10 Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Interactive Search Processes in Complex Work Situations

A Retrieval Framework

von Raiko Eckstein

UNIVERSITY OF BAMBERG PRESS

Schriften aus der Fakultät
Wirtschaftsinformatik und Angewandte Informatik
der Otto-Friedrich-Universität Bamberg

Schriften aus der Fakultät Wirtschaftsinformatik
und Angewandte Informatik

Band 10

University of Bamberg Press 2011

Interactive Search Processes in Complex Work Situations

A Retrieval Framework

von Raiko Eckstein

University of Bamberg Press 2011

Bibliographische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Informationen sind im Internet über http://dnb.ddb.de/ abrufbar

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität als Dissertation vorgelegen
1. Gutachter: Prof. Dr. Andreas Henrich
2. Gutachter: Prof. Dr. Wolfgang Benn
Tag der mündlichen Prüfung: 15. Juli 2011

Herstellung und Druck: Digital Print Group, Nürnberg
Umschlaggestaltung: Dezernat Kommunikation und Alumni

http://www.uni-bamberg.de/ubp/

ISSN: 1867-7401
ISBN: 978-3-86309-017-3 (Druckausgabe)
eISBN: 978-3-86309-018-0 (Online-Ausgabe)
URN:urn:nbn:de:bvb:473-opus-3326

It's not information overload. It's filter failure.

Clay Shirky

Acknowledgements

This thesis presents the results of my research as a member of the Chair of Media Informatics at the University of Bamberg, Germany. However, this thesis would not have been possible without the help of numerous people.

First and foremost, I would like to thank my thesis advisor Prof. Dr. Andreas Henrich for his enduring support, his guidance and his always helpful ideas that provided valuable input for my research. I also thank Prof. Dr. Ute Schmid and Prof. Dr. Elmar J. Sinz for their support as members of my thesis committee.

Furthermore, I would like to express my appreciations for the valuable discussions and inspirations that were part of my work at the Chair of Media Informatics, in particular, Adrian Hub, Daniel Blank, Nadine Weber, Stefanie Sieber, Tobias Fries, and Dr. Volker Lüdecke. Additionally, I thank Silvia Förtsch and Siegfried Hofmann for keeping the Chair running smoothly in the background.

Apart from work, my thanks go to my parents and sister for providing the sometimes necessary gentle motivating push to bring this journey to an end.

Munich, February 2011 Raiko Eckstein

Zusammenfassung

Seit einigen Jahren ist ein stetiges Ansteigen der Menge an Informationen, die in Unternehmen erzeugt werden, festzustellen. Um als Unternehmen wettbewerbsfähig zu bleiben, ist es notwendig, vorhandenes Wissen wiederzuverwenden, um aus vergangenen Projektergebnissen profitieren zu können. Weiterhin ist ein vollständiges Informationsbild unabdingbar, um informierte Entscheidungen treffen zu können. Die Informationsvielfalt in modernen Unternehmen übersteigt häufig die Fähigkeiten aktuell anzutreffender unternehmensweiter Suchlösungen. Die Gründe hierfür sind vielfältig und reichen von nicht verknüpften Informationen aus verschiedenen Softwaresystemen bis hin zu fehlenden Funktionen, um den Nutzer bei der Suche zu unterstützen. Vorhandene Suchfunktionen im Unternehmen unterstützen häufig nicht die Suchparadigmen, die in diesem Umfeld notwendig sind. Vielfach ist den Suchenden bei der Formulierung ihrer Suchanfrage nicht bekannt, welche Ergebnisse sie finden werden. Stattdessen steht der Aspekt des Wissensaufbaus und der Gewinnung neuer Einsichten in den vorhandenen Daten im Vordergrund. Hierzu werden Suchparadigmen benötigt, die dem Nutzer Werkzeuge zur Verfügung stellen, die ein exploratives Navigieren im Datenbestand erlauben und ihn bei der Erkennung von Zusammenhängen in den Suchergebnissen unterstützen.

Das Ziel dieser Arbeit ist die Vorstellung eines Rahmenwerks, dass explorative Suchvorhaben im Unternehmensumfeld unterstützt. Das beschriebene LFRP-Framework baut auf vier Säulen auf.

1. Die Multi-Layer Funktionalität erlaubt es Nutzern, komplexe Suchanfragen zu formulieren, die sich auf mehr als einen Ergebnistyp beziehen. Dies ermöglicht beispielsweise Suchabfragen, die – ausgehend von einer Menge von relevanten vergangenen Projekten – Selektionen auf den dazugehörigen Dokumenten erlauben.

2. Das Suchparadigma der facettierten Suche unterstützt Nutzer bei der inkrementellen Formulierung von Suchanfragen mithilfe von dynamisch angebotenen Filterkriterien und vermeidet leere Ergebnismengen durch die Bereitstellung gültiger Filterkriterien.

3. Die Erweiterung der facettierten Suche um die Möglichkeit, die Suchergebnisreihenfolge basierend auf Filterkriterien zu beeinflussen, erlaubt es Nutzern feingranular vorzugeben, welche Kriterienausprägungen im Suchergebnis stärker gewichtet werden sollen. Für den Nutzer geschieht die Beeinflussung des Rankings transparent über sogenannte Nutzerpräferenzfunktionen.

4. Die letzte Säule umfasst die Visualisierung mit parallelen Koordinaten, die in der Suchoberfläche des LFRP-Frameworks zwei Aufgaben übernimmt. Zum einen formuliert der Nutzer damit die Suchanfrage ausschließlich grafisch über die Visualisierung und zum anderen erhält er eine grafische Repräsentation der Suchergebnisse und kann so leichter Beziehungen zwischen Suchergebnissen und deren Facetten erkennen.

Das Framework, welches in dieser Arbeit formal aus Sicht des Anfragemodells sowie als prototypische Umsetzung betrachtet wird, ermöglicht Nutzern den navigierenden Zugriff auf große vernetze Datenbestände und stellt einen Baustein einer umfassenden Informationsstrategie für Unternehmen dar.

Contents

List of Figures

List of Tables

Listings

Part I

Introduction to the Subject

Chapter 1

Introduction

In 1945, Vannevar Bush introduced his vision of the *memex*—short for **Mem**ory **Ex**tender—in the article "As We May Think" [Bush, 1945]. He envisioned the memex as "a device in which an individual stores all his books, records and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility". He proposes a mechanical device which supports users in storing information and helping them to retrieve the information at a later point in time. Users are supported by the concept of *associative trails* which are linear sequences of different knowledge items that are linked together. By organizing information this way users shall easily be able to follow trails of related knowledge.

In modern organizations, a multitude of information and knowledge is created and stored during the development of products and services which often is considered a competitive advantage. To retain competitiveness and innovative capability, organizations begin to realize the potential of reusing existing knowledge from the organization. Therefore, organizations strive for solutions which on the one hand help to capture the produced information and on the other hand support users to retrieve this information when necessary with the help of methods from the field of knowledge management[1]. This reuse of information can help preventing duplicate work as well as support users in coming to better decisions.

Transferring the idea of Bush to organizations, the memex would be represented by a single, self-contained system which captures the existing information of an organization and associates related concepts and entities. At the same time, the memex would abstract from the multitude of application systems. Users can access this organizational memory and easily search and access information whenever necessary. Although, individual parts of this vision are currently available as applicable products, still, large parts are lacking for its realization.

The focus of this publication lies in the analysis of interactive search processes in complex work situations which occur in enterprise scenarios and in the proposition of an approach which supports knowledge workers in retrieving existing information and knowledge in an organization. A special emphasis is put on search situations occurring in product development processes where several examples are based on. The portrayed application and information landscape is characterized by

[1] For additional information refer to [Nonaka and Takeuchi, 1995] and [Alavi and Leidner, 2001].

a variety of different applications which store various types of information ranging from structured to unstructured information.

This publication embraces the idea of bringing users and search user interfaces closer together by providing a search framework which helps users to satisfy complex information needs easier. Although, current search engines in the web focus on providing users with a single search box from which they can enter the keywords that describe their information needs, this publication advances the view that providing users with a search interface which adapts to different types of information is more helpful than common keyword searches. This approach allows users to explore the available information and discover new information which can be helpful for their current work tasks.

1.1 Motivation

In knowledge-intensive working environments a large fraction of work time is spent on searching for internal and external information and knowledge. The exact amount of time varies based on the tasks and domains which are considered. For instance for the field of engineering, developers are spending between 20 and 30% of the complete development time on the retrieval of information (e. g. [Beitz et al., 1997] and [Marsh, 1997]), but being only successful in about 50% of their searches [Feldman, 2004]. del Rey-Chamorro and Wallace [2005] predict that the amount is likely to increase in the future, as technology evolves and the complexity of product development further increases. The improvement of search engines for domains alike can help to improve the successful retrieval of information in two ways. First, the amount of time necessary to find information can be reduced which directly helps shortening development times which is a crucial requirement for companies to stay competitive. Second, search tasks can be more effective and efficient when users are given a central search engine which federates information that is spread across several application systems. This approach might not shorten search times, but can help to improve the quality of the search by supporting users better in finding the relevant information.

According to Leszinski [2001] the field of product development can be characterized by growing product complexity which originates from the increasing use of mechatronic components in the products and dynamic customer requirements [Schichtel, 2002]. Complex products consist of an increasing number of components and relationships among each other. Additionally, products are designed for customization to satisfy individual customer needs. To stay competitive, companies try to shorten product development processes to get products ready for the market.

An approach to achieve shorter development times is to encourage users to reuse existing information, parts and products from the organization. This reuse helps users to make better decisions by considering (documented) experiences from past projects. Additionally, duplicate work is avoided which shortens development times and reduces development costs. This manifests not only in the reduced time to develop a specific part, but also by reducing duplicate product tests, which might

already be done in the original project where the product was designed. Information which cannot be found or is found to late might lead to wrong decisions during the development of products, and thus is decreasing productivity. In general, reuse can lead to reduced product costs based on more efficient warehousing and optimized lot sizes. Additionally, the complexity and the risk of new designs can be reduced by depending on existing and established parts.

Successful project management captures the experiences after the project execution in form of best practices/lessons learnt which can be materialized in the form of project reviews. The retrieval of these experiences can help users to avoid mistakes and to apply successful methods in their current projects. Additionally, potential problem fields of the current product development can be identified prematurely.

Various studies of engineers' information seeking behavior found that *accessibility* is the factor that influences their selections of information sources most (for instance [Fidel and Green, 2004]). Based on that finding, search engines should aim for improving the availability and accessibility of information by providing an entry point to the various systems available in common system landscapes.

The process of developing new products can be characterized by very creative tasks which often draw on existing designs as a basis for the inspiration for new developments. Sarkar and Chakrabarti [2007] emphasize how important search and exploration is during design since these activities enable idea generation capabilities of engineering designers. The retrieved solutions can be used directly or modified iteratively to be consistent with the current requirements. These activities occur during the whole development process and should be supported by search engines which allows the exploration of information spaces.

The diverse and complex information needs found in organizations, the available variety of places to search as well as the complexity resulting from the fragmented application landscape complicate the successful retrieval of information. By providing users with search tools that support query refinement in a dialog-like style (as propagated by Human-Computer Information Retrieval (HCIR) introduced below in Section 2.1 on page 11) users get instant feedback how changes to their queries would affect the intermediate search results. This approach helps users to better explore the information landscape by providing additional insights about the search results such as highlighting correlations and dependencies. By integrating structured and unstructured data about knowledge items interesting new applications can be provided which include data analysis functionality known e. g. from the Business Intelligence (BI) domain. The complexity of the information landscape due to the manifold entities which are stored in various management systems can be reduced by this kind of tools.

1.2 Problem Statement

As outlined above, organizations can strongly benefit from reusing existing information. The domain of enterprise search (which will be described in detail in Sections 2.6 on page 40 and 2.7 on page 53) tries to provide solutions for search prob-

lems in organizations which are different from those in the web. Both academia and industry are proposing various solutions for the specific search problems existing in organizations.

Application landscapes in companies can be characterized as heterogeneous and complex. Often many different systems are used to store the manifold information which is created during the development of products or services the company produces. Often redundant systems exist in organizations because of grown company structures resulting from acquisitions and mergers after which the applications systems were not consolidated.

Considering companies from the domain of product development, a variety of artifacts is produced describing the different aspects of a product. In addition to geometric information in two and three-dimensional form, various test reports and calculations are created during the development. The descriptions are both stored in multiple document types as well as in various document formats which complicates searching due to the necessary support for all types and formats in the search engine. The storage of these result documents can take place in various forms ranging from the plain storage on network shares to sophisticated management systems which allow the storage in addition to descriptive data such as in Product Data Management (PDM) or in Product Lifecycle Management (PLM) systems. This heterogeneity leads to multiple access points when users need to find data about certain products. Thus, they are forced to search in several systems to find the necessary information. Although, there exist approaches in the PDM field trying to provide users with comprehensive information about products, these solutions lack the inclusion of other information necessary in product development, such as project or expert information. Furthermore, the existing search functions which are included in applications for the product development domain often resort to parametric searches. This kind of search requires users to state a complete query by giving the criteria based on multiple text fields or drop down boxes. Often these types of queries lead to empty result sets which can result in abandoned queries. To overcome this restriction, Belkin [1993] already proposed in 1993 that information retrieval should focus more on information-seeking activities, i. e. ways how users interact with information for a better support and better search results by stronger integrating users in the search process.

Another difficulty is introduced as information is not always stored in a structured form which would allow more precise and concrete searches. Much information is only stored in unstructured form such as in textual documents. Although humans might be able to understand the structure of a document, automatic processing of this structure during the indexing step can be difficult and often information about the structure is lost.

In addition to various types of documents which are created to describe information from different viewpoints, additional entity types can be identified which become relevant in certain work situations for users. Entity types can for instance be *projects*, *persons*, or certain *materials*. Each of these entity types and their different subtypes can be described by a specific set of different descriptive attributes. A search engine trying to meet these challenges would first need to extract all infor-

mation from the different systems, transform it in a homogeneous representation and eliminate duplicate information before indexing the data. Second, the identified wealth of information needs to be provided to users in a way that allows them to explore the entities by using the available entity descriptions as search criteria.

This variety of information types can lead to complex search situations when no appropriate search support is provided. This includes the mapping of these descriptions to comprehensible search user interfaces. Additionally, information needs vary from being clearly defined to being very vague. The former can be answered comparatively simple by a known-item search where users know what search result they seek. However, the latter situation is characterized by users not being able to exactly describe their information need. In situation like that, users should be supported in the query formulation process. For instance, search user interfaces can help users by providing relevant search criteria and insights about the search results. Additionally, this is helpful in situations where information needs are changing during the search task, i. e. situations where the goal of the search task evolves during the exploration of the data.

Another challenge lies in the relationships which exist between the different entity types which should be exploitable by users. For instance, in product development projects users might want to find all projects in which certain products were used or created. A scenario for this type of an information need could be the search for documents which describe best practices for the development of a specific product group. A search task then would start with the search for all products adhering to specific search criteria. Without a search engine supporting these different related entities, users would have to search all projects for the products which they identified in the first step. The difficulty for this type of queries is the necessary data quality. In addition to the description of the entities, data quality is concerned with the relationships between the entities which need to be curated.

Especially in the domain of product development, similarity searches on product data are an interesting approach to find relevant products. During the development of products, 3D representations are created. In situations where users need a certain product which adheres to a certain form (i. e. its geometry) a similarity search should be invoked where users provide a sketch. In addition to this (geometrical) criterion, users might want to restrict the search results by additional search criteria such as product features. Currently available search engines for the domain of product development often only allow the query for geometrical similarity. An example for a geometric search engine is the product *Geolus Search*[2].

1.3 Goals of the Publication

The goals of this publication can be derived by examining the difficulties and problems currently existing in knowledge-driven organizations and are described below:

[2] `http://www.plm.automation.siemens.com/en_us/products/open/geolus/index.shtml` (last accessed (08/01/10)

- **Description of the information landscape in organizations.** In contrast to information found on the web, the information landscape in organizations is characterized by a variety of different information sources providing different information types. Information often is managed wide-spread in different application systems which introduce additional complexity when users need to search for information and have to query different systems to satisfy their information need.
- **Description of complex search situations.** This publications describes search situations which can occur in enterprise scenarios with a focus on the domain of product development. Especially search tasks with a high complexity are described and reasons for this complexity are assessed. Amongst others this comprises the existence of search situations where users are not able to describe their information need clearly, but only have a vague understanding of what kind of search result they need. Additionally, the availability of various entity types in the search result complicate the retrieval of information. Search engines which are deployed in environments like this must show an increased awareness for these kinds of difficulties to support users appropriately.
- **Improvement of the supply of information.** Modern enterprises should strongly try to reuse existing information to benefit from past experiences and their knowledge base to stay competitive. Often the insufficient search solutions currently available in organizations represent an obstacle for the successful supply of information. The availability of comprehensive search solutions which consider all available information across various systems is often limited. Thus, users need to access multiple systems which they have to query for the same need which can get frustrating and can lead to abandoned search tasks.
- **Definition of a search framework for complex search situations.** The characteristics of the information landscape in organizations demand specific solutions for search engines which help users to access the available information easily. This publication introduces a search framework which provides various search features customized for the difficulties found in the portrayed domains and overcomes restrictions of currently available search engines. The framework especially focuses on providing users with an interactive user interface which allows them to incrementally formulate a query. Additionally, it allows searching over a data collection consisting of different entity types derived from various application systems.

  - **Provide a formal query model for complex search situations.** The analysis of the difficulties of information retrieval in enterprise settings leads to the definition of a formal query model which is capable to represent user queries and therewith their information needs. This query model describes the available query options of the framework in a formal way. By representing it in an SQL-like form, expert users are able to understand the scope of the query model.

– **Propose a search user interface for complex search situations.** This publication proposes a search user interface implementation for the realization of the formal query model. Usually, end users do not come into contact with the formal query model, but are given a graphical representation which allow them to state queries in a visual way. At the same time users are presented with previews helping them to understand how different search criteria would affect the search result. The graphical representation of the query and the search results aim for a better provision of insights about the data collection, for instance relationships between different items in the search results. The user interface includes the statement of similarity searches as well as options to influence the search result ranking based on visual representations of so-called user preference functions.

The scope of this publication is the conception and the description of the general search framework with a strong focus on the query side. This comprises the definition of the query model with all available features and the description of the search user interface which realizes the introduced query functionality. For a complete definition of a search engine additional functionalities need to be described that are out of scope of this publication. Amongst others, these comprise the explanation of how the indexing engine operates including efficient index structures as well as integration aspects into application landscapes. The latter would describe the federation of data by using interfaces on how to extract information from multiple systems and how to consolidate the information to achieve an accurate description of an entity which might be constructed based on information from multiple systems. For the described work, a certain quality standard of the indexed data is assumed. This approach is chosen to introduce the necessary functionality to improve the supply of information. Nevertheless, this publication considers solutions for situations where the data quality is below the assumed level in the respective parts of this work. Additionally, performance aspects are only considered on a shallow level.

1.4 Organization of the Thesis

This work is organized in two main parts. The first part Introduction to the Subject lays the groundwork for this publication by introducing background information and basic concepts in Chapter 2 on page 11 comprising search paradigms such as exploratory and faceted search which are picked up later in the conception of the search framework as well as forms of visualizations which can be used for the presentation of queries and search results. Furthermore, this chapter provides an overview of related work for the topic of enterprise search and concludes with an introduction of several exemplary enterprise search engines currently available on the market. It is followed by Chapter 3 on page 69 where both information-seeking behaviours of engineers as well as search scenarios typical for enterprise search are introduced.

The second part of this publication covers the description of the retrieval model for complex search situations called *Multi-Layer Faceted Search with Ranking using Parallel Coordinates (LFRP)-search framework*. Chapter 4 on page 91 focuses on the description of the formal model including definitions of the used terminology such as *artifacts* and *facets*.

Chapter 5 on page 127 describes the prototypical realization of the concept described in the chapter before. Section 5.1 on page 127 provides an introduction of the visualization type of parallel coordinates which represents one constituent part of the framework. It is followed by the description of the architecture of the framework in Section 5.2 on page 131 where the different necessary buildings blocks are linked together. The main part of this section is found in Section 5.3 on page 134 which describes the search user interface which realizes the LFRP-query model. After the introduction of the control options of the user interface, the focus lies on the usage of parallel coordinates as a visual tool for search query formulation and at the same time as a visualization of the search results. Additionally, the presentation of the search results as well as the handling of complex queries which range over multiple artifact types is described. The chapter closes with a comparison of concepts from Data Warehouse (DWH) systems with those of the LFRP-search framework.

Chapter 6 on page 157 provides an evaluation of the introduced framework with regard to the compliance to concepts of the field of HCIR. Additionally, the search user interface is contrasted with certain established rules of search user interface design coined by Shneiderman et al. [1997].

The publication closes with a summary of the presented work and provides an outlook of additional research aspects.

1.5 Origins of the Material

Parts of the work described in this publication were accomplished while researching in the joint research project FORFLOW[3] consisting of six Bavarian universities working in the fields of engineering design and computer science. It collaborated with 21 companies and is promoted by the Bavarian Research Foundation (Bayerische Forschungsstiftung BFS). The primary aim of the research project was the development of the concept for a *Process Navigator* that guides engineers through the product development process and gives assistance in decision-making[4]. This is achieved by providing engineers methods and information which is useful for their current work tasks. The sub-project "Context-based search for reusable components" especially examined solutions which help users to cope with the large heterogeneity of the information landscape. The results of the research project are summarized in the final report published in [Meerkamm et al., 2009][5].

[3] `http://www.forflow.org` (last accessed 09/01/2010)

[4] `http://www.bayfor.org/en/portfolio/research-cooperations/world-of-material/forflow.html` (last accessed 09/01/2010)

[5] Also available as an e-book at `http://www.shaker.de/shop/978-3-8322-8640-8`

Chapter 2

Background and Basic Concepts

In this chapter the groundwork is laid out for the main part of this thesis by providing background knowledge and related work necessary for the concepts described in Part II on page 91. Sections 2.1 and 2.2 on page 15 outline the challenges of HCIR and detail the aspect of exploratory searches. Section 2.3 on page 18 introduces the search paradigm of Faceted Search. Exemplary search systems are described in Section 2.4 on page 24.

Section 2.5 on page 33 focuses on existing approaches which use visualizations in search user interfaces. Both visualizations for the query formulation as well as for the result presentation are covered.

Section 2.6 on page 40 illustrates problems and challenges for the field of information access technologies with a focus on enterprise search and gives a characterization. It is followed by an introduction of several (commercial) systems which aim for providing enterprise search solutions in Section 2.7 on page 53.

2.1 Human–Computer Information Retrieval

Much of the Information Retrieval (IR) research focuses on the classic retrieval approach of matching a query with the documents in the search engine index to determine a ranking of search results for the given query. Many search engines pursue the paradigm that users can paraphrase their information need in form of some keywords. The search engine then assumes a certain relevancy model and determines a ranked list of search results. As early as in 1964, Goffman [1964] found that "the relationship between the document and the query, though necessary, is not sufficient to determine relevance". In other words, the query does not contain enough information for the search engine to reliably determine how relevant a document is for the user's information need.

Borlund [2003a] gives a thorough overview of the research of the concept of *relevance* and points out that no consensus has been yet reached on the relevance concept due to its multidimensionality.

Schamber et al. [1990] draw three central conclusions about the concept of relevance:

- "Relevance is a multidimensional cognitive concept whose meaning is largely dependent on users' perceptions of information and their own information need situations."
- "Relevance is a dynamic concept that depends on users' judgments of quality of the relationship between information and information need at a certain point in time."
- "Relevance is a complex but systematic and measurable concept if approached conceptually and operationally from the user's perspective." [Schamber et al., 1990]

Traditional search engines often produce a relevance ranking based on several criteria of the documents which are then reduced to a single score based on a proprietary formula which defines relevancy[1]. But from a user viewpoint, the information that a document is for instance 45.7% relevant is hard to interpret. Thus, users lack the understanding why a document appears in a certain position in the ranking.

These findings show that it is beneficial to allow users to stronger influence the relevance measures than it is currently possible in many search engines. Certain information needs not only require the retrieval of relevant information but additionally its understanding and connections between this information. Therefore, users should be given the opportunity to clarify what is relevant to them through query refinements.

In recent years, IR research more and more embraced approaches that bring human intelligence more actively in the search process. Researchers began to blend findings from the Human-Computer Interaction (HCI) and the IR field to create new kinds of search systems which rely on continuous human control of the search process. Marchionini [2006b] coined the term HCIR for this hybrid approach. HCIR should support people in exploring large information spaces but at the same time demands more responsibility for this support by expending cognitive efforts. This is achieved by providing advanced search user interfaces and by improving the understanding of search strategies of users in different domains.

This paradigm shift is also partially rooted in the change of the information which needs to be retrieved. Contents are not anymore only text-based, but also include multimedia and structured parts adding additional complexity. Thus, more sophisticated search engines, especially with emphasis on the human interaction part, should be conceived to consider these changes in the information landscape.

As a result of this combination of research fields, users are seen as *active humans* with *information needs* and *information skills* and powerful digital resources situated in global or locally connected *communities* [Marchionini, 2006a]. This conception of HCIR demands search systems with the following basic requirements according to Marchionini [2006b]:

- Search systems should not only get their users closer to the information they need, but they should provide tools for *making meaning of the results* in addition to the delivery.

[1]The score is also called Retrieval-Status-Value (RSV).

- Search systems should delegate responsibility as well as control to the user; thus *requiring more human intellectual effort* which is then rewarded by *improved search results*.
- Search systems should have *flexible architectures* to easily be able to access the connected information sources.
- Search systems should *support the entire information life cycle* from the creation to the dissemination of the information.

The goal of these search systems should be that users are interacting more closely with the contained information than it is possible with current search systems. Thus, users need to be provided with several views of the same information which supports understanding as well as retrieval. The grand challenge of HCIR design lies in making the IR part transparent to the users so that they can focus on how information matches their needs and how they can apply or transform this information. This additional responsibility of the users necessitates the raising of user literacy in terms of user involvement.

Marchionini [2006b] also points out the challenge of evaluation of HCIR efforts. The IR measures of *Recall* and *Precision*[2] and the HCI measures of time to completion and general satisfaction are not easily applied to assess information interaction. Belkin et al. [2009] proposes a model for the evaluation of interactive IR based on the criterion of *usefulness* applied to the complete search task and its subtasks for achieving the overall goal. Borlund [2003b] also proposes an evaluation model especially designed for interactive IR as an alternative to the Cranfield model [Cleverdon et al., 1966a,b; Cleverdon and Keen, 1966] which itself relies on controlled experiments to evaluate retrieval performance which are difficult to apply in interactive settings. This is especially significant in use cases where the information need evolves during a search session due to newly discovered information.

Tunkelang [2009] proposes three goals a search engine has to adhere according to the HCIR paradigm: *Transparency*, *Control* and *Guidance*.

The *Transparency* of a search engine should ensure that users understand why the search engine returned a particular response to a search query. The emphasis here lies on the "why" rather than the "how" since users are usually interested whether the search engine understood their query and attempts to address the users' information need they expressed with their query. Users are not interested in the underlying relevancy measures and algorithms the search engine applies.

By knowing why the search engine returned a certain result, users can try to adapt to the "limited cognitive capabilities" of the search engine and establish a more effective communication with it.

When users see that the search engine misunderstood them, they need *control* to give the engine the necessary information to express their actual information

[2] *Recall* and *Precision* are two statistical classifications used widely in IR evaluation. *Recall* is a measure of completeness and is defined as the ratio between the *number of relevant documents retrieved* and the *total number of relevant documents*. On the other hand, *precision* defines the ratio between *the number of relevant documents retrieved* and the *total number of documents retrieved* [Baeza-Yates and Ribeiro-Neto, 2008].

need and to adjust the query. For instance, Boolean search engines offer much control to users as they can build arbitrarily complex search queries, but their lack of guidance (see below) leads to degraded user acceptance. According to [Turtle, 1994] most users—even expert searchers—are not adept at constructing boolean queries, especially when they are querying unfamiliar content.

Control involves that users can influence the filtering as well as the ranking of search results which makes them more active in the information-seeking process than before. Although, *control* adds more options for users to influence their search process, on the downside, additional complexity for the users is introduced.

To counteract this complexity, search engines should provide users with *Guidance*. A search engine should not only respond to users' queries, but also help them formulate (complex) queries. This requirement can be based on the different human information-seeking behaviors.

In 1989, Bates proposed the "berrypicking" model of information retrieval, which is based on the assumption that typical queries are not static, but rather evolve during the search process. Information needs are not satisfied by a single search result set, but usually are answered by consecutive queries which gather information in bits and pieces. When users find an interesting or important piece of information, they can "pick" this information and start a new query based on these findings. Therefore, searchers apply a wide variety of search techniques and might need to access a wide variety of sources [Bates, 1989].

Another approach to understand the human perception of information retrieval is the *Information Foraging* theory developed by Pirolli and Card [1999]. The authors apply an analogy of the information seeking process to the way our animal ancestors found food. The main concept of this theory is the "information scent" which guides "informavores"[3] to the next step of their information-seeking process. Users estimate how useful the found information is for their information need and decide whether they want to follow this lead or whether turn to other search techniques or information sources.

Conveying these theories to search engines following the HCIR search paradigm means that these search engines need to provide users with guidance (a form of an "information scent") at every step in the information seeking process. Techniques for guidance can include query suggestions, relevance feedback[4], faceted search, visualizations, result clustering, and others. These techniques prevent situations where users would have to restart the information seeking process by offering them the refinement of their query to adjust it to their information needs. Both Belkin [2008] and Saracevic [2007] emphasize the necessity of more interaction in information retrieval systems to cater users with more efficient solutions to satisfy their information needs.

[3]The term was coined by George A. Miller in [Machlup and Mansfield, 1984] and characterizes an information consuming organism.

[4]Relevance feedback is based on implicit or explicit user feedback about search results which then is used by the search engine to improve subsequent searches [Salton and Buckley, 1990]. Refer to [Ruthven and Lalmas, 2003] for an overview of relevance feedback for information access systems.

Nordlie [1999] examined searchers using different online catalogs to conduct complex search tasks. As a result of this study, Nordlie suggests a more conversational-style search process which should incrementally reveal the real intentions of the searcher which is termed by the author as “user revealment”.

In summary, search engines adhering to HCIR should strive for an optimized communication with users in contrast to “guessing” the user intent based on automated, difficult to comprehend algorithms. In other words, a dialog between users and the data should be established. By understanding that for certain information needs, search processes consist of more than a single query, search engine features such as query refinement, exploration and discovery become more important than the current focus on the top ten documents in the result list.

2.2 Exploratory Search

One of the aspects of HCIR is the field of *exploratory search* which describes certain search activities from the class of *information seeking*. White and Roth [2008] define exploratory search based on the description of Marchionini [2006a]:

> “Exploratory search can be used to describe an information-seeking problem context that is open-ended, persistent, and multi-faceted; and to describe information-seeking processes that are opportunistic, iterative, and multi-tactical. In the first sense, exploratory search is commonly used in scientific discovery, learning, and decision-making contexts. In the second sense, exploratory tactics are used in all manner of information seeking and reflect seeker preferences and experience as much as the goal.” [Marchionini, 2006a]

Marchionini proposed a differentiation of search activities called *lookup*, *learn*, and *investigate* as seen in Figure 2.1 on the next page. He separates lookup searches from those associated with exploratory search processes. Lookup searches usually aim at retrieving a single answer as for instance in fact retrieval or in known-item searches [Reitz, 2004]. Lookup searches return discrete and well-structured objects and can for instance be found in database queries. They usually follow the simple “query-response” paradigm where a search engine returns a set or ranked list of results for a given query and demand only minimal need for result set examination and item comparison.

The activities *learn* and *investigate* associated with exploratory searches demand more user involvement in the query statement process in terms of interaction and query refinement—users should be brought more actively into the search process. By examining and comparing results as well as refining and reformulating the query to discover new information, users are acquiring new knowledge.

Since exploratory search sessions usually span over more than one search query and are highly interactive, Exploratory Search Systems (ESSs) must provide the ability to specify information needs in the form of search queries and allow to refine

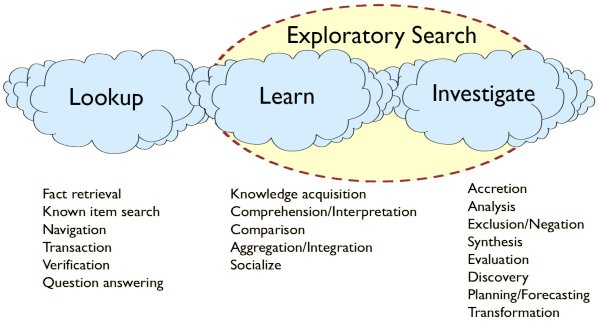


Figure 2.1: Search Activities according to Marchionini [2006a].

these during the search session. The navigational search for information is also denoted as *Browsing* which is "[...] the activity of engaging in a series of glimpses, each of which exposes the browser to objects of potential interest; depending on interest, the browser may or may not examine more closely one or more of the (physical or represented) objects."[5] [Bates, 2007]. Marchionini and Shneiderman [1988] define *Browsing* as "an exploratory, information-seeking strategy that depends on serendipity. It is especially appropriate for ill-defined problems and for exploring new task domains."

