

# Metadata Visualisation

Visual Exploration of File Systems  
and Search Result Sets based  
on Metadata Attributes

Erwin Weitlaner

# **Metadata Visualisation**

Visual Exploration of File Systems  
and Search Result Sets  
based on Metadata Attributes

Master's Thesis

at

Graz University of Technology

submitted by

**Erwin Weitlaner**

Institute for Information Processing and Computer Supported New Media (IICM),  
Graz University of Technology  
A-8010 Graz, Austria

December 1999

© Copyright 1999 by Erwin Weitlaner

Advisor: Univ.Ass. Dr. Keith Andrews

# **Visualisierung von Metadaten**

Graphische Darstellung von File Systemen  
und Suchergebnissen  
basierend auf deren Metadaten

Diplomarbeit  
an der  
Technischen Universität Graz

vorgelegt von

**Erwin Weitlaner**

Institut für Informationsverarbeitung und Computergestützte neue Medien (IICM),  
Technische Universität Graz  
A-8010 Graz

Dezember 1999

© Copyright 1999, Erwin Weitlaner

Diese Arbeit ist in englischer Sprache verfaßt.

Betreuer: Univ.Ass. Dr. Keith Andrews

## **Abstract**

Current Internet search tools often provide too many search results, are often lacking in recall and precision, and typically present search results only in the form of a textual list ranked by estimated relevance.

This thesis describes the Search Result Explorer, a program which visualises search result sets by plotting matching documents on a two-dimensional display. Metadata attributes of the search results such as document size, modification date, relevance, and number of links can be selectively mapped to the display axes. Further attributes can be mapped to the size and colour of a document's graphic representation in the display itself. The overall result is an explorable visualisation of a search result set with the possibility to compare them regarding to different attributes.

A second visualisation tool for file systems, the File Attribute Explorer, served as a prototype and is also described in this thesis.

## **Kurzfassung**

Gängige Internet Suchdienste liefern typischerweise zu viele Ergebnisse, geben keine oder unzureichende Zusatzinformationen und präsentieren die Dokumente ausschließlich in einer Liste geordnet nach abgeschätzter Relevanz.

Diese Diplomarbeit beschreibt den Search Result Explorer, ein Programm, das Suchergebnisse mit Hilfe einer zweidimensionalen Darstellung visualisiert. Verschiedene Metadaten der Dokumente, wie zum Beispiel Größe des Dokuments, Zeitpunkt der letzten Änderung, Relevanz und Anzahl der Verweise können auf den beiden Achsen aufgetragen werden. Weiters hängen Größe und Farbe der graphischen Darstellung von weiteren Dokument-Eigenschaften ab. Dadurch erhält man eine interaktive Darstellung von Suchergebnissen mit der Möglichkeit, die einzelnen Dokumente bezüglich verschiedener Eigenschaften zu vergleichen.

Eine zweite Anwendung zur Visualisierung von File Systemen, der File Attribute Explorer, diente als Prototyp und wird ebenfalls in dieser Diplomarbeit beschrieben.

*I hereby certify that the work presented in this thesis is my own and that work performed by others is appropriately cited.*

*Ich versichere hiermit, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.*

## **Acknowledgments**

I am indebted to my colleagues at the IICM, who have provided invaluable help and feedback during the course of my work.

I especially wish to thank my advisor, Keith Andrews, for his immediate attention to my questions, and for correcting the draft versions of this thesis.

Special mention to the xFIND team, Christian Gütl for his support in describing xFIND and Josef Moser for his answers and help concerning Java related xFIND questions.

Last but not least, without the great support and understanding of my family, girlfriend, and friends, this thesis would not have been possible.

Erwin Weitlaner  
Graz, Austria, December 1999

**Credits** The following figures are used to explain different visualisation techniques and they

are used with permission.

- Figure 3.1 was taken from [Cha95b] a Parallel Visual Explorer resource page.
- Figure 3.2 was taken from [CRO99].
- Figure 3.3 was taken from [NFH<sup>+</sup>96] the Envision homepage.
- Figure 3.4 was taken from [TSDS96], a paper about the Influence Explorer.
- Figure 3.5 was taken from [BF93] a resource at the Columbia University.
- Figure 3.6 was taken from [KK95].
- Figure 3.7 was taken from [CIP96].
- Figure 3.8 was taken from [VIS98].



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Metadata</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Metadata on the Net . . . . .	3
2.3	Categories of Metadata . . . . .	4
2.4	The Dublin Core (DC) . . . . .	5
2.4.1	History . . . . .	5
2.4.2	Elements of the Dublin Core . . . . .	5
2.4.3	Storage of Dublin Core Metadata . . . . .	7
2.4.4	Outlook . . . . .	8
2.5	Learning Object Metadata (LOM) . . . . .	9
2.5.1	Basic Structure . . . . .	10
2.5.2	Example: . . . . .	10
2.6	xFIND Quality Metadata Scheme (xQMS) . . . . .	11
2.6.1	Introduction . . . . .	11
2.6.2	Attribute Set . . . . .	11
2.6.3	Rating Process . . . . .	11
2.7	Resource Description Framework (RDF) . . . . .	12
2.7.1	Introduction . . . . .	12
2.7.2	Data Model . . . . .	13
2.7.3	Example . . . . .	14
2.7.4	Transporting RDF . . . . .	16
2.8	Hyperwave Metadata . . . . .	16
2.9	Introduction . . . . .	16
2.9.1	Hyperwave Object Attributes . . . . .	16
2.10	Summary . . . . .	17

<b>3</b>	<b>Multidimensional Visualisation</b>	<b>18</b>
3.1	Introduction . . . . .	18
3.2	Interface Issues . . . . .	18
3.3	Visualisation Techniques . . . . .	19
3.3.1	Parallel Coordinates . . . . .	20
3.3.2	Scatterplot Techniques . . . . .	21
3.3.3	Permutation Matrix - Data Histograms . . . . .	25
3.3.4	Worlds Within Worlds . . . . .	25
3.3.5	Pixel Representation and Distance Functions . . . . .	27
3.3.6	Force-Based Methods . . . . .	30
3.3.7	Combined Systems . . . . .	30
3.4	Summary . . . . .	33
<b>4</b>	<b>Java</b>	<b>34</b>
4.1	Introduction . . . . .	34
4.2	Java History . . . . .	34
4.3	The Java Programming Language . . . . .	34
4.4	The Java Foundation Classes (JFC) . . . . .	37
<b>5</b>	<b>File Attribute Explorer</b>	<b>38</b>
5.1	Introduction . . . . .	38
5.2	Design Principles for the File Attribute Explorer . . . . .	38
5.3	File Attribute Explorer Implementation . . . . .	39
5.4	Usage of the File Attribute Explorer . . . . .	40
5.5	User Interface . . . . .	40
5.5.1	The Button Toolbar . . . . .	42
5.5.2	The Location Field . . . . .	42
5.5.3	The Drawing Area . . . . .	43
5.5.4	The Status Bar . . . . .	48
5.5.5	Table of Files . . . . .	49
5.5.6	File Attribute Window . . . . .	50
5.5.7	Group of Files Window . . . . .	51
5.5.8	The Options Panel . . . . .	51
5.5.9	The About Window . . . . .	52
5.5.10	The Configuration File . . . . .	52
5.6	Selected Implementation Details . . . . .	52
5.6.1	Formatting Date and Time . . . . .	52
5.6.2	Changing the Look and Feel . . . . .	54
5.7	Future Work . . . . .	55

<b>6</b>	<b>Search Engines</b>	<b>56</b>
6.1	Introduction . . . . .	56
6.2	Types of Search Tools . . . . .	56
6.3	Relevance Calculation . . . . .	57
6.4	Intelligent Agents . . . . .	59
6.5	The Z39.50 Protocol . . . . .	59
6.5.1	Codifying Mechanics . . . . .	59
6.5.2	Content Semantics . . . . .	60
6.5.3	Future Use . . . . .	60
6.6	Result Set Visualisation . . . . .	60
6.6.1	Tile-Bars . . . . .	60
6.6.2	Arrow-Spheres . . . . .	61
6.6.3	Metadata Visualisation on Axes . . . . .	61
6.6.4	Force-Based Methods . . . . .	62
6.7	Summary . . . . .	62
<b>7</b>	<b>xFIND eXtended Framework for Information Discovery</b>	<b>63</b>
7.1	Introduction . . . . .	63
7.2	Concept . . . . .	63
7.3	Implementation . . . . .	64
7.3.1	The Gatherer . . . . .	64
7.3.2	The Indexer . . . . .	66
7.3.3	The Knowledge Broker . . . . .	66
7.4	Query Communication Format (QCF) . . . . .	67
7.5	xFIND Quality Metadata Scheme (xQMS) . . . . .	67
7.6	xFIND on the Web . . . . .	67
7.6.1	Operating fields . . . . .	68
<b>8</b>	<b>Search Result Explorer</b>	<b>69</b>
8.1	Introduction . . . . .	69
8.2	Design Principles . . . . .	69
8.3	Options Window . . . . .	72
8.4	Communication with the xFIND Server . . . . .	72
8.4.1	Query Language . . . . .	72
8.4.2	Receiving Document Attributes . . . . .	74
8.5	Selected Implementation Details . . . . .	75
8.5.1	Controlling a WWW Browser . . . . .	75
8.5.2	Customising Table Display and Event Handling . . . . .	76
8.6	Future Work . . . . .	78
<b>9</b>	<b>Concluding Remarks</b>	<b>79</b>

<b>A</b>	<b>File Attribute Explorer User Guide</b>	<b>81</b>
A.1	Installation . . . . .	81
A.2	Functions . . . . .	81
A.2.1	File Menu . . . . .	82
A.2.2	Navigation Menu . . . . .	83
A.2.3	View Menu . . . . .	83
A.2.4	Options Menu . . . . .	84
A.2.5	About Menu . . . . .	84
A.2.6	Flatten Option . . . . .	84
A.2.7	Search and Filter . . . . .	84
A.2.8	Zoom . . . . .	84
A.2.9	Axis Settings . . . . .	84
A.2.10	Display Settings Panel . . . . .	85
A.2.11	Mouse Functionality . . . . .	85
A.2.12	Sort Functions . . . . .	85
<b>B</b>	<b>Search Result Explorer User Guide</b>	<b>87</b>
B.1	Installation . . . . .	87
B.2	Functions . . . . .	87
B.2.1	File Menu . . . . .	88
B.2.2	Navigation Menu . . . . .	88
B.2.3	View Menu . . . . .	88
B.2.4	Options Menu . . . . .	88
B.2.5	About Menu . . . . .	90
B.2.6	Search and Filter . . . . .	90
B.2.7	Zoom . . . . .	90
B.2.8	Axis Settings . . . . .	91
B.2.9	Display Settings . . . . .	91
B.2.10	Mouse Functionality . . . . .	91
B.2.11	Sort Functions . . . . .	91
	Bibliography . . . . .	93

# List of Figures

2.1	A simple RDF statement. . . . .	13
2.2	RDF directed labeled graph. . . . .	15
3.1	The Parallel Visual Explorer uses parallel coordinates. . . . .	21
3.2	Visualisation using a scatterplot matrix. . . . .	23
3.3	Visualisation using Envision. . . . .	24
3.4	Histogram visualisation of a 4D function. . . . .	26
3.5	World Within Worlds. . . . .	28
3.6	VisDB visualising a large database. . . . .	29
3.7	Bead using force-based layout. . . . .	31
3.8	Visulab using different visualisation techniques. . . . .	32
4.1	Java Program Execution. . . . .	36
5.1	File Attribute Explorer Main Window. . . . .	41
5.2	The FAE button toolbar. . . . .	42
5.3	Current settings of the Drawing Area. . . . .	43
5.4	The Display Settings Panel. . . . .	44
5.5	Placement Algorithm using a Sweep Stripe. . . . .	46
5.6	Removing a file from the tree. . . . .	47
5.7	Adding a new file to the tree. . . . .	47
5.8	Rotation and grouping of items. . . . .	47
5.9	Table of Files. . . . .	50
5.10	File Attributes Window. . . . .	50
5.11	Group of Files Window. . . . .	51
5.12	File Attribute Explorer Options Panel. . . . .	52
5.13	File Attribute Explorer About Window. . . . .	53
6.1	TileBar visualisation of 5 documents. . . . .	61
6.2	Arrow-sphere visualisation of 3 documents. . . . .	61
7.1	xFIND Architecture . . . . .	65
7.2	xFIND Query Window. . . . .	68
8.1	The Search Result Explorer Main Window. . . . .	70

8.2	Table of Documents. . . . .	70
8.3	Document Detail Window. . . . .	71
8.4	Search Result Explorer Options Window. . . . .	72
A.1	File Attribute Explorer Main Window. . . . .	82
B.1	File Attribute Explorer Main Window . . . . .	89

# List of Tables

2.1	Typology of Metadata Formats. . . . .	4
2.2	Dublin Core Information Classes. . . . .	6
2.3	Learning Metadata Structure. . . . .	10
2.4	Quality Attributes in xQMS. . . . .	11
2.5	Hyperwave Object Attributes (system-defined, excerpt). . . . .	17
3.1	Multidimensional data visualisation techniques. . . . .	19
5.1	Possible zooming actions. . . . .	45
5.2	Runtime of placement operations. . . . .	48
5.3	Date formats using different locales. . . . .	54
5.4	Time formats using different locales. . . . .	54
6.1	Internet Search Tools Overview. . . . .	58
8.1	Return Attributes requested by the Search Result Explorer. . . . .	74
A.1	Keyboard shortcuts. . . . .	83
A.2	Possible Zooming actions. . . . .	85
A.3	Mouse Functionality in the File Attribute Explorer. . . . .	86
B.1	Keyboard Shortcut for the Search Result Explorer. . . . .	89
B.2	Possible Zooming actions. . . . .	90
B.3	Mouse Functionality in the Search Result Explorer . . . . .	92

# Chapter 1

## Introduction

This thesis describes two programs which visualise metadata. It contains the theoretical basics which influenced the development of the applications, describes similar approaches, the languages and systems used, and explains the usage of the programs themselves. One motivation to develop the programs is the failure of currently available information discovery systems to present explorable results. Some reasons for this fact and possible steps to overcome these shortcomings are also covered in this thesis.

Chapter 2 covers the field of metadata. After a short definition of the term itself, the chapter focuses on metadata used to characterise electronic resources. Examples of currently used and developed schemes and standards, with their advantages and shortcomings, are described. Richer metadata can be a first step to deal with the huge, exponentially growing information on the World Wide Web. Only a few of many interesting formats are analysed. The fact that there are different formats fitting different kinds of data lead to the realisation that an intelligent system has to implement an open concept to benefit from many kinds of metadata. Another related issue is the question of who provides, keeps up to date, and guarantees the trustworthiness of the metadata. Quality and audience ratings require both an expert who does the rating and a method to verify the data. The existence of metadata is especially important in search result visualisation, however it does not in itself protect against retrieving thousands of search results as a linear list.

Graphical presentation allows the visualisation of hundreds of results in one display. Chapter 3 gives an overview of existing multidimensional visualisation techniques. The difference to standard textual output, problems arising from high dimensionality, and the general advantages of graphical presentations are also issues in this chapter. The variety of different multidimensional data sets leads to an enormous number of different approaches and systems. Some fundamental technologies are explained and their suitability for document visualisation is discussed. Furthermore possible display manipulation techniques and their effect on understanding and handling large data sets are listed for each system. Once retrieved or extracted from given resources such information has to be displayed in a user friendly manner. The representation should benefit from the enormous capacity of human visual information processing. In a few tenths of a second, humans can recognise features in megapixel displays, recall related images, and identify anomalies, which would take much more time if the output were textually presented. Concepts of known multi-dimensional visualisation techniques and some interesting implementations are explained and presented in Chapter 3. The possibilities of the different techniques and their suitability for specific datasets are also discussed there.

Chapter 4 describes Java, the high level programming language, used in the implementations. The main reason for choosing Java is its platform-independence. Other characteristics are also described



briefly. One section covers the Java Foundation Classes, which are used to build the user interfaces of the programs and to configure their look and feel, and their date and time formats during program execution.

Chapter 5 describes the first of two presented programs. The File Attribute Explorer (FAE) presents file systems in a two dimensional plot with two further dimensions mapped to colour and size of the displayed graphic primitives. The program provides an intuitive, powerful interface with many features and functions, which gives an overview and understanding of the hierarchy and the contained files. The design decisions were influenced by the facts and guidelines presented in Chapter 3, whereby the attribute values of the files are seen as metadata.

Current search engine tools in the World Wide Web are analysed in Chapter 6. Information discovery is one of the most important tasks and it is not yet sufficiently addressed. Often the result sets are too large and the ranking is inappropriate. How the tools deal with metadata now, and how they should do in future is one factor to their further success. One key to improve the search process is to present the metadata of the resources in an intuitive, explorable fashion.

The xFIND search system is described in Chapter 7. Its future oriented concept tries to overcome many problems of other commonly available tools. One of those is its special treatment of different metadata formats, and its ability to provide this data to other applications. It is a scalable, distributed information system, which offers more than a common search engine. With xFIND useful information can be found in a wide range of information sources inside and outside organisational units. xFIND improves search results using metadata sets, quality rating labels, and Web site descriptions.

The second application benefits from the capabilities of the xFIND search system. It is called Search Result Explorer (SRE) and is presented in Chapter 8. It sends queries to, and visualises search result sets from xFIND. The dimensionality of the data increases compared to the file system exploration tool, because xFIND offers richer metadata. These additional attributes and their possibilities to create a meaningful result set visualisation can be one way to improve the search process.

In Appendix A and B user guides for the applications are found. Their similarities and differences especially in their usage are pointed out. These appendices are complete descriptions of the two applications and deliberately overlap parts of Chapters 5 and 8.

# Chapter 2

## Metadata

### 2.1 Introduction

Information which describes a data set is known as “metadata”. It is not the data itself, but rather “data about the data”. As the amount of information continues to grow exponentially and especially as the needs for learning expand equally dramatically, the lack of this information or metadata produces a critical and fundamental constraint on the ability to discover, manage and use information.

According to the Second World Wide Web conference [Cro95], metadata has two main functions:

- “To provide a means to discover that the data set exists and how it might be obtained or accessed.”
- “To document the content, quality, and features of a data set and so give an indication of its fitness for use.”

Typically, metadata supports a number of functions including location, discovery, documentation, evaluation and selection. These activities may be carried out by human end users or their (human or automated) agents. There is a variety of types of metadata. Traditional descriptive information, of the kind found in library catalogues, typically includes such attributes as author, title, some indication of intellectual content, and so on. Other types cover a variety of different research disciplines like physics, mathematics, or geography where standards are already defined, or will be defined in the near future. This chapter looks especially at metadata formats and schemes for resources found in the World Wide Web.

### 2.2 Metadata on the Net

Used in this context, metadata is the background information which describes the content, quality, condition, and other appropriate characteristics of the data. The aim is that the prospective user should be able to find out information about the data set without needing to access and investigate the data itself. It is recognised that, in an indefinitely large resource space, effective management of networked information will increasingly rely on effective management of metadata. The need for metadata services is already clear in the current Internet environment. It is unlikely that some monolithic metadata format will be universally used. The variety of different formats represent an attempt to meet the diverse requirements of different users.

	<b>Band One</b>	<b>Band Two</b>	<b>Band Three</b>
<b>Characteristics</b>	Simple Formats Proprietary Full Text Indexing	Structured Formats Emerging Standards Field Structure	Rich Formats International Standards Elaborate Tagging
<b>Examples</b>	Lycos Altavista Yahoo etc.	Dublin Core IAFA templates RCF 1807 SOIF LDIF	ICPSR CIMI EAD TEI,LOM MARC,xQMS

Table 2.1: Typology of Metadata Formats.

## 2.3 Categories of Metadata

For purposes of analysis three groups along a metadata spectrum which become successively richer in terms of fullness and structure is suggested by the metadata initiative of the World Wide Web Consortium [MET99]. The three bands within this spectrum allow the comparison of characteristics across groups of formats. Any one metadata format may not have all the characteristics of the band in which it is placed, but this grouping has proved beneficial in identifying the differences and similarities between formats.

**Band One** includes relatively unstructured data, typically automatically extracted from resources and indexed for searching. The data has little explicit semantics and does not support searching by field. Many Web services exist based on such data, and several global services are in heavy use. The metadata is not full enough to allow the user to make an objective relevance judgement in advance of actually retrieving the resource. It is seldomly possible to retrieve such data from the services and the information would be difficult to use in different applications.

**Band Two** includes data which contains full enough description to allow a user to assess the potential utility or interest of a resource without having to retrieve it or connect to it. The data is structured and supports fielded searching. Typically these records are simple enough to be created by non-specialist users, or not to require significant domain specific knowledge. Descriptions tend to be of discrete objects and do not capture multiple relationships between objects. Typically, but not essentially, descriptions are manually created, or are manual enhancements of automatically extracted descriptions, and they include a variety of descriptive and other attributes. They may be created to be loaded directly into a discovery service or to be harvested.

**Band Three** includes fuller descriptive formats which may be used for location and discovery, but also have a role in documenting objects or, very often, collections of objects. Typically, they are associated with research or scholarly activity, require specialist knowledge to create and maintain, and cater for specialist domain specific requirements.

The following sections present the Dublin Core (DC), the Learning Objects Metadata (LOM), the xFIND Quality Metadata Scheme (xQMS, see Chapter 7) and the Resource Description Framework (RDF) in detail. The first is located in Band Two (see Table 2.1) and influenced substantially the development of RDF. LOM and xQMS are richer formats and belong to Band Three. Dublin Core and Learning Object Metadata are presented because they give a good overview of the purpose of metadata. The xQMS format provides the ability to classify Internet resources depending on their quality and fitness for use. A further reason is that these three standards are supported by the xFIND

search system. RDF is an initiative of the World Wide Web Consortium [W3C99] and promises to provide a flexible syntactic foundation for Web based metadata including a variety of metadata association models that go beyond embedding descriptive metadata within resources.

## 2.4 The Dublin Core (DC)

The Dublin Core initiative is an international and interdisciplinary effort to define a core set of elements for resource discovery. Effective interchange of resource discovery information requires that there is an underlying architecture that supports conventions for the semantics, structure, and syntax of generalised metadata [Wei98].

### 2.4.1 History

The Dublin Core initiative has resulted in consensus concerning a base set of elements for descriptive metadata. The Dublin Core which emerged at a workshop held in Dublin in March 1995 were a core set of elements judged to be a reasonable foundation for the description of electronic resources. The initial workshop focused on description semantics directed expressly to the problem of discovery of electronic documents. A conceptual foundation for an architecture for metadata was laid at the second meeting, the so called Warwick Framework. “This framework, along with the Meta Content Framework, formed the nucleus for the development of the Resource Description Framework” [RDF98]. This Dublin Core Standard was broadened and improved in three further workshops, the last held in Helsinki in November 1997. A short summary of the results of this process is presented in this section. In the future the DC and RDF communities together want to provide a common architecture to support generalised metadata.

### 2.4.2 Elements of the Dublin Core

The Dublin Core consists of fifteen metadata fields, called the base elements. Each element has a descriptive name intended to convey a common semantic understanding of the element, as well as a formal single word label intended to make the syntactic specification of elements simpler for encoding schemes. Each of the fifteen elements is optional and repeatable. Furthermore, the DC elements may appear in any order, and with no significance being attached to that order. One design goal of the DC is global interoperability. Thus a controlled vocabulary for the element values is suggested with the element descriptions. Other controlled vocabularies are, and will be developed for interoperability within certain local domains. A metadata element’s meaning is unaffected by whether or not the element is embedded in the resource that it describes. The metadata elements fall into three groups which roughly indicate the class or scope of information stored in them (Table 2.2) [Wei98]:

- Elements related mainly to the Content of the resource.
- Elements related mainly to the resource when viewed as Intellectual Property.
- Elements related mainly to the Instantiation of the resource.

The following list is taken from [Wei98], a report from the DC Workshop 1997 held in Helsinki and describes the fifteen elements of the Dublin Core in detail:

1. **Title** Label: “Title” The name given to the resource, usually by the Creator or Publisher.

Content	Intellectual Property	Instantiation
Title	Creator	Date
Subject	Publisher	Type
Description	Contributor	Format
Source	Rights	Identifier
Language		
Relation		
Coverage		

Table 2.2: Dublin Core Information Classes.

2. **Author or Creator** Label: “Creator” The person or organisation primarily responsible for creating the intellectual content of the resource. For example, authors in the case of written documents, artists, photographers, or illustrators in the case of visual resources.
3. **Subject and Keywords** Label: “Subject” The topic of the resource. Typically, subject will be expressed as keywords or phrases that describe the subject or content of the resource. The use of controlled vocabularies and formal classification schemes is encouraged.
4. **Description** Label: “Description” A textual description of the content of the resource, including abstracts in the case of document like objects or content descriptions in the case of visual resources.
5. **Publisher** Label: “Publisher” The entity responsible for making the resource available in its present form, such as a publishing house, a university department, or a corporate entity.
6. **Other Contributor** Label: “Contributor” A person or organisation not specified in a Creator element who has made significant intellectual contributions to the resource but whose contribution is secondary to any person or organisation specified in a Creator element (for example, editor, transcriber, and illustrator).
7. **Date** Label: “Date” A date associated with the creation or availability of the resource. Such a date is not to be confused with one belonging in the Coverage element, which would be associated with the resource only insofar as the intellectual content is somehow about that date.
8. **Resource Type** Label: “Type” The category of the resource, such as home page, novel, poem, working paper, technical report, essay, dictionary. For the sake of interoperability, Type should be selected from an enumerated list that is currently under development in the workshop series.
9. **Format** Label: “Format” The data format of the resource, used to identify the software and possibly hardware that might be needed to display or operate the resource. For the sake of interoperability, Format should be selected from an enumerated list that is currently under development in the workshop series.
10. **Resource Identifier** Label: “Identifier” A string or number used to uniquely identify the resource. Examples for networked resources include URLs and URNs (when implemented). Other globally unique identifiers, such as International Standard Book Numbers (ISBN) or other formal names are also candidates for this element.
11. **Source** Label: “Source” Information about a second resource from which the present resource is derived. While it is generally recommended that elements contain information about the

present resource only, this element may contain a date, creator, format, identifier, or other meta data for the second resource when it is considered important for discovery of the present resource; recommended best practice is to use the Relation element instead.

12. **Language** Label: "Language" The language of the intellectual content of the resource. Where practical, the content of this field should coincide with RFC<sup>1</sup> 1766 (Tags for the Identification of Languages, <http://ds.internic.net/rfc/rfc1766.txt>). Examples include en, de, es, fi, fr, ja, th, and zh.
13. **Relation** Label: "Relation" An identifier of a second resource and its relationship to the present resource. This element permits links between related resources and resource descriptions to be indicated. Examples include an edition of a work (IsVersionOf), a translation of a work (IsBasedOn), a chapter of a book (IsPartOf), and a mechanical transformation of a dataset into an image (IsFormatOf). For the sake of interoperability, relationships should be selected from an enumerated list that is currently under development in the workshop series.
14. **Coverage** Label: "Coverage" The spatial or temporal characteristics of the intellectual content of the resource. Spatial coverage refers to a physical region (e.g., celestial sector); use coordinates (e.g., longitude and latitude) or place names that are from a controlled list or are fully spelled out. Temporal coverage refers to what the resource is about rather than when it was created or made available (the latter belonging in the Date element).
15. **Rights Management** Label: "Rights" A rights management statement, an identifier that links to a rights management statement, or an identifier that links to a service providing information about rights management for the resource.

### 2.4.3 Storage of Dublin Core Metadata

As already described, the metadata element's meaning is not effected by the way it is stored. The two most common ways to assign DC metadata description to a resource are listed next:

**Metadata may be included as part of the resource.** Embedded HTML tags is probably the simplest example. The <meta> element should be used, with name and content attributes set to the metadata element's name and value respectively. Example:

```
<html>
  <head>
    <title>Metadata and Visualization</title>
    <meta name="title" content="Metadata and Visualization">
    <meta name="creator" content="Erwin Weitlaner">
    <meta name="date" content="10/6/99">
  </head>

  <body>
    ...
  </body>

</html>
```

**Advantages:** No change is needed to existing browsers or search engines. Any set of attribute values can be represented.

---

<sup>1</sup>Request for Comment

**Disadvantages:** No constraint can be imposed on the semantics of the attributes names used, and name clashes may occur.

**Metadata may sit separately from the resource it describes.** The metadata is kept in an external document and a reference is made from within the HTML <head> element.

Example:

```
<html>
  <head>
    <title>Metadata and Visualization</title>
    <link rel = 'metadata'
      href = 'MetaVisualization.meta'>
  </head>

  <body>
    ...
  </body>

</html>
```

**Advantages:** Metadata is cleanly separated from the data. More powerful structuring abilities (e.g. nesting, repetition) are potentially available.

**Disadvantages:** There may be significant additional costs, in ensuring that metadata and data are kept in step and consistent. Conventions need to be established about how the metadata descriptions are to be mapped to HTML elements.

## 2.4.4 Outlook

As work on unqualified Dublin Core concludes, there is substantial momentum gathering behind the specification of qualifiers. This is partly in recognition of additional sub elements that have already appeared in many deployment projects (and the desire to formalise them), and partly in recognition of the need for substructure to support scheme qualifiers that are expected to provide the means for improving the precision of the Dublin Core.

Work on these subjects can be expected to change substantively as the work of the Dublin Core data model matures, and additional functionality will be essential for effective applications. Although simplicity remains one of the fundamental tenets of the Dublin Core metadata initiative, the option to refine the semantics of the element set through the addition of qualifiers is essential to allow effective deployment of the Dublin Core for resource discovery. To address these shortcomings, the qualifiers **SUBELEMENT**, **SCHEME** and **LANG** were proposed at the fourth Dublin Core workshop in Canberra, Australia. These optional qualifiers, now known as the “Canberra Qualifiers”, are:

- **SUBELEMENT**, which allows the refinement and clarification of an element’s content, such as, for example, refining the definition of DATE into Date Created, Date Acquired, etc.

