfaces for information access, an overview of existing interfaces as well as a description of what I find are some of the most interesting challenges specific to such interfaces.
  - ◆ *Chapter 5* outlines the four design ideas that form the basis for later work.
-

---

## **Part 2: Design, implementation and evaluation**

This part of the thesis presents design, implementation and evaluation of tools for accessing information for each of the three different types of information repositories mentioned in Section 1.2.2. Textual metadata databases are studied in greatest detail and covered by Chapters 6, 7 and 8. Image databases are discussed in Chapter 9, while temporal metadata databases are the topic of Chapter 10.

- ◆ *Chapter 6* introduces SESAM – an approach for searching textual metadata databases based on the four design ideas from Chapter 5.
- ◆ *Chapter 7* presents the implementation of the first SESAM prototype. It also contains a description of a usability evaluation and a summary of the lessons learned in this study.
- ◆ *Chapter 8* presents the second prototype of SESAM redesigned with basis in chapter 7. This chapter also includes a new usability evaluation.
- ◆ *Chapter 9* is a short chapter presenting a modified version of SESAM extended to support an image database. The aim is to examine if the SESAM approach can be applied to content-based image access.
- ◆ *Chapter 10* contains a description of the design, implementation and evaluation of, Savanta, a tool for accessing data in temporal annotation databases which integrates several different methods for information access.

## **Part 3: Assessment**

- ◆ *Chapter 11* discusses the results from chapters 5 to 10.
  - ◆ *Chapter 12* contains overall conclusions and the contributions of this thesis.
-



---

## Chapter 2

# An introduction to user interfaces

---

In this chapter, I present a brief and general introduction to user interfaces. As this thesis' focus is on usable interfaces for accessing information, this chapter serves as a background for later chapters.

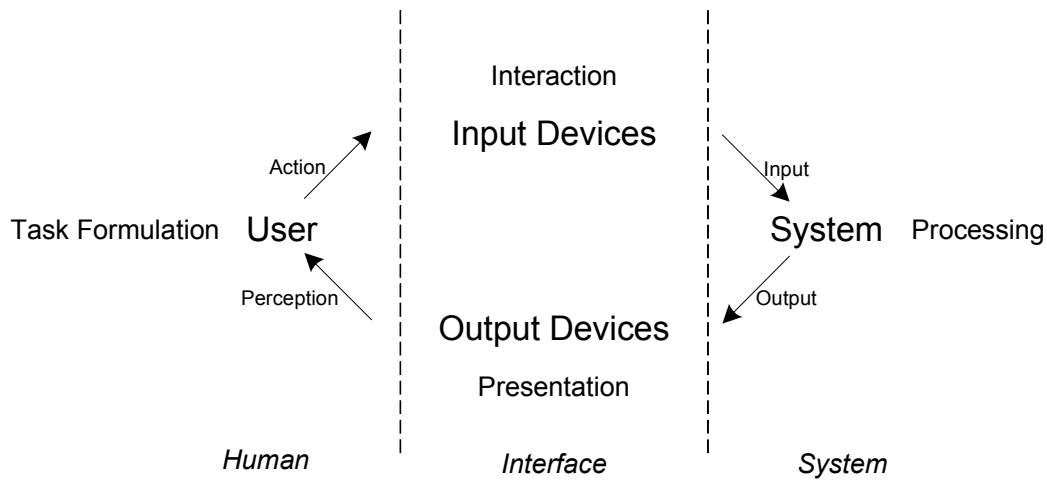
The first part of this chapter contains a general definition of user interfaces, while the second part examines general guidelines for good interface design. The third and final part presents a broad classification of existing interfaces.

## **2.1 The role of user interfaces**

---

The interaction between a user and a computer system can be seen as a cycle where the user performs actions using input devices, the system process these inputs and displays its output on output devices, which in turn is perceived by the user and used to formulate new plans of action.

This model was presented in (Abowd and Beale, 1991) as “The interaction cycle” and is illustrated in Figure 2.1. It highlights the role of the user interface as the intermediary between the user and the system.



**Figure 2.1 The Interaction Cycle (Abowd and Beale, 1991).**

According to this model, the interface has two duties – presenting system output and allowing user interaction. Presenting system output is done using output devices. Several visual or audible devices exist – see (Preece et al., 1994) and (Dix et al., 1998) for a thorough discussion.

Input devices are the interface components used to communicate a user's wishes to a system. In today's systems, the normal input devices are computer mice and keyboards, but several other more specialized devices exist. See (Preece et al., 1994) and (Shneiderman, 1997) for a discussion of the different alternatives and how they influence the interaction as a whole.

By observing the state of the system by the means of the output devices, users formulate a plan of action and execute it by the means of the input devices. Both observation and planning involve some sort of interpretation. Exactly how this interpretation is done is important to understand in order to design understandable interfaces. But because of the enormous human diversity with regards to, among other things, cognitive ability, physical ability and previous experiences, this task is far from easy. In the classic book “The Design of Everyday Things”, Donald A. Norman notes:

*“[...] designers are not typical users. They become so expert in using the object they have designed that they cannot believe that anyone else might have problems; only interaction and testing with actual users throughout the design process can forestall that.” (page 151)*

Due to the important role of usability testing, Chapter 3 is in its entirety devoted to this topic.



The user interface thus consists of both input and output devices. This corresponds well with Moran's (Moran, 1981) definition of user interface as

*“those aspects of the system that the user comes in contact with.”*

This obviously indicates that the user interface is of great importance – it is in fact the only part of a system that users ever see. As a logical consequence of this definition, the responsibility of user interfaces should be to make functionality in a program accessible to its users.

Further examples of the importance of the user interface exist. One previous study (Myers and Rosson, 1992) measured that the user interface constitutes 48 % of a program's code and a similar amount of the total time spent during design and implementation. Due to their importance, one would expect that designing usable interfaces should be a solved problem. Unfortunately, this is far from being true. One study (Nielsen, 2001) reports that 44 % of 496 attempts at performing tasks on e-commerce web sites failed. Another e-commerce study (Hurst and Terry, 2000) measured that as much as 43 % of those that try to purchase something on-line, fail because of usability problems. Clearly, there are still room for improvements.

## **2.2 Characteristics of user interfaces**

---

The purpose of a user interface is described in (Catarci, 2000) as:

*“It is responsible for informing the user about the possibilities, limits and functionalities of the system”.*

The degree in which this purpose is fulfilled, is often expressed as an interface's *usability* (Preece et al., 1994). Thus, an interface with high usability makes it easy for users to access most or all of the functionality provided.

Designing user interfaces is far from an exact science. Different programs can have very different objectives, target audiences and platforms, making it difficult to develop a single recipe for success. Nevertheless, general characteristics of successful interfaces have been and still are a topic of some interest. In the absence of any narrow and exact

---

rules for interface design, several sets of broad guidelines aimed at making it easier to design usable interfaces have been developed. One of these is Jakob Nielsen's list of ten usability principles (Nielsen, 1993) presented below.

### **1: Simple and natural dialogue**

The interaction between man and machine forms a dialogue. As for all other forms of communication, it is important that the dialogue is relevant to both parties. This suggests that it is desirable to keep user interfaces as simple and clean as possible – retaining only those interface components that are required as each additional component increases the cognitive burden. Not only must each additional component be interpreted, they also increase the number of possibilities to consider when users are trying to make a plan for further interaction.

As an example, it would make sense to move advanced or rarely used options from the main interface area to drop-down menus where they are more out of sight. Novice users, while probably not missing the advanced features, would then have to scan fewer components to find what they want.

But removing superfluous components will not help much if the remaining components make no sense to the user. When output is presented to the user, he or she perceives this output and interprets the way in which the computer concepts have been presented to map these concepts to their own mental model. A similar mapping must then be performed to express indented user actions using available interface controls. A natural dialogue between system and user can greatly simplify these mappings – for example by presenting information in a sequence that matches the way in which the user intends to work.

### **2: Speak the users' language**

In a human dialogue, one normally has the opportunity to ask for an explanation if a spoken term is not understood. In contrast, a man-machine interaction is far more static as the machine seldom is designed to dynamically adjust its dialogue according to the needs of the user. This makes it all the more important to design an interface which uses terms familiar to its users in the first place. For user interfaces, this means that they should strive to use terms that the user find natural and avoid technical dialogue. This minimizes the amount of interpretation needed to understand the presentation given by the system. For example, when combining several different query terms, it is usual to

---



allow the use of operators such as AND and OR. But as they require an understanding of boolean algebra, many systems have replaced them with the more intuitive “all of these terms” and “any of these terms” respectively.

A potential pitfall in this regard is the fact that most interfaces are designed by people with very different backgrounds compared to the backgrounds of their target users. Unintentional use of jargon and system-oriented terms can make interaction needlessly difficult.

An example of using system-oriented terms could be to ask a user if she wants to save a file to drive “A: or C:” These abstract labels carry no inherent meaning and would therefore not be of any use to new users. It would therefore be better to use the terms “floppy disk” and “hard drive” (preferably illustrated with icons).

B. Shneiderman (Shneiderman, 1997) argues that user interfaces should contain user-centered phrasing. This is achieved by making users initiators instead of responders by using forms such as “ready for data” rather than “enter data here”. The point is that the first phrase identifies the user as the one in command – the interface is simply waiting to be used. The latter phrase on the other hand, contains a commanding tone. The user is here being told what to do – the interface is clearly in command. As having the feeling of being in command, is a more pleasant experience than being told what to do, the first option is argued as superior.

### **3: Minimize the users' memory load**

One of the most prominent advantages a computer has compared to a human brain is its faultless memory storage. Therefore, it is bad practice to design a user interface which requires its users to remember information given earlier in the dialogue. This task should instead be handled by the interface by making relevant information easily accessible. Accessible information should also include the choices made by the user earlier in the program execution. However, the “less is more” rule still holds. By making too much information accessible, users become less likely to be able to take advantage of this aid at all.

This design principle is strongly tied to the principle concerning “Help and documentation” (described later) – especially regarding the use of online documentation. The use of pop-up “balloon help” windows containing short descriptions of the interface component currently under the mouse pointer is a good example of how to make relevant information accessible without cluttering up the interface. However, this must be used

---

with care as there are no visual clues to their existence before they pop-up. Users unfamiliar to “balloon help” windows can therefore easily miss them altogether.

#### **4: Consistency**

In a consistent interface, any given user action will always have the same effect and any given symbol always has the same meaning. This makes it easier for users to construct a mental model of how a program works. A user which has experienced that blue, underlined words represents clickable hyperlinks, will become very confused if clicking other underlined words produces no results.

An application should not only be internally consistent, consistency with other applications is also very helpful. If a user can draw upon previous experiences when adopting a new application, training time can be reduced (Nielsen, 1999). This has promoted the use of interface standards and guidelines such as (Sun Microsystems, 2002), (Microsoft Corporation, 2002) and (Apple Computer Inc., 1992).

#### **5: Feedback**

When mechanical machines are used, they typically offer visual, audible or tactile clues as to what they are doing. This is a great aid for their operators. Imagine trying to use a camera without any clues as to when the photo actually has been taken. As (non-digital) cameras automatically produce audible clues due to shutter movement, photographers have grown accustomed to listening for this sound as confirmation.

Computer programs, on the other hand, will not automatically offer such clues. A user has no means to monitor a time-consuming operation if this is not specially provided by the interface. It is therefore important that designers construct interfaces which permit users to observe the progress and result of an internal operation. This will prevent the user from wondering whether the program has crashed, expects further input or simply is behaving normally.

#### **6: Clearly marked exits**

As it is human to err, interfaces must be designed to handle a wealth of different user errors. One of the more common types of errors is starting the wrong program or invoking the wrong option. These errors become only a minor annoyance if they are easily escaped by the means of a “undo” or “cancel”-button or similar. If the user is instead “trapped” and

---

---

forced to endure an extended dialogue in order to exit, the user will become more insecure and unlikely to explore the interface.

This point can be illuminated by an example from my own experience. An early, text mode based editor I once used, contained an extensive online help system. In order to leave this help system, you had to press a seemingly random key combination. Not knowing this combination, I was unable to return to the editor and became forced to terminate the whole program. Needless to say, the help system went largely unused for my part.

## **7: Shortcuts**

Some of the previous usability principles have described the need to craft interfaces that are suited to the needs of the users. This is in itself not an easy task. A further complicating factor is that the needs of a given user change as she becomes more experienced. A detailed step-by-step guide to completing a task might be just what a novice user needs, but will likely be of hindrance to more advanced users.

Advanced users have understood so much of the inner workings of a system, that they focus more on the task than the tool. To let these users work efficiently, it is important that an application offers shortcuts to commonly used operations. This allows them to stop spending time navigating the user interface and have shorter time between perception and action. Shortcuts can for instance be implemented in the form of keyboard shortcuts as an alternative to menu options.

## **8: Good error messages**

Some sort of error situation is likely to appear during the execution of an application. Whether the cause is the user, the software or the hardware, error messages should inform the user of the event. An ideal message not only describes the error and what caused it, but also suggests constructive measures to be taken.

Error messages should also obey the principles regarding simple, natural and understandable dialogue. Internal error codes, debug messages and similar are of very limited use to the average user – in fact, they are more likely to do harm than good. In addition to user-centered phrasing, error messages should also avoid using a negative and accusing tone and instead try to be polite. No good could possibly be the result of blaming the user.

---

## **9: Prevent errors**

Even if good error messages are important, it would be better if errors could be avoided in the first place. Careful interface design can minimize user error by clearly indicating if an operation is legal or not. This can for instance be implemented by disabling buttons and menu options when invoking them would be an illegal operation, or by asking a user to select a name from a list of allowed choices instead of typing it in.

Another way of preventing errors is requiring confirmations for especially destructive operations. As a user can initiate the deletion of a file or the formatting of a hard disk by mistake, it makes sense to not perform them without confirmation.

## **10: Help and documentation**

Ideally, a program should be so easy to use that documentation in any form is unnecessary. This can however be difficult to accomplish – at least if the task to be solved is somewhat complicated. Thus, the quality of both online and offline help and documentation, becomes a factor in the overall usability of an application.

Online documentation can be designed to be easily accessible, eliminating the need to locate and manually search printed manuals. As a user typically needs help to execute a particular task, it makes sense to organise the online help by task. Studies have shown (Magers, 1983) that if the steps that need to be carried out are clearly listed (i.e. task-oriented), utility of the online help is improved.



These ten user interface guidelines are all general enough to be applicable to almost any user interface – both textual and graphical. That makes them useful for any user interface designer regardless of application type. I also find them helpful in that they offer a structured way of (informally) evaluating a given interface by examining how well each guideline is implemented.

Other attempts at providing general guidelines for interface design have also been published. Among these are Ben Shneiderman's “Eight Golden Rules of Interface Design” (Shneiderman, 1997). These rules agree with Nielsen's usability principles to a very high degree and therefore serve to confirm their validity. Minor dissimilarities include Shneiderman's emphasis on easy reversal of actions in order to reduce anxiety and the

---

---

weighting of user-centered phrasing. For a discussion of Sheiderman's rules, see (Baeza-Yates and Ribeiro-Neto, 1999).

A different perspective as to what constitutes good user interfaces is offered by B. Reeves and C. Nass in their book “The Media Equation” (Reeves and Nass, 1998). They claim that media equal real life – i.e. that people interact with media as they interact with real people. As a result, lessons learned in the social sciences can be applied to user interface design. An interesting example used by Reeves and Nass is H. Paul Grice's four maxims for social conversations (Grice, 1975): Quality, quantity, relevance and clarity.

◆ **Quality**

*“Speakers should say things that are true.”*

Thus, if a user feels misled by a program's user interface, the user will be less likely to interact with the program in the future.

◆ **Quantity**

*“Each speaker in an interaction should contribute only what the conversation demands, not too much or too little.”*

It is considered impolite both to be confusingly brief and to be boringly lengthy. This rule is made more difficult to adhere to because what one user considers suitable might be too lengthy to another. Also, the appropriate amount of text might change as the user grows more experienced. This conforms with interpersonal interaction where more and more information becomes implied rather than expressed as the participants grow accustomed to each other.

◆ **Relevance**

*“What people (and media) say should clearly relate to the purpose of the conversation.”*

This rule can be violated by for instance displaying buttons that can not be pressed and menu options that can not be selected. These violations will be perceived as deceptions – promising something that can not be done. This is clearly impolite in a social con-

---

text and will also be perceived as such in man-machine interaction.

◆ **Clarity**

*“Contributions to an interaction should not obscure.”*

It might seem obvious that user interfaces should not contain texts that are incomprehensible to its users. Nevertheless, this rule is often violated – particularly in the interest of keeping the texts short. This is somewhat of a paradox as comprehensibility clearly is more important than brevity.

One important property not explicitly covered by either of the presented set of guidelines, is that of *responsiveness* – that is the delay between user input and system output. It is common (Miller, 1968) to talk about four different categories of response time. If the delay is shorter than 0,1 second, the user will perceive the response as instantaneous. Between 0,1 and 1,0 second, the user will probably manage to stay focused – i.e. not lose the train of thought. Response times larger than 1,0 second will require the user interface to notify the user that the system is processing data in order to have the users not losing interest. Finally, with response times longer than 10 seconds the system will be perceived as boring and the user will want to perform other tasks in the meantime. These limits, while clearly not absolute, are helpful to keep in mind when deciding on the amount feedback a system should provide.

The user interface guidelines presented in this chapter have all in common that I find them both intuitive and self-evident. Yet, many interfaces, both designed by myself and others, fail to adhere to them. From this simple observation I draw three conclusions:

- ◆ Great truths are often characterized by being obvious only once they have been explained. To a degree, I believe this to be the case here.
  - ◆ General user interface guidelines should form the foundation for interface design choices.
  - ◆ User interface design is far from easy. Even if designers know how good interfaces should behave, making an interface that conforms to all guidelines is not straightforward.
-

---

Having discussed the characteristics of successful user interfaces, I now turn to how interfaces can be classified.

## **2.3 A general classification of user interfaces**

---

One of the most important concepts for human understanding of a complex world is the notion of classification. Grouping a set of items according to some common attributes allows us to treat them as one – thereby simplifying understanding and interaction. It also makes it far easier to get an overview of the features of a large population.

It is therefore not surprising that several attempts at classifying interfaces have been made. One of the most used is to group interfaces with similar *interaction styles*. Interaction styles refer to how the communication between the user and the application takes place – how actions are expressed, concepts are displayed and which metaphors are used to display system state.

The choice of which interaction style to adopt, is thus one of the most fundamental choices in interface design. Each different interaction style has its unique advantages and disadvantages. The optimal interaction style will therefore vary from application to application. I therefore find it useful to use the remainder of this chapter to examine the different alternatives. In the following discussion, I adopt the classification used by Jenny Preece et al. (Preece et al., 1994) which divides interfaces into the following categories:

- ◆ **Command entry**
- ◆ **Menus and navigation**
- ◆ **Form-fills and spreadsheets**
- ◆ **Natural language dialogue**
- ◆ **Direct manipulation**

This classification closely matches the classifications used in (Shneiderman, 1997) and (Dix et al., 1998), although the latter includes more exotic styles such as three-dimensional interfaces.

Even though most user interfaces contain a combination of at least a couple of these interaction styles, I will in the following sections discuss

---

the properties of pure implementations of each of them in order to promote clarity.

An alternative way of classifying interfaces is to group them by generation (Nielsen, 1993). User interface generations could include batch systems, line-oriented interfaces and graphical user interfaces. However, as all but the latest generations have long been abandoned, I find this classification scheme less interesting with respect to understanding how modern interfaces work.

### **2.3.1 Command entry**

Command entry interfaces are interfaces where users indicate their intent by entering textual commands, for example by typing “exit” and pressing “enter”. Command languages typically have a strictly defined syntax and thus requires both training and memorization. However, once a user has mastered the language, such interfaces allow complicated actions to be expressed swiftly and precisely.

As all input is textual, compound operations may be defined simply by including the individual commands in a text file (batch file). Textual commands also frequently offer great flexibility in that they typically offer numerous possibilities for fine-tuned control by the means of *arguments* – that is additional text specified in connection with the command. For example, the plain “dir” command in the Windows command shell simply lists the files in a directory. Using arguments, this list can be customized in a number of ways – “dir /o:s” for instances sorts the list by file size. These properties make command entry a popular

---



choice for expert users who welcome the power and flexibility and do not mind having to memorize the commands.

```
C:\>cd windows
C:\WINDOWS>attrib win.ini
A             C:\WINDOWS\win.ini
C:\WINDOWS>echo %CLASSPATH%
C:\Java
C:\WINDOWS>ver
Microsoft Windows XP [Version 5.1.2600]
C:\WINDOWS>dir *.txt /b
EnvChecklog.txt
ntbtlog.txt
OEWABlog.txt
SchedLgU.Txt
setuplog.txt
C:\WINDOWS>
```

**Figure 2.2 Command entry using Microsoft Windows XP CMD.EXE.**

Unfortunately, the amount of training necessary to master such interfaces, severely limits their use. High error rates are common, both because the strictly defined syntax leaves little room for error and because of the often non-intuitively named commands. In the classic paper “The truth about Unix: The user interface is horrid” (Norman, 1981), Donald Norman discusses the usability of command line interfaces in early versions of the UNIX operating system. Among other problems, he critiques the lack of command name consistency and the often missing relationship between command name and functionality. He also notes:

*“A common theme runs through the commands: don’t be nice to the casual user -- write the system for the dedicated expert.”*

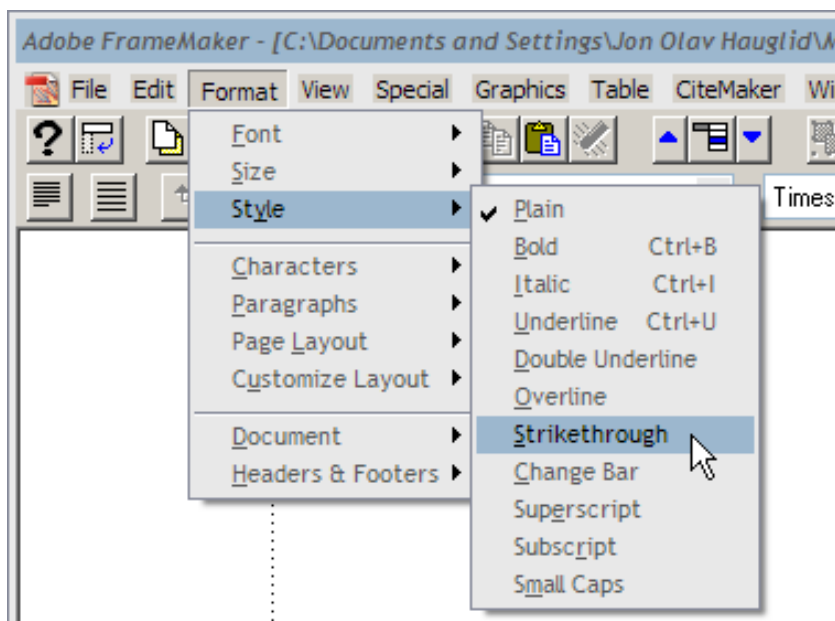
It is also frequently difficult to provide good error messages as so many different error conditions are possible due to a high number of commands and possible arguments. This is the flip-side to great flexibility – many things can be accomplished, but many things can go wrong.

### 2.3.2 Menus and navigation

A menu has been defined (Paap and Roske-Hofstrand, 1988) as:

*“[...] a set of options displayed on the screen where the selection and execution of one (or more) of the options result in a change in the state of the interface.”*

By the fact that the available choices are displayed on the screen, this interaction style lessens the need for memorization compared with command entry interfaces and thus makes them easier to learn. As all available choices are presented (and temporarily unavailable choices can be disabled), menus also make it comparatively easy to avoid many user errors. Misspelling a command name is for example not a factor.



**Figure 2.3** Drop down menus from Adobe FrameMaker.

A negative effect of displaying every alternative on screen is that menus frequently use a large part of the available display area. This problem has been greatly reduced by the introduction of pull-down and pop-up menus that hide the menu choices until they are activated by the user. Such menus have become an integral part of graphical operating systems such as the Windows and Mac OS families.

To make menus appealing to more than just novice users, the most used menu entries often have separate keyboard shortcuts. This addition, conforming to Jakob Nielsen's "Shortcuts" usability principle, makes it possible for experienced users to perform actions without leaving the current work area to activate a pull-down menu option.

### 2.3.3 Form-fills and spreadsheets

As menus only are suited for the selection of one or more predefined options, an alternative interaction style is needed for data entry. Command entry can be used, but has proved difficult to learn and use for novices and casual users.

Form-fills and spreadsheet interfaces resemble paper forms and thus represent a type of dialogue that probably already is familiar to users. A form typically contains several labelled fields where users are asked to input data. This can make form-fills rather space-consuming, as illustrated by the form-fill shown in Figure 2.4<sup>1</sup>.

Required fields are indicated with a red triangle( ▼).

---

### Account Settings

Username ▼  *between 6 and 15 characters (no spaces)*

Password ▼  *between 6 and 15 characters (no spaces)*

Confirm Password ▼  *between 6 and 15 characters (no spaces)*

---

### Member Information

First Name ▼  Middle Initial

Last Name ▼

Company ▼

Title

E-mail Address ▼

Phone ▼

Postal/ZIP Code ▼

State/Province  ▼

Country ▼  ▼

I wish to receive informational e-mails.

**Figure 2.4 Registration form for Oracle's Technology Network.**

These types of interfaces are most suited when the data to be entered consists of a multitude of individual entries. This could for instance be the registration of name and address as shown in the figure above. The clear labelling of the individual fields and the ability to restrict the input to the permissible length and classes of characters make it possible to avoid many user errors. In addition, the ability to position the individual fields so that related fields are located near each other, makes it easier to read and locate information.

---

1. Oracle Technology Network, <http://otn.oracle.com/index.html>

---

### 2.3.4 Natural language dialogue

A natural language dialogue interface is an interface where user's intent is communicated to the application by the means of complete sentences in a human language (such as English or German). This is a very natural concept – after all this is the way in which humans communicate with each other. Ideally, natural language interfaces could remove the need to learn a complex syntax and thus require little training compared to command entry. In practice however, this approach has proved difficult to implement as noted in (Shneiderman, 1997):

*“[...] high-quality reliable translations of complete documents without human intervention seem difficult to attain”. (page 293)*

One of the reasons for these difficulties, is that natural language often is ambiguous and therefore difficult for a machine to interpret. The nonverbal communication vital to human-to-human communication, is usually unavailable in human-computer interaction. Even if research effort is being spent on for example hand gesture recognition (Sato et al., 2001), such technologies is still far from mainstream. Nevertheless, natural language interfaces have enjoyed some success in applications where the scope is limited.



**Figure 2.5 Natural language interface from www.askjeeves.com.**

The characteristics of these interfaces clearly make them better suited for novices rather than experienced users. The sheer amount of text that needs to be entered even to express only moderately complex actions can make alternative interaction styles far faster to use. For example, imagine replacing the registration form shown in Figure 2.4 with a natural language interface.

Back in Section 2.2 the claim that media equal real life was briefly discussed. The implications of this statement are that people interact with

---

media as they would with real people and that rules that govern social interaction also can be applied to human-computer interaction.

One should expect that this “media equation” holds especially strong for natural language interfaces as these interfaces are designed to mimic human dialogue. But this need not be a positive effect. In my opinion, it in fact makes it easier for natural language interfaces than for other types of interfaces to appear impolite to users.

Due to the difficulties of implementing a perfect system for natural language interpretation, mistakes are bound to happen. Results of such mistakes could easily be interpreted as impolite responses if users (consciously or unconsciously) are applying the same reasoning here as in human-to-human conversations. This can in turn confuse the user and could damage their view of the interface in addition to breaking H. Paul Grice's relevance maxim (as discussed in Section 2.2).

### **2.3.5 Direct manipulation**

Direct manipulation is a term coined by Ben Shneiderman and is based on the following four principles (Shneiderman, 1983):

- ◆ Continuous representation of the object of interest.
- ◆ Physical actions or labelled button presses instead of complex syntax.
- ◆ Rapid, incremental, reversible operations whose impact on the object of interest is immediately visible.
- ◆ Layered or spiral approach to learning that permits usage with minimal knowledge.

In other words, direct manipulation advocates a visual representation of the world where the user manipulates “real objects”. For example, the command entry method for moving a window would be for the user to type in a command. Using a direct manipulation interface, on the other hand, the user would “grab” the window using a mouse or a similar pointing device and physically move the mouse the corresponding distance.

For this approach to work, responsitivity is a key factor – if the window movement lags behind the mouse movement with several seconds, continuous representation is no longer achieved and the illusion of direct manipulation is destroyed. It is therefore reasonable to expect delays

---

shorter than 0,1 seconds (referring to the previous discussion in Section 2.2).

Direct manipulation has proved (Ziegler and Fahrnich, 1988) to be a very natural method for interaction – both easy to learn and with significantly lower error rates than other interaction styles. In fact, it is a key element of almost all graphical interfaces. The advent of touch screen devices such as Personal Data Assistants (PDAs) has brought direct manipulation to the next level by allowing direct interaction with interface components (by touching the on-screen representation directly) instead of using a mouse as an intermediate.



**Figure 2.6 Equipping a character in Divine Divinity using direct manipulation.**

The accessibility of direct manipulation has made it a natural choice for computer games. As continuous representation of objects is a vital component of direct manipulation, it is difficult to capture the concept with a still image. Figure 2.6<sup>1</sup> is an attempt at an illustration – here the

---

user is in the process of equipping a shield on a representation of the user's in-game avatar.

The problems with direct manipulation mostly relate to implementing it properly. First of all it can be difficult to find a suitable graphical representation of the concepts contained in the system in question. Abstract terms and concepts prove especially difficult. Even when a proper representation has been found, these graphical interfaces often prove difficult to implement mostly due to their graphical nature and the demand for responsiveness.

### **2.3.6 Concluding remarks**

As the above discussion indicates, the interaction styles have different advantages and varying applicability. These characteristics are therefore important to keep the differences in mind when designing user interfaces. Fortunately, it is possible to combine several types of interaction styles in a single program – this is in fact more the norm than the exception.





---

## Chapter 3

# Evaluating user interfaces

---

*“Evaluation is concerned with gathering data about the usability of a design or product by a specified group of users for a particular activity within a specified environment or work context.” (Preece et al., 1994, page 602)*

In this chapter, I discuss why usability evaluation is a vital component in successful interface design and present various methods for how usability evaluations can be carried out. This serves as a methodical foundation for my own evaluations presented in later chapters.

The discussion in this chapter concerns interfaces in general, but I will illustrate the different methods with examples from evaluations of interfaces for information access.

### 3.1 Why evaluate?

---

A user interface is the communication link between a computer system and its user – allowing interaction and presenting output. As for every other type of communication, a message is best understood if it is made to suit its target. This is easier said than done as the user seldom is a constant and because human characteristics can vary widely. Marti A. Hearst has described the human differences in (Baeza-Yates and Ribeiro-Neto, 1999):

*“Important differences for information access interfaces include relative spatial ability and memory, reasoning abilities, verbal aptitude, and (potentially) personality differences [...]. Age and cultural differences*

*can contribute to acceptance or rejection of interface techniques [...].”*  
(page 261)

These differences can explain why users often will perceive or respond to an interface in ways never imagined by the designer. As a result, it is very difficult to design an interface without input from the intended target audience. By having a representable group of users evaluating the interface in question, such information can be gathered. This can in turn be used to gain insight into what needs to be changed in order to improve the usability. An interesting account on the importance of involving users in the development of interfaces to information systems, is presented in (Catarci, 2000).

Improving the usability of an interface is however not the only purpose of evaluations. (Preece et al., 1994) lists the following other reasons for performing evaluations:

◆ **Comparing designs**

When it is difficult to choose between two or more alternative designs, evaluation can provide both a direct answer and a detailed report on the highs and lows of each alternative. Because it is much easier to compare alternatives than just evaluating a single item, it might be useful to implement a second, different, design just to make comparison possible.

◆ **Understanding the real world**

Evaluation can also be of use before the actual implementation has begun. Understanding the user environment in which a program is meant to be used, makes it much easier to achieve success. This can for example include evaluating any existing programs that are meant to be replaced, discussing with future users, etc. This can also be extended to include evaluations of early prototypes to further help with the requirements gathering.

◆ **Engineering towards a target**

Every, at least moderately complex, program presents almost unlimited possibilities for improvement. Needless to say, absolute perfection is often an unattainable goal. Evaluation can play an important role here by defining metrics usable for deciding on an explicit target quality. This might for example be the total time that an average user uses to complete a certain task, or the percentage of users that prefers an application over its competitors.

---

---

**◆ Checking conformance to a standard**

Computer users typically use more than one program. By having new interfaces consistent with interfaces users are already accustomed to, the learning process becomes easier. To assure that this is the case, evaluation can be used to check if new interfaces conform with existing standards and guidelines.

Having discussed the usefulness of user interface evaluation, the next sections address how evaluations can be performed.

## **3.2 Gathering information**

---

Usability evaluation is all about gathering data about the usability of a design or a product. The results of an evaluation can be interpreted to gain information which in turn can aid the designer in improving her product or in comparing it with existing designs. In order for the evaluation to serve its purpose, it is important to decide exactly what kind of data is to be gathered.

In the social sciences, it has been customary to differentiate between *quantitative* and *qualitative* data. Much has been written about the different research practices that stem from the differences between these two – see for example (Creswell, 1994) and (Neuman, 2002). Rather than reiterating these discussions, I focus on how these two types of data relates to interface usability evaluation. For my purpose, quantitative data contains quantified data (typically numbers) typically used for statistical analysis describing how well an interface performs on average for a medium to large group of users. Qualitative data (typically text) is more focused on detailed descriptions of individual samples and uses these to describe how and why something works.

### **3.2.1 Quantitative data**

Quantitative data are typically (more or less) objective measurements of some kind. Examples include number of errors made, time spent on a given task and number of interactions required. Alternatively, they can be subjective thoughts towards a specific feature or product quantified into (for example) numbers. Asking a person to rate a feature from 1 (terrible) to 5 (excellent) is a good example.

The advantage of using quantitative data in a usability study is first and foremost that processing large amounts of results is feasible. Averages and standard deviations can provide at-a-glance information regardless

---

of the size of the collected evaluation data. Another advantage is the objectivity of measured data – although this is not as clear-cut as it initially might seem. For example, simply by selecting tasks to be measured, bias is introduced. This problem is discussed in more depth in Section 3.5.

By quantifying a test subject's sentiments or performance into a few numbers, one is bound to lose quite a bit of detail. After all, knowing that a test group on average gives a specific program three marks out of five, provides no clues as to what works and what does not. Only using averages also hides details about the individual numbers. Therefore, quantitative data is not very suited for understanding how something performs and why.

### **3.2.2 Qualitative data**

Collecting qualitative data is, in my mind, all about understanding how and why. Understanding what the user thinks, understanding why some interfaces are incomprehensible and understanding the interaction between human and computer. This understanding can be gained either by allowing test subjects to express themselves directly or by observing the actual use of a product.

Either way, gathering qualitative data is far more labour intensive than gathering quantitative data as it typically includes observing or interviewing users. As a result, much fewer test subjects are typically involved. Luckily, this does not render qualitative evaluation unusable. Studies (Nielsen, 1992) have in fact shown that as few as five users can be enough to discover a majority of usability problems. Increasing the number of testers beyond five was shown to generate ever more marginal improvements.

As qualitative evaluations concentrate on how and why an interface works (or not), they can be of great help during software development. Even before the first prototype has been implemented, a qualitative evaluation of design plans can save many hours of work. At such an early stage, evaluation of similar interface can also provide invaluable insight into which features to include and which features to avoid. Later on in the development process, qualitative evaluation can make it easier to spot potential usability problems before they become too costly to correct. As it is very easy for designers to become “blind” to other ways of thinking about their interfaces, outside designers or users might be necessary to spot even pretty obvious mistakes.

---

---

It is important to note that the choice between qualitative and quantitative data gathering is not necessarily either-or. In fact, most actual interface studies contain elements of both. However, as quantitative methods generate more reliable results with large test groups and everything larger than small groups can be quite labour intensive for qualitative methods, using both methods at once can be somewhat problematic and might make compromises necessary.

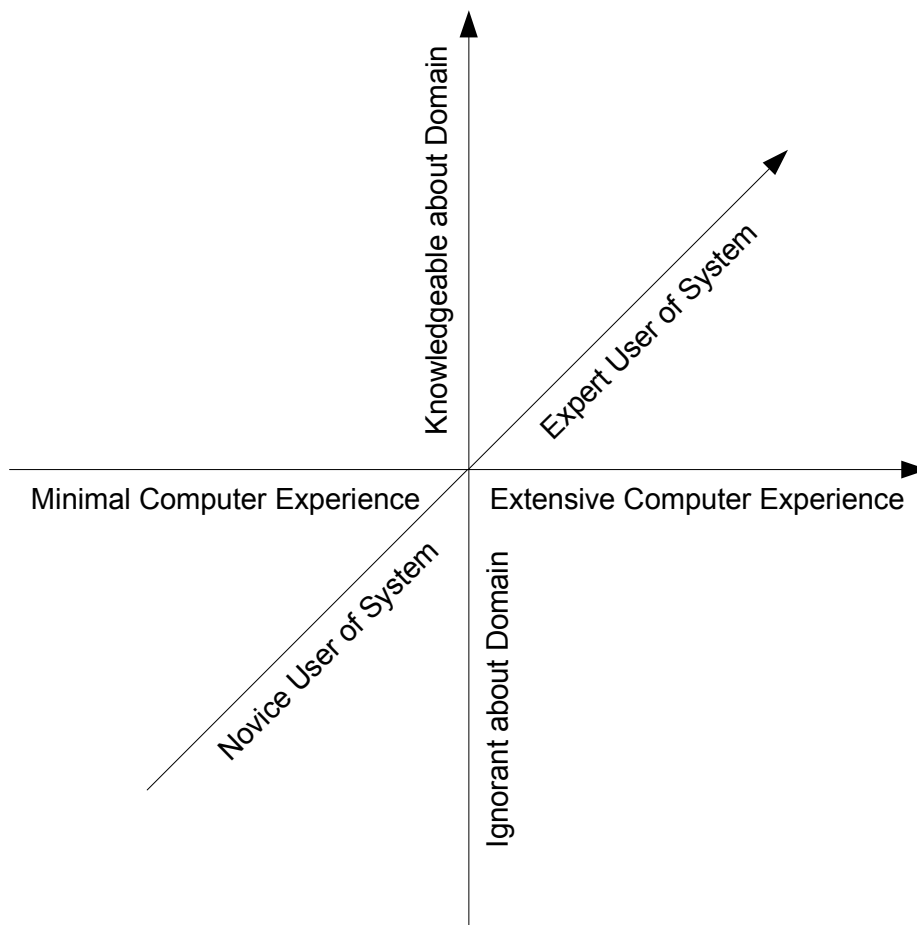
### **3.3 Selecting test subjects**

---

Social scientists have long been aware that the answer to any given question depends very much on the person being asked (Weisberg et al., 1996). This also holds true for user interface evaluation (Dumas and Redish, 1999). Selecting the right test subjects is therefore of great importance in order to get a valid result. It makes for example little sense to ask novices to evaluate a program intended for computer professionals – they are simply not a group representable for the intended audience. An exception to this rule is evaluation methods where experts use their expertise to predict usability problems regardless of target audience – more on such inspection methods later.

Test subjects can be categorized in a multitude of different ways. Jakob Nielsen (Nielsen, 1993) suggests classifying users according to three different axes: Previous computer experience, knowledge about the domain in question and experience with the tested interface. This is illustrated in Figure 3.1. Establishing where a given test subject is located in this three-dimensional space is usually done by means of background knowledge questionnaires.

---



**Figure 3.1 Three dimensional classification of users (Nielsen, 1993).**

P. Reisner (Reisner, 1981) takes a different approach by classifying tests rather than test subjects, according to the amount of exposure they have had to the system in question. Some examples include:

◆ **Immediate comprehension tests**

Which kinds of mistakes are first time users likely to make? Are the features present, possible to comprehend and use without specific training?

◆ **Productivity tests**

This is used to measure the productivity of users who have been using a given system for some time – i.e. when the training phase is well over.

◆ **Retention tests**

Subjects are first trained in the use of a program. Their comprehension is then tested after a period away from the system. This

---

measures how easy an interface is to remember and how fast re-familiarization occurs.

For other ways of categorizing test subjects, see (Cuff, 1980), (Catarci et al., 1997) and (Fisher, 1991). These include looking at factors such as age, gender and cultural background and how these might impact users' performance.

For my own use, I've decided to classify test subjects into only three different classes. This allows me to keep the discussion fairly straightforward without losing too much detail. These classes can of course also be used to describe the intended target audience of a program. The three classes are:

◆ **Expert users**

Expert users are users with an in-depth general computer knowledge – typically formal computer science education. Having used a multitude of different interfaces, they can often spot potential usability problems quickly. What they might lack in domain knowledge, they can make up for by drawing upon previous experience.

◆ **Casual users**

Compared with expert users, casual users are above all characterized by a lesser degree of general computer experience. This class represents users that are not computer professionals, but might still use the application on a day- or weekly basis. Test subjects should be given training in a new interface to properly simulate this class of users.

◆ **Novice users**

This class of users represents users that use a given program for the first time. Background knowledge may be limited, although some motivation to use the program should exist. Given the way computers have infused almost all levels of society, it is reasonable to assume basic computer experience. For information searching interfaces, a good example of a novice user could be a first-time visitor to a web-based system.

Having discussed both which kinds of information that can be gathered as well as the issue of choosing test subjects, I now turn to the actual methods by which evaluation can be carried out.

---

## 3.4 Methods of evaluation

---

This section presents a brief overview of six different evaluation methods serving as a methodical foundation for usability evaluations presented in later chapters. These six methods were selected on the basis of a review of existing methods, in particular as presented in (Preece et al., 1994), (Nielsen, 1993), (Shneiderman, 1997), (Dix et al., 1998) and (Preece et al., 2002).

The six evaluation methods described in this section:

- ◆ **Performance measurement**  
Having test subjects perform tasks using the given interface while collecting data on one or more metrics.
- ◆ **Questionnaires**  
Recording user responses to a set of predetermined questions.
- ◆ **Logging**  
Automatically record actions taken by the user.
- ◆ **Inspection**  
Experts examine design or implementation and predict usability problems.
- ◆ **Interviews**  
Recording user responses to questions asked by an interviewer.
- ◆ **Observation**  
Test subjects are observed (or recorded on video and/or audio) while performing tasks using the given interface.

Other evaluation methods, including interpretive evaluation (Preece et al., 1994), acceptance tests (Shneiderman, 1997) and focus groups (Nielsen, 1993), were either closely related to the methods I have covered or not too relevant given my purpose and constraints (available resources, domain, etc.).

For evaluation methods which requires test subjects, an important choice is deciding whether to use between-subject or within-subject design (Mitchell and Jolley, 2001). With between-subjects, any given user only tests a single interface. In effect, this means that the test subjects are partitioned into groups with one group per interface. This can be prob-

---



lematic if two or more groups are very different, for example one consisting of mainly negative individuals while another has mainly positive. Obviously, this might skew the results.

This problem is solved with within-subjects as each interface is tested by all test subjects. However, one must take care to avoid being subject to order effects – performance results dependent on the order in which the interfaces are tested. For example, if time to complete a specific operation is measured for several interfaces, one would expect the results to improve as the test goes on simply due to training – regardless of the actual usability of the tested interfaces. Such and similar order effects can be reduced by taking care when deciding the ordering.

One alternative is randomized within-subjects design where the order is randomized for each participant. This also removes any ordering bias on part of the test giver. Unfortunately, it is still possible to get ordering effects if one particular order is drawn more often. Counterbalanced within-subjects design (Mitchell and Jolley, 2001), on the other hand, makes sure to cancel order effects completely by assigning all subjects (randomly) to equal-sized groups. Members of each group test the interfaces in a particular order such that on the whole, each interface is tested first, second, etc. an equal number of times.

With this in mind, I now turn to a presentation of different evaluation methods, including references to actual evaluations of systems for accessing information where they have been used.

### 3.4.1 Performance measurement

In this method, test subjects are instructed to perform specific tasks for which one or more *metrics* are measured. A large number of such metrics can be imagined – some applicable for interfaces in general and some specific for information searching systems. Examples of the former include time used, errors made and number of interactions, while average *precision*<sup>1</sup> and average *recall*<sup>2</sup> are examples of the latter. To use this method, it is common first to define a hypothesis that one wishes to examine and then select test subjects and tasks to match.

- 
1. Precision: “[...] fraction of the relevant documents [...] which has been retrieved.” (Baeza-Yates and Ribeiro-Neto, 1999)
  2. Recall: “[...] fraction of the retrieved documents [...] which is relevant.” (Baeza-Yates and Ribeiro-Neto, 1999)
-

The quantitative nature of the results of this method means that it is well suited for comparison purposes. This has made it a popular choice for research publications. However, as some metrics can be ill-suited for automatic measurement, the method can be quite labour-intensive.

While performance measurement can be performed with test subjects of all categories, it is in my opinion not too well suited for novices. A completely untrained user will after a short time reach a certain level of familiarity with the subject. This makes it difficult to compare the results from the first part of the session with the last part – especially a problem with evaluations which compare several interfaces. A brief training session will therefore improve the reliability of the results.

A good example of the use of performance measurement was presented in (Yen and Scamell, 1993). This study presented an experimental comparison of SQL<sup>1</sup> and QBE<sup>2</sup>. In total, 65 test subjects were used – all of them students taking a database course. Each of the subjects were given training over a period of four weeks. As is discussed later in Section 3.5.1, this comparatively high number of subject and the long training period, both improve the reliability of the results. The actual test consisted of having the subjects construct queries both on paper and on-line. Both the correctness of the queries and the time used to code and debug them was measured. This allowed for a detailed statistical analysis of a number of different tasks. On this basis, they concluded that QBE was in general easier and faster to use than SQL.

### **3.4.2 Questionnaires**

Questionnaires are most often used to measure the subjective reactions of test subjects after exposure to the system in question. In addition, questionnaires also include questions aimed at establishing the background knowledge of the test subjects. An example of a much-used questionnaire for interface evaluation, is QUIS: The Questionnaire for User Interaction Satisfaction (Chin et al., 1988).

Typically, the individual questions are closed – i.e. ask the user to select from a set of predefined responses. An example of a closed evaluation question is shown in Figure 3.2. Alternatively, questions can be open – that is allowing the user to compose a free-form response. This will result in qualitative rather than quantitative response and make the questionnaire unsuited for large test populations due to the manual work needed

- 
1. Structured Query Language – See Section 4.2.1.
  2. Query By Example – See Section 4.2.2.
-

to summarize the results. In practice, most questionnaires I have found have used closed questions.

3.1 Overall reactions to the system:	terrible	wonderful	NA
	1	2	3
	4	5	6
	7	8	9

**Figure 3.2 Example of a closed evaluation question from QUIIS.**

At least for closed questions, questionnaires make it possible to measure the reactions of comparatively large groups. With smaller groups, other methods might give the designer more detailed feedback. On typical example is when experts are used as test subjects. Still, questionnaires are a reasonably universal method – generally suitable for all types of test subjects.

(Hibino and Rundensteiner, 1997) is an example of an evaluation of interfaces for accessing information where questionnaires are used. The test was a comparative analysis of two interfaces using 20 test subjects. A questionnaire (a subset of QUIIS) was used to record subjective reactions after each test subject had completed a number of tasks. The questions were organized in five groups and the paper present average ratings for each of these groups (overall reaction, learning, screen, terminology and system capabilities) as well as a total average. Further, a post-hoc Scheffé analysis (Scheffé, 1959) was used to detect if the differences in user satisfaction can be seen as significant.

### 3.4.3 Logging

In this method, the software being tested automatically records (logs) actions taken by the user. These recorded actions can then later be analysed by the interface designers to get an indication as to how the application was used – i.e. which options were used or left unused, the timing between actions etc. (Helms et al., 2000).

One way of using logging, is to record only a few aspects of the interaction – for example the frequency and types of error messages displayed. This could help the designer to identify potential usability problems by pointing out the dominant error situations. Alternatively, one can record complete logs, recording everything the user does – every keystroke and every mouse action. This way it becomes possible to completely recreate a session. Unfortunately, the amount of information gathered using complete logging prohibits large scale testing as log analysis becomes very time consuming.

The fact that this method is highly automated means that it can be applied to a large number of users effortlessly. However, it raises some difficult issues regarding privacy – especially if data is gathered without the consent of the users. Also, knowing interaction is being logged will probably affect the performance of the test subjects. This problem is on the other hand shared with most other methods of evaluation.

By its virtue as a quantitative method suited for automation, logging can be used for larger groups. As for questionnaires, more qualitative methods might work better for smaller groups. Also, logging is not really suited for experts. The whole idea is that logging allows you to study large groups without having individuals expressing their feelings and thoughts toward a system. With experts, feelings and thoughts are exactly what you want and what experts are (at least usually) good at expressing.

### **3.4.4 Inspection**

User interface experts can be of great help in the development of new interfaces. Inspections or expert reviews are the most common evaluation methods used to utilize this source of knowledge. By inspecting a prototype, an experienced professional can predict usability problems that will affect all classes of users. Heuristic evaluation (Nielsen and Molich, 1990) is a common inspection technique in which the evaluator checks how the given interface agrees with established user interface heuristics. Many such heuristics and guidelines have been presented earlier in Section 2.2.

Other methods for expert evaluation include walk-throughs and conformance checks. Using walk-throughs, the evaluator performs typical tasks and records any usability problems on the way. Conformance checks, on the other hand, is used to verify if an interface conforms to a given standard or to check for internal inconsistencies.

As experts are a limited resource, inspection methods are typically performed with few test subjects, often colleagues. They therefore require few resources and are fast to use. These characteristics make them especially useful during development as they can be performed at the same time as fundamental choices are made. Inspection methods are also useful to verify other evaluation procedures before they are implemented. Such pilot studies can help detect potential methodical problems and thus improve the validity and reliability of the main evaluation.

---

---

Even though experts are experts for a reason, it may be dangerous to rely on them solely for evaluation. As discussed earlier, the characteristics of user groups can vary greatly and it can be difficult even for an expert to predict how a given group is likely to react. Another negative factor is that professional pride might make interface designers less likely to listen to advice given by their peers. Some persons tend to take negative feedback personally and easily turn defensive. This may not only limit the number of advices taken, but also the number of advices given. In my opinion, inspections should therefore be used as a supplement to other methods.

(Plaisant et al., 1999) describes a user evaluation in which inspection played a important role. In this study, expert user review was used to get feedback as to the usability of a prototype before a larger evaluation was conducted. The prototype in question was a query preview (see Section 4.2.6) interface to search in data from NASA's Earth Observing Systems – Data Information Systems project (EOSDIS). For the expert review, 12 NASA earth scientists was asked to use the prototype. Each review consisted of a group of two or three evaluators and one observer and lasted for about 1,5 hours. No training was given. As is typical for inspections, the evaluators did not only give feedback as to their own feelings toward the interface, they were also able to use their experience to predict how other users would react (“Others remarked that some users would ...”). As a result of this evaluation, (Plaisant et al., 1999) states that at least one important change was made to the prototype.

### **3.4.5 Interviews**

Interviews are fairly similar to questionnaires in that both are most often used to gather the opinions of test subjects after they have used the program. The difference is, of course, that interviews are direct conversations between subject and interviewer rather than communication using textual questionnaires as an intermediary.

This makes interviews a highly dynamic form of information gathering as the interviewer can change the course of the interview as he or she sees fit. By adjusting the questions dynamically, the interviewer can remove any ambiguities and follow up on especially interesting answers. This is impossible with questionnaires where the questions are predetermined.

The flip side is that interviews are far more time consuming than questionnaires. While answering questions on paper can be done by the test subjects on their own, interviews require the interviewer to spend time

---

with each subject. As the results are qualitative rather than quantitative, manual analysis is also required. Also, interviews is vulnerable to bias. The interviewer can (unintentionally) influence the subject, either directly with wording of questions and comments or indirectly using face expressions, general posture, etc.

As for suitable test subjects, interviews are a reasonable universal method. Novices might however have problems expressing themselves in a manner that is easily understood as they lack the reference frame more experienced users have. Thus, observation might be more ideal here. Also, as far as experts are concerned, the difference between interview and inspection can often be vague.

### **3.4.6 Observation**

The aim of this method is to gain insight into how users interact with an existing product or prototype by examining its actual use. Usually this is done by so-called direct observation where the test subject is watched by an observer or a camera. This can either take place in a controlled laboratory environment or more informally in the users' own setting.

Pure observation can be improved upon by asking the test subject to describe what she is thinking while the test is being performed. This method, called *think aloud* (Nielsen, 1993), (Preece et al., 1994), allows the observer not only to see what the user is doing, but also to understand the motivation. Interestingly enough, this method also has its use when the user stops describing her thoughts. This typically indicates that the user is using her whole mental capacity to cope with the interface – nothing remains to keep the dialogue going. Thus, a mental breakdown – a passage requiring too much mental processing has been found. This is typically a good indication of an aspect of the user interface that needs to be improved.

As human interpretation is required when this method is used, it is quite labour intensive and thus not suited for large groups. Another weakness is that the results are very much dependent on the subjective judgments done by the observer – and therefore open for bias. Observation is however great for gaining an understanding of how a user interface is perceived and used.

I find this method to be applicable to users of all categories, but especially suited for novice users. For many applications, the initial response of users is of great importance for its success – this is particularly true for

---

web-based applications where it is so easy to reach competitors. Observing the initial actions of a new user on a new system can therefore support the development of more successful interfaces.

Observation is a very popular method for usability evaluation, probably due to the value of qualitative feedback early in a development process and the ease in which this method can be combined with other evaluation methods.

A typical example of the use of observation is described in (Hearst and Pedersen, 1996). The presented evaluation was of a single interface based on the Scatter/Gather paradigm. The actual study consisted of four graduate students executing 13 queries in an otherwise empty room while being recorded by video camera. After a 10 minute demonstration and a 10 minute warm-up phase, the subject was instructed to find as many documents as possible in 30 minutes that matched the given query. The resulting video recordings appear to largely have been used to clarify results gained from the concurrent performance evaluation.

### 3.4.7 Concluding remarks

While several of the six evaluation methods described above can be employed for a variety of uses, I find it helpful to conclude this presentation by looking at how they relate to each other. One way of doing this is to classify them according to their most suited type of test subjects (see Section 3.3) and the type of data given (see Section 3.2). My classification is shown in Table 3.1.

	Quantitative data	Qualitative data
Expert users	Performance measurement / Questionnaires	Inspection
Casual users		Interviews
Novice users	Logging / Questionnaires	Observation

**Table 3.1 Methods of evaluation classified according to available test subjects and type of data gathered.**

It should be emphasized that this classification is by all means not clear-cut – it only represents what I find to best fit each method.

## 3.5 Potential pitfalls

---

As with all other experiments or observations, user evaluations have several potential pitfalls that might invalidate results. To improve the quality of evaluations, it is therefore useful to have an understanding of these pitfalls in advance. Typically, the causes for inaccurate or invalid results are spilt into two groups, *reliability concerns* and *validity concerns* (Nielsen, 1993).

### 3.5.1 Reliability

Reliability describes the level of certainty in a set of results. If the result of an evaluation is reliable, it is very likely that repeating the evaluation would give very similar results.

One of the major sources for unreliability, is the use of humans as test subjects. As no two humans are alike, having a small group representing the population as a whole is bound to introduce an element of inaccuracy. The selected test subjects might for example be faster, brighter or more positive than the average user and thus paint an erroneous picture of the application evaluated.

There are several ways to reduce the extent of this problem. Increasing the number of subjects is a simple solution, but the increased cost might be prohibitive. Alternatively, one can screen prospective test subjects using a background survey to improve the chances of having a representative test group. Choosing evaluation methods such as comparative evaluation of similar products, also helps as these methods are not that influenced by the overall attitude of the test subjects.