Many (more complex) information needs cannot be satisfied by a single query, but require successive refinement of the query. Studies in the field of user behavior showed that users usually prefer a search approach which consists of successive refinements of their query [Spink et al., 2002].

Although relevance feedback [Ruthven and Lalmas, 2003] is a powerful approach to improve and refine queries, it is up to the users to execute the revised query. To engage users in the query refinement process, interactive user interfaces are necessary which continuously encourage them to provide additional information about their information need. *Dynamic query* interfaces aim at providing instant responses to changes in the user query by providing certain graphic tools to adjust queries. By changing search criteria visually using slider adjustments and brushing techniques which are executed on the client-side, users can immediately assess the results of these changes to the query. Shneiderman [1994] characterizes these dynamic queries as a way to apply direct manipulation to the database environment which is characterized by the following principles:

- "visual presentation of the query's components;
- visual presentation of results;
- rapid, incremental, and reversible control of the query;

[5]Here the browser refers to the user and not to the application of a web browser.

- selection by pointing, not typing; and
- immediate and continuous feedback." [Shneiderman, 1994]

Plaisant et al. [1999] emphasize the feature of *query previews* for query interfaces of search systems. By providing summary data about the search results users are guided through the query statement process. The query previews provide an overview of the search results from different perspectives. Query previews provide several advantages for users:

- Queries which would lead to zero results are omitted.
- The retrieval of undesired records can easily be omitted by filtering the summaries.
- The comprehension and exploration of the search results can be enhanced for users by representing statistical information of the dataset visually.
- Dynamic queries can be used to help users discover data patterns and exceptions.
- Query previews are suitable to novice, intermittent, or frequent users.
- Query previews can help users to recognize certain search criteria rather than remembering them.

Exploratory search systems should also consider available contextual information about users, their situation and their current exploratory search task to help them formulating their queries. Especially work tasks have significant effects on people's search performance and behavior [Vakkari, 1999]. Exploratory search tasks usually are less structured and searchers lack the knowledge of the topic and have a poorer conceptualization of the problem. Thus, task complexity has to be taken into account when considering context in exploratory searches [Byström and Järvelin, 1995].

To support learning and understanding it is beneficial for users that ESS offer support for search histories and workspaces. For instance, these can be realized by providing mechanisms to save interim search queries and results for later evaluation. This can be useful if a search process is interrupted and has to be resumed later.

Shneiderman et al. proposed eight golden rules of interface design in [Shneiderman and Plaisant, 2005] which have been rephrased in [Shneiderman et al., 1997] for the context of information retrieval. A search user interface should have the following characteristics:

- **Strive for consistency.** User interfaces should be consistent in the use of the terminology, the instructions, and the layout including colors and fonts.
- **Provide shortcuts for skilled users.** All functionality should be reachable easily, for instance by providing keyboard shortcuts.

- **Offer informative feedback.** Users need to be informed about all aspects of the search process including the used sources, query fields, etc. After the search is complete, users should be aware of what was searched and why the search results were returned.
- **Design for closure.** Users should easily notice when they searched the complete data set or the result list.
- **Offer simple error handling.** When errors occur during the search process users should be presented with comprehensible support explaining the occurred error. Additionally, changes to the query should be easy to apply.
- **Permit easy reversal of actions.** User interfaces should give users the ability to reverse single actions in their search process, for instance by allowing them the revert selections by accessing a search query history.
- **Support user control.** Users should be able to specify queries in any order. Thus, a search engine should not force users to follow a strict sequence of steps during query refinement. Users can also get additional control over the search by being presented visual overviews over the data set. This helps to gain a better understanding of the data and make more controlled search query refinements.
- **Reduce short-term memory load.** Search interfaces should be designed that users always are shown the relevant information avoiding situations where they need to remember or keep track of it.

2.3 Faceted Search

The search paradigm *Faceted Search* supports exploratory searches by providing tools to filter search results of multi-dimensional information spaces. The filtering is conducted on so-called *facets* which are descriptive attributes of each search result and provide users with guidance and control of the search process.

Hearst [2006b] defines goals which should be considered during the design of a faceted search systems. These systems should

- support the flexible navigation of the dataset,
- integrate seamlessly with directed (keyword) search,
- alternate fluently between refining and expanding the search query,
- avoid empty results sets and
- at all times retain a feeling of control and understanding.

A faceted search user interface provides users with flexible navigation over the attributes of a domain and offers query previews which empower users to interactively refine their queries for a better matching with their information need. Kules et al. [2009] conducted a study where they examined how searchers interact with search engines when conducting exploratory search tasks. The main results are that

facets help users to gain an overview over the data set and provide guidance on which sub-topics they should focus on for the current search task. Faceted search engines help users to explore search results more broadly than without faceted categories while feeling more organized and less lost in their searches [Kules and Shneiderman, 2008].

Käki [2005] presents a study showing that users make use of categorized overviews when they are provided with them by the search engine. Especially, when the ranking of search results does not provide the expected results, the selection of descriptive categories helped users to filter the search result for vague, broad, or general queries.

2.3.1 Definition of Faceted Categories

Faceted search relies on the existence of faceted categories which are organized in a category system describing a set of meaningful labels organized in such a way as to reflect the concepts relevant to a domain [Hearst, 2006a].

Well-known category systems were developed in library science to make documents or books retrievable in a (physical) library. For instance, the Dewey Decimal Classification (DDC) originally developed by Melvil Dewey in 1876 [Dewey, 1876] helps to organize books in library shelves in a repeatable manner and ensures short paths between topically similar books. The DDC spans a taxonomy tree which defines categories for each publication. Each publication is assigned to one specific node in the taxonomy tree which resembles one specific shelf in the library. The DDC built the groundwork for the Universal Decimal Classification (UDC) which is more expressive but also more complex. Additions to the DDC include relationships between subjects.

In general, these classification systems span a tree where each child node has one parent, i. e. each node in the tree has one unique path to the root node. This may pose a problem, since each object is assigned to only one node which is too rigid. Taking for instance a book about Database programming in Java, it might be assigned to the Database section or the Java Programming section. To overcome this restriction of single hierarchy taxonomies, S. R. Ranganathan—being a mathematician and a librarian—introduced the *colon classification* scheme [Ranganathan, 1933]. His goal was to introduce a notation which could accommodate a general class of compound subjects. Thus, Ranganathan introduced the first ever library classification based on facet analysis. He used the term *isolates* for an independent category which nowadays is referred to as a *facet*.

The classification of a compound subject is denoted as a sequence of letters and numbers separated by colons. Ranganathan gives an example of a specific subject representing the "Statistical study of the radium treatment of cancer in soft palate" in [Ranganathan, 1950]. This subject is represented as L2153:4725:63129:B28. The four elements between the colons describe this subject. Here, the exemplary elements *L2153* and *4725* represent the following hierarchical facets:

- Medicine (L) → Digestive System (L2) → Mouth (L21) → Palate (L215) →

Soft Palate (L2153)

- Disease (4) → Structural Disease(47)→ Tumor (472) → Cancer (4725)

In combination the cancer of soft palate is described, which shows the independent combination of facet hierarchies which are not possible in single hierarchical taxonomies.

Taylor [1992] describes facets as "clearly defined, mutually exclusive, and collectively exhaustive aspects, properties or characteristics of a class or specific subject". These orthogonal sets of categories can be used together to describe a subject [Hearst, 2000].

In this publication a *facet* describes an orthogonal category of an item which is assigned a name and a set of possible *facet values*[6] it can be assigned to. Each of these facets are orthogonal to each other, i. e. they are independent of each other and freely assignable to an item.

The idea of decomposing compound subjects by the application of faceted categories also was picked up in the area of knowledge representation. Hearst [2006a] refers to these decomposed subjects as Hierarchical Faceted Categories (HFC) which consist of creating a set of category hierarchies where each corresponds to a different facet. Each facet is associated with a set of possible values (mostly terms). An individual facet can be *flat* ("authored by Person A") or *hierarchical* [Yee et al., 2003] where an example for the latter are geographical references such as the facet "produced in" which could contain a facet value for a certain dimension such as city, country or continent (e. g. Bamberg $>$ Germany $>$ Europe). Furthermore, the cardinality of a facet must be considered. *Single-valued* facets allow only one facet value assignment for a document, whereas *multi-valued* facets allow multiple facet values for one facet per document. An example for the former is the facet *document type* where only one (unique) value is permitted. An example for the latter case can be the authors of a document which might consist of multiple persons who are responsible for the document.

Although mostly text-based facets are found in faceted search engines, an interesting approach is taken by Müller et al. [2008] for the domain of Content-Based Image Retrieval (CBIR). The authors extracted visual features of images such as the dominant color of a region or the coarseness level and used them as facets in the *VisualFlamenco* prototype.

2.3.2 Creation of Faceted Categories

A thorough overview about the field of faceted classifications and (manual) facet analysis can be found in [Vickery, 1960] and [Vickery, 1966]. As the manual creation of faceted categories is obviously a huge effort, several research approaches try to (semi-)automatically determine these facet hierarchies and assign objects to nodes in these hierarchies.

[6] In the literature the term *facet label* is also found and here regarded synonymously to *facet value*.

Early approaches to automatically determine categories for a set of documents mainly relied on *clustering* techniques [Zamir and Etzioni, 1999; Meila and Heckerman, 2001; Zeng et al., 2004]. Clustering describes the task of grouping documents by a certain measure of similarity among certain features of the documents such as words and phrases automatically. The main advantage of clustering approaches is the possibility to fully automate the cluster computation. The research of the Scatter/Gather system showed that a cluster-based approach supports browsing of large (text) document collections [Cutting et al., 1992]. Initially, the Scatter/Gather system *scatters* the document collection into a small number of document groups (the clusters) and presents them to the users along with short summaries. Then, users can choose a number of these clusters which are then *gathered* as a new "subcollection" and scattered again into new clusters based on this subcollection. However, the resulting labels of the applied automatic clustering technique are typically very long consisting of a set of keywords resulting in category titles such as "battery california technology mile state recharge impact official hour cost government" [Hearst and Pedersen, 1996]. Although this provides users with insight about the different determined clusters of documents, it is difficult for presentation in a user interface. Other disadvantages of clustering methods are the lack of predictability of the resulting clusters, the conflation of many dimensions simultaneously and the counterintuitiveness of cluster subhierarchies [Hearst, 2006a]. Additionally, clustering techniques usually show labels whose terms are associated with one another instead of showing hierarchical parent-child relationships. Several studies indicate that users mostly prefer the navigation with hierarchical categories over associational clusters [Chen et al., 1998; Pratt et al., 1999].

Several approaches try to automatically determine useful hierarchical faceted categories which are outlined below.

[Stoica and Hearst, 2004] and [Stoica et al., 2007] present the Castanet algorithm which provides an approach for automatically creating hierarchical faceted metadata structures. Castanet converts the lexical hierarchy provided by WordNet[7] into an appropriate concept hierarchy using a tree-minimization algorithm. A study found that this approach delivers higher quality results than other automated category creation algorithms.

Sanderson and Croft [1999] propose a method for building a category hierarchy for a set of documents called *subsumption*. For two terms x and y, x subsumes y and is a parent of y if the documents which y occurs in are a subset of the documents which x occurs in. The authors express this by the following condition for x subsuming y $P(x|y) \geq 0.8, P(y|x) < 1$, whereas the probability of 0.8 was chosen through information analysis of subsumption term pairs.

In two related papers, Dakka et al. introduce an approach for automatically constructing multifaceted hierarchies from a large collection of text or text-annotated objects. Their approach includes the discovery of useful facets and their possible values, as well as an algorithm for efficient hierarchy construction based on lexical

[7] A publicly available lexical database for English. In WordNet words are organized into synsets (synonym sets) which are linked by different relations [Fellbaum, 1999]. (`http://wordnet.princeton.edu/` last accessed 08/01/2010)

subsumption [Dakka et al., 2005, 2006]. The algorithm to identify useful facets relies on WordNet hypernyms [Fellbaum, 1999] and a Support Vector Machine (SVM) classifier which assigns keywords to facets.

2.3.3 Description of Faceted Search

The search paradigm of *Faceted Search* can be defined as the combination of Faceted Navigation and text/keyword searches. The understanding of faceted search in this publication will later be broadened to include—in addition to text searches—Query-by-Example (QbE) queries with faceted navigation for easier filtering. For instance, users should be able to query for similar products based on 3D geometric similarity (cf. Section 4.3 on page 100). Thus, the search paradigm is applied to a more general means to support searches which induce a ranking.

Faceted Navigation is an improvement over parametric searches by providing additional *guidance* (cf. Section 2.1 on page 11). Parametric search interfaces usually offer users the possibility to create Boolean queries by choosing several filter criteria based on a faceted collection. The huge disadvantage is that users have to specify the whole query at once and therefore might conduct searches which lead to empty result sets.

Figure 2.2 on the facing page shows an exemplary parametric search user interface where users can specify queries for searches of used cars. The example shows a variety of applicable search criteria, but users have to provide their complete query before getting any results.

Different studies showed that many users have strong misconceptions about Boolean operations in search tasks [Michard, 1982; Greene et al., 1990; Young and Shneiderman, 1993; Muramatsu and Pratt, 2001]. The identified difficulties of users with Boolean queries are related to the counterintuitive syntax and the misconception about the logical operators conjunction (logical AND) and disjunction (logical OR). Many users expect the search query of “Germany and France” to deliver documents dealing with Germany and documents dealing with France, but not documents dealing with both countries. With strict boolean search engines users did not comprehend why the search result did not have any specific order. When search queries resulted in empty result sets for a given query, many users could not explain the reason since they expected a wider query by applying the conjunction.

The concept of faceted search tries to avoid these pitfalls of Boolean searches (and thus, parametric searches). Several usability studies showed that hierarchical faceted categories can provide a flexible and intuitive approach to access and explore large semi-structured data collections [Hearst et al., 2002]. With a proper presentation in faceted user interfaces, users are guided through the data collection without getting the feeling of being lost [Yee et al., 2003]. A four-week longitudinal study which investigated the usage of exploratory and keyword search forms showed that exploratory forms of search are used as often as keyword searches by users [Wilson and schraefel, 2008]. Additionally, users were often able to produce more rich and expressive queries with exploratory approaches than with plain keyword searches or boolean queries.

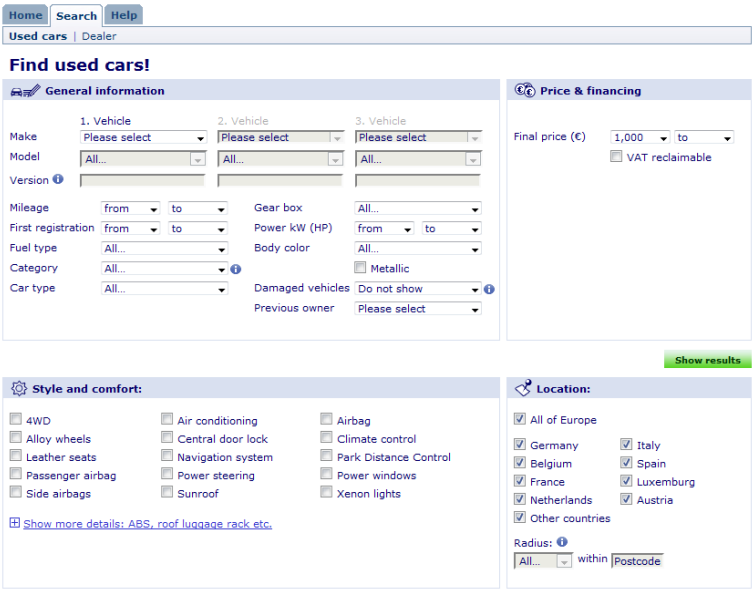


Figure 2.2: Example of a parametric search user interface from AutoScout24 (`http://www.autoscout24.eu/Search.aspx` last accessed 08/01/2010).

With faceted search, queries are not given in one step, but refined iteratively, i. e. users can add and remove search criteria incrementally to refine the query to match their information need. Since the faceted categories are orthogonal to each other, they can be arbitrarily combined in a query which allows for flexible access to the contents of the collection. Users incrementally narrow the interim search results by adding another filter criterion to the query.

Faceted search engines provide users with guidance to support their query statement. Users are guided in the query refinement process as they are shown query previews which denote the facet values of each facet and the amount of search results which would remain, if they select this facet value. Facet values for which no objects exist in the current search result set are not shown to users to prevent queries which would yield empty result sets. Furthermore, the set of available facets from which users can conduct selections is adjusted after each refinement step, i. e. users are presented search criteria which are relevant for their current query. Taking the example of searching in an online store, where users select the product group of digital cameras, available facets might be the sensor size, the maximum aperture of the lens, and the optical zoom. Obviously, these facets are not applicable when searching for a TV set where other facets will be offered to users.

The order of the different facet selections conducted by users is insignificant compared to the search in a strict hierarchy. Thus, users are unrestricted which facets they choose and do not need to know all associated facet values of the result object. Assuming a strict hierarchy, it would be necessary to follow the unique path through the hierarchy to retrieve the sought search result. Selections of different facets and associated values are combined by the conjunction (logical AND) to filter the search results. When users select a sub-hierarchy of a hierarchical facet, the different values are combined by a disjunction (logical OR). Considering a geographical facet which supports the layers continent, country, and state, a selection of the country Germany, would include all 16 states (such as Bavaria, Saxony, etc.) in the filtering.

Taking all the described features of faceted search together, this search paradigm provides an intuitive means to help users to navigate multi-dimensional information spaces without being overwhelmed due to the inherent complexity. Thus, faceted search interfaces can be found in a wide variety of domains such as media centers (e. g. iTunes[8]), photo management software (e. g. Adobe Photoshop Lightroom[9]) and others.

2.4 Examples for Exploratory Search Systems

In this section several systems are introduced which allow users to search datasets of different domains in an exploratory fashion. The actual systems were chosen to provide a wide overview of available exploratory approaches and at the same time incorporating the faceted search paradigm.

2.4.1 Flamenco

The group around Prof. Marti Hearst presents a faceted search system called Flamenco[10] which was developed in a research project at the University of California, Berkeley [Hearst, 2000]. The search system is built on a relational database and a web application which can be viewed from a web browser. The used data set must be provided in a special text format which can be read by Flamenco and is then stored in the database according to a generic database schema. All parts of the user interface views are dynamically generated based on Structured Query Language (SQL) database queries. Later, the group made the Flamenco faceted search system available as an open-source project.

The search process in the Flamenco prototype consists of three stages, entitled *Opening*, *Middle Game* and *Endgame* as a rough analogy to the game of chess [Yee et al., 2003].

[8] `http://www.apple.com/itunes/` (last accessed 08/01/2010)

[9] `http://www.adobe.com/products/photoshoplightroom/` (last accessed 08/01/2010)

[10] short for **Fl**exible information **A**ccess using **Me**tadata in **N**ovel **Co**mbinations (`http://flamenco.berkeley.edu/` last accessed 08/01/2010)

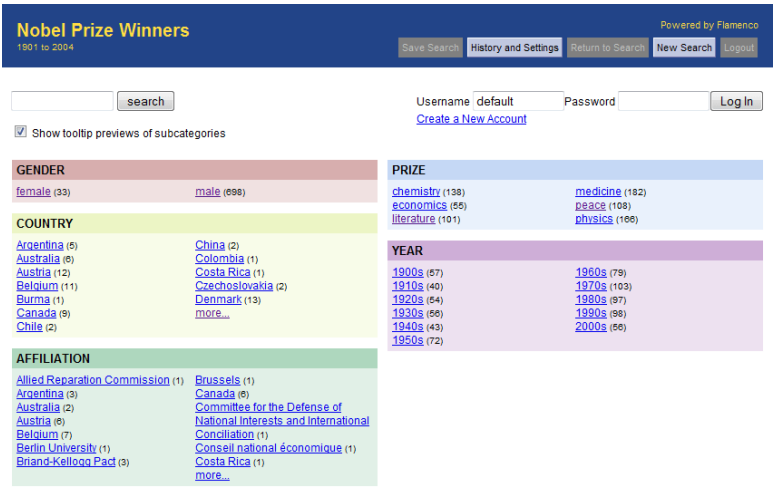


Figure 2.3: Example for a search for Nobel Prize laureates from 1901 to 2004 with Flamenco: The Opening [Flamenco Search Interface Project, 2007].

The *opening* mainly provides users with a thorough overview of the data in the collection and offers entry points for exploratory searches. The opening page shows each available facet aggregated at the highest hierarchy level, and thus already offers various navigation options and helps users to familiarize with the organization of the data collection. In addition to the selection of filter criteria by choosing facet categories, users can conduct a keyword query by entering text in the search field. Figure 2.3 shows the user interface in its initial state in the opening stage of a search for Nobel Prize laureates from 1901 to 2004. The selection of a facet category or the provision of search terms initiates the *Middle Game*.

In this stage, users can evaluate and manipulate the result set. Usually, this includes the narrowing of the search result by adding additional filter criteria. The user interface is divided into three main parts. The largest area contains the result set and shows a preview picture of the Nobel Prize laureate data set. The left area in the user interface contains the available facets with their values and the query previews which denote the number of search results that would remain in the result set when users select that facet value. These facet values can be used to refine the current search query. Additionally, users can limit the result set by providing keywords in the available text search field, which then is based on the current interim result. The top part contains the composition of the current query.

To further understand the data set, users are given different possibilities to organize the search results. The organization can be adjusted by sorting the items in the result list on various fields (such as name, year of birth and year of death) or by

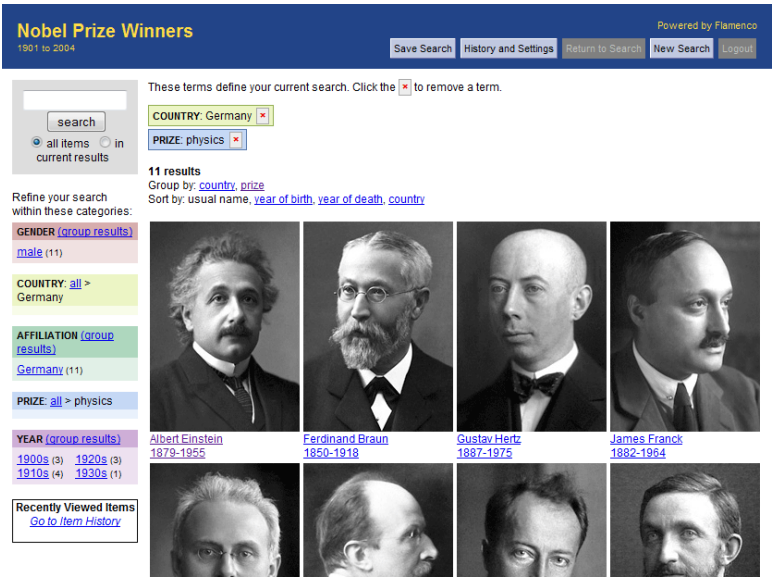


Figure 2.4: Example for a search for Nobel Prize laureates from 1901 to 2004 with Flamenco: The Middle Game [Flamenco Search Interface Project, 2007].

grouping the results into categories by a facet (e. g. country or prize). When users select a value of a hierarchical facet such as *location*, the search results are automatically grouped by the subcategories, i. e. when the location "Europe" is chosen, the results are grouped by countries, such as France, Italy, Germany, and so on. Figure 2.4 shows the result when users choose "Germany" and "physics" of the facets "Country" and "Prize" in the Middle Game. When users remove a facet category from the current query the search results are broadened again. The selection of a single item in the search results leads users to the endgame.

The *endgame* as seen in Figure 2.5 on the next page shows detailed information of the chosen search result in the context of the current search query. The available metadata facets are displayed in a tabular view which shows the location of each facet category in the according hierarchy of that facet. This is realized by a hybrid-tree layout that shows all metadata terms of the current item and their locations in the hierarchical facets. Users can start new searches from this stage by clicking on one of the facet categories in the result view. This functionality allows users to navigate to an item of interest and from there search for similar items based on similar metadata categories.

Based on an assessment of Hearst et al. [2002], Flamenco supports six out of the above mentioned eight design desiderata of Shneiderman et al. The system is

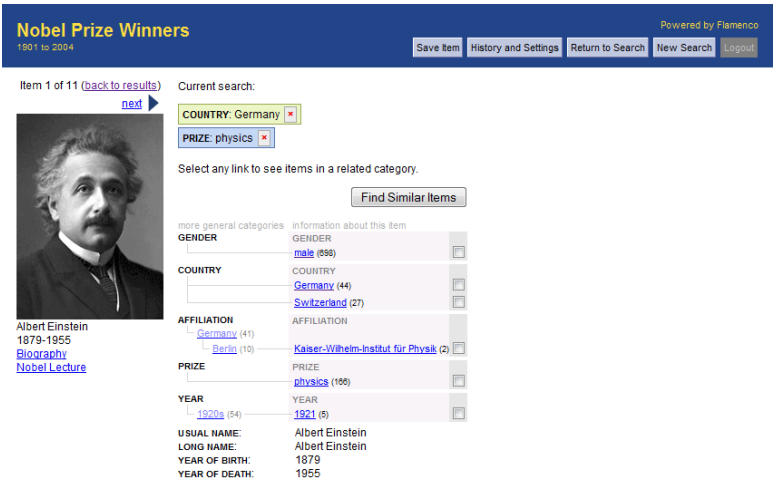


Figure 2.5: Example for a search for Nobel Prize laureates from 1901 to 2004 with Flamenco: The Endgame [Flamenco Search Interface Project, 2007].

consistent in regard to the layout throughout the whole search process and continuously gives feedback by showing the current query state. Each user action is reversible as the system supports the removal of a single facet selection and the start of a new search. As the entire query state is maintained in the Uniform Resource Locator (URL), the use of the back button of the browser is allowed and bookmark features can be used to save intermediate queries. This helps users to stay in control of the search process. The provision of query previews supports users in finding logical but perhaps unexpected alternatives for further selections and emphasizes recognition over recall during a search. Thus, the query previews help to lower the short-term memory load. This also provides users with an information scent (cf. Section 2.1 on page 11) which leads to their next steps. Hearst [2006b] gives additional design recommendations for hierarchical faceted search interfaces which were derived by studying several revisions of the Flamenco user interface.

Müller et al. [2008] enhanced the Flamenco search engine to be applicable for CBIR where they used visual features of images as facets and represented them in an adapted Flamenco user interface.

2.4.2 mSpace Explorer

The mSpace project from the University of Southampton defines an interaction model [McGuffin and schraefel, 2004] and a software framework [Harris et al., 2004] to support the exploration and access of information. schraefel et al. [2006]

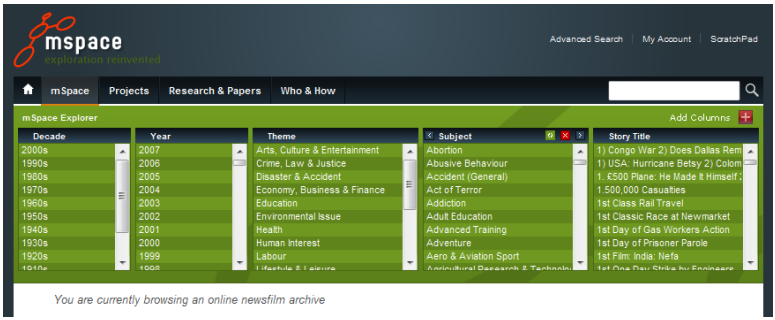


Figure 2.6: The initial view of the directional column-faceted browser mSpace for a search in an online newsfilm archive.

describe the mSpace prototype as a "multicolumn-faceted spatial browser" [11].

In the mSpace framework information of a domain is conceptualized in a multi-dimensional representation from which only a subset is represented in the user interface at a time. This subset is referred to as a set of "slices" where users can execute "slice and dice" operations [Marchionini, 1995]. The slices are dynamic and can be altered to fit the current information need of the users, i. e. dimensions can be rearranged, added or subtracted which helps users to organize the domain data according to their needs.

According to McGuffin and schraefel [2004] mSpace is an approach to visualize and interact with multidimensional or multivariate data. The structure of an mSpace is a special kind of polyarchy which itself is a structure composed of multiple intersecting hierarchies [Robertson et al., 2002].

mSpace is an example for a directional column-faceted browser where facets (the dimensions from above) are aligned in columns from left to right in the user interface[12] in contrast to non-directional faceted browsers such as Flamenco. Figure 2.6 shows the initial view of the mSpace user interface for a dataset from an online newsfilm archive. The currently displayed facets are *Decade*, *Year*, *Theme*, *Subject*, and *Story Title*. Users can remove unnecessary facets from the view and add other facets to the user interface. Additionally, users can swap the columns to rearrange slices and access new relations between the facets.

The selection of facet filter criteria only has an affect on the facets to the right. If users conduct selections in a middle-column facet, only the facets to the right are constrained by this selection. Figure 2.7 on page 30 shows selections on two different facets. The user chose the *theme* "Human Interest" and the *subject* "Aero

[11] The prototype is available at `http://www.mSpace.fm` and can be viewed and tested online (last accessed 08/01/2010).

[12] Another common directional column-faceted browser is the digital media library iTunes (`http://www.apple.com/itunes/` last accessed 08/01/2010) where users for instance can easily filter their music by Genres, Artists and Albums.

& Aviation Sport" which restricted the search to 63 results which are displayed below in a paginated list view. The facet selections are highlighted in orange in the according facet column. mSpace allows multiple selections on one facet which corresponds to a disjunction of facet values of one facet. After each selection the user interface is updated to reflect the current state of the search result. Thus, facet values of facets positioned to the right from the facet the selection took place, are limited to those values currently valid for the selection.

Although this directional faceting already provides information about additional inter-facet relationships, the potential information in the columns to the left is given away. To this end, Wilson et al. [2008] propose *backward highlighting*, which describes an intuitive way to reveal associations in columns to the left of a selection. Backward highlighting can be seen in Figure 2.7 on the next page where facet values of facets left from the current selection are highlighted in a brown color to show these associations to the selections. Users can easily see, that under the current selections there are no items in the search result from the 1900s and 1960s. Wilson et al. [2008] calls this information which is usually lost in non-directional faceted search interfaces "Added facts". Users are given indications of alternative paths in the collection they could have chosen to reach the sought information by these highlighted associations. This additional guidance helps users to refine their search query by facets located left from the current selections which would not have been easily achievable without this highlighting.

The current version of mSpace (at the time of writing in 2010) omitted the display of numerical query previews since Wilson and schraefel [2006] claim that currently the mental models for including them in flexible faceted browsers is not completely understood from a research viewpoint.

For standard users which are not familiar with the data collection, mSpace support *preview cues*, which are a lightweight mechanism to assist exploration of multimedia content[13]. This mechanism provides a multimedia preview of the information associated with a certain area which helps users to assess if a certain area of the domain is of interest for further exploration. For the domain of music, preview cues could be musical snippets relevant to a certain musical era such as Baroque which are triggered when users hover over the cues [schraefel et al., 2003, 2004]. The goal is to provide these cues early in the search process at the category level rather than at the instance level. schraefel et al. [2003] found that these preview cues help users to explore a domain and make better filter decisions.

2.4.3 Relation Browser

The Relation Browser[14] from the University of North Carolina is an ongoing effort to provide a general-purpose search interface which can be applied to different data sets [Marchionini and Brunk, 2003]. The prototype underwent several different

[13] The preview cues of mSpace are not to be confused with the numerical query previews found for instance in the Flamenco user interface (cf. Section 2.4.1 on page 24.