Example:

**CREATOR** The Dublin Core CREATOR element has the following subelements:

- **PersonalName** The name of an individual associated with the creation of the resource. The PersonalName subelement itself has the following subelement:
- **Address** An electronic or physical address for the individual in question. This could be an electronic mail address, web page URL, postal address, etc.

- **CorporateName** The name of an institution or corporation associated with the creation of the resource. The CorporateName subelement itself has the following subelement:
  - \* **Address** An electronic or physical address for the institution or corporation in question. This could be an electronic mail address, web page URL, postal address, etc.
- **SCHEME**, which allows an element's value to be identified as part of an existing classification system, coding scheme, glossary or thesaurus, such as the Dewey Decimal Classification for books or the Art and Architecture Thesaurus for cultural heritage.
- **LANG**, which specifies the base language of an element's attribute values and text content; it is recommended that the values for this element are compliant with the scheme for creating language tags described by RFC 1766.

By optionally applying any or all of these qualifiers to the 15 elements of the Dublin Core, more detailed and semantically specific information about a resource can be encoded, thus assisting precision in the discovery and retrieval process for systems that support the use of these qualifiers.

As mentioned in the description of some of the DC elements their content should be taken from enumerated lists to provide and improve interoperability. These lists are currently under development in the workshop series.

## 2.5 Learning Object Metadata (LOM)

Learning Object metadata is defined as the attributes required to adequately describe a Learning Object. Any entity, digital or non-digital, which can be used, re-used or referenced during technology-supported learning can be seen as a Learning Object. Examples of technology-supported learning applications include computer-based training systems, interactive learning environments, intelligent computer-aided instruction systems, distance learning systems, web-based learning systems and collaborative learning environments [LOM99].

The IEEE<sup>2</sup> Learning Objects Metadata specification focuses on the minimal set of properties needed to allow Learning Objects to be managed, located, and evaluated. Relevant properties of Learning Objects include among others type of object, author, owner, terms of distribution, and format. Where applicable, Learning Object Metadata may also include pedagogical properties, such as:

- **Teaching or Interaction Style**
- **Grade Level**
- **Mastery Level**
- **Prerequisites**

It is possible for any given Learning Object to have more than one description, i.e. Learning Object Metadata set or instance of other metadata schemes. The standard supports security, privacy, commerce, and evaluation, but only to the extent that metadata fields are provided for specifying descriptive tokens related to these areas; the standard does not concern itself with how these features are implemented. The standard conforms to, integrates with, or references existing open standards and existing work in related areas. For example, the data scheme takes into account the efforts to standardise the description of content objects in general, as developed in the Dublin Core group.

<sup>2</sup>Institute of Electrical and Electronical Engineering



LOM Levels	Fields
<b>Categories</b>	Characteristics, Educational Use Dependent, General, Life Cycle Meta-Meta-Data, Relation, Rights Management, Technical
<b>Data Elements</b>	Amount, Attribution, Approach, Catalog Entry, Concept, Conditions, Contact, Contribute, Coverage, Create, Date, Description, Difficulty, Discipline, Duration, Educational Objective, Format, Granularity, Identifier, InstallationRemark, Interaction Quality, Keywords, Kind, Language, Level, LocSpec, Maximum Version, Meta meta data, Minimum Version, Monetary Unit, Operating System, Organization, OSRequirements, OtherPlatformRequirements, Person, Prerequisite, Price, Publish, Resource, Reciprocity, Role, Schema, Semantic Density, Size, Source, Structure, Support, TaxonPath, Terminate, Title, Type, Unit of Pricing, Validation, Version
<b>Abstract Data Types</b>	DBoolean, DCoverage, DDate, DDecimal, DEntry, DFormat, DIdentifier DInteger, DLangString, DLocSpec, DOrganisation, DPerson, DSource, DTimespan, DVocabulary

Table 2.3: Learning Metadata Structure.

### 2.5.1 Basic Structure

LOM is a hierarchical model with three levels:

1. **Categories** can have only data elements. Only categories can be used at the top level. Obligations are applied to categories.
2. **Data Elements** can have values of either Data Elements or Abstract Data Types. Obligations are applied to data elements.
3. **Abstract Data Types** define values. They do not contain Data Elements.

The general form of the structure is a top level of categories, each category contains a middle level of data elements which in turn are defined by the final level of abstract data types. The categories, there are eight currently, provide the top level of organisation of the data elements, and create the context within which the data elements are evaluated. The actual metadata values appear at the level of the final data elements, i.e. the data elements that draw their value from an abstract data type. A sequence of a category, one or more data elements, and a terminating abstract data type comprise a property. A data element has either one or more sub-data elements or exactly one abstract data type. The values of a data element are defined through abstract data types. All abstract data types have a first character of “D” to clearly differentiate them from the data elements and categories. An abstract data type is a semantic description of the value. A particularly useful abstract data type is DVocabulary. DVocabulary can be used to attach specific vocabularies within contexts in a schema.

### 2.5.2 Example:

- Technical
  - OS Requirements
    - \* Operating System: DVocabulary
    - \* Minimum Version: DDecimal
    - \* Maximum Version: DDecimal

Label	xQMS Field
kind of resource	tech.resourcekind
MIC message integrity check	tech.MIC-md5
topic	subject.topic
additional topic	subject.topic
authority (in science)	authority.sci
depth of information	accuracy.depth
width of information	accuracy.width
rater	rating.creator.person
rater service	rating.creator.organisation
label bureau	rating.creator.publisher
rating language	rating.language.nation
date of rating	rating.date.last_modified
signature	rating.signature-rsa-md5

Table 2.4: Quality Attributes in xQMS.

This structure describes the data element *OSRequirements* of the *Technical* category. This Requirements are described via three further data elements and their corresponding abstract data types.

## 2.6 xFIND Quality Metadata Scheme (xQMS)

### 2.6.1 Introduction

This standard is used by the xFIND search System described in Chapter 7. It extends existing metadata schemes by storing fields that describe the quality of the resources. The set of attributes and labels is influenced by Dublin Core and LOM formats, by suggestions made in classification catalogues, and by existing schemes used to classify articles in scientific journals [xFI99].

### 2.6.2 Attribute Set

xQMS is a more general format than Dublin Core in that it is not limited to documents, it can be used to describe servers, indexers, brokers, sites, pages, and paragraphs. A paragraph can be everything from an image, a table or a written text paragraph. Table 2.4 shows the attributes of xQMS used to describe the quality of resources, and the following description shows how the rating could be done. A list of the complete attribute set can be found on [xFI99].

### 2.6.3 Rating Process

While xQMS provides the fields to describe resource quality, the rating itself has to be done manually. At the moment, this information is held in the search engines database, and can be accessed via a special protocol. It seems possible that in the future similar services will be available on the World Wide Web (e.g. for specific topics). Especially for quality information these services will have to specify equal or corresponding values like those listed below for xQMS.

**Authority** The field *authority* can have different suffixes (“.gov”, “.edu”, “.sci”, “.org”, “.com”) depending on the area the data comes from. The value specified is a number between -3 which

means not applicable to 9 which indicates high quality (e.g. a PhD Thesis). This field addresses a major shortcoming of the present WWW by giving information about the quality of resources.

**Accuracy** The accuracy describes resources with regard to the depth and width of information given in a resource. The information with measures the coverage of different subthemes in the resource. The depth specifies the amount of detail information in the subthemes. The *accuracy.depth* and *accuracy.width* fields present this information and range from -3 for less coverage or information to +5 for full coverage and detailed information.

**Rating Information** The following fields specify information about the rating. In fact they can be seen as metadata about metadata. The field *rating.creator.person* names the person, who did the rating. This person can either be the author itself or a qualified person who rates resources. *rating.creator.organisation* specifies the name of the rating organisation and *creator.publisher* names the organisation which makes the rating public. Often the two last named fields will have the same entries but that is not a condition. The field *rating.date.last modified* specifies the date when the resource was qualified. If this date is older than the value of *date.last modified* (specifying the date when the resource itself was changed) the rating is not up to date and probably not accurate. Finally the *rating.language.nation* gives the language of the rating.

**Security** The user must be sure that the given metadata is trustworthy. This is guaranteed by the use of digital signatures: The rating organisation signs the metadata. It calculates the MD5 Message Digest and encodes it with its private key. This value is converted to a human readable format and stored in the field *rating.signature-rsa-md5*. Users who want to verify the metadata execute the same procedure in reverse order. If the Digest equals the MD5 Message Digest of the current resource it is (almost) sure, that the rating is trustworthy.

## 2.7 Resource Description Framework (RDF)

### 2.7.1 Introduction

The Resource Description Framework is a foundation for processing metadata. It provides interoperability between applications that exchange machine understandable information on the Web. RDF emphasises facilities to enable automated processing of Web resources. RDF metadata can be used in a variety of application areas [RDF98]:

- **Resource Discovery:** To provide better search engine capabilities.
- **Cataloguing:** For describing the content and content relationships available at a particular Web site, page, or digital library.
- **Intelligent Software Agents:** To facilitate knowledge sharing and exchange.
- **Content Rating**
- **Describing Collections** of pages that represent a single logical “document”.
- **Describing Intellectual Property Rights** of Web pages and resources.

RDF with digital signatures will be a key to build the “Web of Trust” for electronic commerce, collaboration, and other applications. RDF encourages the view of “metadata being data” by using *XML* (the eXtensible Markup Language) as its encoding syntax. The resources being described by RDF

are, in general, anything that can be named via a URI<sup>3</sup>. The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines the semantics of any application domain. The next section describes the RDF data model in brief, followed by an example describing a Web resource via RDF.

### 2.7.2 Data Model

RDF is a model for representing named properties and property values. Properties may be thought of as attributes of resources. In this sense properties correspond to traditional attribute-value pairs. Additionally properties can represent relationships between resources. In object-oriented design terminology, resources correspond to objects and properties correspond to instance variables. The RDF data model is a syntax-neutral way of representing RDF expressions. According to [RDF98] the basic data model consists of three object types:

**Resources:** All things being described by RDF expressions are called resources. A resource may be an entire Web page, such as the document “http://www.test.com/some.doc” for example. Unlike in DC, where resources are Web pages, a resource may also be a part of a Web page, a whole collection of pages, or an object that is not directly accessible via the Web (e.g. a printed book). Resources are always named by URIs and the extensibility of URIs allows the introduction of identifiers for any entity imaginable.

**Properties:** A property is a specific aspect, characteristic, attribute, or relation used to describe a resource. Each property has a specific meaning, defines its permitted values, the types of resources it can describe, and its relationship with other properties.

**Statements:** A resource together with a property and the value of that property form a RDF statement. These three individual parts of a statement are called subject, predicate and object. The object of a statement can be another resource or it can be a literal, or a simple string, or other primitive datatype defined by XML (see Figure 2.1 for a simple example).

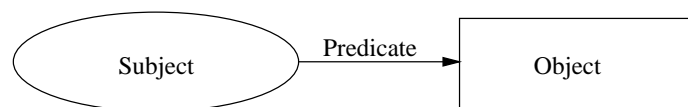


Figure 2.1: A simple RDF statement.

Frequently it is necessary to refer to a collection of resources. For example, to say that a work was created by more than one person, or to list the students in a course, or the software modules in a package needs such a construct. RDF containers are used to hold such lists of resources or literals. RDF defines three types of container objects:

**Bag:** An unordered list of resources or literals. Bags are used to declare that a property has multiple values and that there is no significance to the order in which the values are given. Bag might be used to give a list of part numbers where the order of processing the parts does not matter. Duplicate values are permitted.

**Sequence:** An ordered list of resources or literals. Sequence is used to declare that a property has multiple values and that the order of the values is significant. Sequence might be used, for example, to preserve an alphabetical ordering of values. Duplicate values are permitted.

---

<sup>3</sup>Uniform Resource Identifier

**Alternative:** A list of resources or literals that represent alternatives for the (single) value of a property. Alternative might be used to provide alternative language translations for the title of a work, or to provide a list of Internet mirror sites at which a resource might be found. An application using a property whose value is an Alternative collection is aware that it can choose any one of the items in the list as appropriate.

RDF statements can be shown using directed labeled graphs (also called “nodes and arcs diagrams”). In this data model both the resources being described and the values describing them are represented as nodes in a directed labeled graph. The arcs connecting pairs of nodes correspond to the names of the property types. Each arc is said to be labeled by the corresponding property type. The triple composed of a property type, a resource, and a value is an RDF property. Such a property can itself be the target node of an arc (i.e. the value of some other property) or the source node of an arc (i.e. it can have properties). In these cases, the original property must be “reified”, that is, converted into additional nodes and arcs. The term “reification” means converting the relation expressed by the arc into a concrete node. Reification allows the expression of modalities (e.g. beliefs about properties) or simply the attachment of properties to other properties.

### 2.7.3 Example

With the core defined, directed graph models of arbitrary complexity can be constructed and exchanged. Simple things, such as “Erwin Weitlaner” is the author of the document with the title “How to write java applications” whose URL is “http://www.test.com/some.doc” can be exchanged in the XML serialisation syntax as:

```
<?xml:namespace name="http://docs.r.us.com/bibliography info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD rdf syntax" as="RDF"?>

<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.test.com/some.doc" RDF:BAGID="D_001">
    <BIB:Author>Erwin Weitlaner</BIB:Author>
    <BIB:Title>How to write java applications</BIB:Title>
  </RDF:Description>
</RDF:RDF>
```

This serialisation can be modeled with a directed graph. As shown in Figure 2.2, the description itself becomes a bag node where the contents of the collection list the individual properties that were part of that particular description. The graph has ten nodes and thirteen arcs. The first three nodes are the resource “http://www.test.com/some.doc” and the property (string) values “Erwin Weitlaner” and “How to write java applications”. Arcs labeled *BIB:Author* and *BIB:Title* respectively connect the *some.doc* node with these string value nodes. The other seven nodes are added to fit into the RDF scheme. The bag node, identified as *D\_001*, is the source of three arcs; one arc labeled *RDF:InstanceOf* pointing to the node identified as *RDF:Bag* and two arcs labeled *RDF:1* and *RDF:2* respectively pointing to two unnamed nodes. From each of these two unnamed nodes there are four arcs. One arc from each node is labeled *RDF:PropObj* and points to the *some.doc* node. A second arc from each unnamed node is labeled *RDF:InstanceOf* and points to the node identified as *RDF:Property*. The remaining two arcs from each of the unnamed nodes are labeled *RDF:Value* and *RDF:PropName*. The *RDF:Value* arcs point respectively to the node containing the string “Erwin Weitlaner” and “How to write java applications”. The *RDF:PropName* arcs point respectively to the *BIB:Author* node and the *BIB:Title* node.

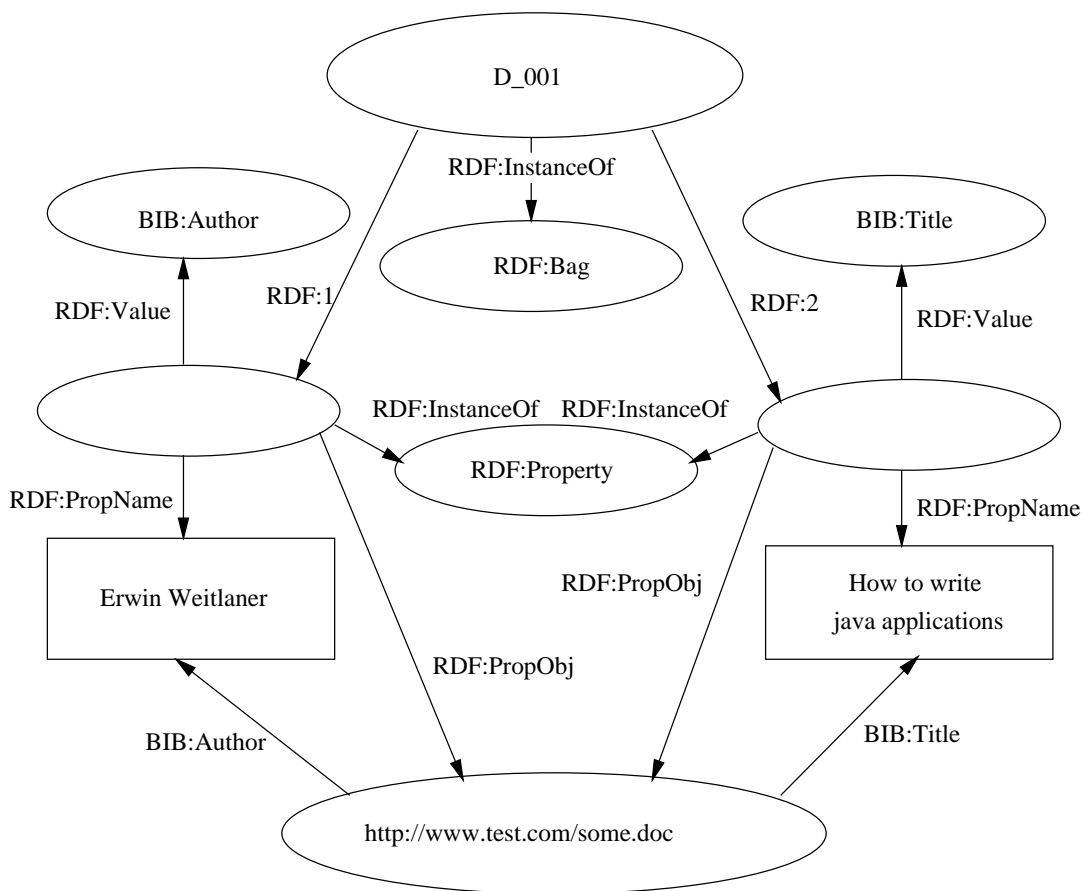


Figure 2.2: RDF directed labeled graph.

### 2.7.4 Transporting RDF

Descriptions may be associated with the resource they describe in one of four ways:

1. The Description may be contained within the resource (“embedded”; e.g. in HTML).
2. The Description may be external to the resource but supplied by the transfer mechanism in the same retrieval transaction as that which returns the resource (“along with”; e.g. via HTTP).
3. The Description may be retrieved independently from the resource, including from a different source (“service bureau”; e.g. using HTTP).
4. The Description may contain the resource (“wrapped”; e.g. RDF itself).

All resources will not support all association methods. In particular, many kinds of resources will not support embedding, and only certain kinds of resources may be wrapped.

## 2.8 Hyperwave Metadata

### 2.9 Introduction

Hyperwave [HYP99, Mau96] is the name of a hypermedia server, developed at the IICM<sup>4</sup> since 1990. Many intranet and Internet Web servers use a static, filebased architecture of HTML pages logically connected by static hyperlinks. In contrast, Hyperwave Information Server dynamically generates HTML pages and provides additional structuring elements. Hyperlinks are automatically generated without the need for programming. Thus they can not point to nowhere like as the static approach. Hyperwave Information Server is a broad and feature-rich application development platform that has been used to build applications in the areas of knowledge management, generalpurpose intranets and extranets, document management, web based training, project management, content management, and more. In fact, its customisability is one of the major strengths of Hyperwave Information Server. Hyperwave’s Object Model contains many different object attributes which can be realized as object’s metadata.

#### 2.9.1 Hyperwave Object Attributes

Unlike many other Web-based systems, Hyperwave does not store documents (which can be of any type) as plain files in a file system. Rather, it keeps them in an object-oriented repository (a kind of database), together with other information objects. These objects carry metadata, i.e. attribute names and values. The attributes differ in how, and by whom they are created. Each attribute has additional characteristics (e.g. if it is editable, or if it is indexed to enable rapid searching) [Kap99].

##### **System-defined attributes:**

These attributes have meaning to the system, i.e. they are set and/or interpreted by the system itself. The system also defines which of them need to be indexed. These attributes can be read-only if their values are set and maintained by the system (e.g. TimeCreated, TimeModified, FileSize) or read-write attributes which can be set by the user and are only interpreted by the system (e.g. the URL of the object).

---

<sup>4</sup>Institute for Information Processing and Computer Supported New Media

Attribute	Object class	Editable	Indexed	Multiple
Author	User	system/server	no	no
Document Type	Document	no	no	no
Group	User	system	no	yes
Home	User	system	no	no
Keyword	HGObject	user	yes	yes
Name	Collection	no	yes	no
Parents	HGObject	server	yes	no
Rights	HGObject	user	no	no
Subdocs	Collection	synthetic	no	no
TimeCreated	HGObject	user	yes	no
TimeExpire	HGObject	user	no	no
TimeModified	HGObject	server	yes	no
TimeOpen	HGObject	user	no	no
Title	HGObject	user	yes	yes
Type	HGObject	no	no	no

Table 2.5: Hyperwave Object Attributes (system-defined, excerpt).

**User-defined attributes:**

The user in this case is the application developer, who may define any number of attributes that should be attached to documents and other objects. The attributes can be indexed by the system administrator to make them searchable. The other “nonindexed” attributes cannot be searched for and have only informational purpose. Table 2.5 shows some system-defined Hyperwave attributes with their location in the class hierarchy and their other properties. “Object class” gives the “highest” object class in the class hierarchy where this attribute is applicable. “Editable” describes who may set this attribute. The value “User” means the attribute can be set by users with write access, “System” means the attribute can be set by the system user only, “Server” means the attribute is generated and stored by the server itself, “Synthetic” means that the attribute is not directly stored but set by the server on the fly. The “Indexed” field specifies whether or not the attribute is indexed (indexing allows rapid searching on the attributes values), while “Multiple” specifies whether the attribute may appear more than once in a single object record.

**2.10 Summary**

Metadata can be one key to provide effective information discovery on the Internet. Unfortunately metadata usage is far behind the possibilities. However providing rich trustworthy metadata for resources is tricky and time-consuming. In future maybe there will be inexpensive (but probably not free) services who guarantee for their metadata and, in combination with search tools, offer effective knowledge management.

It is obvious that the Hyperwave attribute structure provides richer metadata than the Dublin Core Standard requires. An advantage of the authoring process in Hyperwave is that every document has appropriate metadata because the creation of this data is an integral part of the authoring itself. It is reasonably straightforward to extract the elements of the Dublin Core from existing Hyperwave attributes.



## Chapter 3

# Multidimensional Visualisation

### 3.1 Introduction

Multidimensional information visualisations present data that is not primarily spatial. The number of attributes of the given data is more than three. These visualisations benefit from the fact that users can distinguish positions, colours, textures, shapes and relationships. Relationships can be shown in such displays by proximity, by containment, by connected lines, by colour-coding, etc. Displays containing hundreds or thousands of data elements can be scanned rapidly and efficiently for clusters, outliers, trends, and gaps.

For a visualisation to be effective, it must provide the user with a sense of the overall composition and layout of the space. Several questions have to be answered, when a data set is to be mapped to an interface, such as how to make the best mapping from attributes of the data to attributes of objects in the interface. Another issue that arises when addressing the overview of visualisations is how to fit large spaces on the screen and still allow some appreciation of the detail that resides there. Direct manipulation of visualisations can be accomplished with a variety of methods, such as pointing to select, dragging, and zooming. Feedback is immediate and intuitive in such environments. “The eye, the hand and the mind seem to work smoothly and rapidly as users perform actions on visual displays” [Shn97, p. 340].

Due to the extremely varying types of multidimensional data many approaches were and are made to provide meaningful visualisations of the information. Most methods are suitable only for some types of data (e.g. hierarchies can be viewed as trees or networks as graphs), but there exists no general solution for all kinds of data. Example applications of multi-dimensional visualisation schemes may use stock market statistics, factory production line data sets, a set of books in a library, a movie database, and almost any abstract and statistical information about any phenomenon. The used techniques are different but often the visualisations support similar functions in their displays, for example overview, detail view, relation between or history of the data sets. The most common functions are presented in the next section.

### 3.2 Interface Issues

- **Overview** Understand or get an overview of the whole or a part of the n-dimensional data. For example, finding patterns, relationships, clusters, gaps, and outliers of the data.
- **Zooming** is the technique for allowing a user to select a smaller region of the screen for display. Zooming includes any change in view from a larger portion to a smaller portion of a field or

Technique	Dimensions	Data Examples	Implementations
Parallel Coordinates	4 ... 10	financial data	Parallel Explorer
Scatterplot Matrix	4 ... 20	social data	Crossgraphs
Histograms	tens	relational databases	DataSplash
Rule Based Methods	tens	mathematical data	World within Worlds
Nested Dimensions	tens	financial data	n-Vision
Pixel Representation	tens	scientific data	VisDB
Force Based Methods	thousands	sets of documents	Bead
Combined Systems	tens	many kinds	VisuLab

Table 3.1: Multidimensional data visualisation techniques.

vice versa. Usually such views are available simultaneously to help users to preserve their sense of place.

- **Filtering** is the activity of weeding out uninteresting elements in a collection.
- **Details on Demand** At some point in interacting with a visualisation system, the user may decide to take a closer look at one or more objects in the field of view. When the requested view provides the content of the object, detail-on-demand has been provided. Most systems support this function and it is usually invoked by clicking on an item or group of items or by allowing the cursor to dwell on an object. In the former case, a dialog pops up that contains detailed information. In the latter case, a lens might be provided.
- **Relate** The relate function seeks to make explicit the relationships between objects in a display. It can also refer to representing relationships between data in multiple associated windows.
- **History** Maintaining histories is important for several reasons including place-keeping and supporting the ability to undo actions. Exploration in visualisations is a creative process and involves many sequential user actions to arrive at a satisfactory solution. The ability to retrace steps on a particular path is important.

### 3.3 Visualisation Techniques

The following techniques and projects are only a small selection of existing visualisations. Interested readers can find more information in [CMS98], a collection of papers describing different techniques and systems. Special interest is given to systems which try to visualise textual documents e.g search results or collections of topic specific papers. As a result of the existence of many kinds of different multidimensional datasets differing in size and dimensionality many approaches were, and are made to provide visualisations that do the above listed tasks. The usefulness depends on finding the best scheme for the data to visualise. Table 3.1 lists some concepts with their possible data dimensionality, area of application, and examples of implemented products. All of the shown visualisations provide several interactive manipulation and query features to improve imageability, exploration of and navigation through the given data. Some of the presented projects use more than one technique to provide the best possible overview over the data.

### 3.3.1 Parallel Coordinates

#### Display Concept

The display is obtained by taking the dimensions as vertical axes thereby arranging them parallel to each other. The individual data values are then marked off for each dimension onto the corresponding coordinate. The representation of a vector  $x = (x_1, x_2, \dots, x_n)$  is thus obtained by marking  $x_1$  on axis 1,  $x_2$  on axis 2 and so on through  $x_n$  on axis  $n$ . The resulting points on the axes are then joined by line sequences such that each vector is represented by a polygonal line. A point in  $n - dimensional$  space is hence equivalent to a broken line through  $n$  parallel coordinates in this particular visualisation. From the structure of the resulting display one can draw conclusions for the relationship of the corresponding data values. A group of lines with a similar gradient can, for example, indicate that their data records correlate positively. Since each vector is represented in a planar diagram, each vector component has essentially the same representation. Another advantage of this visualisation method is that the representation of all vectors in the same diagram means that a comparison of two vectors can easily be made. Different publications have referred to the parallel coordinate display as a method to solve problems from all kinds of domains, e.g. representing polytopes and hyperplanes, finding line neighbourhoods or detecting  $p$  flats in  $n$  dimensional space [Ins98].

#### Possible Operations

In the parallel coordinate display the following operations are possible and used in many systems:

- **Operations on Dimensions**

1. **Exchange Dimensions:** The selected axes are exchanged, that is, the corresponding dimensions are permuted.
2. **Dimension Hide/Unhide:** The selected dimension is hidden (one axis in the display disappears) or is shown again.

- **Operations on Data Items**

1. **Data Hide/Unhide:** The selected data items (or broken lines) are hidden or become visible.
2. **Data Pick:** Only selected data items (or broken lines) stay visible, the other data items are hidden. Flexible selection possibilities improve the overview. Often an interval on one axis is specified and all the data items in this interval (their lines) are colour coded.

#### Example

Figure 3.1 shows financial data taken from the global money markets from 1985 to 1993. Data sets of different years have different colour codings. The years 1986 and 1992 are being examined. In 1986, the Yen was the most volatile currency among the three, stocks were low, bonds were medium, and there was a gap in the gold prices. In 1992, the British Pound Sterling was the most volatile currency, bonds and gold were low, and stocks were high.