For quantitative evaluations, statistic computations can assist in asserting the level of reliability of the results. Typical quantities used are standard deviation and confidence intervals. Both are of course impossible to use for qualitative data. Reliability is therefore typically of even greater concern for methods such as observation where it is difficult to get even an indication as to the degree in which the data is reliable.

An added factor for observations, is the role of the observator. As the observator is human, observations are subjective descriptions. Thus, several observators might have different observations of the same event. Thus, the use of several observators can make intersubject comparisons difficult.

---



---

The reliability problems stemming from using humans as evaluators and test subjects, are typically the introduction of random error (Mitchell and Jolley, 2001). But, reliability problems can also be caused by more systematic distortions of the results. This might for example be an observer being predisposed to finding what he or she expects to find. Another source for trouble is the Hawthorne effect (Mayo, 1933). The Hawthorne effect states that simply showing concern for users' situation improves their performance. Thus the fact that users know that they are being observed could in itself limit the reliability of the findings.

### **3.5.2 Validity**

While reliability addresses the degree in which a test result can be trusted, validity is looking at if what was tested was relevant. For example, a performance measurements results of the use of a query system might be perfectly reliable and reproducible. However, this will not help much if the group consisted of expert users and the target audience for the query system was novices. Designing an evaluation suitable for the intended user environment is therefore necessary. This includes both selection of methods and test subjects.

Most evaluation methods include a part where users perform given tasks using the interfaces to be evaluated. Because these tasks have to be selected by someone, it is quite possible to introduce bias. For example, long tasks that might be representable for natural use of a product, are usually excluded because of time constraints. Other biases include only selecting tasks that the evaluator deems “interesting” or selecting tasks based on limited domain knowledge. Such biases can hurt the validity of the evaluation by introducing a mismatch between the objective for the evaluation and what is actually evaluated.

In (Cordes, 2001) R. E. Cordes argues that perhaps the most important bias is only selecting tasks that the product supports. It is of great concern to end-users if a given operation is supported by a product or not, and this aspect is usually missing from evaluations. By only assigning tasks that can be performed, the degree of feature-completeness is not evaluated at all. Cordes suggests that the usual product-supported tasks should be supplemented with user-defined tasks in order to better take into account the users' requirements and expectations.

---

## **3.6 Summary**

---

The purpose of user interface design is to improve the interaction between human and computer. As no two humans are identical, no universal and objective measurements of interface quality exist. Usability evaluation therefore serves an important role in making it feasible to discuss the characteristics of user interfaces. The different methods presented in this chapter thus represent necessary background knowledge when I now turn to interfaces for information searching.

---

---

## Chapter 4

# Accessing information

---

The previous two chapters have presented an introduction to user interfaces and to usability evaluation. Both these chapters have focused on user interfaces in general. In this chapter, I turn to interfaces for accessing information and present a general state-of-the-art overview of such interfaces.

I start in Section 4.1 by discussing different strategies for information access. This serves as an introduction to the main section, Section 4.2, where I present an overview of existing user interfaces for accessing information in order to examine different existing approaches and get a better understanding of the domain. On the basis of the descriptions of the various interfaces, I in Section 4.3 identify what I find to be the key challenges for designing such interfaces. Finally, Section 4.4 gives a short summary of chapters 1-4, which together form the background for the work presented in the following chapters.

### 4.1 Different strategies for information access

---

Imagine the following two information needs:

- ◆ Who was British prime minister in 1936?
- ◆ What are the interesting sport news today?

It is evident that they represent two quite different ways of seeking information. The first concerns a single, specific and objective fact, while the second is both more vague (what is sports?) and subjective (interest-

ing to whom?). Not surprisingly, different strategies might be required in order to fulfil these and other information needs. The first will probably be best satisfied by entering precise query terms while the second could perhaps better be served by browsing different information sources.

The issue of different strategies for information access is important to consider when designing an interface as each strategy may have different requirements. For example, locating an object using known property values requires an interface that lets the user express these values, while this will not be of much use to users with more vague information needs. Therefore, the designer needs to decide if a given strategy should be supported and if so, to what extent.

Different strategies should also be considered when designing usability evaluations. As not every interface is equally suited for all types of searching, several strategies should be tested in order to give a more valid and complete picture of the characteristics and performance of each tested interface. This has been addressed earlier in Section 3.5.2.

Several different ways of classifying strategies have been published. In (Baeza-Yates and Ribeiro-Neto, 1999) Marti A. Hearst describes the methods that a user interface for accessing information repositories should support:

◆ **Browsing**

Casual, undirected exploration of one or more information structures. Example: Reading a web-based newspaper and using hyperlinks to read news stories that appear interesting on the spur of the moment.

◆ **Querying**

Producing new, ad-hoc collections of information objects. Example: Querying an information collection for objects related to a specific term.

◆ **Navigating**

Following links towards a clear goal using a sequence of scan and select operations. Example: Using a hyperlinked news archive to locate a specific story.

◆ **Scanning**

Systematic examination of a single information structure. Example: Reading link titles on a map of a web site.

---

---

This classification has a lot in common with the one published by Bates (Bates, 2002). She refers to navigation as linking and to querying as directed searching, but the terms are otherwise comparable.

Shneiderman (Shneiderman, 1997), on the other hand, has a classification which focuses more on what type of information a user is looking for, rather than the strategy used to find it. He identifies two types of fact finding tasks. The first, *specific fact finding* is looking for a single, narrow piece of information that is known to exist – such as the author of the Harry Potter books. The second, *extended fact finding* concerns items that are not known to exist but where the outcome is replicable. An example could be find movie directors that have received more than two Oscar awards.

Beyond fact-finding, he defines the task of identifying the possible existence of an information item as *exploration of availability*. An example could be to try to determine if any earthquakes in Africa have been reported last year. The fourth and final task identified by Shneiderman, is called *open-ended browsing* and is similar to what is denoted as unstructured exploration in Hearst's classification.

A different classification is given by O'Day and Jeffries (O'Day and Jeffries, 1993). They interviewed 15 clients of professional intermediaries in order to discover how they gathered information. Based on analysis of these interviews, they were able to identify three different “search modes” (or strategies):

- ◆ Monitoring a topic or a set of variables over time.
- ◆ Following an information-gathering plan based on the task at hand.
- ◆ Exploring a topic in an undirected fashion.

Which ever way one chooses to classify strategies for information access, simply supporting one strategy might be too limiting. Supporting a multitude of strategies, if at all possible, will provide the user with more ways to reach his or her goal and therefore make the capabilities of the interface less of a limiting factor.

---

## **4.2 Overview of existing interfaces**

---

In the remainder of this chapter I seek to gain a deeper understanding of interfaces for accessing information by locating and discussing key design issues. As the first step towards this goal, this section contains an overview of existing interfaces. Rather to present several similar interfaces, I have chosen to focus on different approaches and have therefore organised this section on capabilities rather than concrete implementations.

For each of these categories, I present a brief general introduction before I examine a concrete implementation in some detail. Based on this description, I discuss the capabilities of the implementation and the key issues highlighted. In Section 4.3, these key issues are collated and expanded upon.

The following categories are examined:

- ◆ Textual query languages
- ◆ Query-by-example
- ◆ Forms
- ◆ Content-based multimedia queries
- ◆ Information retrieval
- ◆ Dynamic query & query preview
- ◆ Natural language queries
- ◆ Category browsing

### **4.2.1 Textual query languages**

Textual query languages are the natural extension of command entry interfaces into the domain of information access. Thus, these interfaces share the characteristics, advantages and disadvantages of command entry interfaces as presented in Section 2.3.1.

Perhaps the best example of a textual query language is Structured Query Language (SQL) (ISO/IEC, 1992). It has become a de-facto standard for both querying and updating relational databases. Supported

---

by almost every database vendor, SQL is both an ANSI and an ISO standard. For an in-depth description of the latest SQL-standard, see (Melton and Simon, 2002).

Many different ways of accessing a database using SQL exist, however in this discussion I use Oracle SQL\*Plus<sup>1</sup> as an example of a SQL front end. Figure 4.1 illustrates the SQL\*Plus user interface which consists of a text window where SQL commands can be entered interactively. The result of each command is displayed in the same window.

```

Oracle SQL*Plus
Fil Rediger Søk Valg Hjelp

SQL> desc v$instance
Navn                                Null?    Type
-----
INSTANCE_NUMBER                     NUMBER
INSTANCE_NAME                       VARCHAR2(16)
HOST_NAME                            VARCHAR2(64)
VERSION                             VARCHAR2(17)
STARTUP_TIME                         DATE
STATUS                              VARCHAR2(7)
PARALLEL                             VARCHAR2(3)
THREAD#                              NUMBER
ARCHIVER                             VARCHAR2(7)
LOG_SWITCH_WAIT                     VARCHAR2(11)
LOGINS                               VARCHAR2(10)
SHUTDOWN_PENDING                    VARCHAR2(3)
DATABASE_STATUS                     VARCHAR2(17)
INSTANCE_ROLE                       VARCHAR2(18)
ACTIVE_STATE                         VARCHAR2(9)

SQL> select instance_name, status, version from v$instance;

INSTANCE_NAME    STATUS    VERSION
-----
gigabase        OPEN     9.0.1.1.1

SQL>

```

**Figure 4.1 Oracle's SQL\*Plus.**

In order to use SQL\*Plus successfully, the user is required to learn and recall the strict SQL syntax and input textual commands. This makes it apparent that SQL\*Plus, as most command entry interfaces, is more suited for expert users rather than a novices. Users have the power to express a query of any complexity on any available attribute, but this power is only available to those willing spend the time required to master the query language.

---

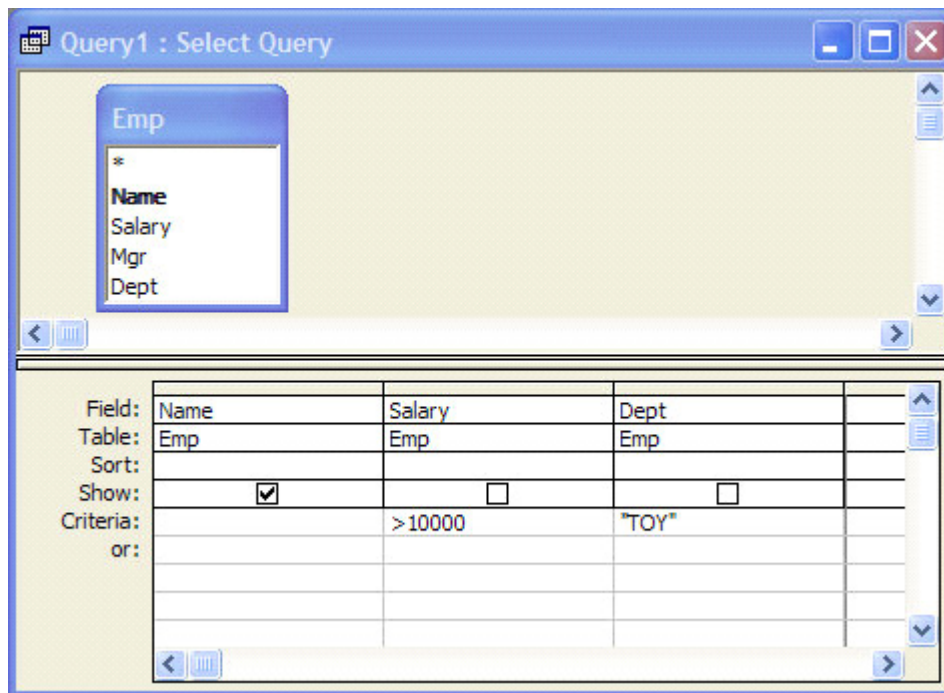
1. Oracle Corporation, <http://www.oracle.com/database/>

Even learning the query language is not enough. As SQL\*Plus provides direct access to all details in the database, learning the data model is pretty much also required in order to use SQL efficiently. The difficulty of this task will of course be highly dependent on the complexity of the data model.

Further, as SQL is based on using boolean expressions, it is pretty much impossible to use SQL for anything beyond exact queries. Browsing, expressing uncertainty, etc. is very difficult to perform.

### 4.2.2 Query-by-Example

Query-by-example (QBE) is a relational database language originally published by Moshé M. Zloof (Zloof, 1977). QBE has been implemented in several database systems – Figure 4.2 shows an example from Microsoft Access<sup>1</sup>. Similar to SQL, QBE is not only a query language, but can also be used to enter data into the database.



**Figure 4.2 Query-by-example from Microsoft Access.**

As the name suggests, QBE is designed to have database interaction be done by having the user provide examples of the solution. Actual queries are specified by filling out a two-dimensional skeleton table. Figure 4.2 shows a query for the name of all employees in the toy department which

1. Microsoft Corporation, <http://office.microsoft.com/home/default.aspx>



---

has a salary of more than 10000. This was done by selecting all relevant attributes and specifying restrictions (>10000 and “TOY”) for Salary and Dept attribute.

Although newer implementations of QBE use some direct manipulation features, form-fill is still the major interaction style (see Section 2.3.3 for general form-fill characteristics). This helps QBE to mimic manual table manipulation, but makes complex queries difficult due to the amount of screen space required.

In contrast to SQL, QBE makes it possible to construct queries without first learning a query language. This makes it easier for non-programmers to use QBE (Yen and Scamell, 1993). Thomas and Gould (Thomas and Gould, 1975) reported that less than three hours of instruction is needed for casual users to acquire the skill necessary to make “fairly complicated queries”. Three hours is however still far too long to make QBE suitable for environments where providing training is difficult (web based interfaces being the token example).

As QBE is a database language and the actual implementations can vary greatly, it is difficult to present general characteristics. However, its reliance on boolean expressions in the construction of search constraints makes anything but exact queries difficult. It also seems reasonable to expect the usability of QBE to not scale too well with the complexity of the data model. As the end users always operate directly on the data model, the complexity of the data model will be visible to the users. If the end users instead operated on some abstraction of the data model, the complexity could be masked by the interface designer.

### **4.2.3 Forms**

Forms is a very direct application of the form-fills interaction style described in Section 2.3.3. Figure 4.3 shows an example from BIBSYS – a Norwegian bibliographic database for university and college libraries.

---

The screenshot shows the BIBSYS Search Library database interface. At the top, there are navigation links: Information, Search library holdings, My BIBSYS, and Sitemap. The main header includes the BIBSYS logo, the text 'Search Library database', and a note: 'From this screen you may search the complete holdings at libraries participating in BIBSYS'. There are also links for Home, Contact, and advanced search, along with the text '- BibSøk Nett -' and a Norwegian flag icon.

The search interface is divided into several sections:

- Navigation:** Simple search, Advanced srch, Journals, Instructions.
- Buttons:** Search, Browse index, Reset.
- Search Fields:**
  - Library:** All libraries (dropdown menu)
  - Author:** (Lastname, firstname)
  - Title:** Title word (dropdown menu)
  - Free text:**
  - Subject:** All (word) (dropdown menu)
  - Class.:** DDC (dropdown menu)
  - ISBN/ISSN:**
  - Other:** Inst./conf. (word) (dropdown menu)
  - Sort by:** (No sort) (dropdown menu) with a link to 'More fields (Refine direct)'
- Instructions:** 'If you enter more than one word in Title word, Free Text or Subject, then boolean AND will be inserted. Check here if you instead choose OR:
- Buttons:** Search, Browse index, Reset.

**Figure 4.3 Forms-based query interface from www.bibsys.no.**

A query form consists of several structured fields with each field typically having a closely defined domain (for example publication title, year, subject etc.). The user fills in one or more of these fields with query expressions and clicks “Search” (or similar) to execute the query. Typically the result will be items that satisfy all the query expressions, but some forms allow the user to choose between “all of these” and “any of these”.

If the form fields are well chosen and understandable to the user, query forms make it possible to construct precise queries in an user-friendly manner without having to learn complex query languages or know the details of the data model queried. But if the meaning of the fields are not evident, or if the number of fields is very large, users are easily forced to use hit-or-miss strategies in order to find the correct field.

Compared with SQL and QBE, forms does not expose the users to the data model. This complexity of this model can therefore be hidden from the end user simply by restricting the number of fields presented or by mapping more than one property in the data model to a single query field.

### 4.2.4 Content-based multimedia queries

The meaning of multimedia data types, such as images, audio and video, is highly dependent on the viewer and is the result of a high-level cognitive process difficult to recreate in a computer. This makes multimedia repositories difficult to support for information access tools. Therefore, it is common to describe complex data types by manually attaching textual annotations which in turn can be used for accessing the data (for example by textual queries).

An alternative approach is to investigate which kinds of information that can be automatically extracted from multimedia, and subsequently design matching query systems. For images, this could for example include average colour, major shapes, etc. This method is dubbed content-based queries.

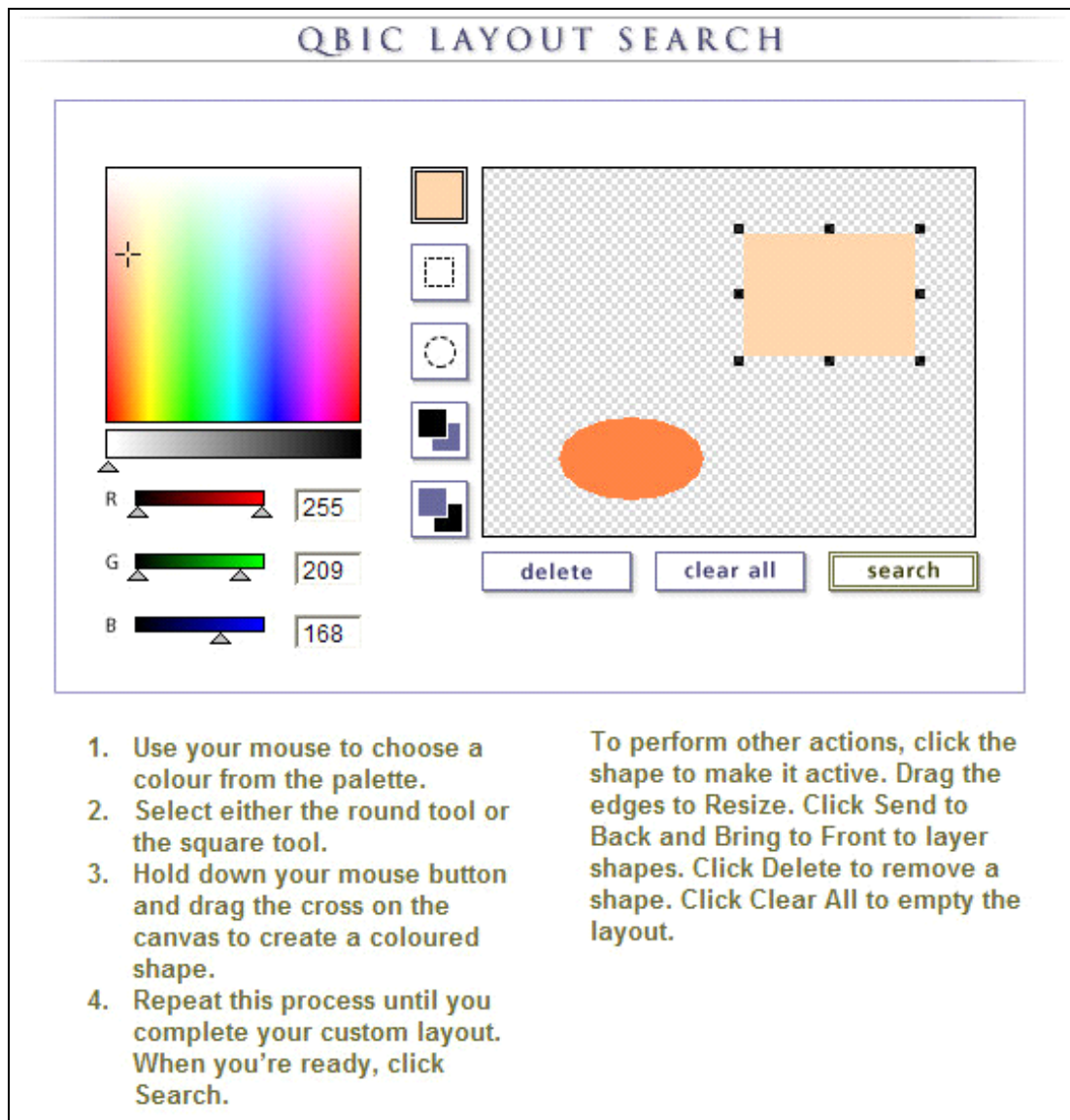
One of the most prominent content-based image and video retrieval systems, is Query by Image and Video Content (QBIC) (Flickner et al., 1995). QBIC is currently a part of IBM's DB2 Universal Database<sup>1</sup> and is in use on several web sites like the one shown in Figure 4.4.<sup>2</sup>

---

1. IBM Corporation, <http://www-306.ibm.com/software/data/db2/>

2. The State Hermitage Museum, <http://www.hermitagemuseum.org/>

---



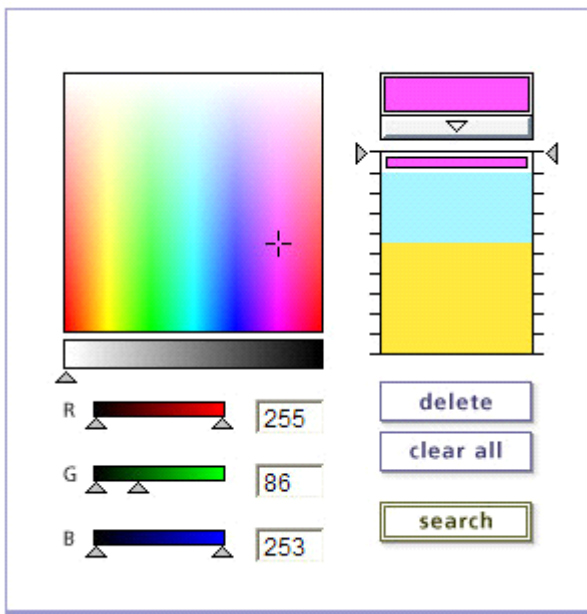
**Figure 4.4 QBIC Layout search – from The State Hermitage Museum.**

The QBIC system allows the user to generate queries based on example images, user-constructed sketches and drawings, or selected colour and texture patterns. Queries made using either of these methods are sent to an image database system. The images in this database must already have been preprocessed to extract information about features such as colour, texture and shape as such extractions are too computally expensive to be done in real time. Similar computations are made in real-time on the user-supplied query before these two sets of feature descriptors are compared to produce the query result.

In addition to images, QBIC also handles video data. This is done by first partitioning a video into shots and then extracting representative frames for each shot. These frames are handled in the same way as regular images, with the exception that object movement can be tracked in time and thus allows movement to be used in queries.

QBIC's query interface is based mostly on the direct manipulation interaction style (see Section 2.3.5). The “layout search” consists of a drawing area, a colour chooser and a drawing toolbox. Here users can draw sketches to be used for querying much in the same way as one uses drawing programs. This approach is illustrated in Figure 4.4.

An alternative interface used to specify colour queries is shown in Figure 4.5. After the colours of interest have been selected and added to the right side bar, the area of each colour band can be adjusted (by direct manipulation dragging) to indicate their respective weight.



**Figure 4.5 QBIC Colour search.**

The two interfaces described above are clearly more suited for querying than browsing. However, QBIC also offers the possibility of using a given image (for example from the previous result) as input to a new query. This feature makes browsing through an image collection easier than it otherwise might be, and makes QBIC support browsing at least to some degree. Still, this interface is focused on recall rather than recognition – if you can not describe (or in this case draw) the image you are looking for, it is difficult to be successful.

Problems associated with large results are reduced by QBIC's ability to rank query results. This is made possible by evaluating the degree in which each image matches with the query expression and sorting them accordingly.

### 4.2.5 Information Retrieval

With respect to retrieval systems, it is customary to separate *data retrieval* from *information retrieval* (Baeza-Yates and Ribeiro-Neto, 1999). While these two terms are not exactly defined, the former usually refers to retrieving data which satisfy a clearly defined condition – examples include both SQL, QBE and Forms. The latter is most often used to describe systems where information is retrieved from (unstructured) documents. As these documents typically contain natural language and thus semantically ambiguous text, it is impossible for the retrieval system to be completely accurate. It therefore has to support interpreting of documents and ranking them according to their relevance with respect to the user query. This of course has consequences with regards to the user interface – both for query specification and presentation of results.

The tremendous growth of the World Wide Web the last decade has spawned renewed interest in information retrieval. With the number of available documents on Web increasing at an alarmingly rate, on-line tools for searching the web have become more and more necessary. Google<sup>1</sup> (Brin and Page, 1998) is one of the most popular of these. Other web searching tools include AltaVista<sup>2</sup>, AllTheWeb<sup>3</sup> and Excite<sup>4</sup>. As all these tools function in a fairly similar manner and provide good example of user interfaces for information retrieval systems, Google is used as an example in this text.

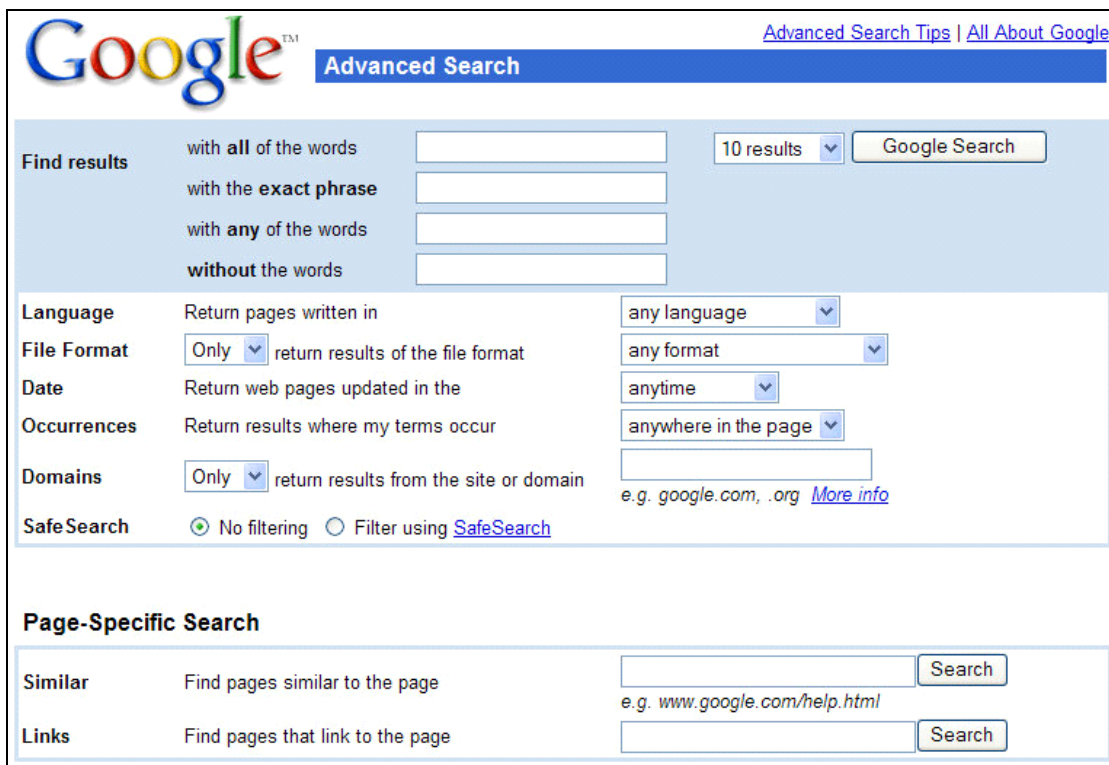
Google's main interface is shown in Figure 4.6. This simple interface consists of a text entry field for the query expression, a "Google Search"-button for starting the search and an "I'm Feeling Lucky"-button that brings the user directly to the first web page in the search result. The query expression can consist of one or several words – Google will automatically use the boolean AND operator between the individual words listed. It is also possible to perform a phrase search by putting the query expression in quotation marks.

- 
1. Google, <http://www.google.com>
  2. Overture Services Inc., <http://www.altavista.com>
  3. Overture Services Inc., <http://www.alltheweb.com>
  4. The Excite Network, <http://www.excite.com>
-



**Figure 4.6 Google.**

To also cater to the needs of more advanced users, Google provides an alternative “Advanced search” interface shown in Figure 4.7. This form-fill (see Section 2.3.3) interface both includes possibilities for fine tuning of a query and alternative functionality like similarity based search.



**Figure 4.7 Google – Advanced search.**

Figure 4.8 shows how results are presented in Google. For each hit the passage containing the query expression is extracted and shown with query words highlighted. A click on the heading will take the user to the listed web page while “Similar pages” can be used to start a similarity based query.

[Movie Review Query Engine](#)  
... Movie Review **Query** Engine. ... Film titles, wherever possible are linked to the Internet Movie **Database**, unquestionably the most useful film resource on the Web ...  
Description: Online directory of movie reviews. Contains more than 100,000 reviews about 16,000 titles.  
Category: [Arts > Movies > Reviews > Review Hubs](#)  
[www.mrqe.com/](#) - 9k - 14 Feb 2002 - [Cached](#) - [Similar pages](#)

[CDC WONDER Public Health information, reports and data](#)  
... and many other topics are available for **query**, and the requested data can be readily ... Machine Safety. AIDS School Health Education **Database**. AIDS. Air Toxics. ...  
[wonder.cdc.gov/](#) - 46k - [Cached](#) - [Similar pages](#)

[CancerNet - PDQ-NCI's Comprehensive Cancer Database](#)  
... PDQ® - NCI's Comprehensive Cancer **Database**. Table of Contents. ... PDQ Clinical Trials **Database**. PDQ ...  
[cancermet.nci.nih.gov/pdqfull.html](#) - 16k - [Cached](#) - [Similar pages](#)

**Figure 4.8 Google – Result presentation.**

In order to fit the web user environment, Google has clearly been designed for users with little prior knowledge of searching and thus offers a simple and intuitive interface. Google also functions very similarly to its competitors, making it possible to draw upon experience gained using other alternatives. Advanced options are available, but these are only available upon request so that they do not clutter the main interface.

Google conforms to the standard model of information access interactions described in (Baeza-Yates and Ribeiro-Neto, 1999). According to this model, users start by issuing a query to the system. The system then performs the search and sends the result back for evaluation. If the result does not contain the sought objects, the user can reformulate the initial query and start the query-process all over again. In latter implementations of Google, this process has been extended to include the possibility to search within results.

Although this model is familiar to most users, it has some shortcomings. For example, large result sets can pose problems as the relevant parts can be hidden among heaps of irrelevant data with little or no means to separate them. Intelligent ranking algorithms have therefore been a hot topic lately and this effort has arguably somewhat reduced the problem.

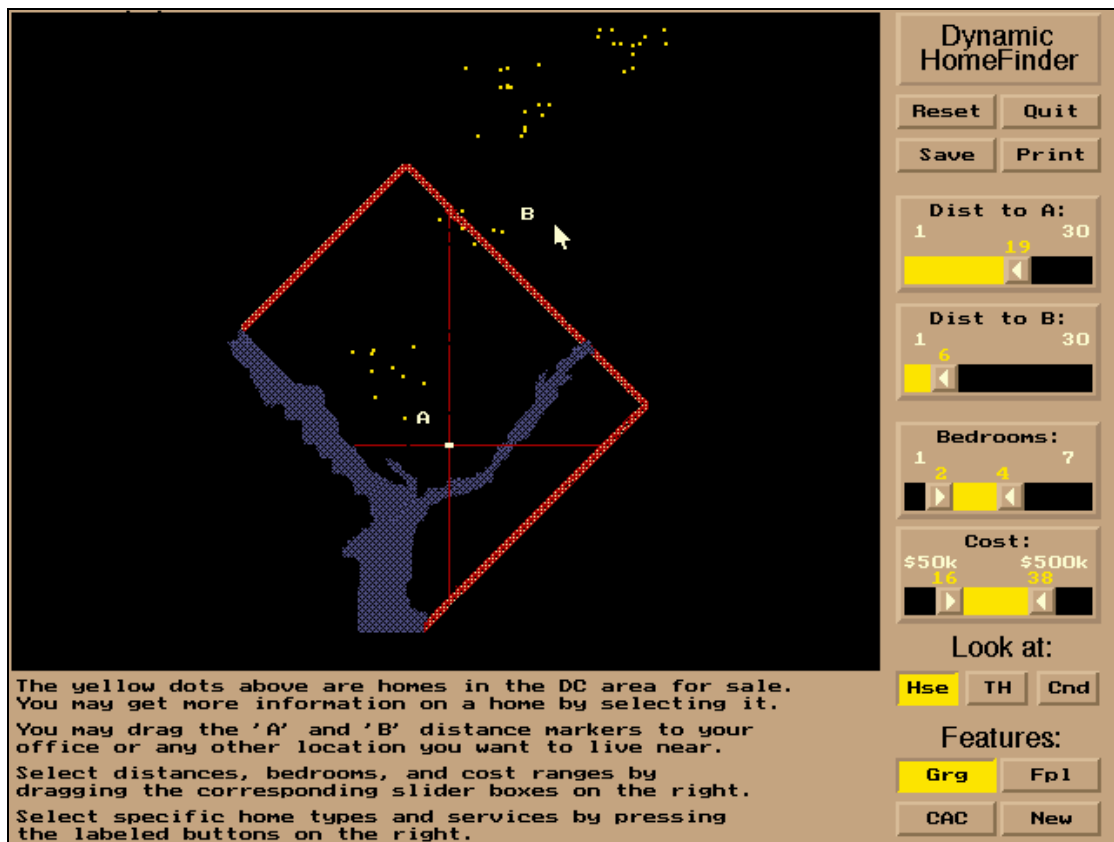


As Google relies on exact match search, relevant information can easily be missed due to misspellings or the use of synonyms. Google also does not include a word stemmer. On the other hand, the lack of features for relaxation of search constraints is understandable as they are difficult to implement when the query expression is not restricted to a single language. A related problem is the lack of expressive power. As the documents you can search using Google are unstructured, it is impossible to construct queries such as “find all documents concerning cancer written by N.N.”. This kind of information is simply not available to the query engine.

### **4.2.6 Dynamic queries and query preview**

The motivation behind the design of dynamic queries was to apply the highly successful direct manipulation paradigm (see Section 2.3.5) to database querying (Shneiderman, 1994). Before dynamic queries, most querying interfaces had been based on the command line entry method – a method which had proved error prone and difficult to learn. In contrast, dynamic queries allow users to construct queries by using graphical widgets such as sliders and buttons and see the result dynamically updated accordingly (hence the name dynamic queries).

Figure 4.9 shows the “Dynamic HomeFinder” (Williamson and Shneiderman, 1992) one of the earliest dynamic query applications. It was designed to help users locate homes based on cost, features and distance to user-specified locations. As the user adjusts these attributes using the right-hand controls, the matching available homes is highlighted on the map.



**Figure 4.9 Dynamic HomeFinder.**

Designed as a direct manipulation interface, dynamic queries exhibit features such as continuous visual representation of objects, rapid and reversible operations and immediate feedback. Studies (Ahlberg et al., 1992) have shown significant improvements in user performance compared to form-fill interfaces.

One of the strong points of the dynamic query interface, is that it handles complex data models well. This is due to the fact that the interface presents an simplified abstraction of the data model rather than the gritty details of the model itself. As this makes it unnecessary for users to learn the data model, the challenge is rather for the interface designer to make the proper abstraction. This feature is ideal for casual users, making dynamic queries fitting for web-based applications.

The downside of working on an abstraction provided by the designer, is that not all imaginable possibilities for operating on the result will be available. Specifying other restrictions than what there are provided interface controls for in Figure 4.9, is not possible. Thus, some expressive power is typically lost to gain increased usability.

Dynamic queries are also reasonably well suited for handling large result sets. At interface level, the ability continuously to monitor the result while the query specification is adjusted, makes it possible to evaluate if the size of the result is low enough to make manual browsing and scanning feasible. It is however not given that any query always can be adjusted so that the result set becomes small enough – especially since the attributes used for restrictions are predetermined and not subject to a dynamic analysis. Thus, users might at some level be forced to resort to manual browsing before they really want.

In order for the users to experience the updating of result as truly dynamic, it is important that a certain level of interface responsiveness can be achieved. In broad terms, a response time of less than 100 ms is desirable (see Section 2.2 for a discussion of responsiveness).

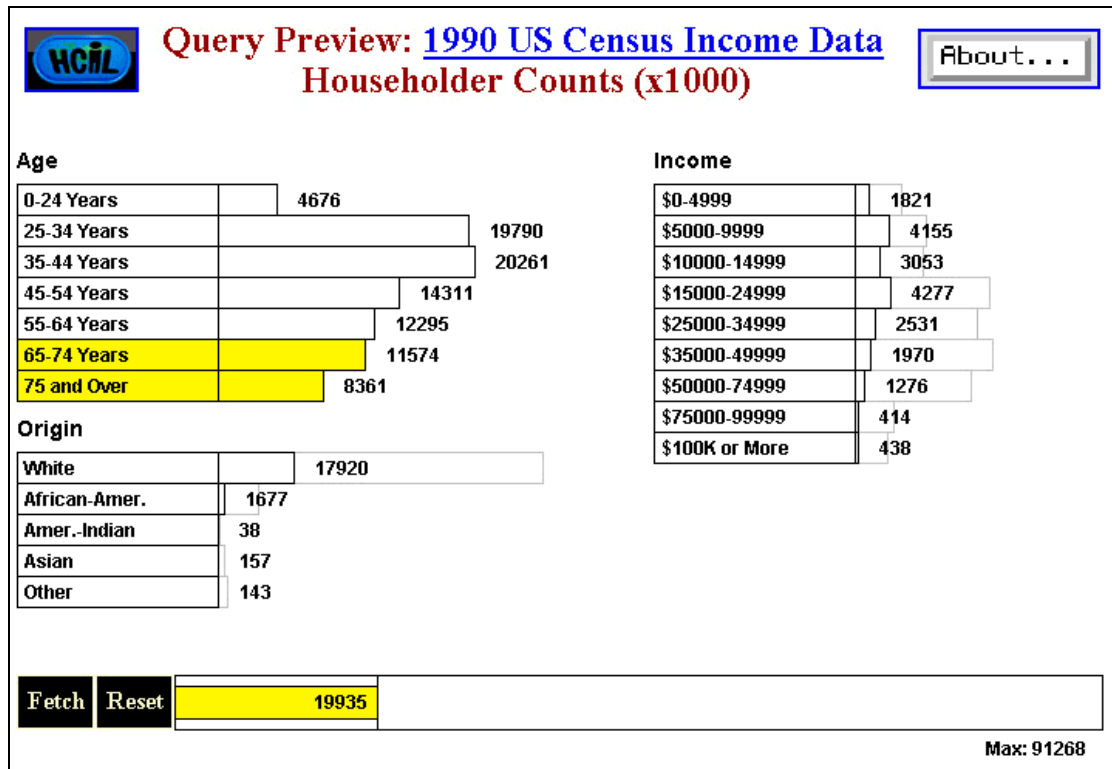
The first implementations of dynamic queries proved to have difficulties with handling of large result sets simply because such short response times could not be achieved. This problem was alleviated to some extent by the development of specialized data structures and algorithms (Tanin et al., 1997), but performance still remained an issue. This led to the development of query previews.

The concept of query previews was originally introduced in (Doan et al., 1996) and extended in (Plaisant et al., 1999). Query preview is the first phase in a refined two-phase approach to dynamic queries. This first phase allows users to formulate an initial query by selecting desired property values while simultaneously being shown the volume of matching data. This is made possible by pregenerating a volume preview table that indicates the number of data sets for each property value and intersections. Thus, the first phase is really not that dynamic as the display is based on pregenerated data. This phase is illustrated in Figure 4.10<sup>1</sup>. In the second phase, query refinement, actual metadata is collected from the database to be used in a dynamic query interface (as described above). Using this two-phase approach, the performance of dynamic queries was greatly improved.

---

1. <http://www.cs.umd.edu/hcil/eosdis/>

---

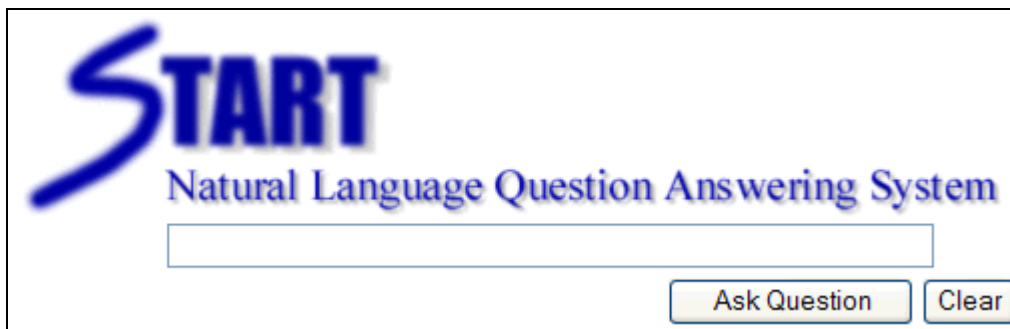


**Figure 4.10 An example of Query Preview.**

Being an extension of the original dynamic query concept, query previews naturally also use the direct manipulation interaction style. In fact, the whole point with the increased performance offered by query previews was better to conform to the rapid feedback criteria. The use of static pregenerated data, while going a long way towards solving performance problems, is less suited for very dynamic data repositories.

### 4.2.7 Natural language queries

The START Natural Language System (Katz, 1997) serves as an example of a query system based on natural language. It is a World Wide Web-based answering system designed at the MIT AI Laboratory and was first published in December 1993. As noted in Section 2.3.4, natural language dialogue is very difficult to do in the general domain. In its current version START is therefore limited to answering questions regarding the MIT AI Labs, geography, arts and entertainment, science and nature, history and culture, and reference information.



**Figure 4.11 The START Natural Language System.**

START's interface for entering queries is shown in Figure 4.11<sup>1</sup> while an example of the result of a query is shown in Figure 4.12. As the latter figure illustrates, START uses external sources to complement its knowledge base. This technique, coined virtual collaboration, involves analyzing the content of external web sites that are deemed relevant and incorporating the central knowledge base with links to this new information. In (Katz, 1997) START's author proposes that web publishers should annotate their own material (including multimedia data types) so that natural language systems such as START can index them. In this way it could be possible to have a system that acts as “a smart reference librarian for the World Wide Web”.

---

1. <http://www.ai.mit.edu/projects/infolab/>

---

## START's reply

---

====> Which country in Europe is smallest?

Vatican City has the smallest area among countries in Europe.

### Vatican City



**Area:** *total:* 0.44 sq km

*land:* 0.44 sq km

*water:* 0 sq km

**Source:** [The World Factbook 2001](#)

**Figure 4.12** Example of a result from a START query.

The prime reason for the development of natural language interfaces is that these interfaces should feel more natural to users and thus easier to operate. As discussed earlier in Section 2.3.4, this does not always work as intended. Ambiguity can be a problem in human-to-human conversation and even more so in human-computer interaction, in part due to the lack of any means for non-verbal communication. Another problem is the reliance on recall rather than recognition – users must find the appropriate way of wording themselves in order to use the system. It is impossible to find something that you can not accurately express.

An integral component of natural language systems is of course the interpretation of the entered query. As this phase includes relaxation features such as word stemming and examination of synonyms, they can avoid many of the problems associated with exact-matching boolean queries. Further, as START incorporates links to external databases of knowledge, it can be used as a hub for web browsing – not only as a straight question-answer application.

---

### 4.2.8 Category browsing

Yahoo! is perhaps the most popular web site directory with over 1.62 billion page views per day (numbers from March 2002) (Yahoo! Inc., 2004). Originally started by two Ph. D. students in 1994 as a way to keep track of their personal interests on the Internet, Yahoo! now contains thousands of web site links in any number of categories, as well as a large number of other services.

Of special interest in this discussion is the Yahoo! Directory section. It contains thousands of web site links divided into a number of inter-linked, hierarchical categories. An example of a category, “Alternative Fuel Vehicles”, is shown in Figure 4.13. Each category contains a number of links sorted alphabetically (not shown in the figure) as well as a brief “Most Popular” list on top. In cooperation with the category searching feature (top right), this lessens the need for manual intra-category searching. The interface also contains links to more general categories (“Automotive” and “Recreation”) and more specific categories (“Biodiesel”, “Organizations” etc.) to make inter-category navigation easy.

---

**Yahoo! Directory**  
Automotive > Alternative Fuel Vehicles

Search [Advanced Search](#)  
[Help](#)

all of Yahoo!  just this category

[Home](#) > [Recreation](#) > [Automotive](#) > Alternative Fuel Vehicles

---

**Inside Yahoo!**

- [Yahoo! Autos](#) - everything you need to buy a car.
- [Start or Join a Group](#)
- [Shop Online](#)

---

**Categories**

- [Biodiesel](#) (17)
- [Business to Business@](#)
- [Electric](#) (43) **NEW!**
- [Hybrid Electric](#) (9)
- [Magazines](#) (2)
- [Organizations](#) (14)
- [Shopping and Services@](#)
- [Solar](#) (35)
- [Steam](#) (11)

---

**Site Listings**

Most Popular

- [Office of Transportation Technologies - U.S. Department of Energy](#) - learn about the latest technologies and about the department's involvement in the development of advanced transportation.
- [Alternative Fuels Data Center](#) - providing accurate, objective information and statistics about alternative fuels and alternative fuel vehicles to interested organizations.
- [Argonne National Laboratory - Transportation Technology R&D Center](#) - developing cost-effective solutions for problems such as vehicle emissions and energy supply.
- [Natural Gas Vehicle Coalition](#) - a national trade association representing natural gas companies, equipment manufacturers, service providers and government entities interested in promotion of natural gas vehicles.

**More Yahoo!**

[Ask Yahoo!: Automotive](#)

---

[Y! Autos: Consumer Reports](#)  
learn before you buy

---

[News: Automotive](#)

---

[Y! Travel: Rental Car Search](#)  
make reservations

**Figure 4.13 Yahoo! Directory.**

The interaction style used in Yahoo! is mostly menus and navigation (see Section 2.3.2). The use of hypertext to link categories and web sites comes natural as this is what is expected on the World Wide Web.

Yahoo! allows for several different information access strategies. As Section 4.13 illustrates, both browsing and query components are present on any given page and therefore highly intertwined. This makes it possible to change the information searching strategy dynamically.

Yahoo's site directory is clearly designed with casual users in mind. Browsing categories and subcategories are often easier than using a search engine as you must not come up with a suitable search expression yourself. This is, in my opinion, one of the primary reasons for the success of Yahoo!.



However, the use of predetermined categories also has its drawbacks. For example, Yahoo! is very much dependent on users finding Yahoo's way of categorizing web sites natural and intuitive. Even though an advanced user can customize their view of the content of Yahoo! to some extent, this is beyond the reach of casual users. They must accept and adapt to Yahoo!'s classification to get the most out of the category browsing.

Another problem of categorization is that only a limited amount of content can be handled. In Yahoo!'s case, only a minority of existing web sites is included – a much smaller fraction than what is indexed by web search engines. Yahoo!'s category browser has therefore been supplemented with a standard query interface similar to what was presented in Section 4.2.5.



While the approaches and user interfaces presented above all are made for accessing information, they vary widely in capabilities, functionality and design. To conclude this section, I therefore present a classification based on four of the five<sup>1</sup> key challenges described in Section 1.2:

◆ **Large information repositories**

As support for such repositories is central to the overall research question, it is helpful to identify any features the interfaces might include in this direction.

◆ **New classes of users**

What I find to be the most suited target audience. The user categories are those presented in Section 3.3.

◆ **Complex data models**

One of the differences between the various presented interfaces I find most evident, is whether the data model is exposed or hidden from the user. As discussed above, this has usability consequences if the model is complex.

◆ **Different information access strategies**

As discussed in Section 4.1, support for more than one strategy might be required to fulfil different information needs.

---

1. The fifth key challenge, multimedia data types, is mostly relevant for the “Content-based Multimedia queries” category only, and is therefore omitted.

---

My classification is presented in Table 4.1. Please note that as many different implementations of the different approaches exist, I only present what I find to be the most typical characteristics.

	Support large repositories	Typical target audience	View of data model	Different strategies
Textual query languages	No special support	Expert	Exposed	Query
Query-by-example	No special support	Casual-Expert	Exposed	Query
Forms	No special support	Novice-Casual	Hidden	Query
Content-based MM queries	Ranking	Novice-Casual	No model / Hidden	Query (browsing)
Information retrieval	Ranking	Novice-Casual	No model	Query (navigation)
Dynamic query	Continuous representation	All	Hidden	Query (browsing)
Natural language queries	No special support	Novice	No model / Hidden	Query
Category browsing	Static category hierarchy	Novice-Casual	Exposed	Browsing, navigation (query)

**Table 4.1 Classification of different approaches for accessing information.**

## 4.3 Key challenges

In this section, I discuss each of the five key challenges from Section 1.2 in light of the existing interfaces presented in Section 4.2. This serves both as a conclusion to this chapter and an introduction to my own designs and implementations presented in later chapters.

### 4.3.1 Large information repositories

The research question presented in Section 1.2 explicitly states that examining the design of interfaces for information access in large repositories is *the* key issue in this thesis.

The size of an information repository impacts how it can be accessed in a number of ways. Based on the presented interfaces in Section 4.2, I find the following three facets to be most apparent:

#### ◆ Power vs. usability

As an information repository grows, the size of the haystack increases while the needle often tends to remain a constant. One way of making certain that this needle can still be found, is to give

---

the user more power to express what he or she is looking for. Interfaces such as SQL (Melton and Simon, 2002) and QBE (Zloof, 1977) are well suited for this approach as they allow fine-grained control. However, these kinds of interfaces tend to be difficult to use – requiring much training and relying extensively on recall rather than recognition. Dynamic query interfaces have taken a different approach, sacrificing some expressive power to make the interface usable to novices with minimal training.

◆ **Handling large results**

Regardless of the amount of expressive power available, query results will sometimes be too large to make sense of with irrelevant items obscuring the relevant items. This issue can be handled in several different ways. One of them is to improve the presentation of the result. Intelligent ranking algorithms is one example, typically used in web-based information retrieval interfaces (see Section 4.2.5). Manual controls for sorting is another example.

Another possibility is to view a result as only an intermediate stage in the process of accessing information. The basic idea is to use a result as a starting point for further interaction – for example by allowing new queries within a previous result rather than of the whole data repository. Another example is to use filters to remove irrelevant objects from the query result.

◆ **Performance**

Although not a focal point of this thesis in itself, performance is an important factor for usability – for example by limiting the amount of processing that can be done to enhance the usefulness of a result. For example, dynamic query interfaces has proved to have performance problems with very large results (see Section 4.2.6). A different example is web-based information retrieval engines (such as Google) where it is very important that the searchable information repository is as large as possible. This limits number the techniques that can be used without compromising performance.

### **4.3.2 New classes of users**

Traditional user environments, such as inside a company, made it possible to design interfaces that required an (modest) amount of training. By knowing what kinds of users that would be using an inter-

---

face, one could be fairly certain of their requirements, background knowledge and experience. All this made it reasonable to design user interfaces tailor made to a pretty uniform group of users.

Increased communication capabilities and the emergence of the Internet, provide a new kind of user environment where anything that is published potentially has a very broad user base. A casual Internet user can not be expected to know the underlying data model (if any) or to have received training in the use of an interface. In addition, such users will most probably only be willing to spend a minor effort and tolerate only a few problems before moving on. They are in other words notoriously unfaithful (Nielsen, 1997).

This still comparatively new environment poses challenges to the interface designer. Ideally, new users must be able to operate a web application without any prior knowledge (besides some general interface knowledge). A key example is web-based information retrieval interfaces (see Section 4.2.5) where the interface has been made as simple as possible (typically only a text field and a button).

Making an interface so simple that it can be used by everyone, typically comes at an expense. For example, the expressive power tend to be quite lower than more complex interfaces such as query languages. This could potentially hurt an interface's chances with expert users. Therefore some interfaces include an advanced mode which offers greater control over the query.

### **4.3.3 Complex data models**

When the information repository is a database, there is by definition a data model involved. A data model (or database schema) is a collection of concepts used to describe the static structure of a database. According to such a model, every object will have a number of properties, which typically includes both attributes and relations to other objects. Complex data models will typically have a great number of such properties.

When a database is to be searched, the values of the properties of objects must be compared to the search criteria. Naturally, it is very important that the correct properties are used in this process. It makes little sense to examine an attribute which contains names of countries when the user is searching for a person. As it is often impossible for a program automat-

---

---

ically to select those properties that are to be used in a specific query, this task is usually delegated to the user or the user interface designer. This poses two separate challenges:

For one thing, a data model might be seen as ambiguous or even incomprehensible to those that wish to access its contents. This can be a result of imperfect database modelling, but often domain to be modelled just does not allow a solution that is intuitive to everyone. Masking such deficiencies by good interface design is not an easy task – the solution has thus often been to train users in the understanding of the underlying data model.

Secondly, complex data models often result in complex interfaces. Making a streamlined and “less is more” interface which allows the individual selection of hundreds of properties is indeed a challenge in itself. Thus the choice is often between easy interfaces with little room for detailed query specifications and complex interfaces where every possible query can be made. Dynamic query is a good example of the former, while SQL and QBE are examples of the latter.

#### **4.3.4 Different information access strategies**

As discussed in Section 4.1, several different strategies for accessing information can be imagined. In my opinion, designing an interface to support just a single strategy, limits its usefulness. While some users can have an almost exact idea of what they are looking for, others can just be interested in browsing an information collection.

In addition, the user’s information need can change during a session. Most information searching systems are designed for an interaction model where the user first issues a query, receives the results and then either stops the query process or restarts the process (Baeza-Yates and Ribeiro-Neto, 1999). This might, however, not be the model that closest corresponds to the real behaviour of information seekers.

In (Bates, 1989) Marcie J. Bates suggest an alternative, “Berry-picking” model of information seeking. She argues that the information needs of a user constantly change as the information searching process progresses and that information therefore is gathered in bits and pieces rather than in one single set. Systems for information access should therefore be able to handle a variety of search techniques and information sources.

---

The challenge for the interface designer is to design an interface which seamlessly integrates several strategies – from planned querying to more casual browsing. Browsing capabilities are of special interest to new users as they probably lack an understanding of the contents of the information source they are accessing. Similarly, query capabilities are of more interest to those that know exactly what they are looking for and master the vocabulary in which to express this information need. Some category browsing interfaces such as Yahoo!, includes a query component, but this often seem to have been added in afterthought and therefore not fully integrated.

### **4.3.5 Multimedia data types**

Searching complex data types such as images, video and audio presents a host of unique interface challenges. A typically characteristic of complex data types is their richness as captured in the saying “a picture is worth a thousand words”. The problem is that these “thousand words” can be very difficult for a computer to extract, and very time consuming for a human to register manually (Aigrain et al., 1996). Therefore, one typically resort to two different alternatives: Query only what can be automatically extracted or query manually registered metadata.

For the latter alternative, querying multimedia data is somewhat similar to querying textual data. The most important difference is that several multimedia types have a spatial and/or temporal dimension. This difference should be accounted for by for instance allowing users to specify how the objects of interest are related in time.

Querying automatically extracted metadata offers greater challenges. The main problem is that the extracted information, typically colour, shape or texture, seldom have intrinsic value. Most users are more interested in more high-level features, such as the name of the persons shown in a picture or the location where video was shot, as these are the terms by which humans remembers such objects. The interface designer becomes (at least partially) responsible for limiting the size of the gap between what is available and what is useful (Rui et al., 1999, Enser and Sandom, 2003). QBIC (Flickner et al., 1995) is an example of an interface where the user can query automatically extracted metadata.

In addition to challenges in query specification, it can also be difficult to visualize a search result consisting of multimedia data. Data with a spatial dimension is naturally screen space intensive to present and therefore difficult to gain an overview of (Aigrain et al., 1996). Even

---

---

more difficult to handle are temporal data types. For example, audio only carries meaning when it is being played back – with video you at least get a still image when you pause the playback, paused audio clips give you nothing at all. This makes it very time-consuming to gain an overview of a large collection of audio clips without listening to all of them in full.