[14] `http://ils.unc.edu/relationbrowser/` (last accessed 08/01/2010)

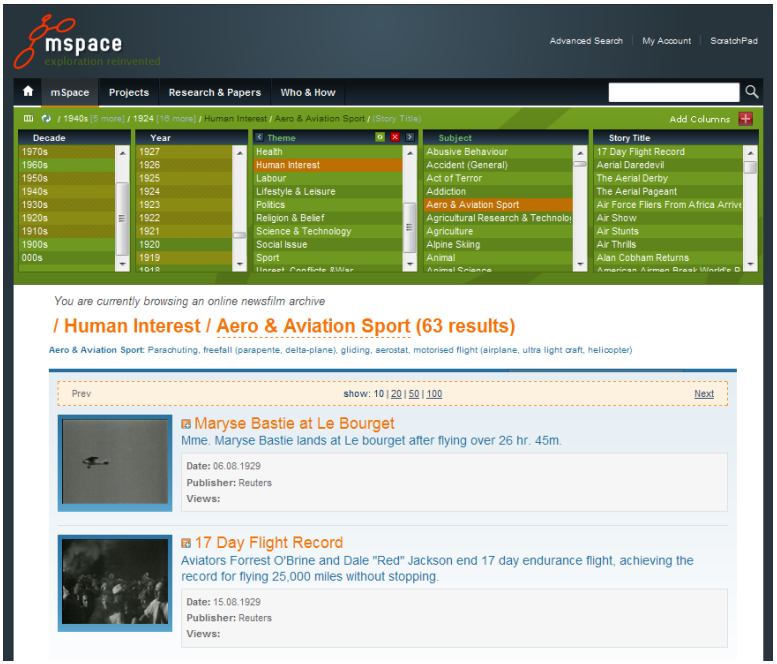


Figure 2.7: The view of the directional column-faceted browser mSpace for a search in an online newsfilm archive with selections for the facet Theme and Subject.

stages. The introduced version of the Relation Browser is the latest version called RB07.

The Relation Browser aims at providing support for the exploration of data spaces by giving users a better understanding about documents and insights about relationships between or among different facets of these documents. Almost all user interaction with the prototype is done by mouse actions with the exception of being able to provide text for keyword searches. This allows both searching and browsing of a dataset by applying dynamic queries.

Figure 2.8 on the next page shows a screenshot of the Relation Browser during the query process. Below the input box for keyword queries, the currently selected facet values are shown (in the example “HTML”, “Mathematics” and “science”) in the order the users conducted the selections. In this query representation users can remove parts of the query. Additionally, the different partial queries are highlighted in red in other areas of the user interface. User selections and exploration of facets is done in the center of the prototype in the facet view. The Relation Browser supports

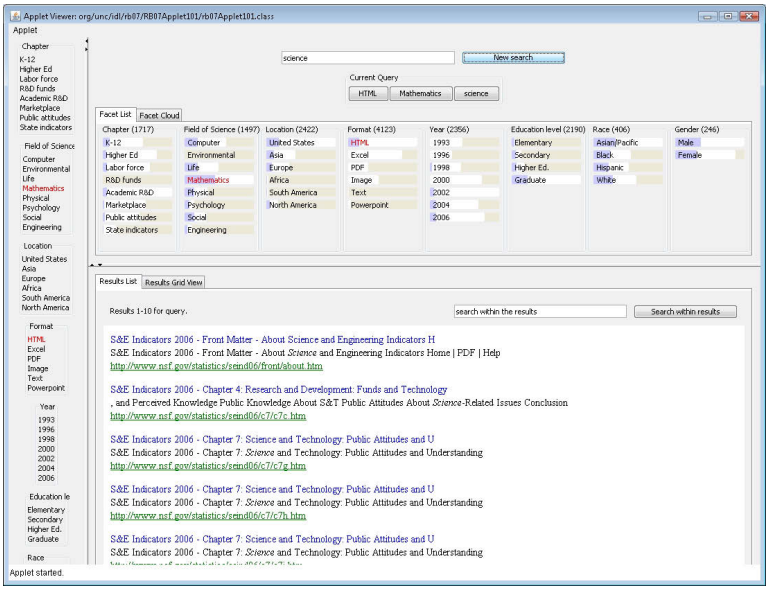


Figure 2.8: Relation Browser 07 (`http://www.ieee-tcdl.org/Bulletin/v5n1/Capra/img/rb07.jpg` last accessed 08/01/2010).

multiple, pluggable facet views which users can switch by using the tabs in the prototype. The classic facet visualizations (also present in previous versions) is the *Facet List*. The facets are shown horizontally next to each other according to the directional-column faceted style.

Each facet column uses graphic bars to show relative proportions between the facet values. This acts as an indication of the counts associated with the different facet values of a facet [Capra et al., 2007]. Without any facet selections this bar graph visualization gives an overview for the distribution of facets for the whole data set. By hovering over a facet value, users can instantly see how the selection of the respective facet value would affect the facet distribution for the resulting set of documents. Selections of facet values can be simply executed by clicking on the corresponding facet value and the user interface is instantly updated to reflect the change of the (dynamic) query.

The more recent addition is the *Facet Cloud* view [Capra and Marchionini, 2008] that displays facets in the style of a tag cloud view where facet values are shown in different font sizes to give an indication of the corresponding number of documents. In the leftmost area of the prototype users can display static facet controls which allow them to add facet selections which is especially useful when interacting with

the facet cloud visualization.

The search result presentation is located below the facet view where users can choose from two different types. Users can display the results in a common one-dimensional list view or choose to see the results in a tabular view which displays facets in addition to a document summary.

Several user studies showed the Relation Browser can support users to accomplish data exploration and analysis tasks and is advantageous in comparison to classic keyword based search interfaces [Zhang and Marchionini, 2005; Capra et al., 2007].

2.4.4 Freebase Parallax

The Freebase Parallax project[15] provides a user interface which allows users to query the Freebase database. Freebase describes itself as "an open, Creative Commons licensed repository of structured data of more then 12 million entities"[16]. Freebase connects these entities together in a large entity graph structure. Parallax supports users in querying this graph and provides insights by visualizing the connected data. Taking an example information need such as trying to find out which schools the children of the Republican US presidents did attend is relatively difficult to satisfy using common Internet search engines or by querying Wikipedia[17].

Figure 2.9 on the facing page shows the Freebase Parallax user interface after several selections over related entities. The query started with querying for US presidents which were then filtered by their political affiliation which reduced the number of presidents from 43 to 18. The left column in the user interface provides facets as filters for the current set of entities. In the upper right corner, users can select related entities from the violet box with provides the currently available relations. This allows users to browse the dataset from one search result set of entities to another set of entities. One of the provided relations is *children* which returns a set of 61 persons based on the previous set of 18 presidents. The choice of the educational institution relationship finally returns the information the user was looking for consisting of 35 schools. Parallax supports several visualizations such as map or timeline visualizations which give additional insights into the search results.

This approach relies on the modeled dependencies between different artifacts, which need to be available in the data of Freebase. This is one of the downsides of Freebase as often necessary information is not populated due to the wide scope of the database.

[15] For a demonstration of the prototype refer to `http://www.freebase.com/labs/parallax/` (last accessed 08/01/2010) and for an introductory screencast to `http://vimeo.com/1513562` (last accessed 08/01/2010).
Freebase Parallax is provided as an open source project which can be found at `http://code.google.com/p/freebase-parallax/` (last accessed 08/01/2010)

[16] `http://wiki.freebase.com/wiki/What_is_Freebase%3F` (last accessed 08/01/2010)

[17] `http://www.wikipedia.org/` (last accessed 08/01/2010)

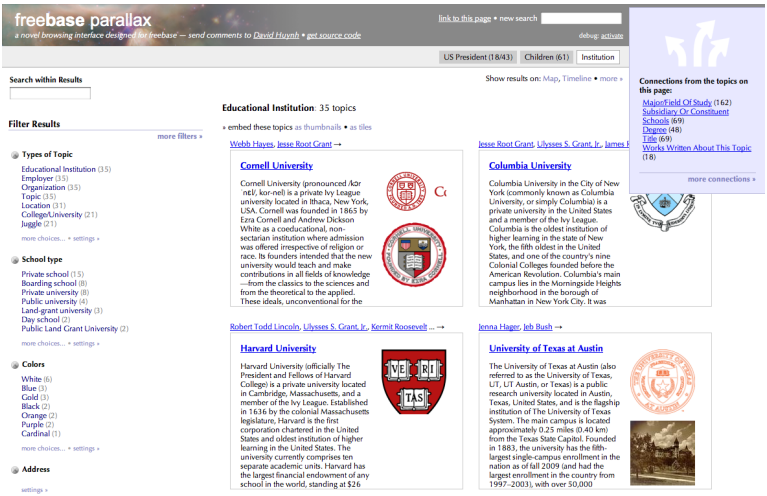


Figure 2.9: Freebase Parallax.

2.5 Visualization in Information Retrieval

This section introduces the use of visualizations in search user interfaces. A visualization can target two different use cases. First, the user query can be visualized to help users to state queries more easily. Second, the understanding of the search results can be improved by visualizing the search results (or some of their descriptive attributes). This distinction can be blurry when visualizations show both the query and the search results. By interacting with the visualizations users can analyze the search results and conduct additional selections to refine their queries. An example for this twofold functionality is the parallel coordinates visualization introduced in Section 5.3 on page 134 which shows the current query as well as a visual representation of the search results.

Visualizations help to translate abstract information in a form which provides users with new insights. The field of information visualization introduces a widespread spectrum of various visualizations forms and guidelines on how to design them [Tufte, 1983, 1990].

The use of visualizations in user interfaces should always be justified by clear purpose which should be achieved by providing a visualization. Sutcliffe et al. [2000] tested the acceptance and usage of visualizations in search user interfaces and found that users had only partially understood the purpose and the working of the visualization. The authors came to the conclusion that user training is important for the success of visualizations in user interfaces. This is necessary to provide users

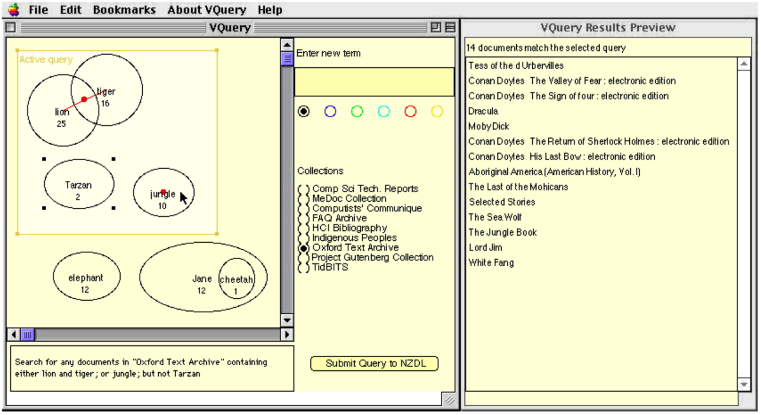


Figure 2.10: The Venn diagram visualization for queries in the VQuery system [Jones, 1998].

with an understanding of the used metaphors and functionality of the visualization.

2.5.1 Visual Query Formulation

As mentioned above, the use of Boolean operations for conducting queries has proven difficult for some user groups. Thus, several approaches exist which try to reduce the misconceptions about Boolean operators by using visualizations for the query statement.

An approach to specify Boolean queries visually is proposed by Jones [1998] with the *VQuery system*. It uses Venn diagrams to visualize query terms and the combination operators. Query terms are visualized by circles and can contain single keywords or phrases. Each circle represents the set of documents which contain the respective query term or phrase. Each circle is labeled with the keyword it represents and the size of the document set. The Boolean conjunction of query terms is conducted by placing the circles so that they overlap as seen in Figure 2.10 for the term *lion* and *tiger*. The disjunction of query terms can be achieved by adding the respective circles (or conjunctions of terms) to the query area. The term *jungle* is combined by a disjunction to the conjunction of *lion* and *tiger*. Query terms which are not allowed in the search results (logical NOT) are added to the query area and selected. The selection is shown in Figure 2.10) by the four small squares around the circle.

For the selection of the final query users can draw a rectangle around the desired circles (depicted as “Active query” in Figure 2.10). Terms which are added to the visualization but lie outside that rectangle are not considered for the current query, but allow the easy refinement of the query by moving them into the “active query”

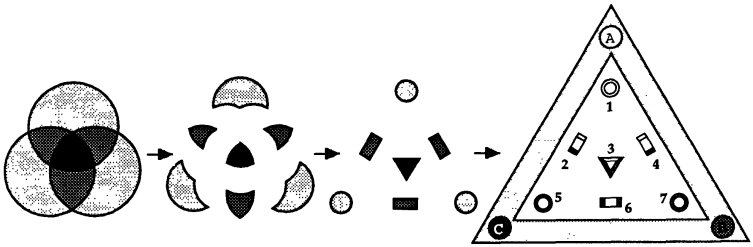


Figure 2.11: Transformation of the subsets of a Venn diagram into the InfoCrystal Visualization [Spoerri, 1993].

area. The current query is also presented in form of a natural language preview below the Venn diagram visualization.

A disadvantage of the intersection of sets in Venn diagrams is the difficulty to combine more than three different search criteria in an easy and visually comprehensible way. Spoerri [1993] introduces the InfoCrystal™ framework to overcome this limitation. InfoCrystal is a visual tool for information retrieval & management. It can be used as a visualization as well as a visual query language. The language supports Boolean as well as vector-space queries visually.

The InfoCrystal framework aims for a two-dimensional representation of the multidimensional space spanned by the multiple criteria of a search query. Spoerri abstracted from the Venn diagram visualization which shows intersecting sets. Figure 2.11 shows this transformation from the Venn diagram to the InfoCrystal visualization. In the first step the diagram is exploded to isolate the different subsets of the Venn diagram. The resulting subsets are abstracted by icons reflecting the rank of a subset, i. e. the number of criteria that are met. The InfoCrystal then contains these iconic representations of the subsets which are surrounded by a frame built of n borders representing the n search criteria. Each corner visually represents the original sets also called criterion icons.

Figure 2.12 on the following page shows an example for the textual query "visual query languages for retrieving information and that consider human factor issues". This query can be represented by the keyword sub-queries (Visual OR Graphical), Information Retrieval, Query Language, and Human Factors. The corners of the quadratic InfoCrystal represent the number of documents which adhere to one of the four search criteria. The central icon represents the documents which fulfill all four criteria which in this example only one document does. The surrounding icons show the number of documents which fulfill three criteria. The different textures of the frames of an icon indicate which criteria are met. For instance, users can easily see that there are 12 documents which belong to the subsets *Human Factors*, *Visual OR Graphical*, and *Information Retrieval*, but not *Query Language*. Selections in the InfoCrystal visualization can easily be conducted by selecting the icon representing the relationship between the designated search criteria.

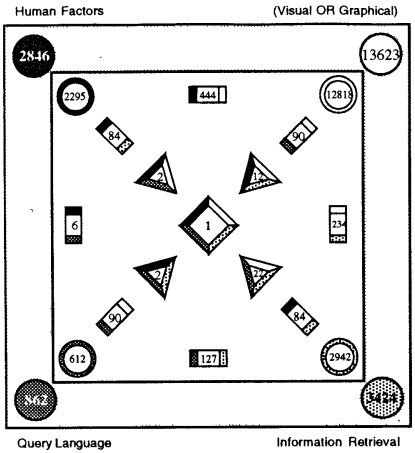


Figure 2.12: An example for the InfoCrystal visualization [Spoerri, 1993].

The InfoCrystal concept includes the assignment of relevance weights whereby users can change from the Boolean model to a vector-space model. Weights can be assigned to each search criterion with a slider which ranges from -1 to 1, where negative weights indicate the interest in documents not containing the search criterion and positive weights show a preference for documents containing the criterion. InfoCrystal supports another view called the Bull's-Eye layout which places relationships with higher relevance more closer to the center of the visualization.

Other exemplary research approaches which use visualizations for the representation of the query are the block-oriented diagram visualization by Anick et al. [1990] and the Magic Lens approach by Fishkin and Stone [1995].

2.5.2 Visualization of Search Results

The field of information visualization offers a wide spectrum of available visualizations. In the following only a small selection is introduced. Shneiderman [1996] introduces the *Visual Information Seeking Mantra* which can be summarized as

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

The description of this mantra contains seven requirements for visualizations to support users that are exploring a data collection. Initially, the visualization should aim for providing users with an overview of the data collection. In the next step users can zoom in on items of interest. By filtering out uninteresting data items users can focus on the items they are interested in. Details about certain items in the dataset can be requested on-demand which are then displayed to the users.

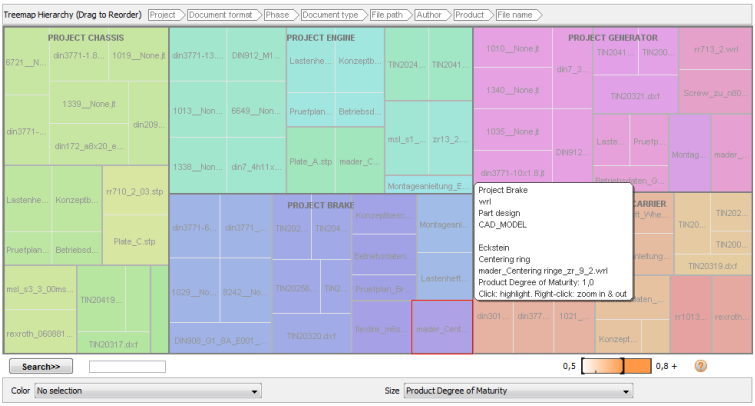


Figure 2.13: An example for the treemap visualization for documents from the product development domain.

Additionally, the visualization should allow users to discover relationships between items. By keeping a history of the conducted actions, users can undo actions, or iteratively refine their analysis. Finally, an extraction mechanism should be provided to allow users to export the found information from the data set.

Visualizations of search results can be applied to different properties of each item in the result.

A visualization type for hierarchical information is the concept of the treemap which maps the hierarchical information to a two-dimensional rectangle [Johnson and Shneiderman, 1991]. The visualization makes use of all available space and targets the demonstration of proportions between different items by displaying specific attributes in sizes proportional to their frequency in the data set. Hierarchical structures are visualized by nested rectangles whereas their size is proportional to the size of a chosen attribute. Figure 2.13 shows an exemplary treemap which visualizes documents from product development processes[18]. The example shows all available documents grouped by the respective projects they were created in. Above the visualization users can change the hierarchy the treemap is built upon. In this example, initially the documents are grouped based on projects. The second grouping is based on the document format which can be seen as documents of the same format are placed near each other and are visualized in the same color. Users can easily change the order of the different levels of the hierarchy by reordering the attributes above the treemap. Treemaps support the selection of subsets by drill down operations. By right-clicking a rectangle the user is presented with the avail-

[18]The visualization was created with Many Eyes from the Collaborative User Experience research group at IBM. The visualization can be accessed at `http://manyeyes.alphaworks.ibm.com/manyeyes/visualizations/treemap-product-data` (last accessed 08/01/2010).

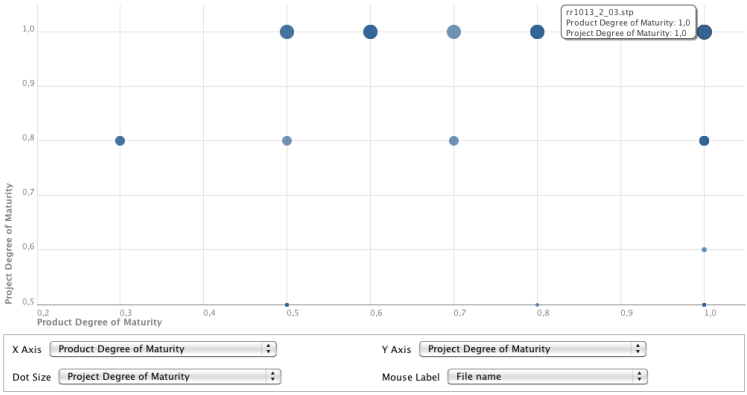


Figure 2.14: An example for the scatter plot visualization for documents from the product development domain.

able levels to which they can filter the result set. Additionally, users can select the attribute based on which the size of each rectangle is computed. In the example in Figure 2.13 on the preceding page the product degree of maturity is chosen. Thus, larger rectangles show documents describing products which are more advanced in the product development process.

The visualization of two variables (or attributes) of items from a search result can be accomplished by a scatter plot. This visualization uses a diagram with Cartesian coordinates to show the results as a collection of color-coded points. Figure 2.14 demonstrates this visualization type with data from the product development domain[19]. The x-axis displays the values of the degree of maturity of a product, whereas the y-axis depicts the degree of maturity of the according project. This type of visualization can for instance be used in the analysis of the progress of projects in the portrayed domain. The shown scatter plot implementation from the Many Eyes project allows users to base the size of a data point in the plot on a certain numeric attribute. Additionally, users can simply change the variables by choosing the variable from drop-down lists below the visualization. Details about a search result can be viewed by hovering over a data point which activates a preview of the item.

This visualization helps to identify dependencies between the search results. This can be positive or negative correlations which are shown as a line pattern. Furthermore, non-linear relationships between variables are made visible such as showing clusters of items which have similar attribute combinations. An interesting use case

[19]The visualization was created with Many Eyes from the Collaborative User Experience research group at IBM. The visualization can be accessed at `http://manyeyes.alphaworks.ibm.com/manyeyes/visualizations/scatterplot-product-data` (last accessed 08/18/2010).

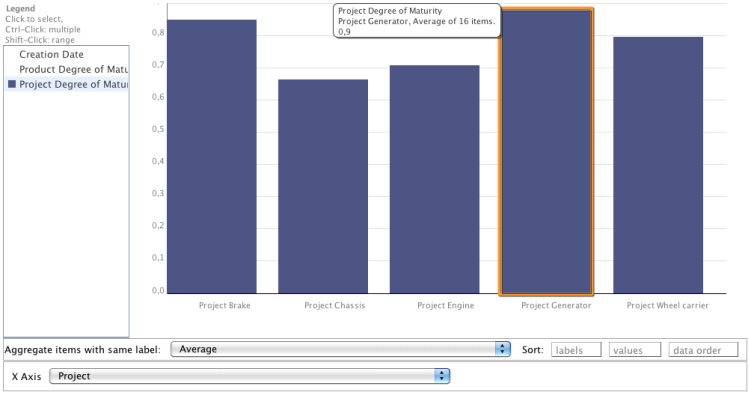


Figure 2.15: An example for the bar chart visualization for documents from the product development domain.

for this type of visualization is the use of the similarity score of an artifact as a variable which is visualized. The plot then displays one other user-chosen attribute dependent on the similarity score. A disadvantage of scatter plots is that multiple items from the result set which share the same values for the two displayed attributes are not distinguishable as they are drawn over each other.

Bar charts help to visualize proportions of an attribute between (sets of) items. The x-axis shows different items of the search result which are then visualized by an attribute which is assigned to the y-axis. The data is charted with rectangular bars whose length is proportional to the value of the attribute. For instance, Figure 2.15 shows the different products which are currently in development and their degree of maturity[20]. Thus, users can easily see which products are finished and which are still in development.

Reiterer et al. [2005] introduce the *INSYDER* system which offers a wide variety of visualizations for results of web searches. In addition to a tabular view of search results, the system uses bar graphs, scatterplots and tile bars [Hearst, 1995] to provide additional ways to approach the search results. The visualizations are used both for the query formulation as well as for the analysis of the search results.

Spence and Tweedie [1998] propose an approach which uses histograms to visualize attributes of a data set in the Attribute Explorer. Users can interact with these histograms by changing minimum and maximum values by direct manipulation. Relationships between attributes are highlighted by color-coding the different objects in the histograms.

[20]The visualization was created with Many Eyes from the Collaborative User Experience research group at IBM. The visualization can be accessed at `http://manyeyes.alphaworks.ibm.com/manyeyes/visualizations/bar-chart-product-data` (last accessed 08/18/2010).

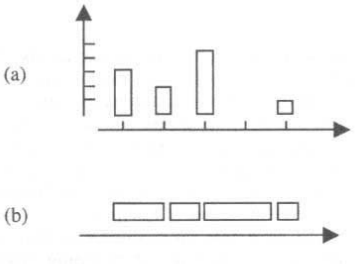


Figure 2.16: A histogram of a data set (a) and the respective bargram resulting from "tipping over" the histogram bin and omitting empty ones (b) [Wittenburg et al., 2001].

Wittenburg et al. [2001] extend bar graphs and histograms and introduce the visualization form of parallel *bargrams* to visualize hierarchical multi-dimensional data. Bargrams are created by taking a histogram and "tipping over" the columns of the histogram and laying them next to each other on a horizontal line omitting empty bins as shown in Figure 2.16. The bargrams can be visually enhanced by displaying value distributions by shading the bargram differently. Their prototype of this visualization resulted in the *EZChooser* where bargrams are vertically positioned parallel to each other. Users can then visually select the values of each attribute and form dynamic queries [Shneiderman, 1994].

The previous visualizations aimed for the presentation of nominal, ordinal and cardinal scale attributes. Another category is temporal data which can be for instance represented by timeline visualizations. Alonso et al. [2007] introduce an implementation which allows the display of both points as well as periods of time.

For a thorough overview of the topic of search user interfaces and the application of visualizations both for the query as well as for the search result refer to Chapter 10 in [Hearst, 2009]. A good overview of visualizations for the field of Web Retrieval can be found in [Mann, 2001].

2.6 Enterprise Search and Information Access Technologies

With the growth of the Internet in the last decade, the area of IR became omnipresent in the form of Internet search engines like Google[21], Yahoo[22] or Microsoft Bing[23]. These search engines simplify the retrieval and the access of information from the Internet. Similarly, enterprises are interested in providing their employees possibilities to retrieve existing information and knowledge from the enterprise to

[21] `http://www.google.com/` (last accessed 08/01/2010)
[22] `http://www.yahoo.com/` (last accessed 08/01/2010)
[23] `http://www.bing.com/` (last accessed 08/01/2010)

increase their productivity by the means of search engines. Efficient retrieval of existing knowledge avoids duplication of work and bad decision-making. Additionally, the time spent for information acquisition can be decreased.

In their study "The High Cost of Not Finding Information", Feldman and Sherman [2003] describe the results of a lack of information:

- "Poor decisions based on faulty or poor information.
- Duplicated efforts because more than one business unit works on the same project without knowing it has already been done.
- Lost sales because customers can't find the information they need on products or services and give up in frustration.
- Lost productivity because employees can't find the information they need on the intranet and have to resort to asking for help from colleagues (Studies by AIIM and Ford Motor Company estimate that knowledge workers spend 15-25% of their time on nonproductive information-related activities.)." [Feldman and Sherman, 2003]

The authors conclude that "knowledge workers need unified, universal access to all information, but they only need that portion of the information that actually solves the information problem at hand."

In recent years the domain which tries to support users in retrieving information in an enterprise context is depicted as *enterprise search*. However, there does not exist a generally accepted definition of this term. Enterprise search solutions comprise the identification of specific contents across the enterprise, the indexing of this information and the retrieval of these contents. They aim for supporting employees to retrieve existing information and knowledge from inside the company as well as external information (e. g. from the Internet). More general, the field is subsumed under the moniker of *Information Access (IA) Technologies* which take enterprise search and supplement it with technologies such as categorization, classification and visualizations. This approach allows the more "intelligent" display of search results than the common plain linear search result list. The main goal is the provision of a better understanding of the organization's information which is distributed over various systems in the diverse software landscapes, such as Document Management (DM) systems, Content Management (CM) systems and other enterprise applications [Andrews, 2009].

Enterprise search solutions strive for providing users with more insight into the organization's knowledge. To achieve this goal these search engines do not limit themselves to plain keyword searches, but offer additional analytical features common in fields like BI. They apply tools such as navigational access, visualizations and charting to provide users a means to understand the search results better.

According to a survey by White [2009] conducted between July and August 2009 with 175 participating companies from which 118 are actively using enterprise search solutions, the main drivers for enterprise search were the improvement of decision making due to a better overview of the available information and the increase of employee productivity.

Braschler et al. [2009] introduce a survey where 233 decision-makers from companies and the administration in Switzerland were questioned about enterprise search. The participants estimated that about 18% of the work time is spent on search tasks. The main motivators for enterprise search according to this study are the loss of time due to the companys' lack of comprehensive search facilities over all information sources, the quality loss due to non-retrievable information, the loss of time due to too large result sets with irrelevant items and the lack of meaningful result presentation.

Enterprise search solutions help to improve the provision of information by providing tools to retrieve information unknown to employees. They help to avoid the duplication of work that has been done in the past but could not be retrieved due to unawareness of the existence or the location of the information. Furthermore, the quality of decision-making can be improved since important internal and external information can be retrieved faster and more complete. Additionally, since the indexed data is delivered in near real time, decisions can be made on current data. Enterprise search should help users to make decisions they would have never made before based on information they might not have known it even exists in the organization.

Enterprise search should deliver the same simplicity as provided by web search engines, but adapted to very complex and security-sensitive information found in enterprises. In a first step, enterprise search should support users in finding existing information, followed by advanced support for making sense of the found information. Enterprise search solutions should aim to be an integral part of business productivity. By making information available not only from the employee's department, but from the whole company a first step is taken towards that goal. Users now need tools to make sense of the obtained information. Quantrill [2008] calls this the transition from "the user interface model into the user experience model" by integrating tools to be able to interact with the data after finding it.

Abrol et al. [2001] gives a characterization of an *enterprise portal* which is one option for an implementation of an enterprise search system. It is characterized as follows:

- The need to access information from heterogeneous repositories.
- The need to respect fine-grained security permissions on a document basis.
- The need to consider a large variety of document types and formats, which additionally can occur in different languages.
- The need to combine structured and unstructured information from a document for the determination of the search result, its personalization and result presentation.

An enterprise portal plays an important role for knowledge management. Employees are offered a centralized, personalized and role-based access to the necessary information needed for their work [Friedrich, 2007]. The above mentioned survey from White [2009] showed that the provision of a single view of all enter-

prise assets helps companies to increase employee productivity. An enterprise portal implements such a single view.

An approach to query multiple application systems from a central search system is the field of federated or distributed search [Callan, 2000; Meng et al., 2002]. Therewith, multiple systems which provide their own search mechanisms are integrated in one system which handles the task of source selection [D'Souza et al., 2000; Nottelmann and Fuhr, 2003], i. e. making the decision which systems need to be queried. The federated search engine then translates the user query to the query language of the targets search systems. After the search results have been returned from the systems, they are merged into a single ranking which is presented to the user [Si and Callan, 2003].

The use of federated search systems offers the benefit for users that instead of searching multiple repositories sequentially, users only have to provide one query which then is handed to the search systems and computed in parallel. A drawback can be the large number of results which can result from querying every source. Additionally, many federated search systems do not provide additional guidance to the users such as faceted search approaches which would help them to filter the search result efficiently.

Ideally, an enterprise search solution provides users with one search result for a query—the right one. Although, these types of (simple) problems are existent in organizations, more complex information needs are more common. Taking for instance design engineers which are designing a new assembly and need a component for certain given requirements. If they could—in addition to the information from the PLM system—access additional information about components e. g. from Supply Chain Management (SCM) systems, the engineers could base their decision of which component to choose on more relevant data.

Therefore, the search process in enterprise search scenarios should be conversation-like. Users should refine their search queries by adding and removing search criteria in a dialog with the search engine which presents additional descriptive attributes of the search results to give users ideas how to detail their queries and new ways to filter and sort the search results.

Hawking [2004] and Mukherjee and Mao [2004] try to define the field of enterprise search and outline challenges in this domain. The next subsections outline these main challenges found in enterprise search.

2.6.1 Heterogeneous Information Spaces

Current enterprises can be characterized by complex information spaces which consist of complex IT system landscapes with a variety of heterogeneous content repositories. Amongst others, these repositories comprise DM-, CM-, Database Management (DBM)-, Enterprise Resource Planning (ERP)-, Customer Relationship Management (CRM)-, and PDM-systems as well as e-mail services, plain file systems and intranets. Many of these software systems provide some kind of search functionality to retrieve the stored information from each individual system. Users seeking for a comprehensive answer for a complex information need might need to query

several information systems consecutively and merge the different search results on their own. An enterprise search engine should integrate these "information silos" and provide a single entry point for the retrieval across these different information systems. For this purpose, an enterprise search engine should ingest and index all the contained structured and unstructured information from these diverse content sources and make it easily retrievable for the users of the organization.

An enterprise search vendor has to provide specific adapters or connectors to access the source repositories and gather the contained content during the indexing step. Since there is no generally accepted standard for comprehensive data exchange across several repositories[24] an enterprise search vendor has to provide connectors for most commonly used information systems.

Often the enterprise search engine offers connector Application Programming Interfaces (APIs) which allow customers to integrate unsupported or internally developed software systems. For instance, the enterprise search solution offered by Google called Google Search Appliance (GSA)[25] provides intranet and website search out-of-the box. Connectors to several enterprise software systems are provided, e. g. for Lotus Notes, Microsoft SharePoint, Salesforce.com, for file systems and databases[26] (cf. Section 2.7 on page 53 for more detailed information on enterprise search solutions).

The manifold information systems store different types of information referred to as *artifacts*[27] in this publication. Each artifact is of a certain artifact type which describes the information it contains. In an enterprise context, artifacts may be of different types e. g. a *document*, *person*, *product* or *project*. Each of these artifact types is characterized by a specific set of descriptive attributes.