#### Projects

Projects using parallel coordinates to visualise multidimensional data include:

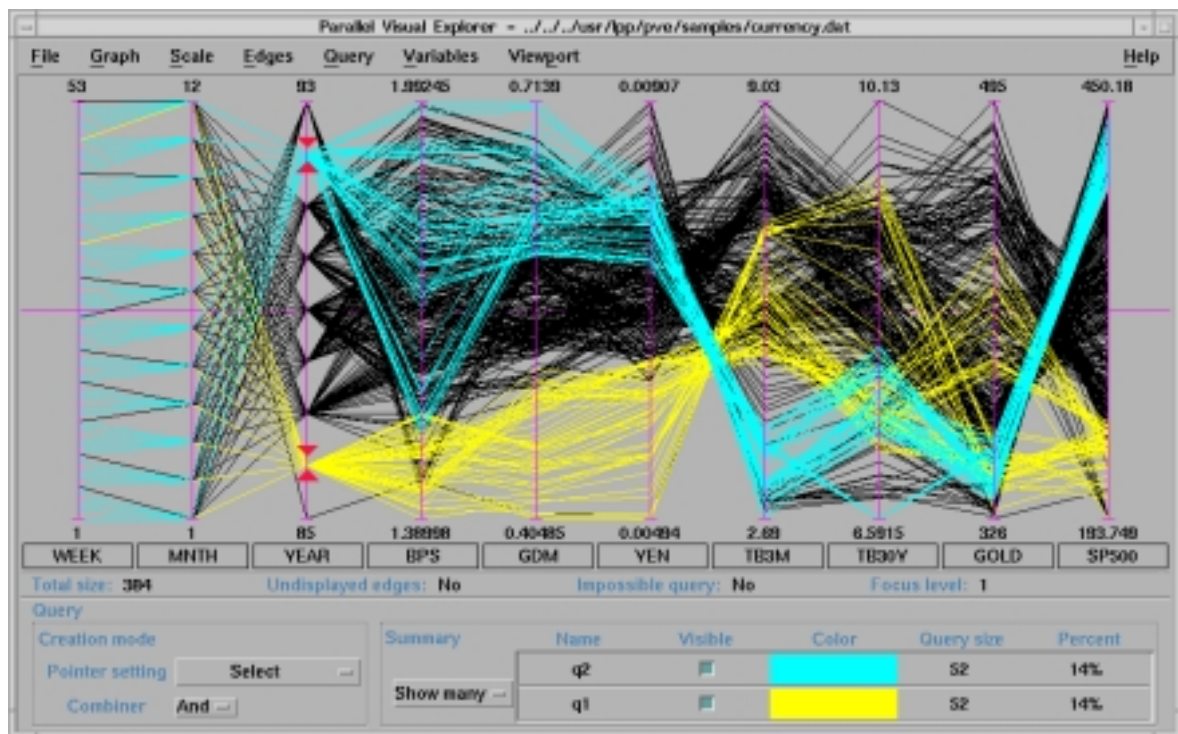


Figure 3.1: The Parallel Visual Explorer uses parallel coordinates.

- **Parallel Visual Explorer** [Cha95b]
- **WinViz** [WIN99, OL96]
- **SPSS** [SPS98]

### 3.3.2 Scatterplot Techniques

#### Display Concept

The classic scatter diagram is a fundamental visualisation method, showing the relationship between two variables. An individual scatterplot does, however, not generalise readily beyond three dimensions. For the visual representation of multivariate data a more elaborate construct is needed: In the Scatterplot Matrix  $n$  dimensions are projected onto  $n * (n - 1)$  scatterplots, where each pair of dimensions has two scatterplots showing their relation. The visualisation consists of an array of scatter diagrams arranged in the form of a  $n * n$  matrix. Each dimension of the original data defines one row and one column of the matrix. The entry where row  $i$  intersects column  $k$  is a scatter diagram of  $x_i$  versus  $x_k$  where the data records are  $n$  dimensional vectors  $x = (x_1, x_2, \dots, x_n)$ . Since  $x_k$  versus  $x_i$  shows the same relationship with only the axes interchanged, the scatterplot matrix is symmetrical. The data values for a particular dimension are shown both in the respective row on the y-axis and in the respective column on the x-axis. The dimension names are often written into the main diagonal of the matrix. Due to the projection of  $n$  dimensions onto  $n * (n - 1)$  scatterplots each data record (that is, each row of the original table) appears as a point in all these 2-dimensional plots, each value therefore appears  $2 * (n - 1)$  times. Variants of this approach use fewer plots but the attributes which are mapped to the axes in these plots can be changed by users. The result is more resolution with

less overview over the data. The higher resolution is often used to display more information (e.g. by iconic representations of data items and colour encoding) about the displayed items to the user. Particularly in document visualisations the mapping from attributes to graphic primitives is becoming more sophisticated, for example the graphics itself may indicate similarities to certain keywords.

### Possible Operations

The following manipulation techniques are commonly used in visualisations using Scatterplot Matrices:

- **Axis Setting** The user may set and change the categories and their ordering assigned to the axis.
- **Direct Manipulation** The user interface provides abilities (e.g. sliders or filters) to set ranges in each dimension. Items which lie within the same range have different colour codings than other items.

### Examples

- **Classic Scatterplot Matrix:** Figure 3.2 shows the scatterplot matrix of a four dimensional data set.
- **Plot with Axis Selection:** For example Envision [NFF96] uses the variant with only one plot with switchable axes to display “multidimensional” metadata of documents. The programs File Attribute Explorer and Search Result Explorer use a similar display concept to visualise their display items.

### Projects

The following list shows projects which use scatter plots to visualise multidimensional data:

- **CrossGraphs** [CRO99]
- **xGobi** [DB98]
- **Envision** [NFH<sup>+</sup>96, NFF96] provides powerful information visualisation by displaying search results as a matrix of icons, with layout semantics under user control. Envision’s Graphic View (see Figure 3.3) interacts with an Item Summary Window giving users access to bibliographic information, abstracts, and full content. While many visualisation interfaces for information retrieval systems depict ranked query-document similarity, Envision graphically presents a variety of document characteristics and supports an extensive range of user tasks. Formative usability evaluation results show great user satisfaction with Envision’s style of presentation and the document characteristics visualised. Its uniqueness in the variety of document characteristics visualised and in the flexibility afforded users to change the visualisation to suit their current information needs was an impulse to implement an application with at least this flexibility.

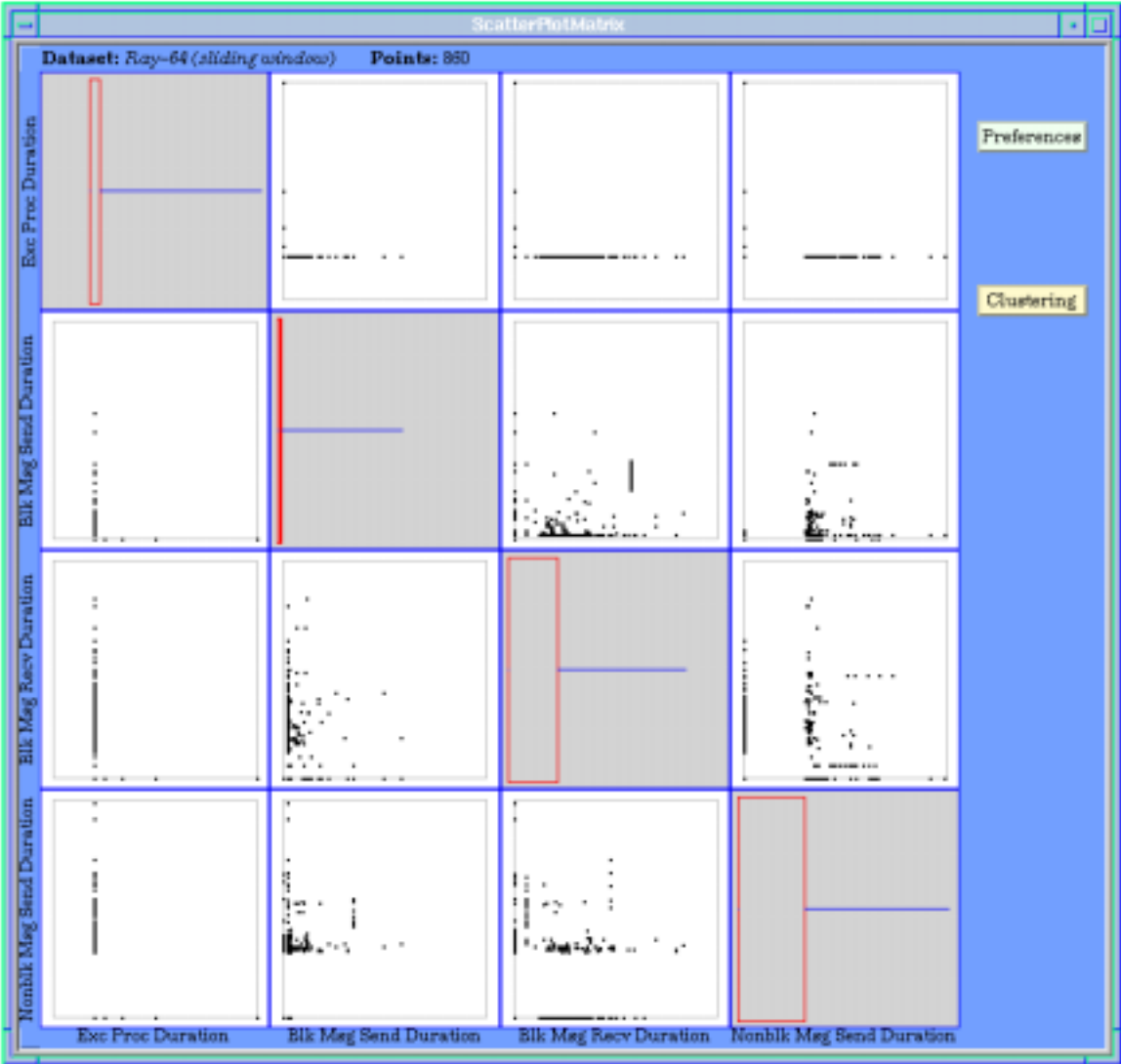


Figure 3.2: Visualisation using a scatterplot matrix.

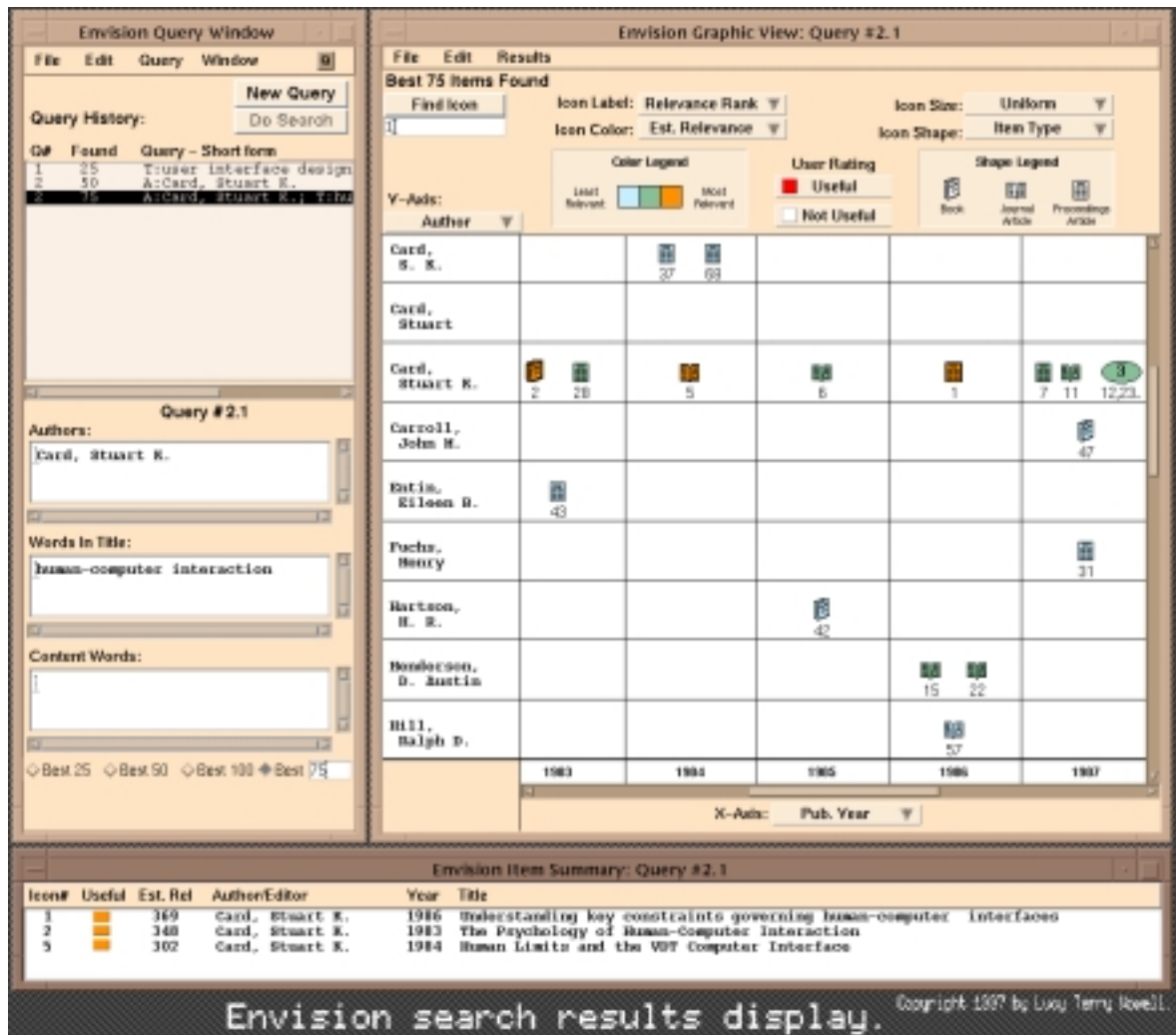


Figure 3.3: Visualisation using Envision.

### 3.3.3 Permutation Matrix - Data Histograms

#### Display Concept

The numerical values of the contingency table are transformed into a matrix of simple graphical elements such that the structure of the data set becomes immediately visible. The basic structure of the permutation matrix consists of the rows and columns and the labels of these rows and columns. The data values can be displayed with multiple column charts, bars, coloured rectangles or other appropriate graphical objects where colour represents individual data values. The permutation matrix is a visualisation method suited to show the overall appearance of the data as a collection and not so much the individual quantitative values.

#### Possible Operations

The listed operations and manipulations are possible especially in the permutation matrices.

- **Automatic Permutation** The program will permute the rows and columns of the matrix automatically such that a pattern emerges if one exists. It can detect similarities or relationships between variables itself and therefore avoid bias.
- **Range Definition** The user can place exploratory limits on parameters, thereby defining ranges of those quantities. This action leads to colour linking those items that lie within the selected range on all histograms.

#### Example

The example graph shown in Figure 3.4 plots example values of the abstract mathematical function:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = f(s_1, s_2, s_3, s_4)$$

with limits set to the parameters  $s_1 \dots s_4$  and colour encoded results for  $x_1 \dots x_4$ .

#### Projects

The following list contains projects which use the Permutation Matrix as their display concept.

- **Influence Explorer** is an interactive visualisation tool to support engineering design. “This interactive visualisation allows fluent exploration of such problems and subsequent acquisition of insight” [TSDS96].
- **ViSta** [HMV96]

### 3.3.4 Worlds Within Worlds

#### Display Concept

“Worlds within Worlds” is a powerful method, capable of displaying many dimensions, and yet is readily understandable. One common approach to reducing the complexity of a multivariate function



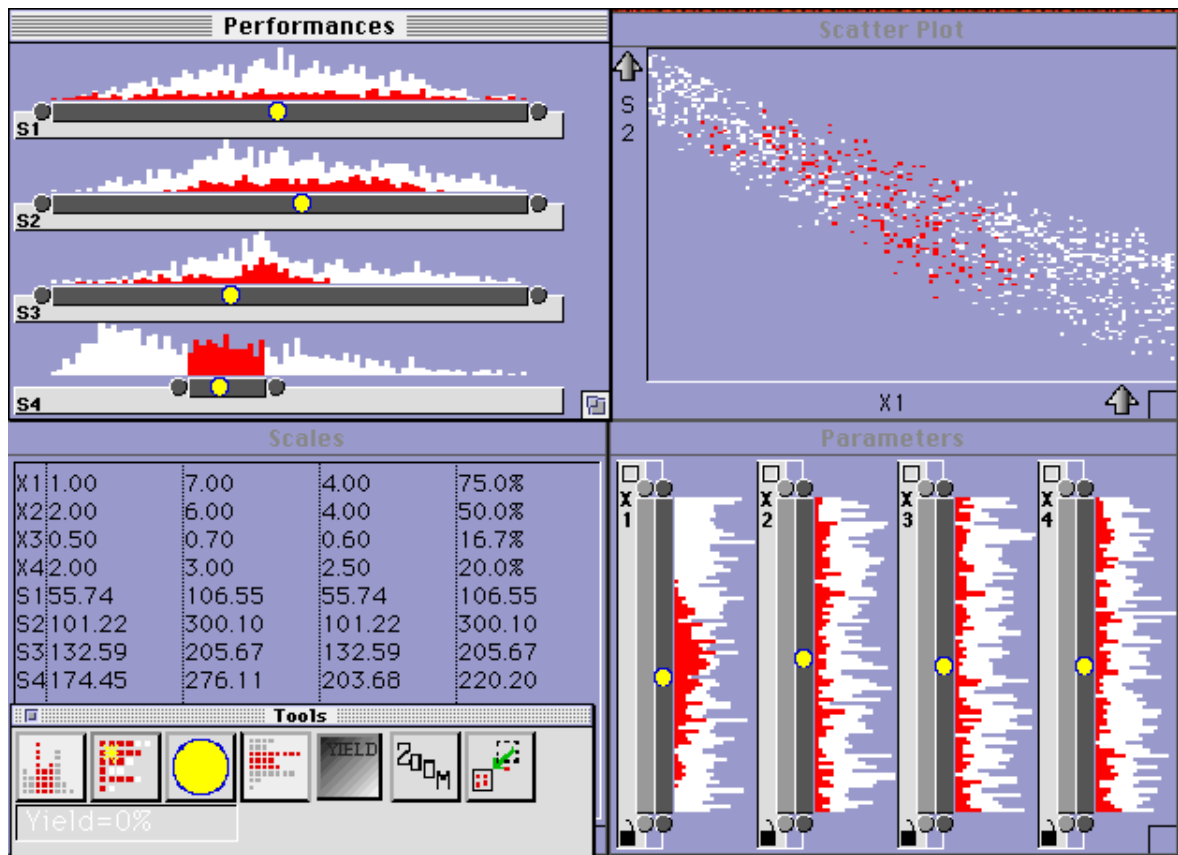


Figure 3.4: Histogram visualisation of a 4D function.

is to hold one or more of its independent variables constant. Each constant corresponds to taking an infinitely thin slice of the world perpendicular to the constant variable's axis, reducing the world's dimension. If the dimensionality is reduced to 3D, the resulting slice can be manipulated and displayed using conventional 3D graphics hardware. Although this simple approach effectively slices away the higher dimensions, it is possible to add them back. To do this, the 3D world is embedded in another 3D world. The position of the embedded world's origin relative to the containing world's coordinate system specifies the values of up to three variables that were held constant in the process of slicing the world down to size. This process can then be repeated by further recursive nesting.

### Possible Operations

User can create these virtual worlds by specifying the assignment of variables to coordinate system axes. They can deposit multiple copies of the same world, or copies of different worlds within a containing world, to allow the copies to be compared visually. Each copy has a different constant set of values of the containing world's variables, based on its position. Worlds may be rotated and scaled about their origins, and viewed from different positions [BF93].

### Example

As an example a function  $f(x_1, x_2, x_3, x_4, x_5)$  is considered. Following the description above, first constant values for three variables  $x_3, x_4, x_5$ , called  $c_3, c_4, c_5$  are selected. This selection results in a new function  $f'(x_1, x_2) = f(x_1, x_2, c_3, c_4, c_5)$ . The function  $f'(x_1, x_2)$  is easy to graph in 3D as a surface plot, with  $x_1$  on the x-axis,  $x_2$  on the z-axis, and the value of the function on the vertical y-axis. This graph is shown in Figure 3.5. The values of  $x_3, x_4$  and  $x_5$  are shown on a separate set of axes to let the user select particular values for these parameters. Selecting a point within this larger graph determines the particular values of  $c_3, c_4$  and  $c_5$  used in the smaller graph. Thus, the contents of the smaller graph depend on the location of some interactive mark in the larger graph.

### Project

- **AutoVisual** [BF93]

## 3.3.5 Pixel Representation and Distance Functions

### Display Concept

The major goal of this concept is to visualise large amounts of arbitrary multidimensional data. In this approach reference points (or regions) in multidimensional space are introduced and only the data items that are "closest" to the reference point are visualised. The "closeness" is determined using distance functions for each of the dimensions. The distance functions are datatype and application dependant and must be provided by the application. Having calculated the distances for each of the dimensions which are part of the reference point specification, the distances are combined into the closeness factor. Important aspects such as normalising and weighting the distances of the different dimensions speed up the layout process and make it more flexible. The basic idea for visualising the data items is to map the value ranges of the different dimensions to colour and represent each data item by multiple pixels being coloured according to the distance values for each of its dimensions. The coloured pixels are then displayed on the screen with data items fitting into the reference region centred in the middle of the window and the other data items are arranged rectangular spiral shaped

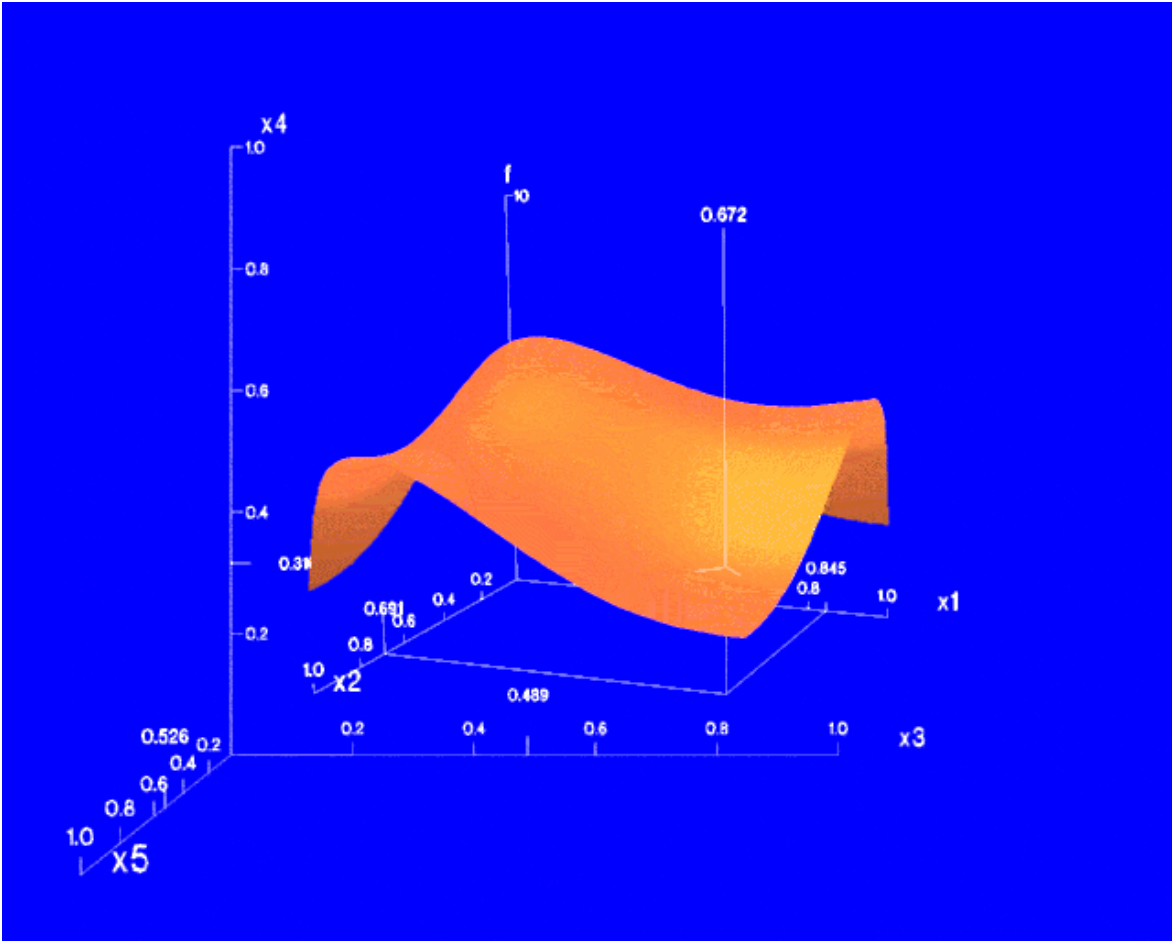


Figure 3.5: World Within Worlds.

around this region according to the overall closeness factor. A separate window is provided for each of the dimensions. In these separate windows, the pixels for each data item are placed at the same relative position, allowing the user to relate the visualisation of the different dimensions [KK95].

### Possible Operations

- **Reference point selection** the user may interactively change the reference point (or region).

### Example

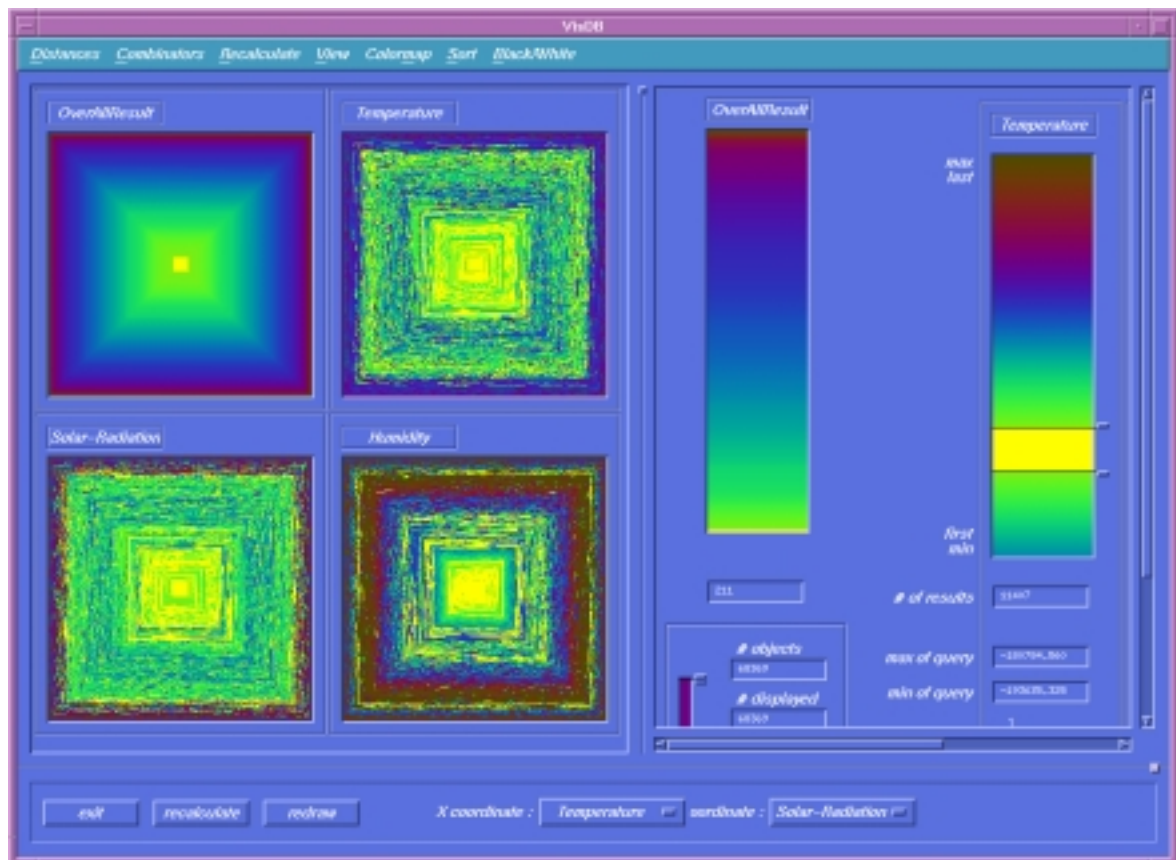


Figure 3.6: VisDB visualising a large database.

### Project

- **VisDB:** is a system developed at the University of Munich and is a sophisticated tool for visualising and analysing large databases. The key idea of the VisDB system is to support the exploration of large databases by using the abilities of the human vision system to analyse visualisations of large amounts of data very efficiently. The goal of the VisDB system is to provide visualisations of large portions of a database, allowing properties of the data and structure in the data to become perceptually apparent. By arranging and colouring the pixels according to the relevance of the data items with respect to the query, the user gains a visual impression of the resulting data set. Using sliders for each condition of the query, users may change queries

dynamically and receive immediate feedback from the visual representation of the resulting data set [KK94].

### 3.3.6 Force-Based Methods

#### Display Concept

This concept tries to visualise data where dimensionality may be in the thousands. Examples are sets of documents or financial data. The layout is produced by using a metric of similarity or “data distance” between data pairs. In the case of sets of documents, this metric is often based on word co-occurrence. In this process the members of the set of documents iteratively push and pull on each other to create an emergent structure. Similar documents which are far apart pull towards each other. Dissimilar documents which are too close together push away from each other. The system works to reduce these forces and energies i.e. tries to minimise the total stress of the system. As these forces gradually ease off, the model of the corpus settles into shape. By defining a distance metric and employing a similar layout/optimisation algorithm, other types of information can be laid out. Due to the layout process, the axes of the resulting two dimensional space have no inherent meaning.

#### Example

Figure 3.7 shows an example from *Bead*, a view from far above on an “island” of about 500 documents.

#### Projects

- **Bead** [Cha93, Cha95a] is a visualisation system which has evolved over years in both the layout and presentation of “maps” of multidimensional data. Layouts are built using physically based force models with special effort to enhance legibility and the model of how past searches, selections etc. relate to each other within the layout.
- **MDS** [ZC87]

### 3.3.7 Combined Systems

Many existing projects and systems use the idea of having various of the above described visualisations at the same time. Users can see changes on all visualisations simultaneously by just manipulating one. Further dimensions are pushed into the system, by colour encoding, iconic representations, 3D visualisation approaches etc. but all of the systems use kinds of the above described techniques.

#### Example

Figure 3.8 is a screenshot taken from [VIS98] and shows a multidimensional dataset, visualised with different techniques.