## **4.4 Summary**

---

In this chapter, I have focused on user interfaces for accessing information repositories. I first examined how users might go about locating information (Section 4.1). To further gain an understanding of the field, I then presented an overview of existing interface paradigms (Section 4.2), before I on this basis presented some of the key issues I have identified (Section 4.3).

---





---

## Chapter 5

# Four fundamental design ideas

---

Back in Section 1.2, the overall research question to be investigated in this thesis was stated as:

*How can user interfaces for accessing information in large repositories be designed to provide assistance to users without impairing usability?*

In the previous chapters, I have presented an introduction to user interfaces in general and to user interfaces for information searching in particular. With this and the design challenges discussed in Section 4.3 in mind, this chapter presents four fundamental design ideas designed to address the research question. As previous work exist for each of the ideas to a varying degree, I also briefly mention related approaches. More is presented in connection with my actual implementations in later chapters.

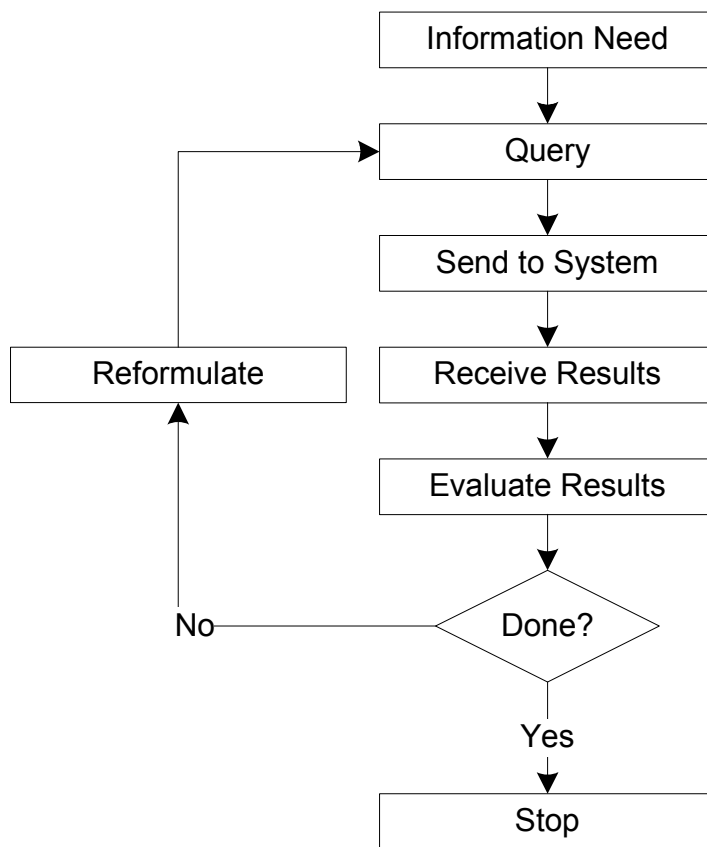
These four ideas are:

- ◆ Revised interaction model
- ◆ Intra-result analysis
- ◆ Active user interfaces
- ◆ Dynamic user interfaces

Together, they form the basis for designs and implementations presented in later chapters. These implementations are then used as a platform for evaluating the merits of the design ideas – separately, as well as in combination with each other.

## 5.1 Revised interaction model

As observed by Hearst (Baeza-Yates and Ribeiro-Neto, 1999), most of the interfaces used for information access (including several of the most popular WWW search engines) use the standard model of interaction (Salton, 1989). This model is illustrated in Figure 5.1.



**Figure 5.1 Standard model of interaction (Baeza-Yates and Ribeiro-Neto, 1999).**

According to this model, users start by issuing a query to the system. The system then performs the search and sends the result back for evaluation. If the result does not satisfy the information need, the user can reformulate the initial query and start the query-process all over again.

In my view, this model is not without its shortcomings. Important with respect to my research question, is the fact that the user is offered no help after the result has been presented. If a result is not satisfactory, the

only options are to reformulate the query or abandon it altogether. Unfortunately, reformulation is often difficult, no clues are given as to which reformulation of the initial query will produce the best result.

If a result does not contain the desired objects, nothing is lost by restarting the query process. On the other hand, if the relevant objects are buried beneath heaps of irrelevant objects, it is wasteful to throw them away. As databases and search results grow larger, this is a more and more likely situation. Few users have the time and motivation to manually browse several thousand objects just to find a handful. Consequently, such results are almost always useless.

It is my view that handling of large results<sup>1</sup> can be improved by using a revised interaction model with two important differences compared to the standard model given above. For one, the interaction should be an iterative cycle rather than a strictly start-to-finish process. Using an interactive model, a user is free to modify a result – it does not have to be final. If the result is not satisfactory to the user, he or she has the choice between restarting the process or continuing to work with the result in order to produce a satisfactory result.

Secondly, the interaction model should integrate more than one method for accessing information. With more than one method available, the user can dynamically select the method most suitable for his or her needs. It also makes it less likely that users will get stuck without the right means to continue.

In addition to methods which always generates new results (such as query), it is possible to allow an existing result to be *modified*. One way to achieve this is to use filters. Filters are constructed on the basis of presented results, removing objects that do not satisfy a particular restriction. For example, a query in a housing database might give a result of unmanageable size. Based on this feedback, the user decides that housing in a particular area is not of interest and constructs a filter to remove them from the result. Thus, the average relevance (or precision) of the remaining housing alternatives, improves.

A different way to allow result manipulation, is to have several different available visualisations (or presentations) of the result. Possible examples include an overview of as many objects as possible, categoriza-

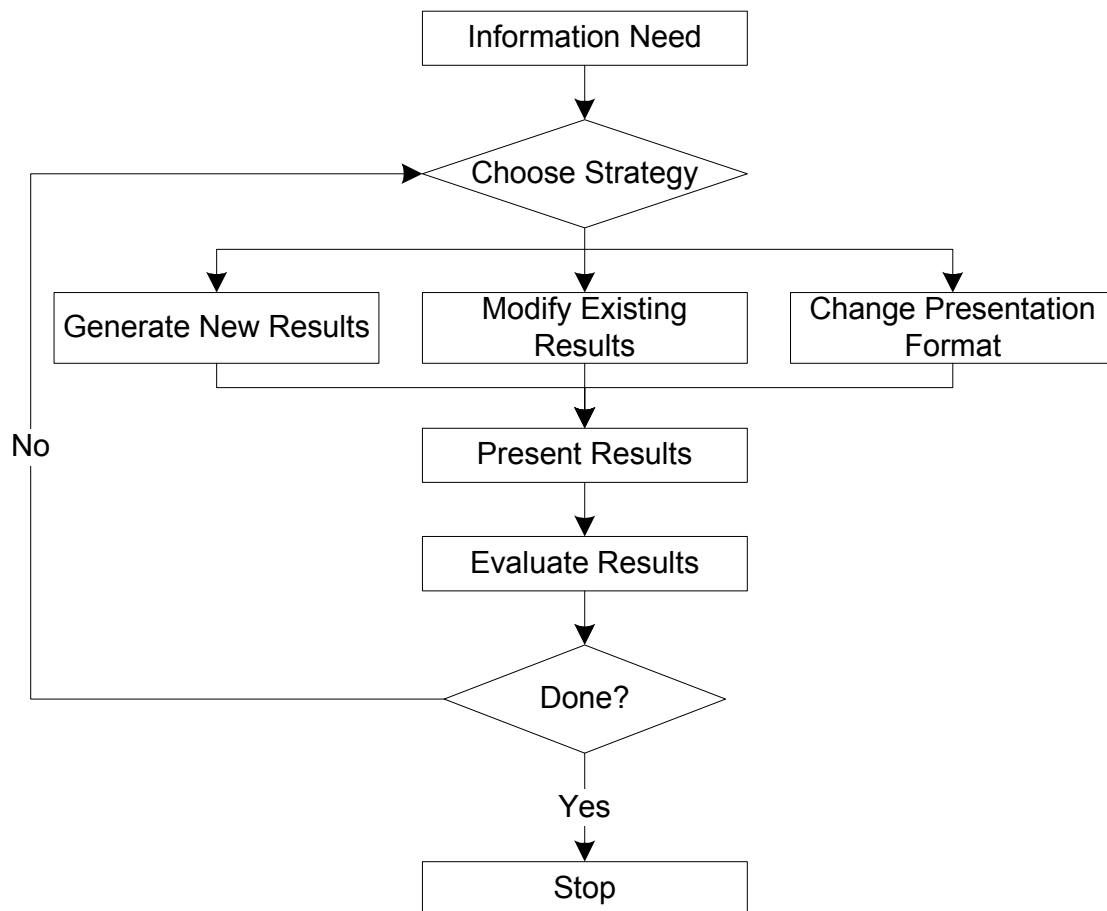
---

1. This does not have to be a query result, it can be any set of objects (for example the whole database). But to keep the terminology similar to the one used in the standard model of interaction, “result” is used throughout the discussion.

---

tion based on user-specified restrictions, detail view of a single object, lists of objects ranked according to a specific criteria etc. Each of these examples serves specific purposes and therefore complement each other. Consequently, having several different *presentation alternatives* available should be an advantage when accessing information archives. With several presentations available, some way of switching (or navigating) between them, must be present.

A revised interaction model integrating result generation, result modification and presentation alternatives in an iterative manner is illustrated in Figure 5.2.



**Figure 5.2 Revised model of interaction.**

Similar to the model illustrated in Figure 5.1, a user can generate a new result (for example by issuing a query). The user can then navigate between different presentations and modify the existing result – hopefully thereby making it easier to get relevant data out of the information repository.

---

A priori advantages and disadvantages with this design idea:

◆ **No result is final**

Instead of constantly starting over, the user has alternatives that use the currently result as a starting point. This should improve the interface's ability to handle large results as they can be predictably reduced.

◆ **Several ways to reach a goal**

More methods for information access mean more tools available to the user. If one of these are unusable (e.g. the user can not find a good reformulation of the query), other means are available to bring the process forward.

◆ **Positive synergies**

By integrating several methods for accessing information it should be possible to construct an interface that is greater than the sum of its parts by taking advantage of synergy effects. For example, by having filters constructed on basis of the current presentation, the filters could become more context-sensitive – i.e. adapted to the current presentation.

◆ **Handling of shifting information needs**

By the virtue of its interactive nature, this model should be able better to cope with shifting information needs.

◆ **Increased user interface complexity**

More options available to the user tend to result in increased interface complexity. It remains to be seen whether this offsets the possible gains of the revised interaction method.

◆ **Difficult to design good interfaces**

The interface designer is tasked with presenting more information and providing more controls while still keeping the interface as intuitive and user friendly as possible. Clearly, the quality of the implementation will have a great impact on applicability of this revised model.

The idea of using an iterative interaction model to improve information access, is not new. A typical approach that have been used, is to allow the user to rate the relevance of different query terms related to earlier queries and use this feedback to assist the user in reformulating the query. Examples of this and related approaches include *relevance feedback* (Salton, 1989), *iterative query refinement* (Rao et al., 1995) and

---

*scatter/gather* (Cutting et al., 1992). While such approaches originally were used for text repositories, they have also been applied to, for example, image databases (Wood et al., 1998). These approaches all have that in common that they are not focused on supporting shifting information needs – the assistance the user receives is geared towards the information need evident from earlier queries.

The importance of supporting different strategies for accessing information, have already been discussed in Section 4.1 and Section 4.3.4 along with an overview of to which degree existing classes of interfaces support multiple strategies in Table 4.1.

## **5.2 Intra-result analysis**

---

Perhaps the most important challenge for information access tools mentioned in Section 4.3, is the handling of large results. A key point in this regard is Shneiderman’s mantra for information visualization (Shneiderman, 1997):

*“Overview, zoom & filter, details on-demand.” (page 523)*

Gaining an overview of a large result can be very difficult if no support is given by the interface. Manual scanning of thousands of individual results is something few, if any, users are likely to do. One possible way to assist the user is to look at the techniques that have proved successful for much larger data collections (such as data warehouses) and apply similar techniques to query results.

A key concept in this regard, is *data mining*. Data mining can be defined as (Han and Kamber, 2000):

*“[...] extracting or “mining” knowledge from large amounts of data.” (page 5)*

Techniques such as association, classification and clustering have long been used in a wide variety of disciplines – typically to analyse and understand amounts of data too large to be handled manually. Data mining is about exploring data interrelations to, for example, find typical buyer patterns, common characteristics in a population or predicting risky loans.

While query results are by nature much more transient than data warehouses, they are also typically several magnitudes smaller. In my opinion, it should therefore still be possible to apply the same principles

---

---

used for data mining to dynamically analyse the properties of a query result. This could very well provide information that would greatly aid the user in gaining an overview of a large result. For a result containing pieces of music, information as simple as most common artist, composer, genre, etc. would go a long way towards providing at-a-glance information about the general properties of the result – perhaps enough for the user to decide if the contents warrant closer examination.

Further, information derived from an analysis could very well be useful beyond providing an overview of a result. It might also be very well suited for filtering operations – i.e. remove music by most common composer. This would make the process of constructing filters more reliant on recognition than recall and thus ties nicely with the active user interface idea to be explained in Section 5.3.

A priori advantages and disadvantages with this design idea:

◆ **Easier overview**

As described above, the derived information gained from an analysis of the properties of a collection of objects, should make it easier to understand the nature of said objects.

◆ **Useful for filtering**

If you wish to reduce the size of a query result, there are few better ways than removing the more common objects. Such operations would be simplified if these objects were automatically identified.

◆ **Novelty factor**

Potentially, analysis techniques could be perceived as something approaching artificial intelligence. This could have a positive effect on at least the subjective performance of the interface.

◆ **Computationally expensive**

By its very nature, results are temporary constructs. Thus intra-result analysis can be expected, at least to certain degree, to be on-the-fly computations. As the response time of information access systems are of concern, computation requirements might be a limiting factor for this idea's applicability.

◆ **Derived information trivial?**

Once any novelty factors have worn off, the quality of the derived information will be the deciding factor. If only irrelevant or trivial information can be extracted, no ground has really been gained.

---

While used in a quite different setting, data mining is clearly an related approach. Extraction of implicit information to make it easier to cope with large amounts of data, is exactly what I am interested in doing – only this time for query results. For an overview and classification of data mining techniques, see (Chen et al., 1996).

An often used approach to make it possible to handle large query results, is to rank them according to some measure of relevance. While this method does not explicitly provide the user with derived information, the ranking is typically based on such information. For example, the PageRank method (Page et al., 1998) used by the Google WWW search engine (see Section 4.2.5) uses link structure to establish the importance of a given page by looking at the importance of the pages that link to this page.

Rather than computationally extracting high-level information, such information can be supplied by other users. Collaborative filtering is one example where opinions of other users that have agreed with the current user in the past, is used to filter or rank information objects. For instance, (Resnick et al., 1994) presented GroupLens, a system for collaborative filtering of bulletin board news posts.

### **5.3 Active user interfaces**

---

In order to construct usable interfaces, one must consider the human capabilities. As described in Section 1.2.1, one of the most important areas in this respect is the human memory system and the concepts *recognition* and *recall*. Recognition has been proved to be much easier to perform than recall (McCracken and Wolfe, 2004). For example, recognizing a cow is easy – recalling the visual image needed to draw a cow, however, is another issue altogether.

It therefore makes sense to take this into account when designing user interfaces by allowing users to apply recognition whenever possible. A good example is the different interaction styles discussed in Section 2.3. The advantages of recognition compared with recall goes a long way to explain why menus and direct manipulation interfaces have gained prominence at the expense of command entry. While the first two have users selecting (recognizing) something, the latter requires the user to type (recall) commands.

---



---

The way in which I intend to make recognition more prominent, is to make the user interface more *active*. Rather than simply waiting for users to initiate interaction, user interfaces should in my opinion be more active by having the users respond to relevant choices presented to them. These could for example be a limited number of suggestions as to how to proceed with the interaction, thereby hiding the more advanced options from non-expert users.

A priori advantages and disadvantages with this design idea:

◆ **Easier to learn**

As it is based on recognition rather than recall, active user interfaces should be easier to learn as memorization can be kept at a minimum.

◆ **Fewer user errors**

One of the key differences between menus and command entry, is that the designer can make sure that menus include all available operations and nothing more while the user of command entry interfaces can input anything he or she might think of. By only selecting from allowed choices, menus thus have much fewer possibilities for user errors.

◆ **Faster to use**

Recognizing the relevant action to select from series of allowed choices is typically faster than recalling commands from memory – especially for novice users. However, if many choices are available, simply gaining an overview of them might take longer than it would take to remember and enter a command. The speed of use of an interface based on recognition is therefore difficult to predict as it will depend on implementation, application and user.

◆ **Low relevance?**

Again, when many choices are possible, active user interfaces might not perform at its best. Who has not experienced searching through several long menus to find the one option one is looking for? Such frustrating experiences could very well be the bane of such interfaces.

◆ **Limited applicability?**

Not every kind of user action can be expressed by an interface solely reliant on recognition. For example, a text field for entering query expressions (recall) is difficult to get around.

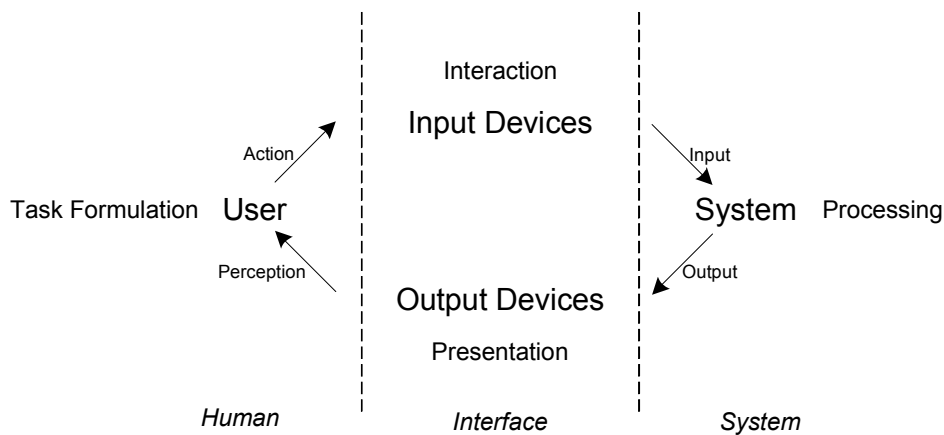
---

The term “Active user interface” has been used before. It was for example used in (Santos Jr. et al., 2001) to describe Kavanah, a information retrieval user interface. This interface contains an interface agent which tries to determine the user’s goal by maintaining a model of the user’s interaction. This information is then used to construct queries on the user’s behalf. In difference with my design idea, this interface is active by automatically taking action for the user, rather than being active by initiating interaction or dialogue.

A different approach which also is related to my “Active user interface” design idea, is online help systems using wizards and cue cards (Phelps, 1997),(Tidwell and Fuccella, 1997). Here the idea is that the documentation is active – that is that information is presented in response to the user’s actions. Wizards, for example, ask the users only the questions it needs to do a given task thereby relieving the users of having to navigate the user interface looking for the relevant options themselves.

## 5.4 Dynamic user interfaces

In Section 2.1, I presented the interaction cycle model by (Abowd and Beale, 1991), repeated here in Figure 5.3.



**Figure 5.3 The Interaction Cycle (Abowd and Beale, 1991).**

As shown in this figure, the user interface is responsible for presenting output from system processing to the user as well as making interaction controls available which the user then can use to express his or her actions. Typically, the presentation format, as well as the available interaction controls, remains constant regardless of the output.

It is my belief that having static presentation format and interaction controls has a few disadvantages. In Section 2.2, I presented the view that human-computer interaction has much in common with human-

---

human interaction and that much of the same principles can be applied. Of particular interest in this discussion, is Gricce's maxim of relevance (Grice, 1975):

*“What people (and media) say should clearly relate to the purpose of the conversation”.*

A static interface, can easily have interaction controls that bear no relevance to, or at least are sub-optimal with respect to, the presented data. According to the relevance maxim, it would have better if such controls were omitted so that every presented control was clearly relevant.

I intend to investigate if it is feasible to have dynamic user interfaces where the system output determines the way in which data is presented, as well as the interaction controls made available to the user. If realised, this would give a context-sensitive user interface with what-you-see-is-what-you-need – that is a dynamic adjustment of the interface depending on presentation content.

A priori advantages and disadvantages with this design idea:

◆ **Increased relevance**

By minimizing the use of irrelevant controls and maximizing the use of relevant controls, users will not have to struggle to understand something that is of no use anyway.

◆ **More suitable presentation format**

For a given output, many different presentation formats are imaginable – not all of the equally suited in all cases. For example, overview rather than detail might be more useful for large results with the opposite being true for small results.

◆ **Dynamic user interface might be confusing, limiting recognition**

Depending on the degree in which the interface is dynamic, sizable parts of a system's interface might change on a regular basis. This resulting confusion might reduce the usability of the interface and thus make this idea less helpful.

◆ **Achievable in practice?**

If a system is to decide what is relevant in a given situation, it is critical that these decisions are agreed upon by the users. After all it is what users find relevant that counts, not the designer's opinion.

---

Online help systems, mentioned in Section 5.3, also have clear relations to this design idea. For many such systems, see for example (Taboada et al., 1996), the idea is to provide context-sensitive assistance to the user. I intend, however, is to apply this concept to the interface as a whole, rather than to the help system only.

An approach perhaps closer related to the idea presented above, is *adaptive user interfaces* (Langley, 1999). This class of user interfaces adapts the interface content (and to a lesser degree its presentation) to the current user according to what the system has learned from previous interactions. Thus it tries to improve interaction by personalizing the interface. In comparison, my approach is to adapt the interface to the content rather than adapt the content to the user.

## **5.5 The road ahead**

---

In the following chapters of this thesis, I present designs, implementations and evaluations that aim to show the merits of these four design ideas with respect to the overall research question.

Chapters 6, 7 and 8 present a study of a specific interface for information access in textual metadata databases. As text represents the fundamental and most used data type, it was natural to start there. In order to gain an understanding of the usefulness of usability evaluations, I have performed two cycles of design – implementation – evaluation so that the improvements made from the lessons learned in the first evaluation can be investigated.

Chapter 9 describes an effort to extend the textual design to image databases and content based queries. Images are a quite different data type compared with text, and it is therefore interesting to study how the design ideas work for images. In particular to see if they can be used to bridge the semantic gap between what can be automatically extracted from an image and the high-level concepts natural to humans.

In Chapter 10 I turn to temporal data – or to be exact temporal annotations of video. This domain is interesting as the temporality not only complicates interaction and presentation, but also adds another dimension to the data thereby making manual identification of patterns and trends more difficult. I wish to apply the design ideas of intra-result analysis and integration of several information access methods to investigate if tools can be constructed that make accessing such repositories easier.

---

In Chapter 11 the results from the previous chapters are discussed and used to examine the viability of the four design ideas. Chapter 12 concludes the thesis with a presentation of main contributions.

---



---

## Chapter 6

# Searching supported by analysis of metadata

---

In this chapter, I present an approach for accessing information in textual metadata databases based on the four design ideas presented in the previous chapter. Design, implementation and evaluation of two interface prototypes based on this approach is presented in Chapters 7 and 8.

The choice of metadata databases as test case, was in part made because one such database suitable for my purposes became available to me at an opportune moment in the work on this thesis. The database in question, Primus, is a large real-world relational database used by the Norwegian Folk Museum to store information about old artifacts and pictures. Section 6.1 presents details about this database, its data model and current user interfaces.

Traditionally, searching in databases has either been done using forms-based interfaces or by issuing queries using a query language (for instance SQL – see Section 4.2 for details). Both these alternatives are characterized by a tight coupling with the underlying data model. As a result, some understanding of this model is almost a prerequisite for accessing the data. This works well as long as the user-base is restricted to persons who use the database regularly and have been given training in how to use it. However, for novice users with minimal background knowledge and training, such interfaces have proved very difficult to use

(see Section 4.2). With this in mind, I presented four design ideas in Chapter 5 which all are aimed at giving user-friendly interfaces that can cope with large information repositories and complex data models.

In Section 6.2, I discuss how each of these four design ideas can be used in a user interface for accessing Primus. Section 6.3 presents key components of my proposed approach, while Section 6.4 presents some related approaches. Section 6.5 concludes the chapter.

The contents of this and the following chapter is an expanded version of (Hauglid and Midtstraum, 2002).

## **6.1 Primus**

---

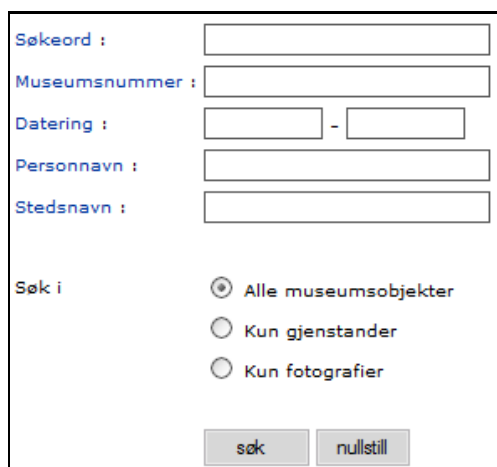
The Primus project was initiated in 1996 in order to make a unified, digital information system for registration, administration and presentation of Norwegian museum collections (Museenes Datatjeneste, 2003). The resulting Primus system consists of a data model, a database configuration and application programs to access this database. Primus is now being used by more than 30 Norwegian institutions.

In the start-up-phase of my thesis, I got access to the version of the Primus database used at the Norwegian Folk Museum. This relational database contains information (metadata) about more than 150.000 artifacts and 260.000 photos. The metadata consists of textual descriptions stored in more than 150 different relational tables as well as 236.000 images. Typical stored information include:

- ◆ Place of origin (hierarchical – county, district, etc.)
  - ◆ Date artifact was made, photo taken
  - ◆ Techniques used to construct artifact
  - ◆ Motif on photos
  - ◆ Material
  - ◆ How the artifact or photo was put into the collection
  - ◆ Previous owners
  - ◆ Classification (using a custom-made ontology)
-







The screenshot shows a search interface with the following elements:

- Søkeord :
- Museumsnummer :
- Datering :  -
- Personnavn :
- Stedsnavn :
- Søk i
  - Alle museumsobjekter
  - Kun gjenstander
  - Kun fotografier
- søk
- nullstill

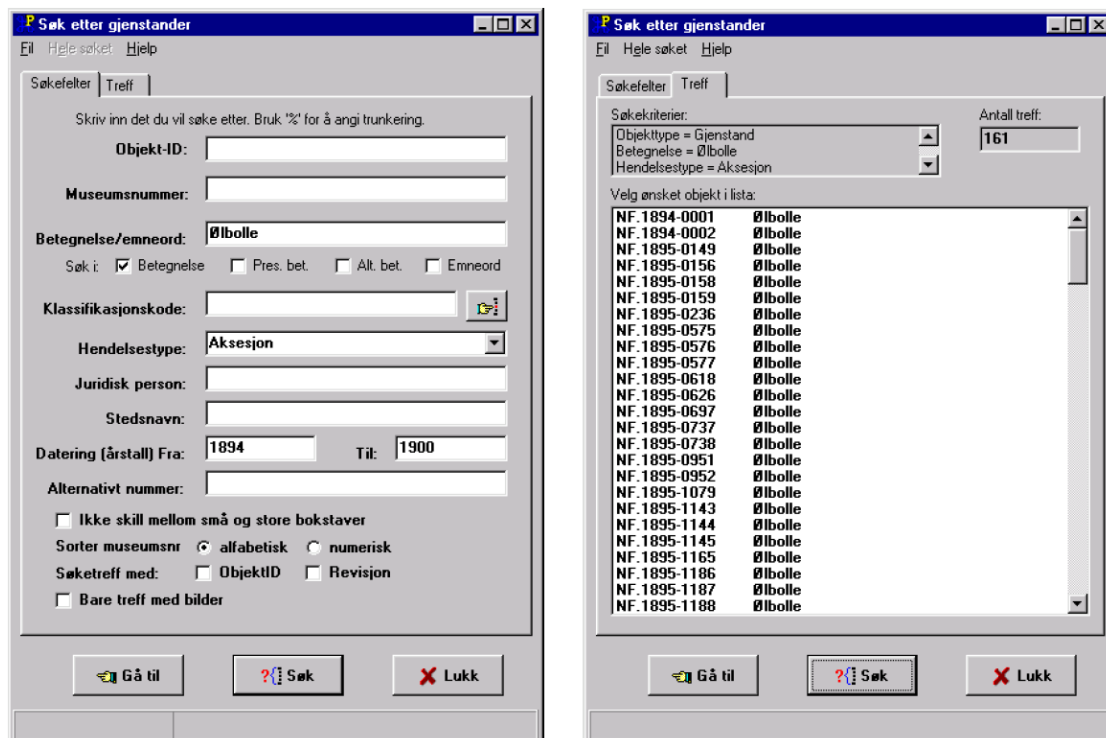
**Figure 6.2 PrimusWeb.**

This interface is forms-based, with five searchable attributes (object name, museumnr, date, person and place of origin). If more than one of the form fields are used, the result will contain only objects that satisfy all of the entered query terms. The three radio buttons at the bottom is used to select if both artifacts and photos, only artifacts or only photos are to be searched.

So few searchable attributes results in an easy-to-use interface with few controls to understand and reduced chance to use the wrong field by mistake. On the other hand, it limits the expressive power as complex queries is impossible to construct. Still, with novice users as target audience, this is an acceptable trade-off.

The second user interface of these is shown in Figure 6.3. This interface allows (trained) users to construct precise queries by entering restrictions (left image) for a large number of attributes. Results are presented as a list with item number and name (right image). More details can then be found by accessing each item individually.

---



**Figure 6.3 Primus query interface.**

Even if these two interfaces both have room for improvements, it was not this that attracted me to the idea of using Primus as a case study for testing the four design ideas presented in Chapter 5. Rather, the data model and the way in which objects are described, have several useful properties:

◆ **Complexity and richness of data model**

One of my early ideas was that a rich description model could be used to improve the handling of large results (see Section 6.2.1). As indicated above, the data model for Primus is so complex that it was well suited for testing this idea.

◆ **Size of database**

As the focus of this thesis is to investigate interfaces for accessing large information repositories, it was important to find a database that was large enough. With more than 400.000 described objects, Primus was found to be almost ideal in this regard.

◆ **Real-world database**

By making a custom, artificial database to test a set of ideas, it is possible to introduce bias that might hurt the validity of later evaluation results. With a database that has been made for and used in

the real-world, this is less of an issue. However, the act of selecting Primus as a test case, is still a possible source of bias.

## **6.2 Applying design ideas to textual metadata databases**

---

The overall objective of my work, as stated in the research question, is to investigate if it is possible to construct user interfaces for information access which combines usability with support for handling large information repositories.

The four design ideas presented in Chapter 5 are the cornerstones of this investigation as they form the foundation for the user interfaces I have designed, implemented and evaluated. The first of these targets textual metadata databases such as Primus. Below, I examine how each of these four ideas can be used to improve information access in such databases.

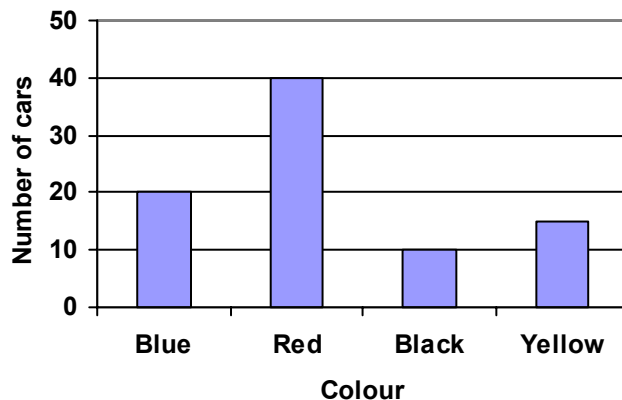
### **6.2.1 Intra-result analysis**

The objective of this design idea is to aid the user in handling large results by using a run-time analysis of the properties of objects in a result to derive high-level information. I intend to implement this idea by taking advantage of the structured way in which database objects are described.

A data model (database schema) is a collection of concepts used to describe the static structure of a database. According to such a model, every object will have a number of properties, which typically includes both attributes and relations to other objects. Each of these properties provides a means, or a dimension, to classify the objects. This observation holds for both relational and object-oriented databases.

For each of the available dimensions, the objects in a result take on a number of different values. By counting the occurrences of each of these values, the properties of a collection of objects can be summarized in a set of frequency-tables. Suppose a database containing information about cars is searched for all cars whose name includes the word “Ferrari”. For the objects returned, the describing dimensions might include year of production, colour and model. An example of a possible distribution of cars with regard to colour is illustrated in Figure 6.4.

---



**Figure 6.4 Colour-distribution of cars.**

This analysis of object properties can be done at run-time and thus be used to give the user key characteristics of a result by summarizing what can be extracted from frequency tables for different dimensions. This can be seen as *data characterisation* (Han and Kamber, 2000).

Further, it should be possible to use such information to construct filters to modify a result. The resulting frequency-tables provide means to partition the objects in a result. Each of the partitions contains objects with a specific value in a given property. If we presume that the user is able to select one or several partitions, objects not included in any of them can be filtered away. Thus, the interesting objects are exposed, not only increasing the average relevance (precision) of the objects in the result, but also reducing the result size.

## 6.2.2 Revised interaction model

The idea of a revised interaction model has two different components. First, interaction should be an iterative cycle where it is possible to modify a query result if it was not 100 % satisfactory, instead of forcing the user to restart the query. Second, more than one way of accessing information should be provided. In this way, the user will be less likely to get stuck with no suitable avenues for further interaction.

Traditional textual query interfaces (such as web-based information retrieval – Section 4.2.5) have proved easy to use and are familiar to a great fraction of the computer-using population. It therefore makes sense to include support for such queries as one way of accessing information.

In addition, a way of modifying existing results should be present. As described in Section 6.2.1 this is done by filters based on the result of an intra-result analysis. Because this analysis is dynamic, it makes sense to support an iterative use of filters. As one filter is made, the properties of the result will change, leading to updated information from the analysis process. By providing the user with a means to make filters on the basis of this updated information, the result can be further modified. In this way, the user can approach a satisfying result step by step.

### **6.2.3 Active user interface**

The active user interface design idea aims at promoting recognition on the expense of recall. In other words, the user should as often as possible be presented with alternatives provided by the system, rather than being forced to recall how to best further the interaction.

The way in which I intend to apply this design idea to the interface for accessing textual metadata databases, is to make it possible to construct filters by answering questions. To make interaction as simple as possible, the system presents questions that suggest alternative ways of partitioning the objects in the result. These questions will, if answered by the user, select both dimension and partition. For instance, if a user answers “yes” to a question like “Are you interested in red cars?”, the partition containing red cars is selected and cars of other colours can be removed. Conversely, a negative answer implies a selection of all but the red cars.

This leads to a modified model of interaction, where answering questions is an alternative to rewriting the query. Users will more seldom need to be imaginative in finding a useful query reformulation – they instead simply have to recognize the question. As a first step, I intend to make fairly simple questions. For example, the users could be asked if they would like to remove red cars from a result. The property value (in this case red) will be determined by the system and not by the user as this would make the interface slightly more complicated.

### **6.2.4 Dynamic user interface**

In order to follow this design idea, the user interface should be dynamic (or context-sensitive) with respect to the current situation. This can be implemented by making sure that the available interaction controls are appropriate with respect to the current result. To build on what has been described above, it makes sense to accomplish this by making sure that

---

only those questions that are deemed most relevant are presented to the user. This way, the number of interface components can be kept low even when the number of available dimensions is large.

In order to achieve this, there must be some way to evaluate the (perceived) usefulness of any given question. This process will naturally be very important. If every question goes unanswered, the application of this design idea has failed. It is therefore necessary to develop a method to estimate the utility of a question on the basis of a given search result. This argument is further strengthened by the fact that there is typically a large set of possible questions. Not only can each object have a great number of properties, it is also possible to imagine many different questions based on a single property. For instance, one could ask about red cars, blue cars etc.

## **6.3 The SESAM approach**

---

As stated in the main research question, my overall objective is to research how user interfaces for accessing information in large information repositories can be made as user-friendly as possible. To this end, Chapter 5 described four design ideas that I intend to evaluate.

The first domain in which these ideas are put into practice, is textual metadata databases such as the Primus database. In Section 6.2, I discussed how each of the design ideas can be applied to this domain. My approach combines and builds upon these discussions. As the central concept is metadata analysis, I have dubbed the approach “SEarching Supported by Analysis of Metadata” or SESAM.

In the remainder of this chapter, I examine two key issues with SESAM. The first is how SESAM can be applied to a database, the second how the utility of questions (filter suggestions) can be estimated and compared. Two prototype interfaces based on the SESAM approach are presented in Chapter 7 and Chapter 8.

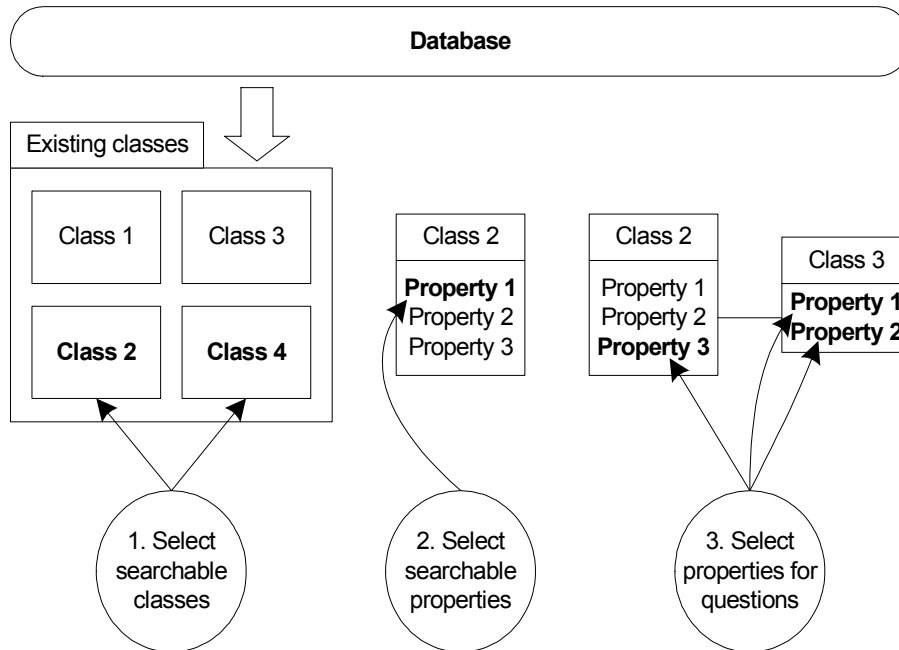
### **6.3.1 Applying SESAM to a database**

Before SESAM can be applied to a database, it has to be configured to support the data model. The database-specific actions that have to be performed before SESAM can be used, can be divided into the following three steps:

1. Identify relevant classes of objects in the database – what kinds of objects should be searchable?
-

2. Determine searchable properties. Which object properties should be used when generating processing textual queries?
3. Identify dimensions to be used to reduce the size of the result – what object properties are to be used in questions?

These steps are illustrated in Figure 6.5.



**Figure 6.5 Database-specific decisions.**

### 6.3.2 Properties of useful questions

As mentioned in Section 6.2.4, a method to compare the utility of questions is very important. Naturally, it is impossible for the program to ensure that every question is relevant for any given user as it is impossible for a program to predict with 100 % certainty what the user wants.

One possible way to approach this problem could be to store information about what questions each user has answered in the past and use this information to predict the relevance of future questions. But as I find it unlikely that any user would use the system enough for detailed history information to be available and as user preferences probably will change depending on subject, I instead intend to use heuristics to estimate the relevance of a given question.

In most cases, it should be safe to assume that as a given value becomes more frequent, the likelihood that the user knows about this value increases (everyone knows of *red* Ferraris). It can also be assumed that



users are more likely to have opinions regarding a known value. They are therefore more likely to answer a question based on a familiar value. As a result, asking questions about values with high relative frequency, should be a useful heuristic.

To improve the handling of large results, the system should help the user to reduce the size of the result as much as possible. To reach this goal, an ideal question should split the result in roughly equal halves – similar to the approach used in binary search algorithms. By having two halves, the size of the result is drastically reduced no matter what the user answers.

One could argue that a 1 % vs. 99 % distribution is ideal as long as the user selects the small partition. Unfortunately, in the general case, it cannot be predicted if one value is more likely to be selected than the others. Therefore, I must assume that 99 % of the users would select the large partition – making questions based on this distribution almost useless.

So far, I have only discussed questions where the answer is either yes or no. This type of question works best if an attribute has a given value for roughly 50 % of the objects. If we have several values with similar frequency, the risk of selecting a value that is irrelevant to the user, increases. To alleviate this problem, questions where the user is asked to choose from a list of the most frequent values should be used instead.

### 6.3.3 Comparing the quality of questions

Yes/no and list-based questions should be applicable to all possible textual dimensions. They also have at least one variable – the value (in case of yes/no) or values (list) that are to be presented. As a result, we have a large number of possible questions to choose from. To be able to select the most suited, we naturally need a way to compare them.

The way in which this is done is by the means of a set of utility functions. These functions return a number between 0 (bad) and 1 (good), indicating the expected usefulness of a question. Developing an utility function for all possible combinations of question type and dimension is not possible. The available dimensions are therefore classified into two categories: Dimensions with *numeric data* and dimensions with *categorical data* (Han and Kamber, 2000). Categorical data are discrete with no ordering among the values, while a set of numeric data can be ordered<sup>1</sup>

---

and the partial ordering relation is *semantically meaningful*. For instance, year is a numeric data dimension, while colour is not. This gives four possible combinations:

- ◆ **Yes/no questions for categorical data**
- ◆ **Yes/no questions for numeric data**
- ◆ **List-based questions for categorical data**
- ◆ **List-based questions for numeric data**

For each of these combinations, a single utility function is applied. These four utility functions are described below.

Every object in a result should ideally be described in every dimension. However, in most real-world databases, this is not the case. Frequently, either the complete set of properties of every object is not known, or it makes no sense to describe a given object in a particular dimension. Dimensions where many objects have a null-value are not well suited to be used in questions as we really do not know if the known values represent the result accurately. This is taken into account by multiplying the result of the utility functions by the percentage of non-null values.

### **Yes/no – categorical data**

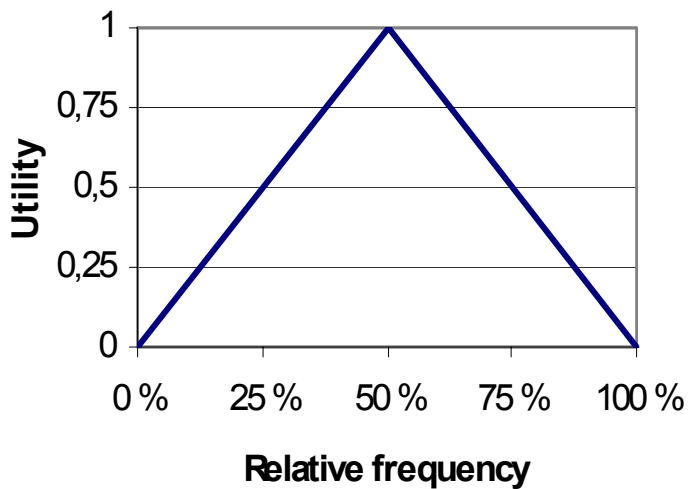
Example question: Are you interested in red cars?

As explained above, the ideal yes/no-question removes 50 % of the result regardless of what the user answers. Therefore, a 50/50-distribution is given 1 in utility. Further, 0/100 distributions are useless – either the result becomes empty or it remains the same. Thus, they should have 0 in utility. To allocate a utility to the rest of the possible distributions, I have chosen to use the triangular function shown in Figure 6.6. This function is applied to all values in the chosen dimension in order to find the question with the highest utility.

---

1. “A set  $S$  is called ordered if it is partially ordered and every pair of elements  $x$  and  $y$  from the set  $S$  can be compared with each other via the partial ordering relation.”, from <http://www.shu.edu/html/teaching/math/reals/infinity/defs/ordering.html>

---



**Figure 6.6 Utility function for yes/no-questions for categorical data.**

Several alternative functions can be imagined. For instance, one could consider distributions close to 0/100 as useless and distributions close to 50/50 as identical to the ideal case. This approach would result in a more bell-shaped function.

One could also argue that an asymmetrical function is more suitable. Users may find it easier to relate to the value present in the question, rather than considering other values. This argument is similar to the recall vs. recognition issues discussed in Section 5.3. If we follow this line of reasoning, the functions should for instance rate a relative frequency of 75 % as better than a relative frequency of 25 %. However, in the initial SESAM design, the simple function shown in Figure 6.6 is used.

In order to explain the utility functions, assume that each property  $P$  has a set of property values  $V_P = (v_1, v_2, v_3, \dots, v_n)$ . Each of the objects in the result will have one of these property values associated with it<sup>1</sup>. Assume further that  $c_{v_i}$  is the number of result objects having property value  $v_i$ . The utility of a value  $v_i$  of a given property  $P$  is then computed using the following function:

---

1. For many-to-many relationships, more than one associated value is possible. This makes it possible for the sum of all  $c$  to be larger than the total number of result objects, but this does not have any real consequences.

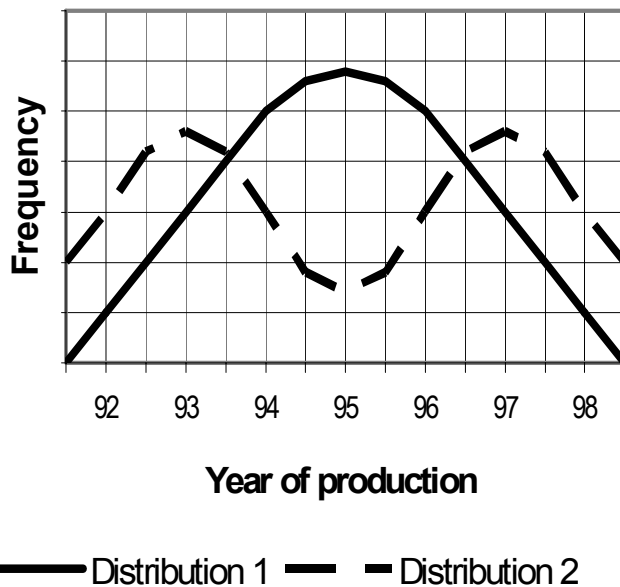
---

$$utility(P, v_i) = 1 - \left| \frac{2 \cdot c_{vi}}{\sum_{v_j \in V_p} c_{vj}} - 1 \right|$$

### Yes/No – numeric data

Example question: Are you interested in cars produced after 1995?

As indicated by this example, this class of questions is different from categorical data category. Instead of answering yes or no to a single value, the user is asked to choose a range of interest (before or after 1995). This has consequences regarding the utility function. While we still seek as close as possible to 50 % yes and 50 % no, we also need to take the shape of the distribution into account. Figure 6.7 illustrates why this is necessary, showing two possible distributions for year of production for an imagined car database.



**Figure 6.7 Different frequency distributions.**

The graph shows two possible distributions. Both are symmetrical and would have a 50/50-distribution if the midpoint (95) was selected. While distribution 1 has most of the cars in close proximity to the midpoint, distribution 2 has a larger spread. This difference is important if we assume that users have a margin of uncertainty. If this margin is 10 %, more of the cars would be inside this margin in distribution 1 compared to distri-

bution 2. The latter should therefore be preferred as fewer users would be likely to be interested in cars produced inside the margin of uncertainty, and thus more likely to be able to answer the question.

Assuming  $v_{max}$  is the largest value and  $v_{min}$  is the smallest, the utility for a value  $v_i$  of a given property  $P$  is:

$$utility(P, v_i) = \left( 1 - \left| \frac{2 \cdot \sum_{v_j \in V_P, v_j > v_i} c_{vj}}{\sum_{v_j \in V} c_{vj}} - 1 \right| \right) \cdot \frac{\sum_{v_j \in V_P} |v_i - v_j| \cdot c_{vj}}{\max(v_{max} - v_i, v_i - v_{min}) \cdot \sum_{v_j \in V_P} c_{vj}}$$

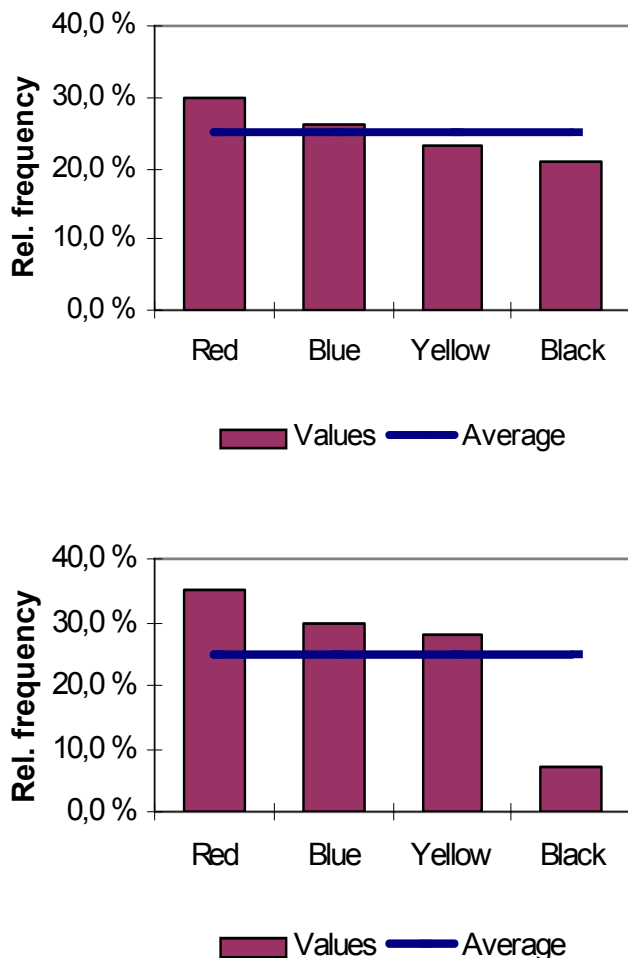
The first term is similar to the utility function for categorical data and thus promotes 50/50-distributions. The second term is used to differentiate distributions like those shown in Figure 6.7. It is a measurement of the degree in which the current distribution (numerator) differs from the most skewed distribution possible (denominator). The numerator is the sum of the distances between  $v_i$  and the actual property values of the result objects. Thus if no objects have a different property value than the one to be evaluated ( $c_{vj} = 0$  if  $v_j \neq v_i$ ), the numerator will be zero.

### List – categorical data

Example question: Choose the colours you are interested in: {Red, Blue, Yellow, Black}

Lists should be used when yes/no-questions are not suited, i.e. when no value dominates. A list contains a small number of items – from which the user can select one or several.

The lists could conceivably contain every occurring value, but this would in most cases make them far too long. In addition, many of the values would almost certainly have very low frequencies. I have used a utility function that promotes list items with similar frequencies. The graphs in Figure 6.8 illustrate why this choice was made.



**Figure 6.8 Sample distributions for categorical data.**

The graphs differ in that the first graph shows four values with similar frequencies, whereas the other graph has one value with much lower frequency. While the inclusion of this last value in a question would help those interested in it, the complexity of the question also increases. In my opinion, the latter consequence outweighs the former. A utility function should therefore select a question with four list items in the former example, and three items in the latter.

The utility for a set of values  $D_P$  of a given property  $P$  considered to be displayed to the user is computed as:

$$utility(P, D_p) = 1 - \frac{\sum_{v_i \in D_p} \left| c_{vi} - \frac{\sum_{v_j \in D_p} c_{vj}}{|D_p|} \right|}{\left( 2 - \frac{2}{|D_p|} \right) \cdot \sum_{v_i \in D_p} c_{vi}}$$

The utility function implements the policy of promoting property values with equal frequencies by first computing the sum of the differences between the frequency of each of the values and the average frequency. This sum is then normalized and inverted in order to gain increasing utility with increased similarity. To find the optimal size of the list, the utility is computed for sizes from two to seven property values. These values will always be those with the highest frequencies in order to have the displayed property values encompassing as many result objects as possible. The number seven was selected based on psychological studies (Miller, 1956) that indicate that our short-term memory is limited to storing information about ca. seven items.

### List – numeric data

Example question: Choose the years of production you are interested in: {1950-1969, 1970-1989, 1990-2000}

This class of questions is the hardest to handle since there are so many degrees of freedom. Each item in the list is a range and each range can be of a different size. Further, the number of ranges displayed to the user can also be varied.

My approach is first to find suitable ranges. When these ranges have been found, the same approach as for categorical data is used. To identify these ranges, it is assumed that a range's start and end points should be where few values are located nearby, as this will limit the impact of a margin of uncertainty (as described for yes/no-questions for numeric data). Basically, a point is well suited if the neighbouring values have low frequencies and are located far away.

Using the  $n+1$  most suited border values, it is possible to construct a partitioning with  $n$  parts. This partitioning is put through the same formula as for categorical data. To find the most suited number of items, utility

values are computed for  $n=2$  to  $n=7$ . As this approach is a combination of the approaches for the two preceding categories, the resulting utility function was omitted.

## **6.4 Related approaches**

---

Using the data model to allow the user to narrow down the result has been the topic of some previous studies. (Ahlberg et al., 1992) proposed a method for direct manipulation of databases called dynamic queries – earlier discussed in Section 4.2.6. Using this method, users can specify queries using graphical widgets such as sliders while the result is constantly updated with regard to the values selected by the user. Unfortunately, this approach cannot be applied to large databases without compromising the performance.

In later papers (Doan et al., 1996, Plaisant et al., 1999), a two-phase approach to dynamic queries was proposed. The first phase, query preview, allows users to formulate an initial query by selecting desired property values while simultaneously being shown the volume of matching data. This is made possible by pre-generating a volume preview table that indicates the number of datasets for each property value and intersections. In the query refinement phase actual metadata is collected from the database to be used in a dynamic query interface (as described above). Using this two-phase approach, the performance problem regarding larger databases was reduced.

There are two key differences between my approach and the dynamic query approach. The first is that while all values for a given property are available to be used in filtering in dynamic query, my approach is to find a most suitable subset of values to present. This is done to reduce complexity and improve relevance. The second key difference concerns which properties are presented to the user. In dynamic query, these are static and determined by the interface designer. I intend to use a dynamic subset of properties, depending on the characteristics of the current result.

A similar approach to dynamic query, called continuous querying, has been described by Shafer and Agrawal (Shafer and Agrawal, 2000). They presented Eureka, a WWW-based database exploration engine. After Eureka has performed an initial search, users can specify predicates on different attributes and immediately have the result updated. It is also possible to search using example records. The Eureka interface is shown in Figure 6.9.