Assuming that the search engine acts as a single point of entry for searching the information from these systems, the different result types need to be merged effectively into an integrated search result ranking. These different types introduce a higher complexity due to their different structure. This is obvious considering for instance the search for persons in contrast to a search for projects which differ inherently in their structure, i. e. the available search criteria.

Thus, the introduced variety of artifact types results in a higher complexity for the search algorithms since search results from different systems with different relevance algorithms need to be merged as well as search results of different types. This diverse content requires different relevance models for each type of artifact to determine a search result ranking. Furthermore, it is difficult to combine different artifact types in a search query, since the contents of the heterogeneous repositories in the enterprise environment usually are not (strongly) cross-referenced by links

[24] A standardization attempt for content repositories is for instance the Java Specification Requests 170 (Content Repository for Java™ technology API) and 283 (Content Repository for Java™ Technology API Version 2.0), implemented e. g. in the Open Source Project Apache Jackrabbit (`http://jackrabbit.apache.org/` last accessed 08/01/2010) or in the CM-system Magnolia (`http://www.magnolia-cms.com/` last accessed 08/01/2010).

[25] `http://www.google.com/enterprise/search/index.html` (last accessed 08/01/2010)

[26] `http://www.google.com/enterprise/labs/index.html` (last accessed 08/01/2010)

[27] For a more detailed introduction to artifacts and their types, refer to Section 4.2 on page 94.

[Mukherjee and Mao, 2004] which makes a ranking based on the link structure difficult.

According to Hawking [2004] "enterprises generally prefer a utility function which reflects what the enterprise wants the searcher to see" which exposes the requirement for enterprise search solutions to be able to promote certain artifacts in search results for certain queries.

Furthermore, the information in these systems is stored in a variety of document types and file formats which need to be parsed to extract as much information as possible.

Although, the different artifact types are linked logically, these connections are often not stored in these systems which decreases the potential semantics which could be used for improving search. Furthermore, the links between the artifacts often indicate relationships rather than authoritative information as used in web IR where web pages with many incoming links are valued more than pages with less links.

For users this might pose a problem if they are searching for documents of a certain product and then want to quickly navigate to the according development project documentation. This might include several searches starting from a DM-system, followed by querying the PDM-system for the product information and finally searching the project planning portal until they find the information they need.

Additionally, the found information can be in different languages which has to be considered during indexing and search time. For more information on Cross-Language Information Retrieval (CLIR) see the Cross Language Evaluation Forum[28].

Enterprise search engines must support the search for structured as well as unstructured data. Structured information can be for instance found in relational databases where each column of a dataset is defined. Unstructured information such as text documents is assigned additional metadata[29]. From an indexing viewpoint, the search engine should try to extract as much structured information as possible which can be realized by specific indexing components that apply techniques from automatic classification, clustering and feature extraction.

In particular, the structured information (e. g. metadata) allows user interfaces which enable users to navigate along hierarchical categorized information (cf. Section 2.3 on page 18). Additionally, a combined approach is feasible where users search for unstructured content in conjunction with a parametric approach, i.e. querying for some keywords in addition to some structured search criteria the results must meet. The integration of structured and unstructured information about the artifacts helps to enrich and enhance the quality of search results for the users.

By utilizing Natural Language Processing (NLP) approaches such as information extraction, automatic identification of named entities (persons, organizations, products, etc.) and their relations, structured information can be obtained automatically to a certain degree[30]. The Apache open source framework Unstructured Information

[28] `http://www.clef-campaign.org/` (last accessed 08/01/2010)

[29] For instance, documents can be assigned author and process information by checking it into a DM-system.

[30] Refer to [Bates, 1995] and [Charniak, 1997] for an overview of NLP techniques.

Management Architecture (UIMA)[31] supports the development of analysis tools for unstructured information [Broder and Ciccolo, 2004].

A more broad approach is taken by the SeMantic Information Logistics Architecture (SMILA)[32] which provides an extensible framework for building search solutions especially for unstructured information sources in addition to the proposed architecture.The extensible framework is based on Service-Oriented Architecture (SOA) principles and utilizes standards such as Business Process Execution Language (BPEL) and Service Component Architecture (SCA) to provide a platform for information access and integration. It also defines components such as data connectors to different source systems and delivers interfaces for the management, operation and monitoring of the framework and its components.

2.6.2 Internet vs. Intranet Search

In Web IR several approaches are followed to determine the relevance of a web page. One approach calculates the similarity between a document and the query by applying the Vector-Space-Model from Salton et al. [1975]. Additionally, the link structure between web pages can be utilized to determine the "importance" of a web page [Page et al., 1998; Brin and Page, 1998]. For each web page in the search engine index a so-called *PageRank* is calculated which weights links from referring web pages to the considered page differently. It is assumed that web pages which are linked to frequently, are more relevant than web pages which are linked to less. During the PageRank calculation the assigned PageRank weights are propagated from the referring page to the linked page. The rank of a page is higher, if the sum of the ranks of the referring pages is high. This approach covers the case when a page is linked often and when a page is only linked from a few highly ranked pages[33]. Pages with a higher page rank appear higher in the ranking. PageRank has been influenced by the field of *citation analysis* where relations between cited and citing publications are analyzed to determine the scientific impact of a publication [Garfield, 1979]. This citation approach is not completely transferable to intranets due to several reasons explained below.

A part of enterprise search consists of the retrieval of relevant information from the company's intranet. Although at a first view the Internet and intranets are quite similar, there are several differences which have to be considered for searching in each respectively. Both share the basic overall structure consisting of a collection of hyperlinked documents. But the differences are rooted in the underlying document content generation process. Content creation in the Internet is more demo-

[31] The UIMA framework provides a component based approach for the analysis of "large volumes of unstructured information in order to discover knowledge that is relevant to an end user" (`http://uima.apache.org/` last accessed 08/01/2010).

[32] Further information about SMILA and the available releases can be found at `http://www.eclipse.org/smila/` (last accessed 08/01/2010).

[33] It has to be noted that the current implementation of Google's search algorithm takes over 200 signals into account, whereas PageRank is only one of them (`http://www.google.com/corporate/tech.html` last accessed 08/01/2010).

cratic, considering e. g. the collective efforts of creating an independent encyclopedia such as *Wikpedia*[34] often summarized under the moniker *Web 2.0* coined by Tim O'Reilly[35]. In contrary, content creation in an enterprise context is often more bureaucratic due to a more centralized documentation process performed by a small group of employees. This leads to the paradox that although the amount of information on the Internet is usually much larger than in the enterprise, IR on the Internet is easier than in the enterprise. This can be explained by the large number of web sites which handle a certain topic and therefore provide advice for many natural queries [Fagin et al., 2003]. That publication also claims that the notion of a "good" answer to a query is quite different for these two domains. Whereas users issuing queries to web search engines usually expect the "best" or most relevant document, in the enterprise context, users are usually interested in finding the single "right" answer. Often users already know or have seen the right document before, but are seeking help to easily retrieve it again. The "right" answer is not necessarily the most "popular" document, which frequently determines the "best" answer on the Internet [Fagin et al., 2003].

Another difference between internet and intranet search is the handling of spam content. Internet search engines need to undertake huge efforts to prevent spam in the search results. Therefore, current web engines rarely rely on the descriptive META tags the Hyper Text Markup Language (HTML) provides, because this was often misused to promote own webpages for certain search keywords. In organizations, this spam problem does not exist due to the above mentioned content creation process. Therefore, this META tag information is much more reliable than in Web search.

With the vast success of Web 2.0 efforts in the Internet, companies try to embrace this approach and port it into the enterprise by trying to motivate their employers to collectively document knowledge by employing (micro-)blogging, wikis, social networking tools, etc. Recently, this was labeled *Enterprise 2.0* [Newman and Thomas, 2009] and aims for supporting the collaborative information creation and discovery.

2.6.3 Consideration of Content Security

The main objective of enterprise search approaches is the improvement of the access to company information for their employees. The simple approach to index all available information regardless of their classification is obviously not feasible. An enterprise search engine has to take the security contexts of the source systems into account and enforce them before delivering search results. The search engine has to ensure that users can only find and access the information they have the permission to. In consequence, this can lead to the situation where two different users issuing the same queries are returned two different result sets.

One basic requirement for enterprise search solutions is the enforcement of the current security permissions of a document [Bailey et al., 2006]. In contrast to col-

[34] `http://www.wikipedia.org/` (last accessed 08/01/2010)

[35] In 2004 the first O'Reilly Media Web 2.0 conference took place (`http://oreilly.com/web2/archive/what-is-web-20.html` last accessed 08/01/2010).

lection level security where security permissions are granted or denied on an entire set of documents, this more granular approach is subsumed under the moniker of *document level security (DLS)*. For enterprise search engines only the read permissions of an artifact are important, since the source systems are still responsible for enforcing write or execute permissions for users. The search engine only needs to decide if users can see a certain artifact or not.

An enterprise search solution should contain various adapters to connect to the different authentication systems used in modern enterprises. Often Single Sign-On (SSO) systems are used in organizations to provide users with a single point of entry where they have to authenticate themselves. The SSO system then transparently passes this authentication to other applications.

In current enterprise search engines on the market two different strategies to enforce the security of the indexed information are realized: *early binding* and *late binding*. With *early binding* (also known as mapped security or Access Control List (ACL) caching / indexing), the search engine not only indexes the artifact, but additionally adds its security properties to the index. During query time, the search engine can immediately access the cached security information of the potential search result and include it in the result ranking, if the users have the necessary security rights. The advantage of this approach is the faster query processing time due to the already present security information in the index. The downside might be the outdated security information if the security information is altered by e. g. revoking the read permissions of a document for a certain user. If the search engine only indexes once a week, users might still be able to see this search result (although they might not be able to open it, since the security is still enforced by the source application). Typically, enterprise search engines are expected to minimize the delay between the creation of new content and its retrievability. The same applies to the currency of the security policies [Bailey et al., 2006].

This potential security violation can be circumvented with the *late binding* approach (also called *mapped security* or *ACL mapping*). When the search engine is queried, an intermediate search result ranking is computed which in a second step is filtered by security constraints before displaying it to the users, i. e. all artifacts are removed for which the users do not have the sufficient rights. This approach requires real-time verification of access rights for the current user and artifact from the source system. Therefore, this approach is the most responsive way to changes in the security access rights. On the downside, *late binding* obviously strongly affects the query processing time and the scalability of the solution. Furthermore, a much higher load is put onto the source systems. Some systems also provide both or a combination of these security approaches.

When planning to introduce an enterprise search solution the security aspect should not be neglected. Organizations need to find a trade-off between search engine performance and the risk of the disclosure of protected or confidential information. Organizations need to decide whether the knowledge of the existence of an artifact is already a security violation, even though users might not be able to open the document because of missing read permission. In large enterprises department specific software systems often exist which are “sealed off” from other departments.

In certain situations it might be beneficial for users to access and reuse information from a past project of other departments to avoid unnecessary duplication of work. The company now needs to decide if it is acceptable to notify the searchers that there might exist information which could prove helpful for them but omit the presentation of the document. The users now could try to approach the respective department and gain access. Obviously, this is not always feasible especially with projects protected by non-disclosure agreements by customers.

Bailey et al. [2006] points out that the enforcement of the underlying security policies also impact the processing time due to the calculation of the total number of search results. Even though users might only view the first search result page, security permissions of each search result have to be checked.

2.6.4 Expertise Retrieval / Expert Search

The field of *expertise retrieval* has recently received increased attention in IR which lead to the introduction of the *Expert Finding* task at the Text REtrieval Conference (TREC) in 2005 [Craswell et al., 2005a]. Expertise retrieval copes with two different types of tasks. *Expert finding* tries to determine a list of persons who are knowledgeable on a certain topic, i. e. the experts on that topic. In contrary, the *Expert profiling* task returns a list of topics a person is knowledgeable about. Early approaches relied on database approaches where the skills of persons within an organization were manually captured and maintained which was very time-consuming and expensive.

Automatic approaches usually examine the associations between persons and topics assuming that the co-occurrence of a person's name and a topic shows evidence of expertise for both tasks usually.

Search systems especially developed for the task of expertise retrieval are known as *expert finders* [Yimam and Kobsa, 2000]. Their goal is to automatically discover expertise information by analyzing secondary information sources.

One approach to automatically determine skills of employees is the analysis of e-mail communication. This type of communication seems particularly well suited for the "expertise location" task, since persons usually communicate their knowledge. Moreover, since e-mail is a directed communication, social networks can be identified in the patterns of communication [Campbell et al., 2003].

Several independent research groups carried out analyses of graph-based ranking measures such as PageRank and the Hyperlink-Induced Topic Search (HITS) algorithms to determine expertise based on e-mail communications. Campbell et al. [2003] compared the analysis of the contents of e-mails communication to the exploitation of the link structure which is defined by senders and receivers of e-mails applying the HITS algorithm [Kleinberg, 1999] to identify experts. The findings of Dom et al. [2003] showed the better applicability of the *PageRank* algorithm over the HITS algorithm for the e-mail communication analysis. D'Amore [2004] examined the expertise finding problem from the perspective of identifying communities of expertise in his *Expert Locator* prototype. The authors motivate an enterprise model organized around *activity spaces* or *work contexts*. These models associate

expertise with expert signaling behavior, i. e. communication used to convey certain knowledge or experience.

According to Balog [2008] many approaches exist which automatically exploit social networks as a source for expert finding tasks. These networks can be based on and constructed from chat logs, online discussion forums, community-based question-answering systems, or co-authorship information from bibliographic databases.

A more comprehensive approach is pursued by Craswell et al. [2001] with their *P@nopticum* system. They built a representation of each person in the organization by concatenating the text of all associated documents of these persons in the organization's intranet. Textual queries for persons with a certain expertise are then matched against this document representation, similar to a classic document retrieval system. The search result comprises a list of experts, along with their contact details and the list of matching documents as supporting evidence.

Enterprise PeopleFinder builds on the findings of the P@nopticum system and refines the representation of a person's expertise [McLean et al., 2003]. The authors additionally accept more documents as evidence of expertise based on the corporate structure and use a more differentiated weighting scheme of the documents (in comparison to the original equal weights).

Balog [2008] proposes a generative probabilistic retrieval framework for estimating the probability of a person being associated with a certain topic of expertise by adapting generative language modeling (LM) techniques. On the one hand, the author uses the associations between people and documents to build a candidate model and matches the topic against this model. On the other hand, the topic is matched against the documents and the resulting associations are evaluated to find evidence for a candidate's expertise. This probabilistic retrieval framework was successfully applied for both the tasks of *expert finding* and *expert profiling* [Balog et al., 2006].

Macdonald and Ounis [2006] consider the expert finding task as a voting problem, where documents vote for candidates with the relevant expertise. Each candidate is assigned a set of documents describing its profile. The user query for a candidate is matched against the documents describing all the profiles. Simplified, a candidate appears higher in the ranking if many documents from its profile appear in the ranking in comparison with other candidates. The authors compared 11 different (adapted) data fusion techniques to aggregate the document "votes". The authors compare the aggregation of votes for the candidates by different measures: (i) the number of retrieved document votings for each candidate, (ii) the scores of the retrieved documents voting for each candidate, and (iii) the ranks of the retrieved documents voting for each candidate. The authors emphasize the advantage of their approach which does not rely on collection specific heuristics, but is deployable widely for different collections.

2.6.5 Evaluation of Enterprise Search

In 2005 TREC introduced the *Enterprise Track* to provide a platform to evaluate different research approaches for certain enterprise search tasks [Craswell et al., 2005a; Soboroff et al., 2006; Bailey et al., 2007; Balog et al., 2008]. The track provides a test collection comprising web crawled intranet pages, e-mail archives, and document repositories. The test collection also comprises certain topics and according relevance judgments for different search tasks:

- **Expert search.** The aim is to provide a ranking of experts for a given topical area.
- **E-mail known item search.** The goal is to provide a certain e-mail which is known by the querying user.
- **E-mail discussion search.** The aim is to provide users with an overview of the pros and cons of an e-mail discussion. The user describes the topic by his query and expects whether the results are relevant and whether they contain arguments for or against the argument.
- **Document search.** The main goal is to construct an "overview page" which lists relevant "key pages" and "key people" of interest for a given topic of interest.

An interesting task was added in TREC 2009 which focuses on entity-oriented search on the World Wide Web. A query for the *entity finding task* comprises an input entity, a type of the target entity (person, organization, or product) and the relation between the two entities given in freeform text [Balog et al., 2009].

2.6.6 Consideration of Contextual Information

Due to the heterogeneity of information in the enterprise, it is beneficial to consider contextual information about the users and the retrievable artifacts to improve the search quality. For instance, the information needs of users working in Research & Development differ from users working in the sales department. Furthermore, several studies showed that users typically use only a few terms to describe their information need (e. g. [Silverstein et al., 1999] and [Pass et al., 2006] who analyzed big web query logs found the number of query terms to be between 1 and 4 on average). While analyzing queries which were conducted against an email archive, Hawking et al. [2005] found that 33% of all queries were one term queries. This discrepancy between what users query and what users actually need makes query processing a challenge for search engines.

The applicability of context information for the target environment needs to be evaluated and adapted [Hawking et al., 2005]. The knowledge about users in an enterprise usually comprises the roles of each user, which give hints about their short and long term information needs and can play an important role during the matching and ranking phase of the enterprise search process.

As described in Section 2.6.1 on page 43 above, the document collection is very heterogeneous with usually unstructured documents of various types originating from a wide variety of information repositories. However, each of these individual management systems can provide a local context for an artifact, as process context information for documents in an DM-system (i. e. information about where in a process the document was created, was revised or is needed for further tasks as input documents).

The consideration of process context such as information about work tasks can help to locate the information source and better understand the information need of the users. Järvelin and Ingwersen [2004] claim that information about task context needs to be considered to improve information seeking and retrieval.

Based on results from a study conducted by Freund et al. [2005b] in the domain of software engineering, a discrete set of contextual variables were found. The most significant variables were the *work task*, the *information task*, and the *genre* [Freund et al., 2005a]. A work task may lead to a number of information tasks which comprises information search and retrieval tasks. The genre defines the document type based on the similarity of form and purpose. The authors try to find associations between specific tasks and genres in contrast to an isolated view by conducting a correspondence analysis which led to the identification of relationships between certain tasks and genres. The further goal of the authors is to incorporate these relationships into the ranking computation to improve the usefulness of the search results.

The knowledge of the users' tasks may help the search engine to understand the user's information need better. This can be used for source selection during a retrieval task where the search engine does not query all potential source systems, but only those which possibly could answer the queries of the users. For instance, if a search query contains the name of a person, it might be appropriate to query an expert finding search tool [Hawking et al., 2005]. Additionally, the knowledge about the current task of the users can be leveraged in providing them with search templates which preselect certain search criteria to limit the degrees of freedom for the users.

A work package from the European integrated project VIVACE dealt with contextual search for engineering knowledge [Redon et al., 2007]. The authors identified six context dimensions they further investigated, namely *activity*, *project*, *gate*, *role* and *discipline*. The connection between a knowledge element and its context can either be specified by an expert or determined automatically by the platform itself. The search process is based on a case-based reasoning system, which utilizes user context descriptions and the applying knowledge element-context associations.

An interesting field for context-based IR concerning enterprise search is the possibility of alerting users that there exists potential information based on previous searches. Additionally, users could define their interests and regularly receive matching information[36]. This type of information retrieval task belongs to the field

[36]Google provides a feature in form of their Google Alerts (`http://www.google.com/alerts` last accessed 08/01/2010), where users can enter keywords and are alerted by Email or web feed such as Really Simple Syndication (RSS) when new web pages are available for the entered keywords.

of *information filtering* [Hanani et al., 2001].

2.7 Enterprise Search and Information Access Technologies: Examples

In this section some examples for enterprise search solutions are given and introduced. The selection of the portrayed systems shows the different types of enterprise search solutions which tackle different problem areas. Whereas the "one size fits all" solution provided by the GSA is simply deployed in the enterprise, the solutions from *Endeca* and *Exalead* try to support more complex and specialized information needs and provide enterprises with more visualization support for exploring search results.

A more thorough market overview and an evaluation is given for instance in the Forrester Wave: Enterprise Search [Owens, 2008] and in the Gartner MarketScope for Enterprise Search [Andrews, 2010].

2.7.1 Google Enterprise Search Solutions

In addition to their web search engine, Google also offers enterprise search solutions. The Google solution is a combination of hardware and software and comes in two different models, the *Google Mini* and the *GSA*[37] [Google Inc., 2009b]. Both product models come in a self-contained rack mounted server unit.

Since the GSA comprises all the features of the Google Mini only the former will be introduced here. The information presented here is extracted amongst others from the public information Google provides on their web site[38], data sheets [Google Inc., 2009a] and from [Edwards, 2009].

Similar to their intuitive approach to web search, Google tries to provide users in the enterprise a similar experience and therefore initially offers the known "one easy, familiar search box" [Google Inc., 2009a]. According to Cyrus Mistry (Product Manager for Google Search Appliance)[39], Google's primary goal is to deliver only one— the right result for a query to the users.

Indexing and Searching

The functionality of the GSA supports various content sources comprising file shares, web servers, DM systems and enterprise applications which it is able to index. Google provides built-in native support for the following systems:

- EMC Documentum[40]

[37] The explanations of the GSA apply to Version 6.0.

[38] `http://www.google.com/gsa/` (last accessed 08/01/2010)

[39] `http://googleenterprise.blogspot.com/2009/08/compare-enterprise-search-relevance.html` (last accessed 08/01/2010)
`http://www.youtube.com/watch?v=XeG3-n9u-3c` (last accessed 08/01/2010)

[40] `http://www.documentum.com/` (last accessed 08/01/2010)

- IBM FileNet[41]
- Microsoft SharePoint Connectivity[42]
- Opentext Livelink[43]
- Lotus Notes[44]
- Salesforce[45]

Third parties provide additional connectors for systems such as BEA AquaLogic[46], IBM Websphere[47] or Oracle Content Server (Stellent)[48].

For other unsupported enterprise applications, Google offers the open *Content Connector Framework* which includes an open Service Provider Interface (SPI) to securely connect with any other content platform. Additionally, a *Third Party Content Feed API* is provided over which source systems can push new non-web accessible content to the GSA by converting the content to a specified Extensible Markup Language (XML) format. This functionality helps the search engine to be notified of changed content which then is re-indexed by the crawler. Additionally, the GSA crawler continuously searches for new content on an ongoing basis to ensure a small gap between content changes and them appearing in search results.

Content-wise Google claims the support for 220 different file types including HTML, Portable Document Format (PDF), Microsoft Office formats and others. Nevertheless, no information is given whether the GSA supports the extraction of structured content out of unstructured file types such as Microsoft Word documents. Additionally, an automatic language detection for several languages is supported during indexing and an automatic translation of search results is provided.

Other features of the GSA comprise a *Self Learning Scorer*, a form of implicit relevance feedback[49], which adapts search results based on user behavior and previous user selections.

Similar to Google Alerts[50], users in the enterprise can get regular email notifications of new documents of interest based on their specified topics/keywords.

With the huge growth of micro-blogging platforms such as Twitter[51], Google now offers the integration of recent Twitter messages in the search results of the GSA.

---

[41] `http://www-01.ibm.com/software/data/content-management/filenet-content-manager/` (last accessed 08/01/2010)

[42] `http://sharepoint.microsoft.com/` (last accessed 08/01/2010)

[43] `http://www.opentext.com/` (last accessed 08/01/2010)

[44] `http://www.ibm.com/software/lotus/products/notes/` (last accessed 08/01/2010)

[45] Salesforce is offering CRM in the cloud `http://www.salesforce.com/` (last accessed 08/01/2010)

[46] `http://www.oracle.com/bea/` (last accessed 08/01/2010)

[47] `http://www.ibm.com/websphere/` (last accessed 08/01/2010)

[48] `http://www.oracle.com/stellent/` (last accessed 08/01/2010)

[49] For a bibliography concerning implicit relevance feedback refer to [Kelly and Teevan, 2003].

[50] `http://www.google.com/alerts/` (last accessed 08/01/2010)

[51] `http://www.twitter.com/` (last accessed 08/01/2010)

Security

The GSA integrates with existing security and access control systems. Therewith, Google supports *document-level security* to ensure that users only see those documents in the search result where they have also access permissions to the content source. Google provides *early* as well as *late binding* for security checking the search results.

Google provides various authentication and SSO mechanisms such as Lightweight Directory Access Protocol (LDAP), NT LAN Manager (NTLM) authentication, Public Key Infrastructure (PKI) authentication with X.509 certificates, Kerberos and Windows Integrated Authentication. By supporting the Security Assertion Markup Language (SAML) Identity SPI a customization of access control can be achieved with legacy systems.

Additionally, the GSA provides an own SSO solution called *Universal Login* which provides a single login page for users and passes their login information to back end system across heterogeneous authentication protocols.

Scalability

The GSA 6 is built upon a new architecture $(GSA)n$ which according to Google offers higher scalability since several GSA units within an organization can be combined. Therewith, organizations can dynamically add another GSA to handle additional query load. Furthermore, this architecture enables unified searching across multiple GSA instances from several departments which might be geographically distributed.

The *Collections* feature allows the creation of different index segments which can be accessible to different users. By that means it is possible to show different search results to different user groups, e. g. by evaluating the domain name, the geography or job function. This functionality provides some aspects of the consideration of application context (cf. Section 2.6.6 on page 51).

Summary

In summary, the GSA is an easy to use solution for enterprise search due to the combination of hardware and software. Additionally, the delivery model comprises product updates and support. The scalable architecture allows the solution to grow with the company requirements.

On the downside, the Google solution does not really support users in finding search results for complex information needs. Google advances the view that the search engine can provide users with the single right answer which works in some query situations. But in situations where users are not able to define their information need by giving a simple keyword query—as the Google solution demands—an emphasis on exploratory approaches would be more helpful for users.

2.7.2 Endeca Information Access Platform

Endeca provides enterprise search solutions for different domains such as retail, media & publishing, distribution, manufacturing and government. Endeca describes itself as a provider of *search applications* in contrast to "one size fits all" applications/appliances such as the GSA. The latter solutions provide generic search approaches applicable in a widespread set of use cases. In contrast, search applications are tailored to help a specific set of business users to complete a specific set of tasks.

Endeca strongly focuses on supporting users conducting *exploratory searches* (cf. Section 2.2 on page 15) and provide a comprehensive set of user interface components to support users in gaining insight into the organization's information. The solution focuses strongly on the integration of structured content into searches by offering additional tools for users to filter search results. They aim for improving the discovery of existing information by providing them with the analytical power of business intelligence in enterprise search tasks.

The information provided below originates from the following publications [Endeca, 2007], [Endeca, 2008], [Endeca, 2009a], and [Endeca, 2009b]. Additionally, Endeca patented their approach in several patents such as [Ferrari et al., 2006a,b].

Presentation / End User Experience

To support exploratory searches in enterprise search, Endeca's search application called Information Access Platform (IAP) is based mainly around their implementation of faceted search called Guided Navigation® in their terms. The main goal of Endeca's IAP is to reveal relationships in enterprise data and content to encourage information exploration and discovery. In addition to the characteristics of faceted search introduced in Section 2.3 on page 18, Endeca provides presentation types for special types of facets. According to [Endeca, 2009a], Guided Navigation is the first commercial version of faceted search.

Users can initiate a search process by giving a keyword query or by filtering the whole result set by stating initial facet criteria. The returned search results contain a set of relevant documents based on the given search criteria and a set of search result summaries. These summaries help to refine the user query. These summaries can be the query previews (cf. Section 2.3 on page 18) which show the size of the result set when users would choose this facet value. Furthermore, users can give a value range by using range filters. If geospatial information is available, it can be displayed on a map which helps filtering by a location. An interesting use case for this type of query summary is the search for a supplier which should be located nearby. Other built-in supported visualizations are tag clouds [Hassan-Montero and Herrero-Solana, 2006] to visualize important terms or facet values of a facet of the result set. Other summaries include charts and graphs which serve as a means to make the result set more understandable and give more insights about the data. The result set summarizations provide users with guidance helping them to refine their search query and explore the dataset further.

During the deployment of the search engine, line-of-business managers have

fine-grained control over the placement and the functionality of the user interface elements. The selection of which summaries are displayed to the users is configurable by precedence rules. By giving these rules, it can be controlled which facets are shown and and which are hidden based on the already chosen facets and their values. The determination of which facets are shown to users is called second-order relevance by Endeca. It is important to determine those facets which most probably help users in finding what they are looking for. Additionally, the right type of summary for the facets needs to be determined to provide effective guidance.

Furthermore, line-of-business managers can promote specific documents in the search results for specific queries. This type of search result elevation is called *Content Spotlighting*® in Endeca terms.

The functionality of the application layer of the Endeca IAP can be extended with new user interface features authored in XQuery[52]. This makes it easy to pull data out of the *MDEX Engine* (cf. Section 2.7.2 on page 59) to provide users with new data visualizations for enhanced guidance through the search process.

As typical for faceted search (and thus Guided Navigation®), after each query refinement step of the users, the facets and their values which are shown to them are recalculated and shown with updated and exact (i. e. no approximations) category counts (the *query previews* introduced in Section 2.3 on page 18 above).

Endeca supports so called "Record Relationship Navigation" which allows users to apply facets from different artifact types which share a relationship. Thus, a navigation by facets of facets is possible. Design engineers can for instance filter search results to only contain subassemblies containing parts of type *Bearing* which are in a second step filtered by facets describing manufacturers. This approach is viable since there exists a relationship between products and their manufacturers. Of course this information needs to be determined during indexing [Endeca, 2009a].

Lately, Endeca released their User Interface Design Pattern Library[53] which aims at providing a knowledge base for best practices for the design of search user interfaces.

Connectors / Indexing

The part of the architecture of Endeca's IAP which handles indexing of enterprise content is called *Information Transformation Layer (ITL)*. In an offline process, the ITL processes and enriches the enterprise data. It consists of two key components: The *Content Acquisition System (CAS)* and the *Data Foundry*. The former enables the extraction of content from diverse source systems.

Part of the enterprise search solution is the Developer Studio which comprises manifold options to configure the retrieval and access of enterprise content sources.

The CAS can process information from a variety of sources and formats. Endeca's CAS natively supports normalized data input formats, including comma-separated

[52] XQuery is a standards-compliant query language for semi-structured content. For more information on XQuery see `http://www.w3.org/TR/xquery/` and `http://www.w3.org/XML/Query/` (both last accessed 08/01/2010).

[53] `http://patterns.endeca.com/content/library/en/home.html` (last accessed 08/01/2010)

values (CSV), vertical[54], fixed width, and XML. The latter one can be converted by an XSLT transformation into the custom Endeca Record XML format. Endeca provides the Developer Studio tool which supports easy reading and formatting of these data formats.

Endeca provides crawlers for content stored on web servers and on file servers which automatically search for new and changed information and pass the found content over to the *Data Foundry*. Endeca claims that they support over 390 file formats from which they can extract content and metadata, but no indication is found in the product descriptions of how much structured content can be obtained of unstructured content. The Developer Studio can be used to configure the crawlers, e. g. adding inclusion and exclusion filters.

In addition to the documents itself, the Endeca crawlers also acquire security information, such as ACLs which enables the above mentioned *early binding approach*.

Enterprise applications running on a Relational database management system (RDBMS) can be accessed via Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) adapters which are standard CAS components. The records can be retrieved for indexing from databases by stating SQL statements from the Developer Studio.

Whereas the introduced access options are of a generic nature, Endeca also provides specific connectors, called Content Adapters, which are made specifically for a particular content source. The advantage of these adapters over the generic approach is the support for system-specific data structures and thus more efficient metadata and relationship extraction. Endeca provides content adapters for enterprise applications such as CM, CRM, ERP, DM or Enterprise Content Management (ECM) systems and can utilize common Extract, Transform, and Load (ETL) tools to index data from these systems. According to [Endeca, 2007], the IAP supports systems such as EMC Documentum, Microsoft SharePoint, Salesforce.com, PTC Windchill[55] and others.

For systems yet unsupported by Endeca, they provide the Content Adapter Development Kit (CADK) which allows companies to develop custom content connectors.

The Data Foundry layer of the ITL aggregates the extracted information from the different sources and tries to infer relationships across the data. Additionally, unstructured content is analyzed and transformed by techniques of classification, language identification, entity extraction and taxonomy generation. For instance, Endeca provides automatic category creation which automatically determines facets from unstructured text by employing entity extraction to automatically recognize people, places, and organizations. Additionally, an auto-tagging mechanism is available which tries to determine the most salient terms of a document set.

The results of these analytical steps can for instance be navigational facets for the documents or key terms for tag clouds.

The Data Foundry layer can be extended by plug-ins for custom in-house enrichment packages and data quality routines to measure or ensure data quality. The

[54]The records are stored as property name/value pairs, each separated by a delimiter character.