#### Projects

- **DEVise** [LRB<sup>+</sup>97]

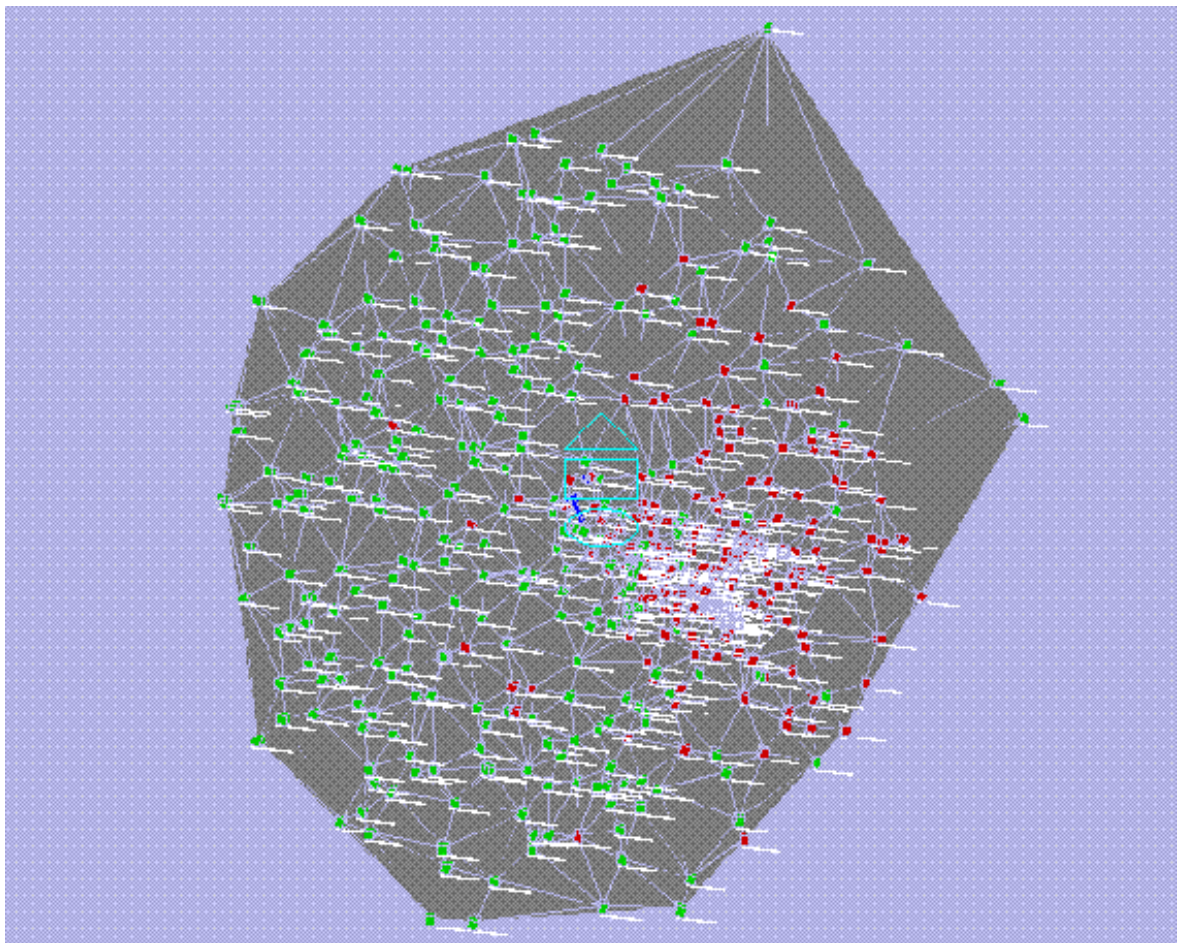


Figure 3.7: Bead using force-based layout.

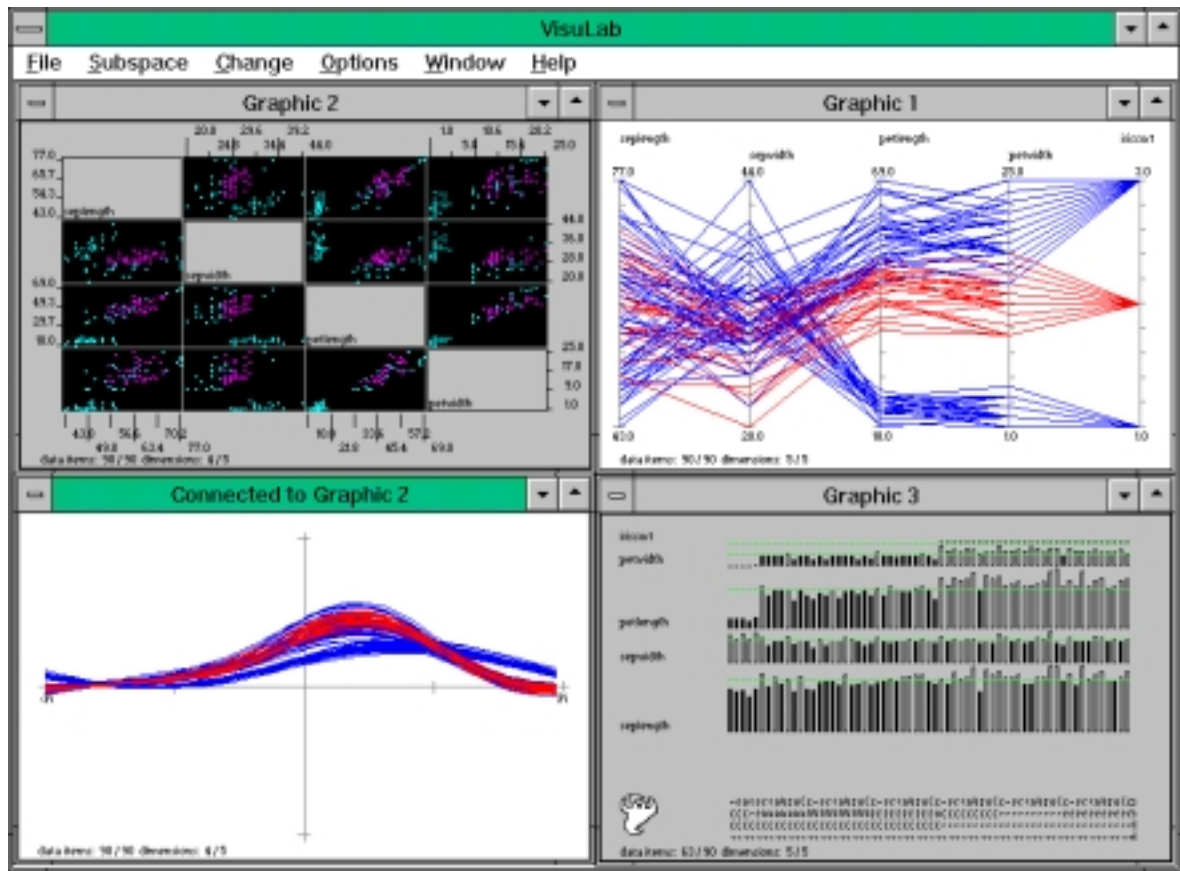


Figure 3.8: VisuLab using different visualisation techniques.

- **Spotfire** is a database exploration system based on interactive information visualisation, dynamic queries, brushing and linking, and other interactive graphics techniques. The target is to minimise the mental effort for information retrieval, i.e. for building queries, recognising and evaluating relations. Starfield displays, node and link diagrams, and cocktail maps are used as visualisation techniques [Ahl96].
- **Visulab** [VIS98]

### 3.4 Summary

A good overview of exiting projects and systems is presented in the Olive (The Online Library for Information Visualisation) [RHS97] homepage. The screenshots presented in the previous section are partly taken from this page and it is recommended to the interested reader.

This chapter presented common functions provided in existing multidimensional visualisation systems. Although the data differs in kind and dimensionality, these functions are found in many existing systems.

The two main possibilities to visualise documents are scatterplots (matrices) with metadata values on the axes, or force based methods. The former was the method of choice in the later described Search Result Explorer. The latter displays the documents in two or three dimensional spaces, where the distance between documents grows according to their dissimilarity.



# Chapter 4

## Java

### 4.1 Introduction

Java is a high-level programming language designed to meet the challenges of application development in the context of heterogeneous, network-wide distributed environments. Paramount among these challenges is the secure delivery of applications which consume the minimum of system resources, can run on any hardware and software platform, and can be extended dynamically. Java incorporates the best features of existing languages. It embodies thirty years of learning about programming languages and tools, software engineering and distributed systems. It focuses on networking and application safety - two very challenging issues not well addressed by other languages.

### 4.2 Java History

The history of the Java language goes back to April 1991, when a group of Sun Microsystem employees began to work on a project to develop advanced software for a wide variety of network devices and embedded systems. The goal was to develop a small, reliable, portable, distributed, real-time operating platform. Java was fulfilling a view that the world would be filled with intelligent devices that would all need a common mechanism for controlling their behaviour. The developers initially attempted to extend the C++ language, but over time the difficulties encountered with C++ grew to the point where the problems could best be addressed by creating an entirely new language platform. Design and architecture decisions drew on experience with a variety of languages such as Eiffel, SmallTalk, Objective C, and Cedar/Mesa. The result is a language platform that has proven ideal for developing secure, distributed, network-based end-user applications in environments ranging from network-embedded devices to the World-Wide-Web and the desktop. Java is unique because it is the first language that can be used for writing general-purpose programs, as well as programs that run within Web clients.

Today popular browsers, such as the Netscape Navigator and Internet Explorer, incorporate Java-based technology and small Java programs, called applets run within these Web clients. General-purpose Java programs which run standalone in the Java Virtual Machine are called Java applications.

### 4.3 The Java Programming Language

The design requirements of Java are driven by the nature of the computing environments in which software must be deployed. Java is a powerful applications-development language, and usually re-

quires a complete book to cover it effectively. In this short chapter, only some basic characteristics of the Java language are described, followed by an overview of tools and possibilities for building graphical user interfaces. A number of books are available for further details [SWM<sup>+</sup>97, Jaw98].

Sun describes Java as being all of the following:

- **Simple:** Java was designed with the intent of keeping the language simple, but, at the same time, powerful enough to perform network computing tasks for the Web. Java can be programmed without extensive programmer training while being attuned to current software practices. To meet the goal of simplicity, the designers of the language kept the number of language constructs as small as possible without compromising power.
- **Architecture-neutral:** Java is designed to support applications that will be deployed into heterogeneous network environments. In such environments, applications must be capable of executing on a variety of hardware architectures. To accommodate this diversity of operating environments, the Java compiler generates bytecodes, an architecture neutral intermediate format designed to transport code efficiently to multiple hardware and software platforms. As shown in Figure 4.1, bytecode can be generated on different operating systems, distributed over networks, and executed in any Java Runtime Environment regardless of where they were compiled.
- **Object-oriented:** The needs of distributed, client-server based systems coincide with the encapsulated, message-passing paradigms of object-based software. To function within increasingly complex, network-based environments, programming systems must adopt object-oriented concepts. Java provides a clean and efficient object-based development platform. The notion of an object in Java is implemented by the class construct. The use of classes is so fundamental to the Java language that it is not possible to write a Java program that does something meaningful without using the class construct.
- **Portable:** The primary benefit of the interpreted byte code approach is that compiled Java language programs are portable to any system on which the Java interpreter and run-time system have been installed. The architecture-neutral aspect is one major step towards being portable. Further, Java eliminates the defect of designating many fundamental data types as “implementation dependent” by defining standard behaviour that will apply to the data types across all platforms. Java specifies the sizes of all its primitive data types and the behaviour of arithmetic on them. Programs are the same on every platform - there are no data type incompatibilities across hardware and software architectures.
- **Distributed:** The architecture-neutral and portable aspects of the Java language make it the ideal development language to meet the challenges of distributing dynamically extensible software across networks. Java provides network capabilities by using a predefined language package called `java.net`. This package contains many classes to simplify network communications between applications running on different computers on a network. Using Java, access of remote or local files can be done with equal ease.
- **High-performance:** Java achieves superior performance by adopting a scheme by which the interpreter can run at full speed without needing to check the run-time environment. The automatic garbage collector runs as a low-priority background thread, ensuring a high probability that memory is available when required, leading to better performance. This strategy leads to high-performance compared to other scripting languages. Using a Just in Time Compiler (JIT), performance can be further improved with the disadvantage that the compiled programs are operating system dependent.

- **Interpreted:** The Java compiler does not produce the machine-language instructions that make up the executable Java program. Instead, the Java compiler produces an intermediate code called byte-code. This code is read by a Java interpreter that executes it by using an internal model of an abstract machine. The Java interpreter can execute Java bytecodes directly on any machine to which the interpreter and run-time system have been ported. In an interpreted platform such as the Java system, the link phase of a program is simple, incremental, and lightweight.
- **Multithreaded:** Java is one of the few languages that provides support for multitasking in the form of multithreading within the language itself. This multithreading capability provides the means to build applications with many concurrent threads of activity, which results in a high degree of interactivity for the end user.
- **Robust:** Robustness in a language means the support for eliminating error-prone constructs both at compile and run time. Java provides extensive compile-time checking, followed by a second level of run-time checking. Java is a strongly typed language, which means that there are well-defined rules on how objects are to be used. Java supports explicit exception handling in the language, which provides the programmer with an additional tool to write robust programs.
- **Dynamic:** While the Java compiler is strict in its compile-time static checking, the language and run-time system are dynamic in their linking stages. Classes are linked only as needed. New code modules can be linked in on demand from a variety of sources, even from sources across a network. The runtime class definitions perform dynamic linking of classes.
- **Secure:** Java is designed to operate in distributed environments, which means that security is of big importance. With security features designed into the language and run-time system, Java applications can not be invaded from outside.

Figure 4.1 shows the progression of a Java source file from creation to execution.

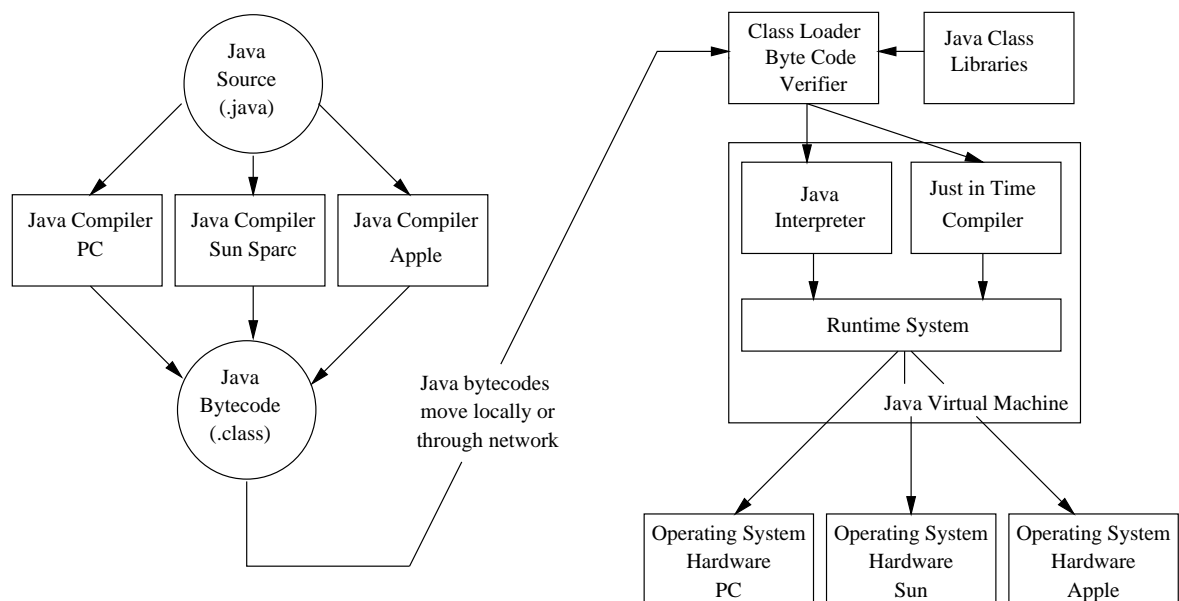


Figure 4.1: Java Program Execution.

## 4.4 The Java Foundation Classes (JFC)

The Java Foundation Classes represent a significant introduction of new functionality to the Java development environment. They are a group of features whose implementation began with JDK <sup>1</sup> 1.1 and is continued in JDK 1.2. One of the most significant parts of the JFC is the Swing component set. The Swing components are all graphical user interface controls that replace most of the platform-native components provided by older versions of the JDK. The Swing components are written in pure Java. They include both Java versions of the older AWT (Abstract Windowing Toolkit) component set (Button, Scrollbar, Label, etc.), plus a rich set of higher-level components (such as TreeView, ListBox, ColourChooser, FileChooser, ProgressBar, etc.) [WC99].

Additional new features first introduced with the JFC are:

**Pluggable Look and Feel:** The Look and Feel of a program consists of the way the program presents itself to the user (its look) and the way the user interacts with it (its feel). The system independence of Swing gives any program that uses Swing components a choice of its Look and Feel, it becomes pluggable. This is done in terms of the model-view-controller (MVC) architecture. MVC is a software architecture that separates the status of an object (the model), the way the object is displayed to the user (the view), and the way that the object's state is updated. By separating these three perspectives, it is possible to define GUI components that are equivalent in terms of information state, but are displayed and respond to the user in different ways. The procedure how to implement the pluggable Look and Feel in a Java program is shown in Section 5.6.2.

**Accessibility API:** These technologies provide non-standard ways of interacting with software applications. They enable Java applications to scale to encompass the physically challenged users. Two of the most important features are Screen Readers and Screen Magnifiers. The Screen Reader creates an off screen representation of the GUI components enabling the information to be provided via a text to speech and/or a Braille terminal. The user can then interact with the GUI through the alternate computer input and output device. The Screen Magnifier enables the user to adjust the magnification of the screen.

**Java 2D API:** The Java 2D API enables developers to incorporate high-quality two dimensional graphics, text, and images in applications and in applets. It enables the development of richer user interfaces and new types of Java applications. The Java 2D API also supports enhanced colour definition and composition, hit detection on arbitrary geometric shapes and text, and a uniform rendering model for printers and display devices. Paths, text, and images are treated uniformly, they can all be rotated, scaled, skewed, and composited using the new *Graphics2D* class [WC99].

**Drag and Drop Support:** Provides the ability to drag and drop within Java applications, between Java applications, and between Java and native platform applications. It is typically used to organise desktops, manage files, open documents, and execute applications.

The programs described later in this thesis make extensive use of the Java Foundation Classes. In particular the user interface code consists completely of Swing classes. Some interesting code samples and implementations which are part either of the File Attributes Explorer or the Search Result Explorer are described in detail in Section 5.6 and Section 8.5.

---

<sup>1</sup>Java Development Kit

## Chapter 5

# File Attribute Explorer

### 5.1 Introduction

The intent of developing this tool was to provide a powerful hierarchy explorer for file systems. From the beginning an open concept which should also fit for other kinds of multivariate data was kept in mind. In other applications problems occurred if the view was too complicated. The goal was to provide a tool that is easy to use, and a display that is easy to understand. The visual complexity and the difficulties in orientation and navigation of displays using more than two spatial dimensions led to a concept of using a two-dimensional plot. Further dimensions are brought to the system using colour, size and shape. Interactivity, flexibility in scaling, and functions for details on demand were implemented.

### 5.2 Design Principles for the File Attribute Explorer

This tool was designed to explore files and their attributes, located in any given directory hierarchy. It should become possible, to analyse and navigate through the whole file system with immediate overview over the contained files. Additional search and filter functions improve this overview and can be used to focus on specific types of data. For example, it is possible to view all images larger than a certain size and older than one year.

Graphical presentations take advantage of the enormous capacity for human visual information processing. In a few tenths of a second, humans can recognise features in megapixel displays, recall related images, and identify anomalies. This should help to reduce users' fear of the flood of information and make browsing fun. The following principles used in many visualisations were the starting point for the File Attribute Explorer design:

- **Operating system independence** has the advantage that a single software distribution can be installed on a wide range of platforms.
- **Iconic, point or circle representation** of files to display as many objects as possible in the given space. The size of the displayed items can be changed interactively, so users can respond to cluttered displays.
- **Flexible scaling of axes** was a design goal from the beginning. The user can change the scale on any axis. The sort order of an axis is changed with a single click on a button.

- **Zooming** to any part of the display is possible to further analyse regions of special interest. If the number of displayed items becomes too high in such a region they are grouped together into a single icon. Such groups can be further explored by zooming in on them.
- **Selection by pointing** (not typing) and obtaining details on demand is a principle which improves usability, and is implemented by making each item active (like a button). When pressed the item shows its attributes textually in a special window.
- **Output can be used as input** to explore contained directories. Every displayed directory can become the next root of visualisation by simply choosing it with the mouse. A history list offers the possibility to navigate back to previous directories.
- **Immediate and continuous display of the file system** by implemented the loading procedure in its own thread and giving the user a feedback.
- **Display as many file attributes as possible** by mapping icon colour and size to selectable file attributes enables users to build their own visualisations with respect to the attributes they are interested in.
- **Searching** is done by making files which match selectable constraints visible by colour coding.
- **Filtering** means temporarily excluding non-matching documents from the view. In this case the removed documents can be redisplayed by turning the filter off.
- **Presenting different views** with the possibility to display the files and their attributes in a traditional table. Much work was done to make the table sortable by any displayed attribute and to synchronise selection, searching, and filtering in both the graphic and the table displays.

Many of these issues were already mentioned in Chapter 3.

### 5.3 File Attribute Explorer Implementation

The program is implemented in Java to achieve platform independence. All source files, java classes, and the used icons are packed into one single Java Archive (jar) file to make it easier to distribute. This file is named *fae.jar* and consists of about forty base classes. The Java concept of inner classes is used heavily in the program, because various user interface events are handled by these classes. The icons used in the interface and as representations for files are also included. This single file and a Java Runtime Environment are the only two things necessary to run the File Attribute Explorer. The program is an implementation of the scatterplot visualisation technique described in Section 3.3.2 and its display concept is quite similar to Envison (see Section 3.3).

Users specify the directory they want to start from and the program displays icons representing the files located in this directory. If the *Flatten Directories* option is selected, all files in all subdirectories are also included in the display. Files and their attributes are also shown in a second tabular display. In this second view, every file occupies one line and actions (e.g selections) are synchronised between both displays.

The file system hierarchy is loaded recursively from top to bottom. To avoid too long load times and too extensive memory usage, a maximum directory depth can be specified. Selected file attributes can be mapped to the x and y axes in either ascending or descending order. Further attributes can be mapped to the size and colour of the individual file and group icons.

Group icons are created if sets of documents are displayed at the same point in space. In this case, the number of collected files is shown in these icons and users can zoom in to explore this part

of the display in detail. The drawing area becomes scrollable and the labels on the axes are also automatically updated as zooming occurs. This is done either by double clicking a group icon or by dragging a rectangle in either the drawing area or the zoom overview window. If only a part of the whole area is visible in the main window the entire display is shown in the zoom overview in the bottom left corner. The currently displayed part is indicated by a black rectangle. To move the visible part without changing its size, the rectangle in the overview window can be moved with the mouse or the scroll bars can be used.

To obtain more information about a particular file, users move the mouse pointer to that icon. Clicking the icon causes the program to show all the file's attributes. If a group of icons is selected, a list of files in the group is displayed. This list can be sorted on any attribute and clicking on a file name shows the attributes of that file.

Once installed, the program can help to solve a variety of common problems arising in large hierarchies:

- Finding the newest and largest file(s) of a particular type in an entire directory tree, which is not that straightforward with other explorers.
- Analysing file history; which files were updated which remained unchanged since a given date.
- Finding file duplicates which are hidden somewhere in the hierarchy because such files will often form a group of two files.
- Marking all files matching certain conditions in size, age, and location, which also can be used to discover lost files.
- Finding preferred locations of differing file types.

These and other tasks are not that simple with other file explorers and can be helpful for both the PC user and the system administrator.

## 5.4 Usage of the File Attribute Explorer

The File Attribute Explorer was developed with the Java Development Kit (JDK) 1.2. It works on every operating system (Unix, Windows95, WinNT, MacOS, ...) where the Java Virtual Machine 1.2 (or higher) is installed. The program is packed into one executable jar file and is run typing the following statement on the command line:

```
java -jar fae.jar
```

## 5.5 User Interface

During the loading and initialising process the welcome window is presented on screen. After these processes the File Attribute Explorer main window appears. Figure 5.1 shows the FAE's main window. The following sections of this chapter describe the usage and effects of the various GUI components. The drawing area located in the bottom right corner is shown and explained in detail in Section 5.5.3.

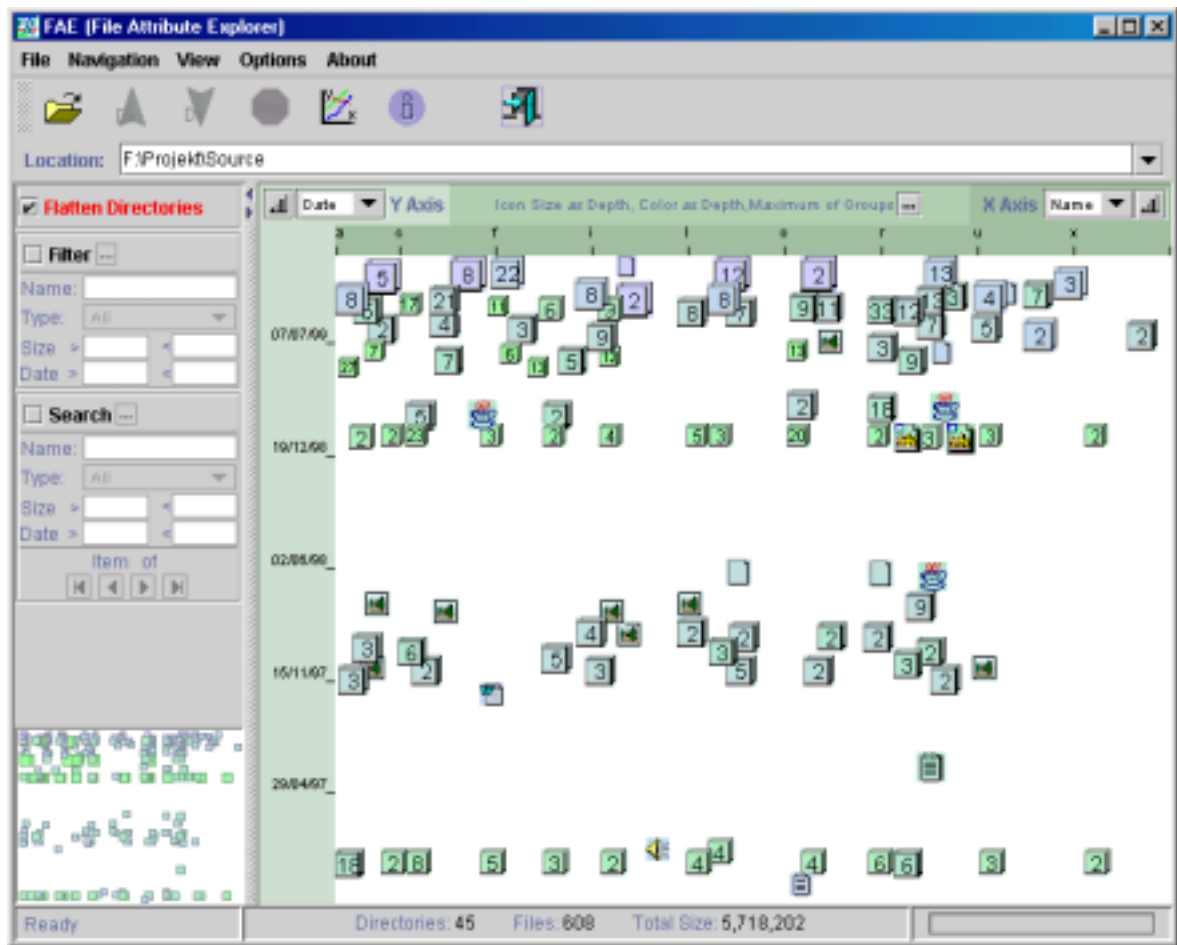


Figure 5.1: File Attribute Explorer Main Window.



### 5.5.1 The Button Toolbar

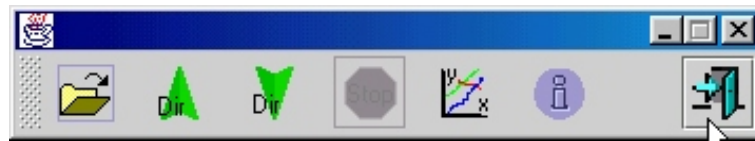


Figure 5.2: The FAE button toolbar.

As a feature of the Java Foundation Class `JToolBar` this toolbar can be hidden (in the View menu) or dragged to any side in the File Attribute Explorer, or it can be put in a separate window. All the actions caused by these buttons are also provided by corresponding menu items. In the View menu the toolbar can be hidden by deselecting the corresponding `CheckBox` for users who prefer working only with menus. The five buttons in the first column of the main window are image buttons and, if they are active, they cause the following actions:

- **Open Directory:** This button brings up a file chooser for navigating the file system, and choosing a directory from a list or entering a directory name. This is implemented with the `JFileChooser` class provided by the Java Swing library. It is a modal dialog which uses an appropriate filter accepting only directories.
- **Back Button:** Goes back to the directory which was previously explored. The back and forward buttons become active, if the user navigates through the directory tree either by double-clicking on directory icons in the drawing area or by simply choosing another directory via the file chooser.
- **Forward Button:** Goes forward to a directory which was previously explored. This button becomes active, if a back operation has been performed. The directories held in the history list are cached and navigating back or forward is faster than loading and analysing a directory for the first time.
- **Options:** Opens a panel to see or change program settings. Users may choose their favourite interface look and feel, date and time formats, and the state of the simultaneous panning option. The new settings are accepted with the *Ok* Button or the old settings remain if the dialog is closed with the *Cancel* Button. The options panel is described in detail in Section 5.5.8.
- **Stop Button:** If user choose a directory to be displayed, a thread which loads attributes of files, located in the current directory, is started. The functionality of the *Stop* Button is to interrupt this process. With this button users can always regain control over the interface, even if the loading process lasts too long.
- **About:** Displays the About File Attribute Explorer window which gives informations about the program and shows the license agreement. This window is closed with the *Quit* Button.
- **Exit:** This button closes all windows and exits the program. All settings changed by users are stored in their private configuration file and are reused as parameters for later program starts.