---

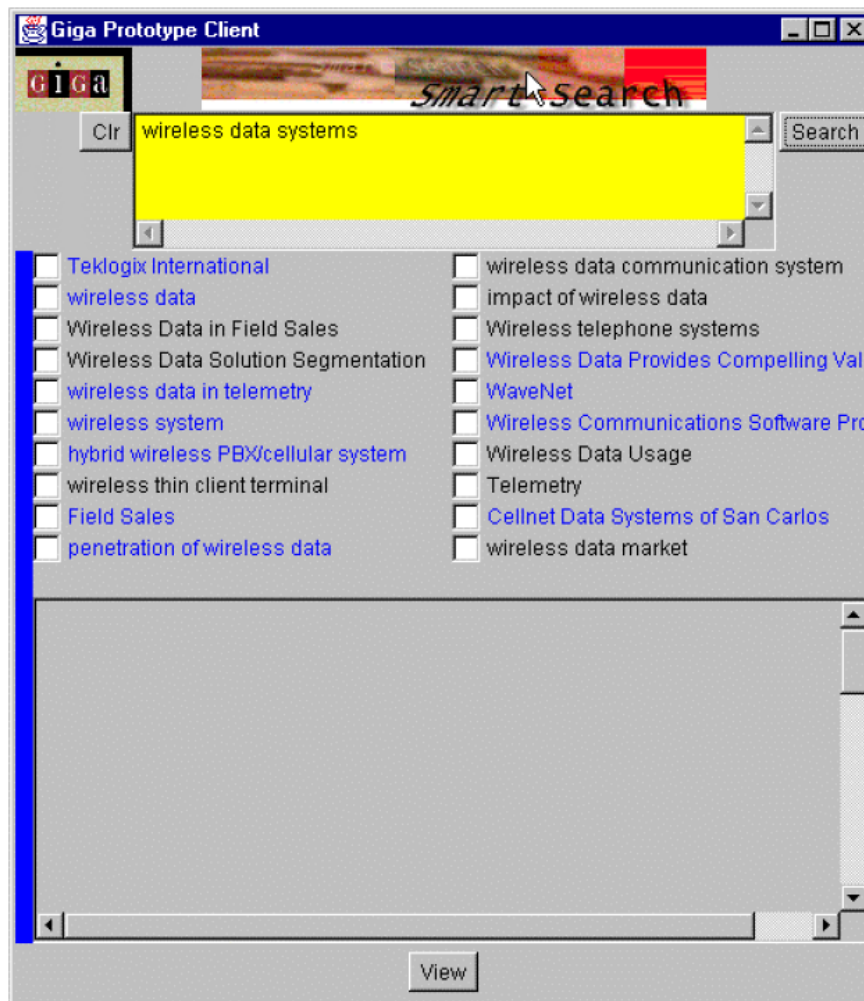


The screenshot shows the 'AutoChoice - Eureka' application window. It features a menu bar with 'File', 'Edit', 'Options', 'Query', and 'Help'. Below the menu is a 'Rank...' button. The main area contains a table of car listings. The table has columns for Manufacturer, Model, Price, Transmission, Cylinders, Horsepower, Weight, 0-60, and Doors. The first row is highlighted in black and contains the number '235' in a green box. The second row is highlighted in blue and contains the number '749'. The third row is highlighted in light blue and contains the number '816'. The fourth row is highlighted in light blue and contains the number '847'. The fifth row is highlighted in light blue and contains the number '863'. The sixth row is highlighted in light blue and contains the number '875'. The seventh row is highlighted in light blue and contains the number '877'. The eighth row is highlighted in light blue and contains the number '924'. The ninth row is highlighted in light blue and contains the number '954'. The tenth row is highlighted in light blue and contains the number '992'.

	Manufacturer	Model	Price	Transmission	Cylinders	Horsepower	Weight	0-60	Doors
235	Acura		\$7,095	Automatic	3	152	1,650	4	2
	Alfa Romeo	100		Manual					
	Audi	190							
	BMW	190E	\$30,769		12	424	5,694	8	4
	Quik	200CV							
749	Chevrolet	CAMARO	\$13,399	Manual	6	160	3,241	6	2
816	Chevrolet	CAMARO	\$13,749	Manual	6	160	3,247	6	2
847	Plymouth		\$13,905	Manual	4	190	2,745	7	2
863	Pontiac	FIREBIRD	\$13,995	Manual	6	160	3,241	6	2
875	Ford	MUSTANG	\$14,070	Manual	8	205	3,035	7	2
877	Dodge		\$14,117	Manual	4	174	3,030	8	2
924	Pontiac	FIREBIRD	\$14,349	Manual	6	160	3,232	6	2
954	Chevrolet	BERETTA	\$14,550	Manual	4	180	2,793	8	2
992	Eagle		\$14,753	Manual	4	190	2,777	7	2

**Figure 6.9 The Eureka interface.**

Query refinement has also been implemented on text document collections that lack any predefined structure. Cooper and Byrd (Cooper and Byrd, 1998) describe a method where a system responds to an initial query by suggesting additional items that could focus the query. These items are found by selecting words and phrases that occur in the documents which are present in the result and that have a high degree of distinctiveness. This selection of items requires pre-constructed indexes on the searchable documents. A screenshot of their example application, OBIWAN, is shown in Figure 6.10.



**Figure 6.10** The OBIWAN system. Items suggest after a query for “wireless data systems”.

Egnor and Lord (Egnor and Lord, 2000) have reported on a system called XYZFind for searching in XML-based databases containing semi-structured data. In their proposed system, a user first issues an unstructured full-text query. The documents that match the user's query are then analysed to find the number of distinct XML-schemas present. After the user has selected the schema of interest, a search form specific to this schema is presented. This allows the user to choose relevant values for each of the attributes in this form. Finally, a fully structured query based on the chosen values is executed and the result is returned to the user.

In order to reduce the size of a result using property value restrictions, the user must be able to interact with the system to select the relevant properties and the preferred values. The systems mentioned above solve this by presenting all available options to the users – giving them total

freedom. This is implemented by the means of lists, sliders, buttons and other graphical widgets that allow individual adjustment of each property value.

This approach is straightforward and well suited as long as the number of dimensions is fairly small. However, as every new dimension requires additional interface components, this approach does in my opinion not scale well. In fact, a high number of dimensions easily results in a complex and confusing interface. Such interfaces might give experienced users no problems, but novice users are likely to suffer.

A somewhat related problem is that these user interfaces tend to require handcrafting by the designer. This makes modifications and adaptations to new databases more unwieldy. Also, as the type and number of dimensions are static, the interface does not take into account the unique properties of a given query result. If every object in a result is red, it is not useful to make it possible for the user to specify colour.

## **6.5 Summary**

---

In this chapter, I have explained my approach. Basically, I try to combat large, unwieldy search results by offering the users a set of questions. If answered, these questions are used to construct filters that remove irrelevant parts of the result.

It is important to note that the questions are constructed using a dynamic analysis of the properties of the objects contained in the result-set. This will ensure that the questions are as relevant as possible as they are based on a run-time analysis of the result rather than being statically determined beforehand.

Traditionally, the effectiveness of retrieval approaches has been measured using precision and recall (Salton, 1989). As my approach only removes irrelevant objects from a result, recall remains constant. On the other hand, precision improves as the relevant objects become more dominant. The precision will therefore improve each time the user answers a question.

The following two chapters present the design, implementation and evaluation of two prototype interfaces based on the approach introduced in this chapter.

---



---

## Chapter 7

# The first SESAM prototype

---

The previous chapter presented the SESAM approach. In this approach, the four design ideas from Chapter 5 have been applied to the domain of textual metadata databases. The purpose is to use SESAM as a vehicle to evaluate these design ideas. In addition, SESAM has two key issues that I am interested in investigating:

- ◆ **Filter suggestions and their impact on usability**

The quality of the suggested filters (questions) is a prime concern. If every question goes unanswered, the approach has failed. The proposed questions must not only make sense to the user, but also lead to an increase of the average precision of the result.

- ◆ **System performance**

The analysis required in order to construct the questions will increase the response time of the query system. If this overhead is too large, users can not be expected to wait and thus the filters will be left unused.

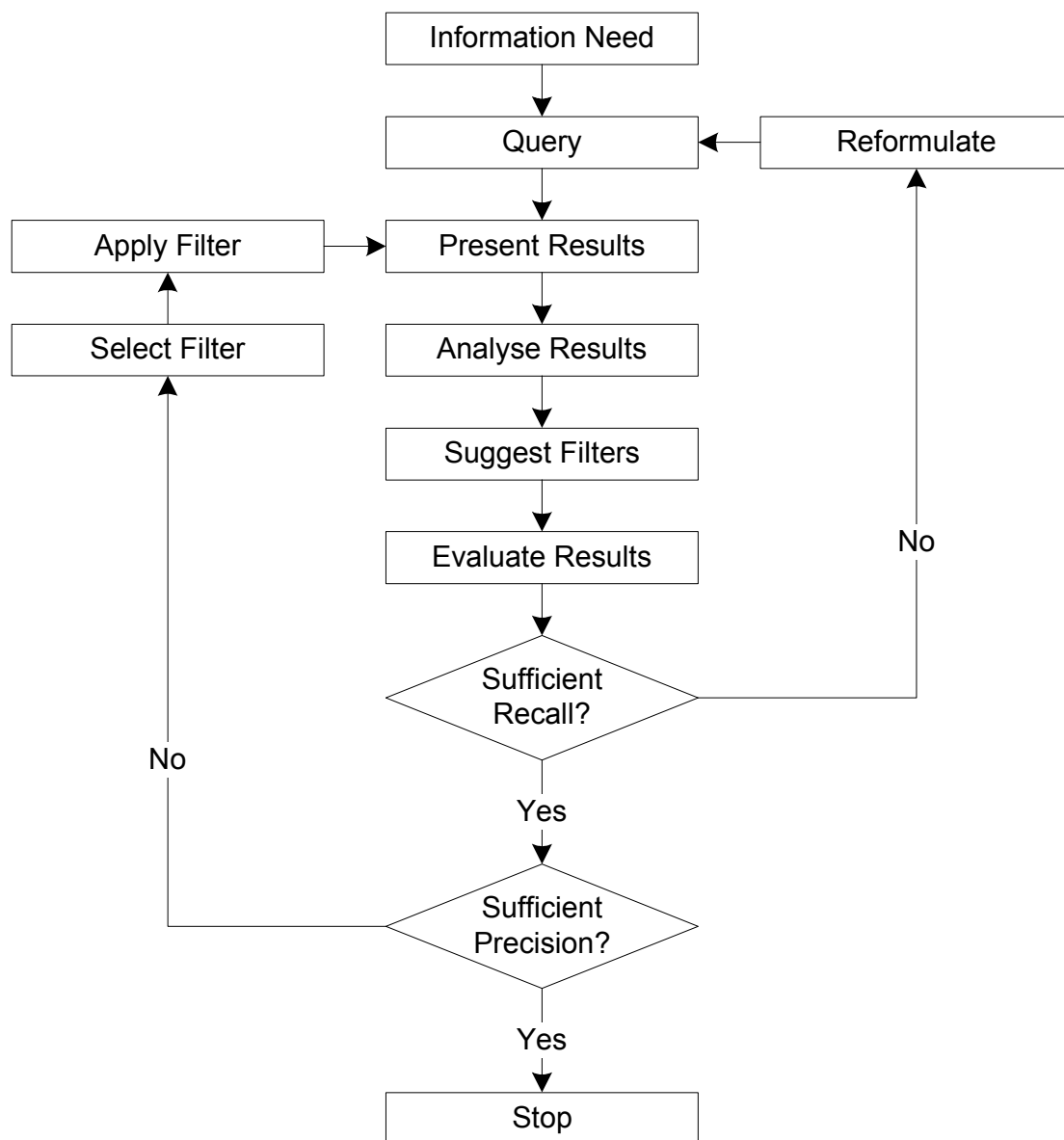
To evaluate the design ideas and investigate these two issues, I have designed and implemented a SESAM prototype interface. This prototype is presented in Section 7.1. In the following sections, I present a usability and a system performance evaluation of this prototype and the results of these evaluations. Finally, in Section 7.5, a discussion of the merits of the prototype and the SESAM approach.

## 7.1 The prototype

In this section, I describe the interaction between the user and the prototype interface (Section 7.1.1), give an overview of a sample session (Section 7.1.2) and finally present a screenshot of the user interface (Section 7.1.3).

### 7.1.1 Interaction model

The interaction model used in SESAM differs from the abstract interaction model presented in Section 5.1 in a few respects. The actual model used by SESAM is shown in Figure 7.1.



**Figure 7.1** SESAM's interaction model.

---

The main difference is that an initial query is required before the result is analysed and filter suggestions presented. This was done mainly for performance reasons. An analysis of the whole Primus database is simply too time consuming to be done dynamically at run-time. The analysis could have been precomputed, but Primus contains too diverse information for filter suggestions to be useful for all information needs. Because when the current result is the whole database, the assumption that the most frequent property values are most relevant, is less likely to hold. Even if the analysis had been precomputed, it is likely that the result-set after the first user-made filter had been applied, still would have been too large for run-time property value analysis.

Filters can only be used to remove objects from the initial result, not add new objects. Therefore only precision improves – recall remains constant at best. Also note that if the user does not find any of the presented filter suggestions to be suitable, he or she will have no choice but to reformulate the initial query or stop the process altogether (not shown in the interaction model). In this case, the SESAM prototype will work similar to the interfaces based on the standard interaction model (see Section 5.1).

### **7.1.2 Overview of a sample search**

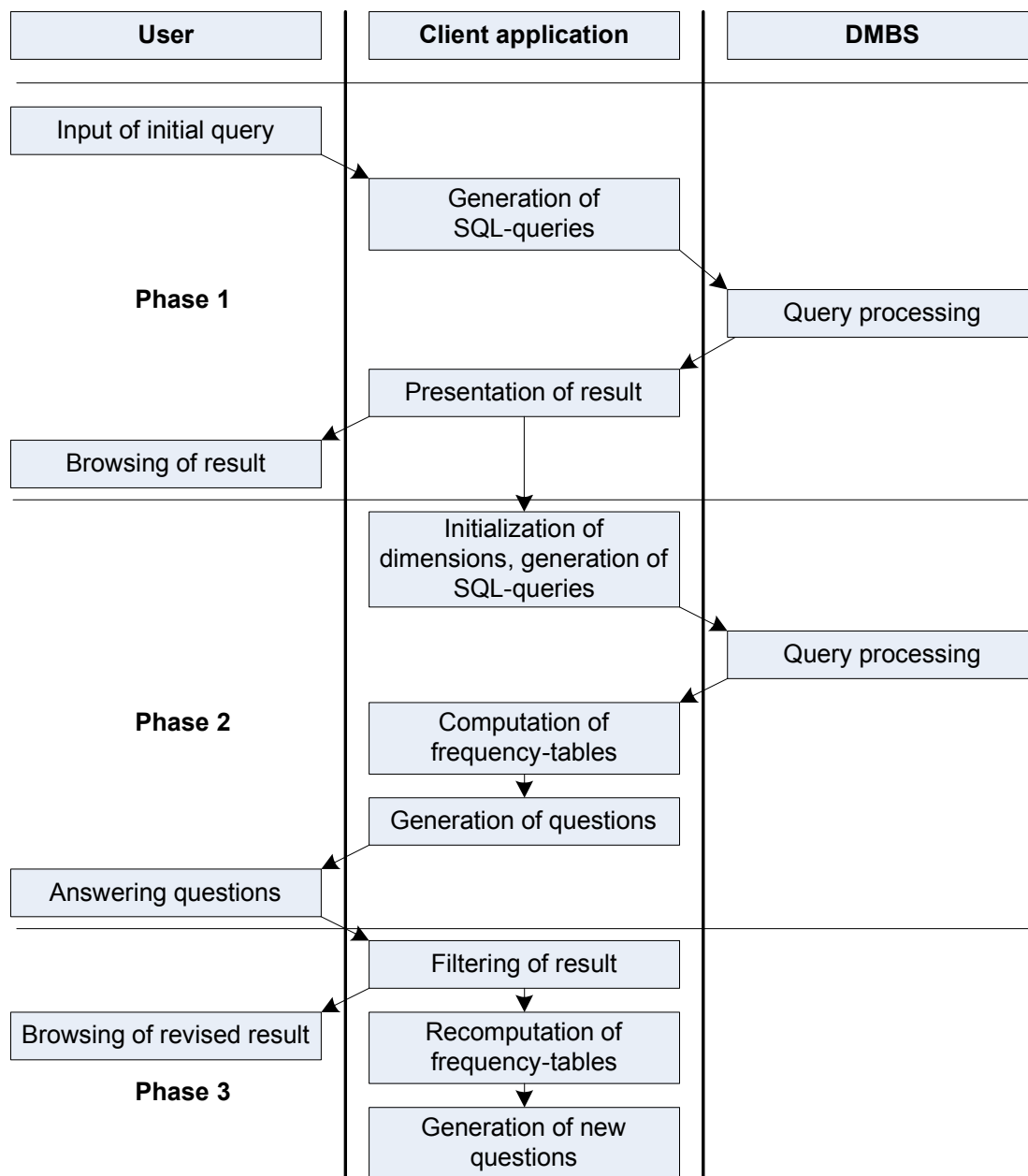
Figure 7.2 illustrates the three main phases of a typical search using the SESAM prototype. In the first phase, the initial query is processed at the database server. The result received by the client is then presented to the user.

The second phase concerns the generation of questions (filters). This requires the prototype to contact the database server in order to retrieve the metadata used to build frequency-tables for each of the chosen dimensions. The prototype then processes these tables to generate the questions. During this phase, the user is free to browse the result of the initial search.

The third and final phase starts when the user has answered one or more of the questions. As previously described, each of the questions corresponds to a particular dimension. Thus, each answer is a selection of one (or several) of the partitions in a dimension. The program can therefore make a filter that removes the objects contained in the unwanted partitions. The updated result is in turn used to generate new questions – thereby repeating the last two phases. However, the database does not

---

need to be contacted as the previously made frequency-tables can be updated at the client. This makes further generation of questions almost instantaneous.



**Figure 7.2 Phases in a typical search.**

It is important to note that this approach is very non-intrusive with regard to the database. The only modifications made to the Primus database were to generate a few extra indexes, in order to facilitate text search and to improve response time. However, changing the data model might be necessary to further improve performance. As object properties need to be gathered for every selected dimension (as described in

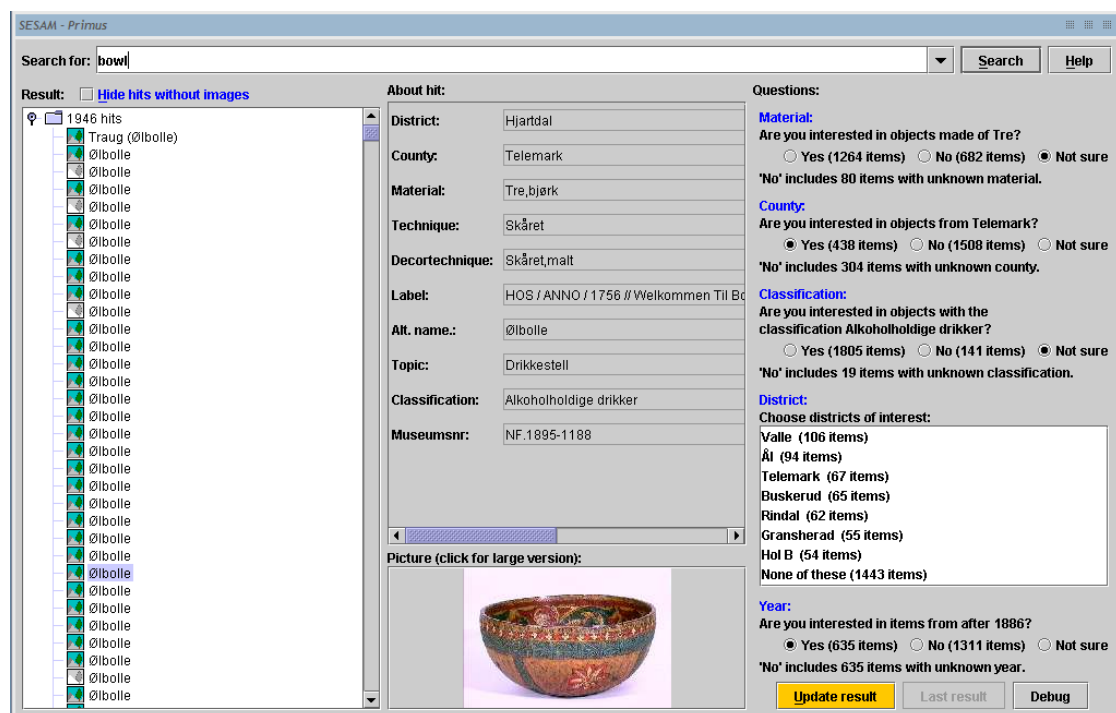


Section 6.2.1), it is important that this can be done quickly. In essence, this limits the number of relational joins that are acceptable when collecting the properties of objects in the result.

### 7.1.3 User interface

The user interface of the SESAM prototype is shown in Figure 7.3. The prototype was written in Java and uses JDBC to send SQL queries to an Oracle 8i DBMS<sup>1</sup>.

The top part contains an input-box for the query and a “Search”-button used to start the query processing. When the query is completed, the result is shown to the user in the left-hand list. The middle part of the interface is used to present details of the object that the user has selected.



**Figure 7.3** The interface of the first prototype.

While the result is shown, a new set of queries is sent to the database to construct the frequency-tables needed to make the questions. When this process is completed, the generated questions appear in the right-hand part of the interface.

1. Oracle Corp., <http://www.oracle.com/>

After the user has answered some or all of the questions, pressing the “Update result”-button will filter the result based upon the answers given. The question-making process is then repeated in order to generate a new set of questions. This is almost instantaneous as no database communication is necessary. New questions are generated until either the user is satisfied with the result, the prototype is unable to find suitable questions, or the last set of presented questions are left unanswered.

## **7.2 Usability evaluation**

---

The design of applications, and user interfaces in particular, is not an exact science where quality can be objectively defined. The subjective evaluation by a group of prospective users is therefore one of the best ways to evaluate the usability of an interface. This section presents the result of a usability evaluation carried out on the SESAM prototype interface. An overview of usability evaluation in general was presented in Chapter 3.

The purpose of this usability evaluation is twofold. First, I want to find the strengths and weaknesses of the SESAM approach compared to other interfaces for accessing information. The evaluation is also performed in order to gain an understanding of how users perceive SESAM. Information about usability problems, program bugs, missing features etc. is important in order to locate possible areas for improvement.

As the purpose of the evaluation is to find the strengths and weaknesses of the SESAM approach, a comparative evaluation with other interfaces for information search was a natural choice. By comparing the SESAM prototype interface with other interfaces, one can gain a clearer view of the merits and the weaknesses of the design. Such evaluations also help to improve the reliability of the study by limiting the impact of the Hawthorne effect which states that simply showing concern for users' situation improves their performance (see Section 3.5.1).

Two other interfaces were selected for use in the evaluation – one based on dynamic query and one using forms. These two interfaces are presented in the following sections. Dynamic query was selected because it, similar to SESAM, focuses on using attribute restrictions to remove objects from a result. The widespread use of forms makes it useful to include as a reference.

---

Rather than to rely on existing implementations, I decided to make my own implementations of both dynamic query and forms. This was done in order to use the Primus database (see Section 6.1) in the whole evaluation. By using the same database for all tested interfaces, it should be easier to focus on interface differences.

### 7.2.1 An interface based on dynamic query

Dynamic query, as a concept for information searching interfaces, has been described previously in Section 4.2.6. Dynamic query is an attempt to apply the concept of direct manipulation to query interfaces. This is done by making it possible for users to specify attribute restrictions using graphical widgets and have the result continuously updated. Several evaluations have shown that dynamic query is a very powerful technique (see for example (Ahlberg et al., 1992)). This, combined with the similarities between the dynamic query approach and SESAM (see Section 6.4), made dynamic query a natural choice for usability comparisons.

A screenshot of the dynamic query implementation used in this evaluation is shown in Figure 7.4.

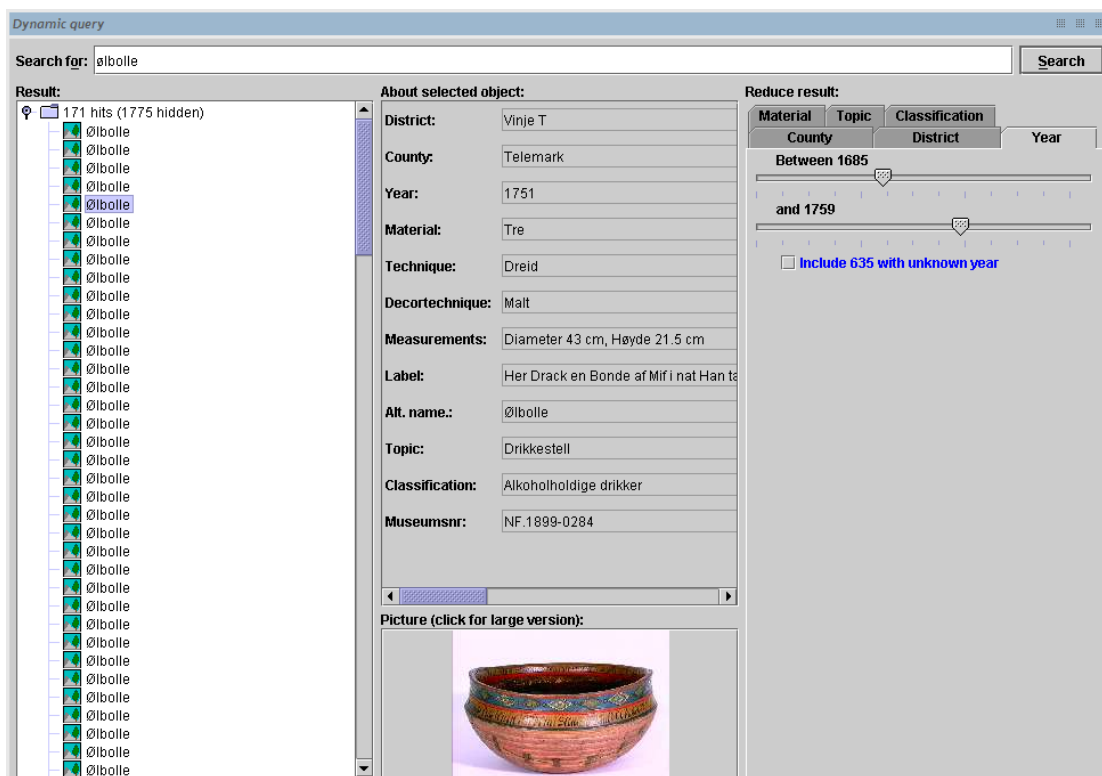


Figure 7.4 Dynamic query interface.

As the Primus database contains more than 400.000 objects, using dynamic query directly on the database contents would have been very difficult without seriously reducing the update frequency. The implementation therefore uses a standard textual query to get the initial result which then can be filtered by direct manipulation of attribute restrictions. This textual query part was copied directly from the SESAM implementation.

After the initial result is presented, the user can modify this result by applying property value restrictions to six separate properties. These six properties are the same as those used by the SESAM prototype. Five of them are textual properties such as county of origin and material. In order to place restrictions, the user first selects the corresponding tab (shown in the right part of Figure 7.4). Textual property values are displayed in a list where the user can toggle each value on or off by clicking it. Only the values present in the result are displayed. Each time the user alters the selection, the result is updated accordingly.

The sixth property is year of origin. As this property is numeric, two sets of sliders are used for restrictions. This is shown in Figure 7.4. Combined, these two sliders let the user select starting and ending year of interest. Unfortunately, a lot of null values exists in the database. Therefore, a separate check box lets the user decide if objects with null values for year of origin should be included in the result or not. Again, the result is updated constantly as the user drags the slider bars.

In addition, the interface contains a component for presentation of the result and a component for displaying details about the selected object. Both these are exactly the same interface components as used in the SESAM prototype interface.

The key difference between SESAM and dynamic query, is that the properties used in the dynamic query interface are constant and that all property values are always displayed. SESAM, on the other hand, tries to select the most suitable subset of both properties and property values.

### **7.2.2 A forms-based interface**

Forms is one of the most popular types of interfaces for queries (see Section 2.3.3 and Section 4.2.5). Typically, such interfaces consist of one or more textual fields where the user can enter query terms. The result is the set of objects that conform to these restrictions.

---

As for dynamic query, a specific forms implementation for the Primus database was made. A screenshot is shown in Figure 7.5.

The screenshot shows a web-based search interface titled "Forms based search". At the top, there is a section "Enter query terms:" with several input fields: "Name:" (containing "bowl"), "Topic:", "Classification:", "Material:" (containing "wood"), "County:", "District:", "Dated before (year):" (containing "1700"), and "Dated after (year):". Below these fields are "Search" and "Reset" buttons. The main area is divided into two parts. On the left, under "Result:", there is a list of 70 hits, each represented by a small thumbnail image and the text "Ølbolle". On the right, under "About selected object:", there is a detailed view of a selected item with the following information: "District: Hemsedal", "County: Buskerud", "Year: 1608", "Material: Tre, bøk", "Technique: Dreid", "Decortechnique: Malt", "Measurements: Diameter 405 cm, Høyde 12 cm", "Label: Drik Skaalen ud Tænk allid Paa din", "Alt. name.: Bolle, Ølboll", and "Topic: Drikkestell". At the bottom right, there is a section titled "Picture (click for large version):" which contains a photograph of a wooden bowl.

**Figure 7.5 Forms based interface.**

The top part of the interface contains the text fields constituting the form. Each text field corresponds to a property attribute in the database and the properties used are the same as in the two other prototypes. In order to perform a query, the user enters query terms into one or more of the text fields and presses "Search". The result is then displayed in the lower part of the interface.

This interface shares the interface components used to display the result as a whole and information about the selected object, with SESAM and dynamic query. This was done as these components are irrelevant with regards to what I want to evaluate.

### **7.2.3 Evaluation methods**

As various evaluation methods have different strong points, it is necessary first to define the purpose of the evaluation. As briefly mentioned above, the purpose of this evaluation is twofold:

1. Examine how the SESAM prototype interface performs compared to the two other implemented interfaces.
2. Gain an understanding of how users operate the interface and use this information as a basis for determining possible areas for improvement of the SESAM approach.

With these objectives and the description of evaluation methods from Section 3.4 as a basis, I decided to use the following evaluation methods:

- ◆ Pilot study: Inspection
- ◆ Main study: Performance measurement
- ◆ Main study: Observation
- ◆ Main study: Questionnaire

Each of these are described below.

#### **Pilot study: Inspection**

As none of the three implementations had seen much use, it was natural to expect that several interface quirks remained. In order not to have the evaluation of the interface designs tainted by poor implementations, a pilot study was carried out before the main study. This study consisted of having three fellow researchers use inspection methods to comment on the usability of each of the interfaces. Several iterations of implementation and evaluation were done before the interfaces had reached a level of consistency and quality considered suitable for the main study.

The main change made to the SESAM interface was to make an interface consisting of a single window as shown in Figure 7.3. Before the pilot study, the presentation of the result and the generated question were in

---

two different windows. This required the users to switch back and forth between them and was soon identified as a source for confusion as well as being cumbersome and slow to use.

### **Main study: Performance measurement**

Quantitative data is often used to compare interfaces for information search (see Section 3.4.1). As evidenced by this method's widespread use, finding suitable metrics is often not too difficult. In this study, the time taken by each test subject to carry out individual tasks in each interface was recorded.

The evaluation subjects were ten master degree students in computer science. Everyone therefore had extensive background knowledge of computers in general, but no one had any previous experience with the test database.

A randomized within-subject design was used – each test subject tested every interface in random order. To make it possible to observe their initial response and to make a more realistic scenario for web based interfaces, the test subjects were given no initial training.

Unfortunately, not training the participants prior to measuring their performance makes the collected data less reliable. This is due to the practice effect – subjects typically will perform better as the test goes on. But as I was more interested in observing the immediate reaction to and comprehension of the interface, training was not included in the study.

### **Main study: Observation**

As established in Section 3.2.2, analysing qualitative data is an important way of gaining an understanding of how an interface is perceived. Due to the relatively small size of the test group, direct observation of interaction was a feasible method to gain qualitative data. Each of the subjects was observed by myself during the execution of the tasks described below, and observational data was recorded on paper. These notes were supplemented by a short informal interview afterwards to clarify some of the observations made.

### **Main study: Questionnaire**

Questionnaires was used to complement observational data and to make a quantitative comparison possible. The questions used were taken from the QUIS – Questionnaire for User Interaction Satisfaction (Chin et al., 1988). Because only some questions were deemed relevant, and to limit

---

the amount of effort required by the test subjects, only a subset of this comprehensive questionnaire was used. In order to make comparison possible, the users answered the same questions for all three user interfaces.

### **7.2.4 Evaluation tasks**

As noted in Chapter 4, user interfaces for accessing information should handle more than just exact queries – uncertainties and shifting information needs should be taken into account. In order for the study to capture the different access modes, the evaluation tasks were split into three different categories.

- ◆ **Exact queries which give manageable results**

By manageable results I refer to results that are so small that they can be browsed without any assistance from the program (i.e. by manually accessing each result object). In this study, results with less than 100 objects were deemed manageable. Each test subject was given three tasks of this type to perform with each of the interfaces.

- ◆ **Exact queries which give unwieldy large results**

Means of reducing the size of a result to improve precision are focal points of both SESAM and dynamic query. It is therefore natural to test these facilities by using queries that give results of several hundreds or even thousands of objects. The test subjects were asked to use the interfaces to reduce the size of the result to a size which they deemed manageable. Three tasks for each interface were given.

- ◆ **Browsing and navigation**

The freeform nature of browsing makes it difficult to find suitable tasks for evaluation purposes. The test subjects were therefore given a general description of the content of the database and instructed to spend a few moments exploring the database using each of the interfaces. As this task had no clearly defined goal, it was impossible to use it for performance measurements.

The full details of the evaluation tasks are presented in Appendix A.1, while the handout given to the test subjects in advance is presented in Appendix C.1.

---



## 7.3 Usability evaluation results

The purpose of this usability evaluation was to identify strengths and weaknesses of the SESAM approach compared with two other approaches and to gain an understanding of how the SESAM prototype interface was perceived by the test subjects.

To address these issues, I first present the results of the performance measurements (Section 7.3.1), the observations (Section 7.3.2) and the questionnaires (Section 7.3.3). These results are then used as a basis for a broader discussion in Section 7.3.4.

### 7.3.1 User performance measurements

The results of the performance measurement are summarized in Table 7.1 below. For full details, see Appendix A.1.

	Forms	Dynamic Query	SESAM	Overall
<b>Tested first</b>	48,2	53,5	53,0	51,2
<b>Tested second</b>	35,1	26,1	33,9	31,9
<b>Tested third</b>	43,6	22,0	33,4	31,9
<b>Overall</b>	42,9	32,7	39,5	38,4

**Table 7.1 Results of performance measurements (in seconds).**

As the test subjects were not given any training before the evaluation started, it is reasonable to expect practice effects – i.e. that subjects on average will perform better testing the third interface than testing the first. To highlight the effect of practice, the table lists separate scores for those that tested a given interface first, second or third. For example, test subjects that tested dynamic query as the second interface in their testing, spent on average 26,1 seconds on each task.

The performance measurement part of the study was plagued by several reliability and validity problems which limited the usefulness of the recorded data. The practice effect has already been mentioned. Further, many of the test subjects spent longer time to complete the tasks than strictly necessary. Especially with dynamic query and SESAM, exploring intermediate results or the interface in general was fairly common. As these detours provided interesting observational data, subjects were not instructed to “stay on course”. Finally, for the series of tasks that required the users to handle unwieldy large results, it was difficult to determine exactly when the task was completed. Combined, this makes the reliability of the recorded data questionable.

However, some observations can still be made with a high degree of certainty. The effect of practice is perhaps the most evident – overall the times were almost cut in half from the first interface tested to the last two. It is also apparent that this effect was least profound with forms. This can be attributed to the fact that this interface was the least complex and also familiar to the test subjects due to its widespread use.

### **7.3.2 Results from observation**

Most of the information gathered by observing the ten users was fairly low level – placement of buttons, clarity of screen prompts etc. By collating these observations, I have extracted four high level observations.

#### **1: High complexity limits usability**

Even if it is obvious that complex interfaces are less usable than simple interfaces, it is sometimes difficult for designers to spot complexity in self-designed interfaces. Perhaps the most interesting observation is therefore to find where and why users stumble and get confused.

It came as a bit of a surprise that the amount of text used to display the questions generated by the SESAM prototype was a problem. Several users complained that it took too long to read each question and that it made the interface too complex. The idea of using text to offer a natural dialogue therefore proved less effective than hoped.

Another complaint raised by several of the test subjects was that the different questions in SESAM should have been displayed in the same order each time. As it was, the questions were ranked by utility (see Section 7.1) and displayed with the highest ranking question on top. It was therefore possible to have a question about classification on top for one result and on bottom the next time. This made it almost necessary to always read every question and thus made the SESAM prototype slower to use.

The dynamic query interface also received some complaints with regards to complexity. In order to offer the users every possible option for making restrictions, long lists and several tab panels were required. Some users commented that they were forced to spend time searching the interface for the option they wanted. Others clearly felt bewildered by the sheer number of available options.

---

## **2: Actions should be easily reversible**

This observation should not come as a big surprise as it matches one of Shneiderman's eight golden rules of interface design (see Section 2.2). As could be expected, the dynamic query interface proved to be the best interface in this respect. Users enjoyed trying different options as they quickly realised that actions easily could be reversed. This was evident both from the initial reaction to the interface and from observing the browsing evaluation tasks. The forms-based interface was on the other end of the spectrum in this regard. Each result presented by the interface is final – it is impossible to make any modifications. The test subjects were therefore forced to change the query expression rather than modify the result directly. As the consequences of a query rewrite are notoriously difficult to predict, the users typically issued many ineffective queries (for example queries giving empty results).

Even though SESAM had an “undo”-button, it was not used nearly as much as similar functionality in the dynamic query interface. Observations indicate that the placement of this button (bottom right) was very awkward. SESAM's undo functionality was also much slower and limited in the number of undo levels. Interestingly, one user commented that he looked for a “back”-button (similar to web browser).

## **3: Number and appropriateness of controls**

The interface that offered most possibilities for control, dynamic query, was generally agreed to be the most powerful interface. As soon as users had figured the interface out, they became pretty apt at using it to get what they wanted. Similarly, forms offered few options and users were frequently bewildered as to how they could achieve what they wanted. Most users clearly felt hindered by this interface.

SESAM took up some sort of middle ground. It presented only a few ways to reduce the size of a result and improve precision. When the “right” questions were presented, the interface worked great and no one complained. However, users often did not find a suitable question. This soon became annoying – offering a series of irrelevant choices made users lose faith in the SESAM prototype interface in general.

By observing the evaluation tasks related to browsing, it was noticeable that an intermediate number of controls better afforded exploration of both the interface and the underlying database. Too few controls give few possibilities for interaction, while too many controls results in a complex interface.

---

#### **4: Type of task affects preference**

By observing the test subjects during the execution of the different evaluation tasks, it was evident that each interface was not equally suited for all types of tasks. As could be expected, forms was nice for exact queries – at least as long as the result was of manageable size. Precise queries could be constructed if the user knew which restrictions to enter. On the other hand, several users had problems locating the correct field for a given restriction. By having no support for handling large results beyond manual browsing, forms was ill suited for freeform navigation. More often than for the other interfaces, users became stuck when they were asked to explore the database on their own.

Almost the exact opposite was true for SESAM. When users knew exactly which restriction on which attribute they were interested in (for example items from the year 1850), SESAM was ill suited. Several commented that they would have liked to be able to edit the questions they were asked by the interface. On the other hand, this was not a problem when their goal was not specific. In these cases, the ease by which the result could be modified afforded exploration. For some users, the novelty of the questions made them answer a question just to see which questions would appear next.

Dynamic query offered a little of both. Specific restrictions could be easily expressed, even more so than for forms. It was also possible to use the interface for exploration even if the number of available options sometimes was a bit bewildering.

### **7.3.3 Results from questionnaire**

The questionnaire consisted of 15 questions drawn from QUIS (Chin et al., 1988) for each of the interfaces, 45 in total. For each question, the users could select their response on a 9-point scale (1-9).

The first three questions were assessments of the interface's ability with respect to each of the three evaluation tasks presented in Section 7.2.4. The fourth was an overall assessment, while the latter 11 questions have been collated into the two following categories to simplify the presentation of the results:

---

---

**Visuals:**

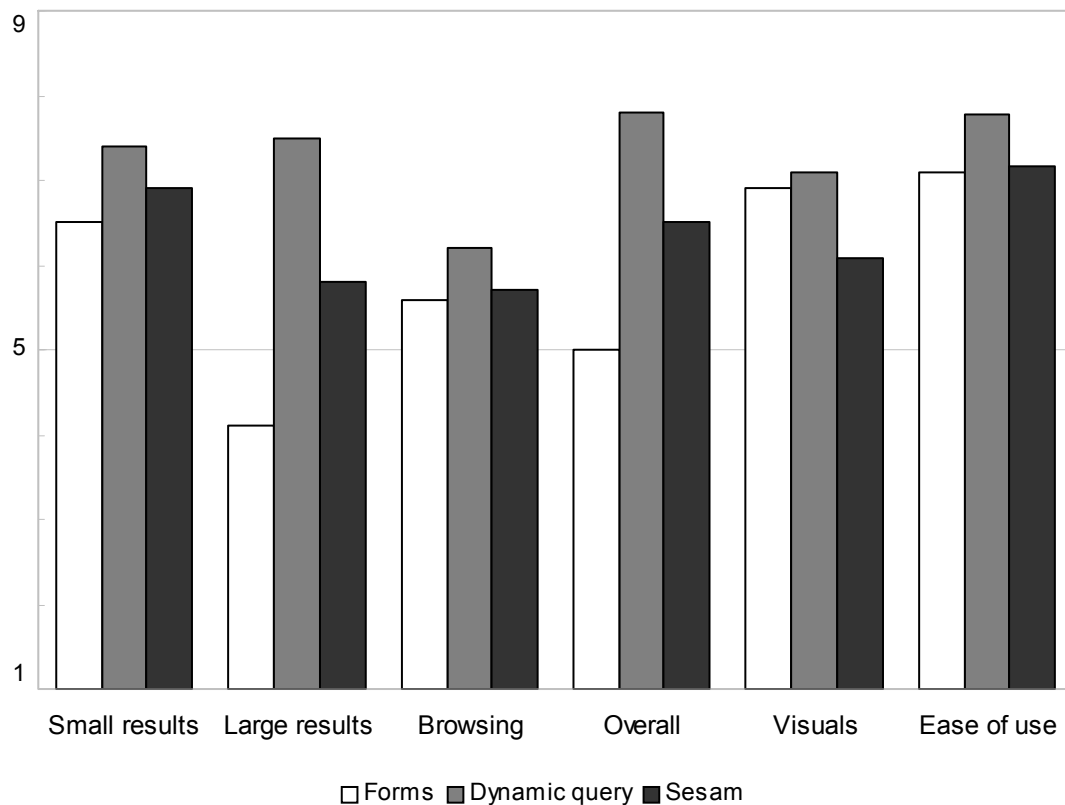
- ◆ Screen layouts were helpful (1:never – 9:always)
- ◆ Amount of information that can be displayed on screen (1:inadequate – 9:adequate)
- ◆ Arrangement of information on screen (1:illogical – 9:logical)
- ◆ Messages which appear on screen (1:confusing – 9:clear)
- ◆ Computer keeps you informed about what it is doing (1:never – 9:always)

**Ease of use:**

- ◆ Learning to operate the system (1:difficult – 9:easy)
- ◆ Getting started (1:difficult – 9:easy)
- ◆ Time to learn to use the system (1:slow – 9:fast)
- ◆ Exploration of features by trial and error (1:discouraging – 9:encouraging)
- ◆ Exploration of features (1:risky – 9:safe)
- ◆ System speed (1:too slow – 9:fast enough)

An overview of the results is presented in Figure 7.6 while the complete results can be found in Appendix A.1.

---



**Figure 7.6 Results from questionnaires.**

Interpreting these results, I consider the following three findings to be the most notable:

◆ **Overall ranking**

The overall picture is pretty clear – in general, the test subjects favoured dynamic query and found forms lacking. With one exception (SESAM’s visuals as discussed below), this holds for every category.

◆ **SESAM impeded by poor visuals**

With regard to visuals, SESAM comes out quite inferior to the other two interfaces. This indicates that the quality of the implementation might be the limiting factor in the test subject’s overall feelings towards SESAM. This also corresponds with the observational data.

◆ **Validity problems with evaluation of browsing capabilities**

Perhaps the single most surprising result is that all three interfaces were deemed almost equally suited for browsing. As forms

---

offer no real support for browsing, I attribute this result to validity problems with the evaluation rather than as a statement of ease of using forms for browsing. This assumption was supported by the observational data. Validity problems with the study are further described below.

### 7.3.4 Discussion

Forms as a query interface have been around for a long time. Unsurprisingly, it therefore proved easy to learn and use. However, the lack of any support for modifying results and for browsing was clearly apparent – at least as far as observational data was concerned. The results from the questionnaires in fact indicate that forms' browsing support was comparable to the other two interfaces. But this anomaly was most probably due to problems with the evaluation design.

Dynamic query performed best in almost every respect. Even if I had to take some liberties with my implementation (most importantly by introducing an initial query), the main ideas carried through and were appreciated by the testers. Easy handling of large results, reversible actions and using direct manipulation to apply property value restrictions proved a successful combination. However, some concerns with the complexity of the interface were raised. By presenting the user with every possible option for restricting the size of the interface, some users complained (directly or indirectly) about a cognitive overload. It may be that this approach, as expected, is not too scalable with regards to number of properties for restriction and the number of values for each property. Unfortunately, the number of properties used in SESAM and dynamic query was a bit too low to really highlight the differences between them.

SESAM differs from dynamic query in that it presents the users with only a subset of possible property value restrictions for reducing the result size and improving its precision. Unfortunately, shortcomings in the implementation limited the chances for a real assessment of this concept. Almost every test subject complained about too much text on screen, interface components switching places etc. Evaluation results indicate that SESAM is better suited for navigation and browsing than for handling exact queries. This is understandable as the process of programmatically deciding the most promising subset of options will not give a subset that matches everyone's needs.

---

## Reliability and validity

The usability evaluation presented in this chapter was my first real-world evaluation. It is therefore only reasonable to expect room for improvements. In this section I present some concerns regarding reliability and validity that appeared before, during and after the evaluation had taken place. A general introduction to such issues was given in Section 3.5.

Before the evaluation started, I decided to let the users start without having a separate training session first. This in contrast to most comparable evaluations such as (Ahlberg et al., 1992), (Hearst and Pedersen, 1996), (Hibino and Rundensteiner, 1997) and (Tanin et al., 2000). While this reduced the reliability and validity of the performance measurements, I wanted to observe the initial reaction to each of the interfaces. In this manner, the observation and performance measurements had conflicting requirements. I could have extended the study to have a separate session first where the subject could familiarize themselves with the interfaces while being observed, but the test already took about an hour so this idea was abandoned.

Observations are always susceptible to bias in the observer. Observing and interpreting what one wants to observe rather than what really happens is a well known issue. In this evaluation, I had bias towards the SESAM prototype interface as it was based on my own approach. Involving external observers or interpreters might therefore have been a good idea, but as the objective was for *me* to learn how the interfaces operated, this is not a clear-cut case.

During the planning of the evaluation, it became clear that it is difficult to design tasks that highlight the ability of an interface to handle large results and support browsing. Some anomalies in the test results indicate that at least the browsing tasks could have been better designed. Similarly, the lack of a clear definition of when the subjects had completed some of the tasks, made it difficult to record reliable performance data. On the other hand, it is not guaranteed that performance measurements really give meaningful information in all cases. This is especially true for browsing and freeform navigation where it is not easy to tell when the objective has been reached and the information need fulfilled.

The choice of test subjects is a clear candidate for improvement. It is not uncommon to use university students due to easy access, but they might not be representable for the intended audience of the interfaces. It would certainly have been better, but far more time consuming, to

---



include several types of test subjects in order to evaluate whether interfaces are perceived different according to background and interests. It would also have been better to have used counterbalanced within-subjects tests rather than randomized. By increasing the number of test subjects to 12 (or any other multiples of 3), one would have eliminated any reliability concerns due to order effects. As it is, four persons tested forms as their first interface while only three tested dynamic query and SESAM first. This is perhaps the single most obvious mistake with this usability evaluation.

## **7.4 System performance evaluation**

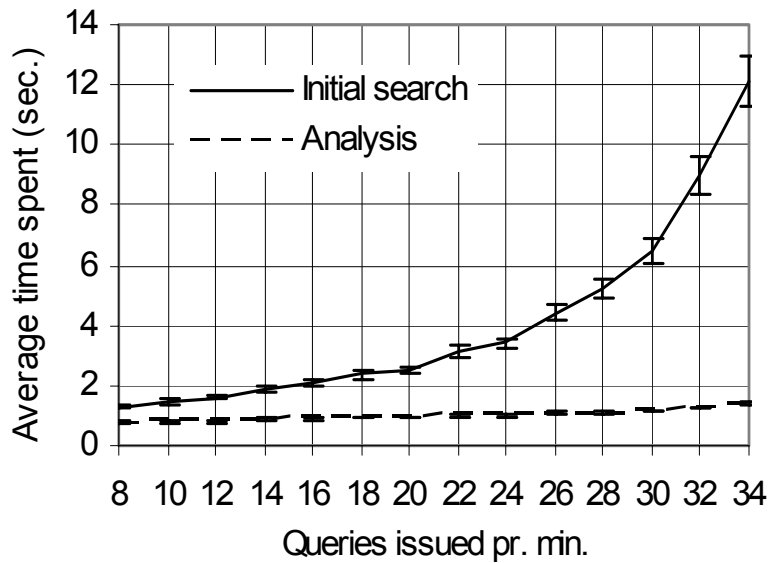
---

Not only usability is of interest when the validity of the approach presented in Chapter 6 is to be examined. As discussed back in Section 2.2, responsitivity is an important factor – perhaps even more so for interfaces for accessing information. This section therefore describes the result of a performance evaluation where the aim was to examine if the extra processing required to generate the questions, is too time-consuming.

To evaluate the performance of the prototype, two museum conservators helped me to make a list of 180 typical search expressions where each on average would give a result of about 700 objects. The prototype was modified to randomly draw words from this list and perform initial search and generation of questions (analysis), logging the time spent on each. The database server used had a Pentium Pro 200 MHz processor and 256 MB RAM.

The individual queries were organized into series with arrival rates of 8 to 34 queries pr. minute. Each series was executed until the average response-time had stabilized – something that took between 6.000 and 50.000 queries. The exact number depended on an analysis of the size of the 95 % confidence interval. The results are shown in Figure 7.7 with error bars indicating the confidence intervals.

---



**Figure 7.7 Time spent on initial search and analysis as a function of load.**

The graph shows that while the total response time increases about 50 % at light load due to the analysis, this value is 10 to 20 % at higher and probably more realistic loads. This indicates that the analysis can be made without compromising the response time. This conclusion is further supported by the fact that users can browse the initial result while the analysis is performed. Thereby the actual time users spend idle, waiting for the questions, is further shortened. Another helpful feature with regard to the performance, is that the database is only contacted for the first set of questions in a particular search. Thus, the following sets of questions can typically be made in less than 500 milliseconds. Other performance tests showed that time spent on analysis, not surprisingly, is linearly correlated to the size of the initial result.

It is important to note that these tests show how the SESAM prototype performs with regard to a specific database. As the response-time is dependent on the complexity of the data model and the size of the database, other databases might give slightly different results. Still, as the test database (as described in Section 7.2) is both fairly large and complex, I find the results to be a clear indication that the extra processing required does not invalidate the approach.

---

## 7.5 Summary

---

Chapter 6 introduced the SESAM approach. In order to test the usefulness of this approach, I in this chapter presented a SESAM prototype interface as well as a usability and a performance evaluation of this prototype. The usability evaluation consisted of a comparative evaluation of three interfaces: the SESAM prototype, an interface based on dynamic query and a forms-based interface. The purpose of this evaluation was twofold:

1. Examine how the SESAM prototype performs compared to the two other implemented interfaces.
2. Gain an understanding of how users operate the interface and use this information as a basis for determining possible areas for improvement.

Overall, results from the usability evaluation indicate that SESAM performs better than forms but worse than dynamic query. This was especially evident in the handling of exact queries which give large result-sets.

SESAM's poor performance compared with dynamic query can at least in part be explained by the somewhat lacking implementation. The SESAM prototype scored low with regards to visuals and several usability problems were identified. On the other hand, it was found to work well for browsing and freeform navigation. The results from the performance evaluation indicate that the extra processing required, does not seriously affect the response time.

While these results in general were somewhat disappointing, the implementation shortcomings may very well taint the results so that they give little real evidence concerning the merits of the approach. The usability evaluation presented in this chapter also had several reliability and validity concerns that probably have affected the results. By using the lessons learned to refine SESAM and enhance the SESAM prototype, a new and improved evaluation could paint a different and more accurate picture.

---



---

## Chapter 8

# The second SESAM prototype

---

In the previous chapter, I presented the first SESAM prototype. The evaluations of this prototype incited me to continue the development of SESAM in order to make a more reliable evaluation of the design ideas presented in Chapter 5. Three different areas where the design, implementation and evaluation of a second prototype could be improved while keeping the core concept intact, were identified:

First of all, experience with the first prototype made me realize that it could be helpful to focus more on supporting different strategies for information search (see Section 4.1). The different tasks assigned to the test subjects in the first evaluation were also in hindsight not sufficiently representative for the different strategies the interface was meant to support.

Second, the first evaluation spawned several ideas for improving SESAM. Qualitative data gathered by observation identified a number of usability issues that should be addressed. By continuing to improve the quality of the implementation, test subjects will be less likely to get hung up on implementation issues irrelevant to the overall approach. This will make it easier to evaluate the design ideas on which the SESAM approach is based.

Third, the first usability evaluation had a number of reliability and validity concerns that made it difficult to reach clear conclusions. On the basis of the lessons learned in that evaluation, it should be possible to get more trustworthy results from a new evaluation.

Section 8.1 details of the design changes made in the second prototype. The implementation is presented in Section 8.2, while Section 8.3 describes the new evaluation. Finally, Section 8.4 concludes the part of the thesis concerning accessing information in textual metadata databases (Chapters 6, 7 and 8).

## **8.1 Design**

---

Based on the results of the evaluation of the first prototype, the following overall objectives for the new prototype were identified:

### **System related:**

- ◆ **Keep the core concepts**

The objective of SESAM continues to be improving precision using intra-result operations based on a dynamic analysis of the properties of the objects in a result set.

- ◆ **More freedom of choice**

The new prototype should make more options for property-value restrictions available. As it was, users felt too restricted. Improved support for different strategies is therefore required.

- ◆ **Improve utility functions**

The indirect feedback from the evaluation of the first prototype suggests that the utility functions should be reworked. For example, the evaluation of list-based questions for categorical data should take into account how many of the property values are not presented to the user. Also, allowing more freedom of choice will necessitate a modification of the selection process.

### **User-interface related:**

- ◆ **Improved quality of implementation**

The old prototype had too much text to read. Also, several large components in the interface often changed appearance and this frequently led to bewilderment.

- ◆ **Easier and faster to undo actions**

To afford exploration, it is very important that the users feel they can try out operations without fear of never being able to undo them if things go wrong.

- ◆ **Make it possible to follow links**

When reading the detailed description of a given information

---

---

object, it should be possible to use the displayed property values to modify the result – for example to find objects with similar property values.

Two of these objectives require closer examination before I turn to implementation issues. As a part of the desire to improve freedom of choice, the new prototype should focus more on supporting different strategies for information access. This is addressed in Section 8.1.1. The process of finding suitable filter suggestions is discussed in Section 8.1.2. As this process is a key component of the SESAM approach, it is natural to use the experiences from the first prototype and on this basis reconsider the selection process.

### **8.1.1 Different strategies for information access**

In Section 4.1, I described different strategies for accessing information in information repositories. Based on that discussion, my own experience, and the observations made in the first evaluation, I have identified the following five different searching tasks that SESAM should support:

- ◆ **Exact query**

Users should be able to enter textual terms used for finding a collection of related objects. The matching should be done against as many properties as possible without adversely affecting response time.

- ◆ **Noise removal**

A collection generated as the result of a query often contains many objects with low relevance to the user (“noise”). The interface should assist the user in improving precision and average relevance by facilitating removal of such objects.

- ◆ **Exploration of availability**

When browsing a large or unknown collection, the user usually first wants to get a quick overview over what it contains in order to establish the relevance of the collection and to plan further actions to be taken. The interface should therefore make it possible to identify common characteristics, types of objects, etc. in the collection.

- ◆ **Navigation**

Browsing objects in a collection often triggers new information needs. Finding an interesting painting made by a renaissance

---

painter in an art collection could for instance make user interested in locating paintings made by the same painter or by others in the same time period. The interface should make it easy to follow links to related information and thus generate new information collections.