[55]PTC Windchill is a PDM system for storing and managing product data in the product development domain.

processed information from the ITL is then passed to the so called MDEX Engine which is Endeca's index structure.

Back End Engine: MDEX Engine

Since Endeca's search solution heavily relies on the accommodation of structured, semi-structured and unstructured content, a data model capable of representing these types of information is required. Thus, Endeca developed an index structure based on semi-structured databases[56] called MDEX Engine®.

According to [Endeca, 2009a,b] the MDEX Engine® incorporates techniques from relational databases, traditional search engines, and XML databases. It is built on a flexible data model, where each record is effectively self-describing (similar to XML). Every indexed field of a document (including its full text) and every value it contains (including all indexed terms) become a unique dimension that slices across the data. Each record representing a document consists of a collection of attribute/value pairs which are not dictated by an overarching schema of the data model.

Since user queries are not known in advance, no profound query optimization can be preprocessed in advance. Endeca filed several patents for their techniques used in query evaluation which includes tailored indices, query planning and memory caching [Ferrari et al., 2006b], [Ferrari et al., 2006a], [Ferrari et al., 2007]. Their index structure achieves high compression and efficient memory scans by employing a "vertical record store", which is slightly analogous to column-oriented databases[57] in a RDBMS but differs due to the characteristics unique to semi-structured data such as sparsity and hierarchy.

The IAP answers queries according to the *set retrieval* approach where documents are assigned to two sets: the documents which adhere to the search criteria and to those which are not. Additionally, the search results are summarized along several dimensions of search criteria. The summaries of the search result are processed on the server side to ensure high performance.

The MDEX Engine® offers different access option for data indexing, query construction and application development such as XQuery and Web Services technology

[56] Semi-structured data is a type of data which does not adhere to a general structure, but contains part of its structural information in itself [Abiteboul et al., 2001]. One standard to express semi-structured data is the Object Exchange Model [Papakonstantinou et al., 1995] which can be applied for exchanging semi-structured data between object-oriented databases. In the Object Exchange Model data is modeled as a directed graph where the nodes represent objects and the arcs their attributes. Objects can be atomic or complex. The former can be of type *Integer*, *String*, or similar and represent the leafs of the graph. The attributes of the latter are depicted as arcs which reference complex objects or atomic objects. Another common notation of semi-structured data is XML (`http://www.w3.org/TR/REC-xml/` last accessed 08/01/2010).

The graph-like notation supports navigational or path-based queries efficiently such as *XQuery* (`http://www.w3.org/TR/xquery/` last accessed 08/01/2010) and its subset named *XPath* (`http://www.w3.org/TR/xpath/` last accessed 08/01/2010).

[57] Column-oriented databases (in contrast to row-based databases) store their content by columns rather than rows [Stonebraker et al., 2005]. This can be advantageous when aggregates over many rows (such as query previews) have to be computed often, which is useful in applications such as Online Analytical Processing (OLAP) or the here applied faceted search.

for high interoperability.

To improve the scalability of Endeca's search application it is built on the Endeca Virtual Engine (EVE) which acts as an abstraction of the query evaluation from physical processors to improve performance across multi-core processors. Furthermore, parallel query processing is supported.

Extensibility

Endeca's search application is extensible by a plug-in mechanism using so-called *Cartridges*. New features such as query mechanisms, filters or summaries can be authored in XQuery and exposed as a cartridge which then can be included in the search application.

Endeca already provides a set of packed components such as for rendering Guided Navigation®, search boxes or search results. They are both available for the JAVA[58] and .NET[59] platform and are integrated in common integrated development environments (IDEs) such as Eclipse and Visual Studio where developers can easily integrate them via *Drag and Drop* into the search result page template.

The search application is built on a SOA to allow easy integration into the enterprise software stack. Newly developed custom services can be published to Web Service endpoints and accessed via Simple Object Access Protocol (SOAP) or Hypertext Transfer Protocol (HTTP)-based web services (including Representational State Transfer (REST) and XML-Remote Procedure Call (XML-RPC)).

Example Domain: Manufacturing

One specific tailored search application is Endeca's *Discovery for Manufacturing* suite which strives for the improvement of decision-making during product development. The suite includes the modules Design for Supply, Spend Analysis, and Warranty Analysis [Endeca, 2009c]. The main goals are the support of engineers to provide them with a better visibility of existing parts or assemblies in the company to avoid re-inventing already existing parts and to support re-use of existing parts. The suite integrates data from several information silos such as product catalogs, PLM and SCM systems and presents them in a unified manner and allows exploration of this data.

Figure 2.17 on the facing page shows an example of the Manufacturing Suite with user selections on the criterion *part family*, a geographic filter and the *inner diameter* of the parts. Users are presented with a list of search results in the center of the user interface and various additional filter options. For instance, users could filter the search results by *material* or the favored *supplier*. Other visualizations provide the users with insight about the *average fulfillment time* in a pie chart, a map for location data or auto-completion of keyword queries.

[58] `http://www.oracle.com/technetwork/java/index.html` (last accessed 08/01/2010)
[59] `http://www.microsoft.com/germany/net/net-framework.aspx` (last accessed 08/01/2010)

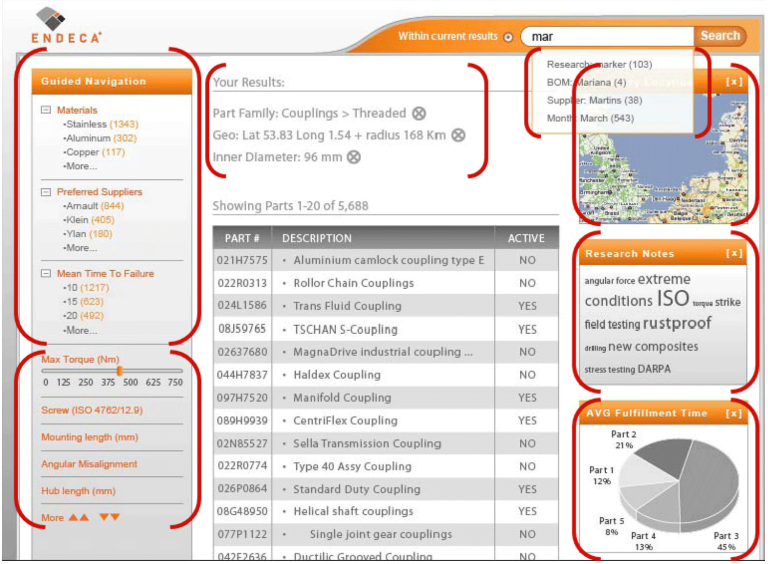


Figure 2.17: Screenshot of Endeca's Discovery for Manufacturing user interface that shows search results (center) and the guidance visualizations (red brackets) such as faceted filters, tag clouds, charts and map visualizations (Screenshot from [Endeca, 2009c]).

Summary

The Endeca IAP provides organizations with a suitable enterprise search application which enables search approaches supporting discovery and exploration in the organization's data landscape. To provide this kind of insight into the company knowledge more customization during deployment of this search application has to be done to address the specific information needs—in comparison to the "one-size-fits-all" solution such as the Google GSA.

Endeca approaches enterprise search by providing several domain-specific *search applications* which are already partially tailored to domain requirements. Considering the product development domain, Endeca offers their *Discovery for Manufacturing* suite which covers structured data in that domain. However, support for search scenarios which comprise the search for similar parts based on geometry or topology is not available.

2.7.3 Exalead CloudView™

The France-based company Exalead[60] offers CloudView™ which is their "unified information access platform" [Exalead, 2009a]. The platform was designed for Web retrieval as well as enterprise search scenarios and thus, is capable of collecting data in many different formats from many information sources (the Web, email servers, databases, intranets, multimedia archives, etc.) and automatically transforms it into structured information as far as possible.

The CloudView Technology can be utilized in different scenarios for which Exalead provides several editions of their platform. Customers can simply deploy the search engine in the enterprise as is and resort to the built-in index modules and the provided user interface using the *CloudView Search* or *CloudView 360* editions. In addition, the latter solution—at the time of writing only available as a beta version—features semantic tools to analyze the source data further and provide additional insight into the data (in Exalead parlance a "360 view" on the data).

Another scenario applies to Original Equipment Manufacturer (OEM) which can embed the search technology in their own commercial applications, such as ECM, Messaging, Information Lifecycle Management systems and others. Therefore, the *CloudView OEM* edition provides special open APIs which enables the companies to access the platform functionality.

Architecture

Exalead's CloudView is built on a SOA and provides an extensive set of APIs to access the functionality of this search solution from other applications or services [Exalead, 2009c].

Figure 2.18 on the next page shows the core components of the Exalead CloudView Platform: *Connectors*, *Document Processing Workflow*, *Index Database*, and *Front-End Processes*. The tasks which are executed in these core components are described in the subsequent sections.

Indexing and Content Analysis

The initial step for using the CloudView platform consists of collecting the designated data from internal or external sources. So-called *connectors* use the data source's native protocol to access the contained information. The connector retrieves all documents, their associated metadata and their ACLs which are all converted to an XML document, which then is sent to the *Index Server*.

Exalead provides a set of native, built-in connectors to information sources such as groupware applications, intranets, CM systems, file servers, email systems and databases. For a complete overview refer to [Exalead, 2009d]. Additionally, CloudView contains Web connectors which can work as a focused Web crawler[61] and thus

[60] `http://www.exalead.com/` (last accessed 08/01/2010)

[61] In comparison to global crawling, focused crawling only indexes documents which belong into a specific subject area. The crawler decides based on an analysis of the current document contents whether it falls into the subject area or not. This technique was first introduced by Chakrabarti et al.

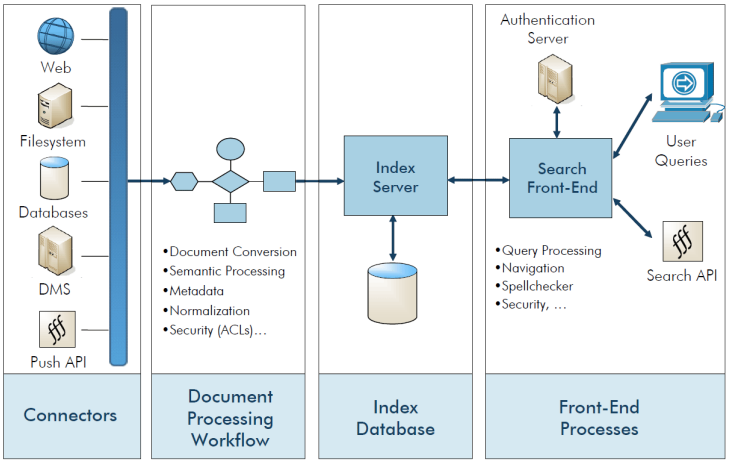


Figure 2.18: Overview of the Exalead CloudView core components *Connectors*, *Document Processing Workflow*, *Index Database*, and *Front-End Processes* (Figure from [Exalead, 2009c]).

can be built to do thematic Internet crawls around specific subjects, e. g. covering competitors, related industries or products.

For other unsupported applications Exalead provides the *Push API* (or PAPI) which can be used to add connectivity for legacy and non-standard systems. Therewith, organizations can develop custom connector components using a simple HTTP / REST protocol or a higher level API available for Java, C# or Exascript[62] to add or update documents as well as retrieving the current list of indexed documents.

After the collection of the data from the different source systems, the CloudView Platform processes and transforms the data in the *Document Processing Workflow*. The results then can be fed into the CloudView Index Server.

The document filters and their execution order in the *Document Processing Workflow* can be fully customized to match the organizations demands. The platform provides native support for over 300 document formats which can be parsed for text and metadata [Exalead, 2009a,d]. The metadata found by different connectors is then "normalized" to be consistent across different naming schemes in the source systems. The attributes of the different naming schemes are mapped to fields of the indexable document which are common to all data sources. The platform automatically determines the language of the documents during indexing (native support for

[1999].

[62]Exascript is an object-oriented XML language where Java and XML are blended provided by Exalead.

over 50 different languages is provided).

Furthermore, the processing workflow comprises the automatic analysis and classification of the gathered documents. Additional attributes such as document keywords (and their variants), entities (e. g. people, places and organizations) and metadata such as document location, file type, authors and creation date are determined. Also special processing filters try to identify relationships within and in between the found artifacts which can be beneficial during the actual search process.

Exalead tries to transform unstructured content into a fully classified resource which can be synthesized with structured content from other systems in the organization to provide new insights into the data. An interesting use case can be the interpretation of a product sales report from a BI system by incorporating Web "notoriety" statistics for this product (i. e. the results of what people are saying about this product in the Web in the sense of *sentiment analysis*[63]) as well as qualitative data gathered from support forums, email messages to product support and others [Exalead, 2009a].

The transformation is accomplished by the utilization of techniques from the NLP domain. Amongst others, Exalead employs language detection, stemming, lemmatization, morphological and syntactic processing and Part-of-speech (POS) tagging[64].

All types of gathered data is transformed into an XML document in the Exalead CloudView XML document format which is the internal representation of a document in the Index Server. This format is used to associate a unique identifier for the document with a set of named fields (such as the document content, its different metadata fields and the categories).

Back End: Index Server

The transformed data with the features necessary for searching as well as security information about the artifacts is then transferred to the central component of the CloudView platform: the *Index Server*.

One of the main functions of the *Index Server* is to maintain the index structure by processing the XML documents provided by the connector components.

The frequency of the index updates can be configured to happen in real-time, at specified intervals (e. g. hourly, daily, etc.) or "just in time". The latter checks, if the resources are current when queries are received from certain applications to guarantee current data. In addition to the index, CloudView assures that the utilized language processing modules such as thesauri, keyword recognition engines, spellcheckers and dictionaries are current as well.

[63] *Sentiment analysis* or *opinion mining* is a subarea of text mining which tries to infer the opinion of the author about a subject by statistical text analysis. Pang and Lee [2008] give a broad overview of the research in this field.

[64] POS tagging describes the assignment of terms of a text to parts of speech (so-called lexical categories such as *noun*, *verb*, *adjective*, etc.) utilizing statistical methods and techniques of machine learning. For an introduction to POS tagging see [Charniak, 1997] and [van Halteren et al., 2001]. The Stanford University offers their "Stanford Log-linear Part-Of-Speech Tagger" at `http://nlp.stanford.edu/software/tagger.shtml` (last accessed 08/01/2010).

The Index Server supports automatic replication of the index structure. When documents are added or updated to the master index, this information is automatically transferred to the replicas which are then updated as well.

Exalead provides an export option for their index which makes it possible to make its contents available for other systems as well. Among other formats, Exalead provides an XML export.

Presentation / End User Experience

Another function of the core component *Index Server* consists of the processing of user queries.

User queries with Exalead CloudView usually start by users having entered their query into a single text box. User queries can be interpreted as a *boolean search*, a *phrase search*, *proximity search* and others. For these keyword queries Exalead provides common functionality such as spell checking, phonetic matching, fuzzy queries, etc.

When the index server processes a user query, a numerical value called RSV is assigned to each search result depending on multiple criteria. CloudView considers criteria such as the relative frequency of the query terms in the index and specific indexing weights given for specific terms during indexing. User front-end applications can take control of these different criteria by giving relative weights for them and for each boolean term in the query by the "low level query language" provided by the index server. With these weight modifiers organizations can implement advanced ranking features to boost the score of specific documents based on terms or attribute values. These advanced options can be used for user-specific search results where the ranking options are adjusted based on the querying user. Alternatively, multiple user interfaces with different ranking functions and weights can be used to provide user specific result lists.

The fourth core component of the CloudView platform is the *Search Front-End* which covers the interaction with the actual users conducting searches. Cloudview offers an interaction framework which provides a built-in search interface, which is fully customizable employing Cascading Style Sheets (CSSs), JavaServerFaces[65] or Java Portlet[66] technology to integrate the search interface in existing systems such as enterprise portals. In addition to the option of giving a keyword query, the user interface provides users with a (patented) faceted navigation system which contains options for narrowing or broadening the search query as well as providing links to related content (e. g. related terms and categories).

The navigation system presents users—besides the search result list—with additional summaries which provide them with guidance for refining their search. These

[65] A Java standard for server-side user interfaces. For more information refer to `http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html` (last accessed 08/01/2010).

[66] A Java standard for interoperable user interface components displayed in web portals. For the specifications refer to `http://www.jcp.org/en/jsr/detail?id=168` and `http://www.jcp.org/en/jsr/detail?id=286` (both last accessed 08/01/2010).

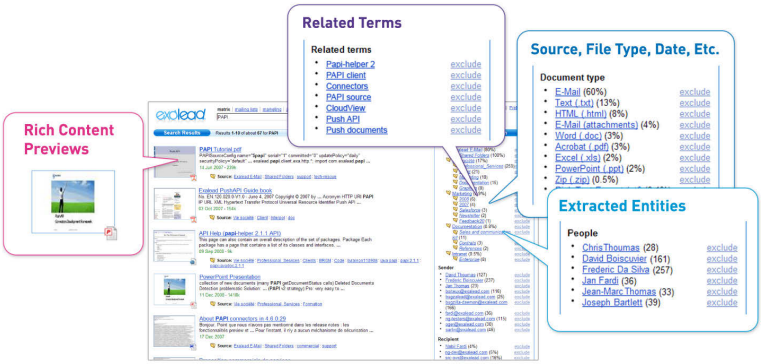


Figure 2.19: Built-in user interface widgets for search result navigation in Exalead CloudView [Exalead, 2009a].

additional query results help navigating the results by presenting related terms and concepts, user ratings, file types and size, language, author and others.

Exalead calls these summaries *table of contents* which are built for each selected metadata field. Fields considered during an email search could be the sender, the attachment types or the message date.

Figure 2.19 shows examples of the built-in user interface parts which provide user guidance, such as the content previews which are generated for documents, related terms for the textual user query and faceted navigation for the document type and extracted entities (e. g. people).

If the built-in user interface solution does not suffice the organization's requirements, Exalead's search functions can be accessed by CloudView's Search API and therewith integrated into other systems. The API supports Java, .Net, PHP, Ruby, Python and Perl and web formats such as SOAP, REST, XML, Resource Description Framework (RDF) and Web Ontology Language (OWL). Exalead proposes the creation of information-rich business applications such as information dashboards and unique content mash-ups to make the knowledge of the organization more available.

Scalability

All Exalead CloudView editions are built on the same distributable architecture which provides simple scalability by the addition of standard hardware on demand. This is realized by making the core services distributable over several servers. The configuration of all services can be controlled by the provided web interface or the Monitoring and Management Interface (MAMI) by using protocols such as SOAP or REST. Amongst others, management tasks comprise the assignment of user priv-

ileges, the control of ranking criteria, replication and backup, the modification of search result pages or the monitoring of usage statistics. Due to the ability to partition and replicate the CloudView index over several servers, a high performance and availability can be ensured.

The CloudView platform can also be used as a federated search engine where multiple search results from CloudView instances from e. g. different locations are merged. A scenario could be the federation of a CloudView instance covering the intranet and an instance used for file and email search.

The different Exalead products such as Exalead Desktop and Exalead CloudView are able to work together and thus provide a federated overview over the results from different indexes. For this scenario it would be possible to search for documents in the company and the users' computer and get a merged search result list. The *table of content* summaries of the different indexes would be syndicated as well and presented to the users in a unified way.

Security

Similar to other enterprise search solutions CloudView respects the existing security rules and only shows users the search results they have access to in the source system. The CloudView platform supports many common authentication mechanisms such as LDAP, OpenLDAP, Active Directory and Domino Directory.

Exalead CloudView supports to work with an SSO system where users only have to login on one system (e. g. the search engine) and the SSO system takes care of passing the login information to the source systems when search results are opened by the users.

In addition to the documents and its metadata, the *Connectors* also include security information such as ACLs. Security can be configured to comply with the default security policy of the underlying data source or by the security rules issued from a given security source based on users or groups such as an LDAP system.

Exalead CloudView supports both the early (in Exalead's terms called *ACL Indexing*) and the late-binding approach introduced in Section 2.6.3 on page 47 for checking ACLs. In [Exalead, 2009b], Exalead recommends a combined approach of these two which can be implemented in two ways. The first option is the usage of the *real-time indexing* capabilities which ensure that during query run-time the document's contents and the permissions stored in the index are current as in the source system. If this is not the case, the index is updated before the query is answered.

The second possibility filters the search results list comprising all found documents without respect to user permissions by the indexed ACL to a manageable size. In a second run, the remaining results are checked against the ACLs in the source systems. With this approach the situation can occur that users do not see all results due to recent permission changes which were not yet reflected in the ACLs stored in the search index and checked in the first run.

Summary

The Exalead CloudView Platform strongly builds on rich content analytics to extract and infer as much structured content from unstructured data as possible. The company strongly relies on techniques from the NLP domain. Similar to the approach Endeca takes, they are aware that many information needs in organizations need to be transferred to queries stepwise, i. e. users need facilities to navigate along structured information to refine their queries.

The platform provides a comprehensive set of APIs to access almost the complete functionality from external systems which makes it very convenient to include partial functionality in the organization's software stack.

Chapter 3

Searching in Complex Work Situations in an Enterprise Context

In this section, in addition to the review of Enterprise Search systems and their requirements in Section 2.6 on page 40 above, information needs and information behavior in an enterprise context are reviewed with a focus on design engineers. Furthermore, the characteristics for the retrieval of information in an enterprise context are illustrated. This serves the purpose to identify further requirements for a retrieval system for complex work situations.

The chapter closes with a summary of primary requirements necessary for an IR system in an enterprise context.

3.1 Introduction to Information Needs

Before examining complex work situations in enterprise scenarios, a short introduction of information needs in general is given.

First published in 1962, the concept of an *information need* was described by Taylor [1962] who identified four levels of question formulation for human inquirers.

- "Q_1—The actual, but unexpressed, need for information (the visceral need);
- Q_2—The conscious within-brain description of the need (the conscious need);
- Q_3—The formal statement of the question (the formalized need);
- Q_4—The question as presented to the information system (the compromised need)." [Taylor, 1962]

This differentiation especially highlights that users usually have to adapt the formulation of their need of information to fit the search system they query. According to Mooers' Law that adaptation should not be too difficult otherwise users just omit searching.

> "An information system will tend not to be used whenever it is more painful and troublesome for a customer to have information than for him not to have it." [Mooers, 1959]

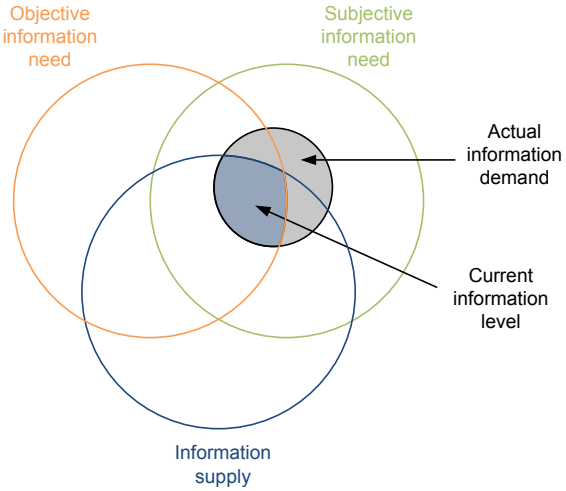


Figure 3.1: Information need and supply model based on [Wigand et al., 1997].

According to Stelzer [2001] an information need is defined as "type, quality and amount of information [...] that a task bearer (person or organizational unit) needs to perform a certain task" (English translation by author)[1]. A distinction is made between "objective" and "subjective" information needs. The objective information need describes the information from the task viewpoint, i. e. the necessary information to perform and finish the task. The subjective information need describes the information needed to finish the task from the task bearer's perspective. It is dependent on a variety of influencing factors, e. g. the current level of knowledge or the degree of structuredness of a task.

Figure 3.1 shows the different views of *information needs* in context by relating them to the actual available *information supply* and the *actual information demand* of the inquirer. The actual available information supply comprises all information which is provided by all available data sources. Usually, the information need and the information supply will not be congruent with each other. The actual requested information demand refers to the demand of a person in a certain work task. The information demand is a subset of the subjective information need.

As shown in the Venn diagram, the intersection of the objective information need, the information demand and the information supply results in the current level of information for a certain work task [Wigand et al., 1997].

[1] "Art, Qualität und Menge der Informationen [...], welche Aufgabenträger (Personen oder Organisationseinheiten) zur Erfüllung einer bestimmten Aufgabe benötigen." [Stelzer, 2001]

Another view on the topic of information needs is given by Ingwersen [1992] who introduces the *label effect*. The effect states that users are not expressing the whole information they have about their information need, but only the amount they think is enough for a human recipient [Ingwersen and Järvelin, 2005]. Therefore, search systems should try to elicit more of the available information about the users' information needs to provide better search results.

Due to these explanations, it is necessary to describe information needs found in the domain under research and describe the way users are searching for information in order to being able to design a supporting IR system.

3.2 Types of Information in Product Development Processes

To understand current issues concerning the information provision in product development processes and information needs of engineers, a quick characterization of this type of a knowledge worker is necessary. Hertzum and Pejtersen [2000] describe engineers—although they are a diverse group of professionals—as subject specialists who perform rather complex tasks. Due to the involvement of creative stages during the development of products, engineers usually have certain degrees of freedom of choosing the approach they want to accomplish their work with. Since there are many available solutions for a design problem, engineers have to make informed decisions. Their choices depend strongly on their understanding of the context of the task and therefore on the success of obtaining information about the context. Although, technical solutions and the result of the design are usually well documented in the design documentation, information about the context of the design process is often not available or indexed for easy retrieval at a later time.

In an empirical study Carstensen [1997] analyzed which information were used by designing engineers of a large Danish manufacturing company during a design project. Additionally, the chosen access to the information was examined. As the used information relates to a wide series of topics, Carstensen (based on research in [Pejtersen et al., 1997]) grouped the requested information into six expertise information areas: *technical*, *end users*, *marketing*, *maintenance*, *disposal*, and *production*. The identified information types which should be retrievable by an electronic search engine are shown in Table 3.1 on the following page.

The study also resulted in two other main findings. First, at the time of the study the main information type were potential contact persons which have special knowledge and expertise on a certain topic. Secondly, the most common type of search was similarity search, e. g. the search for similar problems, ideas, components, project types, etc. Due to these diverse information sources, a search engine should provide a means for seamless switching when several information sources are accessed.

Furthermore, Carstensen names different general requirements for an exploratory search system in the product development domain. It should provide users with easy access to different information types and enable a seamless switching

Table 3.1: Different types of information in product development processes according to [Carstensen, 1997].

Type	**Description**
Previous Designs	From past projects or from competitors and suppliers including requirement specifications, functional specifications, overall architectures, technical drawings, descriptions of methods used, etc.
Design Rationales	Information on the reasons for or against a certain solution.
"Similar products"	Information on similar products to the one currently being developed should be provided.
"Known problems" in products	Explicit descriptions of known problems, weaknesses, etc.
Component specifications	Browsing and searching on specifications of products which are considered useful in the design should be enabled. The search should include facilities for component information related to several different aspects, e.g. performance, price, basic technology, supplier, etc.
Standards and norms	Up-to-date information on the standards and norms for different types of components. This includes both company specific and national and international standards and norms.
Working procedures	Procedures, standard methods and techniques etc., which define how the task must be performed, managed, coordinated, etc. should be accessible.
Production line characteristics	Specifications on bills of materials, production tool descriptions, rationales for work place designs and salary structures.
New materials and components	Up-to-date information on new materials, technologies, components.
Literature and research results	Information on new techniques and findings from research.
Relevant persons	Information on colleagues and other persons who have knowledge or experience on special topics should be retrievable. This includes structured and easy-accessible information about persons (both internal and external).
Project documentation	General project information such as plans, specifications, drawings, communication with external partners, contracts, and agreements.

between different search strategies and information sources. This study suggests that information needs of engineers not only center around documents, but also deal with project information and persons. Thus, search engines developed for these

work situations should be able to use the relations between the different process inputs and outputs to make complex searches possible that interlink between the different artifacts.

When providing access to information it is important to support information needs which on the one hand demand very specific and concrete information but on the other hand also support more general information needs which require a more general overview over the data. The former is used to retrieve information for a specific problem or question. The more general overview serves in situations when users are browsing to learn new aspects of a general problem or to get awareness of new information. Finally, Carstensen highlights the necessity for similarity search approaches for the different information types and for the provision of search engines which help users to refine their search queries step-wise.

The examination of product development processes by chronological aspects leads to the identification of different types of information which are relevant for the different process phases. Based on different Verein Deutscher Ingenieure (VDI) guidelines (especially 2201 [VDI, 1993] and 2222 [VDI, 1997, 1982]) the planning and design process can be divided into four main phases (based on [Pahl et al., 2007]):

- **Planning and task clarification:** The result of this phase is the *specification of information* in the form of a requirements list which acts as an input for the next phases and is updated constantly.
- **Conceptual design:** In this phase the specification of the *principle solution (concept)* is developed by abstracting the main problems, determining function structures and searching for suitable working principles. These findings are then combined into a working structure.
- **Embodiment design:** Based on the results of the previous phase, engineering designers determine the construction structure of the product which should be designed. Finally, a specification of the *layout* of the construction is achieved.
- **Detail design:** The focus of this phase is the specification of the production documentation. Characteristics of the arrangement, forms, dimension and surface properties of the individual parts of the final product are resolved. Additionally, production possibilities are evaluated, cost calculations are conducted and all necessary drawings and production documents are created.

Of course, in reality product development is not as linear as described and contains iterations where phases are repeated until each result of a phase conforms to the requirements.

Figure 3.2 on the next page shows the four introduced phases and assigns different types of documents to them. The assignment of the documents to the phases was partially derived from [Pahl et al., 2007] and from an information analysis conducted with the industry partners of the FORFLOW project. The found information can be classified into different groups. The majority of the documents describe the product itself. Pahl et al. [2007] refers to these product-based documents as product

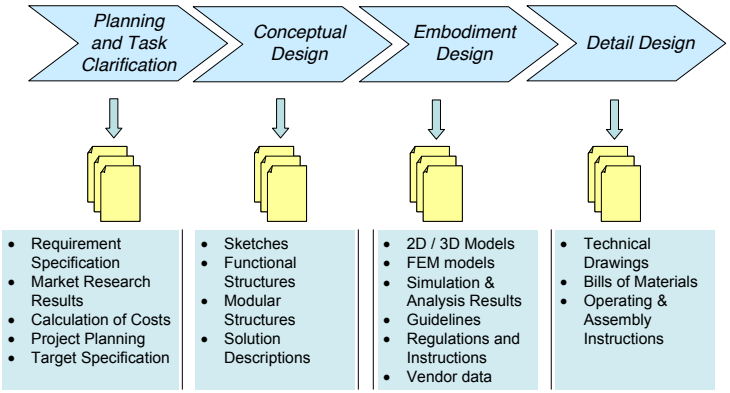


Figure 3.2: Types of information found in the different phases of a product development process.

models. This category comprises textual documents such as requirements specifications, solution descriptions and operating and assembly instructions. Additionally, special document types such as 3D Computer-Aided Design (CAD) models, technical drawings, and bills of materials are found. The second category contains documents which are used for the project management such as project plans, work breakdown structures, project hand books, and milestone plans. General documents describing regulations, instructions, norms and company guidelines are summarized in the third category of documents.

During the development of a product, engineers need access to these documents which are stored in various systems previously outlined in Section 2.6.1 on page 43. Not all of the described documents are equally relevant in every process phase. The information needs are specific to the task the engineer is working on. To leverage reuse of existing parts or products in the company, the engineers need an efficient way to get access to artifacts from past projects.

3.3 Information Seeking Patterns in Product Development Processes

Several studies have been carried out to identify information needs of engineering designers. Additionally, the accessed information and information types were examined to better describe information-seeking in product development processes.

Wallace and Ahmed [2003] show the results of two studies which examined how engineers obtain information in product development. The first study was con-

ducted in 1993 [Marsh, 1997] and studied the information search of several groups of engineers in an aerospace company. The main result was that for 90% of all queries the designers approached other persons which answered these mainly from memory which was confirmed by a study described in [Court et al., 1996]. The reason for this behavior is that engineers did not know what information is needed and therefore relied on a dialog with a person to determine the information need. This is an indication that search systems should support searches in a conversational-style as introduced in Section 2.2 on page 15. Additionally, there existed a tendency among engineers to not trust documents because they might be outdated. On average, the designers spent 26% of their working time with the search for information. These search tasks do not occur at one point in time but are fragmented throughout the workday, which is counter-productive considering the concentrated creative work needed for design projects.