### 5.5.2 The Location Field

This field is implemented as a `JComboBox` provided by the Swing library. It is a second possibility to directly specify the directory which should be explored next. It also holds a history list of the

five previously explored directories which can be redisplayed by choosing them from the list. These previously explored directories are held in a cache to provide fast redisplay. A third possibility to open subdirectories is to double-click directory icons in the Drawing Area. In both cases a thread which loads the directory is started.

### 5.5.3 The Drawing Area

All functions concerned with the visualisation are implemented in this part of the tool. In this area the files are represented as icons, squares or circles. The positions of these objects depend on the attributes of the corresponding files and the settings of the axes. Size and colour depend on other user selected attribute values. The main parts and tasks of the Drawing Area are:

- **Axis Settings:** The following interesting attribute values can be displayed on each axis:
  - **File Size**
  - **File Date**
  - **File Name**
  - **Directory Depth** the relative depth of the file in the hierarchy.

The buttons in the upper corners determine whether the axes are in ascending or descending order.

- **Axis Labels:** The axis labels dynamically adapt to current range of values when zooming occurs. The labels differ depending on the type of data represented on the axis. Their format also depends on the preferred date, time and number formats. To improve legibility the background of axes is drawn in slightly different colours.
- **Display Settings Panel:** At the top of the Drawing Area a line of text indicates the current display settings. The button to the right of this line brings up the Display Settings Panel. The text in the first line (see Figure 5.3) of the Drawing Area describes the selected mapping settings for size and colour and the group display behaviour. The panel is implemented as a

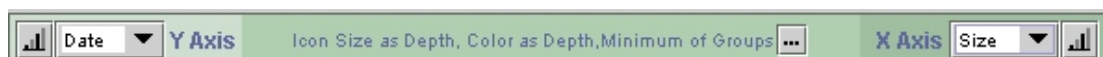


Figure 5.3: Current settings of the Drawing Area.

modal dialog which allows to switch between main window and settings dialog and results in simultaneous visualisation update when changing display settings. The following settings can be changed:

- **Icon Shape:** Users can choose the graphical representation for files in the Drawing Area. The files can be displayed as icons, squares or circles.
- **Icon Size:** Gives users choice which file attribute is mapped to the size of its icon. The numbers are pixel values and the text field components accept only integers as their values.
- **Icon Colour:** Again, the user can choose which file attribute to map to icon colour. The range of attribute values is smoothly interpolated between two colours. The used colours are displayed as two small image buttons. These buttons bring up a JColourChooser dialog which provides a comfortable user interface for selecting a desired colour.

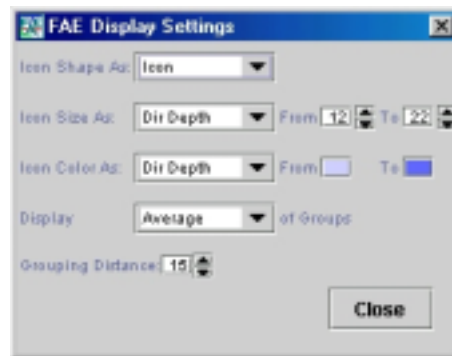


Figure 5.4: The Display Settings Panel.

- **Group Display Behaviour:** A group contains the number of files shown in the middle of the icon. The selected size and colour settings normally depend on attributes of single files. Grouped files have inherently similar values for the attributes scaled on the axes, however the attributes mapped to size or colour may differ. The chosen solution gives the user control over how to aggregate values for all files in a group into a single value for display:
  - \* **Minimum:** The file in the group with the minimum attribute value determines the size and colour of the whole group icon.
  - \* **Medium:** The values for size and colour are calculated as the average value of all contained files. The mediums are calculated during the layout process for all groups and all their attributes.
  - \* **Median:** The file with the median attribute value determines the size and colour of the whole group. This value can be obtained from the group object because the files inside the groups are held sorted by their attributes in the data structure.
  - \* **Maximum:** The file with the maximum attribute value determines the size and colour of the whole group.
- **Grouping Distance:** This integer value is used to decide whether two side by side lying objects are represented as one group or as two items. Choosing a higher number forces the display to show less single file objects and to group more objects. Decreasing this number results in a display of many single file objects which are possibly overlapped by other icons. This setting is unique in this dialog in terms that changing it forces the program to do a new layout process (see 5.5.3) which lasts noticeably longer than changing other settings.
- **Mouse functionality in the Drawing Area:** In the Drawing Area the graphic primitives are placed. Following the concept, this area is filled with powerful functions to gain overview, zoom in and out, navigate and get details-on-demand. How to perform these tasks using the mouse is described next. The mouse events differ from the position where they occur:
  - **Mouse events in the blank area** are used to control the display. Two mouse events are possible:
    - \* **Right Mouse Click:** A context menu which controls the current view is brought up. It offers possibilities to control the zoom status, the history list and the used graphic representation.

- \* **Mouse Drag:** A rectangle from the position where the mouse was pressed is drawn. Releasing the mouse zooms into the rectangle.
- **Mouse events on icons:** These three events can be used to display details-on-demand. To provide this functionality each graphic representation has its own listener which reacts to the following events:
  - \* **Enter:** If the mouse pointer enters an icon, it becomes an image button and the corresponding filename is shown in the bottom line. If the mouse pointer leaves the icon, the button functionality is removed, the icon becomes a normal image again.
  - \* **Mouse Click:** Clicking on an icon selects the corresponding file. The attributes of the file are shown in the Current Object Window and the filename is highlighted in the Table of Files.
  - \* **Double Click:** Double clicking is implemented for graphic primitives that represent directories or group objects.
    - **Directory:** This directory is explored now. This means that the user moves down one (or more if the hierarchy is flattened) step(s) in the directory tree. The directory which was explored until this happened, is put in the history list and can be accessed via the *Back* button.
    - **Group:** The group is zoomed into. The new display range is calculated from the attribute range of the group.
- **Zoom Control:** As already mentioned the ability to zoom guides the user from overview to detail and vice versa. It is an important feature in many visualisations and it is in particular in the File Attribute Explorer. Table 5.1 summarises the possible zooming operations and where they are offered. The *Zoom-Out* button withdraws the last zoom-in action and the *Zoom-Reset* button returns to initial settings without zoom.

	<b>Zoom in</b>	<b>Zoom out</b>	<b>Zoom reset</b>
<b>Drawing Area</b>	Double click on group. Dragging a rectangle	Popup menu via right mouse button.	Popup menu via right mouse button.
<b>Overview Window</b>	Dragging a rectangle.	Popup menu via right mouse button.	Popup menu via right mouse button.

Table 5.1: Possible zooming actions.

- **Positioning of Icons:** The position of the icons depend on the attributes of the corresponding files and the axes settings. A suitable algorithm which splits the items into single file objects and group objects is used. The straightforward method would calculate all  $\binom{n}{2}$  distances. These distances could then be tested whether they are smaller than the given grouping distance to decide if the involved files are drawn as single objects or form groups. This process would result in a runtime of  $O(n^2)$ .  
  
The similarity between the layout task and the “Line Segment Intersection Problem”, well known in the field of computational geometry led to the used strategy which is described and analysed below [dBvKOS97]. The algorithm first sorts the items according to their attribute values on the x-axis. Then a stripe is “swept” along the x-axis. The width  $b$  of this stripe can be changed in the Settings Dialog, where it is named *Grouping Distance*. If the distance between items is smaller than  $b$  they are grouped together into one display item. This type of

algorithm could be called a “plane sweep algorithm” and the stripe  $b$  could be called the “sweep stripe”. Figure 5.5 illustrates the algorithm. The squares  $f_x$  represent files. The previous

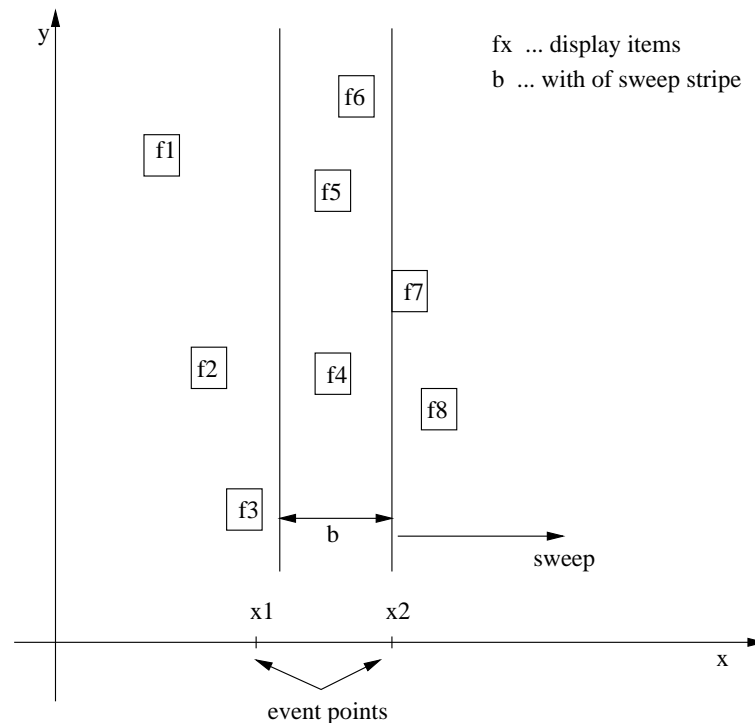


Figure 5.5: Placement Algorithm using a Sweep Stripe.

calculated array of the sorted x-values are called *event points*. The moments, when the sweep stripe reaches or leaves an event point are the only moments when the algorithm actually does something. The items currently lying in the sweep stripe are stored in a balanced binary search tree sorted by their y-values. At an event point two things may occur:

- **A file leaves the stripe:** This means that the difference between the file’s x-coordinate and the right border of the stripe is more than  $b$ . This item (a single file or a group) can be stored in the vector of display objects and is deleted from the binary search-tree. Figure 5.6 shows the sweep stripe leaving event point  $x_1$  and removing the file  $f_3$  from the tree. This file is added to the vector that holds the items which will finally be displayed in the Drawing Area. To keep the tree balanced it is sometimes necessary to force a rotation operation.
- **An item hits the sweep stripe on its right border:** The new item is inserted into the tree(see Figure 5.7). Again the balance must be kept and the distances between the inserted file and its neighbours is checked. If one of them is smaller than  $b$  than these two objects form a group object and the tree is updated again (see Figure 5.8).

Insertion, deletion, and rotation take  $O(\log(n))$  time. Every display item is inserted once into the tree. The number of deletions from the tree also equals the number of files, because if a file leaves the stripe, a deletion will be forced and if files are grouped this is the case, too. Insertion and deletion can potentially destroy the balance of the search tree. In these cases rotation of the tree has to be done. If the balance condition is hurt after one insertion or deletion operation it

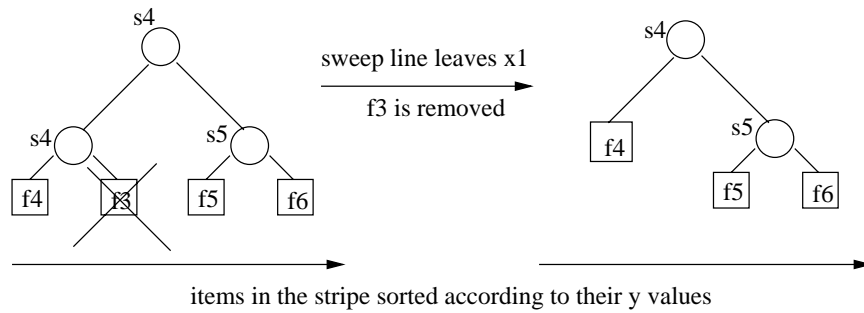


Figure 5.6: Removing a file from the tree.

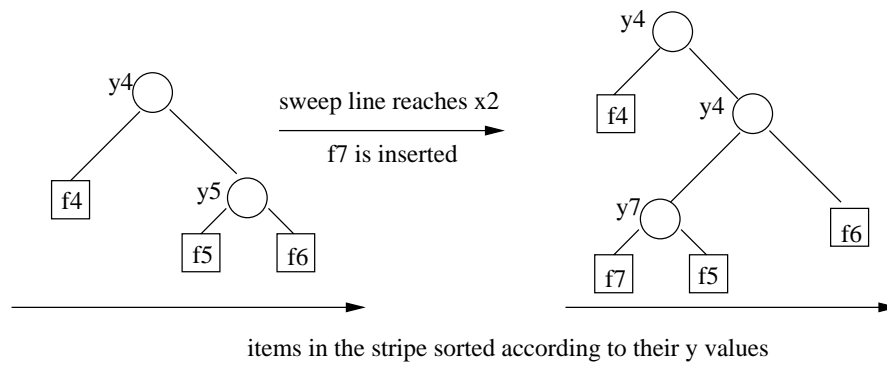


Figure 5.7: Adding a new file to the tree.

balanced binary tree after left rotation

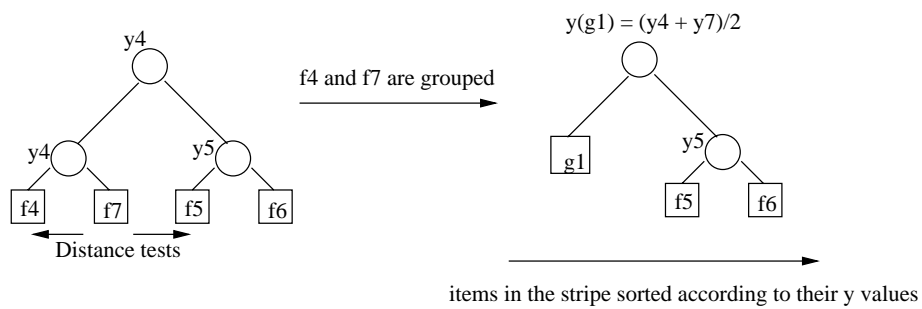


Figure 5.8: Rotation and grouping of items.

Operation	Time for one Operation	Number of Operations	Worst Case Overall Time
<b>Insertion</b>	$O(\log(n))$	$n$	$n * O(\log(n))$
<b>Deletion</b>	$O(\log(n))$	$n$	$n * O(\log(n))$
<b>Grouping</b>	$O(\log(n))$	$n$	$n * O(\log(n))$
<b>Rotation</b>	$O(\log(n))$	$2 * n$	$2n * O(\log(n))$

Table 5.2: Runtime of placement operations.

can be re-established by at most two rotations. As shown in Table 5.2 the complexity for one operation is always  $O(\log(n))$  in time and every operation occurs  $n$  times (except there can be  $2 * n$  rotations which does not influence the upper border of the runtime). Thus the layout algorithm calculates the display objects in  $O(n * \log(n))$  time which is the best reachable runtime.

- **Realised File Types:** At the moment the program recognises the following file types by analysing the suffix of filenames:
  - **Text Document:** The document is a text document.
  - **Picture:** The document is an image.
  - **Audio File:** The document is a sound file.
  - **Postscript:** The document is a postscript file.
  - **Unknown Format:** The document has no known format.
  - **Word File:** The document is a Microsoft Word file.
  - **Excel File:** The document is a Microsoft Excel file.
  - **Java Source:** The document is a Java file.
  - **Latex File:** The document is a Latex file.
  - **Html File:** The document is a Html file.
  - **Group** This icon represents a group of files. The distance between the icons is too small to show each icon. The number of files contained in this object is displayed in the middle of the group icon. A double click on this icon causes a zoom-in action in order to show more items as single files, and make their attributes accessible.

#### 5.5.4 The Status Bar

This part is divided into three areas, which show important information about ongoing tasks and statistical characteristics of the explored directory.

- **Left Panel:** The text gives status information about what is going on during calculations or shows the name of the object beneath the mouse in the Drawing Area.
- **Middle Panel:** The text lists the characteristics of the currently displayed directory. The three properties listed are the number of displayed files, directories and the total size of the viewed hierarchy. In the case a filter is active, only the items which pass this filter are taken into account.

- **Right Panel:** The progress bar shows the progress of loading, calculating, and drawing operations. If a new directory is loaded, the bar is informed about the number of files contained in this directory, having one as their depth relative to the explored directory. If the *Flatten Directory* option is set, this number increases with every directory that is visited during the analysis. This process causes the flickering of the bar. Thus it is no quantitative indicator of how long the analysis will last, but it shows that the program is still working.

### Zoom Overview

This area in the bottom left corner (above the Status Bar) is a small view of the Drawing Area. If a zooming action is performed, this area shows the currently viewed part of the display area as a black rectangle. This rectangle can be moved by dragging it to the desired new position with the mouse. A mouse click in the overview centres the viewing rectangle in the mouse position. If the *Simultaneous Panning* option is activated, this move process is simultaneously shown in the Drawing Area, which gives a good overview but it may be very slow. This window also offers a popup menu to change the current zoom status. The intent of this window is to show the data as a whole and to inform about the performed zooming actions. It does not provide the other functionality of the Drawing Area.

### Filter and Search

These functions offer possibilities to concentrate on specific file types, and to narrow size or date attributes of the displayed files. Filenames are matched using the *FNMatch* class. This class is based on the GNU-Library *fnmatch* routine which provides an intuitive “file name matcher” rather than using regular expression matching. This class can be configured to either respect or ignore case. It also provides possibilities to match whole worlds only. The user can specify how the actual search or filter operation should work via a popup menu. The buttons in the search and filter panels bring up this popup menu. To clearly indicate whether search and filter are active, the *Search* and *Filter* labels are coloured red when active.

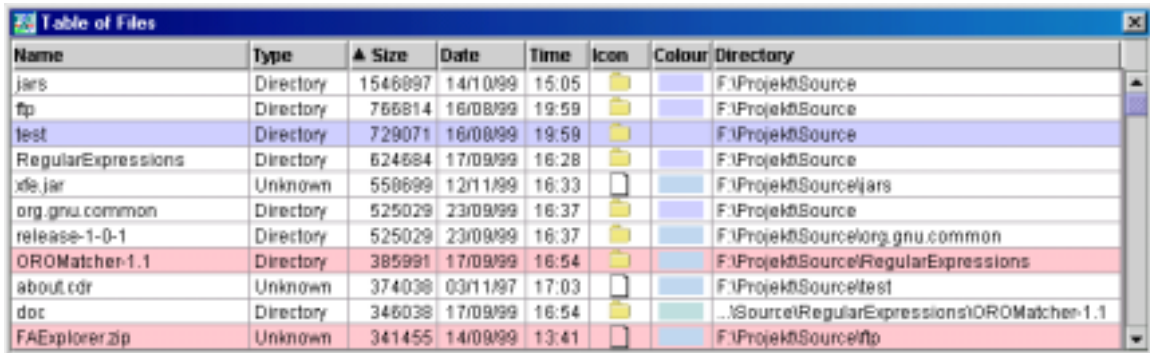
The results matching a search are outlined with a red border and can be viewed in succession using four navigation buttons. The currently viewed icon is outlined with a blue border, and a blue arrow pointing to it. All file attributes are displayed in the separate File Attribute Window shown in Figure 5.10. If a filter is applied, only the files which pass this filter are displayed in the visualisation and listed in the Table of Files, but the original display ranges remain in force.

#### 5.5.5 Table of Files

This table (see Figure 5.9) displays all files located in the current directory and shown in the Drawing Area. The currently selected file is highlighted. This selection can be changed by clicking on another file with the left mouse button. If multiple selections occur, a group object has been selected in the Drawing Area. In this case, all files contained in this object become highlighted. If a line in the table is selected, the corresponding item in the graphic view is drawn with a blue border and a blue arrow pointing to it. The selection process is the same in the Drawing Area and in the Table of Files. Due to this implementation a double click on a line holding a directory forces the program to analyse this selected directory and a right mouse click shows the File Attribute Window of this file.

If a search was applied the results in the table are given a red background and in the Drawing Area the found items are outlined with red borders. If a filter is applied, files which do not pass this filter are removed from the table.





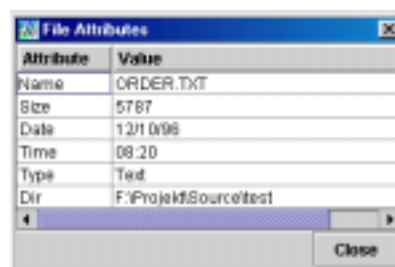
Name	Type	▲ Size	Date	Time	Icon	Colour	Directory
jars	Directory	1546897	14/10/99	15:05	📁	🔵	F:\Projekt\Source
ftp	Directory	766814	16/08/99	19:59	📁	🔵	F:\Projekt\Source
test	Directory	729071	16/08/99	19:59	📁	🔵	F:\Projekt\Source
RegularExpressions	Directory	624684	17/09/99	16:28	📁	🔵	F:\Projekt\Source
xde.jar	Unknown	550699	12/11/99	16:33	📄	🔵	F:\Projekt\Source\jars
org.gnu.common	Directory	525029	23/09/99	16:37	📁	🔵	F:\Projekt\Source
release-1-0-1	Directory	525029	23/09/99	16:37	📁	🔵	F:\Projekt\Source\org.gnu.common
OROMatcher-1.1	Directory	385991	17/09/99	16:54	📁	🔵	F:\Projekt\Source\RegularExpressions
about.cdr	Unknown	374038	03/11/97	17:03	📄	🔵	F:\Projekt\Source\test
doc	Directory	346038	17/09/99	16:54	📁	🔵	..\Source\RegularExpressions\OROMatcher-1.1
FAExplorer.zip	Unknown	341455	14/09/99	13:41	📄	🔵	F:\Projekt\Source\ftp

Figure 5.9: Table of Files.

The files in the table can be sorted based on the clicked column. The small arrow in one of the column headers shows the current sort column and sort direction. It can be changed by clicking another column header which will cause this column to become the new sort column. If this column header is clicked again, the sort order changes, and the arrow changes its direction. The same functionality is provided by the popup menu which is displayed when users right-click a column header.

### 5.5.6 File Attribute Window

This window holds a table where all attributes of one file are displayed. If users right-mouse-click an image button representing a single file or a line in the Table of Files the corresponding file becomes selected. The following attributes of the selected file are displayed in the File Attributes Window



Attribute	Value
Name	ORDER.TXT
Size	5787
Date	12/11/99
Time	08:20
Type	Text
Dir	F:\Projekt\Source\test

Figure 5.10: File Attributes Window.

(shown in Figure 5.10):

- **Name:** The name of the file or directory.
- **Size:** The size of the file, respectively the size of all contained files if the item represents a directory.
- **Date:** The last modification time of the file or directory in the user selected format.
- **Type:** The type of the file if the program is able to determine it from the suffix of the file name.

- **Directory Depth:** The depth in the hierarchy relative to the explored directory.
- **Parent:** The directory where the file or directory is located.

### 5.5.7 Group of Files Window

If the selected icon represents a group object, the file names of the contained objects are displayed in the Group of Files Window (see Figure 5.11). A right mouse click on the table header brings up a

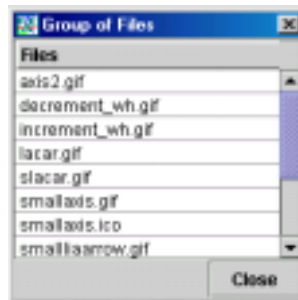


Figure 5.11: Group of Files Window.

popup menu which allows to select one attribute to become the new sorting attribute. The ordering of files can also be changed by choosing either ascending or descending in this menu. Clicking on one file name causes this table to get two new columns which now display all attributes of this selected file.

### 5.5.8 The Options Panel

This panel (see Figure 5.12) gives the user the possibility to customise the program's look and feel, date and time format, and graphical navigation behaviour.

The three possible settings for the look and feel are:

- **Java** also called cross-platform look and feel. This selection provides good looking components on all platforms.
- **Windows:** The program's appearance is looking like any standard windows application. Due to restrictions this look and feel is not allowed on other operating systems than Windows.
- **Motif:** The appearance and behaviour of the user interface is like on Unix and Linux systems.

A detailed description of functions and procedures which implement the pluggable look and feel and make it changeable during program execution is given in 5.6.2. Java also provides powerful possibilities to format Date and Time values. These features are used to let users specify the format they are familiar with. How to use the *java.text* package is described in 5.6.

The *Simultaneous Panning* option has its effects only if the user has zoomed into a particular region of the Drawing Area. In this case a black rectangle in the Zoom Overview Window shows this region and its position relative to the overall display borders. This rectangle can be moved with the mouse. If the option is selected the update in the Drawing Area is done simultaneously. This gives the impression of "flying over the surface", but it may be intolerably slow on some machines.



Figure 5.12: File Attribute Explorer Options Panel.

### 5.5.9 The About Window

This window (Figure 5.13) shows the current project status. The author, the supervisor, and the license agreement of the IICM are also contained.

### 5.5.10 The Configuration File

Any program which wants to offer usability and comfort in use has to be configurable. In the File Attribute Explorer this mechanism is implemented via a property object. On start up this object loads the settings from a file named *FAEsettings.par*. This file is stored in users home directories to ensure that this personal settings file can not be overwritten by other users. If users run the File Attribute Explorer for their first time, normally no configuration file will exist in their home directory. This exception is caught, the program uses defaults and creates the file to hold the settings for the future.

During program execution, if settings like the used display shape, the look and feel, the chosen date and time formats, mapped colours, window positions or sizes are changed, they are stored and remembered in the property object and its configuration file. The file itself stores the date it was accessed and, in the present version of the File Attribute Explorer, about fifty key value pairs, each representing one setting.

## 5.6 Selected Implementation Details

### 5.6.1 Formatting Date and Time

Java measures date and time values in milliseconds since 1.1.1970. To display date or time objects they have to be converted into strings that are in the proper format. First, the format should conform to the conventions of the end user's locale. For example, Austrians recognise 20.4.98 as a valid date, but Americans expect that same date to appear as 4/20/98. Second, the format should include the



Figure 5.13: File Attribute Explorer About Window.

necessary information. This section explains how to format dates and times in various ways and in a locale-sensitive manner. Following these techniques the File Attribute Explorer displays dates and times in the appropriate format chosen by users in the Options Panel. Formatting dates and times depending on a selected locale with the *Java.text DateFormat* class is a two-step process. First, an instance of the *DateFormat* class is created with the *getDateInstance* method. The formatted date string is then obtained by invoking the *format* method. The following example is taken from the File Attribute Explorer class *Rule.java* which does the labeling on the axes in the Drawing Area. If the file date is drawn on one of the axis, this class puts the date in the chosen format on the axis with the *drawString* method.

```
import java.text.*;
...
DateFormat dateFormatter;
DateFormat timeFormatter;
...
dateFormatter = DateFormat.getDateInstance(DateFormat.SHORT, LocalDateString);
timeFormatter = DateFormat.getTimeInstance(DateFormat.SHORT, LocalDateString);
...
text = timeFormatter.format(new Date((long)currentValue*65536));
text = dateFormatter.format(new Date((long)currentValue*65536));
g.drawString(text, i+pixelGap+4+timelength-fontMetrics.stringWidth(text), 8);
```

The preceding code example specified the SHORT formatting style. This style is just one of the predefined formatting styles that the *DateFormat* class provides. In Table 5.3 the different date styles depending on format and locale settings are displayed. For time values and even for numbers the process is analogous. The possible time formats are listed in Table 5.4.

Style	US Locale	British Locale	German Locale
DEFAULT	10-Apr-98	10-Apr-98	10 Apr.98
SHORT	4/10/98	10/4/98	10.04.98
MEDIUM	10-Apr-98	10-Apr-98	10 Apr.98
LONG	April 10, 1998	April 10, 1998	10 April 1998
FULL	Friday, April 10, 1998	Friday, April 10, 1998	Freitag, 10 April 1998

Table 5.3: Date formats using different locales.

Style	US Locale	British Locale	German Locale
DEFAULT	3:58:45 PM	3:58:45 PM	15:58:45
SHORT	3:58 PM	3:58 PM	15:58
MEDIUM	3:58:45 PM	3:58:45 PM	15:58:45
LONG	3:58:45 PM PDT	3:58:45 PM PDT	15:58:45 GMT+02:00
FULL	3:58:45 o'clock PM PDT	3:58:45 o'clock PM PDT	15.58 Uhr GMT+02:00

Table 5.4: Time formats using different locales.

## 5.6.2 Changing the Look and Feel

In Java the Swing user interface manager must figure out which look and feel to use. In the File Attribute Explorer users specify their preferred look and feel in the Options Panel. To specify it, the `UIManager.setLookAndFeel` method is invoked with the desired look and feel as parameter. The following code is part of the `FAExplorer` class and shows how the program sets its look and feel at program start or when the user selects a new look and feel.

```

...
static String winString    = "com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
static String javaString   = "javax.swing.plaf.metal.MetalLookAndFeel";
static String motifString  = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
...
public void setLookAndFeel()
{
    String command;
    switch(selectedLookAndFeel)
    {
        case 0 : command = javaString;break;
        case 1 : command = winString;break;
        default: command = motifString;break;
    }
    try
    {
        UIManager.setLookAndFeel(command);
    }
    catch (Exception ex) { }
    if (!init)
    {
        SwingUtilities.updateComponentTreeUI(window);
        if (filesTable!=null) SwingUtilities.updateComponentTreeUI(filesTable);
        DrawArea.changeLookAndFeel();
    }
}

```

To be able to change the look and feel during program usage, again the *setLookAndFeel* method with the new selected parameter is called. Finally this change is reported to the UI-Components by performing the *updateComponentTreeUI* method with these components as parameter.

## 5.7 Future Work

A visualisation tool always can be improved in many ways. In File Attribute Explorer problems can occur when too many files have exactly the same attributes. During tests this occurred sometimes when “program directories” were analysed, because all contained files had the installation time as their last modification time attribute. Users of course want to zoom into these groups wondering why they do not get the contained files spread over the whole display. To overcome this problem, these files were laid out side by side if zooming did not help. The results were not really satisfying because the labels on the rules were no longer correct, and a better solution should be found.