◆ **Scanning**

Even if all irrelevant objects have been removed from a collection so that only objects of interest remain, it can still contain a large number of objects. It is therefore important that the interface makes it easy to scan the result and to view the individual properties of the objects contained within.

The new prototype should ideally be designed with all these strategies in mind. In particular, exploration of availability and navigation were not too well supported in the first prototype. For example, as the generated questions showed only a few alternatives, this did often not constitute enough high-level information to provide a good overview of the result.

### **8.1.2 Finding suitable filters**

In the first prototype, six different properties were considered when the questions for property restrictions were generated (see Section 7.1). Of these six properties, only five were presented to the user. These five were selected on the basis of a calculation of estimated utility. By only excluding a single property, the merits of this selection process were difficult to determine.

The new prototype was extended to include a total of eight properties while still displaying only a subset of five. This should make it possible better to evaluate the selection process as well as the obvious benefit of making more properties available for restrictions.

After each dimension has been evaluated individually, a way of comparing them with each other is still needed (categorical and numeric data alike). With eight available properties and only a maximum of five to display, the method used for comparisons obviously becomes more important. As for the previous prototype, this is done by the means of utility functions. They are designed to estimate the usefulness of a given property value restriction for the current result by returning a number from zero (useless) to one (ideal).

---



In the first prototype, four different utility functions were used. In order to offer more freedom to the user, the new prototype will no longer will support yes/no-questions (i.e. filters based on a single property value). This makes four of the old utility functions obsolete. In addition, I have decided to allow the user to freely select an interval of interest for numerical data rather than trying to find suitable intervals programmatically. This is done in the interest of providing more freedom to the user. As a consequence, the utility function for numerical data can be simplified.

In fact, in the new prototype only a single utility function is used. This function is based on the four old utility functions with changes motivated by the indirect evaluation of the utility functions in the first usability evaluation and my own experiences with the first prototype. The new utility function is defined as a multiplum of three different terms (only two applicable to numerical data). Each of these terms are presented below.

To analyse the properties of a result, property values for all result objects are retrieved from the database. Each property  $P$  has a set of property values  $V_P = (v_1, v_2, v_3, \dots, v_n)$ . The downloaded data are used to construct sets  $S_P$  for each property  $P$  that contains tuples  $(v_i, c_{vi})$  where  $c_{vi}$  is the number of result objects that has the property value  $v_i$ . As null values often occur, each set also includes a special tuple  $(null, c_{null})$ . The set of values of a given property  $P$  considered to be displayed to the user is  $D_P$ .

### 1: Equal distribution desirable

The reasoning behind wanting property values with equal frequency was explained in Section 6.3.3. The main term in the revised utility function is therefore the same utility function presented in Section 6.3.3 for lists of categorical data. It is a a measure of how close the actual distribution of the (selected) property values are to an equal distribution.

$$term_1 = 1 - \frac{\sum_{v_i \in D_P} \left| c_{vi} - \frac{\sum_{v_j \in D_P} c_{vj}}{|D_P|} \right|}{\left(2 - \frac{2}{|D_P|}\right) \cdot \sum_{v_i \in D_P} c_{vi}}$$

## 2: Hidden property values undesirable

For categorical data, only a subset of the existing values are displayed. Often, this means that some values are not displayed and therefore not selectable. This is clearly undesirable, as such values can not be chosen by the user, thereby limiting the expressive power of the interface. This was discovered to be a very real problem in the first usability study (see Section 7.3). This is captured in the utility function by multiplying it with the percentage of objects that have their property values displayed.

As this prototype has been changed to display all property values for numeric data dimensions, this term is ignored for such dimensions.

$$term_2 = \frac{\sum_{v_i \in D_p} c_{vi}}{\sum_{v_i \in V_p} c_{vi}}$$

This term is the amount of objects having property values in  $D$  (i.e. showed to the user), divided by the total number of objects.

## 3: Null values undesirable

Null values represent incomplete or missing knowledge about the properties of an object. If the amount of null values is large, one can not be sure if the known values represent the result accurately. Therefore selecting a specific property value could in fact reduce recall as objects that in reality have the same value but are registered as null, would be removed from the result.

As this is a undesired effect, the final term of the utility function the percentage of result objects that have non-null property values.

$$term_3 = 1 - \frac{c_{null}}{\sum_{v_i \in V_p} c_{vi}}$$



For categorical data, a list of property values is displayed. To find the exact number of values to display, the utility function is applied to lists of different sizes. The list with highest utility is selected and used to compete with other properties in order to select the subset of properties that are to be displayed.

---

---

## 8.2 Implementation

---

The database used is still the Primus database from the Norwegian Folk Museum (see Section 6.1). However, some changes have been made to the test setup compared with what was described in Chapter 7. The DBMS used continues to be Oracle, but the installation has been upgraded to version 9i. The database server used now has a 1,1 GHz AMD Athlon processor and 1 GB of RAM with the data is stored on 10.000 rpm SCSI disks. These improvements to the database server are helpful as they should make it possible to consider more properties (see Section 8.1.2) without seriously affecting the response time of the prototype.

The interaction model described in Section 7.1.1 is also used in this prototype. The only change is better support for navigation – bringing the used model more in line with the abstract model presented in Section 5.1.

In the following subsections, the design and implementation of the different parts of the second prototype are discussed. The emphasis is on the changes made from the first prototype. For the details of the first prototype and the Primus database, refer to Chapters 6 and 7.

### 8.2.1 The initial query

In the first prototype, the initial search consisted of a single text field where the user could enter a term. This term would only be matched against the subjects of the photos and the names of the artifacts.

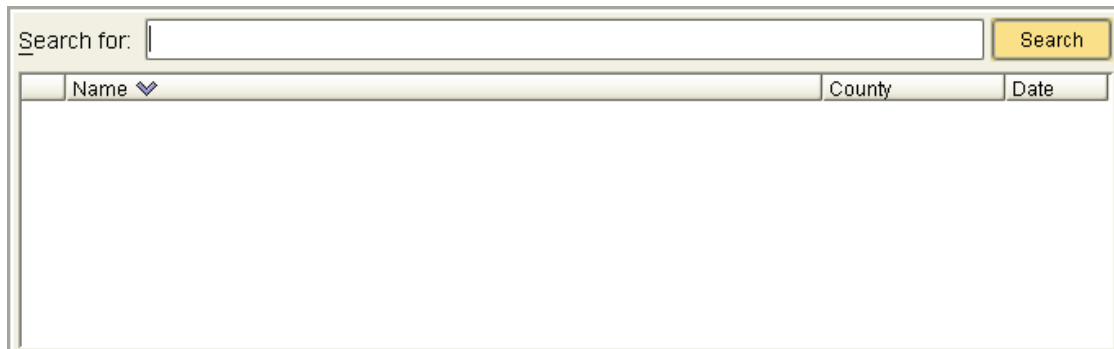
In the first evaluation, it was evident that this was too limiting. For example, several of the test subjects expressed a desire to be able to query for all photos and artifacts from their home county. This indicates that the interface was not too well suited for exact queries.

To remedy this, the new prototype was extended so that the initial query term now also is matched against:

- ◆ County of origin
- ◆ Municipality of origin
- ◆ Person involved (previous owner, subject on photo, etc.)

Optimizations to the database index structure and a hardware upgrade of the database server meant that these additions could be made without increasing the response time.

---



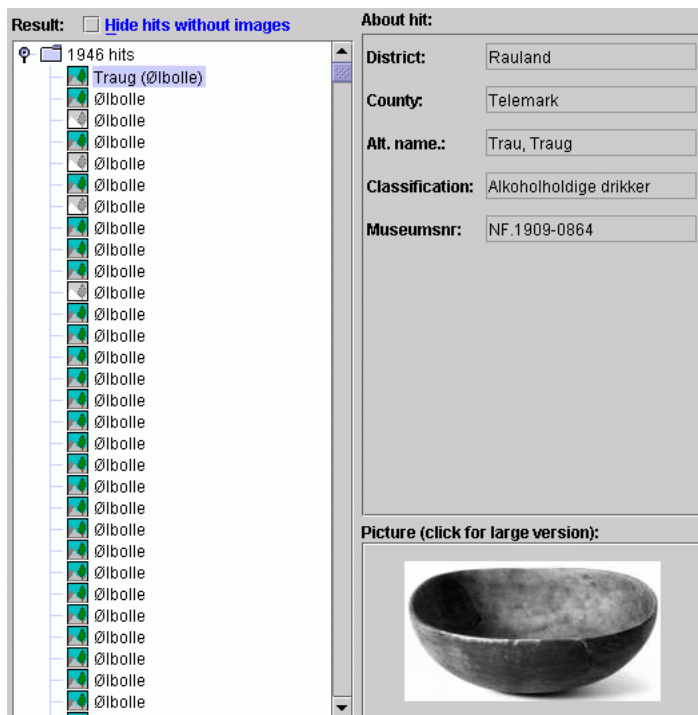
The image shows a web interface for a search system. At the top, there is a search bar with the text "Search for:" followed by an empty input field and a yellow "Search" button. Below the search bar is a table with three columns: "Name" (with a dropdown arrow), "County", and "Date". The table is currently empty.

**Figure 8.1 Initial query interface.**

As Figure 8.1 shows, the interface still has only a single text field for entry. The query term must therefore be matched against all the properties listed above in addition to subjects of photos and names of artifacts. An alternative solution would have been to add separate text fields for each searchable property. But as this could have spurred confusion as to which text field to use (as experienced during the evaluation of the forms based interface), this alternative was deemed less desirable.

## 8.2.2 Presenting results

Figure 8.2 shows the user interface component from the first prototype used to display the result.



The image shows a result display interface. On the left, there is a list of 1946 hits, with the first item "Traug (Ølbolle)" selected. The list is titled "Result:  Hide hits without images". On the right, there is a detailed view of the selected hit, titled "About hit:". The details include:

- District: Rauland
- County: Telemark
- Alt. name.: Trau, Traug
- Classification: Alkoholholdige drikker
- Museumsnr: NF.1909-0864

Below the details, there is a section titled "Picture (click for large version):" which contains a photograph of a dark, bowl-shaped artifact.

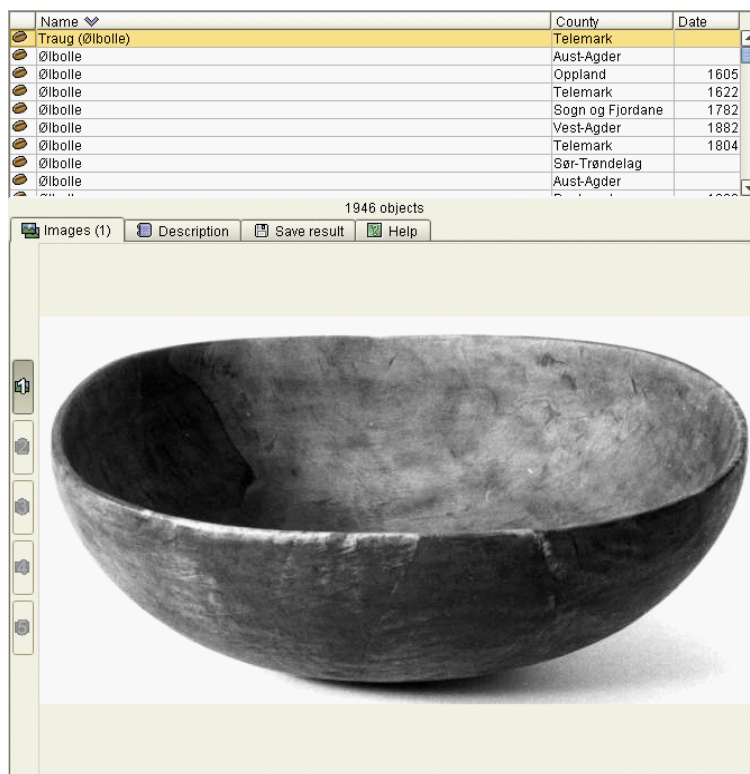
**Figure 8.2 Result display from first prototype.**

The following shortcomings with this component were identified:

- ◆ Pictures proved to be of great interest. In the first prototype, only a very small picture is displayed and the user is required to click on it in order to get a larger and more interesting one. This makes the interface slow to use for users that are primarily interested in pictures.
- ◆ Only the name (artifact) or subject (photo) is displayed in the result list. This makes it difficult to get a good overview of the result without selecting each object and reading more in the “about hit” pane.
- ◆ No ways of sorting the result are available.
- ◆ Only a few details are displayed about the selected object.

In total, this results in poor interface support for scanning. The user is required to select each item to get even the barest of details and to click the corresponding picture to get a picture of a decent size. This makes manual scanning slow.

The new implementation of this component is shown in Figure 8.3.



**Figure 8.3 Result display from second prototype.**

The changes are as follows:

- ◆ A much larger picture is displayed.
- ◆ Up to five pictures are available for each object instead of just one. They are selectable by buttons displayed on the left hand side.
- ◆ The result display now also contains county and date (year of origin).
- ◆ The result is sortable on name, county and date (year of origin).

In order to get the screen space required for the large picture, the detailed description of the selected object had to be moved to a different tab pane (displayed in the same screen space as the picture). This component is shown in Figure 8.4.



**Figure 8.4 Detailed information from second prototype.**

In addition to displaying more information than the last prototype, several items are displayed as hyperlinks. The user can click on these to make property restrictions matching the clicked value. For example, clicking on the county “Telemark” will hide all objects not from “Telemark”.

This makes it easy to locate objects similar to the one displayed. Note that this is still only intra-result – it will hide objects not having this property value and not add objects not part of the current result.

In summary, these changes will make it easier to perform scanning and navigation tasks.

### 8.2.3 From questions to filters

The user interface needs to include some way of letting the user specify restrictions on object properties. In the first prototype, this was done by asking the user questions as illustrated in Figure 8.5.

Questions:

**Material:**  
Are you interested in objects made of Tre?  
 Yes (1264 items)  No (682 items)  Not sure  
'No' includes 80 items with unknown material.

**County:**  
Are you interested in objects from Telemark?  
 Yes (438 items)  No (1508 items)  Not sure  
'No' includes 304 items with unknown county.

**Classification:**  
Are you interested in objects with the classification Alkoholholdige drikker?  
 Yes (1805 items)  No (141 items)  Not sure  
'No' includes 19 items with unknown classification.

**District:**  
Choose districts of interest:  
Valle (106 items)  
Ål (94 items)  
Telemark (67 items)  
Buskerud (65 items)  
Rindal (62 items)

**Year:**  
Are you interested in items from after 1886?  
 Yes (635 items)  No (1311 items)  Not sure  
'No' includes 635 items with unknown year.

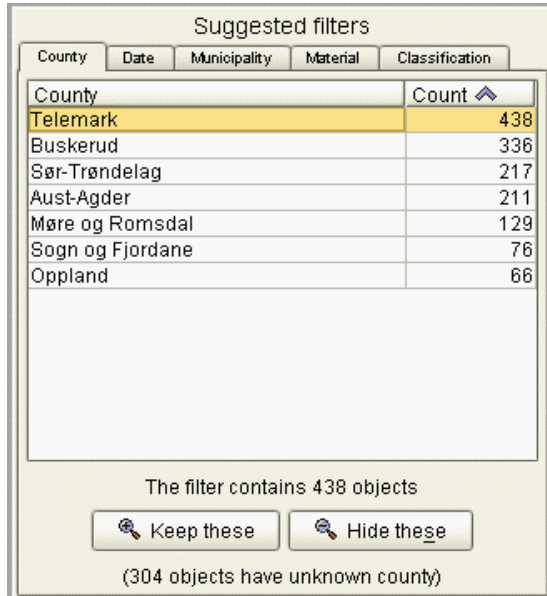
**Figure 8.5 Display of questions from first prototype.**

The evaluation identified a number of problems with this approach:

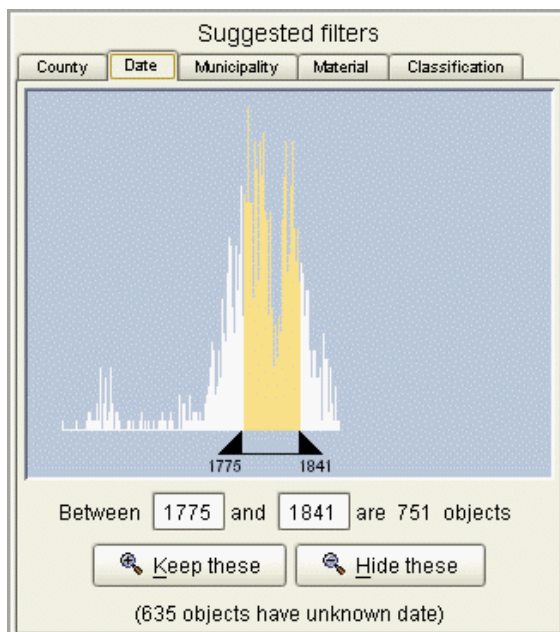
- ◆ Too much text to read.
- ◆ Yes/no-questions concern only a single property value. This was felt too restricting. Users often wished they could answer yes or no to a different property.

In order to solve the first problem, it was decided to turn away from using questions as the concept for interaction. The use of pseudo-natural dialogue in the application simply required too much text and slowed everything down.

The new prototype instead allows the user to construct filters which hide objects having or lacking a specific property value. These filters are either constructed by selecting one or more values from a list as shown in Figure 8.6 or by selecting a range of values as shown in Figure 8.7.



**Figure 8.6 Constructing filters for categorical data.**

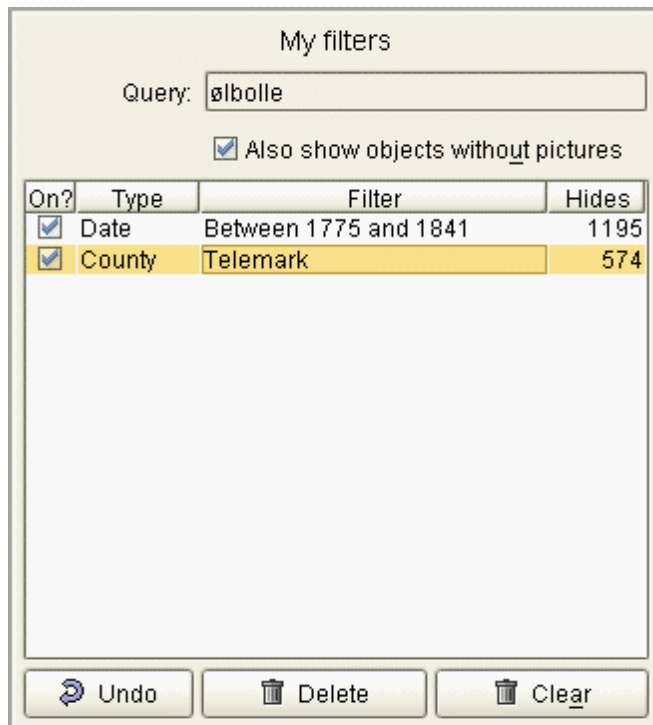


**Figure 8.7 Constructing filters for numeric data.**

The lists contain between 2 and 13 entries. 13 were selected as it is the maximum number that can be displayed while keeping all entries visible (not requiring scroll bars).



Filters are constructed when the user clicks either the “Keep these” or the “Hide these” button. Afterwards, the filter is added to a specific list called “My filters” as shown in Figure 8.8.



**Figure 8.8 Display of “my filters”.**

This user interface component provides a number of advantages over the old implementation:

- ◆ Provides at-a-glance overview over operations done so far. Thus minimizing the users’ memory load (see Section 2.2).
- ◆ Each of the filters can be temporarily toggled on and off using the check mark or permanently deleted using the “Delete” button. Using these two features, actions can easily be reversed – matching one of Shneiderman’s golden rules of interface design (Shneiderman, 1997).
- ◆ All filters can easily be removed using the “Clear” button, thereby reverting to the result of the initial query. This feature provides clear exits (see Section 2.2).
- ◆ Undo button makes it possible to unmake a new filter or return a recently deleted filter. Thus further promoting easy reversal.

These changes should make SESAM more suited for navigation as operations can be undone effortlessly. Exploration of both the result and the interface should therefore feel safer. It also provides a more clear view of exactly which modifications have been made to the result.

Another change is that filters can be kept from one query to the next. Each time a new query is submitted, the users are asked if they want to keep existing filters (if any). This way it is possible to build a set of preferred property restrictions and have them automatically applied to any new query result.

### **8.2.4 Improved looks**

As shown in the included screenshots, the new prototype includes a different “Look & Feel” (Sun Microsystems, 2002). I felt the old version looked somewhat cold and grey, and therefore wanted to change to warmer colours. Although this has no impact on the functionality of the program, it could make it more enjoyable to use the application. It has been, in my view successfully, argued that looks are an important factor for perceived performance and satisfaction (Norman, 2004):

*“The surprise is that we now have evidence that aesthetically pleasing objects enable you to work better. [...] products and systems that make you feel good are easier to deal with and produce more harmonious results.” (page 10)*

The improved looks were largely achieved by using the Alloy look and feel<sup>1</sup>.

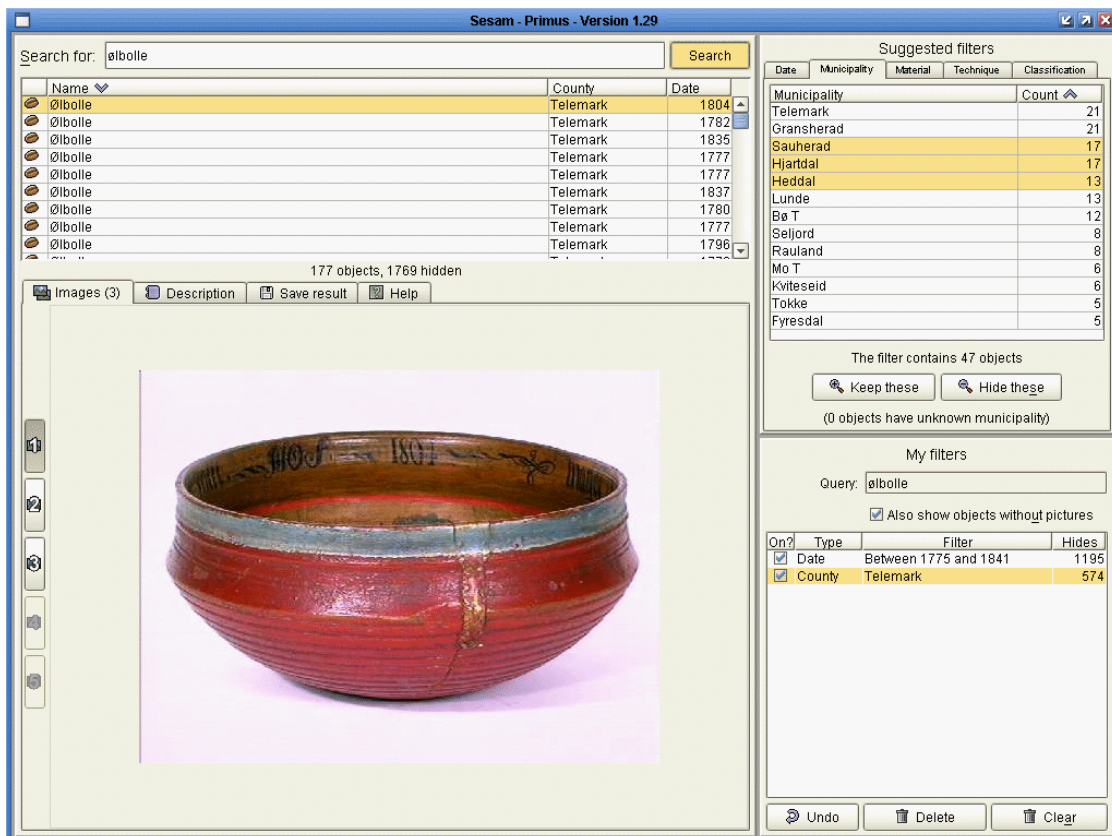
For similar reasons, the new prototype uses a lot more icons to make the interface more colourful and visually appealing. Icons also make it possible to visual relate different components. For example, the hyperlinks used in the display of the detailed information (see Figure 8.4) use the same magnifier icon as the “Keep these” button when making filters (see Figure 8.6 and Figure 8.7).

A screenshot of the whole second prototype interface is shown in Figure 8.9.

---

1. INCORS GmbH, <http://www.incors.com>

---



**Figure 8.9** The user interface of the second SESAM prototype.

## 8.3 Usability evaluation

This section presents a comparative user evaluation carried out using updated versions of the three different user interfaces used in the evaluation presented in Chapter 7. The purpose of the evaluation is threefold. First, I wish to find the strengths and weaknesses of the revised SESAM prototype compared with two other interfaces and to see if the comparative results have changed since the first evaluation. The two other interfaces are updated versions of the ones used in the original evaluation. Second, I plan to investigate if the improvements made based on the first evaluation, had any effect on the performance of the three interfaces. Finally, I expect to gain some insight in the value of user evaluation in general by measuring any improvements made.

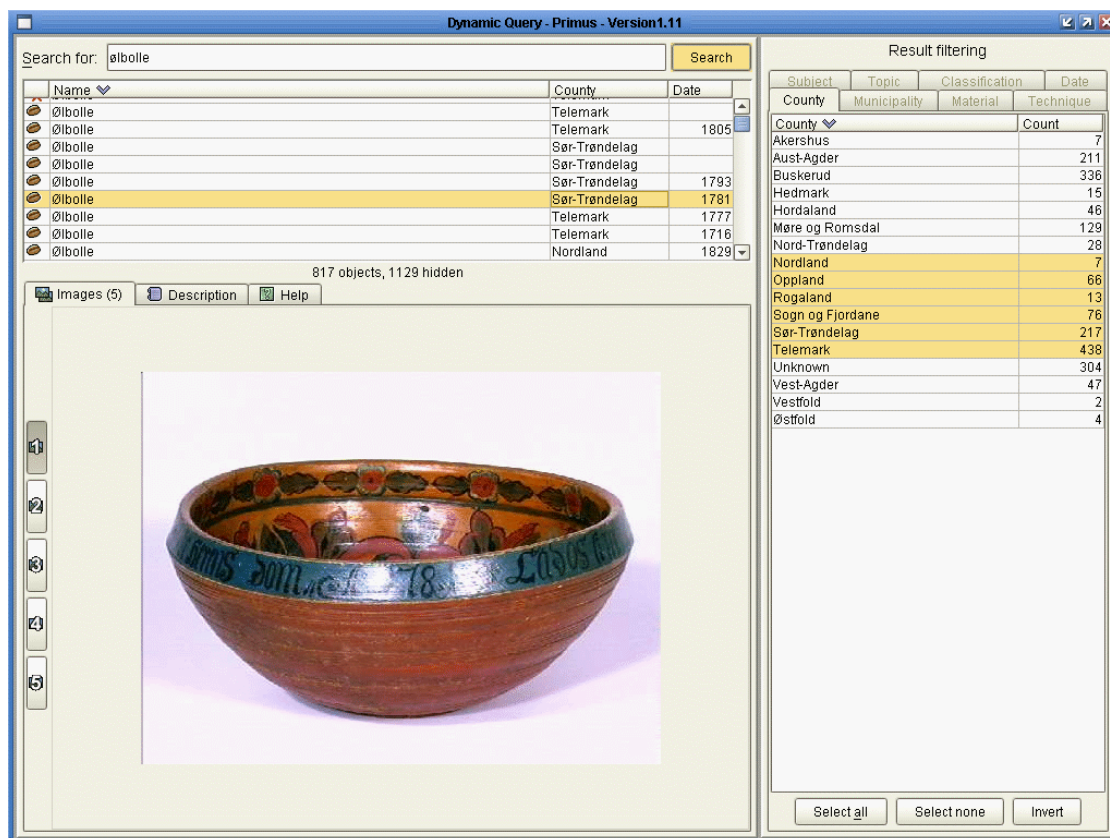
The organization of this section closely matches the outline of the evaluation presented in Chapter 7. The section starts with a description of the evaluation setting – including the evaluated interfaces and the evaluation methods. The second and final part, presents and discusses the results.

### 8.3.1 Setting

This evaluation is a comparative evaluation of three interfaces. The first, SESAM, has already been presented, while the last two are presented below. In order to make direct comparison with the last evaluation possible, the setting is kept more or less identical. Any changes that have been made are all in the interest of improving the reliability and validity of the evaluation results.

#### Updates to the dynamic query interface

This interface is an updated version of the interface presented in Section 7.2.1. Most of the changes made are a result of using the updated components already presented in Section 8.2. This includes extending the initial search to more property values and an updated component for presenting results. The updated interface is shown in Figure 8.10.



**Figure 8.10 Updated version of dynamic query interface.**

---

Compared with SESAM, the only part that differs is the right hand side of the interface. Here eight different tab panes are displayed – one for each of the property that can be used for value restrictions. These eight are the same used to construct filter suggestions in SESAM. Seven of these are categorical data dimensions where the user can select one or more values from a list (as shown in the figure). The last, year of origin (date) uses an interface similar to what is shown in Figure 8.7 where the user is shown a column chart of the distribution of property values and can select a range of interest using a custom slider. Direct manipulation is still used – any changes made to any of the restriction tab panes, automatically update the result accordingly.

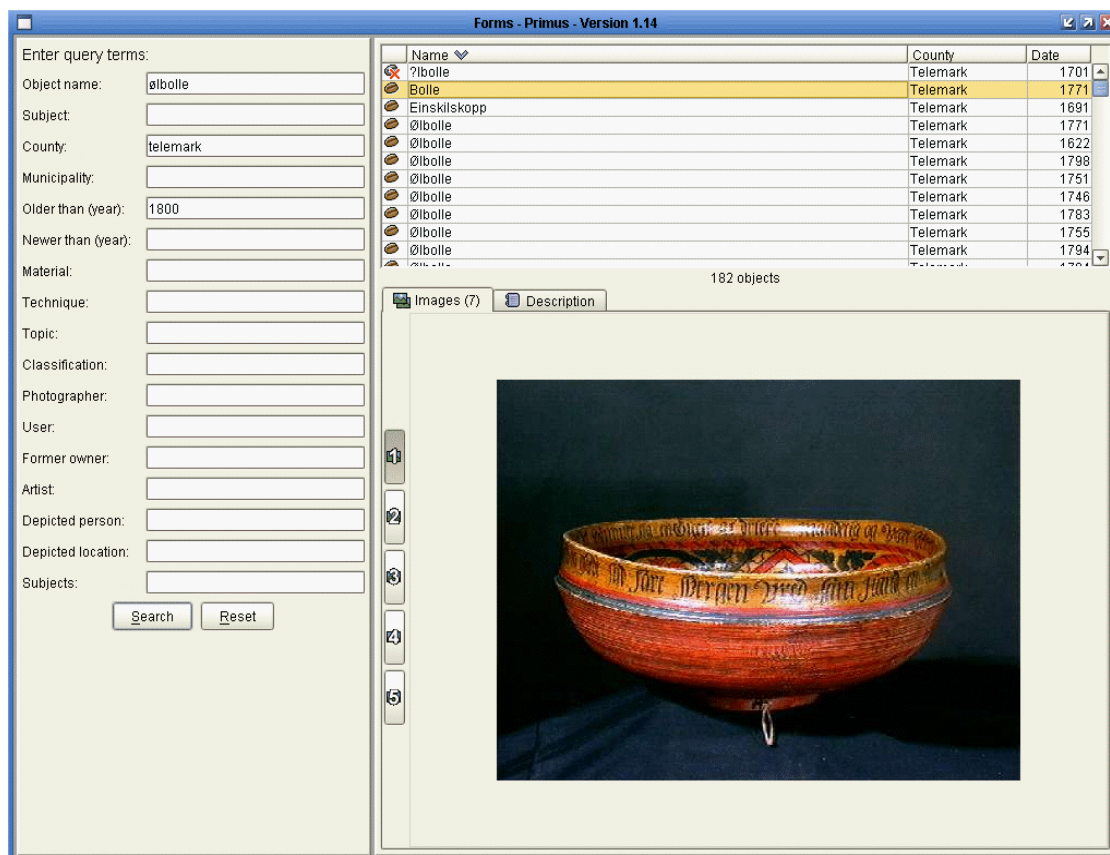
The key differences between this dynamic query implementation and the new SESAM prototype, remains almost the same as in the last evaluation. One exception is the handling of numerical data. Here SESAM takes a step towards dynamic query by making all property values available to the user (see Figure 8.7). However, numerical properties are subject to utility evaluation to compare them with the categorical properties to select which of them to display. As more properties are considered in this evaluation than in the last, these differences between SESAM and dynamic query should stand more out in this evaluation than the first.

The updated component for displaying object details described in Section 8.2.2 is also used here. Descriptions displayed as hyperlinks can be used to specify property restrictions without using the lists on the right hand side of the interface. This functionality is similar to what is used in the SESAM prototype interface.

### **Updates to the forms based interface**

The last interface is an updated version of the forms based interface first presented in Section 7.2.2. A screenshot is shown in Figure 8.11.

---



**Figure 8.11 Updated version of the forms based interface.**

The right hand side contains the display of the result and the selected object. Both these components are the exact same components used in the updated versions of the SESAM and dynamic query prototype interfaces.

The left side contains the text fields constituting the query form. In comparison with the last version, this updated version contains several additional text fields. They were added to match the extended number of properties used for filtering in the two other interfaces in order to make the comparison as fair as possible.

## Evaluation methods

In order to have the evaluation fulfil its objectives, it is important to select the right evaluation methods. A discussion of available methods was given in Section 3.4. To get results that can be compared with the results from the previous evaluation, it makes sense to consider the same methods.

I decided to use the following evaluation methods:

### **Pilot study**

As a single developer, it is very easy to become blind to one's own mistakes. This includes both obvious usability problems and programming bugs. As noted by Jakob Nielsen (Nielsen, 2000), most of these can fortunately be found by testing the program with only a few users.

All the three interfaces therefore went through a pilot study which involved four expert users using inspection methods. Several iterations of testing and implementation were performed before the interfaces were found ready for the main study.

### **Main study**

In the original evaluation, all the test subjects in the main study were master degree students in computer science. In order to improve validity of this new evaluation, it was carried out on two different groups of users. The first group was students similar to the ones used in the last evaluation, thus making it possible to compare their results. They are all experienced computer users, but had no experience with the Primus database and only basic knowledge of the subject matter. The second group was conservators from the National Folk Museum. They had average experience with computers, but all knew the subject matter very well – some had even been using the Primus database daily for several years. Together, these two groups represent a wide spectrum both with regards to general computer experience and more specific experience with Primus – its data model and data.

This new evaluation also used a counterbalanced within-subjects design (Mitchell and Jolley, 2001) where each tester tested all the different interfaces in order to make comparison possible. The order in which the interfaces were tested, was arranged such that each interface was tested first, second and third equally often. This should remove any bias due to order of testing.

As for the last evaluation, no initial training was given with the exception of a brief introduction. This makes it possible to observe initial reactions and is a more realistic scenario for interfaces designed for web use.

Two different methods of evaluation were used in the main study. Each test subject performed a series of predefined tasks using each interface (described below) during which they were observed. This allowed me to gather qualitative data as to how the interfaces were used. In addition, the test subjects were given a questionnaire similar to the one used in

---

the last evaluation. The questionnaire contained a subset of the questions in the QUIS – Questionnaire for User Interaction Satisfaction (Chin et al., 1988), repeated for each of the interfaces.

This time, performance measurement was left out of the usability evaluation. As described in Section 7.3.4, there were several reliability and validity concerns with the measurements last time – most of these difficult to fix. I have also come to the conclusion that measuring the time spent to solve tasks might not be so important for these interfaces. This is because performance measurements make little sense for tasks with vague or no clear objectives as it is difficult or even impossible to define when such tasks have been completed. It would still have been possible to include performance measurements for exact queries, but as this would have made an already lengthy evaluation even longer and with questionable rewards, the idea was dropped.

## **Evaluation tasks**

In Section 4.1, I identified several possible information searching strategies that apply to the interfaces used in this evaluation. It makes sense to design evaluation tasks so that the performance of the interfaces with regards to each of the different strategies can be tested. I have therefore selected the following types of evaluation tasks (also see Section 8.1.1):

### **Exact queries**

The test subjects were given two different queries. The first only consisted of a single term, while the second had a combination of two terms (e.g. material = “wood” and year = “1850”). These queries also served as an easy introduction to each interface and to the more challenging tasks.

### **Noise removal**

In this category, the users were given a single task. It consisted of a query term that gave an initial result with a high degree of noise. Typically this would be a term with two different interpretations where the users were told which interpretation to focus on. The users were then asked to modify the result so that the amount of noise became bearable (as defined by the test subject) without too much of the relevant objects being lost in the process.

### **Exploration of availability**

Here the objective was to examine how well each interface was suited for gaining an overview of a large result. This was done by asking each test subject to search for objects originating from their home county. Such

---



queries give results which typically contain 2000 – 4000 objects of a wide variety. They were then asked to find the most common type of object in the result as well as objects that were of interest to them.

### Navigation

Navigation typically involves following links from one object (or collection) to another. In this evaluation, navigation was simulated by first having the user enter an initial query. The user was then asked to locate an object of interest in the result and use the interface to find similar objects. For example, given a result of the query “bowl”, the user could have found an object made of a special type of wood, proceeding to find bowls made of the same material.



In addition to these four searching strategies, I earlier also defined scanning as a central task. However, as scanning is interwoven with each of the tasks given above, I found it unnecessary to include tasks especially designed to evaluate scanning performance.

## 8.3.2 Results

This section presents the results of the observation and the questionnaires. Overall discussion of the results is left to Section 8.3.3.

### Results from observation

The observational data was recorded on paper during each of the evaluations. Similar to the last evaluation, most of the notes concerned minor interface issues that separately are of little interest. The number of such issues was notable reduced compared to the first evaluation, even if the number of test subjects was increased. I see this as a clear indication of more mature implementations. By collating these minor issues, the following five high level observations were extracted:

#### 1: Postprocessing of results vital

It soon became clear that forms, while sufficient for exact queries, are far from usable for tasks such as exploration and noise removal. The inability to do any kind of modifications to a result often made users give up completing these types of tasks.

In the exploration tasks, users were asked to use the interface to gain an overview of the result. Using forms, the only way to do this was manual scanning of the result. With often more than 1000 result objects, it was

---

clearly too much work to examine them all individually. Users therefore had to resort to reading the one line of description in the result display as well as reading detailed descriptions on individual objects. This took quite a while and users soon became too bored to continue.

The noise removal tasks had the users trying to remove irrelevant objects from a result. The only way to do this with forms, was to find a new and more restrictive search expression and restart the query. Finding this new expression proved difficult to many users. They could use descriptions of existing result objects for inspiration, but often it came down to pure guesswork. Needless to say, this often resulted in empty or otherwise irrelevant results. This was less of a problem for users with background knowledge of the database as they could typically make educated guesses as to useful query reformulations.

## **2: Background knowledge affects performance**

Observation highlighted a number of differences between the computer professionals and the museum conservators. The former group was on average able to cope with SESAM's slightly more complex interface better than the latter. Experienced computer users generally had little problems figuring out the "my filters" component, while it remained a mystery to several of the less computer experienced users.

Similarly, improved background knowledge of the database also changed how the different interfaces were perceived. Museum conservators generally had less problems using forms successfully – they used the correct form fields and often knew which values made sense in each field. This also made them better at handling the sometimes very long filter lists displayed by the dynamic query interface. While users new to the database often had to read every list entry, conservators usually had an idea of what they were looking for.

## **3: Negative operators difficult to use**

Both SESAM and dynamic query offered negative operators. In SESAM it was possible to construct a negative filter that would hide all objects with a specific property value (e.g. show only objects not made of wood). Similarly, users using the dynamic query prototype could either start with a complete result and remove unwanted property values, or start with an empty result (all objects hidden) and add objects with specific values.

---

The observations show that negative filters in SESAM were almost never used even if several of the tasks would have benefited a lot from their use. And for dynamic query, it was a clear preference for starting with an empty result and adding objects. These trends were especially clear for the test subjects with the least computer experience.

On the whole, it seemed that negative operators are more difficult to use and probably should be left out of user interfaces which especially target novice users.

#### **4: Use of metaphors must be consistent and thorough**

The detailed result display in all three tested interfaces contained functionality for expressing property value restrictions simply by clicking on the displayed values. (See Section 8.2.2 for a more detailed explanation.) The implementation borrowed the hyperlink metaphor (blue, underlined text label) used by web browsers to indicate that these values could be clicked.

From the observations it became apparent that this functionality was not intuitive to several of the users. It simply did not occur to them that these text labels could be clicked. But there was a marked difference between the computer professionals and the museum conservators. While almost everyone in the first group figured this out, only a few in the latter did. When I asked a test subject directly why she did not try to click the text label, she told me that she did not expect the application to work like a web page. I.e. she did not apply previous knowledge from web browsers because these applications did not match the environment where this knowledge was gained.

This indicates that metaphors can only be expected to work if they are used consistently and throughout the application. One solution would have been to make the whole application look more like web pages. This could for example be done by substituting undo buttons with forward/back-controls, supplying a “home”-button, etc.

#### **5: Clear way back to known state helpful**

One of the observations in the last evaluation, was that actions should be easily reversible (see Section 7.3.2). Presumably due to the improvements made to the applications used in this evaluation, this was not a big issue this time.

However, a related problem was identified. For both SESAM and the dynamic query prototype interfaces, it is possible to perform a large number of operations on a given result. Especially in the initial phase

---

when users experiment to figure out the interfaces, they often got “lost” – i.e. did not know how they could get back to the initial result. While this was possible to do in both interfaces, it was obvious that this functionality was not made clear enough. Several users stated that they were looking for a “Reset” button and they often ended up restarting the query unnecessarily to remove the effects of any post-result operations.

### **Results from questionnaire**

The questionnaire used in the evaluation consisted of 16 different questions for each of the three interfaces. For each question, the users could select their response on a 9-point scale (1-9).

The first four questions concerned the performance with respect to the four evaluation tasks. The fourth was an overall assessment, while the latter 11 questions have been collated into the two following categories to simplify the presentation of the results (as was done the first usability evaluation):

#### Visuals:

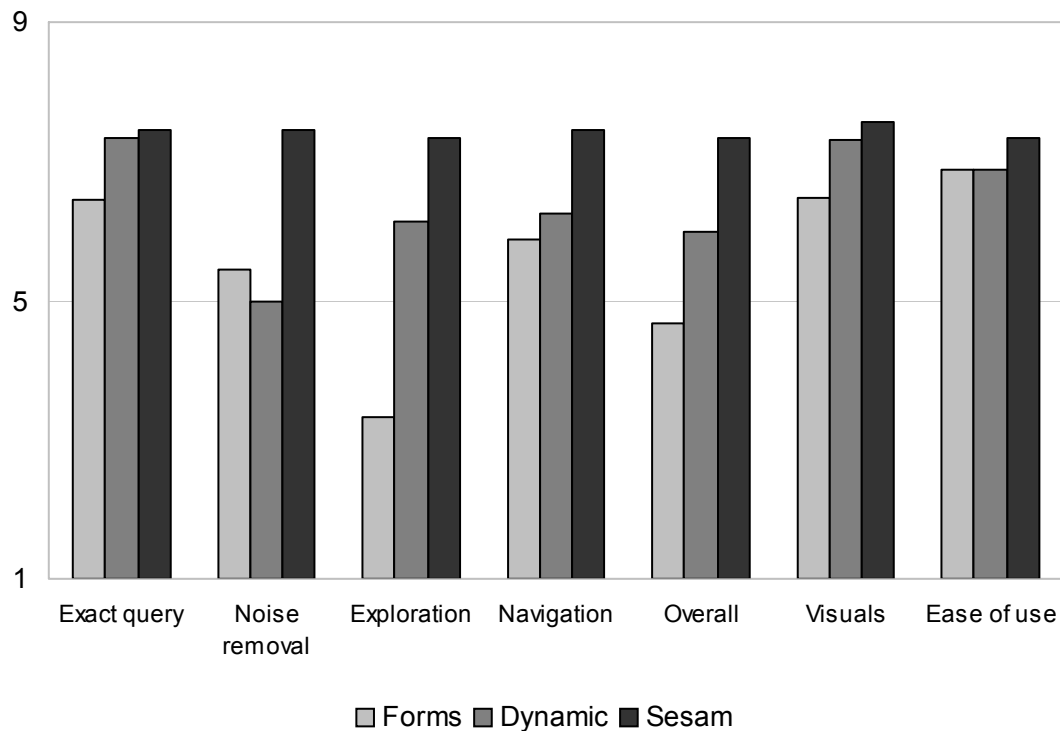
- ◆ Screen layouts were helpful (1:never – 9:always)
- ◆ Amount of information that can be displayed on screen (1:inadequate – 9:adequate)
- ◆ Arrangement of information on screen (1:illogical – 9:logical)
- ◆ Messages which appear on screen (1:confusing – 9:clear)
- ◆ Computer keeps you informed about what it is doing (1:never – 9:always)

#### Ease of use:

- ◆ Learning to operate the system (1:difficult – 9:easy)
  - ◆ Getting started (1:difficult – 9:easy)
  - ◆ Time to learn to use the system (1:slow – 9:fast)
  - ◆ Exploration of features by trial and error (1:discouraging – 9:encouraging)
  - ◆ Exploration of features (1:risky – 9:safe)
  - ◆ System speed (1:too slow – 9:fast enough)
-

An overview of the results from the results gathered from the computer professional test subjects, is presented in Figure 8.12. A complete overview of the results can be found in Appendix A.2.

### Questionnaire results - Computer professionals



**Figure 8.12 Results from questionnaires - Computer professionals.**

This group of test subjects is comparable to the group used in the previous evaluation. A meaningful comparison of the results is therefore possible. Please note, however, that as the users have compared each interface to the other two rather than to some abstract and universal standard, any questionnaire result is not directly comparable with the corresponding result from the first evaluation.

Interpreting these results, I consider the following three findings to be the most notable:

#### ◆ Overall ranking

A clear trend throughout this evaluation was that the changes made to SESAM had a positive impact on the results. While the last evaluation showed that the first implementation was impeded by poor visuals, the new version rates at least equal to the other two tested interfaces in all categories.

**◆ Noise removal performance**

Examining the results for the four different types of evaluation tasks, two notable differences in performance are apparent. The first is that SESAM was deemed clearly best for noise removal. A discussion of why this might be, is presented in Section 8.3.3.

Dynamic query was found to perform worse here. This corresponds well with observational data. Users were often forced to spend time scanning long several lists looking for a property value that they were not interested in and which would make noticeable difference to the result.

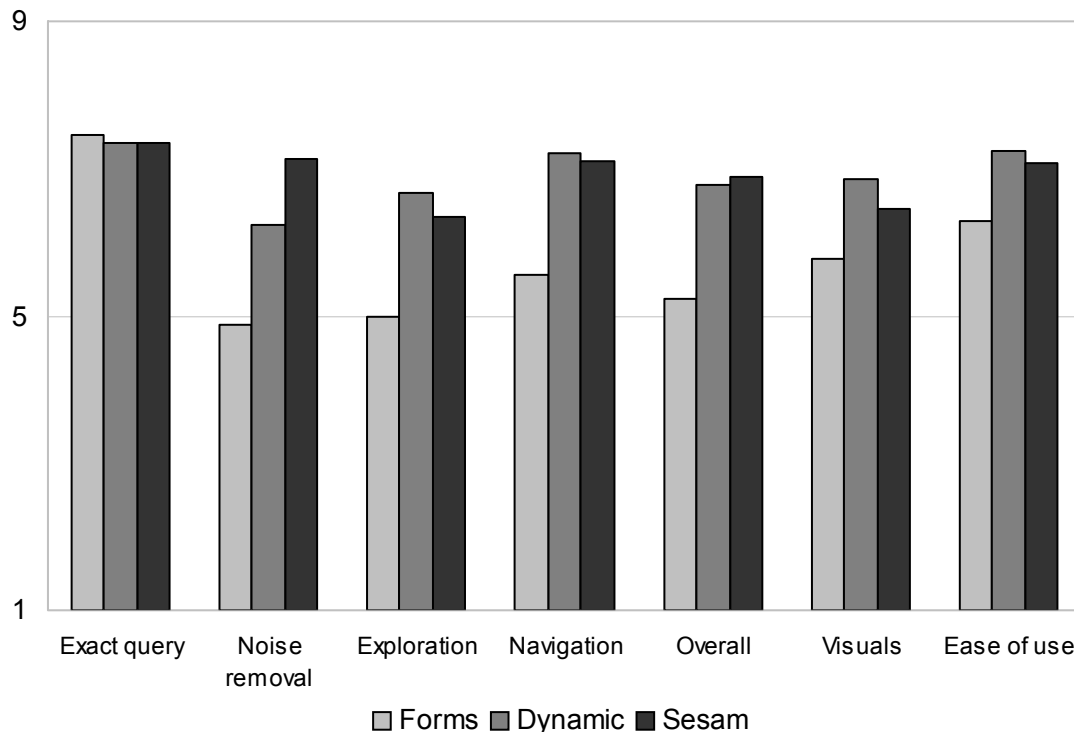
**◆ Using forms for exploration**

The second marked difference in task performance, is that forms was found poor for exploration of (large) results. This corresponds well with the observations described above. Note that this is a marked change compared with the results from the first evaluation. In my view, this indicates that the exploration tasks were better designed in this evaluation.

An overview of the results from the questionnaires filled out by the museum conservators is shown in Figure 8.13.

---

### Questionnaire results - Museum conservators



**Figure 8.13 Results from questionnaires - Museum conservators.**

This user group is characterised by lower general computer expertise but much higher domain knowledge and knowledge of the underlying data model in the Primus database. It is therefore interesting to compare the results with the results from the computer professionals group:

◆ **Forms and dynamic query better for conservators**

Both forms and dynamic query had better results here than for the computer professionals group. For forms, knowing the database is an obvious advantage when forced to come up with property restrictions yourself. Testers in the first group often complained about the sheer number of options available in the dynamic query interface (number of restrictions possible). It seems this problem had lesser impact for conservators – probably because they had a better idea of what they were looking for. These questionnaire results also corresponds well with the observational data.

◆ **Visuals of SESAM are found poorer**

Overall, SESAM was less liked by the museum conservators than

by the first group. Judging by the results in the “visuals” category, the increased complexity of this interface is at least one of the reasons. By examining the individual questions in the “visuals” category (see Appendix A), SESAM did receive especially low marks with regards to the clarity of the on-screen messages.

### **8.3.3 Discussion of evaluation results**

In this section, I first summarize the results for each of the tested interfaces. I then discuss several reliability and validity concerns that are important to keep in mind when assessing the results. Finally, I examine if the three purposes for this evaluation presented in the start of Section 8.3 have been fulfilled.

Forms performed about similar to the last evaluation. It proved easy to learn but difficult to use – at least for users unfamiliar with the Primus database. The lack of any support for result modification made forms inconvenient for exploration and handling of large results.

Dynamic query was found best suited for results of small to medium size. Large results often resulted in simply too many presented possibilities for imposing property restrictions. This problem was however not so pronounced for users with background knowledge of the database as they had an easier time navigating the large property value lists.

In comparison with dynamic query, questionnaire results and observational data indicate that SESAM sacrifices interface simplicity in order better to handle large results. This interface was generally preferred by computer professionals, while the interface complexity had greater impact for less experienced users. There was however an overall agreement that SESAM was best suited for noise removal.

No system performance evaluation comparable to the one presented in Section 7.4 was performed on this prototype. Almost all changes to the new prototype have been user interface changes which do not have any impact on the system performance. The approach with an initial search followed by an attribute value analysis remains the same.

#### **Reliability and validity**

In Section 7.3.4 I presented a number of reliability and validity concerns in the first evaluation. Many of these concerned recorded performance data. In this evaluation, performance measurement was totally left out of the test procedure. This might seem like opting for the easy way out, but

---



as explained in Section 8.3.1, time-to-completion might not be a very well suited metric for tasks other than exact query. For these types of tasks it is typically difficult to objectively define when they have been completed.

In this evaluation, more time have been spent on considering different searching strategies and using this knowledge to design suitable evaluation tasks. I am therefore confident that this evaluation paints a more complete and correct picture of the advantages and disadvantages of each tested interface.

The selection of test subjects and the use of a randomized design, were one of the weak points of the first evaluation. The situation was improved in this evaluation by using two groups of test subjects – from experienced computer users with no database knowledge to the opposite. The counterbalanced design also helped to remove any ordering effects. With more time available, the use of test subjects could have been further improved by adding more classes of users – for example people with both little computer background and knowledge of the Primus database.

Concerns regarding observator bias remained constant compared with the last evaluation. For a discussion of any possible impact, see Section 7.3.4.

### **Summary of evaluation**

In the beginning of this section, I declared the purpose of this evaluation to be threefold:

1. *Find the strengths and weaknesses for the revised SESAM prototype compared with two other interfaces.*

Unlike the last evaluation, SESAM was this time found to be the overall best interface. However, close examination of the results indicates that preference is highly dependent on background knowledge and task at hand. For example, SESAM is better at large results and exploration, while exact queries was done equally well on all interfaces.

2. *Investigate if the improvements made based on the previous evaluation, had any effect on the performance of the interfaces.*
-

Both informal observations and more formal questionnaire results indicate that all of the tested interfaces benefited from improved visuals and ease of use compared to the last evaluation.

3. *Gain insight in the value of user evaluation in general.*

As almost every change was implemented based on lessons learned in the evaluation, this is a clear sign of the usefulness of usability studies.

## **8.4 Discussion**

---

In this section I discuss my experiences with the four fundamental design ideas presented in Chapter 5 with respect to SESAM (based on Chapters 7 and 8).

### **Revised interaction model**

SESAM employs an interactive interaction model where filters can be used to remove objects from a result. Filters can be used not only to retain what you are interested in, but also to remove objects you are not interested in. Further, SESAM supports an integration of search, filtering and intra-result navigation techniques for accessing information.

The results of the evaluations show that some means of manipulating a result is very helpful – especially for large results. The filter solution implemented in SESAM was found particularly useful to remove objects of low relevance (“noise removal”), more so than for retaining objects of high relevance. The reason for this discrepancy will be discussed below in the “Active user interfaces” section.

The downside of the revised interaction model as implemented in SESAM, was that it made the interface more complex. This led to some usability problems, especially in the first prototype, and users also needed some time to adjust to the interface. However, in the end the test subjects found that the increased power more than made up for the increased complexity – at least for all but the most simple searching tasks.

---

### **Intra-result analysis**

This design idea was implemented in SESAM by the means of dynamically generated filter suggestions. These suggestions were made by analysing the properties of the objects in a result at run-time. For example, in a result containing artifacts from all around Norway, filters for each of the five most common counties of origin for these artifacts might be suggested.

Evaluations indicated that this technique was very helpful for gaining an overview of large results. Often test subjects did not have to manually scan hundreds of objects in order to get an understanding of what they had got. This information was not surprisingly found to be most useful for users with little background knowledge of the database as they could not draw upon previous experiences to interpret the result.

By comparing the experiences from the first and second evaluations, some difficulties in implementing this design idea can be identified. To keep the derived information relevant, it was recomputed each time the user altered the result. This led to large parts of the interface changing – components shifting places, text updated etc. The test subjects in the first evaluation found this to be disconcerting – they did not like to have to “reparse” the interface on a regular basis. The second prototype was designed with this in mind and the second evaluation confirmed that the problem was greatly reduced.

### **Active user interfaces**

Instead of having the user enter query terms to modify an existing result (recall), SESAM allowed the user to choose a filter from suggestions made by the program (recognition). As this meant only a limited number of filters were possible, some expressive power were unfortunately lost.

In the first prototype, filter suggestions were presented as questions (e.g. “Are you interested in artifacts from before 1800 (yes/no)?”). This interaction method, although natural, was not found to be suitable. It simply required users to read too much text and thus slowed the process needlessly down. Questions as an interaction method was therefore abandoned in the second prototype with improved results.