The second observational study conducted in the same aerospace company focused on the question, which approach is most suitable to help novice and experienced designers to find the relevant information [Ahmed, 2001]. Thus, the different approaches for design tasks between novice and experienced developers were examined. The authors assume that novice engineers have less than two-and-a-half years experience and experienced designers over 10 years of experience in the respective domain. The main result showed that it is beneficial to guide novice engineers during engineering and during the necessary search for knowledge and information. The study identified 8 different design strategies of developers [Ahmed et al., 2003]. With respect to this publication, only the activity "Refer to past designs" is introduced here. Designers usually referred to past designs from sources like their memory, drawings, reports and colleagues. Usually, they tend to refer only to projects that were known to them, i. e. an active search for projects which might include helpful information did not take place. The reasons for referring to past designs are manifold:

- "to understand why a particular component had been used;
- find designs similar to the current design problem;
- to find designs that were similar with regard to environmental or functional conditions, and to understand how the components had behaved in these conditions [...];
- to find projects where similar problems had arisen and how they were resolved;
- to use as a starting point for new designs and to ensure consistency between designs;
- to obtain numerical data for the current problem, e. g. to estimate a weight, a designer used the ratio of the weight and the volume of a past design." [Ahmed et al., 2003]

The main findings were that designers prefer to get information from other persons, since they consider this a more effective and efficient approach rather than

querying a retrieval system. By speaking to a person, the designers can add further contextual information to their information need. The approached person then can help to rephrase and state the query more precisely. It has been shown that the initial query often was not identical to the query in the end.

Given that the first study was conducted in the early nineties, it has to be noted that since the publication application system support grew steadily and it is assumed that nowadays a similar study would not convey this fear of outdated information. Nevertheless, these studies show clearly that engineers, especially novice engineers, need support in phrasing their search queries to efficiently retrieve the necessary information in an exploratory way which enables engineers to incrementally define search queries.

Zipperer [1993] gives three reasons why engineers prefer getting information from their colleagues.

- Engineers are seeking feedback on ideas or designs, either as trusted opinion or as impetus for creative discourse.
- Colleagues act as a pointer to information which they remember, which is far easier than manually looking through files.
- Due to close working relationships engineers are able to pick the most promising person as an information source.

Allen examined the differences between engineers and scientists with respect to information seeking in a series of publications [Allen, 1988, 1995]. To illustrate the differences he makes use of an input-output model to explain the relationship between design documentation and oral communication. Scientists use information to produce new information, which denotes that the input and output of a task are compatible so that the output of one stage can act as input for the next one. Information supply can be seen as a matter of collecting and organizing these outputs and making them accessible.

On the contrary, engineers consume, transform and use information to make a product which is information-bearing but does not provide the information in verbal form. Verbal information is produced as a by-product in form of design documentation. The type of the output is usually different from the type of the input and therefore cannot serve as direct input in the next stage. According to Allen the documentation is often not sufficient alone and needs the responsible person for additional context. “Thus, technological documentation is often most useful only when the author is directly available to explain and supplement its content” [Allen, 1988].

Hertzum and Pejtersen [2000] examined the information seeking practices of engineers. The authors conducted two different case studies. The subject of the first study was how engineers acquire information from previous internal projects as well as from external sources. The second study observed one single project in the same company and recorded which information was needed and how it was acquired and retained. The main conclusion the authors drew, was that “engineers search for documents to find people, search for people to get documents, and interact socially to get both oral and written information without engaging in explicit

searches" [Hertzum and Pejtersen, 2000]. The reason for the strong use of persons as mediators to retrieve information lies in the fact, that—according to the authors—the product documentation (seen as the by-product of the process) does not convey enough contextual information about the rationales behind the design.

The authors emphasize the advantage of written information with respect to retrievability through search engines and propose that additional information about the development of this information should be made available through a search engine. This could help finding persons with expertise for the current work task.

Fidel and Green [2004] conducted a study to understand which factors play a role for engineers in selecting sources of information. In particular, they identified the two notions of accessibility and quality of an information need. Additionally, they tried to clarify the notion of *accessibility* of information sources. In many studies, this term was stated as the main factor for the engineers' choice of an information source, but is used inconsistently (e. g. refer to [Leckie et al., 1996] and [Pinelli et al., 1993]).

The study resulted in the identification of several factors of accessibility which influence the choice of an information source by engineers. The main factors will be introduced briefly and implications and requirements for the design of a search engine relevant to this thesis are described. Factors solely referring to human sources (e. g. experts) are omitted in the following list. The main factors are:

- **Sources I know.** Engineers tend to use sources that they used before in past projects due to the fact of being familiar with that source which involves reduced efforts to take advantage of the contained information. Thus, a search engine should aim for making new yet unknown sources to users familiar by providing a unified search interface which covers several systems.
- **Has a lot of different types of information in one place.** Engineers prefer sources which offer multiple information types since these sources make a retrieval over many systems unnecessary. A retrieval system should federate several source systems and offer a user interface which searches these systems in parallel and merges the results in a consolidated ranking.
- **Can give the right level of detail.** This factor centers around the granularity of the search results users need in different situations. A search engine should provide users with tools to get an overview of a set of search results as well as to provide detailed information about each search result.
- **Saves time.** It is important for an information source to provide users with the right information in a timely manner. Thus, a search engine should execute search queries efficiently.
- **Has the right format.** Engineers preferably use sources which contain information in formats they already know or have used before. Thus it can be beneficial for users, when information access systems provide the functionality to give users the same information in different formats, e. g. by automatically converting a document to the preferred document type.

- **Can be searched with keywords or codes.** Users prefer sources which offer search functionalities for easier retrieval of information.
- **Is interactive.** The factor of interactivity of an information source is important to engineers when their information need is not clearly defined. By being offered search tools which allow conversational-style searches users can refine their the query step-wise until they narrowed down the search results to the needed information.

The study of enterprise search behavior of software engineers in [Freund and Toms, 2006] identified four categories which help users to determine the usefulness of an information, namely content, format, currency and authority. The participants assessed the *content* based on the topic, the level of specificity of the information and the degree of situatedness, i. e. if the information was related to a particular case or scenario or whether it is a generic information. The *format* criterion was assessed by the genre of the document, its structure and the possibility of interaction with the authors. *Currency* was defined by the creation date and the version of the product or technology. Lastly, *authority* was measured based on acquaintances, reputation or team affiliation of the author and the organizational source of the information [Freund and Toms, 2006]. These findings are similar to those from [Fidel and Green, 2004]. Modern user interfaces should support users in providing them with these types of information for a quick assessment of the usefulness of a document.

Additionally, the study resulted in the definition of some requirements for enterprise search. Searchers need to be provided with tools which support the creation of complex queries and make use of specialized (domain-specific) terminology. Additionally, the provision of facilities which help sorting and limiting the document collection by multi-dimensional structured data is highly recommended (cf. Section 2.3 on page 18 for an overview of faceted search and Section 2.4 on page 24 for according user interfaces).

Byström and Järvelin [1995] found in a study in a public administration context that there exists a dependency between the complexity of a task and the according information needs. The main findings were that as task complexity increases,

- "the complexity of information needed increases,
- the needs for domain information and problem solving information increase,
- the share of general-purpose sources increase and that of problem and fact-oriented sources decrease,
- the successfulness of information seeking decreases,
- the internality of channels decrease, and
- the number of sources increase." [Byström and Järvelin, 1995]

In two studies del Rey-Chamorro and Wallace [2003, 2005] examined the retrieval of information in the aerospace domain. Both studied how designers use existing product documentation in a new design project. The main finding was that designers still preferred paper-based documentation over electronic resources.

Users claimed that—especially in collaborative work situations—paper provides a stronger and more accessible platform and allows easier face-to-face communication. When accessing documentation, designers preferred to browse through ring binders. To navigate through different sources, the designers usually referred to indexes and cross-references. Especially for novice designers dealing with a task or type of component for the first time, browsing through folders helped them to identify other issues relating to the current design.

More experienced senior designers also searched for standards and manuals, but additionally reverted to other information sources such as past projects. By finding examples in past projects they derived a detailed description of the required type of documentation. Although these studies showed that designers often search for previous designs, their intention often was not to reuse parts of previous designs, but to become aware of the issues and the documents related to a certain subject.

The studies of information behavior in an enterprise context helped to identify requirements for an IR system for that domain.The findings of different studies that engineers still prefer to browse through paper-based documentation (for instance [del Rey-Chamorro and Wallace, 2005]), is a clear indication that search options are not sufficient right now.

Unlike in safety-critical domains like Air Traffic Control (ATC) [MacKay, 1999] or in the military [McGee et al., 2002], where paper-based documentation is preferred due to the robustness to failure and flexibility, in the author's opinion the acceptance of electronic documentation can be increased by providing better IR systems, which support users in accessing different information in an electronic tool.

3.4 Exemplary Search Scenarios

This section aims to introduce some exemplary search scenarios to demonstrate which types of search tasks exist in an enterprise environment.

3.4.1 Search for Existing Parts in the Organization

The starting point of the first scenario is the development of a current generator which is composed of numerous components, such as screws, cooling elements, insulating plates, fans, and others. For instance, if engineers are situated at the step *component design*, more precisely in the *design* stage of the product development process, they have to create CAD models for certain parts of the new product. A search engine can support them by providing customized search options.

A search task could proceed as follows: Different types of query alternatives are available to engineers. They can search for documents in the classic way by giving a *keyword query* (e. g. "fan" or an according part number). This type of search returns documents containing the keyword or documents which are assigned this keyword (e. g. attributes from a PDM system). Furthermore, users could query the search engine by providing an example object/document. For instance, engineers describe their queries in form of a 2D-sketch, which represents a view of the required

product. This representation is then automatically compared with other views from existing technical drawings in the search engine index that were developed in earlier projects. Alternatively, the engineer can use a rough 3D-model of the fan as a query object which then is compared to already existing 3D-fan models. The result of this query is a list of similar parts which are already existent in the organization or are available from parts catalogs. As a drawback the result list also can incorporate gear wheels since they are—at least concerning their geometry—similar as well.

To circumvent that these inapplicable parts are included, the search result set can be optimized by considering additional contextual factors for the search. This information helps to improve the search results by filtering those documents or parts from the results which are inapplicable for the current work situation. These contextual factors can be partially derived from documents of previous process phases. The consideration of the function of a part (which is defined in an early process phase called *function structure*) can help to exclude geometrically similar parts such as gear wheels. By providing users with a faceted search user interface, they are able to apply special search criteria such as Design for X (DfX)-criteria[2], tolerance data (e. g. for temperature ranges), or specifications of dimensions for filtering. Furthermore, by employing project management software the process model can be harvested for process context information about users. By accessing the process and task model the search engine can determine which artifact types are relevant for users in their current work task. Thus, the search query can be augmented and enhanced by this information which narrows down the search result further.

Engineers can then choose an appropriate part for their current work problem from the search result. This part can be used in its entirety or can simply be modified to suit the changed requirements in the current project. Furthermore, the search can contribute to reduce the amount of work necessary. By re-using a fan which is already approved in the organization, necessary calculations can be omitted in certain situations since the fan was already tested in its original development.

A modification of this scenario was examined during the requirements analysis with the industrial partners of the research project FORFLOW. There, engineers were looking for a certain product. Not the product itself was their main objective, but the supplier who delivers it. The engineers wanted to know which other products were offered by that supplier to achieve higher order sizes. This kind of information need demands the combination of different queries which are connected through interim search results. Users first need to restrict the set of products to the desired ones and based on them query for the suppliers of these products. These complex search situations are difficult to support proactively by current search engines, since these often only focus on one type of search results.

[2]DfX describes a collection of specific design guidelines for main requirements denoted by the variable X. X can be substituted with, for example, *Cost*, *Reliability*, *Environment*, *Manufacture* and others [Pahl et al., 2007]. Each guideline provides methods, strategies and tools to realize the chosen X.

3.4.2 Project Reviews

The second search scenario addresses the evaluation of past projects. After finishing a project in an organization, usually reviews are conducted to record best practices and lessons learned during the execution of a project. Taking the example of the analysis of a product development project, a search scenario can be the assessment of how successful different methods were applied during a project by its members. An example for a method from this domain is the utilization of a morphological box[3] to determine possible solutions for a design. This analysis can help to improve future project outcomes by providing better method documentation or training on these methods.

This scenario assumes that projects are supported by project planning systems which help users in proceeding in a project. These systems support document management and enable users to attach results, such as documents and notes to the corresponding process steps[4]. These systems usually rely on a pre-defined process model with which they are parameterized to provide the mentioned guidance.

For the scenario the relations between the classes *process*, *process step*, *methods* and *documents* as depicted in Figure 3.3 on the following page have to be considered. A project planning software as described above can generate and store these relations which then are utilized in search tasks.

For instance, the analysis over this data could be started by a user query which filters for documents of type *review* and by a time interval (e. g. the last half-year) by faceted filtering. Additionally, the resulting set of documents could be narrowed down by a keyword query. In the next step, the search result set containing documents is taken as the basis for further queries. By switching the artifact type to the project level, all projects linked to the current set of review documents are determined and presented to the users. The remaining set of projects then can be filtered by project evaluation measures such as the degree of maturity, the degree of fulfillment of milestones, the compliance with dates or the number of occurred errors in the project result. This serves the next step, where the according process steps of the projects are shown to see where problems occurred with the applied methods.

Figure 3.4 on page 83 shows different artifact layers and relations in and between these layers which have to be considered for this search scenario.

Alternatively, the result of a query for this search scenario could be the display of all types of methods which were used for a set of selected process steps.

3.5 Multi–Criteria Search

As described in the sections above, the information landscape in enterprise scenarios is affected by a mixture of un-structured and structured data. Furthermore, infor-

[3] The morphological box is the result of a morphological analysis which helps to explore solutions to a multi-dimensional, non-quantified problem complex [Zwicky, 1969].

[4] An exemplary system is the project planning portal *Stages* from the company Method Park Software AG (`http://www.methodpark.de/` last accessed 08/01/2010) who was an industrial project partner of this research and provided the software for research purposes.

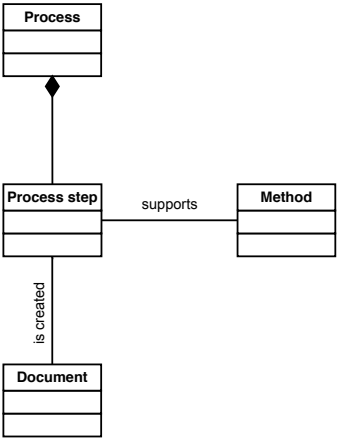


Figure 3.3: Different artifact types and their relationships used in the second search scenario.

mation needs can be quite complex and thus, different contextual information of the users should be considered for augmenting search queries.

The available contextual information about users and artifacts as well as the structured data provide users with a wide variety of search criteria they can use in a retrieval task. In [Eckstein and Henrich, 2008a] an integrated context model for the product development domain was introduced to show the variety of contextual information which can influence user information needs. Figure 3.5 on page 84 shows the different identified dimensions of the context model. The model captures information about the dimensions *user*, *document*, *product*, *process*, *project*, *company* and *task*. Nevertheless, the availability and concreteness of each individual criterion has to be evaluated for each organization, i. e. the applicability and ways of acquisition must be assessed. Whereas criteria such as current process phases or the current project are more easily captured by using project planning software, the integration of criteria from the company model appear more difficult. The contextual information of the different dimensions can describe artifacts in addition to their actual contents[5].

Traditional IR approaches such as the Vector Space Model [Salton et al., 1975] or the probabilistic retrieval model Okapi BM25 [Sparck Jones et al., 2000] regard relevance as a single score by aggregating basic features of the query and the artifacts in question. The applied features usually cover the "topical relevance". For instance, these features comprise the *term* and *document frequency* for the example

[5]The notion of artifacts which is used in this publication is introduced in Section 4.2 on page 94.

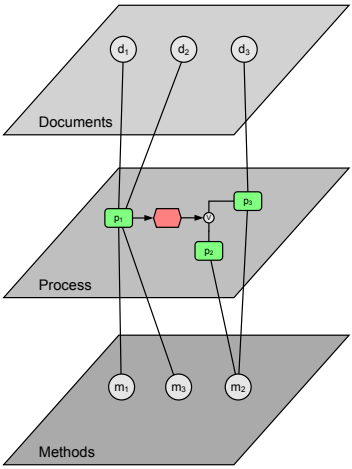


Figure 3.4: Considered artifact types in the second scenario.

of the Vector Space Model. By relating the single score of a document to the query, the RSV can be computed which helps building a relevance ranking of the artifacts. These relevance computations are static and do not adjust to different search situations.

In contrast, several studies advance the view, that there does not exist a single concept of relevance, but many kinds of relevance [Schamber et al., 1990; Borlund, 2003a] (also refer to Section 2.1 on page 11). Moreover, the concept of relevance cannot be explained by considering only one source of evidence. The combination of multiple sources of evidence helps to significantly improve retrieval effectiveness. This comprises several features of an artifact which are used to determine relevance as well as the integration of contextual information about artifacts.

Considering the example search scenario from Section 3.4.1 on page 79, the search for reusable parts for the current project can depend on several sources of evidence. One feature would be the search for geometric similar parts for an example object which returns a ranking of similar parts according to this feature. Usually, engineers possess more available information about the objects they are searching for. The search for additional keywords or the filtering of objects based on size restrictions can help to narrow the search results. Each of these features produce a document ranking which then has to be merged to the final ranking. This combination of many sources of evidence has proven to be superior than the consideration of a single source (for instance in [Belkin et al., 1995]). The automatic merging of multiple result rankings for the different sources of evidence is called *rank aggrega-*

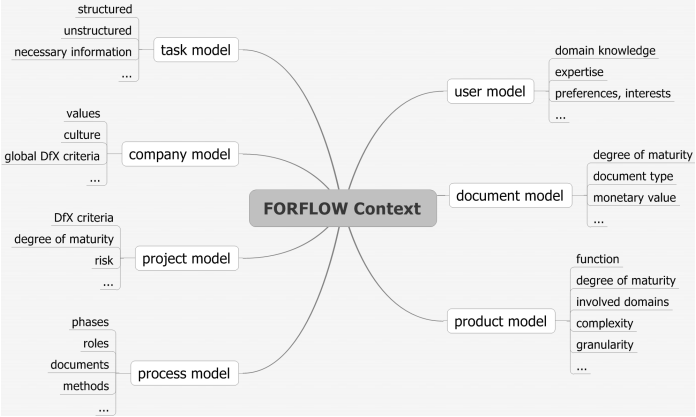


Figure 3.5: Context dimensions for the product development domain [Eckstein and Henrich, 2008a].

tion[6] in the literature. The search engine determines multiple rankings based on the same data collection but uses different approaches to determine similarity for each considered feature. The final ranking of artifacts is then determined by combining the different rankings.

The literature proposes several mechanisms to combine multiple rankings. Shaw and Fox [1993] use the *min* and *max* aggregation operators to combine results from different search schemes and document collections. Since these non-compensatory operators only consider one criterion value such as the worst or best score of the artifacts in the different rankings, a lot of available information about the scores is lost.

Craswell et al. [2005b] consider adding various query-independent features as relevance weights in addition to a baseline ranking such as BM25 to improve relevance ranking. For instance, the authors consider query-independent features such as PageRank [Brin and Page, 1998] or the document length and linearly combine them with the baseline score. The query independent features are applied with a sigmoid transform to receive relevance weights which according to their study is a successful approach to include these features in the final ranking.

Other approaches use aggregation operators such as P-norms [Salton et al., 1983; Robertson, 1977] and fuzzy logic conjunctive or disjunctive operations [Yager, 1988; Dubois and Prade, 1985].

Yager [1988] introduces the use of Ordered Weighted Averaging (OWA) aggregation operators for multi-criteria decision making. When considering the aggregation of multi-criteria, two extremes in the relationships of the criteria can be identified.

[6]The term *data fusion problem* can also be found.

On the one hand, it can be necessary that all given criteria are met. On the other hand, the decision can be made if any of the given criteria is satisfied. These two extremes can be satisfied by the usage of conjunctive (logical AND) or disjunctive (logical OR) operators. The main goal of this work is the proposal of an aggregation operator which lies in between these two extremes. In contrast to the classical weighted average, the introduced operator uses coefficients which are associated with an ordered position rather than directly with a particular attribute.

Farah and Vanderpooten [2006, 2007] propose a multiple criteria framework for IR which uses an aggregation mechanism which is based on decision rules identifying positive and negative reasons for the rank determination of a document.

The approaches introduced above all have in common that they strongly rely on query dependent criteria such as term frequency (tf) and document frequency (df) and do not consider additional search criteria based on structured information such as metadata. Additionally, these approaches do not incorporate users in stating preferences to influence the determination of the ranking.

Belkin et al. [1995] showed that it is useful to combine several different representations of a single information problem with respect to search result quality and not just identify the best search representation. These representations can comprise evidence from different retrieval techniques, different document representation techniques, or even different IR systems. Belkin et al. emphasize that this combination of different query representations is applicable for two scenarios. It can be used to combine evidence from within the same system (also called *query combination*) and from different systems (called *data fusion*). For the latter scenario Belkin et al. point out that there are still open issues in combining evidence from multiple systems due to the different applied computations of similarity scores. The described successful experiments used adaptive weighting for the combination of the different similarity scores based on a test dataset. Another major finding was that documents that appear in most of the rankings (but were derived by different representations) are more likely to be relevant. Additionally, Lee [1997] found that although various retrieval techniques often return different irrelevant documents, many of the retrieved relevant documents are the same.

A difficulty of the referenced rank aggregation methods is that users cannot easily comprehend the underlying aggregation methods and thus lack the ability to interpret the search result ranking. As illustrated in Section 2.1 on page 11 a search engine should provide users with *transparency* so that they understand why a particular response is returned. Thus, the aggregation of different ranking criteria should be comprehensible to users. Moreover, the described approaches themselves choose criteria to describe the users' information needs without giving them the opportunity to interact with the search engine. The case when users themselves want to state their search criteria is not considered in these approaches.

The inclusion of multiple search criteria—both attributes of artifacts as well as contextual factors describing an artifact—evolves the search mechanism from single-criterion matching[7] to multi-criteria matching which introduces several diffi-

[7]For instance, comparing the textual document contents with the query.

culties concerning the processing of this additional information and should (at least partially) be represented in a search user interface.

These are:

- **Determination of weights for the different search criteria.** Depending on the search situation different search criteria can be used. After the choice of the appropriate search criteria, the weighting of the different criteria needs to be determined i. e. which criteria are more important in the current search situation than others. Considering the search for 3D-models of a part which can be used in the current construction, users might put more weight on the geometric similarity than on the recency of a file which is returned.
- **Description of the dependencies between the search criteria.** The different available search criteria for a certain domain are not independent of each other, but condition each other. For instance, the choice of the document type usually determines certain document formats. The choice of a certain product group unlocks the usage of different specific search criteria, e. g. the criterion of the diameter for screws.
- **Determination whether a search criterion acts as simple filter or influences the ranking.** The utilization of a search criterion can serve two different purposes. The selection of a value of a search criterion can simply filter all artifacts which comply with this criterion according to the Boolean search model. Additionally, selections on a search criterion can help to influence the search result ranking in contrast to the previous approach. Artifacts adhering to certain values of a search criterion can be elevated in the search result ranking.

Different solutions to these problems are conceivable. One solution is the application of methods of machine learning [Mitchell, 2008] to automatically determine weights and dependencies between the available search criteria and the contextual information. One big disadvantage of these approaches is the necessary learning set to train the system before it can be applied. Due to the breadth of the researched domain and the scope of the search engine, it is claimed in this thesis that no automatic definition of the weights can be achieved as the information needs in the portrayed search scenarios are too diverse.

Alternatively, human experts could select appropriate search criteria and according weights for different search situations. The obvious problem with this approach is that it cannot easily be ensured, whether these relevance assessments are transferable to other users and how good their quality is.

Thus, the third possible solution consists of handing over the control to end users of the search engine by letting them decide which contextual factors and search criteria have to be considered for the evaluation of the query. The calculation of the ranking is then based on the users' choices. Users in enterprise environments can be in different work contexts throughout a work day due to multiple projects they work on. These multiple contexts complicate an automatic approach due to the frequent task switches. This thesis advances the view that users themselves can be decisive

about the search criteria, if the user interface is intuitive for them and supports them properly in the process of query formulation and search result analysis. The search engine should strive for providing *guidance* through the search process, but still offer users *transparency* and *control*. This advanced support is necessary due to the higher cognitive efforts users have to undertake in searching. Although this approach moves much of the complexity of query elaboration to the user, this more technical approach to a search user interface was chosen since the end users in the portrayed domain are technically knowledgeable.

3.6 Summary of the Identified Requirements

The previous sections identified several requirements for a search solution aiming for the support of the provision of information for users in an enterprise environment. In this section they are summarized and serve as requirements for the search framework introduced thereafter.

Main requirements for a search framework in an enterprise environment are:

- **Support multiple information types.** Information needs in an enterprise setting circle around different artifact types which should be supported natively in a comprehensive search solution. Thus, multiple query mechanisms need to be supported to apply the according similarity measure for each artifact type. Example artifact types for the considered domain are *documents*, *projects*, *products* and *persons*.
- **Support the relations between artifact types.** The different artifact types are linked by relationships. The search engine should gather these links and offer them as navigational options for users. Therewith, complex information needs can be satisfied by traversing the relations between artifacts types.
- **Support heterogeneous information spaces.** The enterprise software landscape can be characterized by a variety of different systems that store information which ranges from unstructured to structured data. The search engine needs to access these systems to index the information. This helps to provide a single point of entry for users which need to access information from these systems.
- **Support different granularities of search results.** The granularity of information needs varies in different situations. Therefore, a search engine should support users in answering more specific as well as vague and more general information needs. This can be achieved by offering different search result aggregations. For instance, search user interfaces can help to understand search results better by providing users with visualizations that aggregate the results and serve as an overview.
- **Support exploratory searches.** Information needs which cannot be formulated completely in the beginning of a search task necessitate mechanisms which help users to refine their queries step-wise. The search engine should

try to establish conversational-style searches where users can incrementally refine and rephrase their queries. This can be supported by query previews and techniques from the field of exploratory search. Furthermore, users need support for creating complex queries which involve queries-by-example with a customizable weighting of contextual information and other facets. Additionally, users should be provided with support for saving search tasks due to the fragmentation of search tasks to help them reassess past searches and to pick them up later.

- **Support query and search result visualizations.** Several information needs cannot be satisfied by providing users with a simple one-dimensional search result list. More complex information needs might necessitate the provision of data visualizations which help users to understand the data collection better and to provide them with guidance through the query formulation process.
- **Unified metaphor for query elaboration.** The search engine should offer the diverse query options in a unified way by introducing a metaphor helping users to easily comprehend the query options which can be very different based on the currently used artifact type. Examples are the search for text documents vs. the search for similar geometric objects. The user interface should provide a similar way to conduct a query and show the results.
- **Federate several source systems.** A successful search engine needs to federate several source systems to allow users to simply query one (search) system and retrieve results from all connected systems.
- **Similarity search.** A search engine should support similarity searches for different artifact types and provide users with a unified way to conduct these queries. Similarity for an artifact type requires different features and thus different algorithms. A query for a geometrically similar object is determined different to the search for similar text objects.
- **Provide support for different user groups.** A search engine should distinguish different user groups (for instance standard and expert users) due to different information seeking patterns. Furthermore, for a broad acceptance of the search engine low entry barriers have to be ensured. Users should easily be able to formulate their queries to satisfy their information needs. Nevertheless, more experienced users should be provided with more sophisticated search tools to foster all available search options.

Part II

Retrieval Model for Complex Search Situations

Chapter 4

LFRP–Search Framework

In this chapter, the generic LFRP-search framework is introduced and defined. The framework supports users in coping with complex search situations as described in Chapter 3 on page 69. The acronym LFRP represents the four constituent parts of the framework: Multi-**L**ayer **F**aceted Search with **R**anking using **P**arallel Coordinates.

The *multi-layer* functionality supports users in defining complex search queries for information needs when coping with multiple artifact types. Users are given a tool to switch artifact types based on interim search results of a related layer such as switching from a set of products to documents which describe these products. The search paradigm of *faceted search* supports users in stating queries incrementally by refining them interactively with filter criteria. This search paradigm usually only returns sets of search results based on the applied filtering. The LFRP-search framework introduces user preference functions to influence the order of precedence of search results—the so-called *ranking*. To give users the control over this retrieval model, an intuitive user interface is introduced which applies the visualization technique of *parallel coordinates* as a tool for stating queries[1].

These four pillars of the LFRP-search framework are defined generically in this chapter, whereas the following chapter shows the realization of this framework. The framework is built on the basis of different search paradigms and comprises aspects of ranked retrieval, exploratory search, faceted navigation, dynamic queries and target search.

Section 4.1 on the following page gives a brief overview of the user interface to make the description of the generic framework more comprehensible to the reader. Section 4.2 on page 94 defines the notion of *artifacts* with respect to the LFRP-search framework. Section 4.3 on page 100 defines the fundamentals for the supported query concept and details the different query and facet types. The introduction of the layer concept allows the search on multiple artifact layers and enables the formulation of complex search queries. Additionally, the LFRP-query model is defined semi-formally in an SQL-like notation. Both the query options for single sub-queries on one artifact layer as well as the combination of search criteria from multiple artifact layers are specified. This chapter closes with the introduction of the concept

[1] An initial sketch of the framework was introduced in [Eckstein and Henrich, 2008b].

of *user preference functions* which enable users to influence the search result ranking by adding preferences for certain facet values. Section 4.4 on page 118 covers the feature of dynamic facet provision which determines the facets which are currently available depending on the users' queries.

One main criterion during the conception of the framework was the generic approach to make the framework adaptable to different domains in order to support users that face complex search situations. Before the deployment the generic framework needs to be customized to specific environments by the definition of a facet schema which will be detailed in Section 4.5 on page 121. The realization of the framework is shown in Chapter 5 on page 127.

Although every search system can be characterized as interactive, the *interactive* characteristic is especially highlighted for the LFRP-search framework. Interactivity in general can be described as an interrelationship between two entities which exchange information[2]. In this publication *interactivity* is not simply covered by having users provide search queries which are answered by the search engine with a search result. Here, use cases are covered where users are communicating with the search engine and vice versa. The retrieval model comprises different features which help users to refine their search query. The framework tries to establish a query elaboration dialog where users are guided through the data in the enterprise by interactive features of the search engine[3].

On the one hand the dialog relies on input from users giving a representation of their information need and on the other hand the search engine helps users to elaborate their queries. This is especially helpful for users when their information need is still vague and they need to explore the data collection. The features of the LFRP-search framework aim at giving users a tool set to describe their information needs incrementally to the search engine. The search engine then returns information tailored to the current situation of the users to help them focus on their information needs. The provided user interface helps users to understand how the search engine determined the current search results in the ranking, thus providing both guidance and transparency for users.

In summary, the use of the characteristic of *interactivity* refers to the query refinement which is explicitly supported by the LFRP-search framework where users are supported after each query step.

4.1 The LFRP–Approach to Query Statement

This section introduces the approach for stating complex faceted queries visually on the basis of the LFRP-user interface. The intuitive search interface allows the

[2]Rafaeli [1988] defined the concept of interactivity as "an expression of the extent that in a given series of communication exchanges, any third (or later) transmission (or message) is related to the degree to which previous exchanges referred to even earlier transmissions", i. e. the two parties involved in the communication are relating their messages to each other.

[3]The metaphor of a dialog for the query elaboration process was borrowed from Daniel Tunkelang. `http://thenoisychannel.com/2008/09/03/query-elaboration-as-a-dialogue/` (last accessed 08/01/2010)

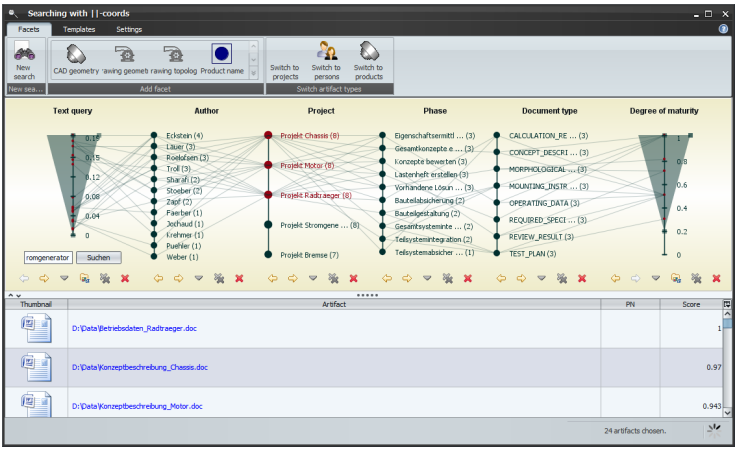


Figure 4.1: The LFRP–user interface.

interactive and incremental search query refinement which is a major requirement for the support of exploratory searches. This introduction aims at providing a better understanding of the formal description of the framework. Section 5 on page 127 will provide a more detailed look at the prototypical implementation.