For the FAE to become a real alternative to existing file explorers two main shortcomings have to be removed. First typical not implemented authoring functions like delete, cut, copy, paste, move, etc. should be added. All these functions change the current display and have to be handled with appropriate events.

A Preview or Open capability would improve functionality and usability, however it seems difficult to implement. The two problems that have to be solved are:

- **Determine the file type:** It seems impossible to implement a file type recognition algorithm which works on all platforms. Existing approaches for example try to match file type suffixes against types without guaranty that a file ending with “.gif” is an image file. A so called “magic number test” tries to find informations about the file type in the file header, but many files without such file header information exist.
- **Determine the application:** This issue needs to write operating system specific code. Knowing the file type would be enough to raise the standard application on Microsoft Windows systems (under the condition, that an application is registered as standard for this type). However this is not possible using other operating systems. To overcome this problem a configurable application registration could be written.

## Chapter 6

# Search Engines

### 6.1 Introduction

Chapter 3 centred on what multidimensional visualisation means and why it is interesting to provide graphical views as an alternative to textual output of information. This chapter pertains to the process of searching for documents and resources on the World Wide Web, including why and how result sets are visualised, and how the extraction of displayable information from metadata, content, and structure is done. Many existing search engines do this extraction in one of many possible ways and use it to calculate estimated relevances. The found resources are then displayed as a linear list sorted by their estimated relevance. Unfortunately, much of the information about the result set which is collected from search engines is not presented to users.

Documents are important sources of information. Document metadata consists of elements such as author, publisher, and date of publication (see Chapter 2). Keywords comprise an intermediate form of document data. They can be viewed as both content and metadata. Metadata forms the primary type of information in library systems. The World Wide Web contains large volumes of documents which are poorly characterised in terms of both metadata and content descriptions. Automated methods for indexing documents have become an important research issue since there are not enough human resources available to index the documents that are published either in paper or in electronic formats. The methods for indexing documents are normally lexically-based but this is not the only kind of information available in the native full-text. Questions in this context are, how to prepare adequate document representations or surrogates, and how to integrate metadata and derived indexes. Another part of this chapter focuses on how to request, and how to retrieve information on certain documents.

Unfortunately, it is not an aim of existing Web search tools to standardise their query languages or their format of returned results. As a contrast, the Z39.50 protocol is described. This standard provides a protocol for information retrieval, which is independent of database and computer environment. The use of this format could improve and speed up the process of searching for information on the World Wide Web and would allow visualisation applications to build on search result sets delivered from arbitrary search engines.

### 6.2 Types of Search Tools

A variety of different services are currently offered on the Internet. They can be subdivided into the following categories which differ in how they gather the information and whether they are topic-specific or not.

**Search Engines** A search engine is a database system designed to index Internet addresses (URLs, usenet, ftp, image locations etc). The typical search engine contains a special program often called a spider (also sometimes called a “bot” or “crawler”), the spider accepts a URL, it then goes to that web site and retrieves a copy of the file found there. Sometime later the search engine processes that copy of the file, distilling it down to the essential data and storing it in a data base. The richness of stored attributes and the speed of searching its database is what determines the quality of the search engine. In short, given a URL, an automated process occurs which results in including Web sites into the search engines database also called its index. Much effort is spent on improving data structures, information extraction, and page ranking by the different search engines. Problems still remain with the exponential growth, dynamic resources and ranking pages. (See Section 6.3).

**Meta-Search Engines:** A meta search engine is a search tool that does not create its own database of information, but instead searches those of other engines. “Metacrawler”, for instance, searches the databases of each of the following engines: Lycos, WebCrawler, Excite, AltaVista, and Yahoo. Using multiple databases will mean that the search results are more comprehensive, but slower to obtain.

**Subject Directories:** Subject directories organise Internet sites by subject, allowing users to choose a subject of interest and then browse the list of resources in that category. Users conduct their searches by selecting a series of progressively more narrow search terms from a number of lists of descriptors provided in the directory. In this fashion, users “tunnel” their way through progressively more specific layers of descriptors until they reach a list of resources which meet all of the descriptors they had chosen. A subject directory will not have links to every piece of information on the Internet, since they are built by humans (rather than by computer programs), and are much smaller than search engine databases. Moreover, every directory is different and their value will depend on how widely the company searches for information, their method of categorising the resources, how well information is kept current, etc.

**Subject Guides:** Subject Guides are Web pages of collections of hypertext links on a subject. Thus they are a specialisation of the above described Subject Directories. and they are maintained by “expert” subject specialists, agencies, associations and hobbyists who are familiar with the topic, which guarantees high quality. A disadvantage for users who just want to discover information on a certain topic is, that they have to find a Subject Guide on this topic first. To overcome this problem so-called “Guides to Guides” are available.

Several Internet sites look at performance, index size and effectiveness of common search tools [Ost99, Bar98]. These sites take also an in-depth look at currently used crawling and ranking methods and compare them. All of the search tools typically produce textual organised output. Documents which match a certain query are displayed, ranked by relevance, starting with the document with the highest estimated relevance.

## 6.3 Relevance Calculation

One of the keys to offer a popular search engine is to calculate appropriate relevances. This comparison between search terms and document content is quite difficult. Often this is done by just counting occurrences of search terms in the content but there is no guarantee that a simple occurrence of a term really makes the document relevant. So they also check if the terms appear near the top of a web page, such as in the headline or in the first few paragraphs of text. They assume that any page



<b>Search Tool Type</b>	<b>Characteristics</b>	<b>Examples</b>
<b>Search Engines</b>	Full-text indexing of Web pages. Search by keyword with exact match compiled by robot programs. No browsing, no subject categories, minimal human oversight.	Alta Vista, Northern Light, Fast Search, Infoseek, Hotbot, Lycos, Excite, Google.
<b>Meta Search Engines</b>	Search many individual search engines. Results compiled into convenient format.	Metacrawler, Inference Find, Dogpile, Metafind, Savvy Search.
<b>Subject Directories</b>	Hand-selected sites picked by editors more or less carefully. Organized in hierarchical categories. Annotated with descriptions. Search only in categories and descriptions.	Librarians' Index, Infomine, Yahoo!, Galaxy Lycos's A2Z, Looksmart.
<b>Subject Guides</b>	Collections of hypertext links on a subject. Compiled by "expert" subject specialists, agencies, associations, and hobbyists.	Argus Clearinghouse, WWW Virtual Library.

Table 6.1: Internet Search Tools Overview.

relevant to the topic will mention those words right from the beginning. This method is often called location/frequency method. Many major search engines follow it to some degree but they differ in some details. To begin with, some search engines index more web pages than others. Some search engines also index web pages more often than others. The result is that no search engine has the exact same collection of web pages to search through.

Other search engines use link popularity to determine the estimated relevance. These systems make the assumption that pages which have a lot of links pointing at them have higher relevance than others because documents with many links to them are probably well-regarded on the Internet. A major disadvantage of this attempt is the fact, that new documents (although they are potentially relevant) do not get high ranks because they are not referenced often when they are new.

Trustworthy metadata could help to improve relevance calculation, but in current search engines they are often left out of consideration which has its reason in the attempt of users who try to push their documents by listing much, often not correct metadata. All these versions lack in some way, they can not guarantee up-to-date data and they are often deceived by advertisers who try to increase their hits.

"Meta tags are what many web designers mistakenly assume are the secret to propelling their web pages to the top of the rankings. HotBot and Infoseek do give a slight boost to pages with keywords in their meta tags. But Lycos doesn't read them at all. Search engines may also penalise pages or exclude them from the index, if they detect search engine spamming. An example is when a word is repeated hundreds of times on a page, to increase the frequency and propel the page higher in the listings. Search engines watch for common spamming methods in a variety of ways, not the least by following up on complaints" [Ost99].

Other approaches try to combine the robot generated index with associated directories. These systems are called "hybrid" search engines and may give a higher relevancy to documents occurring in both the index and the directory.

## 6.4 Intelligent Agents

In connection with searching on the World Wide Web this terminus is often used for tools that try to provide information about structure, environment and content of resources. For example, Alexa [KG99] is a tool which suggests similar documents for each visited resource. Many other systems use link popularity to calculate estimated relevances. Many of these agents also try to learn from user habits and try to suggest documents that are similar to documents marked as interesting in previous sessions.

## 6.5 The Z39.50 Protocol

The importance of having inter-operable information retrieval networks, to find and exchange quality information electronically, has become paramount. To facilitate information retrieval across the diverse collections of data resources now available, a non-proprietary standards-based communications protocol for information retrieval, which is independent of database and computer environment, is essential. Z39.50 is such a standard.

“ANSI/NISO Z39.50-1995 (ISO 23950), is one of a set of standards produced to facilitate the interconnection of computer systems” [ANS95]. The standard specifies formats and procedures governing the exchange of messages between a client and server, enabling the user to search remote databases, identify records which meet specified criteria, and to retrieve some or all of the identified records and is concerned, in particular, with the search and retrieval of information in databases. One of the major advantages of using Z39.50 is that it enables uniform access to a large number of diverse and heterogenous information sources.

Z39.50 recognises that information retrieval consists of two primary components:

- **Selection** of information based upon some criteria.
- **Retrieval** of that information.

The standard provides a common language for both activities. Z39.50 standardises the manner in which the client and the server communicate and inter-operate even when there are differences between computer systems, search engines, and databases.

Inter-operability is achieved through standardisation of

- **Codifying mechanics:** a standard way of encoding the data for shipment along the wire.
- **Content semantics:** a standard data model with shared semantic knowledge for specific communities to allow inter-operable searching and retrieval within each of these domains.

### 6.5.1 Codifying Mechanics

Z39.50 is a wire protocol that simply specifies what gets sent across the wire. Z39.50 defines a network protocol within the application layer of the OSI<sup>1</sup> model and requires a reliable transport service such as TCP<sup>2</sup>. The standard specifies formats and procedures governing the exchange of messages between a client and server. The messages sent between origin and target are specified

---

<sup>1</sup>Open Systems Interconnection

<sup>2</sup>Transfer Control Protocol

in Abstract Syntax Notation One (ASN.1) [Sul99]. The Basic Encoding Rules are used to serialise the ASN.1 structures. The protocol then defines several query languages for specifying search and various record syntaxes that can be used for transferring records from the server to the client.

### 6.5.2 Content Semantics

Coupled with Z39.50 standardised message coding is the concept of shared semantic knowledge. Various classes or domains of information content have been established by community consensus to provide a shared understanding of structure and attributes within that domain. This enables uniform access to heterogeneous information sources within a domain. The basic architectural model of Z39.50 revolves around this concept of content semantics. The server presents a record-based abstract view of its database within a semantic domain, i. e. a virtual database representation containing records with the logical structure of the underlying database being hidden. Associated with each virtual database are a set of access points (or query attributes) that can be used for searching and set of retrieval points (or schema elements) for presenting the data back to the user.

### 6.5.3 Future Use

Users, consumers, or providers need tools to keep up with the explosive growth of networked information. It would be essential to use a common protocol like Z39.50 as an open standard for information retrieval. Based on a client/server architecture and operating over the Internet, the Z39.50 protocol is supporting an increasing number of applications - fulfilling the searching demands of the emerging information age. However the protocol is not really widespread yet, and supporting it for a wider distribution is unfortunately no aim of current available search tools.

## 6.6 Result Set Visualisation

As shown in this chapter, resources on the World Wide Web can be treated as multidimensional data. Thus they become an interesting data source for visualisation. The principles shown in Chapter 3 are already used in many different systems. The mapping from attributes to graphic primitives used in the visualisations becomes more and more sophisticated. Such graphic primitives are often used in several systems to show similarities between documents and keywords. The two presented techniques are reused in popular document visualisation systems.

### 6.6.1 Tile-Bars

Tile-Bars are graphical primitives for users of information access systems, that show the relationship between query terms and the documents which contain them. Tile-Bars simultaneously and compactly indicate relative document length and query term overlap, frequency and distribution. A Tile-Bar shows where a keyword occurs in a document, and maps the number of occurrences to colour saturation. The patterns in a column of Tile-Bars are meant to help users make fast judgements about the potential relevance of the retrieved documents. An unexpected benefit of the interface is that because it requires users to specify their queries as a list of topics, better rank orderings can be obtained than with standard information retrieval ranking mechanisms. Figure 6.1 shows the Tile-Bars of five documents matching the keywords “Osteoporosis”, “Prevention”, and “Research” [HP96].






Query: Osteoporosis, Prevention, Research	Documents
	FR88513-0157
	AP: Older Athletes Run For Science
	AP: Groups Seek \$1 Billion a Year for Aging Research
	SJMN: Women's Health Legislation Proposal
	FR: Committee Meetings

Figure 6.1: TileBar visualisation of 5 documents.

### 6.6.2 Arrow-Spheres

This approach visualises each documents as a sphere with arrows emanating from within. Each arrow corresponds to a particular keyword, and its length corresponds to the number of occurrences of this keyword. Figure 6.2 shows one template with the different keywords “browsing”, “network”, “link”, and “hypertext”. In the first of the three files the keywords “browsing” and “hypertext” occur, while the second document contains the keywords “browsing”, “hypertext”, and “link”. Finally, the third resource matches the keywords “network” and “link” [Hof96].

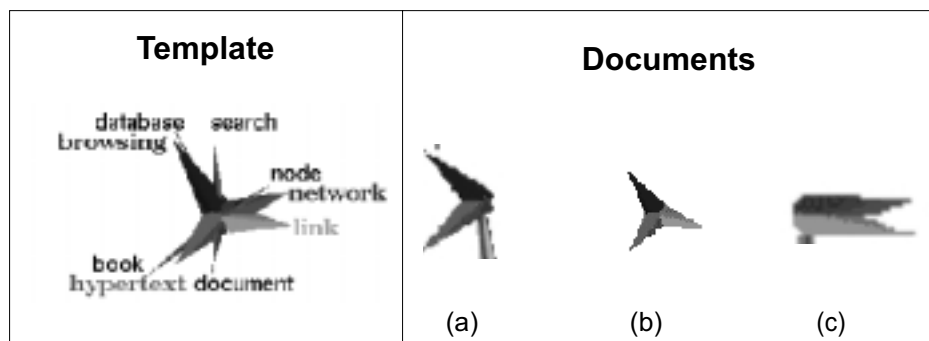


Figure 6.2: Arrow-sphere visualisation of 3 documents.

### 6.6.3 Metadata Visualisation on Axes

Once the metadata is retrieved, it is viewed as multidimensional data. The resources are presented in a two or three dimensional object space or in matrices of plots. It is not necessary to load the content of documents for this approach which results in relatively short load time and makes it applicable for visualisation of search result sets. The lack of trustworthy metadata for many resources is a problem, as is the absence of search engines which provide metadata information via well-defined interfaces. The Search Result Explorer overcomes the second problem by using the xFIND search engine described in Chapter 7. The problem of incorrect, incomplete, or missing metadata still remains. Many other systems followed this approach (e.g Envision). The difference to the second category, is that in

the above described visualisations, the axes have an inherent meaning which they do not have in the following category.

#### **6.6.4 Force-Based Methods**

This technique was already mentioned in Section 3.3.6. The positioning of the documents is an iterative process which tries to minimise the distances between similar items. In the special case of search result sets, the time consuming layout process is one problem. Additional difficulties arise because content information respectively word occurrence frequency has to be known first. As mentioned, specified meta keywords are often not trustworthy and it is better to extract appropriate keywords from the content. Thus a search engine which provides this term frequencies (e.g. xFIND) has to be used, or (not recommended) the content has to be loaded and the analysis has to be done for every document. In visualisations that follow this approach, spatial distances between documents indicate similarity and the axes (two or maximal three) have no inherent meaning.

### **6.7 Summary**

The result of analysing the existing tools is the fact, that although they use completely different techniques, they often provide similar functions in their display to improve usability, understanding and overview of the presented resources.

In summary, much research is currently being done to solve the problem of effective information discovery but no complete solution is yet in sight. Another serious disadvantage is the isolated work of the particular tools, which leads to high network load. Problems also occur due to the absence of effective update mechanisms. The tools have not enough capacity to deal with the enormous number of new documents to say nothing about keeping their index or directory consistent and up to date.

## Chapter 7

# xFIND eXtended Framework for Information Discovery

### 7.1 Introduction

At the Institute for Information Processing and Computer Supported New Media (IICM) much research has been done in the field of information discovery, information management, and computer based training. As described in Chapter 6 commonly available search engines cause high network and server loading because the same Web content is searched by ever more robots. Other shortcomings are that these search engines provide no quality ratings, they can not deal with humans with different skills and interests, and narrowing searches to specific domains is not supported. Thus the number of delivered results is often high, but the majority of documents do not contain relevant information. A solution to these problems is a distributed hierarchical search index, a technique developed by the Harvest research project [BDH<sup>+</sup>94]. Such systems often try to provide additional information and advice relevant to the particular situation with the intention to improve problem solving for users. The possible areas of use of such systems go far beyond search abilities in the World Wide Web, they can cover knowledge management, corporate decision systems, Web based training and many others. Personal needs with respect to the current problem (task specific, position specific), previous experience and references to further domain knowledge (e.g. problem base, background library and communication with experts) are also taken into account.

Considering all these facts, points and requirements a group at the IICM has implemented the first prototype of an advanced search system, HIKS (Hierarchical Interactive Knowledge System) [DGK<sup>+</sup>99]. This prototype seems to meet the requirements described above in a wide range of applications and initiated the development of the future oriented concept xFIND, which is described in brief in this chapter.

### 7.2 Concept

“xFIND is a future-oriented knowledge discovery system” [GPM99]. The distributed concept of xFIND guarantees high scalability and allows the management of a huge amount of information. From the beginning it was designed to be an open system which means that it provides an API to search external databases. Another improvement is the ability to deal with highly dynamic resources. Systems with highly dynamic content can inform the system about changes and also to be re-indexed.

The modular xFIND system is subdivided into Gatherers, Indexers, and Brokers which can be

configured by users to concentrate on their favourite areas of interest or to search only certain information locations. This concept causes a reduction of network and server loads. xFIND also improves search results using meta data sets, quality rating labels and Web site descriptions. In contrast to other systems, xFIND enables users to retrieve the statistical information which is collected during the indexing. With help of xFIND users can influence the search process by specifying different weights for the occurrence of search terms in different parts of the resources. The system can also take protected data into account if the metadata information of these documents is made visible to the system. In this case, if such a document matches a query, the system will report the provided metadata and a comment, that the found resource is write protected. Another interesting feature of xFIND is the ability to specify which attributes of matching documents should be reported. Thus users can get additional information of particular documents on demand which makes it easier to choose the desired document from the search result set. To provide quality information about indexed resources, the xFIND team has defined xQMS (see Section 2.6), a meta data scheme for classification of resources depending on their richness of information and fitness for use. As an important part of the scheme xQMS includes fields which can be used to ensure that the quality metadata is trustworthy.

## 7.3 Implementation

To achieve platform-independence xFIND is implemented in Java. In order to achieve scalability it is split into the following three main parts:

- **Gatherer:** This component visits servers, sites and resources and collects the metadata.
- **Indexer:** Responsible for the management of the knowledge repository.
- **Knowledge Broker:** Interacts with the user and sends queries to one or more Indexers and retrieves the results.

### 7.3.1 The Gatherer

Figure 7.1 shows how these components are arranged and how they work together. Design considerations, functions and possible configurations of these three parts are described next: The Gatherer performs the task of visiting servers and gathering information from various sources as well as pre-processing the document data. It identifies title, keywords, type, language, and creation or modification time. In case of HTML files also links, images and other embedded objects like Java applets and meta data fields of such documents are taken into account. As a leap ahead it is mentioned here, that especially this information extracted from the content become an important part for visualisation in the Search Result Explorer (see Chapter 8). It also creates an electronic fingerprint of each information object. This fingerprint suffices to determine the trustworthiness of information in case of replication and allows detecting the origin of every piece of information. Embedded meta data in the document and external meta data will be processed. xFIND allows a wide range of configuration for pre-processing this data and handling meta data sets. The first implementation supports the following metadata sets:

- **Dublin Core Metadata (DC)** described in Section 2.4
- **Learning Object Metadata (LOM)** described in Section 2.5
- **xFIND Quality Metadata Scheme (xQMS)** described in Section 2.6

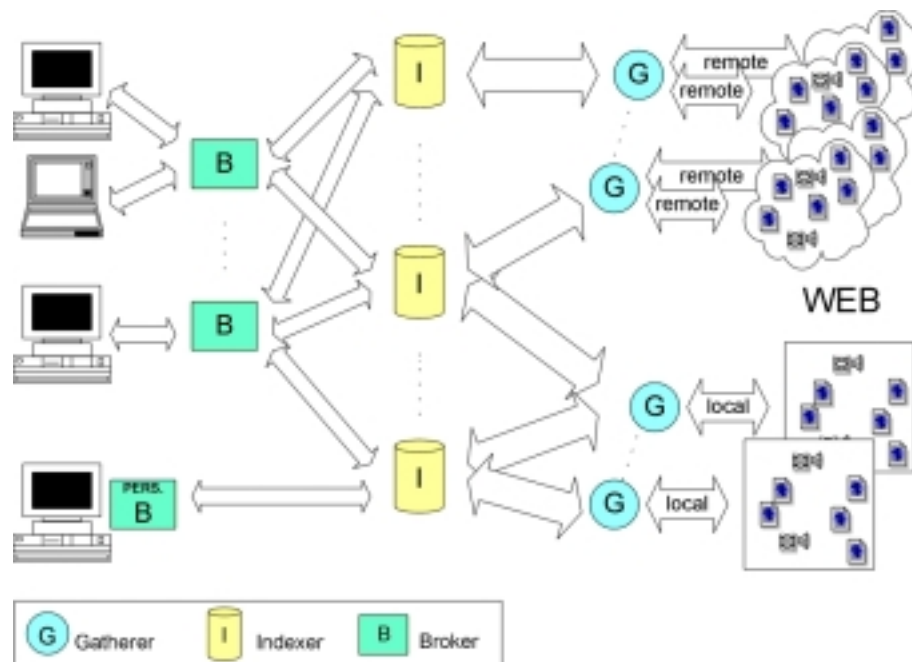


Figure 7.1: xFIND Architecture

- **Special xFIND Metadata** extracted from the content (e.g. number of embedded links or images).

Furthermore, the system allows the conversion between these meta data formats. The pre-processing and the reduction of information along with the usage of additional meta data improves the retrieval process. It should be pointed out that especially for learning environments sets of meta data are very useful. Unfortunately authors of information rarely enrich their documents with such additional description. xFIND supports authors to define meta data for a whole document structure, a directory or a particular document by inserting additional meta data files. More specific meta data overrides more general. Consequently, the enrichment with meta data is easier for authors and will help to improve the quality of information received by users. The first implementation of the xFIND Gatherer is able to process HTTP<sup>1</sup> sources and information stored in local file systems. The open concept allows to handle any other protocols and information systems. Best performance and reduction of server and network load can be reached by using a local Gatherer. A Local Gatherer can even be configured to search for read-protected information. Only pre-processed information will be provided to the xFIND system. This feature enables users to search for such read-protected information. They only get a configurable predefined set of meta information, so the original information remains protected. As already mentioned, xFIND will also serve active information systems. Therefore, an API and a communication layer has been designed. So active information systems (e.g. highly dynamic information like news tickers or stock rates) are able to contact xFIND and inform the system about new and modified pieces of information. External systems for information enrichment (like expert rating systems, announcement systems, collaboration systems, etc.) may be processed similarly. Gatherers also process meta descriptions about information sources and summarise statistical data about information sources. The latter may be used for detecting highly dynamic information or may give a summary about activities of areas of interest. This functionality is one possible approach to support the lifecycle of information in learning environments [GPM99].

<sup>1</sup>Hypertext Transfer Protocol



### 7.3.2 The Indexer

Indexers are the connection between Gatherers and Brokers. Their task is to allow the Knowledge Broker to assign words, phrases and meta data to documents, and to provide statistical data (e.g. term frequencies). They store information about many kinds of resources from Web pages to Web areas (e.g. number of changed or new documents for web areas). The indexer manages the important communication with external systems (ranking systems, ACF<sup>2</sup>, archiving systems, etc.). The pre-processed data can be fetched or sent compressed to one or more Indexers. An Indexer may be specialised on a particular topic or can be dedicated to a project group or a department.

*“Only authorised Indexers are allowed to operate with Gatherers. The latter, if trusted, are allowed to send additional information to the xFIND system or can inform xFIND about new or modified pieces of information”* [GPM99].

Both, descriptions and external additional information will improve the information structure for the learning process with respect to the information lifecycle. Furthermore, xFIND allows either the whole set of information of the Indexer or particular parts (dependent to topics or information sources) to be replicated from one Indexer to arbitrary others. The replication mechanism will reduce network traffic problems and will support a global learning platform. Fingerprints for all pieces of information and the relation to their origin will help to detect changes of contents. This feature is very important for information sharing between several companies and research centres. It should be mentioned that search facilities of external information systems can also be used for the distributed search process by using an xFIND API or a corresponding wrapper.

### 7.3.3 The Knowledge Broker

The starting point for user interactions is the Knowledge Broker. In a centralised system, a high request rate often leads to an intolerably high response time. Taking this problem under account, the Knowledge Broker is also following the distributed design concept of xFIND. As already discussed any Indexer manages parts of the whole knowledge repository and therefore Knowledge Brokers must distribute their search queries to a particular set of Indexers corresponding to the current search query. Due to the open concept searching in existing external information sources is possible. E.g. existing Hyperwave knowledge management systems are directly searchable and results can be merged with xFIND results. Knowledge Brokers can also be specialised for a particular topic. Furthermore, Knowledge Brokers may consider past search results and user ratings to improve future search queries and the quality of information. A Knowledge Broker is also able to transform search queries. Improved quality of information can be achieved by searching only problem-specific Indexers or parts of them. The feedback from users can influence the Indexers being searched. Due to the distributed architecture, Knowledge Brokers can be individually tailored for a division, a department, a group of employees or even for a single user.

*“Unlike common push tools or agent systems for information discovering, the discussed concept will not cause multiple network loads for any user. Quite similar to an agent system the Personal Knowledge Broker is adaptable to user habits and their current problems”* [GPM99].

Personalised Brokers will inform the user in case of new relevant information. In combination with an ordinary web browser, such personalised Brokers can provide additional information. The system is able to find out relevant keywords with respect to the users current interests. Such keywords may provide dynamically generated links, which will process a search by clicking and therefore supplying additional information. Furthermore, relations to similar documents and links to expert

---

<sup>2</sup>Automated Collaborative Filtering

or user opinions can be dynamically added to the original document. Terms of interest can then be marked within each document's result presentation.

## 7.4 Query Communication Format (QCF)

QCF is the query language standard for xFIND systems. Since this system is heavily under construction, here only an example of using this query language to get displayable search result attributes is shown. Papers describing the QCF and xQMS formats will follow soon and will also be present on the xFIND projects home page `xfind.iicm.edu`. The query which is sent to the xFIND system is basically an array of strings. The array length depends on the number of terms the query consists of, the number of query attributes the user wants to specify, and the number of attributes the user wants to retrieve. The array itself is divided into three parts:

- **Query:** This first part contains the query, the search words and boolean operators which combine these search words. The possibility of specifying where the search words should occur enables the user to influence the results. For example it is possible to search for a string "Graz" in the Keywords field AND the string "ICM" in the content using one such query. All these settings influence the estimated relevance of matching documents. Another improvement to common search engines is the ability to provide prefix, infix and postfix notation in the search terms. This offers the possibility to use wildcards such as '\*', '?'.
- **QueryAttributes:** In this second part of the string array users can specify settings which also influence the search. For example, the number of reported results or the amount of returned keywords. The search process can be further configured by varying the weights of words which occur in different parts of the document. For example, the system can be told to give an occurrence of a search term in the keywords the weight five and an occurrence in the content the weight one.
- **ReturnAttributes:** In this section the user or application cooperating with xFIND specifies which attributes of the found results to return. This very flexible procedure allows to send various kinds of queries from general to very special with full control of the result formats.

An example of building the above listed string array parts from a query used in the Search Results Explorer is described in Chapter 8.

## 7.5 xFIND Quality Metadata Scheme (xQMS)

xQMS is a metadata format under definition at the IICM. Besides the attribute set defined in the Dublin Core Metadata Standard (Title, Description, Keywords, Author, Contributor, ...) it will also contain quality information about the found resources (See Section 2.6). These values will also become an interesting field for information visualisation in future.

## 7.6 xFIND on the Web

On the project's homepage up to date informations about the concepts, the current project status, the team and the ongoing work can be found. A Search Query can be formed and sent to the server via the form shown in Figure 7.2 or a servlet can be used. It shows the extended possibilities that are

provided by the xFIND system. The user can influence how the documents should be ranked, by putting different weights on keywords, title, head and contents of the resource. This feature is not possible using other search services and should be a significant improvement. Additional attributes and meta data can also be requested via this dialog and this information is presented with the found documents, which improves the process of choosing one of the found documents. The results are presented in a textually oriented display which can also be configured by users. They are listed, sorted by a chosen attribute (e.g. estimated relevance or last modification time). The form and format of search results range from only the title or URL of the found documents to everything including title, URL, keywords, description and other existing meta data information.

Figure 7.2: xFIND Query Window.

### 7.6.1 Operating fields

The development of xFIND is ongoing, but some projects already cooperate with the system or use it to obtain descriptions of search results. These applications range from the GENTLE environment [GPM99, WBT99], a Web based training system also developed at the IICM which will use xFIND as a dynamic background library, through classic Web applications (Steiermark Hauptserver, Infomed, ccc-Server), to search result visualisations. The Search Result Explorer described in the next chapter provides an alternative to classic Web search tools by displaying search results in a two dimensional plot and providing several functions common in multidimensional visualisation systems. Furthermore, a force-based search result set visualisation project has just been started at the IICM.