The second evaluation showed that this design idea has its merits – especially for non-experts and for more vague information needs where the lost expressive power has less impact and the issue of recognition vs. recall is more important. As mentioned above, the choice of displaying

---

filter suggestions rather than make it possible for users to construct their own filters, results in a loss of expressive power as only a limited number of filters are possible.

Interestingly, this loss of power is more apparent when users are focused on retaining what is relevant rather than removing what is not. When (typically experienced) users focused on removing objects with irrelevant object property values, they almost always found a suitable filter suggestion as it was easy to at least find something that is clearly irrelevant.

The opposite was true for users which focused retaining what is interesting. They were looking for filter suggestions mentioning something they was interested in. Typically, they knew what they were looking for and were annoyed when no matching filter suggestions were found. This hints at that the users' performance might increase when they learn that the former alternative is easier to get to work.

### **Dynamic user interfaces**

The SESAM user interface can be said to be dynamic as the displayed filter suggestions are constantly updated to reflect changes in the result. The user interface for making filters also depends on the type of property the filter is to be a restriction of (categorical or numeric data).

As this idea in SESAM goes hand in hand with intra-result analysis, the same implementation difficulties were evident here. The first prototype had too large parts of the interface constantly changing and this was found to be disconcerting by most test subjects. This effect can be understood as an issue of poor support for recognition (see Section 5.3). By changing large parts of the interface, the users could often not recognize what they were presented with based on previous interaction, and were thus forced to “reparse” the interface.

As most teething problems were all but eliminated in the second prototype, it became clear that this idea was most suited for large results and for exploration. Large results could easily be reduced to a more manageable size (“noise removal”) due to the process of selecting the displayed subset of possible filters being focused on finding filters that would be suitable for size reduction. For exploration, the constantly updating interface made it more easy for users to figure out where they were and to locate possible avenues for further exploration.

---

---

## Chapter 9

# Extending SESAM to image databases

---

In chapters 6 to 8 I described SESAM – an approach for accessing information in a textual metadata database. Evaluations of two SESAM prototype interfaces clearly indicated that the approach has its merits. But as of yet, the approach has only been tested on textual data. To test if they also can be applied to image databases, this chapter presents a proof-of-concept extension of the second SESAM prototype to handle images. This was done by taking advantage of the fact that the Primus database (see Section 6.1) used for SESAM also includes 230.000 images as part of the metadata describing the Norwegian Folk Museum’s artifacts and photos.

As discussed in Section 4.3.5, complex data types such as images, audio and video present unique challenges which make them an interesting topic for study. The most relevant with respect to my work is the semantic gap (Rui et al., 1999, Enser and Sandom, 2003). Techniques for extracting image features in the general domain are typically limited to information such as shape, colour and texture (Aigrain et al., 1996). These kinds of low-level information are clearly different from high-level concepts like people, objects, events, etc. central to human reasoning. This semantic gap makes meaningful interaction between users and system difficult to achieve. I am interested in investigating if the ideas used in SESAM, in particular the reliance of recognition rather than recall, can be used to reduce the negative effects of this semantic gap.

This chapter describes how SESAM was extended also to handle images and my experiences with the new prototype. Section 9.1 details how low-level information were extracted from the images in the Primus database. As this process was computationally very expensive, it was performed in advance with the results stored in the database. In Section 9.2 I examine how the SESAM interface was modified to make it possible to construct filters based on image features. Section 9.3 presents a discussion of the experiences of the modified prototype. In essence, extracting image features proved too unreliable to properly evaluate the design ideas from Chapter 5 in this setting. However, the reasons for these problems are interesting in themselves and thus makes the work as a whole worthwhile. In particular, several problems with making user interfaces suitable for content-based access have been identified.

## **9.1 Extracting image features**

---

While most of the images in the Primus database show one object with a uniform background, the images are so diverse that it is difficult to use specially adapted techniques. Thus feature extractions that work well in a general domain are most suitable.

A thorough discussion of different image feature extraction methods is beyond the scope of this text. For a general description of image processing, see for example (Jähne, 2001), (Sonka et al., 1998) and (Gonzalez and Woods, 1992).

Based on a review of available techniques, I decided to focus on extracting shape, colour and texture information from objects identified in the images. Object identification was done by segmenting the image into two regions, foreground (objects) and background. Each of these image manipulations is discussed below.

### **9.1.1 Segmentation**

Figure 9.1 shows a sample image from the Primus database – an object on a uniform background (discounting shadows). A fair assumption would be that the background is of no interest to most users. Therefore we would like to separate the object from the background so that only the interesting parts of the image (the object) are used for feature extraction. This is why image segmentation plays such a vital role in feature extraction and content-based image retrieval. Unfortunately, while segmentation has long been a topic for study, good image segmentation is still often impossible to attain (Li et al., 1999).

---



**Figure 9.1** Sample image from the Primus database.

As part of my work, several different algorithms were considered. A good overview of different alternatives for colour image segmentation is presented in (Skarbek and Koschan, 1994). Among those that were implemented and tested were edge-based methods such as Snakes (Xu and Prince, 1998) and region-based methods such as Watersheds (Vincent and Soille, 1991).

The method which ended up giving the best results for the Primus database was the EDISON system (Edge Detection and Image SegmentatiON) (Comaniciu and Meer, 2002). This system uses a mean-shift-based image segmentation algorithm enhanced with edge segmentation. Two examples of images segmented with EDISON are shown in Figure 9.2.



**Figure 9.2** Two examples of image segmentation.

The top example is near perfect, the object is separated cleanly from the background (including darker shadows). In the bottom example, a lot of the object is lost due to its colours being fairly similar to the background. This could have been fixed by adjusting segmentation parameters, but as the image database contained more than 236.000 images, individual optimization was impossible. The selected parameters were found by optimizing the segmentation results for a subset of more than 100 images randomly drawn from the database.

### 9.1.2 Shape

After object and background have been segmented, the object's shape is the outline of the segmented shape. Correct shape extraction is therefore highly reliant on correct segmentation.

A good shape description should have to keep properties (Lu, 1999):

1. The shape representation should be invariant to translation, rotation and scaling.
2. Equal shapes should have equal shape representations.



The process of matching images based on shape features can be divided into three. First the shape must be identified. Then this shape must be converted into a representation suitable for fast matching. Finally, the shape representations can be used to compute distance measurements indicating the similarity of image object shapes.

Different stages of the process of identifying object shapes are shown in Figure 9.3. The original image is shown top left. This image is segmented using the process described above. From the segmented image, the largest object is located using JAI<sup>1</sup> and the outline is drawn using the ImageJ image processing library<sup>2</sup>. The result is shown top right.



**Figure 9.3 Stages in the process of extracting shape information.**

1. JAI - Java Advanced Imaging, <http://java.sun.com/products/java-media/jai/>
2. ImageJ, <http://rsb.info.nih.gov/ij/>

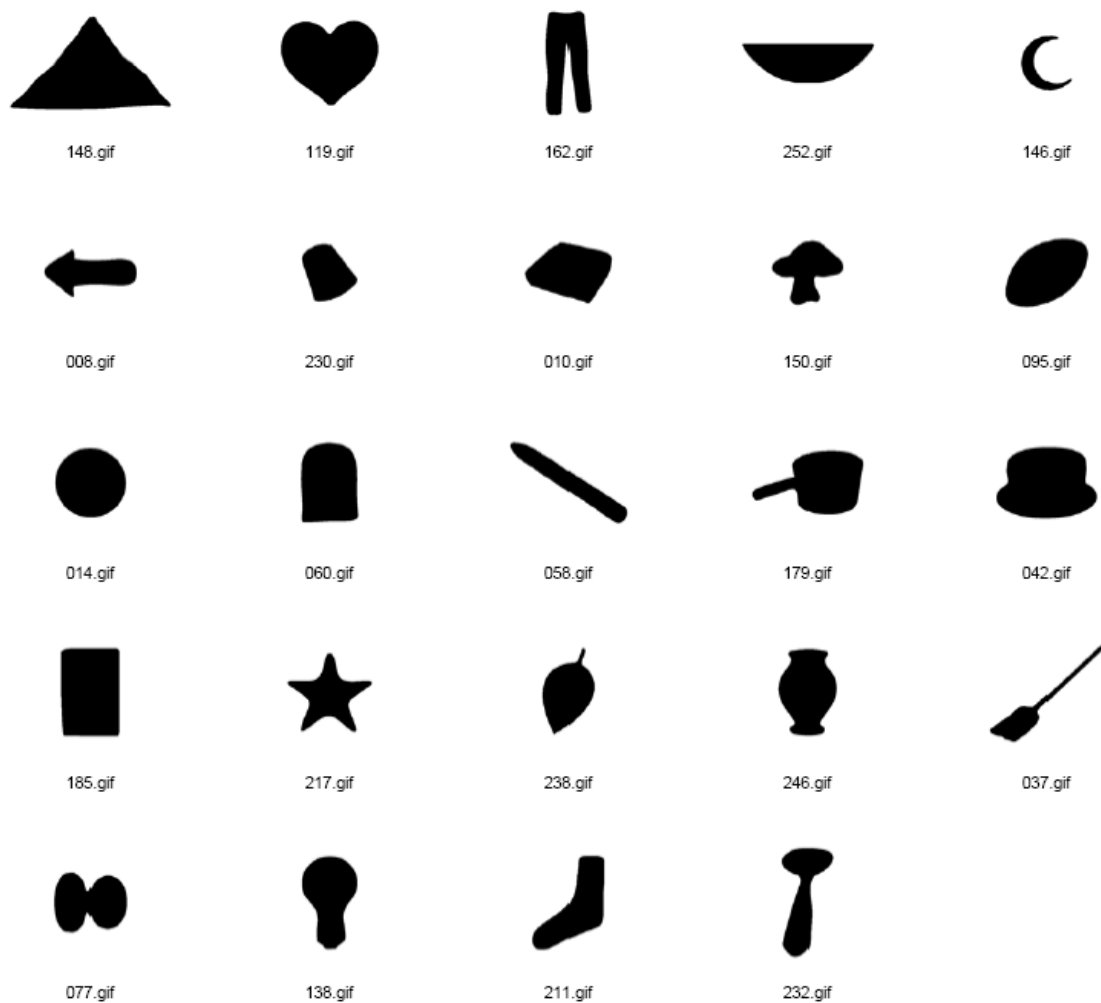
Two different shape representation formats are used. The first is moment invariants (Hu, 1962). Using normalized, central moments this representation is invariant with respect to translation, rotation and scaling. Moments 1 through 7 are computed and stored in the database for each image.

The second format used was fourier descriptors. To make the fourier descriptors less sensitive to noise, the object outline was first smoothed by applying a fourier transformation, removing the higher frequencies and applying the inverse transformation. An example of a resulting image is shown bottom left in Figure 9.3. Finally, the outline was simplified (and points equally spaced), shown bottom right, and fourier descriptors computed. The process used algorithms borrowed from (Rui et al., 1996) and (Zhang and Lu, 2001).

The extracted shape information was always meant to be used to allow users to make filters based on shape. In order to do this, some way of grouping objects with similar shape must be present. Ideally, this grouping should be done dynamically at run-time in order to get a categorization optimized for the result at hand.

However, due to performance reasons, a different approach was used. Rather than to compare image shapes to each other at run-time, a set of standardized shapes was selected in advance with each image being classified according to the closest matching standard shape. The standardized shapes used are shown in Figure 9.4.

---



**Figure 9.4 Standardized shapes used for classification.**

These images are based on a set of 260 shapes from (Snodgrass and Vanderwart, 1980) further modified by (Wagemans et al., 1998). This set was originally derived in order to study differences and similarities in the processing of pictures and words and is considered one of the standard databases of common objects (Weinshall and Kirkpatrick, 2004). From the original set, near-identical and specialized shapes were removed until the 24 shapes I found most distinct, abstract and simple remained.

In order to compute which standardized shape a given object was most similar to, the extracted moment invariants and fourier descriptors were compared. For fourier descriptors, I used a distance metric described in (Rui et al., 1996). This metric has been shown to be highly resistant to

image transformations such as translation and rotation. For moment invariants, a simple scaled sum of differences between the respective moments was used (scaled so that lower moments carried more weight).

### **9.1.3 Colour**

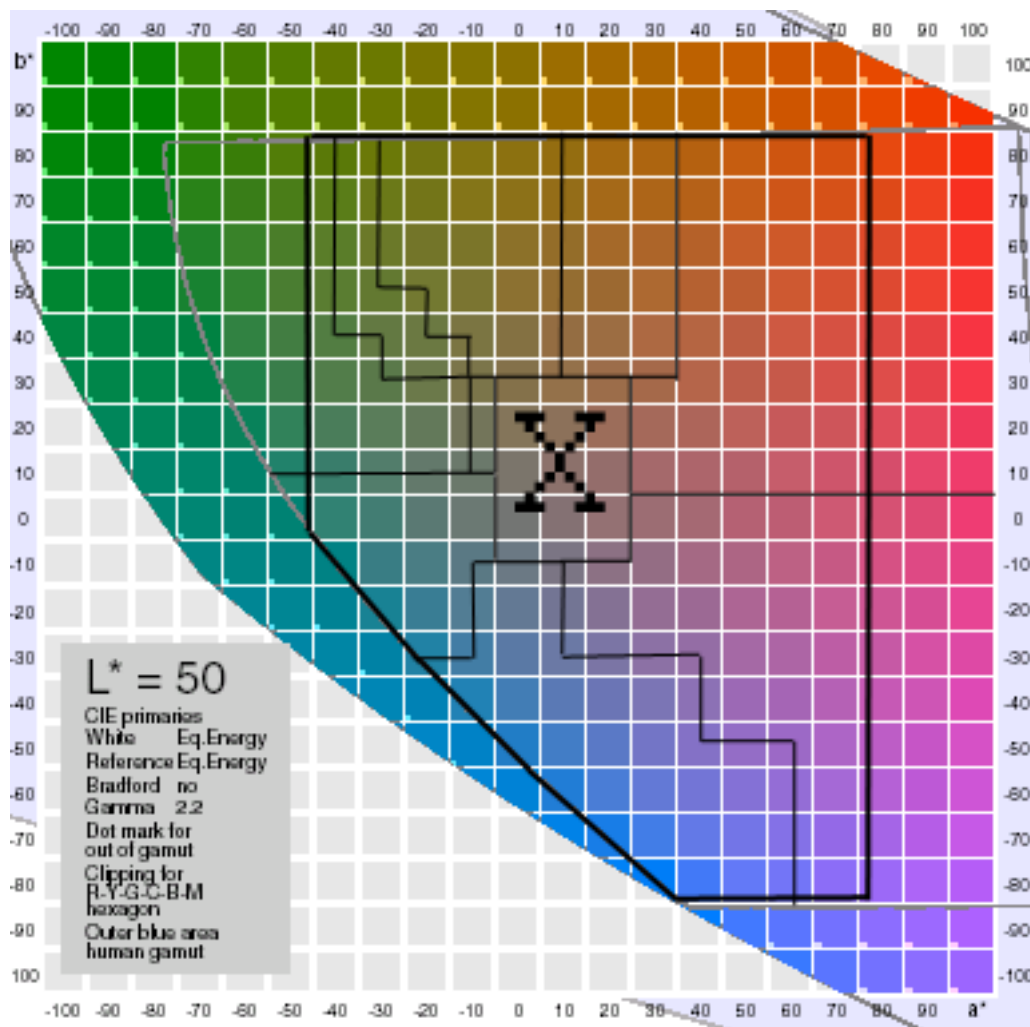
Colour is perhaps the most widely used feature in content-based image retrieval. Colour can be comparatively easily extracted, is independent of object placement and object transformations. As only object colour is of interest, segmentation is also necessary here. It is however important to note that colour extraction is less reliant on a perfect segmentation compared to shape extraction. As long as a majority of object pixels (and not too many background pixels) is included, the colour information is relatively unchanged.

The most usual way of storing colour information, is using histograms (Gonzalez and Woods, 1992). The method implemented in my work differs in that it instead uses statistic moments from said histograms. This idea was suggested in (Stricker and Orengo, 1995). For each colour channel (of which there are typically three, colour model depending), the three first moments are stored in the database – average, std. deviation and skewness.

Before histograms are made, it is necessary to decide which colour models to use (Lew, 2001). Different alternatives include RGB, HSB and CIELab. An useful comparison of several models is given in (Gevers and Smeulders, 1996). Based on previous work and my own evaluations of several models, I decided to use CIELab. This model is perceptually uniform so that the distance between two colours in the CIELab colour space corresponds to how different they appear to humans.

As for shape extraction, the ultimate goal is to be able to classify images based on the colour of the object depicted. Again, this classification was statically determined and stored in the database rather than classified dynamically at run-time. The CIELab colour space was divided into 18 different partitions and all the images in the Primus database were processed and assigned to a partition. Thus, for each image a single number was stored that determined partition number. This made the run-time analysis fast and easy to perform. The colour space partitioning is shown in Figure 9.5. The area inside the large X was divided into even smaller pieces (not shown).

---



**Figure 9.5 Partitioning of the CIELab colour space.**

### 9.1.4 Texture

Texture was the final image feature tested. It describes the structure of a surface, such as coarseness, contrast, directionality, etc. For a detailed discussion on the use of texture for content-based retrieval, see (Haralick et al., 1973) and (Lew, 2001).

My implementation used the method described in (Tamura et al., 1978). This is the same method used in, among others, the QBIC System (Flickner et al., 1995). The method is based upon a series of psychological studies as to how humans see texture. It classifies textures along a number of axis such as coarseness, line-likeness, regularity and roughness.

Unfortunately, preliminary evaluation of our implementation gave so poor results that texture was dropped from the final prototype. The results were for a large part nonsensical – it was difficult to gain an intuitive understanding as to why two objects were classified as having very similar texture. While the concepts of similar colour and shape are instantly understandable, it was difficult for test subjects to grasp what texture was all about.

## 9.2 Extending SESAM

After the feature extraction had been performed on all the images in the Primus database, all the images had been classified according to object shape and object colour. This made it possible to extend the Sesam prototype to include two new types of filters. It was discussed to combine colour and shape into a single filter, but it was decided that it was easiest to understand and more powerful to use if they were kept separate.

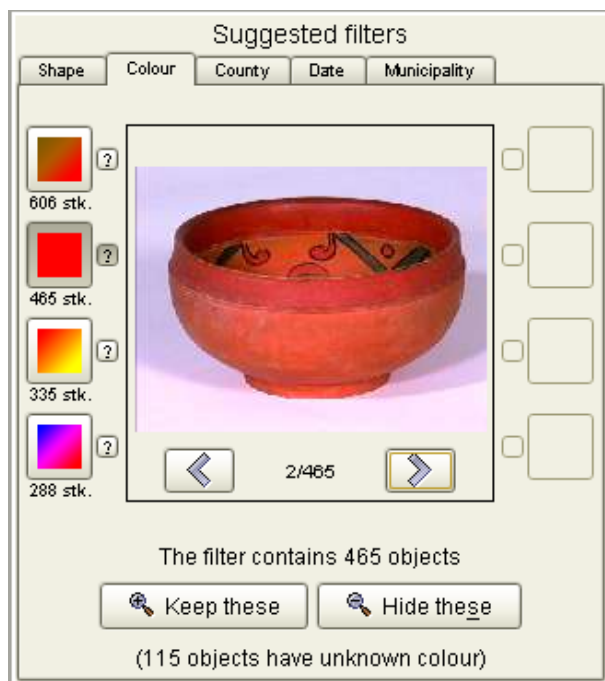
A screen shot of the new shape filter, is shown in Figure 9.6. Here the objects in the result (in this case drinking bowls) have been classified according to which standardized shape they most closely resemble. The resulting classes of images are then put through the same process as described in Section 8.1.2 in order to select the most promising subset of classes to display to the user and to compute a utility measure indicating the assumed usefulness of this filter.



**Figure 9.6** Shape filter interface.

In the example shown in Figure 9.6, the process ended up with showing four classes of objects to the user (of a maximum eight). These are shown as buttons on the left side of the interface. The user is then free to make a filter by pressing the buttons matching the shape class of interest. It is also possible to use the question-mark-buttons to browse the images in each class in order to gain a better understanding of exactly what the filter would accomplish. Finally, the images in the selected classes can either be retained (“Keep these”) or removed from the result (“Hide these”).

The corresponding interface for colour filters is shown in Figure 9.7. The interface component is similar to the one shown in Figure 9.6 except for the shape buttons having been replaced by buttons with icons indicating the corresponding partition of the CIELab colour space.



**Figure 9.7 Colour filter interface.**

## 9.3 Experiences

Preliminary results from informal evaluations during the development as well as from (Hauglid and Midtstraum, 2001) and several master degree student projects (Obrestad, 2002), (Langmyr, 2003, Grøtan, 2003), made it clear that the current implementation is not at a level where a larger, more formal evaluation could be useful. This section therefore is limited to discussing the informal experiences.

I have chosen to describe the experiences in two separate parts. First I discuss experiences from working on feature extraction. While this was not directly related to the topic of this thesis, it was a required sub component in order to implement a content-based image retrieval interface.

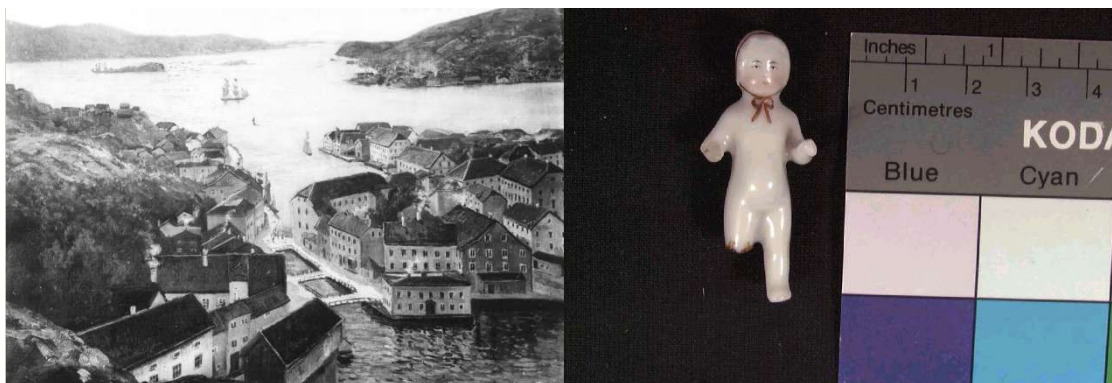
The second part discusses the experiences with the user interface. While the interface was inhibited by the less-than-perfect feature extraction, it was still possible to get some clear indications as to what worked and what did not.

I end this chapter with a discussion of how the different design ideas performed and which conclusions can be made.

### 9.3.1 Feature extraction

The overall experience gained from the work on feature extraction, is that this is a hard problem to solve. A lot of work went into trying out different techniques and algorithms, both for segmentation, feature extraction, distance measurements, etc.

Even for the images which consisted of a single object displayed against a uniform background, it was difficult to get a perfect result. If the image had colours similar to the background or if the shadow was particularly heavy, the outline was seldom 100 % correct. Primus also contained a lot of images where it was difficult even manually to find a useful segmentation or shape extraction. Figure 9.8 shows two examples.



**Figure 9.8 Examples of problematic images to segment.**

For the image on the left, it is not clear what is the object of interest and what is uninteresting background. For the image on the right, the object of interest (the tiny doll), is not the largest object in the image. The shape extraction is therefore likely to focus on the right-side object.



---

Feature extraction was problematic not only as far as techniques and algorithms go. As it is impossible to use computers to evaluate the different implemented alternatives (determining the correct solution is exactly what is the problem), a number of subjective evaluations using human test subject as evaluators were used. They were presented with how the different alternatives ranked a number of images according to similarity with a given image. Even if the test subjects were strongly instructed to evaluate the results purely on shape (or colour), not any other features, this was difficult to achieve in practice. By examining some of the results, it is clear that the test subjects had problems with not (sub-consciously) considering other aspects as well. For example, users could indicate that two depicted dolls had similar shape, even when they were (objectively) clearly different. Similar, it was not seldom that two entirely different objects with almost exact similar shape were recorded as quite different. This, of course, made it more difficult to evaluate the different implemented extraction methods. It also suggests that image features such as colour and shape perhaps are not that well suited for this kind of retrieval.

### 9.3.2 User interface

It proved difficult to fit an image filtering component into the existing SESAM prototype. The screen area needed to display even small images, meant that it was difficult clearly to show which images a filter affected. This problem was of course further amplified by an imperfect feature extraction.

Interestingly, clear differences between colour filters and shape filters were observed. Almost always the colour filters were found by test users to be more intuitive and to work better. In part, this can be attributed to the quality of the feature extraction, but separate evaluations of the feature extractors indicate that this could not have been the only cause. Rather, it seems like colour is *intrinsically* better suited for this purpose than shape. Upon closer examination, the most likely cause is the fact that the photo of a blue object will nearly always contain a majority of blue pixels. But for shapes, the shape extracted from an image will vary *depending on the angle in which the photo was taken*. For example, a bowl can appear circular from above, almost square from a 45 degree angle, and as a half circle from the side. This effect is illustrated in Figure 9.9.

---



**Figure 9.9 Rotating a 3D object alters shape of 2D projection.**

Experiences indicate that users (subconsciously) expected the query system to match objects according to their real-world 3D shape rather than the 2D shape present in the image.

This points to a more fundamental issue with the use of images in this context. In the SESAM prototype, images were just means to an end – a way of locating the objects depicted rather than the images themselves. This helps explain why test subjects were more interested in the real-world 3D shape rather than the 2D projection in an 2D image. Thus, content-based image retrieval would most likely have been more appropriate in a system directly focused on image retrieval.

Still, the image handing extension to SESAM proved of value for colour filtering – both for reasons described above and because it was less reliant on a perfect segmentation of object and background. Even if a lot of effort was spent to improve shape classification, it was in general not reliable enough to be worthwhile beyond a certain novelty factor.

### **9.3.3 The four design ideas**

The overall objective with the work presented in this chapter, was to test the four design ideas from Chapter 5 in the realm of image databases. Experiences with respect to each of these ideas are discussed below.

#### **Revised interaction model**

This design idea was only partly implemented for images. While filters made it possible iteratively to modify a result, no access methods beyond filtering were implemented due to time constraints.

---

---

As described above, only colour proved to be of much use for filtering. Thus, the iterative interaction model was less useful here as only one type of image filter was available. However, when combined with the textual filters from the original SESAM prototype, the experiences with this design idea were comparable to the results presented in Chapter 8.

### **Intra-result analysis**

The intra-result analysis made it possible to see the most prevalent object colours and shapes in a result. For example, it was easy to see that most of the drinking bowls in the database had a reddish colour.

As for all the other design ideas, the quality of the extracted information and thus the value of this idea was heavily affected by the feature extraction. For example, this meant that the colour information was far more reliable than the shape information.

### **Active user interface**

The user interface could be said to be active in that users could construct filters by recognizing the colour or shape they were interested in from a list of presented alternatives. No option of drawing a shape or selecting a colour from a complete palette was provided.

This design idea was perhaps the most successful of the four in this setting – at least discounting the cases where the feature extraction was particularly unsuccessful. Experiences indicate that the information needs of users are often not tied to a specific colour. They rather seem to recognize what they want when they see it. For example, by responding “are there more objects with this colour?” or “yuck! is there anything with a warmer colour?”. This sort of result manipulation is exactly what this design idea was meant to support. Thus, this idea showed promise with respect to narrowing the semantic gap.

### **Dynamic user interface**

This design idea concerned the process of selecting a suitable subset of filter suggestions and present those to the user rather than presenting every available option. The selection process used in this prototype used in essence the same algorithms as the prototypes presented in Chapters 6 to 8.

---

Experiences indicate that this selection process performed worse here than in the text-only case. Again, this can be attributed to the problems with extracting image features reliably. If images are incorrectly classified according to colour or shape, reasoning about the most suited classifications for filter suggestions will invariably be negatively affected.

---

---

## Chapter 10

# Savanta

---

This chapter presents a work with Savanta, a user interface for accessing temporal, semantic video annotations. In Savanta, various methods and paradigms have been integrated, including visualisation, filtering, analysis, navigation and search, in order to explore the possible advantages of doing so. A formal usability evaluation comparing Savanta to systems based on traditional interfaces for accessing video databases is presented. It concludes that Savanta outperforms them with regard to both power and usability, especially for complex and open-ended tasks.

The work presented in this chapter was done in cooperation with Jon Heggland. The purpose was to bring together Heggland's modelling and visualisation of temporal metadata and my design ideas for user interfaces for accessing information (as presented in Chapter 5). See Appendix B for a more detailed description of our individual contributions to Savanta.

### 10.1 Introduction

---

The use of temporal, rich media such as video and audio in computer systems is becoming more and more common. However, to benefit fully from the richness, power and verisimilitude of video, powerful tools for annotation, analysis and retrieval are needed. Systems for creating and storing semantic temporal annotations (high-level descriptions related to video/audio time intervals) are common, but less consideration has been given to how this information best can be utilised by the end user. In

most cases, techniques for accessing information developed for more traditional database applications, such as query languages and form-filling interfaces, are used.

In this chapter, we introduce a novel approach for accessing information in such temporal annotation databases, namely a seamless fusion of visualisation, browsing, filtering, searching and context-aware metadata analysis. This is based on our position that the properties and requirements of temporal annotation databases are significantly different from traditional databases, and that introducing modern user interface techniques – direct manipulation, graphical visualisation, filtering, iterative interaction model – will be of great benefit. To evaluate the validity of this claim, we have implemented an advanced information access application, and have performed an evaluation comparing it with user interfaces based on traditional methods and paradigms.

The remainder of this section presents related work and how we intended to improve the situation with regards to information access in temporal metadata databases. Section 10.2 presents Savanta, a prototype designed to evaluate our design ideas. Section 10.3 presents a usability evaluation of Savanta, while Section 10.4 presents conclusions.

### **10.1.1 Related work**

There is a huge difference between the way video data is coded digitally, and the way it is experienced by a human user. As far as a computer is concerned, video data is sequences of bitmap images – grids of coloured dots – along with a time-dependent air pressure function, i.e. the audio track. The users, however, do not see a bunch of dots constantly changing colour – they see people, buildings, vehicles, landscapes, places, actions, events, discussions, stories, ideas, etc. Most likely, the users want to interact with a digital video library using such high-level concepts, not using the vocabulary of coloured dots and air pressure. Therefore, it is common to use high-level, structured metadata to describe the semantic content of videos. In this way, the knowledge contained in and related to the material is stored in a machine-readable, compact format, and can be used for a multitude of purposes – for indices, tables of contents, references and links; comments, explanations, summaries, critique and augmentation. This enhances the value of the material considerably, and reduces the need for handling large media files, which is expensive both in terms of storage space, viewing time, network bandwidth and processing cycles.

---

---

Data models and systems for generating and storing such annotations have been well researched. Systems such as OVID (Oomoto and Tanaka, 1993), AVIS (Adali et al., 1996), VideoSTAR (Hjelsvold and Midtstraum, 1994), the Algebraic Video Model (Weiss et al., 1995), Vane (Carrer et al., 1997), Smart VideoText (Kokkoras et al., 2002), OntoLog (Heggland, 2002) and BilVideo (Dönderler et al., 2003) present clever, expressive data models suitable for most purposes. Though some find it lacking in flexibility, usability and expressiveness (Nack and Hardman, 2002), (Nack and Putz, 2001), the MPEG-7 metadata standard is also gaining acceptance. However, less thought has been given to how to retrieve, present, browse and mine this kind of information.

This task is markedly different in temporal annotation databases compared to other databases, as noted in (Santini and Jain, 1999) and (Nack and Hardman, 2002). In traditional relational databases storing text and numbers, the data model is fixed and unambiguous, relations are explicit, and the access patterns are straightforward and predictable – you know exactly where to look to find what you want. The meaning of the data is inherent in the data itself – it is not a matter of interpretation and context. In contrast, video and audio data are (as mentioned previously) opaque to the computer when it comes to semantics, so it must rely on metadata to implement content-based access. However, though the metadata model may be straightforward, the richness and ambiguity of temporal media mean that the actual annotations may vary greatly, depending on the idiosyncrasies of the annotator (whether it is human or not). Differences in perspective, focus, level of detail, accuracy and thoroughness may cause the same video content to be represented quite differently, even in the same model. Therefore, precise query languages may be of lesser use than in “old-fashioned” databases – they assume that the user knows exactly what he or she is looking for, and that the queryable metadata comprises a complete and accurate representation of the semantics of the video. And for the same reasons, browsing and navigation systems may be more useful and appropriate.

As for the previously mentioned metadata systems, OVID, AVIS and VideoSTAR offer complex query languages. They are precise and powerful, but not very easy to use, even with forms interfaces. Smart VideoText and BilVideo are logic-based, and use logic programming for querying; this makes them perhaps even more powerful but further out of reach for non-experts. OVID, Vane and OntoLog use visualisations of the annotation intervals for overview and navigation, but are limited to viewing one media resource at a time – inter-media browsing is not

---

provided. Smart VideoText and the Algebraic Video Model suggest a hyperlink paradigm for navigating along semantic links between different intervals of different films, but do not actually implement it, or even specify a concrete user interface. Text-based, tabular overviews of the video contents are used in VideoSTAR and video analysis tools such as QMA (Skou, 2003) and Observer (Noldus Information Technology, 2003). These last two also provide a filtering mechanism, where uninteresting annotations may be hidden to de-clutter the display. They also offer statistical analyses on the annotations. Informedia (Christel and Martin, 1998), SCAN (Whittaker et al., 1999) and VoiceGraph (Oard, 1997) offer information retrieval functionality by searching in video transcriptions, but present the results as ranked lists of videos, with little support for browsing, navigation and exploration.

As is evident by the above description, a number of different methods for accessing information in temporal annotation databases can be used. However, few existing systems implement more than one or two different methods and the implementations tend to target expert users. In this chapter we present a system where modern interface techniques are used to provide an integration of several different access methods. We argue that such a system will not only offer more expressive power to the user, but also a more user-friendly for non-experts – a user group which, in our opinion, has been somewhat neglected by most existing systems.

### **10.1.2 Applying the design ideas to temporal multimedia annotation databases**

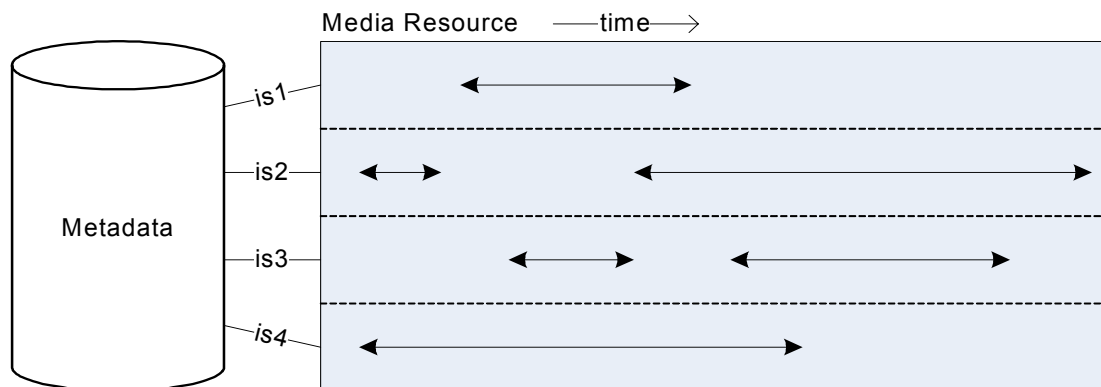
In this section, we examine how the four design ideas from Chapter 5 are used in the design of our proposed system. As this type of information repository is more complex than for example textual metadata databases (mostly due to the added temporal dimension), it is to be expected that this interface will be more complicated than the interfaces presented in previous chapters. Our aim is therefore to design an interface which compares favourably with respect to usability and expressive power to the related interfaces mentioned in the previous section, not to user interfaces for information access in general.

#### **Revised interaction model**

Temporal annotation databases can be seen as consisting of one or more *media resources* – in our case videos, but could also include e.g. audio clips. A conceptual model for the description of a media resource is shown in Figure 10.1.

---





**Figure 10.1** Conceptual model of temporal annotation databases.

The contents of the media resource can be described by attaching metadata (annotations) to interval sets (is1–is4 in the figure) containing one or more intervals.

In order for the user to see this information, some sort of visual presentation must be available. The process of creating such a presentation is called *information visualisation* – more formally defined as (Gershon et al., 1998):

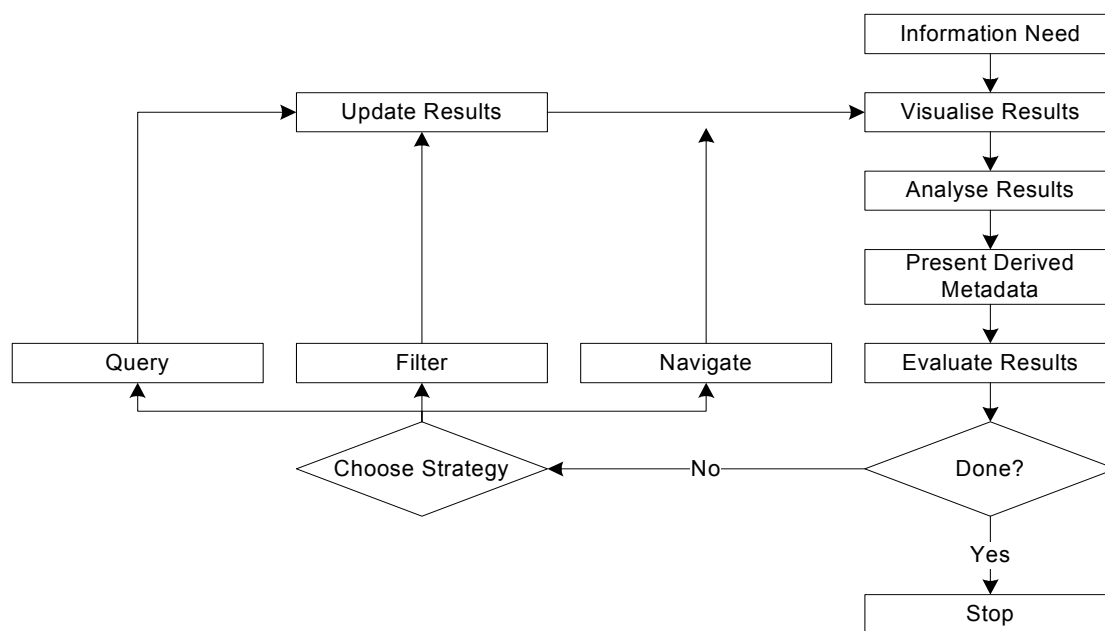
“[...] the process of transforming data, information, and knowledge into visual form making use of humans' natural visual capabilities.” (page 9)

Depending on the size and complexity of the database in question, a large number of different presentations can be imagined – from the detailed display of a single piece of metadata to an overview of the whole database. Limiting a system to a single type of presentation would clearly be too restrictive. Some form of *navigation* between the different presentations is therefore required.

As the media database grows, it becomes necessary to be able to limit the presentation to a subset of the stored data. This is both because displaying everything would make the presentation too cluttered and because not every piece of information is equally relevant to a given user in a given setting. Generating a subset of the database can be done in a number of ways. The most common method is to use some sort of *query language* or *forms interface*. Based on one or more expressions entered by the user, a new ad-hoc collection of matching objects (or *query result*) is constructed.

*Filtering* is an alternative solution where the user removes uninteresting objects from the information collection by creating filters that eliminate objects that do not (or do) match certain criteria.

The integration of visualisation, query, filtering and navigation in a revised model of interaction (see Section 5.1) as illustrated in Figure 10.2.



**Figure 10.2 Interaction model for proposed system.**

However, the integration of several methods for accessing information will result in a more comprehensive interface which could become too complex and therefore difficult to use. In other words, the gain in power could be overridden by a loss of usability. This is a key point we address in the usability evaluation presented in Section 10.3.

### Intra-result analysis

A system for accessing temporal media databases needs not be limited to the data that is explicitly stored in the database. It is typically possible to *analyse* the complex temporal relationships present in the metadata to derive new, high-level information that otherwise might be difficult and time-consuming to extract manually (see Section 5.2). This can include finding statistics such as the average length of registered intervals or identifying properties such as the most prevalent metadata in a given result.

### **Active user interface**

In addition to being valuable information in their own right, results of a temporal analysis also lend themselves to navigation and filtering actions, thereby providing a positive synergy and making it possible to make an interface which relies more on recognition than recall (see Section 5.3). The rich nature of temporal media could very well make such derived information more important here than in traditional database settings.

### **Dynamic user interface**

The temporal analysis can also be made context sensitive with respect to which presentation the user has navigated to. This will result in a dynamic user interface (see Section 5.4).

## **10.2 Design and implementation**

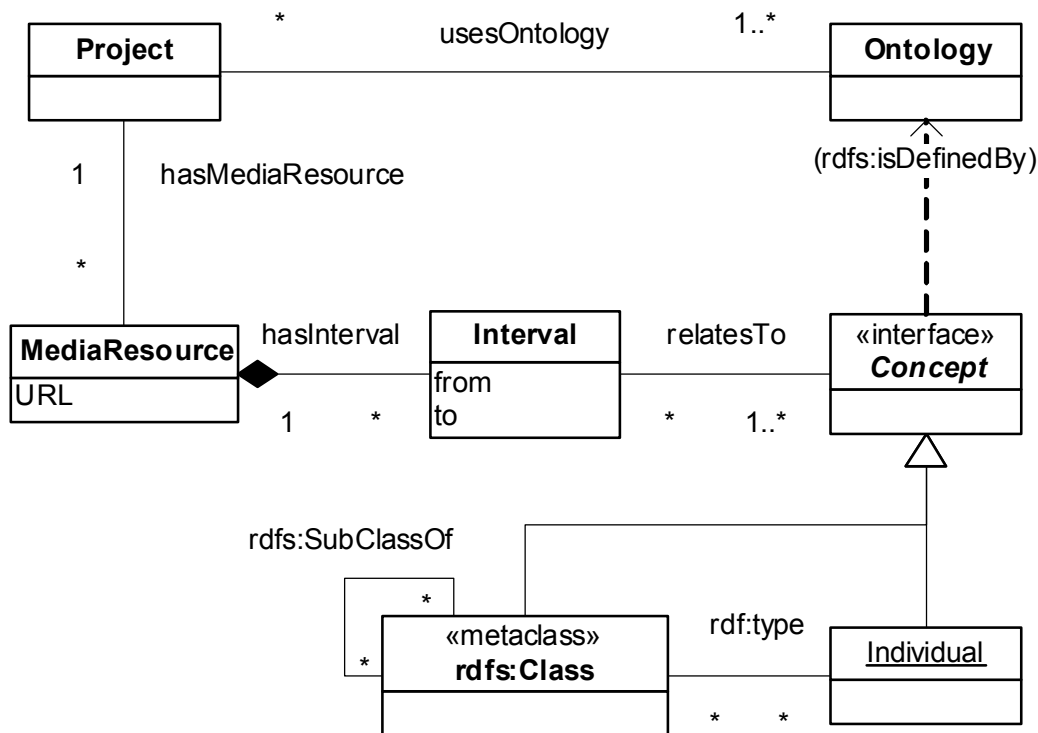
---

To test these ideas in practice, we built a prototype called Savanta (Search, Analysis, Visualisation And Navigation of Temporal Annotations). This section describes the implemented prototype, organised according to the aspects discussed in the previous section: stored and derived metadata, visualisation, navigation, filtering and querying.

### **10.2.1 Stored metadata**

We use the OntoLog model for temporal metadata, first described in (Heggland, 2002), because this provides us with a flexible, expressive model that lends itself well to visualisation and analysis. This RDF-based model, as shown in Figure 10.3, consists of *intervals*, with start and end time, that belong to *media resources*. The intervals are related to *concepts* in *ontologies*. The concepts are organised in directed acyclic graphs of classes, subclasses and instances. This constitutes a stratified annotation scheme (Smith and Pincever, 1991), with the strata (that is, the concepts) organised in a generalisation/specialisation hierarchy.

---



**Figure 10.3 The OntoLog data model.**

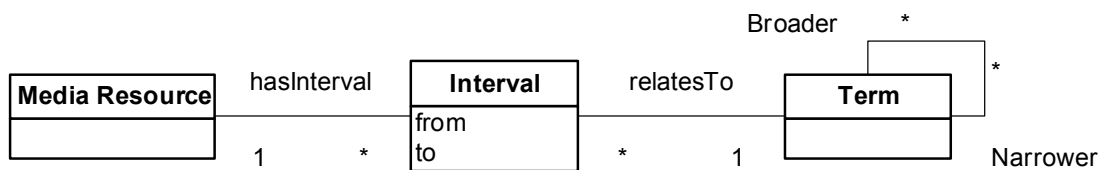
*Projects* act as containers for sets of related media resources and the ontologies that describe them. Additionally, the ontologies define *properties* that are used to describe the elements in the data model.

This is a fairly complex model; the ontologies in particular, with their classes, instances and properties, are potentially confusing for non-expert users. It is reasonable to target this kind of system at users who do not necessarily have a thorough understanding of data modelling – it ought to be possible to use it as a retrieval interface for a digital video library without requiring a lot of training on the users' part. Therefore, we choose to use a simplified view of the OntoLog data model in the Savanta user interface:

- ◆ We do not consider the differences between classes and instances very significant, so we present them as the same thing, and call them *terms*. Thus, both the subclass-of relationship and the instance-of relationship are considered a narrower-term relationship.

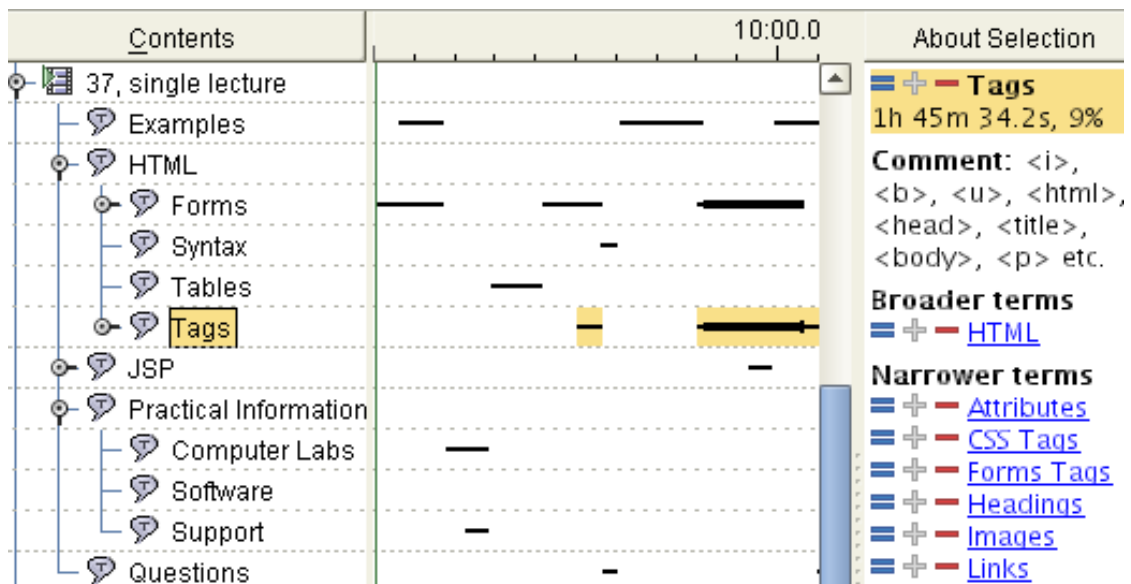
- ◆ Likewise, we consider ontologies top-level terms. That they also define properties is not mentioned, though we of course still use the properties for describing things.
- ◆ We look at one project at a time. The media resources within a project are most likely closely related and homogeneously annotated – they are described using the same ontologies and terms. This will both simplify the data model and better the chances of relevant analyses of the metadata.

This results in the following simplified view of the OntoLog model:



**Figure 10.4** Simplified annotation model.

Stored metadata is presented in Savanta as shown in Figure 10.5. Terms are presented in a tree list, with their associated intervals in a timeline display to the right. The terms, their “relatives” and their properties are also shown in a navigable hypertext panel at the right edge of the window. The user interface is further described in the subsequent sections.



**Figure 10.5** Stored metadata in Savanta.

## 10.2.2 Derived metadata

A temporal annotation database contains much more information than what is explicitly registered. Consider a film clip where the intervals in which each actor appears, have been accurately registered. Implicitly, this database also contains information about the frequency of each actor and the temporal relationships between each of them. Examples of the latter might include actors that always appear together, actors that are always alone, etc. In order to extract such information, some sort of intra-result or temporal analysis is necessary (see Section 5.2 for a description of the intra-result analysis design idea).

While intervals are the fundamental units in the data model, semantic information is nearly always attached to sets of intervals – simply because annotating each separate interval is too time-consuming. As a result, a meaningful simplification is to derive information about sets of intervals rather than individual intervals. Four different, meaningful, types of interval sets can be imagined in our system. The set of intervals,  $T_{term}$ , attached to a specific *term*, is perhaps the most important. Other sets include the whole database  $D$ , the currently displayed result  $R$  and the set of intervals,  $S$ , selected by the user.

First of all, meaningful information can be derived about a single set of intervals. For the system described here, we have chosen to display the total length of the intervals in  $D$ ,  $R$  and  $S$ , both in seconds and relative to the total length of intervals in the database. As  $T_{term}$  is a function of *term*, displaying such information about every (non-selected) term would simply be too overwhelming and confusing.

Temporal analysis gets much more interesting when we start to examine two sets of intervals and how they relate to each other. If we see an interval, and thus a set of intervals, as a set of points in time, we can use standard set operators as *temporal set operators* (Hjelsvold et al., 1995) to manipulate the intervals. This includes operators such as union ( $\cup$ ), intersection ( $\cap$ ) and subtraction ( $-$ ).

Further, we can examine how two intervals, A and B, relate to each other on a common temporal axis. For example, A might have ended before B starts, they might be equal, or A might start exactly when B ends. In fact, 13 such *temporal relations* have been identified (Allen, 1983). While temporal set operators return one or two intervals, temporal relations simply say whether a given relation exists or not between two intervals.

---

---

With a multitude of temporal set operators and possible sets of intervals to look at, a large number of possible computations exists. It is therefore necessary to focus on what gives results that are meaningful to the user. Our novel idea is to use temporal analysis to identify interesting terms by examining  $T_{term}$  in relation to  $D$ ,  $R$  or  $S$  using temporal set operators. The possibilities offered by this method, can be illustrated by a few examples:

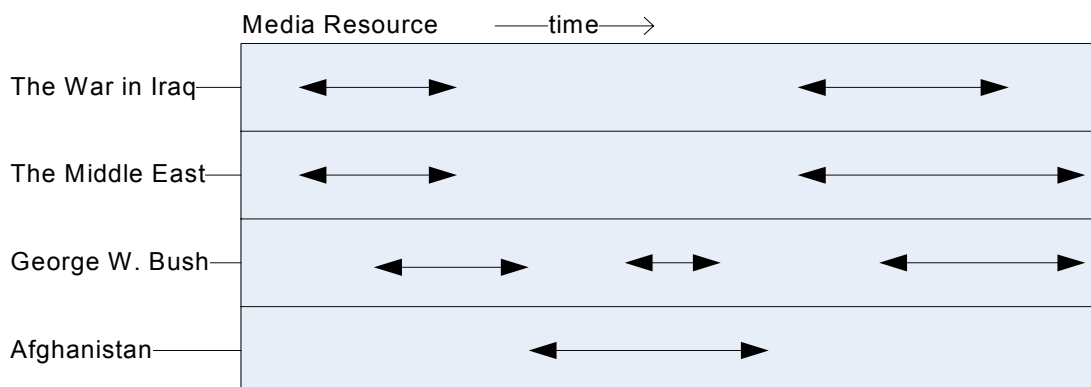
- ◆ Identify terms that have most in common with the result. This is terms that have intervals that overlaps as much as possible with the intervals in the result.
- ◆ Identify terms that have little or nothing in common with the result.
- ◆ Identify terms that have a lot in common with the selected intervals, but little in common with the result.
- ◆ Identify the media resources that topically are most similar to the intervals selected by the user.

These terms can represent interesting information in their own right as well as serve as input for filtering operations (example: remove all intervals attached to an identified term). For this implementation, we have chosen to group terms into three semantic categories. These three categories are:

- ◆ Described by
- ◆ Related to
- ◆ Differs from

An illustration of these three categories used in the following discussion is shown in Figure 10.6.

---



**Figure 10.6** The War in Iraq described by The Middle East; related to George W. Bush; differs from Afghanistan.

Please note that all interval sets can contain intervals from different media resources even if the figure above (for clarity) shows only one. For example, in a database containing media resources  $M_1, M_2, \dots, M_n$ ,  $R$  will equal  $R_{M1} \cup R_{M2} \cup \dots \cup R_{Mn}$ . At any time when an operator is applied to two sets of intervals, intervals from a given media resource are handled separately. This means that  $A \cup B$  is a shorthand for  $\{A_{M1} \cup B_{M1}; A_{M2} \cup B_{M2}; \dots; A_{Mn} \cup B_{Mn}\}$ .

### Described by

This category includes terms that should give a good description of the result or the currently selected intervals. To locate such terms, we identify terms where  $T_{term}$  overlaps  $R$  (or  $R \cap S$ ) to as large extent as possible. In Figure 10.6, “The Middle East” intervals overlap all “The War in Iraq” intervals and is therefore a good candidate for this category. The reasoning is that the degree in which two sets of intervals are equal, closely matches their semantic relation and thus one can be used to describe the other.

### Related to

“Related to” encompasses terms that are somewhat related to the result or the currently selected intervals. In practice, this means terms which have intervals that overlaps  $R$  (or  $R \cap S$ ) as close to 50 % as possible. For example, the “George W. Bush” intervals in Figure 10.6 overlap close to 50 % of “The War in Iraq” and they are therefore thought to be related. The main purpose of this category is to serve as source for filtering operations. For this use, it makes sense to find ways of reducing the result by 50 % – as described in Section 6.3.2 and Section 8.1.2.



### Differs from

To describe an object fully, you not only need to include its properties, but also the properties it does not exhibit. This is covered by this category. In our case, it contains terms that have little or nothing in common with  $R$  (or  $R \cap S$ ) – that is as little overlap as possible. In the example shown in Figure 10.6, the “Afghanistan” interval does not overlap any of the “The War in Iraq” intervals and thus is completely different. This category is also helpful for filtering as described in Section 10.2.5.



To find how well a given term is suited to each of the three categories mentioned above, we employ a set of utility functions. They take  $T_{term}$  as argument and give a result from 0 (no utility) to 1 (high utility), which should indicate the usefulness of term with respect to a given category. The utility functions for each of the categories are given below.

**Described by (term):**

$$\frac{\sum_{\forall M} |T_{term_M} \cap R_M|}{\sum_{\forall M} |R_M|}$$

This is the length of the overlap between  $T_{term}$  and  $R$  totalled for all media resources, divided by the total length of  $R$ .

**Related to (term):**

$$4 \cdot \frac{\sum_{\forall M} |T_{term_M} \cap R_M|}{\sum_{\forall M} |R_M|} \cdot \left( 1 - \frac{\sum_{\forall M} |T_{term_M} \cap R_M|}{\sum_{\forall M} |R_M|} \right)$$

This is  $4 \cdot x \cdot (1-x)$  where  $x$  is the “described by” utility function. This gives a parabola function where the utility is maximized for 50 % overlap between  $T_{term}$  and  $R$  and minimized for 0 % and 100 % overlap.

**Differs from (term):**

$$\frac{\sum_{\forall M} |T_{term_M} - R_M|}{\sum_{\forall M} |M - R_M|} \cdot \left( 1 - \frac{\sum_{\forall M} |T_{term_M} \cap R_M|}{\sum_{\forall M} |R_M|} \right)$$

This function consists of two parts. The first is the total length of the intervals in  $T_{term}$  which are not in  $R$ , divided by the length of the intervals not in  $R$ . This promotes terms with  $T_{term}$  containing as much of what is not in  $R$  as possible. The second part is 1 minus the “described by” utility function. This promotes terms with  $T_{term}$  containing as little of  $R$  as possible.