The LFRP-search framework is a general approach to support knowledge workers in retrieving information which is needed or helpful in the current task for the current information need. An intuitive user interface supports knowledge workers by providing an interactive and flexible search tool.

Figure 4.1 shows an example screenshot of the search user interface after several facets have been chosen and selections on these facets have been applied. The user interface is divided into three horizontal parts. The topmost part consists of a ribbon component which enables users to apply additional facets to the parallel coordinates plot. The list of available facets is adjusted after each query refinement step. This *dynamic facet provision* feature relies on a company-specific artifact and facet schema. Roughly spoken, this schema defines the inheritance hierarchy of the artifacts together with the discriminative attributes (called *discriminative facets* in this context). As soon as a value for a discriminative facet is selected the additional facets of the corresponding subtype become available in the "Add facet" area. For further information see Sections 4.4 on page 118 and 5.3 on page 134.

Additionally, the switching of the considered layers can be conducted (cf. Section 5.3.4 on page 144) and general options for the user interface can be set.

The central part of the user interface consists of a parallel coordinates plot consisting of a dynamic number of axes which represent the different facets. The multi-dimensional faceted metadata is mapped to a two-dimensional area which can be charted and comprehended easily. Each axis can be adjusted by a control panel

situated below the axis which enables swapping, removing and weighting of the corresponding facet. Each chosen facet is drawn as a vertical axis with markers for the currently visible facet values. For instance, the facet values of nominal and ordinal facets can be sorted lexicographically or based on the facet value counts. For metric values users can quickly see how the values for this facet are distributed within the current search result set.

In addition, each artifact in the search result is plotted into the parallel coordinates plot as a polyline. This polyline intersects all axes at the appropriate facet values for the current record. The connecting polylines of two adjacent axes can be used to determine relationships and correlations between facets and artifacts.

Users have the option to add facets to the parallel coordinates plot in order to use them for further analysis of the search result. Moreover, they can use the axes to specify a sub-query. By selecting specific facet values of an axis, users filter the search result set to those artifacts that fulfill this constraint. The plot is adjusted immediately to reflect the modified data in the reduced result set.

The lower part of the user interface simply shows the search results in a list view which can be sorted and allows users to open the search results in a source application or e. g. in a DM system. Additional information about the graphical implementation of the user interface can be found in Section 5.3 on page 134 and in [Eckstein and Henrich, 2009].

The LFRP-search framework comprises—in addition to the concept of faceted search—content-based full text and QbE type queries. The latter takes an example artifact, e. g. a document and retrieves similar documents according to artifact specific similarity measures (e. g. the vector space model for text documents [Salton et al., 1975]). Additionally, users can search for similar projects in the company based on requirements of their current project. These *similarity facets* are displayed as a facet axis as well. The "Text query" in the leftmost axis in Figure 4.1 on the preceding page gives a simple example for such a similarity facet. Similarity searching returns a ranking of artifacts which is ordered according to a similarity score or a distance of an artifact [Zezula et al., 2006]. This score is used to arrange the markers on the axis. Users can simply understand that documents whose markers are drawn higher on the axis are more similar to the given example object.

The statement of the query was harmonized by completely integrating the different search types in the parallel coordinates plot in which the query can be composed. Users simply need to understand the notion of an axis in the plot referring to a sub-query for their information need which they combine with other axes.

4.2 The Notion of Artifacts

4.2.1 Description of Artifacts

One of the main characteristics of the LFRP-search framework is the support for search queries which cope with multiple *artifact types*. In the present publication an artifact is defined as an information carrier or an information object, which can be

returned as a part of a search result in a search process[4]. Each information object is assigned an entity type that describes the associated information of each considered object.

The *principle of polyrepresentation* [Ingwersen, 1994] from the field of Information Seeking and Retrieval suggests that the information need of a person should be represented by a vast variety of influencing "features". The same applies to the artifacts in the index, i. e. their representation is not complete when only the actual contents are taken into account. Therefore, a search engine needs to combine different aspects of an artifact. For example, a 3D similarity comparison should include information about geometry, topology, material properties, and metadata of an artifact to be able to provide improved retrieval results.

In general, an artifact can contain a certain artifact content. Considering documents, the content can be—for instance—a textual or a geometrical description of a product. Additionally, various metadata can be collected from different source systems which provide other views on the artifact. If the artifact comprises tangible contents, the indexing framework might extract additional metadata about that specific artifact or might identify related artifacts. These relationships are indexed and can be used for information needs spanning multiple artifact types as well.

Furthermore, especially in enterprise environments there often exist different management systems which can provide additional metadata about the managed artifacts. PDM systems provide facilities to link documents which contain partial product descriptions to product data. Although, the documents itself can contain specific product properties such as the author of a document and the document format, they might also be stored in the respective management system. DM systems provide additional information about the *release status*, the *document type*, the *access rights* and process information about artifacts. In such cases, rules need to be set which define the valid information and thus should be used for indexing. A more detailed description of the necessary indexing steps can be found in [Weber et al., 2009].

The analysis of the contained information in the artifact content and the possibilities to extract it, strongly depend on the structuredness of the data (structured vs. un-structured content). Although the actual indexing and processing of the data is out of scope of this publication, Section 2.6.1 on page 43 above briefly introduced some approaches and frameworks supporting this kind of information access.

There exist two types of connections between the different artifact types. First, the elements of one layer can be connected. In the product layer, is-part-of relations can be established between products and their components. Second, the linkage between the layers define the anchor points which enable the traversal between the layers. For example, documents are linked to the project in which they were created. The LFRP-search framework allows advancing from one layer to another through those inter-layer connections. For example, a requirements document—found in the document layer—is connected to a project from the project layer. Additionally, different projects consist of subprojects where different links might be propagated.

[4]An alternative name for an information object is an entity, which depicts a unique, definable object, which can be assigned information.

For each project the user can switch easily to the person layer and find people working in that project. In the LFRP-search framework a layer can be connected to each other layer.

The available layers and connections are company-specific and have to be modeled as artifact hierarchies before the search engine can be deployed. These hierarchies are integrated into an artifact type schema by defining the intra-layer and inter-layer relationships which will be introduced in Section 4.5 on page 121.

The enterprise search domain comprises a diverse variety of artifact types which need to be considered for searches, i. e. the available information varies strongly. Figure 4.2 on the next page shows an exemplary compilation of different artifact types with a non-complete list of attributes in Unified Modeling Language (UML) notation[5] in an *artifact type hierarchy*. In this hierarchy, the five different main artifact types *document*, *product*, *project*, *material* and *person* are distinguished. Each of these artifact types is assigned a specific set of attributes which can later be used as search criteria in search queries. For instance, a *document* can be characterized by attributes such as its document type (*documentType*), creation date (*creationDate*), version (*version*) and lifecycle state (*lifecycleState*) amongst others. In contrary, a person can be defined by its first name (*firstName*), last name (*lastName*) and its roles (*role*) in the company or project (designer, test engineer, purchaser, and others). Artifacts in the hierarchy inherit all attributes which are defined in parent-artifacts, i. e. all artifacts which are higher in the hierarchy and are described by an *is-a* relationship. Therefore, each artifact inherits from the root type *Artifact* which specifies that each artifact requires a unique identity (*artifactId*) and its path (*artifactPath*) where the artifact is managed (for instance, the document's file path in a plain file system or the logical path in a DM system or the PDM system. Furthermore, each artifact is assigned an *artifact type* which is shown in Figure 4.2 on the facing page as "virtual" UML attribute which specifies the specialization relationship of the artifacts on the top level of the hierarchy.

Furthermore, Figure 4.2 on the next page shows that artifacts of one specific artifact type can also have different sets of describing attributes based on a *discriminative* attribute as shown in the distinction of the project-based and product-based documents. Project-based documents, such as project plans, possess an attribute which relates the document to a certain project. Similarly, product-based documents—in the product development domain also called product models [Pahl et al., 2007]—are linked to the product. Another domain specific example is illustrated with the two example product groups *Screws* and *O-rings* [Eckstein et al., 2009]. For instance, seals with a circular shape such as the shown *O-rings* can be defined by their outer and inner diameter. On the other hand, *screws* are defined by their length, their threading type and so on. This example shows that each product group or more general each artifact type defines its own set of valid attributes in the hierarchy.

The LFRP-search framework requires the prior definition of the artifact type hierarchies for the domain where the framework is applied. This is necessary due to

[5] `http://www.uml.org/` (last accessed 08/01/2010)

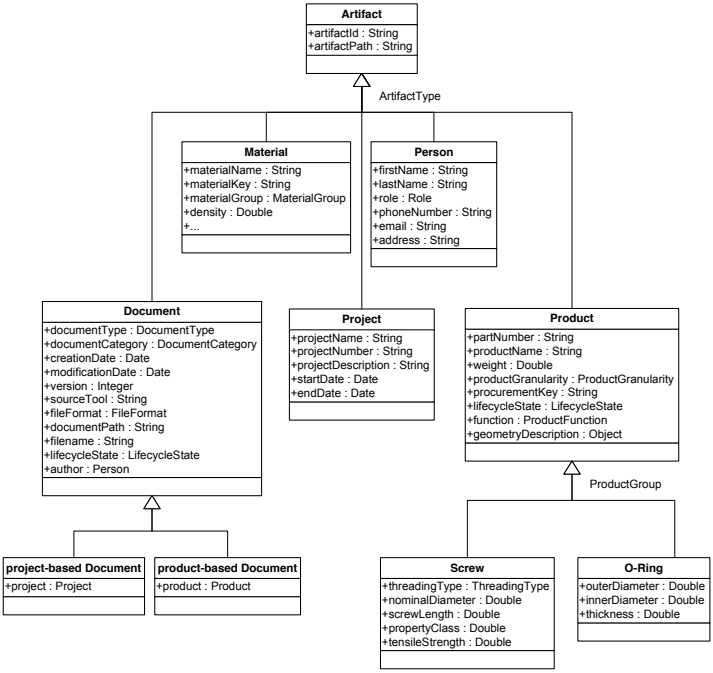


Figure 4.2: Example artifact type hierarchy from the product development domain (Adapted and enhanced from [Eckstein et al., 2009]).

the generic nature of the framework which makes it adaptable to different domains and environments.

As mentioned above, an artifact contains not only information about itself, but about other artifacts as well. This implies that artifacts can be interrelated in any way resulting in a network of various artifact relations. Such networks can be very complex as shown exemplarily in Figure 4.3 for the domain of product development. Projects are initiated in order to develop new products or to modify existing ones. A product is either a part of another product or an assembly consisting of multiple products. These products are usually described in documents which are created, revised, or needed in a certain process phase which itself is linked to a certain project. The projects are carried out by persons who have different roles in the development process (e. g. designer, project manager, etc.) and who create and modify the documents in various process phases.

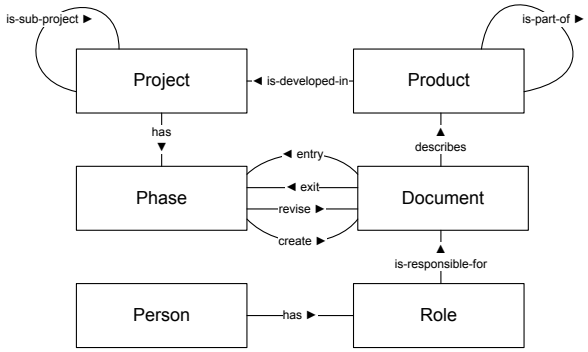


Figure 4.3: Relationships between different layers of artifact types (simplified schema).

The identified relationships hold valuable information about the artifacts and thus should be provided to users in order to help them satisfy their complex information needs. Tasks such as “Find all related documents for a product.” or “Find all projects in which material Z of supplier XY was used for the developed product.” are also manageable by the LFRP-search framework. This necessitates the extension of the artifact descriptions with relationships that are administrated in the index. The definition of possible relations between artifact types is also managed in the artifact type hierarchy which enables their setting during indexing time. During query time, the search component can utilize these relationships by utilizing the layer concept to improve both the search functionalities and the quality of the search results (cf. Section 4.3.3 on page 107).

4.2.2 Attributes and Feature Types

In order to make the manifold attributes of the different artifact types accessible to a search engine, it is necessary to make a classification of the different feature types beforehand. According to the understanding in the field of statistics, a feature describes a measurable property of a unit of observation[6] [Vogel, 2000]. In general, a feature of an artifact can be classified based on different types. The classification is usually based on the type of the feature and the number of possible characteristic values [Bosch, 1999], i. e. the possible values of the feature that a certain object can be assigned to.

According to Litz [2003] there are different approaches to categorize and separate features. The author distinguishes between a content-wise and a formal categorization, whereas here, the formal criteria are the main focus.

[6] In the case of this publication a feature equals a property of an artifact.

On the formal level, the distinction can be made based on

- qualitative (nominal or ordinal) and quantitative (cardinal) variables,
- as well as discrete and continuous variables [Litz, 2003].

Considering the LFRP-search framework, this distinction is necessary on the one hand with respect to available query options of the different attribute types which will be introduced in Section 4.3 on the next page. On the other hand, the visualization of the different facets in the parallel coordinates visualization depends on the type of the variable (cf. Section 5.3.2 on page 136).

The following sections describe the formal categorization of the possible "levels of measurement" (or scales of measure) which can be assigned to attributes: nominal, ordinal, interval, and ratio measurement [Stevens, 1946].

Nominal Scale

Attributes with a nominal scale[7] are a characteristic of qualitative variables. The different, disjoint attribute values which can be assigned to an attribute are equal, i. e. they are distinguishable from each other, but the distances of the differences are neither measurable nor quantifiable. Therefore, no order of precedence of the values is defined. Values of nominal attributes can be accumulated if an artifact supports multiple assignments of values [Vogel, 2000]. A specialization of the nominal attribute is a dichotomous attribute for which only two different attribute values are possible (e. g. the gender of a person which can only be *male* or *female*).

Examples for nominal attributes concerning artifacts of the type *document* can be the document format and the author, whereas the latter supports accumulation when a document was created by more than one person.

Ordinal Scale

Attributes with an ordinal scale are also representatives of qualitative attributes. In addition to the characteristics of nominal attributes, the distinguishable, disjoint attribute values can be put into a specific order of precedence. This can be done by either a definition of a *larger-smaller* or a *better-worse* relation, but this does not include a definition of the distances between the values [Vogel, 2000].

The attribute of the *creation phase* of a document is an example for an ordinal attribute, if a sequential order of the process phases in the process is assumed. The process phases can be put in a defined order, but no information about the distances of the attribute values can be calculated or given.

Cardinal scale

Attributes with a cardinal scale belong to the category of quantitative attributes and analogous to the ordinal scale have distinguishable, disjoint attribute values with a

[7] Additionally, in the literature the denotation *categorical* or *discrete* can be found.

defined order of precedence. Additionally, the distance between the attribute values is defined and measurable. Therewith, the comparability between the different attribute values is possible [Vogel, 2000].

Based on the used scale, the cardinal scale can be specified further as interval and ratio scale [von der Lippe, 1993].

Interval scale A scale where the zero value and the unit of measurement is assigned arbitrarily to an attribute is called interval scale. Thus, only the distance calculation between values is meaningful and not based on the zero point of the scale. A classic example is the temperature in degree Centigrade. The distance between the values of $10\,^{\circ}\mathrm{C}$ and $15\,^{\circ}\mathrm{C}$ is the same as between $30\,^{\circ}\mathrm{C}$ and $35\,^{\circ}\mathrm{C}$, i. e. $5\,^{\circ}\mathrm{C}$. The ratio between two attribute values cannot be computed (e. g. $20\,^{\circ}\mathrm{C}$ are not twice as warm than $10\,^{\circ}\mathrm{C}$).

Ratio scale Ratio scale attributes are characterized by a non-arbitrary zero value. Due to this characteristic, a calculation of ratios of attribute values is possible. The age of two persons can be compared as a ratio and the statement that "A 20-year old person is twice as old as a 10-year old person." is possible and true.

Another classification which is orthogonal to the type of scale is the distinction if the values are discrete or continuous [von der Lippe, 1993].

Discrete A variable is called discrete if it only can contain finitely many or denumerably infinite values in a closed interval [von der Lippe, 1993]. Examples for discrete variables are the number of sub-components of an assembly or the number of project members.

Continuous A variable is called continuous if the values from a closed interval can take "uncountably infinite values" [von der Lippe, 1993]. The age of a person can be seen as a continuous variable as it can be measured arbitrarily accurate. However, continuous variables are sometimes acquired discrete due to reasons of measuring accuracy [Pinnekamp and Siegmann, 2008].

After the introduction to the different categorizations of attributes, the query model of the LFRP-search framework can be elaborated.

4.3 LFRP–Query Model

This section examines the query model of the LFRP-search framework. In addition to the textual explanation of the query model a semi-formal description in an SQL-like notation is given. This description serves two purposes. On the one hand, the query model needs to be defined, so that the complete scope is accessible and understandable. On the other hand, this notation can be used for communication purposes to expert users who might need a deeper understanding of the inner workings of the

LFRP-search framework, e. g. when they are creating search templates for specific information needs in certain process stages (cf. Section 5.3.5 on page 150).

The specification comprises a definition of the supported facet types and their specific characteristics and query options.

Queries in the LFRP-search framework usually do not follow the classic Query-Response Paradigm [White and Roth, 2008] where searchers formulate the complete query before submitting it to the search engine. Instead the model supports an interactive *query refinement* which allows searchers to refine their queries incrementally after assessing interim search results and being guided by the search engine. By following this approach, users are provided with an instrument to better specify their information need by adding or removing search criteria than would be possible by a plain text-based search solution. After each change of the search query, the users are presented with an updated search result list, the visual display of these results in the parallel coordinates and a current list of available (and applicable) search criteria from which they can choose from. By that approach users never face an empty result set after adding additional restricting search criteria. This information is determined based on the artifact type hierarchies where the artifact types with their respective facets are modeled (cf. Section 4.2 on page 94 and Section 4.5 on page 121). The guidance of the search engine minimizes the occurrence of necessary backtracking steps which might happen due to mislead search tasks.

Formally, a query in the LFRP-query model consists of an arbitrary number of sub-queries. Each sub-query represents a facet selection on an axis in the parallel coordinates. Due to the support of arbitrary filter criteria and different types of similarity searches, each sub-query has to adhere to a certain query semantics.

An important goal during the conception of the LFRP-search framework was the omission of lengthy query statements that users would have to type in to invoke a search. Users should be able to compose and refine their query almost completely visually by adding and removing facets and conducting selections.

In general, an SQL query has the form:

$$\begin{aligned} &\textbf{SELECT } columns \textbf{ FROM } sources \\ &\textbf{WHERE } constraints \\ &\textbf{ORDER BY } ranking\ criteria \end{aligned} \tag{4.1}$$

Sections 4.3.2 on page 104 and 4.3.3 on page 107 especially cover the SELECT-, FROM- and the WHERE-clause of the statement. The order of the search results using the ORDER BY-clause is handled in Section 4.3.4 on page 110 where the user preference functions for facets are explained. In contrast to the simple sorting options based on facet values in SQL, here, the order of the search results is based on user preferences of different facet values. The explanations cover how the search results could be derived employing SQL statements. The aggregation of the current facet values and their facet value counts are sketched in Section 4.3.6 on page 118.

4.3.1 Facet Types

Basically, in the LFRP-search framework two different kinds of facets are distinguished: *attribute facets* and *similarity facets*. Attribute facets are describing aspects of an artifact. These aspects usually can be determined during indexing time. The facet values can either be extracted from the artifacts themselves or obtained from the respective management systems. For instance, the information about authors of documents can be extracted directly from the document when this metadata is maintained. Alternatively, this information can be extracted from a DM system and can be indexed if a system alike is used. The usage of attribute facets as filter criteria is usually a binary decision whether an artifact is contained in the result set or not.

Another characteristic of attribute facets is the potential number of facet values which can be assigned to a facet of a specific artifact. *Single-valued* facets allow only one facet value assignment per artifact, whereas *multi-valued* facets allow multiple facet values for one facet per artifact. An example for the former is the facet *document type*. Since this value is unique, only one value is permitted. An example for the latter case is the facet author of a document which might consist of multiple persons which are responsible for the document. This distinction has an effect on the available query options as will be detailed below.

Attribute facets can be either flat or hierarchical. Whereas the former only consists of a single value, the latter contains hierarchical information which specifies different kinds of granularity for this information. For instance, geographical information such as the regional provenance of a product or temporal information is predestined for viewing and filtering at different levels of granularity. Figure 4.4 on the facing page show an example of a hierarchical geographic facet hierarchy with the four levels *continent*, *country*, *state*, and *city*. Artifacts are assigned to a leaf node and thus define the hierarchical information. Another example are product groups which often are organized hierarchically (screw $\rightarrow$ head screw $\rightarrow$ cylinder head bolt). User selections in these hierarchies can be seen as *drill down* and *roll up* operations known from DWH systems (cf. Section 5.4 on page 152).

In addition to directly determinable facet values which can be extracted from an artifact, the values can be derived by pre-defined creation rules during the indexing step. For instance, a creation rule can define calculations on metric values or can combine different metadata fields (e. g. concatenating “firstname” + “lastname” to a *name* field). Facets which are derived by applying a creation rule are called *derived facets*.

Similarity facets are a special facet type where users are given the option to formulate a query based on an example artifact. Similarity facets demand an example object as input which then is transferred to a specific search module that determines artifacts which are similar based on specific similarity measures. Similarity facets are applied when conducting QbE queries. Taking the example of a QbE query for text documents, users upload a text document to the similarity facet placeholder in the user interface which is then used by the according text search module to determine similar documents. This QbE approach is elaborated further in Section 4.3.5 on page 116.

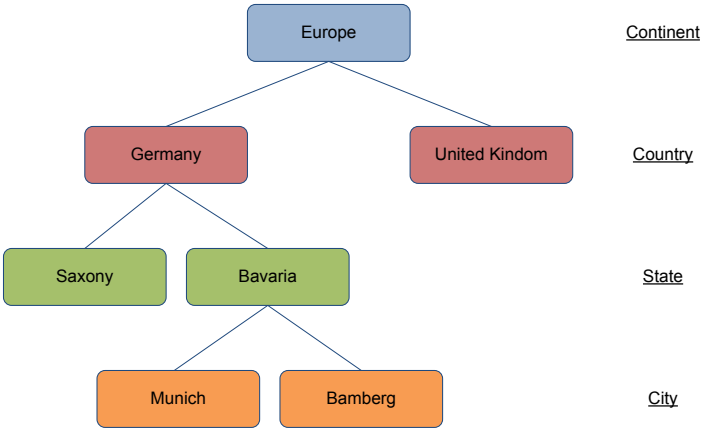


Figure 4.4: Example hierarchical facet with four different levels of granularity for geographical information.

The LFRP-query model supports users in giving preferences of facet values to influence the search result ranking by stating *user preference functions*. Facets which can be parameterized likewise are called *ranking facets* in the framework. A detailed examination can be found in Section 4.3.4 on page 110.

The relationships between the different supported artifact types introduced in Section 4.2 on page 94 have to be considered for the available facets of an artifact type as well.

Facets which are derived by traversing to other layers are referred to as *relation facets*. A relation facet of an artifact type *A* refers to a facet or to a derived facet of an artifact type *B* but gives this attribute a new semantics. Taking the example of the artifact type *document* which can be filtered by authors who created the document. The facet *name* of a person artifact gets the semantic of *creator* for the document artifact by the assignment of a relationship description (e. g. has created). Although, that facet and its value are derived from another layer (in this case the *person* layer) the facet belongs to the document layer and therefore is handled as a document facet. The computation of the facet values still needs to take place on the related artifact layer.

Scale-based Distinction

In addition to the distinction of attribute facets and similarity facets, each facet is assigned a specific facet type which is based on the level of measurement that represents the scale of the possible values of this facet according to the distinction in

Section 4.2.2 on page 98. In this publication the three facet types *nominal*, *ordinal* and *cardinal* are distinguished. This distinction is necessary with respect to the visualization of the respective axis as well as the expressiveness of the query which will be elaborated below.

A *nominal* scale is applicable for facet values for which no order can be defined, e. g. for the authors of a document. Facets with an *ordinal* scale can be ordered according to a natural order, but no distance calculation is supported between the facet values. An example for an ordinal facet is the process phase during which a document was created. The process model defines a sequence of consecutive process phases (simplified) which can serve as a way to order them. If the values can be ordered and a distance between the facet values is defined a *cardinal* scale type facet is given. An example for an attribute facet with a cardinal scale is the degree of maturity of a product, which can be assigned values from 0% to 100%. This value can be obtained during indexing time. Similarity facets which are derived based on a similarity search as outlined above are cardinal scale facets as well, since the returned RSVs are numerical values with a defined order of precedence.

4.3.2 Selections

As outlined above, one goal during the conception of the framework was the harmonization of the query formulation for users by treating the visualization and the query options for similarity searches and facet filtering similar. That way users face a common metaphor for their query statement, as every facet type operates in a similar fashion. In this section, queries are covered which only consist of sub-queries referring to one specific artifact layer. Relationships to other layers are addressed in Section 4.3.3 on page 107.

Users can combine multiple facets in their queries. Each facet for which a selection is made by users is represented as a sub-query Q_i addressing the respective facet which is combined with the other sub-queries by a conjunction (logical AND). This leads to the following SQL Statement (4.2):

$$\begin{aligned} &\textbf{SELECT } * \textbf{ FROM } layer \\ &\textbf{WHERE } Q_1 \textbf{ AND } Q_2 \textbf{ AND } \ldots \textbf{ AND } Q_n \end{aligned} \tag{4.2}$$

Here, n equals the number of facets which are currently chosen and for which user selections are made. Users are able to add facets to the query without conducting selections on them. These facets do not affect the query but support users in understanding the data and getting better insights into the search results.

The support for the disjunction of the different facet selection conditions is intentionally omitted as this combination operator would lead to larger result sets instead of reducing the result set which would add additional complexity for the users.

Each facet sub-query is specific for the facet type it is built upon. For *ordinal* or *nominal* facets both single-valued and multi-valued selections are possible. Whereas the first equals to the classical way found in most faceted search implementations,

the latter incorporates different ways of combining the values. If single-valued facets are assumed where only one value can be assigned to an artifact, the conjunction of selected values would cause empty results. In that case, the disjunction is the only way to combine several selected values for one facet. Considering a multi-valued facet—e. g. the *part function* where a (mechanical) part or assembly implements several functions—the conjunction is applicable and filters out documents describing parts which do not comply with the requirements, i. e. all the required functions. Another example is the facet *author* of a document. Documents can be created by several authors which allows two different ways of querying for documents by author selections. If users query for several authors which are combined by a conjunction, only documents that were jointly created by these authors are returned. Applying a disjunction for several authors weakens this criterion and returns documents where at least one of these persons is amongst the authors.

The distinction from above results in three different cases. In the following statements, a sub-query Q_i describes selections on the facet F_i with values v_j.

For single-valued facets, the corresponding SQL-like expression for Q_i utilizing the disjunction can be formulated as in Statement (4.3).

$$Q_i \equiv F_i \textbf{ IN } (v_1, v_2, \ldots, v_n) \tag{4.3}$$

The values of multi-valued facets are set-valued which usually would demand that the according database tables are normalized. Instead, set-valued entries in the database are assumed. Therefore, the query notation $v_j \in F_i$ is introduced which demands that the facet value v_j is included in the facet F_i. The corresponding SQL-like expression for Q_i can be formulated as in Statement (4.4).

$$Q_i \equiv (v_1 \in F_i) \textbf{ OR } (v_2 \in F_i) \textbf{ OR } \ldots \textbf{ OR } (v_n \in F_i) \tag{4.4}$$

For multi-valued facets additionally the conjunction is possible for Q_i according to Statement (4.5).

$$Q_i \equiv (v_1 \in F_i) \textbf{ AND } (v_2 \in F_i) \textbf{ AND } \ldots \textbf{ AND } (v_n \in F_i) \tag{4.5}$$

The two different applicable query semantics demand that users can switch between the two different approaches. The definition of this option can be found in the facet schema where the multi-value characteristic as well as the default semantic is set.

For cardinal scale facets single value selections are possible but rare. Multi-valued selections can be realized by allowing users to choose an interval of the values they are particularly interested in. The second axis in Figure 4.10 on page 113 shows this approach where users want to restrict the search results to documents whose degree of maturity lies between 60% and 100%.

More formal an interval query Q_i could be written as in Statement (4.6).

$$\begin{aligned} Q_i \equiv & F_i \textbf{ BETWEEN } v_1 \textbf{ AND } v_2 \textbf{ OR} \\ & F_i \textbf{ BETWEEN } v_3 \textbf{ AND } v_4 \textbf{ OR } \ldots \textbf{ OR} \\ & F_i \textbf{ BETWEEN } v_n \textbf{ AND } v_{n+1} \end{aligned} \tag{4.6}$$

The query model supports the statement of multiple inclusive intervals per facet to allow more complex query statements which can be done simply by adding a second interval to the plot. A use case might be the comparison of certain products from different price brackets.

When querying hierarchical facets, users can drill down into the hierarchy in successive queries. For the formulation of the query statement it is assumed that the hierarchical facet is stored in a compressed format containing the facet labels for a facet of an artifact in a concatenated string representation. The facet labels for each hierarchy level of a facet of an artifact are separated by a delimiter. A similar approach was chosen by Ben-Yitzhak et al. [2008].

Below, an example for a geographic assignment is shown where all four different levels are given. Here, a forward slash is used as delimiting character.

```
0: Europe/
1: Europe/Germany/
2: Europe/Germany/Bavaria/
3: Europe/Germany/Bavaria/Bamberg
```

The additional storage of the partial hierarchies has the advantage that an aggregation for the faceted search is possible on every level of the hierarchy.

A sub-query for a hierarchical facet would comprise a prefix query, i. e querying for a substring of the complete hierarchy of the facet. Formally, a sub-query would be written according to Statement (4.7).

$$Q_i \equiv F_i \textbf{ LIKE } \text{facetlabel\%} \tag{4.7}$$

The percent sign acts as a wildcard which represents no or arbitrarily many characters. The content of the facet label would be the concatenated part of the hierarchy to the desired level[8].

An example sub-query which selects all artifacts which are assigned to Bavaria would be formulated as Statement (4.8).

$$Q_i \equiv \text{geography } \textbf{LIKE } \texttt{'Europe/Germany/Bavaria/\%'} \tag{4.8}$$

A selection on a label retrieves all artifacts that have been assigned to this label. Therefore, a selection on a label within a hierarchy is equivalent to the selection of a disjunction of all the labels in the hierarchy level beneath, i. e. all Bavarian cities.

The combination of the above mentioned query selection options offers users a toolset to formulate complex queries which can be stated interactively with the help of the user interface described above. Users do not have to formulate the complete query in the beginning, but can add new criteria after they assessed the interim search results or remove criteria if the filtering was too restrictive. Referring to the SQL Statement (4.2) that corresponds to changing one of the sub-queries Q_i by adding or removing facet value selections.

[8] Of course, the displayed facet label in the user interface is only the description of the current level of granularity and does not have to be identical to the concatenated view which is only necessary internally for the query computation and storage.

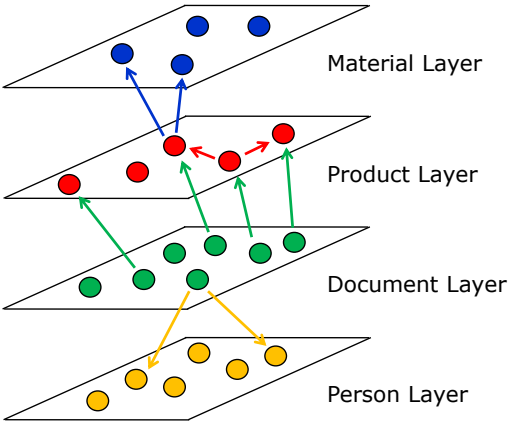


Figure 4.5: Inter-layer and intra-layer relationships of artifact layers shown exemplarily for the *material*, *product*, *document*, and *person* layer.

4.3.3 Multi–Layer Functionality

In addition to selections which concern only one artifact layer, the LFRP-query framework natively supports comprehensive inter-layer selections between artifact layers. The relations between the different artifact types are used for the switching of the search result layer. Figure 4.5 exemplarily shows artifacts of the four layers: *material*, *product*, *document*, and *person*. There are relations between products visible on the product layer (for instance, *is-part-of* relations between the different parts and the overall product) as well as relations between layers (product *is-made-of* material). Based on these relationships the following exemplary questions can be answered by the LFRP-framework.

- Who created the results?
- Which products are described in these documents?
- Which sub-components are used in a product?
- Which materials were used in a product?