## Chapter 8

# Search Result Explorer

### 8.1 Introduction

Since the amount of data increases rapidly on the World Wide Web, it is often difficult to decide which documents or files contain information users are searching for, and which of the results are up to date. Even if search engines are used, they typically produce a linear list of matching documents, ranked by estimated relevance without additional information about the documents (their metadata), and with no possibility to compare the retrieved documents. Thus much information which could be accessed and which would perhaps influence and speed up the next actions can not be taken into consideration. If users are unhappy with the search results, they compose a new query and wait for the new result. This can take dozens of steps and many minutes.

An attempt to improve this searching process is to visually present all attributes, metadata, and statistical information of search results. If users are able to compare age, size, last modification time, estimated relevance, authors, and other meta-information by one look at a graphical presentation, maybe they can find the relevant documents earlier. Several programs try to display and compare document attributes in different areas. It is possible to create meaningful two-dimensional displays by selecting ordinal attributes of the items and using them as axes. Other attributes can be mapped to size, colour and shape of the displayed graphic primitives.

In this chapter the Search Result Explorer is described. The user interface of the File Attribute Explorer was adapted to display search results generated by the xFIND search engine.

### 8.2 Design Principles

The design principles concerning the graphical display overlap those of the File Attribute Explorer described in Section 5.2. Comparing the user interfaces (Figure 5.1 and Figure 8.1) it can be seen, that only necessary changes which would make no sense when visualising search result sets, were made. For example the *Flatten Directory* option was removed. Additional information is displayed in the Table of Documents (Figure 8.2) and in the Document Detail Window (Figure 8.3). The difference to the corresponding components in the File Attribute Explorer program are an increased number of columns (i.e. richer metadata) for each document in the Table of Documents and accordingly more rows in the Document Detail Window. The sorting possibilities in both tables were adapted to make each column sortable. In the File Attribute Explorer the axes could be mapped to size, date, name, and directory depth. The last was removed in the Search Result Explorer and the following attributes were added:

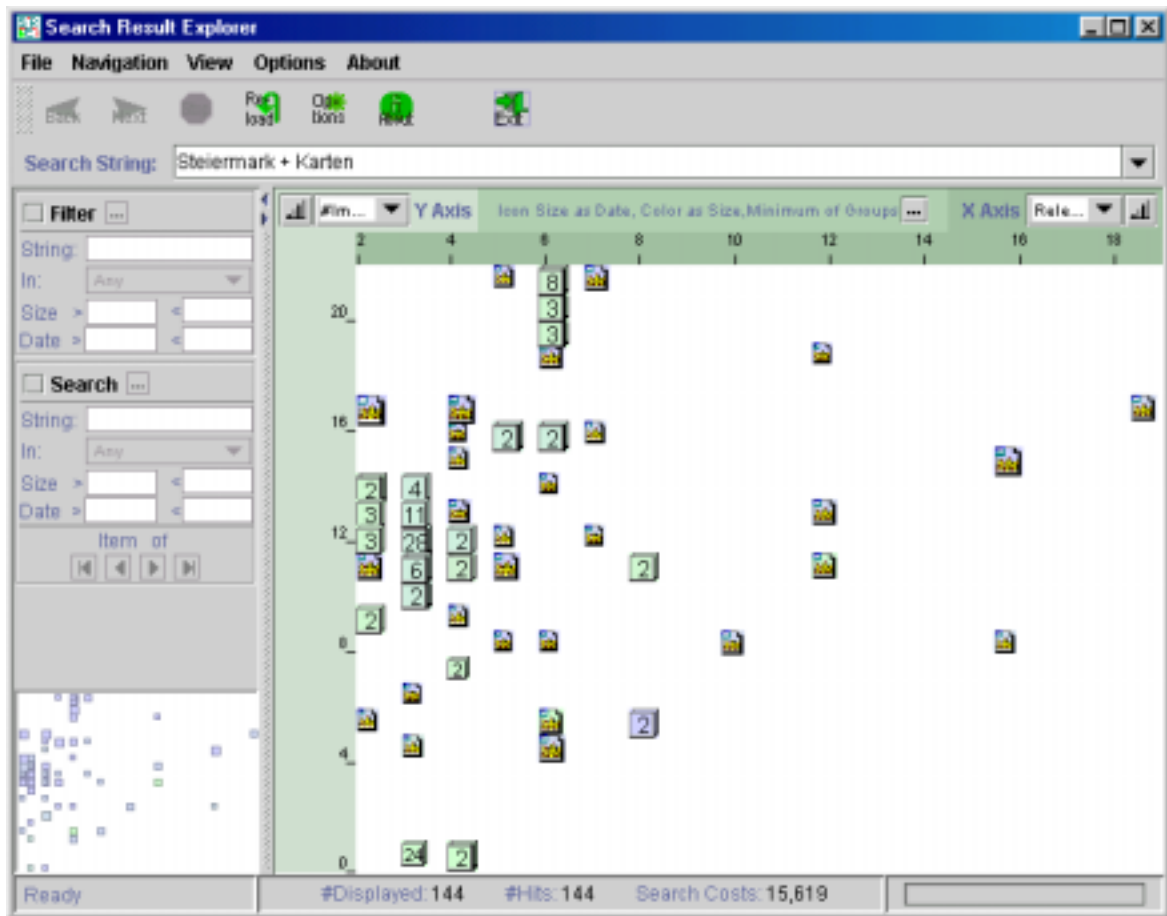


Figure 8.1: The Search Result Explorer Main Window.

Name	Size	Date	Time	Icon	Col...	Li...	I...	S...	R...	Keywords
Online-Mitteilungen Nr. 60 (19...	107075	17/03/98	10:38			0	0	0	2	beilstein 17, elsevier 6, mdl 6, ...
Online-Mitteilungen Nr. 63 (Fe...	122057	26/04/99	11:27			0	0	0	2	dm 4, winkler 3, ulrike 3, kirchg...
Online Mitteilungen 54 (Marz 1...	138070	27/11/97	09:40			0	0	0	2	erläuterungen 11, suchbefehle...
VOEB-Mitteilungen 48 (1995) 1	142189	27/11/97	09:42			0	1	0	3	fuer 4, medizin 3, kommission...
VÖB-Mitteilungen 52 (1999) 2	147697	14/09/99	09:04			0	2	0	21	zwecke 5, aspekte 3, erwägun...
VOEB-Mitteilungen 48-2	188713	27/11/97	09:42			0	1	0	13	bearb 2, schneider 2, vorläufig...
VÖB-Mitteilungen 51 (1998) 1	204782	07/04/98	10:56			0	2	0	16	psychoanalyse 12, werkblatt 6, ...
ODOK '97: Abstracts und Volle...	218185	20/01/98	09:24			0	7	0	3	celex 9, europäischen 8, swet...
VÖB-Mitteilungen 51 (1998) 2	237742	15/06/98	09:19			0	1	0	10	internationale 4, dokumentatio...
VÖB-Mitteilungen 52 (1999) 1	262507	19/04/99	08:10			0	5	0	44	pitfalls 4, müller 4, christa 3, s...
VOEB-Mitteilungen 49-2	265808	27/11/97	09:42			0	0	0	62	gmbh 14, bibliothekartag 5, fac...

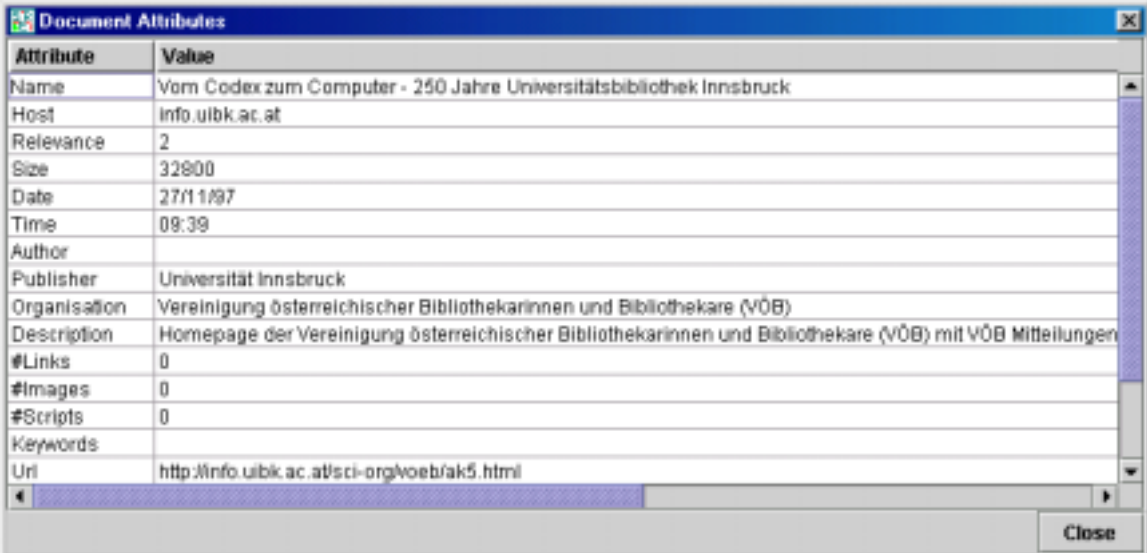
Figure 8.2: Table of Documents.

- **Relevance:** Means the estimated relevance of the document for the given query.
- **Host:** Specifies the server where the resource is located.
- **Number of links:** Can be useful to find documents that are tables of contents themselves.
- **Number of images:** Is the number of all images in the resource.
- **Number of scripts:** Can be seen as a degree of interactivity of resources.

Additional descriptive information (keywords, description, author, ...) can not be mapped to axes but is shown in the Document Detail Window e.g. to decide which document should be opened. Furthermore, keywords can be used as input for search and filter operations.

- **Keywords:** This string contains the keywords and their number of occurrences in the resource. As already mentioned, these keywords are extracted from metadata and content by the xFIND search engine.
- **Author:** The person who created, and is responsible for the content of the resource.
- **Publisher:** The entity responsible for making the resource available in its present form.
- **Organisation:** The organisation where the resource was created.
- **Description**

The search and filter functions can operate case sensitive or insensitive and match whole words only. Section 8.4 describes how the query has to be constructed.



Attribute	Value
Name	Vom Codex zum Computer - 250 Jahre Universitätsbibliothek Innsbruck
Host	info.uibk.ac.at
Relevance	2
Size	32800
Date	27/11/97
Time	09:39
Author	
Publisher	Universität Innsbruck
Organisation	Vereinigung österreichischer Bibliothekarinnen und Bibliothekare (VÖB)
Description	Homepage der Vereinigung österreichischer Bibliothekarinnen und Bibliothekare (VÖB) mit VÖB Mitteilungen
#Links	0
#Images	0
#Scripts	0
Keywords	
Uri	http://info.uibk.ac.at/sci-org/voeb/ak5.html

Figure 8.3: Document Detail Window.

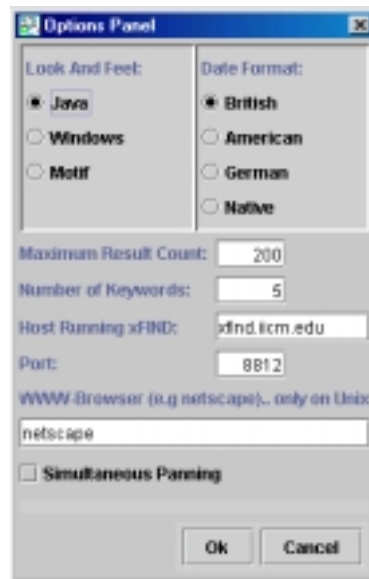


Figure 8.4: Search Result Explorer Options Window.

## 8.3 Options Window

The options different from the File Attribute Explorer are:

- **Maximum Result Count:** Lets the user specify the number of reported documents. When the search engine finds more than this number of resources which match the query, it delivers the number specified here with the highest estimated relevance.
- **Number of Keywords:** The xFIND search engine extracts the keywords from the meta keywords and the document content. This results in a large number of keywords which increases the descriptive information on the one hand, but on the other hand it also increases the load time. To give users control over this fact, they can specify the number of keywords, they want to retrieve. The retrieved keywords are ranked themselves by xFIND and reported according to their estimated relevance.
- **Host Running xFIND:** The domain name of the server where the chosen xFIND search engine is running.
- **Port:** The port number where xFIND waits for search requests.
- **WWW-Browser** On Unix systems the WWW browser which can display the documents must be specified (See 8.5.1 for details).

## 8.4 Communication with the xFIND Server

### 8.4.1 Query Language

This section covers the process of building the query string to obtain the attributes which are displayed in the Search Result Explorer. The grammar of queries used in the Search Result Explorer in Bacchus Naur Form is:

```

query ::= operand
      | operand operator operand

operand ::= term
        | (operand operator operand)

operator ::= _operator_
          | )operator
          | operator(
          | '+'
          | '-'
          | '_AND_'
          | '_ANDNOT_'
          | '_OR_'

term ::= String
      | key Term
      | Term key

key ::= '*' | '?' | '%' | '_'

```

In fact this query language differs a lot from the Query Communication Format (QCF), and not all the capabilities of xFIND are implemented yet.

### Query Example

In the following example, the query is assumed to be “Graz AND Marathon”. The three parts of the string array are:

**Query:** This part of the array holds the strings and the operators between them in the format required by xFIND. The following example shows this part for the above chosen example:

```
{ "3", "Query", "1.0", "6", "", "Graz", "op", "+", "", "Marathon" }
```

The first number specifies the number of parts of the string. The second term, “Query”, gives the current part its name and is followed by the current version number (“1.0”). The following term holds the number of key value pairs in the Query part. The two empty strings in front of “Marathon” and “Graz” tells xFIND to search for these terms in every part of the indexed documents. A non empty string would force the search engine to narrow its search to certain parts of resources. (E.g. “K”, “Graz” would find only documents where “Graz” is part of the keywords). The term “op” is the key of an operator and is followed by a boolean operator or by a parenthesis.

**QueryAttributes:** In QCF this part offers possibilities to influence the search (e.g by specifying different weights for different parts of resources). In the Search Result Explorer the following syntax is used:

```
{ "QueryAttributes", "1.0", "6", "Object.ResultLimit", "200",
  "Object.keywords.Resultlimit", "10" }
"Object.keywords.percentagelimit.maximum", "1" }
```

This part consist of its name (“QueryAttributes”), its version number (“1.0”) and the number of returned documents (“200” in the example). This number determines how many



document-attribute-records are returned if the number of hits is larger than this limit. This value can be specified by the user in the Options Dialog. The *Object.Keywords.Resultlimit* determines the number of returned keywords (“10” in the example). The third key/value pair *Object.keywords.percentagelimit.maximum* is necessary to retrieve object keywords. It specifies, that only keywords are reported, which occur in less than this percentage of documents.

**ReturnAttributes** Finally the interesting attributes which will be visualised have to be requested. This part of the query string starts with:

```
"ReturnAttributes" ,"1.0" ,"26"
```

specifying the name of this part of the query string, its version number, and the number of requested attributes. The attributes requested from xFIND and afterwards used in the Search Result Explorer are listed in Table 8.1. Between each attribute an empty string is inserted, to strictly follow the concept of key value pairs.

ReturnAttribute Name	Return Value	Program Terminus
Object.Title	Title or Filename	Title
Object.Url	URL of the resource	Host and URL
Object.Last-Modification-Time	Time since 1/1/70 in seconds	Date
Object.KeyWords	Keywords and their occurrences	Keywords
Object.File-Size	Size of plain file in Bytes	Size
Object.Url-Reference	String of URLs	Link Count
Object.Scripts	String of Script-Names	Script Count
Object.Pictures	String of Image Names	Image Count
Site.Description	Meta Description	Description
Site.Contributor.Author,	Meta author information	Author
Site.Contributor.Publisher	Publisher	Publisher
Site.Contributor.Organization	Organization	Organization

Table 8.1: Return Attributes requested by the Search Result Explorer.

Once the server has accepted a new request, it matches the query to its index and delivers the attributes of found documents back to the SRE. The SRE takes these values, calculates proper formats, sorts the data, stores the resource attributes into its data structure, and presents the icons in its display.

#### 8.4.2 Receiving Document Attributes

This section describes the process of reading, calculation and storage of the search results sent by xFIND. The sent data itself also consists of three parts:

1. **Header:** In this part the search engine reports the number of documents which match the given query. This value is used to initialise the data structures and to provide a meaningful output of the progress bar.
2. **Results:** The search engine sends key value pairs where the key is one of the requested terms from Table 8.1 followed by its value. This process is repeated for each document which is part of the search result set. To provide meaningful information some of these values have to be

processed into proper formats. For example the last modification time is specified in seconds since 1st January 1970 which has no equivalent Java datatype and the number of links, images and scripts has to be calculated from the returned URLs, image names, and script names.

3. **Footer:** The footer reports the search costs and closes the connection.

## 8.5 Selected Implementation Details

### 8.5.1 Controlling a WWW Browser

The *BrowserControl* class allows to control the system's native browser from within any Java application. With a little platform-specific Java code, the system's default browser can be used to display any URL. The second implementation issue is to recognise that a browser is already running, and to use it if possible.

On Unix, for Netscape, the command is:

```
netscape http://+desired URL}
```

And, if the browser is already running it is:

```
netscape -remote openURL(http://+desired URL)
```

Under Windows, it took much exploration to find something equivalent that would not open a new browser for each request. The used command, in fact, works better than the Unix command, because it invokes the default browser rather than the hard-coded way to run a browser on Unix. If Microsoft's Internet Explorer is the default browser, then the page will be displayed in Internet Explorer.

To display a page, the following command on Microsoft Windows is used:

```
rundll32 url.dll,FileProtocolHandler http://+desired URL
```

The following code shows the static *displayURL* method, which opens the URL of the double clicked icon in an available browser on Unix or Windows.

```
public static void displayURL(String UNIX_PATH, String url)
{
    private String WIN_PATH = "rundll32";
    private String WIN_FLAG = "url.dll,FileProtocolHandler";
    private String UNIX_FLAG = "-raise -remote openURL";

    boolean windows = isWindowsPlatform();
    String cmd = null;
    try
    {
        if (windows)
        {
            cmd = WIN_PATH + " " + WIN_FLAG + " " + url;
            Process p = Runtime.getRuntime().exec(cmd);
        }
        else
        {

```

```

cmd = UNIX_PATH + " " + UNIX_FLAG + "(" + url + ")";
Process p = Runtime.getRuntime().exec(cmd);
try
{
    int exitCode = p.waitFor();
    if (exitCode != 0)
    {
        cmd = UNIX_PATH + " " + url;
        p = Runtime.getRuntime().exec(cmd);
    }
}
catch(InterruptedException x)
{
    System.err.println("Error bringing up browser, cmd='" +cmd + "'");
    System.err.println("Caught: " + x);
}
}
catch(IOException x)
{
    System.err.println("Could not invoke browser, command=" + cmd);
    System.err.println("Caught: " + x);
}
}

```

The *isWindowsPlatform* function determines the operating system, on which the program is run. Under Unix, Netscape has to be running already for the “-remote” command to work.

## 8.5.2 Customising Table Display and Event Handling

The Table of Documents provides features which makes each line in the table equivalent to a graphic primitive in the Drawing Area. In order to achieve this behaviour the detection of row selections and deselections in the table, handled by a selection listener, had to be implemented.

```

ListSelectionModel rowSM = table.getSelectionModel();
rowSM.addListSelectionListener
(
    new ListSelectionListener()
    {
        public void valueChanged(ListSelectionEvent e)
        {
            ListSelectionModel lsm = (ListSelectionModel)e.getSource();
            if (!lsm.isSelectionEmpty())
            {
                if (e.getValueIsAdjusting()) return;
                if (!init)
                {
                    if (!dontSetFocus)
                    {
                        int selectedRow = lsm.getMinSelectionIndex();
                        SRExpl.setFocus(data2tableIndex[selectedRow]);
                        if (e.getClickCount() > 1) SRExpl.exploreInBrowser(
                            data2tableIndex[selectedRow]);
                    }
                }
            }
        }
    }
)

```

```
);
```

In this part of the *TableOfFiles* class, a list selection listener is registered to the table. If the user clicks on one row the *setFocus(int)* method of the *SRExpl* class is called causing the selected document to show its attributes in the Document Detail Window. If a double click is performed, the corresponding document is opened in a WWW browser.

The Table of Documents also provides sorting possibilities for each column ascending or descending. To indicate the current sort column an arrow is drawn in its header, showing the current sort order. To implement this behaviour a “renderer” for the column headers had to be written. To detect mouse clicks on a column header, a “mouse listener” is registered on the table header and causes sorting when a click is performed.

```
ImageIcon sortAscendingIcon, sortDescendingIcon;
...
sortAscendingIcon = new ImageIcon(TableDemo.class.
    getResource("icons/upTableArrow.gif"));
sortDescendingIcon = new ImageIcon(TableDemo.class.
    getResource("icons/downTableArrow.gif"));
...
class HeaderRenderer extends JLabel implements TableCellRenderer
{
    public HeaderRenderer()
    {
        super();
        setForeground(Color.black);
        setHorizontalAlignment(LEFT);
        setBorder(BorderFactory.createRaisedBevelBorder());
    }
    public Component getTableCellRendererComponent(JTable ta, Object ti,
        boolean sel, boolean fo, int row, int col)
    {
        setText(ti.toString());
        if (col == sortBy)
        {
            if (fallend) setIcon(sortDescendingIcon);
            else setIcon(sortAscendingIcon);
        }
        else setIcon(null);
        return this;
    }
}
```

This “custom renderer class” *HeaderRenderer* is a subclass of the Swing class *JLabel*. It also has to implement the *TableCellRenderer* interface and its *getTableCellRendererComponent* method. In this method the text of the label is set via its *setText* method and it is displayed in the column header. The labels icon is set to the “sortDescendingIcon” if the table is sorted descending by this column, it is set to the “sortAscendingIcon” if the table is sorted ascending or it is set to null if the table is sorted by another column.

The things which remain to do are:

1. Setting the “tables header renderer” to the customised *HeaderRenderer* class:

```
for (int i=0;i<11;i++)
```

```

{
    TableColumn currentColumn = table.getColumnModel().getColumn(i);
    currentColumn.setHeaderRenderer(new HeaderRenderer());
}

```

2. Register an “mouse event listener” which processes sort events:

```

MouseListener listMouseListener = new MouseAdapter()
{
    public void mouseReleased(MouseEvent e)
    {
        // code to determine which column is affected
        ...
        sorter.sortByColumn(column, ascending);
    }
}
JTableHeader th = table.getTableHeader();
th.addMouseListener(listMouseListener);

```

## 8.6 Future Work

Many improvements should follow to support all capabilities of the xFIND search engine. The shortcomings which concern the user interface were already discussed in 5.7 and remain an important issue for this application. The xFIND system is not complete and will grow in future. Many of the planned xFIND possibilities to specify queries are not taken into account in the current version of the Search Result Explorer. A search query dialog which supports these features should be implemented to guide users to submit intuitive well-formed queries which can use all those capabilities.

In the current version it is not really possible to bring discrete values to the axis and use the corresponding values as labels on the axis. This seems to be tricky, because taking care of many different types of discrete values is necessary.

The xQMS Quality Information metadata format will be used in the xFIND search system to classify resources according to their quality and fitness. One problem arises providing and holding such data up to date. A meaningful comparison is only possible when each resource is classified.

It will be possible to index other document types as HTML (such as PDF, Word) with the xFIND search engine. Such file types can not be opened with a double click in the current version. The search engine will hopefully deliver the file type. Thus the problem to solve is to find the appropriate application for particular file types.

## Chapter 9

# Concluding Remarks

In Chapter 2 it is argued, that appropriate metadata can be useful to address the exponential growth of information on the World Wide Web. The work of defining qualifiers is already done (DC 2.2, LOM 2.5), problems arise when the metadata information should be provided in machine understandable formats. This is solved by defining frameworks (RDF 2.7) which use a labeled graph to store the information. Thus the technical conditions are in place, however users rarely enrich their documents with meaningful metadata. This would be a first step however problems to ensure trustworthiness and keeping the metadata up to date still remain. Newer proposed metadata standards (xQMS 2.6, RDF with digital signatures) provide fields to address these problems but are not in use yet. Additional fields which describe the quality and suitability for use are also provided by xQMS.

To find an appropriate visualisation mechanism an overview of multidimensional visualisation is given in Chapter 3. The advantages of useful visualisations are shown, ranking from high interactivity to good overview and are a noticeable improvement to textual output. The question of which document visualisation systems already exist and how they work was especially examined. As one result it was found that no general method can be given to find the best visualisation for a given data set. It depends highly on the data, but most of the systems support similar functions, which were found to be useful and were implemented in the two programs.

Chapter 4 describes the Java programming language. Much work was done in the implementations to provide a powerful user interface. The Java Foundation Classes had to be customised in many ways to get the desired behaviour. As Java is supported in modern Web browsers, it will be possible to run the program as an applet in such browsers. The requirement for running Java Virtual Machine would be avoided and the Search Result Explorer could become a real alternative to existing search tools. Since the capabilities of browsers always lag behind the Java Runtime Environment this integration is directed to future work.

The File Attribute Explorer (Chapter 5) is the first program which emerged during the work on this thesis. It takes the metadata attributes delivered from common file systems and displays them in a two dimensional plot. Additional attributes are mapped to size and colour, making it a multidimensional visualisation tool. To improve the usability of this tool, it is currently (December 1999) being tested by students at Graz University of Technology.

Current Internet search tools are compared in Chapter 6. Unfortunately, they are far away from being the key to effective information discovery. There are diverse reasons for this fact. The tools can not effectively deal with huge numbers of documents which are additionally changing their contents and locations. The resources itself provide too little or wrong metadata, the calculation of correct estimated relevances is not straightforward and search tools do not provide interfaces for other applications to benefit from their data. The Z39.50 protocol, which could be used to exchange data about

such search result sets, is not widely used.

The xFIND project is the subject of Chapter 7. It provides the functionality to return the metadata of search result sets which is then visualised in the Explorer. The open concept and possibilities to configure the system for different fields of use makes it an interesting alternative to existing search engines.

The Search Result Explorer (Chapter 8) in its presented version is the conclusion of this thesis. It provides an alternative to current search tools on the Internet. The program benefits from rich metadata which is made available by the xFIND search engine to provide an explorable interface to a search result set. Eight different metadata attributes can be mapped to one of two axes or to the document icon's size or colour.

## Appendix A

# File Attribute Explorer User Guide

The intent of developing this tool was to provide a powerful hierarchy explorer for file systems. The goal was to provide a tool that is easy to use, and a display that is easy to understand. The visual complexity and the difficulties in orientation and navigation of displays using more than two spatial dimensions led to a concept of using a two-dimensional plot. Further dimensions are brought to the system using colour, size and shape. Interactivity, flexibility in scaling, and functions for details on demand were implemented. The hierarchy can also be flattened to compare the contained files without regard to their depth in the directory tree. Additional search and filter functions improve the overview and can be used to scope on specific types of data.

### A.1 Installation

The File Attribute Explorer was developed with the Java Development Kit (JDK) 1.2. It works on every operating system (Unix, Windows95, WinNT, MacOS, ...) where the Java Virtual Machine 1.2 (or higher) is installed. The program is packed into one executable jar file and is run typing the following statement on the command line:

```
java -jar fae.jar
```

Several program settings are configurable and stored in a configuration file. During program execution, if settings like the used display shape, the “look and feel”, the chosen date and time formats, mapped colours, window positions or sizes are changed, they are stored and remembered in the property object and its configuration file.

### A.2 Functions

The main functions of the program can be performed using one of the GUI-components menu bar, tool bar or popup menus. Figure A.1 shows the main window of the File Attribute Explorer and labels particular parts as they are named in this user guide.

Users specify the directory they want to start from and the program displays icons representing the files located in this directory. If the *Flatten Directories* option is selected, all files in all subdirectories are also included in the display. Files and their attributes are also shown in a second tabular display. In this second view, every file occupies one line and actions (e.g selections) are synchronised between both displays.



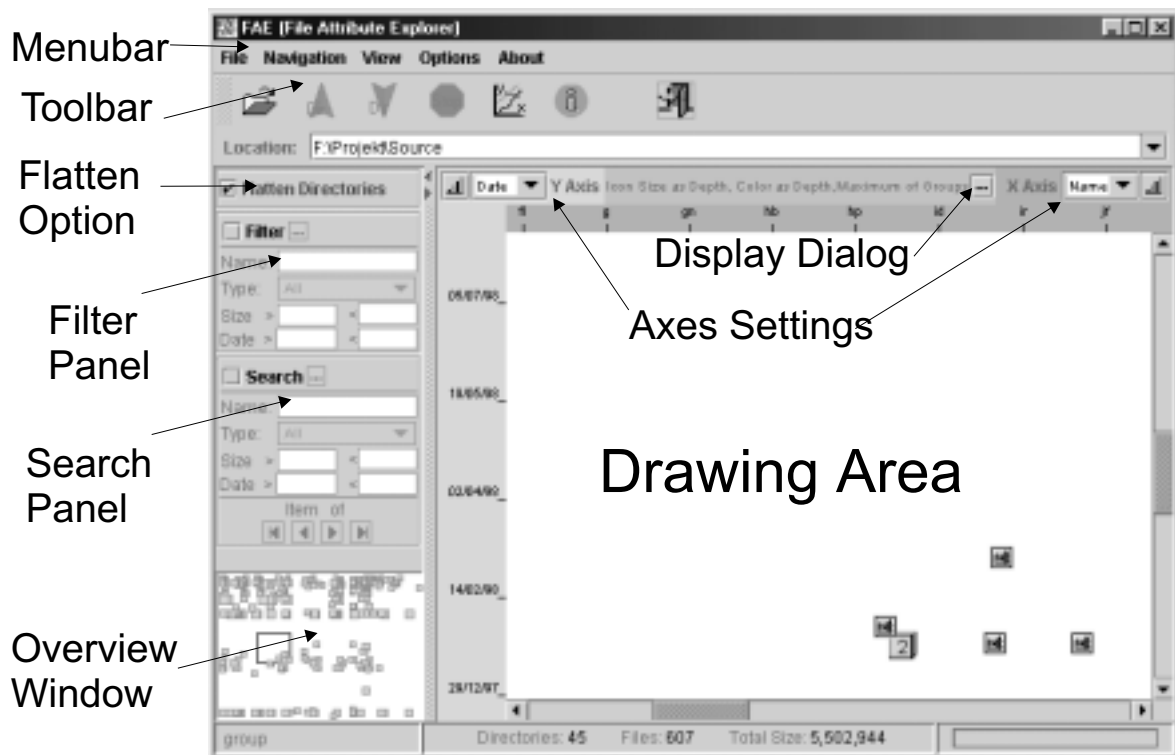


Figure A.1: File Attribute Explorer Main Window.