### 10.2.3 Visualization

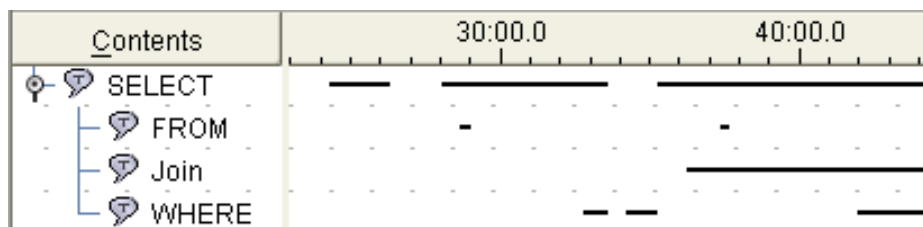
The Shneiderman mantra (Shneiderman, 1997) of

*“overview, zoom and filter, then details on demand.”* (page 523)

is a good rule of thumb when designing interaction systems, and the process of visualisation ties into this in several ways. The system must construct the overview, produce the details on demand, and provide an environment in which the processes of zooming and filtering can be performed efficiently and intuitively.

The main challenge of creating an overview is the amount of data that must be presented. During the development and testing of this system, we created a sample database containing ten weeks of lectures in an entry-level computer science course; this amounts to 20 media resources, over 150 terms and over 500 intervals. While not much in terms of bytes, this is much information for a human to assimilate quickly, especially since much of the useful information is given by (possibly implicit) relationships – which terms are “active” at the same time as others, and how they are connected in the broader/narrower term hierarchy.

To create the overview, we used an extension of the technique introduced by OntoLog (Heggland, 2002). The terms are presented in a tree list – a natural choice, considering their near-hierarchical organisation – and the intervals related to each concept is visualised as segments on a timeline, as shown in Figure 10.7.



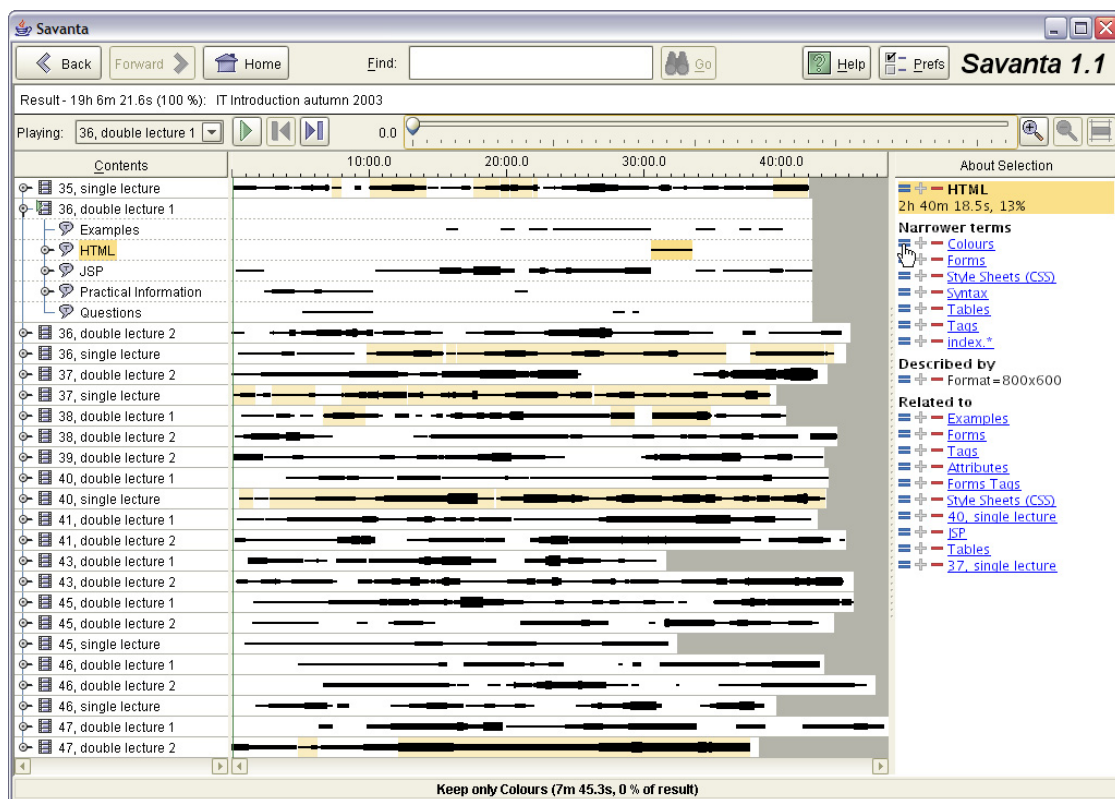
**Figure 10.7** Terms, timeline and intervals.

When a term is collapsed, the intervals of its “children” (narrower terms) are aggregated, so that the representation of the parent term’s intervals includes the intervals of its children, using varying line thickness to denote overlap between the children’s intervals, as shown in Figure 10.8.



**Figure 10.8** Collapsed term and aggregated intervals.

We use the media resources as top level nodes in the tree list, and present the term tree under each media resource node. Initially, all the nodes are collapsed, and this creates a representation compact enough to present our sample database on a single screen, as shown in Figure 10.9 (except that there, a single media resource node is expanded).



**Figure 10.9** Savanta.

Terse as this representation is, it is still possible to extract useful information from it. You get a list of the media resources in the database, and their relative lengths are readily apparent. It is also easy to see which media resources are heavily annotated and which are not, by noting the number and thicknesses of the interval lines, and where apparently nothing interesting happens, judging by the gaps in parts of the timeline.

“Details on demand” is handled in several ways in this visualisation. Selecting an item (a media resource or a term) displays information about it in the hypertext panel to the right. This includes both stored and derived metadata. The stored metadata consists of the term’s (or media resource’s) properties, comment, and broader and narrower terms (if applicable). The derived metadata includes the length of the current selection, as well as the terms, properties and media resources related to the current result/selection, as discussed in the previous section. The hypertext panel also provides controls for navigation and filtering; more on this in the next sections.

Selecting a term also highlights the intervals it is related to in the interval display. Figure 10.9 also illustrates this, with the term “HTML” selected. Thus, it is easy for a user at a glance to find out when and how much a term is used.

Another mechanism for providing details (or “zooming”), is to expand and “drill down” in the tree list. Due to the semantics of the term hierarchy and the aggregation of related intervals, the visualisation at each level of the tree is useful: Even if no intervals are related directly to a given term, the intervals related to all of its narrower terms are displayed (see Figure 10.8 and Figure 10.7). If you want more details – exactly what aspects of JSP (Java Server Pages) are discussed in different parts of the expanded media resource in Figure 10.9, for example – you simply expand the node. To avoid unnecessary clutter, only terms that are actually used in a media resource (or whose narrower terms are used) are displayed in the tree; but if desired, all the defined terms can be shown for completeness.

## 10.2.4 Navigation

Navigation is defined in (Baeza-Yates and Ribeiro-Neto, 1999) as

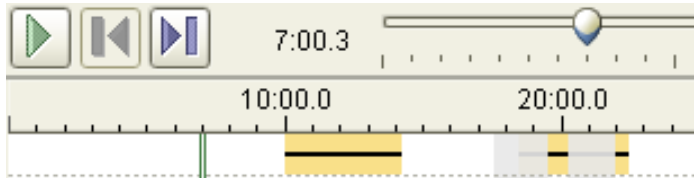
*“[...] following a chain of links, switching from one view to another, toward some goal, in a sequence of scan and select operations.” (page 265)*

Savanta’s navigation capabilities are designed to offer the user relevant, context-sensitive links to related information. The selection, filtering and analysis mechanisms of Savanta directly affect and interplay with the navigation system.

For navigation through a media file, Savanta offers the traditional play/pause button and time slider, illustrated in Figure 10.10. In addition, two skip buttons are keyed to the currently selected part of the current result

---

( $S \cap R$  or  $R$ , depending on whether anything is selected), so that navigating to an interesting media interval is a one-click operation. As default, only the current result ( $R$ ) can be played so that data relevant to the user is not obscured by what he or she has decided to filter away.



**Figure 10.10** Media navigation controls.

Navigating the collection of terms and media resources can be done by manipulating the tree list on the left. This is a common and intuitive method for displaying hierarchical data, familiar from for example Windows Explorer. This also affects the configuration of the interval display – expanding nodes shows more detail, while collapsing them gives a greater overview.

An alternative method is to use the hypertext panel on the right, shown in Figure 10.11.



**Figure 10.11** Hypertext navigation panel.

This panel displays the selected term or media resource (in this case “Style Sheets (CSS)”) within its immediate context – narrower and broader terms. It also shows the terms and media resources related to the current selection, according to a context-sensitive temporal analysis. These are grouped into three categories – described by, related to and differs from – as described in the following section.

All items in the panel are clickable, so the users can easily browse both the term hierarchy and the items related to the current result, thus creating dynamic paths through the material relevant to them.

### 10.2.5 Filtering

As the default presentation in Savanta includes all the registered intervals in the database, it often contains items of little interest to the user. In order to make it possible to focus on relevant pieces of information, some means of eliminating unwanted items are required. In Savanta this can be done either by filtering or searching. These techniques differ in that filters are constructed by direct manipulation using interface gadgets, similar to the dynamic query approach presented in Section 4.2.6, while searching requires the user to input a textual query.

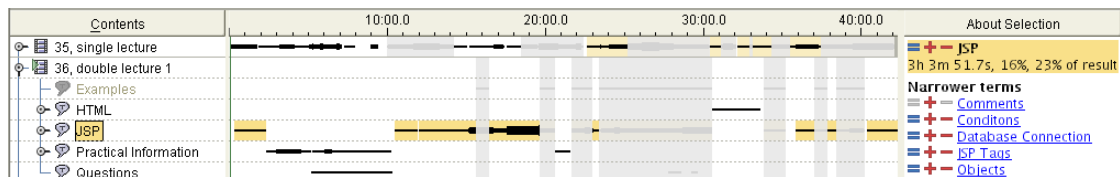
A filter, once constructed, is a set of intervals that changes the subset of the database displayed to the user (i.e. updates the result). In theory, a filter could contain a set of arbitrary intervals. In practice, we have limited the filters to contain intervals connected to a term or a media resource. This makes sense as terms and media resources are the basic semantic units. In addition, these units are also the output of the temporal analysis. A positive synergy between analysis and filtering can therefore be achieved. In this way, a user can construct a filter simply by selecting a term or media resource.

As both results ( $R$ ) and filters ( $F$ ) are sets of intervals, we can also here use set operators. Traditionally, a filter eliminates items from a result, that is  $R \leftarrow R - F$ . This requires the user to select what he or she wishes to remove. However, previous experiences (see Chapter 7) indicate that users when given the choice, are more likely to construct filters based on what they wish to keep rather than what they wish to remove. This suggests that it also should be possible to make filters that retain only what is common to both the result and the filter, or  $R \leftarrow R \cap F$ . Finally, to alleviate the problem of filters only improving precision and not recall,

---

we have also made it possible to make filters that add the contents of the filter to the result, or  $R \leftarrow R \cup F$ . Note that this breaks with the normal semantics of filters.

These three filter operations are respectively called remove (-), retain (=) and add (+) in the user interface. Filters are constructed by selecting the appropriate button on the right side of the interface as illustrated in Figure 10.12. For example, clicking the red + button beside “Comments”, will add everything about comments to the current result.



**Figure 10.12 Filters.**

The filter buttons are disabled if the corresponding filter will not change the contents of the result. Interestingly, this in itself provides yet another way of gaining information about the result. For example, if an add-button for a given term is disabled, this means that the intervals connected to this term are already present in the result – that is,  $R \cup T_{term} = R$ .

Filters are visualised as “greyed-out” areas in the interval display, as illustrated in Figure 10.12. The skip buttons discussed in the Navigation section are aware of the filters, making it easy to navigate to unfiltered, selected intervals. By default, playback of the selected media clip automatically skips filtered intervals. Thus, playing the intervals related to a term (or boolean combination of terms) is a one-click operation.

### 10.2.6 Searching

The searching part of Savanta allows the user to enter a textual query. This query is then matched against all textual attributes of terms and media resources. The intervals connected to these items are collected and used to modify the result. To retain the standard semantic of queries, the revised result is constructed as an intersection between the current result and the set of matched intervals. The interface components related to searching are visible at the top of Figure 10.9.

## 10.3 Usability evaluation

---

This section presents a usability evaluation carried out using Savanta and two other comparable interfaces. The first part presents the setting – the overall purposes of the evaluation, the interfaces to be tested and the evaluation design. The results from the evaluation are presented in the second part, while the third and final part discusses the results and draws conclusions about the usability of Savanta and the usefulness of our approach.

### 10.3.1 Setting

The purpose of this evaluation is to seek answers to the following questions:

- ◆ Is integration of several methods for accessing information a good idea? Does it provide significant benefits compared to interfaces that specialise in a single method?
- ◆ What is the relative importance of simplicity versus power? Can a powerful but complex system outperform a simpler one with regard to user satisfaction, without significant amounts of user training?
- ◆ How does different information access tasks affect the suitability and capability of such interfaces? What, if anything, changes depending on whether the tasks are simple or complex, specific or open-ended?

As the purpose of the evaluation is to find the strengths and weaknesses of Savanta, a comparative evaluation with other interfaces for accessing temporal annotation databases was a natural choice. By comparing Savanta with other interfaces, one can gain a clearer view of the merits and the weaknesses of the design. We chose to compare Savanta with applications based on two common paradigms for database search:

- ◆ Information retrieval as used by web search engines, where search terms entered into a single text field produce a ranked list of documents. This has been used in video databases by e.g. SCAN, InforMedia and VoiceGraph.
  - ◆ Construction of boolean query expressions using forms, with an unranked list of matching intervals as the result. This approach is espoused by e.g. VideoSTAR, Algebraic Video and OVID.
-



We chose to create our own implementations instead of using existing systems. That way, we would be able to query the same database with all three interfaces, and they would be quite similar in look, feel and polish. This would help the test subjects focus solely on the paradigm differences, not on database differences or implementation details.

### **Savantoogle**

Savantoogle, like its name suggests, is supposed to resemble Google<sup>1</sup> and similar search engines in looks and functionality. These search engines (Google, AltaVista<sup>2</sup> and AllTheWeb<sup>3</sup> being the most popular) are familiar to most Internet users today, and are very easy to use. They consist of a single text field where the user can enter one or more words or phrases; executing the query then produces a list of documents, ranked according to how well each document matched the word(s) entered by the user. In a multi-word query, words can be prefixed with a '+', denoting that the word must be present in the resulting documents, or a '-', meaning that the word must not be found. Figure 10.13 shows Savantoogle's user interface.

- 
1. Google Inc., <http://www.google.com>
  2. Overture Services Inc., <http://www.altavista.com>
  3. Overture Services Inc., <http://www.alltheweb.com>
-

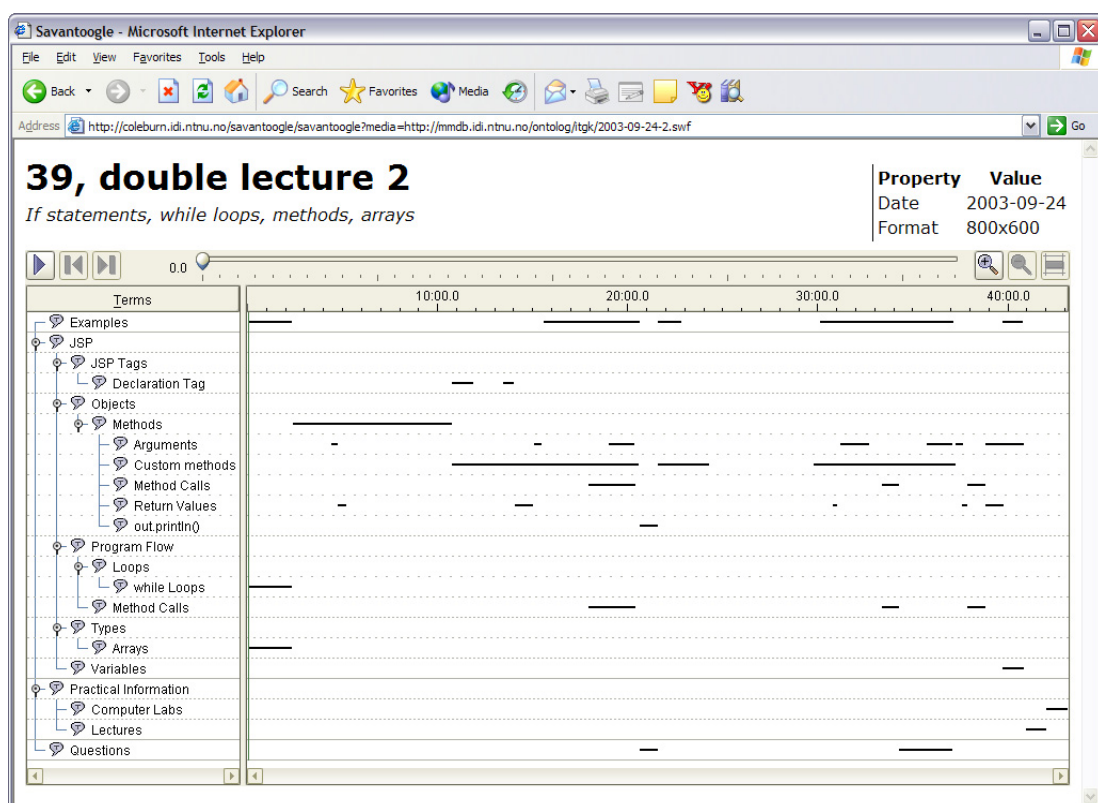


**Figure 10.13 Savantoogle.**

In the web search engines, the documents comprising the result are web pages, and the search is based on the occurrence of words in each document. In Savantoogle, this is slightly different, with media resources taking the place of HTML documents. The query words are matched against the terms, the terms' properties and the media resource titles and comments for each media resource. The ranking is mainly based on how much time is spent on each matching term – analogous to how web search engines rank documents according to the number of occurrences of the search terms. Media resources matching several or all the words in a multi-word query are ranked higher than those matching only a few. Matches in media resources, titles or comments also increase the rank; this corresponds to how web search engines consider matches found in headings more important.

Clicking a search result link (which in the web search engines brings you to the corresponding web page) brings you to an applet visualising the annotations belonging to the media resource in question, shown in Figure 10.14. This resembles the term tree and interval display of Savanta, but has noticeably less functionality. It is meant to correspond to the plain display of a web page in the web search engines, and to the relatively simple video browsing functionality of systems like VideoS-TAR, Vane and Jabber (Kominek and Kazman, 1997). Most notably, the hierarchical aggregation of interval lines is not employed; instead, the term tree is initially completely expanded. Another limitation is the fact that it shows only one media resource at a time. Video playback and skipping work like in Savanta, but there is no filtering or analysis functionality.

Savantoogle's search engine is implemented as a Java servlet, while the media resource browser is a Java applet. Both can be run from a web browser supporting Java 1.4, and both utilise the same basic data structures as Savanta.



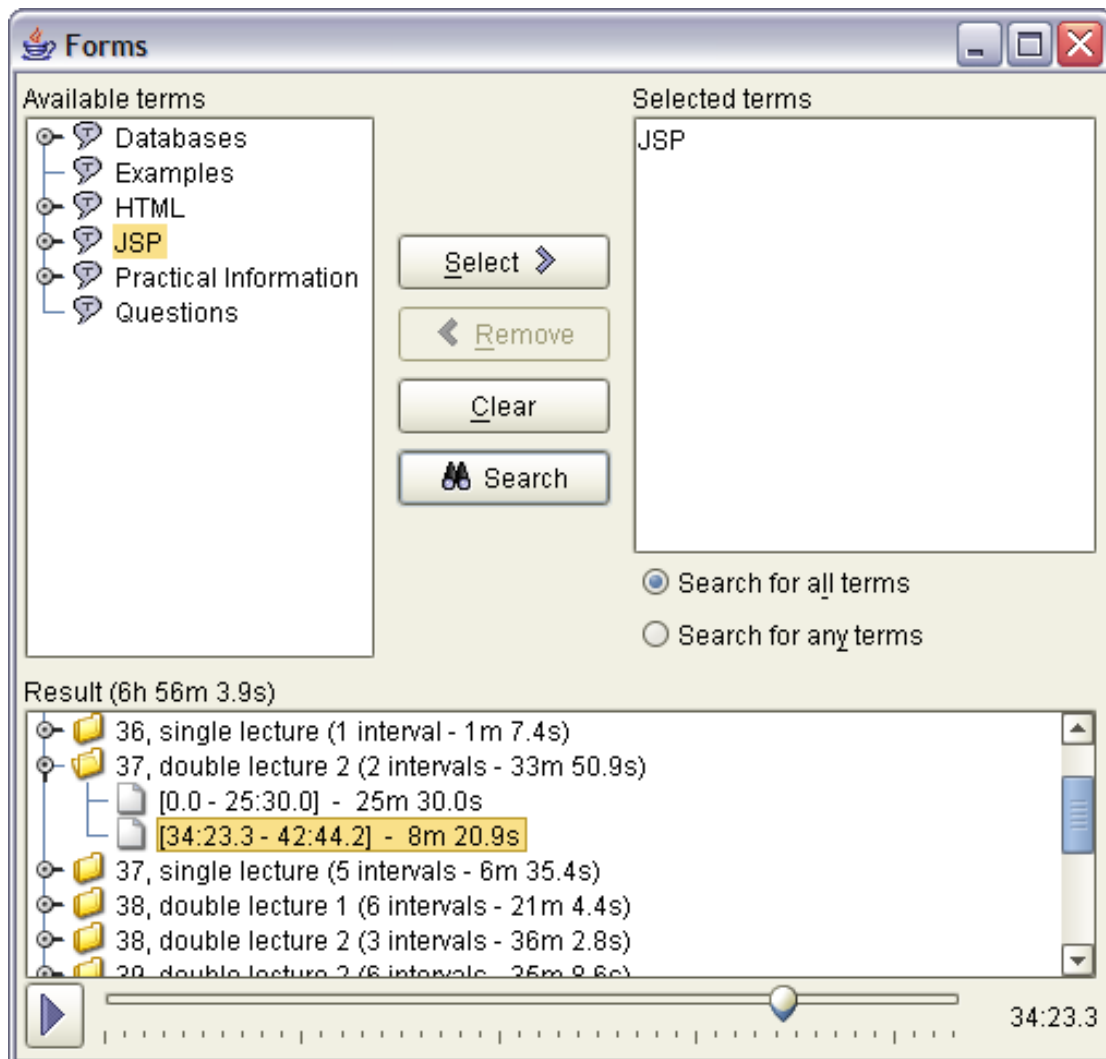
**Figure 10.14 Savantapplet, Savantoogle's interface for presenting a single media “document”.**

## Forms

Forms is built on a quite different search paradigm. Many video database systems, such as OVID, VideoSTAR, AVIS and Algebraic Video define query languages for access to their data structures. These are not very user-friendly, so graphical user interfaces are created that use forms to construct query expressions. The complexity of these interfaces varies, according to the complexity of the underlying data model, but the interfaces have the same basic interaction model: Select or enter query terms, use Boolean and/or temporal operators to combine them, and view the result in an unranked, textual list.

The Forms interface, shown in Figure 10.15, adapts this paradigm for the OntoLog/Savanta data model. The terms are shown in a tree list in the upper left of the window. By selecting term, and then pressing the “Select” button, they are copied across to the “Selected terms” list. Below this list, two radio buttons are used to switch between temporal “and” (or intersection) and temporal “or” (union) of the query terms, when more than one term is selected. When the “Search” button is pressed, the query is performed, and the result is presented in the two-level tree list at the bottom of the window. The top-level nodes are the media resources (sorted lexicographically), and the leaf nodes are matching intervals within each media resource. No visualisation of the intervals is provided (other than the textual indications of start time, end time and length), since this is not common among the existing user interfaces based on this paradigm. The interface includes a simple video player in a separate window, with the same time slider and skip buttons as Savantoogle.

---



**Figure 10.15** Forms interface.

## Evaluation methods

Based on the purpose of this evaluation, as defined in the beginning of this section, this evaluation was divided into two parts: a pilot study and a main study – each using different evaluation methods.

**Pilot study:** Before the actual evaluation phase started, a pilot study was performed. This included presenting the three implemented interfaces to three fellow researchers and having them comment on any potential usability problems they could spot. This also included an evaluation of the main study design. In this way, the pilot study can very well be viewed as a part of the implementation phase rather than the evaluation phase.

As none of the interfaces had been used by anyone but the designers, the pilot study was a critical phase in assuring a consistent level of quality among the implementations. Several iterations of implementation and inspection were necessary before the implemented interfaces were ready for the main study. In particular, several problems regarding consistent wording of labels and buttons as well as confusing interface layout, were fixed.

**Main study:** As is already evident, the main study is a comparative evaluation of three different interfaces for information access in temporal annotation databases. For comparative evaluations, one has the choice of having each test subject evaluating a single interface (between-subjects) or have all testers evaluate all interfaces (within-subjects) (Mitchell and Jolley, 2001). For this evaluation, the latter alternative was chosen – primarily to reduce the number of test subjects needed and to assure that individual differences are cancelled out (e.g. some test subject being more positive in general than others).

Further, to remove ordering effects, a counterbalanced design was used by which the test subjects were randomly divided into three groups. All members of each group tested the interfaces in the same order, and this order was altered between groups such that all interfaces were equally often tested first, second or third. Each of the three test groups consisted of three test subjects – nine in total. This number was kept fairly low as the evaluation was primarily qualitative and thus required less testers and more time per tester. As the database used for all interfaces contained video and annotations from a computer science class, all test subjects were students that had recently taken this class. They were thus familiar with the subject at hand and would be legitimate end-users of the implemented interfaces.

The main study itself consisted of three steps. First, the test subject was given a brief introduction and tutorial for one of the interfaces. Second, the subject carried out several predetermined tasks using the interface while being observed. These two steps were then repeated for the two remaining interfaces. The training of the test subjects was done to equalise the playing field and to lessen the impact of individual differences with regards to previous experiences with similar interfaces. Finally, a questionnaire concerning all three interfaces was handed out. Thus, the main study made use of two evaluation methods: Observation and questionnaires.

---

---

The objective of the observation was to gather qualitative data about the interfaces. This was done by recording observational data on paper while the evaluation tasks were performed by the test subjects. These notes were often supplemented by a short informal interview afterwards to clarify some of the observations made.

The questionnaire was used to get quantitative data concerning the relative performance of the three interfaces. For this reason, all questions were repeated for the three interfaces and the test subjects were instructed to focus on comparative evaluation (i.e. which interface was best in a given category). All the questions were taken from the comprehensive Questionnaire for User Interaction Satisfaction (QUIS) (Chin et al., 1988).

### **Evaluation tasks**

One of the stated purposes of this evaluation was to study the performance of the interface with respect to different types of information access tasks. Three different categories of tasks were defined: Simple retrieval, complex retrieval and exploration. These are based on and correspond roughly to the task types defined in (Shneiderman, 1997): Specific fact finding, extended fact finding, and open-ended browsing & exploration of availability.

We defined *simple retrieval* as tasks where the user is looking for a simple answer – yes or no, or a single interval of video containing something of interest – and where there is no need to combine search terms or consider relations between things. Examples include finding the sole place where a certain term is used, or determining if a given term is mentioned in a particular set of videos. Tasks of this kind are conceptually simple, and a good query interface should be able to handle them simply and efficiently. If an interface is geared towards more complex queries, it may inhibit the formulation of simple ones – it may be too powerful for its own good. Therefore, we expect the simplest interfaces – Savantogle, and to a lesser extent Forms – to score well for simple retrieval.

*Complex retrieval* is here defined as tasks where the result is more intricate – a set of video intervals, for instance, or an aggregation of them (say, total length) – or where the user needs to combine search terms, or establish how different terms relate to each other. Determining how much time is spent on a given term, or finding intervals where two particular terms are active at the same time, are examples of complex re-

---

trieval. This calls for interfaces able to construct composite query expressions and to perform aggregate functions. We expect Savantoogle to perform worse than the other two interfaces at this, since it has little support for aggregation and temporal operations – it considers video documents the unit of retrieval. It is more difficult to predict how Forms and Savanta will perform.

*Exploration* is tasks where neither the goal nor the path towards it is entirely clear. It may be the user trying to find out what exists in the database, what trends can be established, what characterises a subset of the database. Examples include finding out what the most important term is in a given set of videos, or determining which narrower term of a particular term is most used. For this kind of fuzzy fact-finding, we expect the visualisation and multiple access methods of Savanta to give it the upper hand.

### **10.3.2 Results of usability evaluation**

This section presents the observations made during the evaluation as well as the questionnaire results. Finally, a discussion of the outcome of the evaluation and a look at reliability and validity concerns.

#### **Results from observation**

By observing the test subjects, listening to their comments and interviewing them afterwards, we discovered a few interesting tendencies and patterns. Many comments were related to details of each interface, such as button placement and double-click behaviour. This is not in itself very interesting outside the context of our implementations, but in many cases, more general guidelines and tendencies may be distilled from them. Other observations are inherently more general, and thus more directly useful. In the following discussion, we have tried to abstract the observations away from implementation details, and to focus on the more high-level lessons to be learned from them.

**Applying conventions makes sense** – the users were pleased with finding elements of software systems they had used before. The Home, Back and Forward buttons of Savanta, familiar from web browsers, were used extensively and without any problems. This was expected, but it is nice to see such expectations confirmed, especially since these controls in this case were used outside their normal environment, web browsers. Savantoogle was praised for its resemblance to Google, which everybody was familiar with and enjoyed using. Likewise, most of the test subjects

---



expected something useful to happen (video playback, mostly) when they double-clicked on search results and other things; it was a source of annoyance and confusion when it didn't.

**The model and/or domain have clear affordances** – given the organisation of the information in our database, our test subjects expected the user interfaces to be organised correspondingly. The temporal aspect and the list of videos were not directly accessible in Forms and Savantoogle, and this was frowned upon. For instance, to find a particular video in Forms, the users had to perform a query using a disjunction of all the top-level terms, and scan the result manually to find the desired video. This was often referred to as “cheating” and an indirect, unintuitive process, though it was not particularly difficult or labour-intensive to do. The lack of a visual, temporal overview of the media resources in Forms was also a common complaint.

**Feedback is important** – perhaps not a stunning discovery, but the users clearly had a strong need to be certain that their queries produced correct results; to know why the interfaces came up with the results they did. When using the search field in Savanta, some test subjects spent considerable time expanding the term tree to confirm that it actually was a matching term that had been found. When querying in Forms, a few test subjects wanted to see which terms overlapped the search result, for instance with an interval visualisation like in the other two interfaces. In Savantoogle, many repeated essentially the same query several times, varying the use of quote marks and pluses, to be sure that the desired result had been produced. This could perhaps be improved by presenting a better summary of the media resource for each match – our implementation just presented its title and comment, and the list of matching terms.

**Multiple access methods is good** – many felt restricted by Forms and Savantoogle, since those systems had only one method each for finding information. Several users commented that it was easy to “get stuck”, whereas in Savanta, the users had multiple angles from which to approach the problem. While this meant extra complexity – some commented that it was difficult to remember all the possibilities – it was clearly preferred, and “felt faster” (even when it was not). This confirmed our hopes that a richer, more powerful and flexible environment could still be at least as pleasant to use as a simple, minimalist one, even for simple tasks.

---

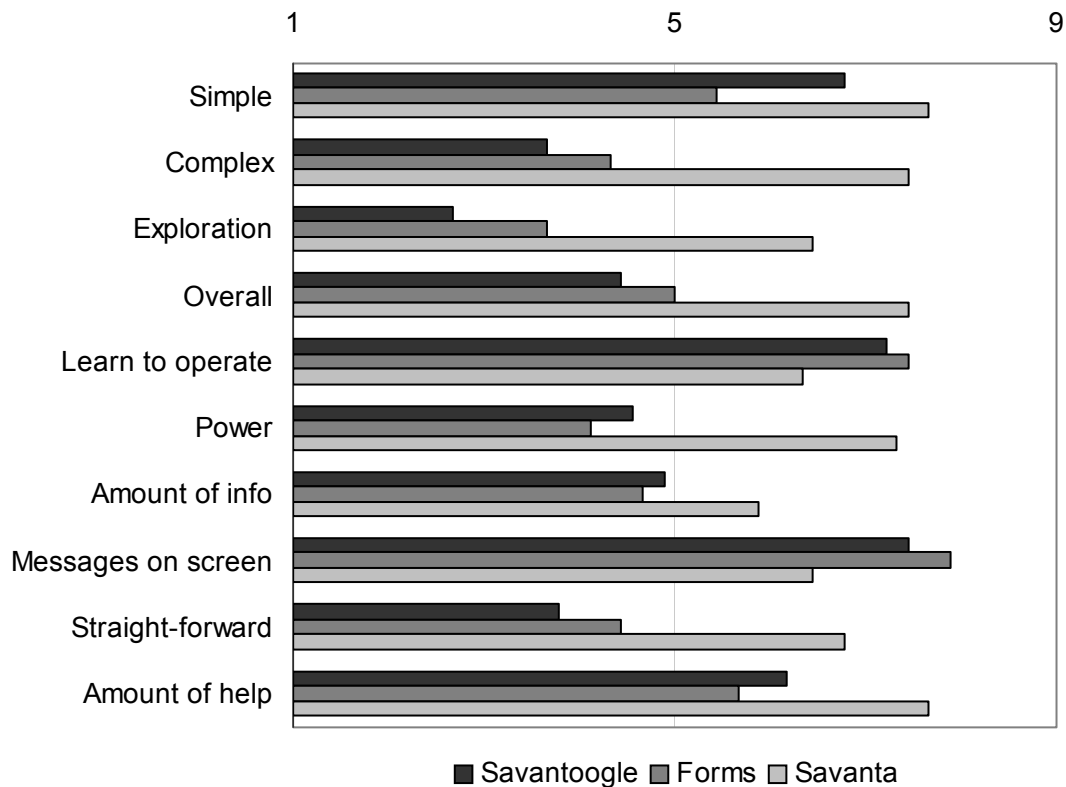
**Complex tasks and simple interfaces do not match** – when faced with a complex or exploratory task, many of the test subjects wanted to solve it in a correspondingly clever manner. They did not like to use a sequence of simple operations, especially when they had to assemble the result manually in their heads (or on paper). Even though it might be easy and obvious, it felt like a chore. In many cases it was, of course – finding the most used term in a subset of the database is very time-consuming when the interface only supports elementary functions like searching for a word. It seems that users expect and require the system to provide functions at the same level of complexity as the tasks the users want to perform, even though they have to spend time finding them and learning to use them.

### **Results from questionnaire**

The questionnaire consisted of 9 questions drawn from QUIS (Chin et al., 1988) and one custom question (number 6) for each of the interfaces, 30 in total. For all questions, a 9-point scale was used (1-9). The following questions were used:

1. Reactions to the system with respect to simple retrieval (1:terrible – 9:wonderful).
  2. Reactions to the system with respect to complex retrieval (1:terrible – 9:wonderful).
  3. Reactions to the system with respect to exploration (1:terrible – 9:wonderful).
  4. Overall reactions to the system (1:terrible – 9:wonderful).
  5. Learning to operate the system (1:difficult – 9:easy).
  6. The expressive power was (1:inadequate – 9:adequate).
  7. Amount of information that can be displayed on screen (1:inadequate – 9:adequate).
  8. Messages which appear on screen (1:confusing – 9:clear).
  9. Tasks can be performed in a straight-forward manner (1:never – 9:always).
  10. Amount of help given (1:inadequate – 9:adequate).
-

The results of the questionnaires are displayed in Figure 10.16. Detailed results are presented in Appendix A.3.



**Figure 10.16 Results from questionnaires.**

Interpreting these results, we consider the following four findings to be the most notable:

◆ **Ranking with respect to task types**

Overall the picture is reasonably clear. Savanta was best liked regardless of type of task. The other two interfaces were found to perform almost equally.

◆ **Savanta best for more complex tasks**

The largest differences were evident for complex retrieval and exploration tasks; while Savanta and Savantoogle performed almost equally well for simple retrieval. Not surprisingly, there is a clear correlation between the results from “Complex” and “Power” questions.

◆ **Learn to operate vs. Straight-forward**

On first glance, there seem to be an inconsistency between Savanta’s results for these two categories. How can it both be the most difficult interface to learn and the most straight-forward to

use? We believe the answer lies in the results from the “Power”-question. While a simple and less powerful interface might be easy to learn, it can require a huge number of actions in order to complete complex tasks. With Savanta, once the user had grasped the interface, complex tasks could often be completed in just a few steps.

◆ **High power wins over high complexity**

As is evident from the results of the “Learn to operate” and “Messages on screen”-questions, the test subject found Savanta more complex and difficult to use. The power of the interface and the straight-forward way even complex tasks could be performed, did however make Savanta the preferred interface regardless.

### **10.3.3 Discussion of evaluation results**

Integration of several access methods is the cornerstone of the Savanta design. The results of the usability evaluation strongly suggest that this was a good idea. Offering multiple ways to reach one's goal made it less likely for users to “get stuck” using Savanta compared with the other two interfaces. Further, the results indicate that graphical visualisation of temporal metadata greatly helps the users in quickly gaining an overview of complex data.

While these findings were expected, it was interesting to examine the usability consequences of having multiple methods for accessing data. Savanta's interface was found to be a bit more complex, but the users found the increase in expressive power more than made up for this.

Even as Savanta in general was the best liked interface among those tested, there were variations depending on information access task. For complex tasks as well as exploration, the multitude of possibilities of Savanta ensured that it was the clear winner. But for more simple tasks, more simple interfaces performed equally well.

#### **Reliability and validity**

The results of a usability study is not worth much if the evaluation design is flawed. In general, possible methodology pitfalls can be classified as either reliability or validity concerns (Nielsen, 1993). An outcome is *reliable* if a new evaluation gives the same result, while *validity* concerns whether the result actually reflects what one really wants to test. This has previously been discussed in Section 3.5.

---

---

The most obvious way to improve the reliability of our evaluation would have been to use more test subjects. People are different, and the impact of these differences is reduced when more testers are involved. Still, the use of a comparative evaluation design reduces the impact of individual differences.

As for validity, the type of test subjects used is a possible source for concern. All test subjects were quite experienced with computers and thus not the best to judge usability with respect to other types of users. However, with the test data we used, the test subject very much represented the intended target audience.

Another possible concern with respect to validity is the choice of the other two interfaces used for comparison. While their design and capabilities were tailored to match existing classes of interfaces for information access in temporal metadata databases, they were still designed and implemented by us. Thus (unintentional) bias on our part is a clear possibility. Still, we felt it was critical for comparison purposes that all interfaces used the same data (and data model) and making our own implementations was pretty much the only way to achieve this.

## **10.4 Discussion**

---

In this chapter, we have described the motivation, design and implementation of Savanta, a novel interface for accessing information in temporal annotation databases. Savanta integrates various user interface techniques and paradigms in order to create a rich and powerful environment for video search, retrieval, browsing, exploration and analysis, while not sacrificing ease of use.

Through a usability study, we have shown that Savanta is perceived as better than traditional methods for video database query, even though it is slightly more complex. The flexibility and multiple approach angles of the Savanta interface made it more straight-forward to use for complex and exploratory tasks than the simple interfaces.

The remainder of this chapter discusses how the four fundamental design ideas from Chapter 5 are reflected in Savanta and how well they were found to contribute to the overall performance of Savanta with respect to the overall research question.

---

## **Revised interaction model**

Savanta was an integration of filtering, searching and navigation where all three could be used iteratively. This means that if you get stuck using searching, you can navigate in the result you have come up with and perhaps find suitable suggestions for filters that might take you further.

The usability evaluation clearly indicated that the test subjects found the Savanta interaction model very powerful – at least when they had familiarised themselves enough with the interface to know which avenues were open to them. They commented that it was difficult to “get stuck” and that the interface “felt fast”.

Observation of the evaluation made it clear that even if users seldom got stuck, they often used a less-than-optimal route to the solution of the evaluation tasks. This indicates that the interface is somewhat complex and that further experience might make the use of it even more efficient.

## **Intra-result analysis**

A dynamic, temporal analysis was used for two purposes: To generate on-the-fly statistics and filter suggestions. As in SESAM, the filter suggestions served a dual purpose, not only for filtering but also as useful information in their own right. For example, it was easy at-a-glance to see which registered terms had the closest (temporal) relationship to the current result. For each of the filter suggestions, three operations were available. If, for example, “HTML” was suggested as a filter, the user could remove anything but HTML from the result, add (or remove) everything related to HTML to (or from) the result. Of these three operations, only those that would alter the result were available. Interestingly, this presented yet another way to get information about the result. (If “add HTML” was disabled, everything related to HTML must already be present.)

The sheer wealth of information available was often difficult for test subjects to get used to – at least in the beginning. Even expert users sometimes only realised in retrospect that a given task could have been solved much easier. This is however also an indication of the power of the technique as implemented in Savanta.

---

---

### **Active user interfaces**

As in SESAM, the filter suggestions represented the active user interface part of Savanta – taking advantage of recognition rather than recall. In difference, Savanta also allowed users to construct their own filters using the searching component.

This meant that the two information access methods complemented each other. Therefore the fact that the set of presented filter suggestions by necessity is much smaller than set of all possible filters, had less impact here than for SESAM. Another fact that worked to Savanta's advantage was that the complex inter-relationships between information objects in the Savanta data model, made it difficult both to fault the automatic selection process and to find better result modifications manually.

### **Dynamic user interfaces**

The dynamic component of Savanta was the hypertext panel shown to the right in Figure 10.9. It contained current statistics about the result and the selected term or media resource, as well as information about broader and narrower terms and the filter suggestions. This information was updated each time the user made a change to the result or selected a term.

Again, the experience with the dynamic component in Savanta was comparable to the experience with SESAM. Users at first found it a bit confusing that parts of the interface other than the part they had interacted with, changed. Later on, this became less of a problem and they more often than not realised how the information presented in the hypertext panel could be used to their advantage.

---





---

## Chapter 11

# Discussion

---

This chapter contains a discussion of the results of the work presented in this thesis. I start by briefly describing the research background and the overall research question. I then discuss how my four fundamental design ideas from Chapter 5 performed based on the results of the evaluations presented in earlier chapters. Finally, I present my own post-project reflections.

### 11.1 Research background

---

The work presented in this thesis is based on a realization of the increased importance of information repositories and how locating information has become an ever more vital task. With this in mind, I aimed at answering the following main research question:

*How can user interfaces for accessing information in large repositories be designed to provide assistance to users without impairing usability?*

To examine this research question, I decided to focus on the following five key challenges:

- ◆ Large information repositories
- ◆ New classes of users
- ◆ Complex data models
- ◆ Different information access strategies
- ◆ Multimedia data types

Based on the research question and the five key challenges, I have presented four fundamental design ideas as to how simple, usable interface for large information repositories could be made. To evaluate these design ideas and thus investigate the research question, a methodology of repeated design-implementation-evaluation phases was used. The results of these evaluations are discussed in the following section.

## **11.2 A discussion of the four design ideas**

---

The four fundamental design ideas as presented in Chapter 5 were as follows:

- ◆ Revised interaction model
- ◆ Intra-result analysis
- ◆ Active user interfaces
- ◆ Dynamic user interfaces

Below, I discuss how these four design ideas have performed with respect to the overall research question.

### **11.2.1 Revised interaction model**

The revised model contains two key components: It is iterative and it integrates several different methods for accessing information. By being iterative, the model supports modifying a result until it fits the user's need. The results of the evaluations show that this was found to work very well – especially for large results which otherwise would have been almost unmanageable.

The integration of several methods for accessing information led to very powerful user interfaces where the user's information need often could be very accurately described. To take full advantage of these features, it was evident that some degree of familiarization was necessary in order to know the different available avenues.

The downside of the revised interaction model was that it made the user interfaces more complex. In the end, however, the test subjects found that the increased power more than made up for the increased complexity, at least for all but the most simple tasks.

---

---

### **11.2.2 Intra-result analysis**

I have used dynamic intra-result analysis for two purposes: To derive on-the-fly high-level information and to suggest possible result manipulations (filters). Evaluations indicated that derived information is very helpful for gaining an overview of large results. It often made it possible for test subjects to get an understanding of the result without time-consuming manual scans of hundreds of objects. The suggested filters was also found to be of great use, even if every suggestion was not equally suitable. In particular, they were found to be helpful for removing objects of low relevance (“noise removal”).

Some test subjects found that the sheer wealth of information made available was difficult to make use of, at least initially. Even expert users sometimes realised after a task had been completed, that it could in fact have been solved much easier using a different approach.

As the derived information was dynamically generated, it also required large parts of the user interface to be updated each time the result was altered. Early on, this was often found unsettling and care was taken to minimize its consequences in later implementations.

### **11.2.3 Active user interfaces**

The idea behind active user interfaces was to rely on recognition rather than recall to as large extent as possible. In the developed user interfaces, this was in large part implemented by suggesting possible result modifications to users, rather than forcing them to take the initiative themselves.

The results of the usability evaluations suggest that this idea only works well in some settings. For example, the suggestions worked best for non-experts and for vague information needs. In these cases, the reduced expressive power did not matter much and the issue of recognition vs. recall became more important. Expert users, on the other hand, did not mind the effort necessary to use interfaces more based on recall as long as these gave them more expressive power.

The process of selecting result modification suggestions was found to work better in Savanta than SESAM. This can be explained by the complexity of the temporal relationships between intervals in Savanta. The complex inter-relationships between information objects in the Savanta data model, made it more difficult both to fault the automatic selection process and to find better result modifications manually.

---

## 11.2.4 Dynamic user interfaces

My aim was to make the user interfaces more context-sensitive – i.e. to make presented information and interaction controls always relevant with respect to the current situation. This was implemented by taking advantage of the intra-result analysis to dynamically update the display of derived information about the result as well as suitable filter suggestions.

How well this idea worked in practice, varied according to how appropriate the users found the presented information. When the information was not found useful, the increased user interface complexity was found confusing. But when the presented information was useful and as the test subjects got more experienced, this design idea tended to make both modification and navigation of results easier.

The best results was achieved for handling of large results and for exploration tasks. Large results could often easily be reduced to a manageable size (“noise removal”) as the suggested filters was focused on size reduction. For exploration, the dynamically updated user interface made it easier for users to get an overview of what they had got and to identify interesting ways to proceed.

## 11.2.5 Design ideas related to type of repository

The experiences with each of the four design ideas with respect to the implemented interfaces, can be condensed into Table 11.1.

	Revised interaction model	Intra-result analysis	Active interface	Dynamic interface
<b>Textual metadata database</b>	Very helpful. Iterative result manipulation makes large results more usable	Very helpful. Makes gaining an overview of a large collection much easier.	Helpful for non-experts, sometimes hindering for experts.	Can be confusing, but relative performance improves with result size.
<b>Image database</b>	Only partly implemented. Comparable experiences to textual metadata databases.	Very dependent on good feature extraction. Good potential.	Very helpful as image features are much easier to recognise than recall.	Very dependent on good feature extraction. Limited value.
<b>Temporal multimedia annotation database</b>	Very helpful. Less likely to get stuck. Felt faster to users.	Helpful. How derived information related to result not always obvious to users.	Very helpful as complex data model and temporal relationships make recall difficult.	Confusing at first, a powerful tool with some training.

**Table 11.1 The four design ideas in relation to repository type.**

---

## 11.3 Post-project reflections

---

This section looks at individual parts of my work and presents my reflections, focused on areas with potential for improvements.

### Research question

Already in the introduction I stated that the research question is too broad to be studied in full width and depth. I therefore focused on five key challenges and limited myself to the study of four design ideas, and how to apply these to different types of information repositories.

In hindsight, one could argue that I should have chosen a tighter and more focused research question. As the consequences could have been far-reaching, it is difficult to be certain of the result – even at this stage. But perhaps my initial slow progression is an indication that a more clearly defined problem statement might have been a good idea.

### Research methodology

For research methodology, I chose design-implementation-evaluation rather than theorem study or abstraction. This is a decision I stand by – user interface design is not an exact science as it is impossible fully to predict users' response. Therefore, developing actual implementations and evaluating them in close-to-real-life situations are, in my view, the most suited approach.

### Approach

Given the research question and the research methodology, I came up with four design ideas that I wanted to study in different settings. In essence, this thesis is a presentation of the designs, implementations and evaluations made towards this goal.

Several aspects of this approach can be discussed:

- ◆ **Using a number of design ideas as a basis**

This was something that was done in order allow more focused designs and implementations than the research question afforded in itself. Therefore much of the same arguments used above also applies here.

- ◆ **The four design ideas themselves**

Given that using a number of fundamental design ideas as a basis is a useful approach, one could argue that I should have chosen them differently. But as user response is very hard to predict, it is

---

difficult to be certain of the outcome without the benefit of hindsight. I also believe that the evaluation results show that all the design ideas had their merits.

◆ **Using three different types of repositories rather than just one**

By looking both at a textual metadata database, an image database and a temporal metadata database, the available time had to be split between several projects. This made each study less in-depth than they would have been if I had just focused on one. But then again, I would have lost the ability to examine how the design ideas performed in different settings. This has given me a wider understanding of the problem domain than I otherwise might have got.

◆ **Distribution of time between the different implementations**

As should be evident from their relative length in this thesis, I spent most of my time on the textual metadata database implementation (SESAM) with less time on image databases and temporal metadata databases. Could the time spent have been split in a different way that might have given better results? That might be, but these decisions were, at least in part, driven by external events. The textual metadata database became available early on in my work and it was natural then to take advantage of this large real-world database right away. Also, the OntoLog database used in the Savanta-project did not become available until the final year. And even if I spent less time on the image database project compared with textual databases, the feature extraction implementations includes the results of several master degree student theses.

---

---

## Chapter 12

# Contributions and future work

---

This chapter presents main contributions (Section 12.1) and possible avenues for further work (Section 12.2).

### **12.1 Contributions**

---

The main contributions of this thesis are:

#### **Demonstrated the power of an iterative interaction model**

Rather than an interaction model where a query result is final, all designs and implementations presented in this thesis used an iterative interaction model which integrates several methods for accessing information. With this model users have more than one viable strategy for locating information and can modify a result until it fulfils their information need. Evaluations clearly indicate that this model performed superior to similar user interfaces based on the standard interaction model – especially with regards to large query results. This was true for all but the most simple tasks where the increased expressive power was not needed and thus could not offset the slightly increased user interface complexity.

#### **Showed feasibility of intra-result analysis**

One of the key ideas presented, is that available processing power on the client can be used to analyse the properties of a query result. This idea has been incorporated into the designs and implementations presented in

this thesis. In these implementations intra-result analysis have been used to derive high-level information which have been demonstrated to greatly assist users in gaining an overview of large results. The analysis have also been used to suggest possible result modifications. From the evaluations it is evident that this was a very helpful feature – especially to remove objects of lower relevance and thus make large results easier to handle.

### **Presented novel interfaces for accessing information**

I have presented SESAM, an application demonstrating a novel approach for assisting users in handling large results. Using a dynamic and active user interface applied to a textual metadata database, users can specify what parts of a result they find relevant. This is done by selecting one or more result modifications from a set of suggested alternatives.

In the Savanta prototype, the ideas and lessons from SESAM were applied to a different domain – a temporal metadata database. This powerful interface was based on an integration of inter-video browsing, filtering, searching, navigation, temporal analysis and visualisation. While slightly more complex than traditional user interfaces for accessing video databases, the increased flexibility and multiple-angled approach presented a straight-forward user interface for performing complex information access tasks.

### **Evaluated viability of design ideas for usable interfaces**

To evaluate the implemented prototypes and the ideas on which they were based, three different usability evaluations were performed. Using a combination of qualitative and quantitative evaluation methods, these evaluations not only demonstrated the viability of the designs but also the value of involving users in the development process. The results of the evaluations show that an interactive interaction model, several methods for accessing information and dynamic data analysis, can be combined in a user interface with high usability and expressive power without corresponding decreases in usability.

## **12.2 Future work**

---

This is a list of some of my ideas for further work:

- ◆ **Design ideas applied to other databases**

In order to validate the generality of the design ideas, it is important to implement and evaluate them in as many settings as possible. With more time available, it would have been helpful to apply

---



---

the ideas to other databases and then evaluate if their performance changed.

◆ **Design ideas applied to semi-structured data**

The text-based work presented in this thesis is limited to a highly structured, textual metadata database using a relational DBMS. This setting has fundamental differences compared to semi-structured data (XML etc.). Combined with the fact that much research effort currently is spent in this domain, this would have made it an interesting topic for study.

◆ **Extended work on image databases**

The work presented in this thesis with regards to image databases is little more than a proof-of-concept demonstration. With more time available, a more thorough design, implementation and evaluation would have been interesting. The semantic gap between how computers and humans reason about images makes interfaces for information access an important and challenging subject.

◆ **More thorough testing of implemented prototypes**

The cycle of design-implementation-evaluation was completed twice for the SESAM prototype. The changes made to the second implementation on the basis of the first prototype, led to very improved results with the second prototype. This clearly indicates the value of evaluations – not only in their own right but as input for further development. I would therefore have liked to be able to complete more cycles on Savanta to refine my ideas and learn more about the problem at hand.

◆ **Publish prototypes and collect data from real users**

As of yet, the prototypes have only been evaluated in a controlled environment. This makes sense for gaining qualitative data in order to understand the workings of the human-computer interaction. With the increased maturity of the prototypes, it would have been possible to publish them on the Web and collect real users' opinions by the means of questionnaires (or similar).

◆ **A closer look at utility functions**

The implemented utility functions, both in SESAM and Savanta, have not been separately evaluated – only as part of the interface. An interesting project would have been to separate them from the interface and evaluate them separately. This would have it possible to implement several different alternatives and perform a

---

comparative evaluation. But as their output must be seen in correspondence with the users' information need, such an evaluation would have required human test subjects and thus much time and resources.

---

---

## Appendix A

# Detailed evaluation results

---

This appendix presents detailed results from the evaluations described in Chapters 7, 8 and 10. No formal evaluations was made of the prototype presented in Chapter 9 as preliminary, informal evaluations indicated that the implementation was not at a level where this was deemed useful.

### **A.1 Evaluation of first SESAM prototype**

---

This section presents the evaluation tasks and the results from the questionnaire and the performance measurements.

#### **Evaluation tasks**

Three sets of evaluation tasks were made. Each test subject performed one set of tasks using one interface and all test subjects performed the three sets in the same order. The only thing that varied was the order of the interfaces. This was determined randomly.

Each set of evaluation tasks was divided into three parts – each with three tasks (for a total of nine per set and 27 in total). The parts were: (A) Exact match – few results, (B) exact match – many results and (C) browsing. The topic for each task is listed in Norwegian.

#### **Set 1:**

- ◆ A1: Find all “skohorn”
- ◆ A2: Find all “busser”
- ◆ A3: Find all “ølboller” from “Arendal”

- ◆ B1: Find all “lokk”
- ◆ B2: Find all “dukker”
- ◆ B3: Find all “mangletrær”
- ◆ C1: Find some interesting “sykler”
- ◆ C2: Find some interesting “tables”
- ◆ C3: Find something you find interesting

**Set 2:**

- ◆ A1: Find all “piano”
- ◆ A2: Find all “kikkerter”
- ◆ A3: Find all “dukker” from “Troms”
- ◆ B1: Find all “tepper”
- ◆ B2: Find all “kister”
- ◆ B3: Find all “stoler”
- ◆ C1: Find some interesting “fløyter”
- ◆ C2: Find some interesting “senger”
- ◆ C3: Find something you find interesting

**Set 3:**

- ◆ A1: Find all “kabinett”
  - ◆ A2: Find all “radioer”
  - ◆ A3: Find all “luer” made from “lin”
  - ◆ B1: Find all “drakter”
  - ◆ B2: Find all “kjoler”
  - ◆ B3: Find all “bukser”
  - ◆ C1: Find some interesting “sverd”
  - ◆ C2: Find some interesting “ølboller”
-

- ◆ C3: Find something you find interesting

### Usability evaluation – Questionnaire

The ten test subjects were asked 15 questions about each tested interface. They could answer using a 9-point scale (1-9). Results are shown in Table A.1, A.2 and A.3. Empty cells indicate that the user did not wish to answer the question (NA).

	User										Average
	1	2	3	4	5	6	7	8	9	10	
Small results	7	6	6	6	7	3	8	8	7	7	6,5
Large results	4	4	2	4	3	3	5	5	7	4	4,1
Browsing	7	5	8	3	2	4	6	6	8	7	5,6
Overall	6	5	3	5	3	4	6	6	7	5	5,0
Layout	7	5	5	9	3	5	7	6	9	8	6,4
Amount of info.	9	3	8	7	6	7	3	6	6	8	6,3
Arrangement	7	5	6	9	9	8	6	7	8	8	7,3
Messages	8	6	9	9	7	7	7	7	7	5	7,2
Keep informed	8	3	9	9	9	7	7	7	9	5	7,3
Learning	7	7	8	9	9	7	8	6	9	8	7,8
Getting started	8	7	8	9	9	7	7	7	9	8	7,9
Time to learn	7	7	7	9	9	7	7	7	9	6	7,5
Trial and error	7	5	3	9	3	3	6	6	8	8	5,8

**Table A.1 Questionnaire results – Forms**

	User										Average
	1	2	3	4	5	6	7	8	9	10	
Small results	7	6	8	7	7	8	8	8	8	7	7,4
Large results	8	8	8	5	7	8	8	7	9	7	7,5
Browsing	3	8	8	4	5	7	7	8	9	3	6,2
Overall	7	8	9	7	7	8	8	8	9	7	7,8
Layout	8	7	7	9	8	5	8	7	8	7	7,4
Amount of info.	8	7	5	8	3	8	8	7	8	9	7,1
Arrangement	6	7	8	8	3	4	8	7	8	8	6,7
Messages	7	6	7	8	7	5	7	8	9	6	7,0
Keep informed	8	5	9	8	9	8	7	7	9	3	7,3
Learning	7	7	6	9	9	8	7		9	8	7,8
Getting started	8	7	8	9	9	8	7	7	9	8	8,0
Time to learn	6	7		9	9	8	7	6	8	8	7,6
Trial and error	8	7	9	9	7	8	7	6	9	8	7,8
Exploration	9	9	9	9	9	9	7	6	8	8	8,3
Speed	4	7	9	5	9	8	8	8	6	9	7,3

**Table A.2 Questionnaire results - Dynamic query**

	User										Average
	1	2	3	4	5	6	7	8	9	10	
Small results	7	6	7	7	8	7	7	7	6	7	6,9
Large results	7	2	9	5	7	6	4	7	6	5	5,8
Browsing	7	3	5	6	7	6	5	7	8	3	5,7
Overall	7	3	8	8	8	7	5	7	6	6	6,5
Layout	7	6	8	5	5	6	5	6	6	5	5,9
Amount of info.	8	5	6	4	4	4	4	6	7	7	5,5
Arrangement	7	5	8	7	2	6	3	7	8	7	6,0
Messages	6	5	9	5	3	6	5	6	7	7	5,9
Keep informed	8	6	9	7	9	7	6	7	9	3	7,1
Learning	7	7	6	6	9	5	8	7	7	8	7,0
Getting started	8	7	7	9	9	5	7	7	8	8	7,5
Time to learn	7	3	7	5	9	7	7	7	8	8	6,8
Trial and error	7	4	9	4	9	8	5	6	8	8	6,8
Exploration	9	9	9	5	9	8	6	6	9	8	7,8
Speed	4	5	9	9	9	6	7	7	6	9	7,1

**Table A.3 Questionnaire results - SESAM**

## Performance evaluation

During the evaluation, task completion times was recorded for three tasks with small results (A1-A3) and three tasks with large results (B1-B3). The results (in seconds) are displayed in tables B.1, B.2 and B.3. The order in which the test subjects used the interfaces is as follows:

- ◆ Forms - Dynamic query - SESAM – users 1 and 3.
- ◆ Dynamic query - SESAM - Forms – users 2 and 8.

- ◆ SESAM - Forms - Dynamic query – users 4 and 7.
- ◆ Forms - SESAM - Dynamic query – users 5 and 9.
- ◆ Dynamic query - Forms - SESAM – user 6.
- ◆ SESAM - Dynamic query - Forms – user 10.

	User										Average
	1	2	3	4	5	6	7	8	9	10	
Task A1	6	5	5	8	8	6	57	3	29	2	13
Task A2	10	6	4	6	2	5	25	2	6	7	7
Task A3	12	15	16	14	8	16	13	11	70	15	19
Task B1	35	78	104	31	56	127	69	31	49	15	60
Task B2	154	118	137	13	86	35	13	53	208	58	88
Task B3	50	123	30	15	29	53	125	238	43	5	71

**Table A.4 Performance measurement results – Forms (in seconds)**

	User										Average
	1	2	3	4	5	6	7	8	9	10	
Task A1	5	15	15	8	6	5	13	3	7	5	8
Task A2	7	5	7	12	4	3	6	3	4	8	6
Task A3	15	26	14	41	42	104	23	27	35	10	34
Task B1	84	159	16	18	37	104	47	233	39	82	82
Task B2	63	43	9	21	10	25	47	37	12	32	30
Task B3	64	55	22	14	51	76	14	40	18	11	37

**Table A.5 Performance measurement results – Dynamic query (in seconds)**

	User										Average
	1	2	3	4	5	6	7	8	9	10	
Task A1	46	5	3	14	3	5	9	3	9	5	10
Task A2	29	4	5	4	3	3	4	3	7	2	6
Task A3	25	45		129	30	101	114	51	120	136	83
Task B1	60	71		59	47	69	96	70	56		66
Task B2	51	36		48	29	86	80	23	22		47
Task B3	64	51		33	53	39	78	43	30		49

**Table A.6 Performance measurement results – SESAM (in seconds)**

## A.2 Evaluation of second SESAM prototype

This section presents the evaluation tasks and the results from the questionnaires for the evaluation of the second SESAM prototype

## Evaluation tasks

Three sets of evaluation tasks were made. Each test subject performed one set of tasks using one interface and all test subjects performed the three sets in the same order. The only thing that varied was the order of the interfaces. This was determined randomly.

Each set of evaluation tasks was divided into four parts. The parts were: (A) Exact match, (B) noise removal, (C) exploration and (D) navigation. The topic for each task is listed in Norwegian.

### Set 1:

- ◆ A1: Find all “skohorn”.
- ◆ A2: Find all “ølbolle” from “Arendal”.
- ◆ B1: Find all “tang”. Remove anything that is not “tenger (redskap)”.
- ◆ C1: Use the interface to gain an overview of what the Norwegian Folk Museum has from your home county.
- ◆ D1: Find all “lokk”. Find one that looks interesting and locate similar objects in the result.

### Set 2:

- ◆ A1: Find all “piano”.
- ◆ A2: Find all “bukse” from “Telemark”.
- ◆ B1: Find all “duk”. Remove anything that is not “duker”.
- ◆ C1: Use the interface to gain an overview of what the Norwegian Folk Museum has from a random county.
- ◆ D1: Find all “kiste”. Find one that looks interesting and locate similar objects in the result.

### Set 3:

- ◆ A1: Find all “kabinett”.
  - ◆ A2: Find all “lue” from “Oppland”.
  - ◆ B1: Find all “penn”. Remove anything that is not “penner”.
-



- ◆ C1: Use the interface to gain an overview of what the Norwegian Folk Museum has from a random county.
- ◆ D1: Find all “teppe”. Find one that looks interesting and locate similar objects in the result.

## Usability evaluation – Questionnaire

The 18 test subjects were asked 16 questions about each tested interface. They could answer using a 9-point scale (1-9). Results are shown in Table A.7 – A.9. Empty cells indicate that the user did not wish to answer the question (NA).

Task	Computer professionals									Museum conservators									Averages		
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	Cp	Mc	Total
Exact match	7	6	9	7	7	8	7	3	4	8	9	7	5	9	5	8	8	8	6,4	7,4	6,9
Noise removal	5	7	4	3	5	7	4	7	7	3	7	4	5	5	5	3	6	6	5,4	4,9	5,2
Exploration	6	3	1	1	6	6	4	1	2	3	6	5	5	7	5	4	4	6	3,3	5,0	4,2
Navigation	3	8	5	4	7	5	4	9	8	6	6	7	5	6	6	4	5	5	5,9	5,6	5,7
Totalt	5	4	5	3	3	5	8	4	5	5	5	6	5	5	5	5	6	5	4,7	5,2	4,9
Layout	8	4	8	5	6	7	8	9	7	7	5	6	6	6	7	5	8	5	6,9	6,1	6,5
Amount of info.	5	5	6		2	7	7	1	7	7	3	6	5	5	5	5	8	5	5,0	5,4	5,2
Arrangement	8	7	6	5	5	7	8	9	4	8	3	6	6	6	7	5	7	5	6,6	5,9	6,2
Messages	7	4	9	6	5	8	4	3	9	7	4	6	5	6	7	6	8	6	6,1	6,1	6,1
Keep informed	7	9	7	5	9	7	9	9	8	3	5	6	3	7	6	6	8	4	7,8	5,3	6,6
Learning	8	7	9	7	6	9	8	9	5	7	7	7	5	7	7	7	7	6	7,6	6,7	7,1
Getting started	9	7	8	7	5	8	8	4	5	7	7	7	5	6	7	8	7	7	6,8	6,8	6,8
Time to learn	9	9	3	7	6	8	8	8	3	5	7	5	6	6	7	7	7	7	6,8	6,3	6,6
Trial and error	7	7	3	3	4	8	4	3	4	5	6	6	4	6	7	7	7	6	4,8	6,0	5,4
Exploration	6	8	9	8	9	9	8	9	5	5	6	6	5	7	7	7	8	5	7,9	6,2	7,1
Speed	8	7	9	5	7	7	8	9	8	7	5	7	4	6	7	7	5	4	7,6	5,8	6,7

**Table A.7 Questionnaire results – Forms**

Task	Computer professionals									Museum conservators									Averages		
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	Cp	Mc	Total
Exact match	6	7	9	6	6	7	7	9	9	8	9	7	5	9	7	8	7	6	7,3	7,3	7,3
Noise removal	3	2		6	3	7	5	6	8	8	9	5	5	7	7	8	2	5	5,0	6,2	5,6
Exploration	5	4		4	6	7	6	9	8	8	9	5	5	7	8	5	7	6	6,1	6,7	6,4
Navigation	5	4		7	7	8	5	5	9	8	9	7	5	8	7	8	7	6	6,3	7,2	6,7
Totalt	5	5	7	5	5	6	5	8	8	8	9	6	5	8	7	6	6	6	6,0	6,8	6,4
Layout	5	4	8	6	6	7	8	9	7	8	9	6	5	9	7	6	7	7	6,7	7,1	6,9
Amount of info.	3	5	7		3	8	8	9	7	8	9	6	5	9	7	6	7	8	6,3	7,2	6,7
Arrangement	4	7	8		8	7	8	9	7	8	9	6	5	8	6	6	6	7	7,3	6,8	7,0
Messages	7	8	9	7	8	8	8	9	9	8	9	6	4	8	7	6	7	7	8,1	6,9	7,5
Keep informed	7	8	9	6	9	8	9	9	9	5	9	6	3	9	6	6	6	6	8,2	6,2	7,2
Learning	7	8	7	6	6	7	7	9	8	7	9	6	5	9	7	8	9	7	7,2	7,4	7,3
Getting started	9	8	4	7	3	7	5	7	7	8	9	6	5	7	7	8	8	7	6,3	7,2	6,8
Time to learn	9	9	6	7	5	7	6	3	7		9	6	6	7	7	7	8	6	6,6	7,0	6,8
Trial and error	7	7	8	7	5	8	5	9	8	9	8	6	4	8	7		7	6	7,1	6,9	7,0
Exploration	6	9	9	7	9	8	8	9	5	9	9	6	5	8	7	8	8	6	7,8	7,3	7,6
Speed	8	6	3	5	7	6	5	9	8	8	9	7	4	9	7	8	8	8	6,3	7,6	6,9

**Table A.8 Questionnaire results – Dynamic query**

Task	Computer professionals									Museum conservators									Averages		
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	Cp	Mc	Total
Exact match	5	8	9	6	8	7	7	9	8	8	9	7	5	9	7	8	7	6	7.4	7.3	7.4
Noise removal	7	6	9	7	9	5	8	7	9	8	9	4	5	8	7		8	8	7.4	7.1	7.3
Exploration	8	7	9	8	8	7	6	6	7	8	9	5	5	7	7	5	5	6	7.3	6.3	6.8
Navigation	7	8	8	7	9	7	8	5	8	8	9	8	5	7	7	8	8	4	7.4	7.1	7.3
Total	7	7	8	7	8	6	7	9	7	7	9	7	5	7	6	7	7	7	7.3	6.9	7.1
Layout	6	9	6	7	8	9	8	9	6	6	9	6	5	7	6	7	7	7	7.6	6.7	7.1
Amount of info.	7	7	8	6	9	9	8	7	7	7	9	6	5	7	6	7	7	8	7.6	6.9	7.2
Arrangement	4	8	6	8	9	8	7	9	4	5	9	6	4	7	6	6	7	7	7.0	6.3	6.7
Messages	7	8	9	8	7	8	8	9	9	3	9	6	3	7	7	7	7	6	8.1	6.1	7.1
Keep informed	7	6	8	5	9	7	9	9	9	4	9	6	4	7	6	5	7	8	7.7	6.2	6.9
Learning	7	7	6	8	8	7	8	9	6	5	9	7	5	7	7	7	8	8	7.3	7.0	7.2
Getting started	9	7	5	8	8	5	8	7	4	7	9	6	5	7	7	8	8	7	6.8	7.1	6.9
Time to learn	9	7	6	8	8	8	8	4	4		9	6	6	7	7	8	8	7	6.9	7.3	7.1
Trial and error	8	7	9	9	8	9	7	9	6	8	9	4	4	8	7	7	8	6	8.0	6.8	7.4
Exploration	8	7	9	8	9	9	8	9	5	8	9	4	5	8	7	8	8	7	8.0	7.1	7.6
Speed	8	6	9	5	7	6	7	7	8	8	9	7	4	8	7	8	6	8	7.0	7.2	7.1

**Table A.9 Questionnaire results – SESAM**

## A.3 Evaluation of Savanta

This section presents the evaluation tasks and the results from the questionnaires.

### Evaluation tasks

Three sets of evaluation tasks were made. Each test subject performed one set of tasks using one interface and all test subjects performed the three sets in the same order. The only thing that varied was the order of the interfaces. This was determined randomly.

Each set of evaluation tasks was divided into three parts. The parts were: (A) Simple retrieval, (B) complex retrieval and (C) exploration. Each task is listed in Norwegian.

#### Set 1:

- ◆ A1: Omtales HTML i uke 41?
- ◆ A2: Hva sier foreleser om egenrelasjoner?
- ◆ B1: Finn eksempler som handler om løkker.
- ◆ B2: Hvor stor andel av forelesningene brukes på praktiske opplysninger?
- ◆ B3: Hva skjer i det første kvarteret av dobbeltime 1 i uke 36?
- ◆ C1: Hva er det viktigste hovedtemaet i første tredjedel av semesteret?

- ◆ C2: Hvilket databasetema brukes det mest tid på?

**Set 2:**

- ◆ A1: Omtales JSP i uke 43?
- ◆ A2: Hva sier foreleser om lokale variabler?
- ◆ B1: Finn eksempler som handler om objekter.
- ◆ B2: Hvor stor andel av forelesningene brukes på databaser?
- ◆ B3: Hva skjer i det første kvarteret av dobbelttime 1 i uke 40?
- ◆ C1: Hva er det viktigste hovedtemaet i midterste tredjedel av semesteret?
- ◆ C2: Hvilket HTML-tema brukes det mest tid på?

**Set 3:**

- ◆ A1: Omtales databaser i uke 40?
- ◆ A2: Hva sier foreleser om evige løkker?
- ◆ B1: Finn eksempler som handler om merkelapper.
- ◆ B2: Hvor stor andel av forelesningene brukes på HTML?
- ◆ B3: Hva skjer i det første kvarteret av dobbelttime 1 i uke 43?
- ◆ C1: Hva er det viktigste hovedtemaet i siste tredjedel av semesteret?
- ◆ C2: Hvilket JSP-tema brukes det mest tid på?

**Usability evaluation – Questionnaire**

The nine test subjects were asked ten questions about each tested interface. They could answer using a nine-point scale (1-9). Results are shown in Table A.10 – A-12. Empty cells indicate that the user did not wish to answer the question (NA).

---

	User									Average
	1	2	3	4	5	6	7	8	9	
Simple	8	7	5	9	4	9	7	7	5	6,8
Complex	6	4	2	1	5	3	4	4	4	3,7
Exploration	5	2	2	1	3	3	3	2	3	2,7
Overall	7	6	4	3	4	3	5	3	5	4,4
Learn to operate	8	8	9	7	5	4	7	8	9	7,2
Power	6	6	2	4	5	4	8	3	3	4,6
Amount of info	6	8	7	4	4	4	6	2	3	4,9
Messages on screen	8	7	9	9	7	5	6	8	8	7,4
Straight-forward	5	5	3	3	4	3	5	3	3	3,8
Amount of help	4	5	9	7		3	9			6,2

**Table A.10 Questionnaire results - Savantoogle**

	User									Average
	1	2	3	4	5	6	7	8	9	
Simple	6	7	7	7	4	6	4	5	3	5,4
Complex	5	6	6	3	5	4	2	5	3	4,3
Exploration	4	3	5	4	4	5	1	5	2	3,7
Overall	6	6	6	5	4	6	4	5	3	5,0
Learn to operate	9	7	7	9	6	8	5	8	8	7,4
Power	5	5	5	2	4	6	2	6	2	4,1
Amount of info	3	7	7	4	3	6	3	7	2	4,7
Messages on screen	8	8	8	9	7	7	9	8	7	7,9
Straight-forward	5	5	4	6	4	5	3	5	3	4,4
Amount of help	3	5	9	8		5	4			5,7

**Table A.11 Questionnaire results - Forms**

	User									Average
	1	2	3	4	5	6	7	8	9	
Simple	9	7	8	8	7	8	6	8	8	7,7
Complex	8	7	7	8	7	7	7	8	8	7,4
Exploration	8	5	7	7	7	6	6	7	5	6,4
Overall	8	7	8	7	7	8	8	7	7	7,4
Learn to operate	7	7	4	5	6	7	8	6	7	6,3
Power	8	7	8	7	7	8	6	8	7	7,3
Amount of info	8	6	3	4	8	7	7	4	6	5,9
Messages on screen	8	6	5	7	7	6	7	7	5	6,4
Straight-forward	8	6	6	5	7	8	6	8	7	6,8
Amount of help	8	7	9	8		8	6			7,7

**Table A.12 Questionnaire results - Savanta**

---

## Appendix B

# About Savanta

---

The research behind Savanta – its conception, design, implementation and evaluation – is a joint effort between Jon Olav Hauglid and Jon Heggland. The purpose was to bring together our two different but related fields of research – Heggland's modelling and visualisation of temporal metadata and Hauglid's iterative, analysis-supported database search – and see what synergy effects could be achieved in the interface between them.

The entire process leading up to the results presented in Chapter 10 was highly iterative. While we assigned one to be overall responsible for each of the identified parts of the project, the other provided frequent feedback. Thus, by the nature of our close cooperation, it is impossible to completely separate one's contribution from the other. But Table A.1 indicates roughly who was the primary contributor to the design (D) and implementation (I) of the various ideas, tools and research described in

the Savanta chapter.

	Hauglid	Heggland
Stored metadata / conceptual model		DI
Derived metadata / analysis	DI	
Visualisation		DI
Navigation / web metaphor	DI	DI
Interaction model / iterative search refinement	DI	
Filtering	DI	
Evaluation	D	I
Savantogle		DI
Forms	I	D

**Table A.1 Primary contributors to various parts of Savanta**

The writing was also done jointly; hence, a substantial part of the material on Savanta is identical in our two theses. The table above is fairly indicative on who did the writing of the various sections as well.

We both agree that we were equal partners and contributors in all major phases of the Savanta project.

Trondheim, 7. June 2004.

Jon Olav Hauglid

Jon Heggland

---

---

## Appendix C

# Evaluation handouts

---

This appendix contains the evaluation descriptions handed out to the test subjects in the three usability evaluations described in this thesis. These handouts are all written in Norwegian.

### **C.1 Evaluation of first SESAM prototype**

---

#### **Evaluering av grensesnitt for informasjonsgjenfinning**

---

##### ***Introduksjon***

Formålet med denne brukerevalueringen er å sammenligne brukbarheten til tre forskjellige typer grensesnitt for informasjonsgjenfinning i databaser. Som testdeltaker skal du utføre en mengde oppgaver i hvert grensesnitt samtidig som det blir registrert hvor lang tid det tar å utføre oppgaven. I tillegg får du en del evalueringsspørsmål om hvert grensesnitt.

##### **Om brukergrensesnittene**

Følgende grensesnitt er omfattet av evalueringen:

- <sup>2</sup> Forms
- <sup>2</sup> Dynamic Query
- <sup>2</sup> SESAM

Siden alle grensesnittene er laget for bruk på web, gis det med vilje ikke ut mer informasjon på forhånd. En ting å merke seg er dog at alle søkefelt bruker case-insensitive substring match og at ordstemming og wildcards ikke er støttet.

## Om oppgavene

Det kan være stor forskjell på hvordan man ønsker å lete etter informasjon. Noen ganger vet man nøyaktig hva man er ute etter, mens andre ganger er letingen mer tilfeldig. I tillegg kan målet endre seg underveis hvis det viser seg at man ved første forsøk fikk et resultat som var uhandterlig stort.

For å se hvordan grensesnittene oppfører seg under varierende forhold, er oppgavene delt opp i tre:

a) **Eksakte oppgaver som gir få resultater.**

Med ”få resultater” menes så få at man typisk orker å se igjennom alle.

b) **Eksakte oppgaver som gir mange resultater.**

Dersom man får svært mange resultater fra et søk, ønsker man som regel å gjøre noe for å redusere størrelsen til et akseptabelt nivå. Hvordan dette kan gjøres, er avhengig av hvordan grensesnittet er utformet – noe som er en av de faktorene som ønskes evaluert.

I denne oppgavetyper blir du først spurt om hvor mange resultater du typisk orker å se igjennom manuelt. Hver av oppgavene gir et stort resultat, og du skal bruke grensesnittet til å redusere dette til den akseptable størrelsen du oppgav.

c) **Vage oppgaver (”browsing”).**

Naturlig nok er det vanskelig å spesifisere en vag oppgave. Derfor vil oppgavene bare gi et utgangspunkt og du blir bedt om å bruke dette til å utforske databasen.

Tre oppgavetyper og tre grensesnitt gir ni muligheter. For hver av disse får du tre oppgaver – tilsammen 27. Rekkefølgen grensesnittene evalueres i er tilfeldig, mens rekkefølgen på oppgavetyperne alltid er a), b), c).

## Om evalueringen

Evalueringen består av to deler:

1. **Måling av tidsbruk.**

Måling har bare hensikt i oppgavetyperne a) og b) ettersom vage oppgaver ikke har et klart definert mål. Husk her at det er grensesnittene som evalueres – ikke du.

2. **Evalueringsspørsmål.**

Spørsmålene er delt inn i to deler. En del besvares etter hver oppgavetype for hvert grensesnitt, mens en del besvares for hvert grensesnitt. Alle spørsmålene er tatt fra ”The Questionnaire for User Interaction Satisfaction” laget av University of Maryland og er av typen:

---



Overall reactions to the  
system:

terrible

wonderful

1 2 3 4 5 6 7 8 9

---

## C.2 Evaluation of second SESAM prototype

---

### Evaluering av program for informasjonsgjenfinning

---

#### *Introduksjon*

Formålet med denne brukerevalueringen er å sammenligne brukbarheten til tre forskjellige program for informasjonsgjenfinning i databaser. Som testdeltaker skal du utføre en del oppgaver i hvert program slik at du kan danne deg en oppfatning av hva du liker og ikke liker. Etterpå får du en del evalueringsspørsmål om hvert av programmene.

#### **Om programmene**

Følgende program er omfattet av evalueringen:

- <sup>2</sup> Forms
- <sup>2</sup> Dynamic Query
- <sup>2</sup> SESAM

#### **Om oppgavene**

Det kan være stor forskjell på hvordan man ønsker å lete etter informasjon. Noen ganger vet man nøyaktig hva man er ute etter, mens andre ganger er letingen mer tilfeldig. I tillegg kan målet med letingen endre seg underveis. For å se hvordan programmene egner seg til forskjellige måter å søke på, er oppgavene delt opp i fire:

a) **Eksakte oppgaver.**

Du får oppgitt nøyaktig hva du skal søke etter.

b) **Støyfjerning.**

Ofte inneholder et resultat mye som man egentlig ikke er interessert i. Søker du for eksempel etter ”mark” kan det hende at du får både dyr og enger tilbake.

Hvis andelen uinteressant er svært stor (mye støy), ønsker man gjerne å gjøre noe for å fjerne disse.

I denne oppgavetypen skal du bruke programmet til å fjerne støy fra store resultat slik at du ender opp med resultat der mesteparten er det en var ute etter.

---

c) **Utforsking.**

Av og til er man bare ute etter å se hva som finnes innenfor et spesielt felt. Man starter bare med en vag ide om hva man er interessert i og bruker innfallsmetoden for å navigere seg fremover etter hvert.

For å simulere dette, skal du søke etter ting fra ditt eget hjemfylke. Bruk så programmene til å danne deg et bilde av hva som finnes derfra.

d) **Finn noe som ligner.**

Når man ser igjennom et resultat er der ikke uvanlig at man finner noe som er særlig interessant. Det kan f.eks. være en bolle laget med en uvanlig teknikk eller et klesplagg av et spesielt materiale. Spørsmålet som da melder seg er om det finnes andre boller med samme teknikk eller andre klesplagg av samme materiale.

Oppgavene av denne typen er slik at du får oppgitt et søkeord. I det resultatet søkeordet gir, skal du finne frem til noe du selv synes er interessant og deretter prøve å bruke programmet til å finne fram til lignende objekter.

Fire oppgavetyper og tre program gir tolv muligheter. For hver av disse får du 1-2 oppgaver. Rekkefølgen programmene evalueres i er tilfeldig, mens rekkefølgen på oppgavetyperne alltid er a), b), c), d).

## Om evalueringen

Evalueringen består av to deler:

1. **Utføring av oppgaver.**

Alle oppgaver vil bli oppgitt på et eget ark. Husk at det er programmene som evalueres – ikke du. Denne delen tar ca. 30 minutter.

2. **Evalueringsspørsmål.**

Spørsmålene er delt inn i to deler. En del besvares for hver oppgavetype for hvert program, mens en del besvares for hvert program. Alle spørsmålene er tatt fra ”The Questionnaire for User Interaction Satisfaction” laget ved University of Maryland.

Her er alle spørsmålene (det vil bli utdelt avkryssingsskjema – dette er bare til orientering):

1	Bedømmelse – Eksakte oppgaver	forferdelig	fantastisk	
		1 2 3 4 5 6 7 8 9		N
				A
2	Bedømmelse – Støyfjerning	forferdelig	fantastisk	
		1 2 3 4 5 6 7 8 9		N
				A

3	Bedømmelse – Utforskning	forferdelig 1 2 3 4 5 6 7 8 9	fantastisk	N A
4	Bedømmelse – Finn noe som ligner	forferdelig 1 2 3 4 5 6 7 8 9	fantastisk	N A

5	Totalbedømmelse av programmet	forferdelig 1 2 3 4 5 6 7 8 9	fantastisk	N A
6	Organiseringen av skjermbildet er	forstyrrende 1 2 3 4 5 6 7 8 9	effektiv	N A
7	Mengden informasjon presentert på skjermen	utilstrekkelig 1 2 3 4 5 6 7 8 9	tilfredsstillende	N A
8	Måten informasjon på skjermen er organisert	ulogisk 1 2 3 4 5 6 7 8 9	logisk	N A
9	Tekst som vises på skjermen	forvirrende 1 2 3 4 5 6 7 8 9	begripelig	N A
10	Datamaskinen holder deg informert om hva den gjør	aldri 1 2 3 4 5 6 7 8 9	alltid	N A
11	Lære å bruke programmet	vanskelig 1 2 3 4 5 6 7 8 9	enkelt	N A
12	Komme i gang	vanskelig 1 2 3 4 5 6 7 8 9	enkelt	N A
13	Tid det tar å lære seg programmet	lang	kort	

			1	2	3	4	5	6	7	8	9	N
												A
1	Utforsking ved hjelp av prøv og feil	nedslående									oppmuntrende	
4												
												N
												A
1	Utforsking av programmet	risikabelt									trygt	
5												
												N
												A
1	Hastighet	for tregt									raskt nok	
6												
												N
												A

## C.3 Evaluation of Savanta

### Evaluering av verktøy for informasjonsgjenfinning

#### Introduksjon

Formålet med denne brukerevalueringen er å sammenligne brukbarheten til tre forskjellige verktøy for informasjonsgjenfinning i videodatabaser. Som testdeltaker skal du utføre en del oppgaver i hvert verktøy slik at du kan danne deg en oppfatning av hva du liker og ikke liker. Etterpå får du en del evalueringsspørsmål om hvert av verktøyene. Totalt vil dette ta ca. 1 time. Alle testverktøyene lar deg søke i en videodatabase som inneholder forelesningene til Steinar Line fra faget Informasjonsteknologi grunnkurs høsten 2003.

#### Om oppgavene

Noen ganger vet man nøyaktig hva man er ute etter, mens andre ganger er letingen mer tilfeldig. I tillegg kan målet med letingen endre seg underveis. For å se hvordan testverktøyene egner seg til forskjellige måter å søke på, er oppgavene delt opp i tre:

- Enkel informasjonsgjenfinning.
- Kompleks informasjonsgjenfinning.
- Utforsking.

Tre oppgavetyper og tre testverktøy gir ni muligheter. For hver av disse får du 2-3 oppgaver. Rekkefølgen verktøyene evalueres i er tilfeldig, mens rekkefølgen på oppgavetyperne alltid er a), b), c).

## Om evalueringen

Evalueringen består av to deler:

### 1. Utføring av oppgaver.

Alle oppgaver vil bli oppgitt på et eget ark. Husk at det er verktøyene som skal evalueres – ikke du. Denne delen tar ca. 45 minutter.

### 2. Evalueringsspørsmål.

Spørsmålene er avkryssingsoppgaver der man gir karakterer fra 1 til 9. Alle spørsmålene er gjengitt under (det vil bli utdelt avkryssingsskjema – dette er bare til orientering):

1	Bedømmelse – Enkel informasjonsgjenfinning	forferdelig 1 2 3 4 5 6 7 8 9	fantastisk	N A
2	Bedømmelse – Kompleks informasjonsgjenfinning	forferdelig 1 2 3 4 5 6 7 8 9	fantastisk	N A
3	Bedømmelse – Utforskning	forferdelig 1 2 3 4 5 6 7 8 9	fantastisk	N A

4	Totalbedømmelse av programmet	forferdelig 1 2 3 4 5 6 7 8 9	fantastisk	N A
5	Programmet var? å bruke	vanskelig 1 2 3 4 5 6 7 8 9	enkelt	N A
6	Uttrykkskraften var	utilstrekkelig 1 2 3 4 5 6 7 8 9	akkurat passe	N A
7	Mengden informasjon presentert på skjermen	for lite eller for mye 1 2 3 4 5 6 7 8 9	akkurat passe	N A
8	Teksten som vises på skjermen	forvirrende 1 2 3 4 5 6 7 8 9	begripelig	N A

---

9	Oppgaver kan løses på en rettfram måte	aldri	alltid	
		1 2 3 4 5 6 7 8 9		N
				A
1	Tilgjengelig hjelp	utilstrekkelig	akkurat passe	
0		1 2 3 4 5 6 7 8 9		N
				A

---

---

# References

---

- [Abowd and Beale, 1991]Abowd, G. D. and Beale, R. (1991). Users, systems and interfaces: A unifying framework for interaction. In *Proceedings of the HCI'91 Conference on People and Computers VI*, pages 73–87.
- [Adali et al., 1996]Adali, S., Candan, K. S., Chen, S.-S., Erol, K., and Subrahmanian, V. S. (1996). The advanced video information system: Data structures and query processing. *Multimedia Systems*, 4(4):172–186.
- [Ahlberg et al., 1992]Ahlberg, C., Williamson, C., and Shneiderman, B. (1992). Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of the ACM CHI'92 Conference on Human Factors in Computer Systems*, pages 619–626.
- [Aigrain et al., 1996]Aigrain, P., Zhang, H., and Petkovic, D. (1996). Content-based representation and retrieval of visual media: A state-of-the-art review. *Multimedia Tools and Applications*, 3(3):179–202.
- [Allen, 1983]Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.
- [Apple Computer Inc., 1992]Apple Computer Inc. (1992). *Macintosh Human Interface Guidelines*. Addison-Wesley Publishing.
- [Baeza-Yates and Ribeiro-Neto, 1999]Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.
- [Bates, 1989]Bates, M. J. (1989). The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407–424.

- 
- [Bates, 2002]Bates, M. J. (2002). Speculations on browsing, directed searching, and linking in relation to the bradford distribution. In Bruce, H., Fidel, R., Ingwersen, P., and Vakkari, P., editors, *Emerging Frameworks and Methods: Proceedings of the Fourth International Conference on Conceptions of Library and Information Science (CoLIS 4)*., pages 137–150. Libraries Unlimited.
- [Bellis, 2000]Bellis, M. (2000). Inventors of the modern computer. <http://inventors.about.com/library/weekly/aa060298.htm>.
- [Brin and Page, 1998]Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117.
- [Carrer et al., 1997]Carrer, M., Ligresti, L., Ahanger, G., and Little, T. D. C. (1997). An annotation engine for supporting video database population. *Multimedia Tools and Applications*, 5(3):233–258.
- [Catarci, 2000]Catarci, T. (2000). What happened when database researchers met usability. *Information Systems*, 25(3):177–212.
- [Catarci et al., 1997]Catarci, T., Costabile, M. F., Levioldi, S., and Batini, C. (1997). Visual query systems for databases: A survey. *Journal of Visual Languages and Computing*, 8(2):215–260.
- [Chen et al., 1996]Chen, M.-S., Han, J., and Yu, P. S. (1996). Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883.
- [Chin et al., 1988]Chin, J. P., Diehl, V. A., and Norman, K. L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems*, pages 213–218.
- [Christel and Martin, 1998]Christel, M. G. and Martin, D. (1998). Information visualization within a digital video library. *Journal of Intelligent Information Systems*, 11(3):235–257.
- [Comaniciu and Meer, 2002]Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.
- [Cooper and Byrd, 1998]Cooper, J. W. and Byrd, R. J. (1998). Obiwan - a visual interface for prompted query refinement. In *Hawaii International Conference on System Sciences*, volume 2, pages 277–285.
-



- 
- [Cordes, 2001]Cordes, R. E. (2001). Task-selection bias: A case for user-defined tasks. *International Journal of Human-Computer Interaction*, 13(4):411–419.
- [Creswell, 1994]Creswell, J. W. (1994). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications.
- [Cuff, 1980] Cuff, R. N. (1980). On casual users. *International Journal of Man-Machine Studies*, 12(2):163–187.
- [Cutting et al., 1992]Cutting, D. R., Pedersen, J. O., Karger, D. R., and Tukey, J. W. (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In Belkin, N. J., Ingwersen, P., and Pejtersen, A. M., editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*, pages 318–329. ACM.
- [Denning et al., 1989]Denning, P. J., Comer, D., Gries, D., Mulder, M. C., Tucker, A. B., Turner, A. J., and Youg, P. R. (1989). Computing as a discipline. *IEEE Computer*, 22(2):63–70.
- [Dix et al., 1998]Dix, A. J., Finlay, J. E., Abowd, G. D., and Beale, R. (1998). *Human-Computer Interaction*. Prentice Hall.
- [Doan et al., 1996]Doan, K., Plaisant, C., and Shneiderman, B. (1996). Query previews in networked information systems. In *Proceedings of the Third Forum on Research and Technology Advances in Digital Libraries, ADL '96*, pages 120–129.
- [Dönderler et al., 2003]Dönderler, M. E., Saykol, E., Ulusoy, Ö., and Güdükbay, U. (2003). Bilvideo: A video database management system. *IEEE MultiMedia*, 10(1):66–70.
- [Dumas and Redish, 1999]Dumas, J. S. and Redish, J. C. (1999). *A Practical Guide to Usability Testing*. Intellect.
- [Egnor and Lord, 2000]Egnor, D. and Lord, R. (2000). Structured information retrieval using xml. In *Informal Proceedings of the SIGIR Workshop on XML and Information Retrieval*.
- [Enser and Sandom, 2003]Enser, P. and Sandom, C. (2003). Towards a comprehensive survey of the semantic gap in visual image retrieval. In Bakker, E. M., Huang, T. S., Lew, M. S., Sebe, N., and Zhou, X. S., editors, *Proceedings of the second International Conference on Image and Video Retrieval Image and Video Retrieval CIVR 2003*, pages 291–299. Springer-Verlag.
- [Fisher, 1991]Fisher, J. (1991). Defining the novice user. *Behaviour and Information Technology*, 10(5):437–441.
-

- 
- [Flickner et al., 1995]Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. (1995). Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32.
- [Garcia-Molina et al., 2002]Garcia-Molina, H., Ullman, J., and Widom, J. (2002). *Database Systems: The Complete Book*. Prentice Hall.
- [Gershon et al., 1998]Gershon, N., Eick, S. G., and Card, S. (1998). Design: Information visualization. *Interactions*, 5(2):9–15.
- [Gevers and Smeulders, 1996]Gevers, T. and Smeulders, A. W. M. (1996). A comparative study of several color models for color image invariant retrieval. In *Proceedings of the first International Workshop on Image Databases and Multimedia Search*.
- [Gonzalez and Woods, 1992]Gonzalez, R. C. and Woods, R. E. (1992). *Digital Image Processing*. Addison-Wesley Longman Publishing Co. Inc.
- [Grice, 1975]Grice, H. P. (1975). Logic and conversation. In Cole, P. and Morgan, J. L., editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, San Diego, CA.
- [Grøtan, 2003]Grøtan, M. (2003). Image retrieval with focus on content based image retrieval. Master’s thesis, Norwegian University of Science and Technology (NTNU).
- [Han and Kamber, 2000]Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.
- [Haralick et al., 1973]Haralick, R. M., Shanmugam, K., and Dinstein, I. (1973). Textural Features for Image Classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3(6):610–621.
- [Hauglid and Midtstraum, 2001]Hauglid, J. O. and Midtstraum, R. (2001). Searching in image databases using the sesam approach. In *Proceedings of Norsk Informatikkonferanse*, pages 186–197.
- [Hauglid and Midtstraum, 2002]Hauglid, J. O. and Midtstraum, R. (2002). SESAM: searching supported by analysis of metadata. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC)*, pages 418–425. ACM Press.
- [Hearst and Pedersen, 1996]Hearst, M. A. and Pedersen, J. O. (1996). Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In Frei, H.-P., Harman, D., Schuble, P., and Wilkinson, R., editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 76–84. ACM Press.
-

- 
- [Heggland, 2002]Heggland, J. (2002). Ontolog: Temporal annotation using ad hoc ontologies and application profiles. In Agosti, M. and Thanos, C., editors, *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 118–128. Springer-Verlag.
- [Helms et al., 2000]Helms, J., Neale, D. C., Isenhour, P. L., and Carroll, J. M. (2000). Data logging: Higher-level capturing and multi-level abstracting of user activities. In *Proceedings of the 44th annual meeting of the Human Factors and Ergonomics Society*, pages 303–306. Human Factors and Ergonomics Society.
- [Hibino and Rundensteiner, 1997]Hibino, S. and Rundensteiner, E. A. (1997). User interface evaluation of a direct manipulation temporal visual query language. In *Proceedings of the fifth ACM international conference on Multimedia*, pages 99–107. ACM Press.
- [Hjelsvold and Midtstraum, 1994]Hjelsvold, R. and Midtstraum, R. (1994). Modelling and querying video data. In Bocca, J. B., Jarke, M., and Zaniolo, C., editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 686–694. Morgan Kaufmann.
- [Hjelsvold et al., 1995]Hjelsvold, R., Midtstraum, R., and Sandst, O. (1995). A temporal foundation of video databases. In Clifford, J. and Tuzhilin, A., editors, *Recent Advances in Temporal Databases, Proceedings of the International Workshop on Temporal Databases, Zurich, Switzerland, 17-18 September 1995*, pages 295–314. Springer.
- [Hu, 1962] Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, IT-8(2):179–187.
- [Hurst and Terry, 2000]Hurst, M. and Terry, P. (2000). Holiday 2000 e-commerce report. <http://www.creativegood.com/holiday2000/>.
- [ISO/IEC, 1992]ISO/IEC (1992). *Information technology — Database languages — SQL*. International Organization for Standardization.
- [Jähne, 2001]Jähne, B. (2001). *Digital Image Processing*. Springer-Verlag, 5 edition.
- [Katz, 1997]Katz, B. (1997). From sentence processing to information access on the world wide web. In *Proceedings of the American Association for Artificial Intelligence Conference, Spring Symposium*, pages 77–86. Stanford University.
-

- 
- [Kokkoras et al., 2002]Kokkoras, F. A., Jiang, H., Vlahavas, I. P., Elmagarmid, A. K., Houstis, E. N., and Aref, W. G. (2002). Smart videotext: a video data model based on conceptual graphs. *Multimedia Systems*, 8(4):328–338.
- [Kominek and Kazman, 1997]Kominek, J. and Kazman, R. (1997). Accessing multimedia through concept clustering. In *Proceedings of ACM CHI'97 Conference on Human Factors in Computing Systems*, pages 19–26.
- [Langley, 1999]Langley, P. (1999). User modeling in adaptive interfaces. In *Proceedings of the seventh international conference on User modeling*, pages 357–370. Springer-Verlag New York, Inc.
- [Langmyr, 2003]Langmyr, A. H. G. (2003). Image retrieval with focus on content based image retrieval. Master's thesis, Norwegian University of Science and Technology (NTNU).
- [Lew, 2001] Lew, M. S. (2001). *Principles of visual information retrieval*. Springer-Verlag.
- [Li et al., 1999]Li, Z.-N., Zaiane, O. R., and Tauber, Z. (1999). Illumination invariance and object model in content-based image and video retrieval. *Journal of Visual Communication and Image Representation*, 10(3):219–244.
- [Lu, 1999] Lu, G. (1999). *Multimedia Database Management Systems*. Artech House.
- [Magers, 1983]Magers, C. S. (1983). An experimental evaluation of on-line help for non-programmers. In *Proceedings of ACM CHI'83 Conference on Human Factors in Computing Systems*, pages 277–281.
- [Mayo, 1933]Mayo, E. (1933). *The Human Problems of an Industrial Civilization*. Macmillan.
- [McCracken and Wolfe, 2004]McCracken, D. D. and Wolfe, R. J. (2004). *User-Centered Website Development*. Prentice Hall.
- [Melton and Simon, 2002]Melton, J. and Simon, A. R. (2002). *SQL:1999 Understanding Relational Language Components*. Morgan Kaufmann.
- [Microsoft Corporation, 2002]Microsoft Corporation (2002). Windows xp visual guidelines. <http://www.microsoft.com/whdc/hwdev/windowsxp/downloads/default.mspx>.
- [Miller, 1956]Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.
-

- 
- [Miller, 1968]Miller, R. B. (1968). Response time in man-computer conversational transactions. In *AFIPS Conference Proceedings, 1968 Fall Joint Computer Conference*, volume 33, pages 267–277. AFIPS Press.
- [Mitchell and Jolley, 2001]Mitchell, M. and Jolley, J. (2001). *Research Design Explained, Fourth Edition*. Thomson Learning.
- [Moran, 1981]Moran, T. P. (1981). The command language grammar: A representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, 15(1):3–50.
- [Museenes Datatjeneste, 2003]Museenes Datatjeneste (2003). Primus. <http://www.mdt.no/primus/>.
- [Myers and Rosson, 1992]Myers, B. A. and Rosson, M. B. (1992). Survey on user interface programming. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 195–202.
- [Nack and Hardman, 2002]Nack, F. and Hardman, L. (2002). Towards a syntax for multimedia semantics. Technical Report INS-R0204, CWI.
- [Nack and Putz, 2001]Nack, F. and Putz, W. (2001). Designing annotation before it's needed. In *Proceedings of the 9th ACM International Conference on Multimedia*, pages 251–260, Ottawa, Ontario, Canada.
- [Neuman, 2002]Neuman, W. L. (2002). *Social Research Methods: Qualitative and Quantitative Approaches*. Pearson Allyn & Bacon.
- [Nielsen, 1992]Nielsen, J. (1992). Finding usability problems through heuristic evaluation. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 373–380.
- [Nielsen, 1993]Nielsen, J. (1993). *Usability Engineering*. Academic Press.
- [Nielsen, 1997]Nielsen, J. (1997). Loyalty on the web. <http://www.useit.com/alertbox/9708a.html>.
- [Nielsen, 1999]Nielsen, J. (1999). Do interface standards stifle design creativity? <http://www.useit.com/alertbox/990822.html>.
- [Nielsen, 2000]Nielsen, J. (2000). Why you only need to test with 5 users. <http://www.useit.com/alertbox/20000319.html>.
- [Nielsen, 2001]Nielsen, J. (2001). Did poor usability kill e-commerce? <http://www.useit.com/alertbox/20010819.html>.
-

- 
- [Nielsen and Molich, 1990]Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, pages 249–256.
- [Noldus Information Technology, 2003]Noldus Information Technology (2003). The observer 5.0. <http://www.noldus.com/products/observer/index.html>.
- [Norman, 1981]Norman, D. A. (1981). The trouble with UNIX. *Datamation*, 27(7):139–150.
- [Norman, 2004]Norman, D. A. (2004). *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books.
- [Oard, 1997]Oard, D. W. (1997). Speech-based information retrieval for digital libraries. In *Proceedings of AAAI Spring Symposium On Cross Language Text and Speech*.
- [Obrestad, 2002]Obrestad, J. (2002). Content based image retrieval. Master's thesis, Norwegian University of Science and Technology (NTNU).
- [O'Day and Jeffries, 1993]O'Day, V. L. and Jeffries, R. (1993). Orienteering in an information landscape: How information seekers get from here to there. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems*, pages 438–445.
- [Oomoto and Tanaka, 1993]Oomoto, E. and Tanaka, K. (1993). Ovid: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629–643.
- [Paap and Roske-Hofstrand, 1988]Paap, K. R. and Roske-Hofstrand, R. J. (1988). Design of menus. *Handbook of Human-Computer Interaction*, pages 205–235.
- [Page et al., 1998]Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University.
- [Phelps, 1997]Phelps, L. (1997). Active documentation: wizards as a medium for meeting user needs. In *Proceedings of the 15th annual international conference on Computer documentation*, pages 207–210. ACM Press.
- [Plaisant et al., 1999]Plaisant, C., Shneiderman, B., Doan, K., and Bruns, T. (1999). Interface and data architecture for query preview in networked information systems. *ACM Transactions on Information Systems*, 17(3):320–320.
-

- 
- [Preece et al., 2002]Preece, J., Rogers, Y., and Sharp, H. (2002). *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons.
- [Preece et al., 1994]Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. (1994). *Human-Computer Interaction*. Addison-Wesley Publishing.
- [Rao et al., 1995]Rao, R., Pedersen, J. O., Hearst, M. A., Mackinlay, J. D., Card, S. K., Masinter, L., Halvorsen, P.-K., and Robertson, G. C. (1995). Rich interaction in the digital library. *Communications of the ACM*, 38(4):29–39.
- [Reeves and Nass, 1998]Reeves, B. and Nass, C. (1998). *The media equation: how people treat computers, television, and new media like real people and places*. Cambridge University Press.
- [Reisner, 1981]Reisner, P. (1981). Human factors studies of database query languages: A survey and assessment. *ACM Computing Surveys*, 13(1):13–31.
- [Resnick et al., 1994]Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM '94 Conference on Computer Supported Cooperative Work*, pages 175–186.
- [Rui et al., 1999]Rui, Y., Huang, T. S., and Chang, S.-F. (1999). Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 3(1):39–62.
- [Rui et al., 1996]Rui, Y., She, A., and Huang, T. (1996). Modified fourier descriptors for shape representation – a practical approach. In *Proceedings of First International Workshop on Image Databases and Multimedia search*.
- [Salton, 1989]Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Publishing.
- [Santini and Jain, 1999]Santini, S. and Jain, R. (1999). Interfaces for emergent semantics in multimedia databases. In *Proceedings of the IS&T / SPIE Conference on Storage and Retrieval for Image and Video Databases*, pages 167–175.
- [Santos Jr. et al., 2001]Santos Jr., E., Nguyen, H., and Brown, S. (2001). Kavanah: Active user interface for information retrieval application. In *Proceedings of the Second Asia-Pacific Conference on Intelligent Agent Technology (IAT 2001)*.
-

- 
- [Sato et al., 2001]Sato, Y., Saito, M., and Koike, H. (2001). Real-time input of 3d pose and gestures of a user's hand and its applications for hci. In *Proceedings of the 2001 Virtual Reality Conference*, pages 79–.
- [Scheffé, 1959]Scheffé, H. (1959). *The Analysis of Variance*. Wiley.
- [Shafer and Agrawal, 2000]Shafer, J. C. and Agrawal, R. (2000). Continuous querying in database-centric web applications. *Computer Networks*, 33(1-6):519–531.
- [Shneiderman, 1983]Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69.
- [Shneiderman, 1994]Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77.
- [Shneiderman, 1997]Shneiderman, B. (1997). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing.
- [Skarbek and Koschan, 1994]Skarbek, W. and Koschan, A. (1994). Colour image segmentation - a survey. Technical report, Technical University Berlin.
- [Skou, 2003]Skou, C. V. (2003). Qualitative media analyzer. <http://www.cvs.dk/qma.htm>.
- [Smith and Pincever, 1991]Smith, T. G. A. and Pincever, N. (1991). Parsing movies in context. In *In Proceedings of USENIX (Summer 1991)*, pages 157–168.
- [Snodgrass and Vanderwart, 1980]Snodgrass, J. G. and Vanderwart, M. (1980). A standardized set of 260 pictures: norms for name agreement, image agreement, familiarity, and visual complexity. *Journal of Experimental Psychology: Human Learning and Memory*, 6(3):174–215.
- [Sonka et al., 1998]Sonka, M., Hlavac, V., and Boyle, R. (1998). *Image Processing: Analysis and Machine Vision*. Brooks Cole, 2 edition.
- [Stallings, 2003]Stallings, W. (2003). *Computer Organization and Architecture: Designing for Performance*. Prentice Hall, 6 edition.
- [Stricker and Orengo, 1995]Stricker, M. A. and Orengo, M. (1995). Similarity of color images. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 381–392.
- [Subrahmanian, 1998]Subrahmanian, V. S. (1998). *Principles of Multimedia Database Systems*. Morgan Kaufmann.
-



- 
- [Sun Microsystems, 2002]Sun Microsystems (2002). *Java Look and Feel Design Guidelines*. Addison-Wesley Publishing.
- [Taboada et al., 1996]Taboada, M., Marin, R., and Mira, J. (1996). On-line automatic help generation systems. *Journal of Intelligent Information Systems*, 7:261–285.
- [Tamura et al., 1978]Tamura, H., Mori, S., and Yamawaki, T. (1978). Texture features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8(6):460–473.
- [Tanin et al., 1997]Tanin, E., Beigel, R., and Shneiderman, B. (1997). Design and evaluation of incremental data structures and algorithms for dynamic query interfaces. In *1997 IEEE Symposium on Information Visualization (InfoVis '97)*, pages 81–86. IEEE Computer Society.
- [Tanin et al., 2000]Tanin, E., Lotem, A., Haddadin, I., Shneiderman, B., Plaisant, C., and Slaughter, L. (2000). Facilitating data exploration with query previews: A study of user performance and preference. *Behaviour and Information Technology*, 19(6):393–403.
- [Thomas and Gould, 1975]Thomas, J. C. and Gould, J. D. (1975). A psychological study of query by example. In *Proceedings of the National Computer Conference*, pages 439–445. AFIPS Press.
- [Tidwell and Fuccella, 1997]Tidwell, D. and Fuccella, J. (1997). Taskguides: instant wizards on the web. In *Proceedings of the 15th annual international conference on Computer documentation*, pages 263–272. ACM Press.
- [Vincent and Soille, 1991]Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598.
- [Wagemans et al., 1998]Wagemans, J., Notebaert, W., and Boucart, M. (1998). Lorazepam but not diazepam impairs identification of pictures on the basis of specific contour fragments. *Psychopharmacology (Berl)*, 138(3-4):326–333.
- [Weinshall and Kirkpatrick, 2004]Weinshall, D. and Kirkpatrick, S. (2004). Passwords you'll never forget, but can't recall. In *Extended abstracts of the 2004 conference on Human factors and computing systems CHI '04*, pages 1399–1402. ACM Press.
- [Weisberg et al., 1996]Weisberg, H., Krosnick, J. A., and Bowen, B. D. (1996). *An Introduction to Survey Research, Polling, and Data Analysis*. Sage Publications.
-

- 
- [Weiss et al., 1995]Weiss, R., Duda, A., and Gifford, D. K. (1995). Composition and search with a video algebra. *IEEE MultiMedia*, 2(1):12–25.
- [Whittaker et al., 1999]Whittaker, S., Hirschberg, J., Choi, J., Hindle, D., Pereira, F. C. N., and Singhal, A. (1999). Scan: Designing and evaluating user interfaces to support retrieval from speech archives. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 26–33. ACM Press.
- [Williamson and Shneiderman, 1992]Williamson, C. and Shneiderman, B. (1992). The dynamic homefinder: Evaluating dynamic queries in a real-estate information exploration system. In Belkin, N. J., Ingwersen, P., and Pejtersen, A. M., editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 338–346. ACM Press.
- [Wood et al., 1998]Wood, M. E. J., Thomas, B. T., and Campbell, N. W. (1998). Iterative refinement by relevance feedback in content-based digital image retrieval. In *Proceedings of the sixth ACM international conference on Multimedia*, pages 13–20. ACM Press.
- [Xu and Prince, 1998]Xu, C. and Prince, J. L. (1998). Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369.
- [Yahoo! Inc., 2004]Yahoo! Inc. (2004). Yahoo! investor relations. [http://docs.yahoo.com/info/pr/investor\\_metrics.html](http://docs.yahoo.com/info/pr/investor_metrics.html).
- [Yen and Scamell, 1993]Yen, M. Y.-M. and Scamell, R. W. (1993). A human factors experimental comparison of sql and qbe. *IEEE Transactions on Software Engineering*, 19(4):390–409.
- [Zhang and Lu, 2001]Zhang, D. and Lu, G. (2001). A comparative study on shape retrieval using fourier descriptors with different shape signatures. In *Proceedings of International Conference on Intelligent Multimedia and Distance Education (ICIMADE01)*, pages 1–9.
- [Ziegler and Fahnrich, 1988]Ziegler, J. E. and Fahnrich, K. P. (1988). Direct manipulation. *Handbook of Human-Computer Interaction*, pages 123–133.
- [Zloof, 1977]Zloof, M. M. (1977). Query-by-example: A data base language. *IBM Systems Journal*, 16(4):324–343.
-