The functions are described in the order they appear in the menubar. From left to right the menubar provides the following menus:

- **File**
- **Navigation**
- **View**
- **Options**
- **About**

Some menu functions can also be executed via the toolbar or via popup menus. All functions from the menus can be alternatively performed with the shortcuts listed in Table A.1.

### A.2.1 File Menu

Contained Functions:

**Open:** Brings up a Filechooser. The used filter only accepts directories as possible input. This dialog can also be opened via the first button of the toolbar. A second possibility to specify a new directory to be loaded next, is typing in into the Location field.

**Exit:** Brings up the exit dialog. After selecting *yes*, the program terminates. This action can also be performed using the last button of the toolbar.

Shortcut	Function
<b>Alt-O</b>	Opens the JFileChooser to select a new directory.
<b>Alt-E</b>	Exits the File Attribute Explorer.
<b>Alt-S</b>	Stops a currently running load thread.
<b>Alt-Z</b>	Forces a Zoom Out action.
<b>Alt-R</b>	All Zoom actions are undone.
<b>Alt-B</b>	Goes Back to the previously explored directory.
<b>Alt-F</b>	Goes Forward to the previously explored directory.
<b>Alt-P</b>	Opens the Options Panel.
<b>Alt-A</b>	Opens the About Window.
<b>Alt-L</b>	Opens (closes) the Table of Files.
<b>Alt-T</b>	Shows (hides) the toolbar.

Table A.1: Keyboard shortcuts.

### A.2.2 Navigation Menu

The three functions located in this menu provide the ability to move between previously explored directories, which were “cached” in a special data structure. The functions can be accessed alternatively, using the popup menu in the Drawing Area, or via the corresponding toolbar button, or by typing a keyboard shortcut.

**Forward:** Goes forward to a directory which was previously explored. This is only possible, if a back operation was performed first.

**Back:** Goes back to the directory which was previously explored. This function is the opposite of the above described Forward action and is provided in the same GUI-components.

**Stop:** The analysis of the hierarchy can be stopped, e.g. if it lasts too long. In this case, specifying a search depth border can help.

### A.2.3 View Menu

Settings, which can be changed in this menu influence the user interface and the zooming behaviour.

**Show Toolbar:** As already mentioned, the program provides different possibilities to access the main function. All the button actions of the toolbar can alternatively be performed using actions and shortcuts. If users prefer these ways, they can hide the tool bar using the checkbox.

**Show Files Table:** The Table of Files is holding the attributes of the displayed files sorted according to a selectable column. This view can be switched on or off using the second checkbox.

**Zoom Out:** The last zoom in action is un-done.

**Zoom Reset:** All previous zoom in actions are cleared and all file items are visible (in a group or as single file) on the surface.

### A.2.4 Options Menu

Settings which can be changed in this menu influence the look and feel, the date format and the view on the hierarchy. and .

**Look and Feel:** In the current version the three settings Java (Metal), Windows, and Motif (Unix) are possible.

**Date Format:** Whenever a date value is presented in the program, it is written in the chosen format.

**Scan Depth:** The analysis of the directory hierarchy stops at the given depth. If no depth is specified, the whole hierarchy is loaded.

**Simultaneous Panning:** This option has its effects only if the user has zoomed into a particular region of the Drawing Area. In this case a black rectangle in the Zoom Overview Window shows this region and its position relative to the overall display borders. This rectangle can be moved with the mouse. If the option is selected the update in the Drawing Area is done simultaneously. This gives the impression of “flying over the surface”, but it may be intolerably slow on some machines.

### A.2.5 About Menu

This menu button opens the About Window. It shows the current project status. The author, the supervisor, and the license agreement of the IICM are also contained.

### A.2.6 Flatten Option

If this option is selected, all files in subdirectories are also displayed.

### A.2.7 Search and Filter

These functions offer possibilities to concentrate on specific file types, and to narrow size or date attributes of the displayed files. Filenames are matched using the *FNMatch* class. This class is based on the GNU-Library *fnmatch* routine which provides an intuitive “file name matcher” rather than using regular expression matching. This class can be configured to either respect or ignore case. It also provides possibilities to match whole worlds only. The user can specify how the actual search or filter operation should work via a popup menu. The buttons in the search and filter panels bring up this popup menu. To clearly indicate whether search and filter are active, the *Search* and *Filter* labels are coloured red when active.

### A.2.8 Zoom

As already mentioned the ability to zoom guides the user from overview to detail and vice versa. Table A.2 summarises the possible zooming operations and where they are offered:

### A.2.9 Axis Settings

The following interesting attribute values can be displayed on each axis:

- **File Size**

	<b>Zoom in</b>	<b>Zoom out</b>	<b>Zoom reset</b>
<b>Drawing Area</b>	Double click on group. Dragging a rectangle.	Popup menu via right mouse button.	Popup menu via right mouse button.
<b>Overview Window</b>	Dragging a rectangle.	Popup menu via right mouse button.	Popup menu via right mouse button.

Table A.2: Possible Zooming actions.

- **File Date**
- **File Name**
- **Directory Depth** the relative depth of the file in the hierarchy.

### A.2.10 Display Settings Panel

The panel is implemented as a modal dialog which allows to switch between main window and settings dialog and results in simultaneous visualisation update when changing display settings.

**Icon Shape:** Users can choose the graphical representation for files in the Drawing Area. The files can be displayed as icons, squares or circles.

**Icon Size:** The attribute which is responsible for the size can be chosen. The borders in which the icon size lies in can also be specified in this line.

**Icon Colour:** Again the mapped attribute can be chosen in a ComboBox. The used colours are displayed as two small image buttons which bring up a JColourChooser to change the colours.

**Group Display Behaviour:** One group contains the number of files shown in the middle of the icon. The problem is, that the grouped files have similar values for the attributes scaled on the axis but the attributes mapped to size or colour may differ. The chosen solution gives the user control over what value should be mapped. The possible settings are *Minimum*, *Maximum*, *Medium* and *Median*.

**Grouping Distance:** This integer value is used to decide whether two side by side lying objects are represented as one group or as two items.

### A.2.11 Mouse Functionality

To provide a powerful application, several features were implemented. Most of them are controlled with the mouse and they are listed in Table A.3.

### A.2.12 Sort Functions

Every table of the File Attribute Explorer which contains more than one file can be sorted. The sort column can be specified clicking on the corresponding table header. The sort order is shown as a black arrow and can be changed with a click on the current sort column. Popup menus further improve the usability of the sort function.

	<b>Left Click</b>	<b>Right Click</b>	<b>Double Click</b>	<b>Mouse Drag</b>
<b>Overview Window</b>	-	Brings up a popup menu.	-	Zooms into or moves the drawn rectangle.
<b>Drawing Area</b>	-	Brings up a popup menu.	-	Zooms into the drawn rectangle.
<b>Table of Files</b>	The icon is marked in the area.	Opens the detail view of the file. Header: Brings up a popup menu.	Opens directory icons.	-
<b>Detail Window</b>	Opens the detail view (group icons).	Brings up a popup menu (header).	-	-
<b>Icon</b>	The file is marked in the table.	Opens the detail view of the file.	Zooms into group icons, opens directory icons.	-

Table A.3: Mouse Functionality in the File Attribute Explorer.

### Realized File Types

At the moment the program recognises the following file types by analysing the suffix of filenames:

- **Text Document** The document is a text document.
- **Picture** The document is an image.
- **Audio File** The document is a sound file.
- **Postscript** The document is a postscript file.
- **Unknown Format** The document has no known format.
- **Word File** The document is a Microsoft Word file.
- **Excel File** The document is a Microsoft Excel file.
- **Java Source** The document is a Java file.
- **Latex File** The document is a Latex file.
- **Html File** The document is a Html file.
- **Group** This icon represents a group of files. The distance between the icons is too small to show each icon. The number of files contained in this object is displayed in the middle of the group icon. A double click on this icon causes a zoom action in order to show all files and their own icons to make their attributes accessible.

## Appendix B

# Search Result Explorer User Guide

This tool was designed to explore search result sets, retrieved from a xFIND search engine. It should become possible, to analyse the result set with immediate overview over the found documents. The basic visualisation concept is a two dimensional scatterplot, with many additional possibilities and functions to improve and fasten the process of selecting the document with most relevant information. Additional search and filter functions improve the overview and can be used to focus on documents with specific attribute values.

### B.1 Installation

This program was developed with the JDK 1.2<sup>1</sup>. It works on every operating system (Unix, Windows95, WinNT, MacOS...) where the Java Virtual Machine 1.2 (or higher) is installed. The program is packed into one executable jar file and is run typing the following statement on the command line: (under the condition, that the PATH variable includes the JDK bin directory.

```
java -jar sre.jar
```

Several program settings are configurable and stored in file to obtain persistence. This file is called `SREsettings.par` and is stored in the personal home directory of every user. If users run the program for their first time this file will be created and it will remember the user settings for further executions.

### B.2 Functions

The main functions of the program can be performed using one of the GUI-components menu bar, tool bar or popup menus (See Figure B.1). The functions are described in the order they appear in the menu bar. From left to right the menubar provides the following menus:

- **File**
- **Navigation**
- **View**
- **Options**

---

<sup>1</sup>Java Development Kit 1.2

- **About**

Some menu functions can also be executed via the toolbar or using popup menus. All functions from the menus can be alternatively performed using the keyboard shortcuts listed in Table B.1.

### **B.2.1 File Menu**

**Exit** Brings up the exit dialog. After selecting yes the program terminates. This action can also be performed using the last button of the toolbar.

### **B.2.2 Navigation Menu**

The three functions located in this menu provide the ability to move between previously explored result sets, which are held in a special data structure. The functions can be accessed alternatively, using the popup menu in the Drawing Area or via the corresponding tool bar button or by typing the keyboard shortcut.

**Forward:** Goes forward to a result set which was previously found. This is only possible, if a back operation was performed first.

**Back:** Goes back to the result set which was previously found. This function is the opposite of the above described **Forward** action.

**Stop:** The loading of result metadata can be stopped, e.g. if the process lasts too long. In this case, specifying a smaller number of documents to return in the Options Panel can help.

### **B.2.3 View Menu**

Settings which can be changed in this menu influence the user interface and the zooming behaviour.

**Show Toolbar:** As already mentioned, the program provides different possibilities to access the main function. All the button actions of the toolbar can alternatively be performed using actions and shortcuts. If users prefer these ways, they can hide the tool bar using the checkbox.

**Show Table of Documents:** The Table of Documents is holding the attributes of the displayed documents sorted according to a selectable column. This view can be switched on or off using the second checkbox.

**Zoom Out:** The last zoom in action is un-done.

**Zoom Reset:** All previous zoom in actions are cleared and all icons are visible (either in a group or as a single document) on the surface.

### **B.2.4 Options Menu**

Settings which can be changed in this menu influence the user interface itself, the zooming behaviour, they determine the used search engine and their settings.

**Look and Feel:** In the current version the three settings Java (Metal), Windows, and Motif (Unix) are possible.

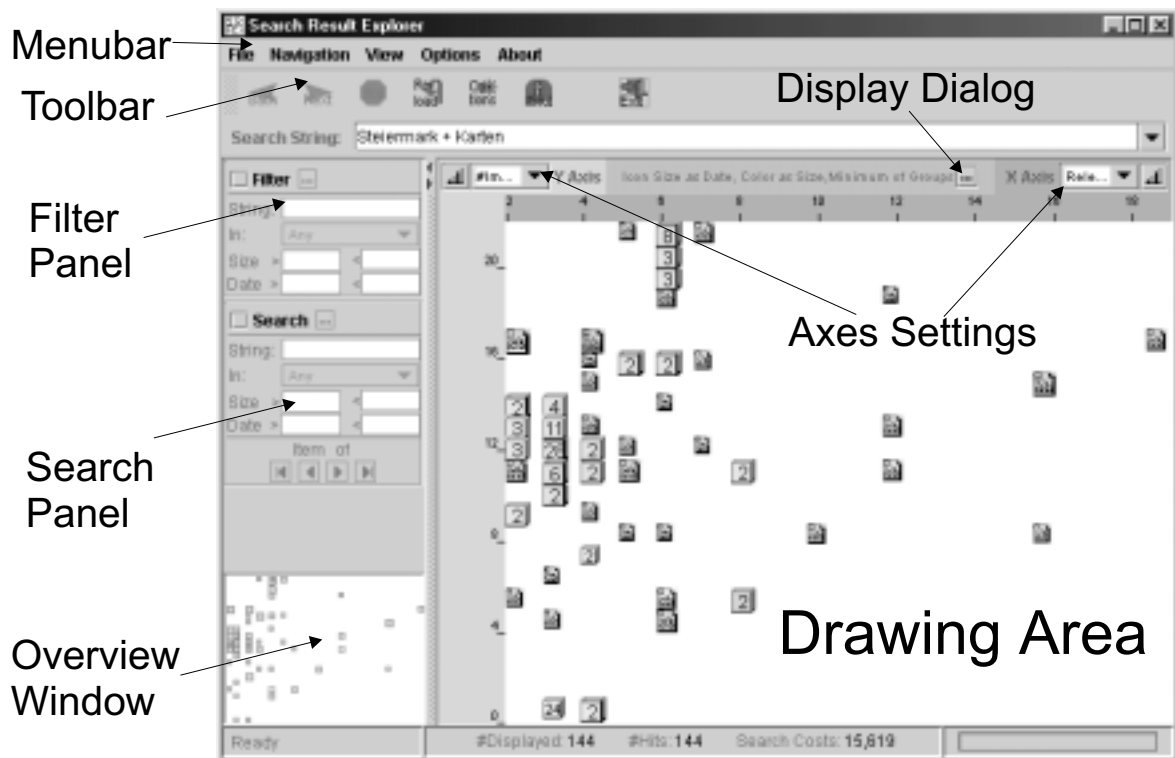


Figure B.1: File Attribute Explorer Main Window

Shortcut	Function.
<b>Alt-E</b>	Exits the Search Result Explorer.
<b>Alt-S</b>	Stops a currently running load thread.
<b>Alt-Z</b>	Forces a Zoom Out action.
<b>Alt-R</b>	All Zoom Actions are undone.
<b>Alt-B</b>	Goes Back to the previously searched Query.
<b>Alt-F</b>	Goes Forward to the previously searched Query.
<b>Alt-P</b>	Opens the Options Panel.
<b>Alt-A</b>	Opens the About Dialog.
<b>Alt-L</b>	Opens (closes) the Table of Documents.
<b>Alt-T</b>	Shows (hides) the Toolbar.
<b>Alt-N</b>	Reloads the current query from xFIND.

Table B.1: Keyboard Shortcut for the Search Result Explorer.



	<b>Zoom in</b>	<b>Zoom out</b>	<b>Zoom reset</b>
<b>Drawing Area</b>	Double click on group. Dragging a rectangle.	Popup menu via right mouse button.	Popup menu via right mouse button.
<b>Overview Window</b>	Dragging a rectangle.	Popup menu via right mouse button.	Popup menu via right mouse button.

Table B.2: Possible Zooming actions.

**Date Format:** Whenever a date value is presented in the program, its format depends on the chosen format.

**Number of returned Results:** The Search Result Explorer delivers the specified number of documents. (If more documents are found, the rest is ignored)

**Number of Keywords per Result:** As mentioned, xFIND extracts document keywords from meta-data and content. The number of keywords can be very high thus this number limits the amount of returned keywords.

**Host:** In this field the server, which hosts the xFIND search engine, is specified (e.g. `xfind.iicm.edu`).

**Port:** Here the port number is specified (e.g. 8810).

**Browser String** The program which should be used to open particular documents can be specified in this line.

### B.2.5 About Menu

To bring up the About Search Result Explorer Window which gives information about the program and shows the license agreement.

### B.2.6 Search and Filter

These functions offer possibilities to concentrate on documents with specific attribute values. The match can be configured to match case sensitive or to ignore cases. The “whole word” option and wildcards offer further possibilities to perform selective searches. Searches and filters can be performed in different attribute fields (title, keywords, URL or host) or in all of them. Users can also specify how the actual search or filter operation should work via a popup menu.

The results, matching a search are drawn with a red border and can be viewed one after the other using four navigate buttons. If a filter is applied, only the documents which pass this filter are displayed in the visualisation and listed in the Table of Documents.

### B.2.7 Zoom

As already mentioned the ability to zoom guides the user from overview to detail and vice versa. Table B.2 summarises the possible zooming operations and where they are offered.

### B.2.8 Axis Settings

The Axis can be scaled with different attribute values. The possibilities are *Relevance*, *File-Size*, *File-Age*, *File-Name*, *#Links*, *#Images* and *#Scripts*. The ordering of the axes-values can be ascending or descending, which can be changed with buttons.

### B.2.9 Display Settings

The current display settings are given in one line of text at the top of the Drawing Area. The following button brings up the Display Settings Panel which controls these settings.

**Icon Shape:** The user can choose his favourite graphical representation: The documents can be displayed as icons, squares or circles. When displayed as squares, no groups are formed, the overlapping icons are drawn side by side.

**Icon Size:** The attribute which is responsible for the size can be chosen. The borders in which the icon size lies in can also be specified in this line.

**Icon Colour** Again the mapped attribute can be chosen in a ComboBox. The used colours are displayed as two small image buttons which bring up a JColourChooser to change the colours.

**Group Display Behaviour** One group contains the number of documents shown in the middle of the icon. The problem is, that the grouped documents have similar values for the attributes scaled on the axis but the attributes mapped to size or colour may differ. The chosen solution gives the user control over what value should be mapped. The possible settings are *Minimum*, *Maximum*, *Medium* and *Median*.

**Grouping Distance** This integer value is used to decide whether two side by side lying objects are represented as one group or as two items.

### B.2.10 Mouse Functionality

To provide a powerful application, several features were implemented. Most of them are controlled with the mouse and they are listed in Table B.3.

### B.2.11 Sort Functions

Every table (Table Documents, Document Attributes Window) which contains more than one document can be sorted. The sort column can be specified clicking on the corresponding table header. The sort order is shown as a black arrow and can be changed with a click on the current sort column. Popup menus further improve the usability of the sort function.

	<b>Left Click</b>	<b>Right Click</b>	<b>Double Click</b>	<b>Mouse Drag</b>
<b>Overview Window</b>	-	Brings up a popup menu.	-	Zooms into or moves the drawn rectangle.
<b>Drawing Area</b>		Brings up a popup menu		Zooms into the drawn rectangle
<b>Table of Documents</b>	The icon is marked in the area.	Opens the detail view of the file Header: Brings up a popup menu.	Opens the document in browser.	
<b>Detail Window</b>	Opens the detail view (group icons).	Brings up a popup menu (header).	-	-
<b>Icon</b>	The document is marked in the table.	Opens the detail view of the document	Zooms into group icons, opens documents in browser.	-

Table B.3: Mouse Functionality in the Search Result Explorer

# Bibliography

- [Ahl96] Christopher Ahlberg. Spotfire: An information exploration environment. *SIGMOD REPORT*, 25(4):25–29, December 1996.
- [And95] Keith Andrews. Visualising Cyberspace: Information visualisation in the Harmony internet browser. In *Proc. First IEEE Symposium on Information Visualization (Info-Vis'95)*, pages 97–104, Atlanta, Georgia, October 1995. <ftp://ftp.iicm.edu/pub/papers/ivis95.pdf>.
- [And96] Keith Andrews. *Browsing, Building, and Beholding Cyberspace: New Approaches to the Navigation, Construction, and Visualisation of Hypermedia on the Internet*. PhD thesis, Graz University of Technology, Austria, September 1996. <http://www.iicm.edu/keith-phd>.
- [ANS95] ANSI/NISO. Z39.50-1995 (Versions 2 and 3) Information Retrieval: Application Service Definition and Protocol Specification, 1995. <http://lcweb.loc.gov/z3950/agency/attrarch/attrarch.html>.
- [Bar98] Joe Barker. Search engines and subject directories teaching library internet workshops, 1998. <http://www.lib.berkeley.edu/>.
- [BDH<sup>+</sup>94] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, August 1994. <ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.ps>.
- [BF93] C. Beshers and S. Feiner. Autovisual: Rule-based design of interactive multivariate visualizations., 1993. <http://www.cs.columbia.edu/graphics/projects/AutoVisual/AutoVisual.html>.
- [Cha93] Matthew Chalmers. Visualisation of complex information. In *East-West International Conference on Human-Computer Interaction: Proceedings of the EWHCI'93*, volume 2 of *Information Visualization/Navigation*, pages 38–50, 1993.
- [Cha95a] Matthew Chalmers. Design perspectives in visualising complex information. In S. Spaccapietra and R. Jain, editors, *Proc. 3rd IFIP Visual Databases Conf., published as Visual Database Systems 3: Visual Information Management*, pages 103–111. Chapman and Hall, 1995.
- [Cha95b] Avijit Chatterjee. Parallel visual explorer at work in the money markets, 1995. <http://www.ibm.com/news/950203/pve-03.html>.

- [CIP96] Matthew Chalmers, Robert Ingram, and Christoph Pfranger. Adding imageability features to information displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Papers: Information Visualization, pages 33–39, 1996.
- [CMS98] Stuart Card, Jock MacKinlay, and Ben Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, 1998.
- [Cro95] David Crossley. WAIS through the Web — discovering environmental information. In *Second International World-Wide Web Conference: Mosaic and the Web, Chicago, IL, October 17–20, 1994*, Urbana, IL 61801, USA, 1995. National Center for Supercomputer Applications, University of Illinois at Urbana-Champaign.
- [CRO99] Crossgraphs: Multiply your Insights, 1999. <http://www.belmont.com>.
- [DB98] Deborah F. Swayne, Dianne Cook and Andreas Buja. XGobi: Interactive dynamic data visualization in the X Window System. *Journal of Computational and Graphical Statistics*, 7(1):113–130, March 1998.
- [dBvKOS97] Mark de Berg, Mark van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry Algorithms and Applications*. Springer-Verlag, Berlin Heidelberg, 1997.
- [DGK<sup>+</sup>99] Thomas Dietinger, Christian Gütl, Bernhard Knögler, Dietmar Neussl, and Klaus Schmaranz. Dynamic background libraries - new developments in distance education using HIKS (Hierarchical Interactive Knowledge System). May 1999. <http://www.iicm.edu/jucs>.
- [GAM98] Christian Gütl, Keith Andrews, and Hermann Maurer. Future Information Harvesting and Processing on the Web. In *Proc. European Telematics: Advancing the Information Society*, Barcelona, Spain, February 1998. <http://www.iicm.edu/~cguetl/papers/fihap/>.
- [GM96] James Gosling and Henry McGilton. The Java Language Environment. May 1996. <http://www.java.sun.com/docs/white/langenv>.
- [GPM99] Christian Gütl, Maja Pivec, and Hermann Maurer. An improved way for ongoing and lifelong learning as a smart module for the GENTLE learning environment. Graz, Austria, September 1999. <http://www.iicm.edu/~cguetl/papers/icce99/>.
- [HMV96] M. Z. Hasan, A. O. Mendelzon, and D. Vista. Applying database visualization to the World Wide Web. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(4), December 1996.
- [Hof96] Patrick Hoffman. Visualization Seminar, 1996. <http://ivpr1.cs.uml.edu/shootout/viz/vizsem/vizsem1.htm>.
- [Hol97] S. Holzner. *XML Complete*. McGraw\_Hill, 1997.
- [HP96] Marti A. Hearst and Jan O. Pedersen. Visualizing information retrieval results: A demonstration of the Tilebar interface. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, pages 394–395, 1996.
- [HYP99] Hyperwave homepage, 1999. <http://www.hyperwave.com>.

- [HZPA<sup>+</sup>96] Mountaz Hascoet-Zizi, Catherine Plaisant, Chris Ahlberg, Matthew Chalmers, Robert Korfhage, and Ramana Rao. Where is information visualization technology going? In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Panel, pages 75–77, 1996.
- [IEE] IEEE The Institute of Electrical and Electronics Engineers. <http://www.ieee.org>.
- [Ins98] Alfred Inselberg. A Survey of Parallel Coordinates. In Hans-Christian Hege and Konrad Polthier, editors, *Mathematical Visualization*, pages 167–179. Springer-Verlag, Heidelberg, 1998. Vismath97.
- [Jaw98] Jamie Jaworski. *Java 1.2 Unleashed*. Sams.net, Indianapolis, IN 46268, USA, fourth edition, may 1998.
- [Kap99] Dr. Frank Kappe. Hyperwave information server 5.0 technical white paper, 1999. <http://www.hyperwave.com>.
- [KG99] Brewster Kahle and Bruce Gilliat. Alexa internet homepage, 1999. <http://www.alexa.com>.
- [KK94] Daniel A. Keim and Hans-Peter Kriegel. VisDB: Database exploration using multi-dimensional visualization. *IEEE Computer Graphics and Applications*, pages 44–49, September 1994.
- [KK95] Daniel A. Keim and Hans-Peter Kriegel. VisDB: A system for visualizing large databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, page 482, 1995.
- [LOM99] Learning Object Metadata, 1999. <http://ltsc.ieee.org/doc/wg12/LOM3.7.html>.
- [LRB<sup>+</sup>97] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger. DEVis: Integrated querying and visual exploration of large datasets. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2), 1997.
- [Mau96] Hermann Maurer, editor. *HyperWave: The Next Generation Web Solution*. Addison-Wesley, May 1996. <http://www.iicm.edu/hwbook>.
- [MET99] Metadata and Resource Description. 1999. <http://www.w3c.org/Metadata>.
- [Mor97] Emile Morse. Document Visualization, 1997. <http://www.sis.pitt.edu/~elm2/DocumentVisualization.htm>.
- [NFF96] Lucy Terry Nowell, Robert K. France, and Edward A. Fox. Visualizing search results with Envision. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, System Demonstrations: Abstracts, pages 338–339, 1996.
- [NFH<sup>+</sup>96] Lucy Terry Nowell, Robert K. France, Deborah Hix, Lenwood S. Heath, and Edward A. Fox. Visualizing search results: Some alternatives to query-document similarity. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Visualization, pages 67–75, 1996.

- [OL96] Hwee-Leng Ong and Hing-Yan Lee. Software report: WinViz—a visual data analysis tool. *Computers & Graphics*, 20(1):83–84, January 1996. ISSN 0097-8493.
- [Ost99] James Ostell. Search Engine Watch, 1999. <http://www.searchenginewatch.com>.
- [RDF98] *Resource Description Framework (RDF) Model and Syntax Specification*, 1998. Available at <http://www.w3.org/TR/WD-rdf-syntax/>.
- [RHS97] Michael Reed, Dan Heller, and Dr. Ben Shneiderman. Online library of information visualization environments, 1997. <http://www.otal.umd.edu/Olive>.
- [Shn97] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing, third edition, 1997.
- [SPS98] SPSS Statistical Product and Service Solutions, 1998. <http://www.spss.com>.
- [Sul99] Danny Sullivan. Using ASN.1 (Abstract Syntax Notation 1): A data description language, 1999. <http://www.nalusda.gov/pgdic/Probe/v2n2/using.html>.
- [SWM<sup>+</sup>97] Karanjit Siyan, James L. Weaver, Jim Mathis, Luke Cassady-Dorion, and Tim Ritchey. *Inside Java*. New Riders Publishing, Carmel, IN, USA, March 1997.
- [TSDS96] Lisa Tweedie, Bob Spence, Huw Dawkes, and Hua Su. The Influence Explorer – a tool for design. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, pages 390–391, 1996.
- [VIS98] European Space Agency Visualization Laboratory, 1998. <http://www.estec.esa.nl/vislabs.html>.
- [W3C99] The World Wide Web Consortium. 1999. <http://www.w3c.org>.
- [WBT99] GENTLE approach to Web Based Training, 1999. <http://wbt.iicm.edu>.
- [WC98] Kathy Walrath and Mary Campione. *The JFC Swing Tutorial: A Guide to Constructing GUIs*. Addison-Wesley, Reading, MA, USA, 1998.
- [WC99] Kathy Walrath and Mary Campione. *The JFC Swing Tutorial*. 1999. <http://www.java.sun.com/books/tutorial>.
- [Wei98] Stuart Weibel. DC-5: The Helsinki Metadata Workshop; A report on the workshop and subsequent developments. Technical report, D-Lib Magazine, February 15, 1998.
- [WIN99] WinViz, a data visualisation product, 1999. <http://www.krdl.org.sg/Research/CurProj/WinViz/WinViz.html>.
- [xFI99] xFIND eXtended Framework for Information Discovery, 1999. <http://xfind.iicm.edu>.
- [XML99] eXtensible Markup Language, 1999. <http://www.w3.org/XML/>.
- [ZC87] Monica C. Zubritzky and Bruce G. Coury. Multidimensional scaling as a method for probing the conceptual structure of state categories: An individual differences analysis. In *Proceedings of the Human Factors Society 31st Annual Meeting*, Mental Models, Visual Displays and Complex Systems, pages 107–111, 1987.