In principle, search queries in the LFRP-query framework can start on every supported artifact layer. Users can take advantage of the linkage between the artifacts in two different use cases.

The first use case consists of the search for artifacts on one layer, where users formulate their queries according to Section 4.3.2. It can be helpful, to add filter

criteria by means of facets from directly connected layers to further decrease the search result set. When searching on the *document layer* it can be reasonable to further filter the result documents by the product facet *ProductName*. As shown in Figure 4.5 on the previous page *documents* (in the product development domain) can describe *products*, i. e. there exists a link between these two artifact types. This enables the LFRP-framework to aggregate facets of directly connected artifacts based on the current search result set from the current artifact layer and use this information to filter search results. The search engine automatically generates a nested query which takes the current artifacts from the search result—in the considered case documents—and retrieves all products which are linked to these documents. In the next step, the facet *ProductName* is determined for this set of products. So users can apply the linkage between artifacts to define filter criteria[9]. The search result which is displayed, still only contains documents which were filtered by facets stemming from a different layer.

For this section, a layer is defined to contain all indexed artifacts $a_{i,j}$ of one artifact type and is identified by $layer_i$ and is depicted schematically in Figure 4.6 on the facing page. Facets of layer i are referred to as $F_{i,j}$ where j is used to distinguish different facets of an artifact type. The mechanism of *dynamic facet provision* ensures that selections on the facets which are offered to the users return valid results (cf. Section 4.4 on page 118 for further details). For inter-layer queries this means that only facets from a connected layer can be offered, when each artifact from the current search result has the relationship to the other artifact type and is defined by valid values. If this is not the case, users have to further restrict their search. For instance, the relationship *describesProduct* which is available for certain types of documents can only be harnessed when the set of documents is restricted to product-describing documents, i. e. general documents like guidelines or norms are not included in the user selection.

In general, an inter-layer query can be notated according to the following Statement (4.9).

$$\begin{aligned}&\textbf{SELECT } F_{i,j}, F_{i,j+1}, F_{k,l}, \ldots \textbf{ FROM } layer_i \\ &\textbf{JOIN } layer_k \textbf{ ON } F_{i,m} = F_{k,n} \textbf{ WHERE} \\ &Q_{i,j} \textbf{ AND } Q_{i,j+1} \textbf{ AND } Q_{k,l} \textbf{ AND } \ldots \end{aligned} \tag{4.9}$$

The three exemplary facet sub-queries $Q_{i,j}$ correspond to the facets of different artifact layers given in the **SELECT** clause. The first index i denotes the number of the layer and the second index j is the number of the facet on layer i.

The use of the **JOIN** operator is valid here, as only facets from connected layers are offered to users which are valid for all currently selected artifacts, i. e. each artifact in the current artifact layer has a relationship to artifacts from another layer.

The constraint to only offer facets from other layers if each current artifact in the search result has a defined relation to another layer is very restrictive. Thus, it is

[9]Obviously, depending on the cardinality of the relationships this can lead to single-valued or multi-valued facets. However, multi-valued facets are common for direct facets as well and their treatment was discussed in Section 4.3.2.

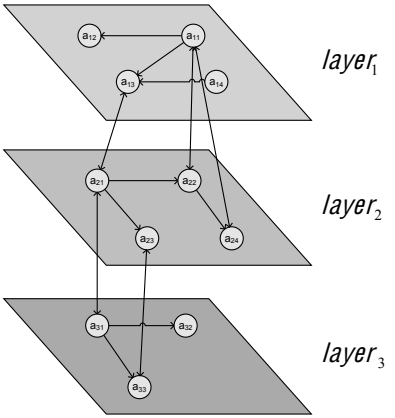


Figure 4.6: Schematic representation of the layer concept with three exemplary layers containing intra-layer and inter-layer relationships connecting two layers.

```
SELECT D.DocumentType, D.DegreeOfMaturity, D.Author, P.ProductName
FROM document D JOIN product P ON D.DescribesProduct = P.ProductId
WHERE D.DocumentType = "Requirements Specification" AND
D.DegreeOfMaturity > 0.8 AND D.Author = "Smith"AND
P.ProductName = "Engine"
```

Figure 4.7: Example query for facet selections from multiple artifact layers.

conceivable that the search engine determines relation facets which are available for the majority of the search results. Especially in situations where it is known, that the data quality is not perfect, i. e. that certain information is not covered in the search index, it is recommended to relax this constraint. Otherwise, many queries with layer-relationships cannot be conducted. In situations like this, users must be notified, that the resulting facets and artifacts from the related layers are based on a subset of the current search result.

The example query in Figure 4.7 comprises the document facets *document type*, *degree of maturity* and *author* and the *product name* facet from the product layer. The query shows selections on both artifact layers. Additionally, the primary/foreign key relationship *DescribesProduct* between the document and the product layer is harnessed for the result determination.

The second use case covers the transition between layers of artifacts. It consists of an initial query on one artifact layer whose search results serve as initial constraints for artifacts of a directly connected layer. Users select facets on which they want to filter by carrying out selections on their current search results—similar to the first use case. For certain information needs it can be useful to switch the artifact layer based on the current search results. A practical example of this approach is a search in the document layer where the users query the search engine with an example query consisting of a similar document. Based on the returned search result the users can do further filtering based on document facets. If they are satisfied by the result set, they might want to see all related products which are described in the documents from the result set. The LFRP-query framework now determines the connected products according to the linkage which is stored in the index. In short, the LFRP-framework takes the current result set on the original layer, navigates along the specified relationships and builds the new result set from the destination objects on the target layer. The users can apply additional selections on the result set and can filter further. Additionally, it is possible to return to the initial selections on the original layer to widen or narrow the selections which are applied to the destination layer.

The initial query on the first layer is shown in Statement 4.10 and the chosen sub-queries in Statement 4.11. The facet sub-queries $Q_{i,j}$ from layer i can be defined according to the options discussed in Section 4.3.2 on page 104.

$$\textbf{SELECT } F_{i,j}, F_{i,j+1}, F_{i,j+2} \textbf{ FROM } layer_i \\ \textbf{WHERE } Q \tag{4.10}$$

$$Q \equiv Q_{i,1} \textbf{ AND } Q_{i,2} \textbf{ AND } \ldots \textbf{ AND } Q_{i,n} \tag{4.11}$$

The actual layer switch happens in the second step where the relationship described by facet $F_{i,j+1}$ is used as a sub-selection as shown in Statement (4.12). The two layers i and k are connected by a relationship defined by the facet $F_{i,j+1}$ from layer i and by the facet $F_{k,l}$ from layer k.

$$\textbf{SELECT } * \textbf{ FROM } layer_k \textbf{ WHERE } F_{k,l} \textbf{ IN} \\ (\textbf{SELECT } F_{i,j+1} \textbf{ FROM } layer_i \textbf{ WHERE } Q) \tag{4.12}$$

The example shown in Figure 4.8 on the facing page harnesses the relationship between the document and the product layer called *DescribesProduct*. In terms of relational databases the connection between the two layers would be represented by a primary key/foreign key relationship. When further facet selections on $layer_k$ are added, these can be easily added in the outer WHERE-clause in Statement (4.12).

4.3.4 Integration of Ranking Functionality

As mentioned above, users should have the possibility to influence the ranking by giving preferences over certain search criteria they chose. Especially in database research several approaches were evaluated to add ranking possibilities to database

1^{st} step: Filter on document layer:
SELECT D.DocumentType, D.DegreeOfMaturity, D.Author
FROM document D **WHERE** Q
Q $\equiv$ D.DocumentType = "Requirements Specification" **AND**
D.DegreeOfMaturity $>$ 0.8 **AND** D.Author = "Smith"

2^{nd} step: Switch to product layer via the relationship DescribesProduct:
SELECT $*$ **FROM** product **WHERE** productId **IN**
(**SELECT** D.DescribesProduct **FROM** document D **WHERE** Q)

Figure 4.8: Example query statements for switching the artifact type.

queries. In the following, four different research approaches are introduced and followed by the specification on the integrated ranking functionality in the LFRP-search framework.

Database queries usually return a set of database tuples which can be sorted by an additional criterion. The query itself usually restricts the search result by Boolean expressions. Beck and Freitag [2008] propose an approach which adds ranking to a search result set by assigning weight annotations to Boolean expressions in the WHERE clause of an SQL statement. The weights are given by assigning single numbers from a range such as 1 ("less important") to 10 ("very important") to a selection criterion. This limitation is a huge advantage for users since they are accustomed to this kind of importance-based ranking. Moreover from a user interface viewpoint, controls for applying these weightings are easily included by displaying slider controls for instance.

For illustration purposes a complex weight annotated query F can be formulated according to Statement (4.13).

$$F := (A^1 \wedge (B^8 \vee C^2)^2)^4 \wedge (D^7 \wedge E^8)^3 \tag{4.13}$$

A through E are five different search conditions which are combined by conjunctions and disjunctions. The superscripts define the weights used for the ranking of the result set. The ranking process is based on the truth values of the Boolean conditions of the query. Thus, an expression tree for each result tuple is built. In short, each level of the resulting expression tree is transformed into an element of a rank vector for each tuple by aggregating the weights w_i of all operands which are true. The aggregation of the levels on the tree can be conducted by any monotonic function. The authors apply the weighted average as aggregation function in their publication. The final ranking of all tuples is determined by sorting all the rank vectors lexicographically.

Figure 4.9 on the next page shows a sample expression tree for a tuple which does not fulfill criteria A and B, but is true for criteria C, D, and E. The labels of each node contain a pair (w, b), where w is the weight of the subformula and b

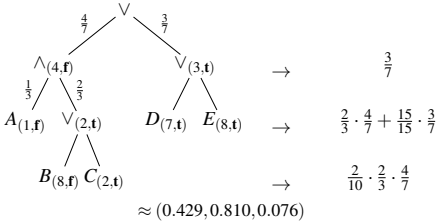


Figure 4.9: Sample expression tree for a tuple with the truth values (false, false, true, true, true) for the criteria A through E [Beck and Freitag, 2008].

is the truth value of this node, determined by the conjunction or disjunction of the child nodes. The rank vector of the sample tuple from above is $(0.429, 0.810, 0.076)$. Additionally, the authors provide an adapted relational algebra and introduce a new operator *classify* and provide insights into the implementation into an existing SQL query engine and the necessary query optimizations.

A similar approach which tries to introduce an extended relational algebra and necessary query optimization is presented in [Li et al., 2005] with the RankSQL approach. In particular, the authors aim at including the ranking functionality of top-k queries as a database operator and not a solution which sits outside of the query engine. The authors show how to integrate a ranking relationship as a property of the data similar to the membership property used for boolean filtering.

Kießling and Köstler [2002] propose a different approach for ranking relational database queries with *Preference SQL* which extends SQL by a preference model based on strict partial orders (in contrast to [Beck and Freitag, 2008] where numerical weights are used to indicate user preferences). In addition to standard SQL, they include additional language constructs which act as an orthogonal extension of standard SQL. Preferences are added using the **PREFERING**-clause which can be parameterized by certain base preference types. The **AROUND** type allows to favor numerical values close to a numerical target value. The **LOWEST** (**HIGHEST**) preference type allows asking for the lowest (highest) value, if possible. Otherwise, the closest value to the minimum or maximum is used. This preference type can be paramterized by a single value or by an arithmetic expression over several attributes. Finally, positive (**POS**) or negative (**NEG**) preferences can be given.

Additionally, different operators are available to combine several single preferences in a complex preference statement. It has to be distinguished between equal importance of preferences or an order of preferences. The former is described by the Pareto accumulation which follows the concept of Pareto efficiency. The latter uses the **CASCADE** operator to define an order of preferences which are conducted consecutively until a unique order is achieved. Both combination operators can be used together. In addition to the definition of these operators, the authors give an insight into the Preference SQL query optimization and provide benchmark results.

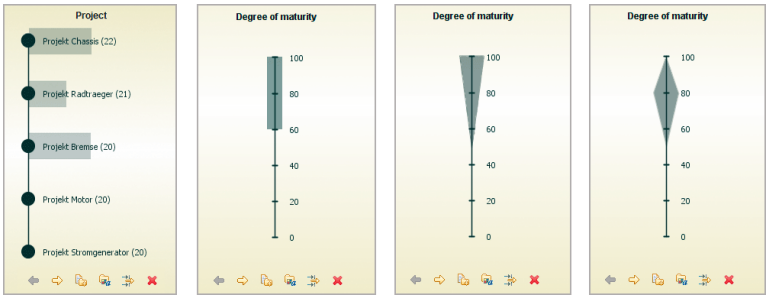


Figure 4.10: Different selections based on user preference functions for nominal and cardinal facets defining interval selections and ranking conditions.

In [Lee, 1994], an overview of Extended Boolean retrieval is given which is an extension of Boolean retrieval that allows weighting of boolean clauses for ranked retrieval. The author introduces an *n-ary soft boolean operator* based on the P-Norm model. The semantics of the Extended Boolean retrieval differ in contrast to the approaches described above. The contents of the result sets from the approaches above were not influenced by the weighting of the boolean criteria, thus they were only used for sorting the boolean result. In contrast, the weighting approach of Extended Boolean retrieval uses a fuzzy set model and therefore can exclude certain search results due to their weights. Thus, a weight for a boolean clause being zero leads to the omission of this clause which is a different view on truth values as in classic boolean retrieval.

The LFRP-search framework supports ranking functionality based on facet selections. As described above, users should easily be able to apply weights to their queries. The framework supports users in augmenting their queries by the notion of *user preference functions* which define a ranking of artifacts by facet values. Since the framework does not automatically determine weights, the ranking mechanism needs to be comprehensible for the users. The definition of weighting criteria is completely optional for the users, but if desired, they have the possibility to “draw” visual functions onto each axis in the parallel coordinates plot to prioritize artifacts differently according to their facet values for a specific facet (cf. Section 5.3.2 on page 135).

In general, this concept is applicable for the three introduced facet types (nominal, ordinal, cardinal), but the available functions are different. For nominal and ordinal facets only a discrete function as shown in the first axis in Figure 4.10 is reasonable. Preferences on cardinal facets can be set by a continuous function as seen in the three rightmost axes in Figure 4.10. Both types of functions allow users to specify facet values which are more important for their current information need.

According to the initial query blueprint in Statement (4.1) the artifact ranking is determined based on the **ORDER BY** clause. The order is conducted according

to the score of an artifact a_j which is determined by computing a weighted average according to Statement (4.14).

$$score(a_j) = \sum_{i=1}^{n} \alpha_i \cdot f_i(x_{i,j}) \text{ with } \sum_{i=1}^{n} \alpha_i = 1 \tag{4.14}$$

Here, α_i is the weight for the facet i, which can be set for each facet by the users to influence the weighting of the functions (the default is $1/n$). By that means users can define higher or lower emphasis for a single ranking criterion. $f_i(x_{i,j}) \in [0, 1]$ describes the user preference for the value $x_{i,j}$ of facet i for artifact a_j as graphically defined by the users.

The continuous function f_i can be defined completely according to preferences of the users in a visual way. But for easier comprehension the system introduces some simple template functions which are often needed like the triangular and quadrilateral shapes found in the third and fourth axis in Figure 4.10 on the previous page. These functions again are completely customizable and shown formally in Figure 4.11 on the facing page. The upper limit of a function is called l_u, whereas the lower bound is called l_l.

An example for a more complex user preference function concerns the document degree of maturity. Users might want to rank more mature documents higher and do not want to include documents where the corresponding value lies below 50%. If they already know the released documents (i. e. documents that are 100% mature) and want to rank documents higher which will be finished soon, the function in the fourth axis in Figure 4.10 on the previous page can be helpful. This ranking functionality enables users to simply apply ranking criteria to their searches to quickly access the information they seek.

Rank Aggregation over several layers

In Section 4.3.3 on page 107 the multi-layer functionality of the LFRP-search framework was introduced. If a multi-layer search is conducted by users which contains a layer switch[10], two different cases need to be distinguished. In the first case, users only use the initial query to determine a set of artifacts from the initial layer. Since there is no ranking in the initial search result no aggregation needs to be undertaken for the results of the directly connected artifact layer to which user switched in the second step.

In the second case where users utilized the ranking functionality of the LFRP-search framework by employing user preference functions, this preference information should be propagated to the target layer and incorporated in the calculation of the target ranking. Depending on the relationship between the two concerned layers, multiple target artifacts can be referenced when the relationship between the two layers is a one-to-many relationship. Additionally, an artifact from the target layer can be referenced by multiple artifacts from the initial layer. For instance, a

[10]The second use case described in Section 4.3.3 on page 107.

Constant function (catch all) for simple filtering without influencing the ranking by the considered facet:

$$f_i(x) = \begin{cases} 1 & \text{for } x \in [l_l, l_u] \\ 0 & \text{otherwise} \end{cases}$$

Example:

$$f_i(x) = \begin{cases} 1 & \text{for } x \in [60, 100] \\ 0 & \text{otherwise} \end{cases}$$

Simple linear function with filtering:

$$f_i(x) = \begin{cases} 1 - \frac{l_u - x}{l_u - l_l} & \text{for } x \in [l_l, l_u] \\ 0 & \text{otherwise} \end{cases}$$

Example:

$$f_i(x) = \begin{cases} 1 - \frac{100 - x}{100 - 50} & \text{for } x \in [50, 100] \\ 0 & \text{otherwise} \end{cases}$$

Complex linear function with filtering:

$$f_i(x) = \begin{cases} \frac{l_u - x}{l_u - l_l} & \text{for } x \in]l_t, l_u] \\ 1 - \frac{l_t - x}{l_t - l_l} & \text{for } x \in [l_l, l_t] \\ 0 & \text{otherwise} \end{cases}$$

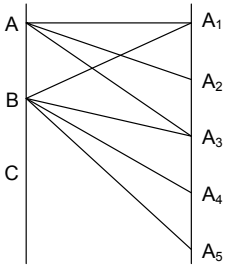
Example:

$$f_i(x) = \begin{cases} \frac{100 - x}{100 - 80} & \text{for } x \in]80, 100] \\ 1 - \frac{80 - x}{100 - 50} & \text{for } x \in [50, 80] \\ 0 & \text{otherwise} \end{cases}$$

Figure 4.11: Formal example functions for the definition of ranking criteria.

document can describe multiple *parts* and thus, the RSV from this document needs to be propagated to all referenced parts. In consequence, multiple documents can reference the same part. In this case, a certain semantic needs to be chosen to combine the multiple RSVs.

Figure 4.12 on the following page shows an example where artifacts A and B from the origin layer are related to multiple different artifacts A_1 through A_5 from the target layer. In this example, the *average semantic* is chosen where the RSVs from relating artifacts of the initial layer are evenly averaged for an initial RSV for



Initial Ranking

Rank	Artifact	RSV
1	A	0.8
2	B	0.6
3	C	0.2

Initial Target Layer Ranking with Average Semantic

Rank	Artifact	RSV
1	A_2	0.8
2	A_1	0.7 (0.8*0.5 + 0.6*0.5)
	A_3	0.7 (0.8*0.5 + 0.6*0.5)
3	A_4	0.6
	A_5	0.6

Figure 4.12: Rank aggregation over multiple layers.

the artifacts of the target layer. In further refinements of the query these scores can be altered by additional ranking criteria. The calculation is then performed as described above. Alternatively, a maximum or a weighted average semantic can be chosen, but for easier comprehension the LFRP-search framework uses the average semantic.

This example also shows artifact C which does not have any related artifacts on the target layer and thus is not included in the result calculations for that layer[11].

4.3.5 Combination of Faceted Search and Similarity Search

The introduced ranking functionality allows a seamless integration of QbE and keyword-based searches into the LFRP-search framework. Similarity queries return a ranking of artifacts based on an example query object. Depending on the similarity computation the ranking is either based on similarity values or distances [Zezula et al., 2006]. The framework allows multiple similarity sub-queries Q_i in a query as illustrated in Statement (4.2).

In general, the query model of the LFRP-search framework maps its query options to subordinate query paradigms (such as filter conditions, QbE, or text retrieval models like the vector space model[Salton et al., 1975]), takes the returned results (which correspond to a partial ranking of the complete query), and makes them available for further filtering and ranking by translating them to the concept of the LFRP-query model. The process of query formulation is harmonized so that users simply need to understand one metaphor on how to create queries and can transfer this to different query paradigms.

[11] In this example the constraint that layer-switching is only available if all source artifacts have a defined relation to the target layer is relaxed.

The LFRP prototype is designed for product development processes in the engineering domain which in addition to text retrieval also demands the retrieval of CAD models (both 2D and 3D) based on geometrical and topological information. For similarity queries on CAD documents see for instance [Bustos et al., 2005]. The framework itself is extensible to various kinds of similarity searches. Examples for other domains could be the search for a piece of music based on a sample recording. Müller [2007] gives a thorough overview of the field of music IR. Another exemplary domain is the field of image retrieval in medical applications. One goal of this domain is to automatically classify x-ray images for similar medical evidence or to find other x-rays with similar diagnostic findings (refer to [Müller et al., 2004] for an overview of this domain).

Similarity queries return a ranking of artifacts where each of the artifacts is assigned a score representing the degree of similarity to the query object. This score acts as the facet value of the respective similarity facet. Its determination is executed in sub-modules of the LFRP-search framework. The generic search framework relays the users' facet selections and the example objects to these sub-modules and maps the returned results to the parallel coordinates visualization. The example query object needs to be converted into a certain representation appropriate for the specific search sub-module (e. g. into a term vector for text searches). The module then returns a partial ranking and determines the facet values for this similarity facet.

By default each similarity facet is assigned a linear function which normalizes the scores of each similarity measure to the interval $[0, 1]$, where a similarity score closer to 1 means a higher similarity. This step is necessary to align scores of multiple similarity facets. Additionally, this normalization ensures the comparability of distances. This function preserves the preference relation of the initial scores.

Users still have full control over the ranking functionality and can adjust the default user preference function to their needs. Typical actions are the pruning of the search results by changing the upper (l_u) or lower boundary (l_l). This can be necessary to remove the least similar artifacts from the result ranking.

In certain situations it might be beneficial to remove the standard linear function and assume a constant function to remove the preference on the similarity and only use this facet for filtering. In a next step, users can parameterize other ranking facets to apply special ranking semantics. Although possible, the case that users want to invert the similarity ranking by flipping the user preference functions seems rare.

Since users can filter these results analogous to other types of facets, the usage of these similarity searches is transparent to filter queries.

The description of the various similarity search approaches for the different artifact types is out of scope of this publication.

In the case that users want to include multiple similarity facets in a search query, the merging of the rankings is simply achieved by applying the weighted average from Statement (4.14). Users can decide how the different features should be weighted relatively to each other.

4.3.6 Determination of Query Previews

The faceted search paradigm supports users in providing query previews which show them how many search results are available for each displayed facet value (as introduced in Section 2.3 on page 18). In general, a facet value count is the number of artifacts that the search engine would return if users select a certain facet value in addition to already conducted selections.

The following statement shows how the facet value count for a facet F_i with the value v_j is determined in an SQL-like notation. The facet sub-queries Q_2 to Q_n are the selections made prior to the current facet value count calculation which have to be considered.

$$\begin{aligned} &\textbf{SELECT COUNT } (*) \textbf{ FROM } layer_i \\ &\textbf{WHERE } F_i = v_j \\ &\textbf{ AND } Q_2 \textbf{ AND } \ldots \textbf{ AND } Q_n \end{aligned} \tag{4.15}$$

Without any selection in the parallel coordinates plot, the sum of all facet value counts of a single-valued facet is the number of all artifacts in the result set. For multi-valued facets the sum is greater or equal than the number of all artifacts of $layer_i$, since multiple facet values can be assigned to one artifact.

4.4 Dynamic Facet Provision

As argued above, users need guidance through the query formulation process. One aspect of this guidance consists of the provision of the available next query options users can undertake to specify their queries more accurately. This is necessary for two different reasons. First, users should not be provided with facets and facet values which would lead to an empty result set, if they would conduct selections on them. Thus, the displayed facet values need to be valid for the current artifacts in the result set. Second, the visual presentation of the facets demands the determination of "important" facets and their facet values due to screen space limitations. Although possible, the display of all facets and their values can lead to a feeling of being overwhelmed by the variety of query options for the users. This is especially important in large domains where many different artifacts with different available query options exist. The research with the faceted search engine *Magnet* showed that the display of all available facets and their values can overwhelm users and can lead to degraded user performance [Sinha and Karger, 2005]. Thus, it is not purposeful to present users with all facets which are available in the search system. The search engine needs to determine the more useful facets with respect to the users' information needs which are then displayed to them.

Koren et al. [2008] examine ways to generate "intelligent" faceted search interfaces by using explicit user ratings to automatically select facets and facet values. The authors introduce a probabilistic framework to build faceted document models and user relevance models maximizing the utility of the selected facets to each individual searcher. They propose several algorithms to determine facet-values which

are presented to users during the different query refinement steps. The *most frequent* algorithm simply aggregates all facet values of the current search result and provides users with the facet values that occur most. The *most probable* approach ranks facet values based on the probability that the artifacts assigned with those facet values are relevant to the user. The determination of these probabilities can be done by collaborative efforts (relevance judgments by the community of users) as well as by personalization for each user. *Mutual information* is a common method used in feature selection. Hereby, the pointwise mutual information between the presence of a facet-value pair appearing in an artifact and the artifact's relevance is calculated. The most informative values are shown to the user for query refinement.

Additionally, the publication proposes different methods to build a starting search page for users beginning their search task where initial sub-queries are already chosen. The baseline is the *null start state* method where no preselection on facets is conducted. The *collaborative start state* method automatically issues a query based on the most likely facet values of the artifacts users are searching for. The authors employ a hierarchical Bayesian model to infer initial facets for the users based on other users' choices. The *personalized start state* method resorts to a constant user relevance model which assumes that documents which were relevant for previous information needs are relevant to users now. The publication uses both the collaborative as well as the user-specific approach since it takes some time to learn a detailed user model and the collaborative data is available earlier.

Dash et al. [2008] cover the facet selection problem from a different viewpoint by trying to provide users with "surprising" aspects of the data collection. The main goal is the discovery-driven analysis similar to work in OLAP exploration. They employ an "interestingness" measure which describes how surprising aggregated facet values are based on a given expectation. In contrast to faceted search implementations found in e-commerce scenarios, the introduced faceted search system focuses on discovery-driven analysis by displaying facets that deviate most from users' expectations. The authors introduce several methods to set the expectation of a user and found that the *navigational* method is particularly suitable in the context of faceted search. This method sets the users' expectations based on how they navigate the search results by relating the facet counts of the current sub-query to the distribution of the facet counts of the complete data collection.

The determination of the "interestingness" of a facet value with respect to the query is done by calculating the probability that in a random sample of the size of the current search result set there will be at least q documents with that facet value. q is the number of artifacts in the search result assigned with the current facet value. In statistical hypothesis testing, this probability is called *P-norms* and denotes the probability of getting a result which at least is as extreme as a given data point assuming that the null hypothesis is true. In the next step, an aggregation of facet values of the different facets takes place to determine which facets are "surprising" enough to be presented to users. The aggregation happens based on the k most interesting values of a facet and is aggregated by different weighting schemes such as a *maximum* weight, *average* weight and *hybrid* weight semantic.

Dakka et al. [2005] compare three different techniques on how to determine the

"best" facets which are shown the users. Their baseline is defined by a *frequency-based* ranking. Facet values assigned to more search result artifacts are preferably displayed. The *set-cover* ranking aims to maximize the number of distinct objects that are accessible from the top ranked categories. This task is an instance of NP-complete set-cover problem [Cormen, 2007]. By utilizing a greedy algorithm the authors approximate this set-cover problem in time linear to the number of categories. The third introduced ranking approach called *merit-based* takes structural properties of sub-hierarchies into account. In particular, this approach ranks categories higher that allow users to access their contents with the smallest average cost.

The LFRP-search framework realizes the guidance for its users with its *dynamic facet provision* feature. This feature takes care of providing users only with valid facets that are currently available for the search results of the current query. Furthermore, users are provided with facets which are useful with respect to the current state of the query. After each refinement of the query, the framework determines the facets which should be displayed to the users dynamically. For this requirement two distinctions need to be made. On the one hand, the framework takes care of providing users with available facets for the current artifact type. On the other hand, facets from directly connected artifacts are determined and provided to the users as well.

As mentioned above in Section 4.3.3 on page 107, the LFRP-search framework supports different connected artifact types which are beneficial for different information needs in enterprise search scenarios. These artifact types are logically organized on different linked layers. In Section 4.2.1 on page 94 the artifact type hierarchies were introduced. These play a major role for the *dynamic facet provision* since they define the inheritance hierarchy of the artifacts and their specializations along with the facets which are available for each level of specialization for an artifact. This hierarchy is represented by a schema defining dependencies between artifacts and facets as described in Section 4.5 on the facing page. A dependency between artifacts of a hierarchy is modeled by providing the parent artifact type along with a facet-value pair and the respective target artifact type. Naturally, a dependency can consist of different mappings based on the same facet, but different facet values.

The list of available facets from which users can choose during a search task depends on the previous facet selections. The search query is specialized by each new facet selection which offers users more specific facets for their information need. For instance, if they specify their search to products and restrict the commodity group to *o-ring seals*, the facets *inner radius* and *outer radius* can be enabled. The update of the currently enabled facets is done instantly after a facet selection of the users.

To achieve this specialization the search framework determines the hierarchy path of each search result during query evaluation and tries to find the lowest common ancestor node in the hierarchy to determine the set of facets which are provided to users as next possible steps. When this node is found in the hierarchy, the schema provides the relations of this type for the current subtree as well as references to other subtrees of the artifact hierarchy which can be used for layer switching. The

specialization of artifact types is completely conducted with the help of the discriminative facets.

A problem already introduced above concerns the situation where the artifact descriptions in the search engine are not complete due to data quality problems. The strict view would prevent certain choices of specializations if not all artifacts in the current search result contain values for the discriminative facet under consideration. If relations of a subset of the current search result are used for the further specialization, users need to be notified, that the current search result was reduced to the artifacts where the chosen relation was available.

A search task does not always have to start at the top of the hierarchy where users would have to decide with which type of artifact they are starting their search task. The LFRP-search framework can also start with a subtree of the hierarchy and thus hide other partial trees from the hierarchy for simplification or more specific search queries.

The relatively strict schema-approach for determining the available facets for further refinement was chosen to support more predictable results for the facet provision during searching. The downside is of course the prior definition of these dependencies in a schema as well as the necessary data curation. Furthermore, this approach was chosen since the observed domains are very complex and many dependencies exist between the different artifact types. Thus, the manual approach promises better results than the automatisms which try to infer relations between various artifacts. The approaches described in the beginning of this section have the advantage of being mostly automatable but may return results which are inexplicable to users.

The realization of this feature is described in Section 5.3 on page 134 from a visual viewpoint.

4.5 Schema of the LFRP–Search Framework

The configuration of the LFRP-search framework is done by a schema, which describes aspects for the indexing as well as for the query modules. The schema is an XML document specified by an XML Schema definition which is shown in Appendix B.1 on page 177.

The schema contains two main sections. The first section defines all facets which exist in the portrayed domain or in the organizational context where the framework is deployed.

The facet descriptions are general, re-usable definitions which can later be used in several artifact definitions in the schema. The main purpose is to define a unique identifier of the facet (attribute `id`), its type (attribute `type`) and its cardinality (attribute `multiIndexable`)[12]. The unique identifier is later used to reference this facet in the artifact descriptions. Since the framework was influenced from search

[12]This definition clarifies whether one facet value or multiple values can be assigned to an artifact. An example for the latter is the facet *author* of a document. Multiple authors can create a document, but for instance the *creation date* is unique and therefore not multi-valued.

situations in an enterprise setting, in addition to the scale-based distinctions for facet types in Section 4.3.1 on page 103, the cardinal scale facets are detailed further. They are distinguished between the following three additional types:

- **FUNCTION_INTERVAL** This type is used for attribute facets which are of cardinal type such as inner or outer diameter of a product.
- **TEXT_QUERY** This type is applied for textual queries where users can enter keywords for a text-based search.
- **QUERY_BY_EXAMPLE** Since users should be able to conduct (similarity) queries based on example objects, facets which support this kind of query are parameterized by this type. The LFRP-search framework supports different types of similarity searches. The exact type is defined in the schema with the attribute `similarityType`. The prototype itself supports the following similarity searches:
  - **TEXT:** In contrast to the aforementioned textual keyword queries, this QbE query targets similarity searches of complete documents. This distinction is made, since complete documents may contain additional structured information which can be used in the query process.
  - **GEOMETRY_2D and GEOMETRY_3D:** This type of QbE query determines similarity based on the found example geometry for two-dimensional objects (e. g. technical drawings) and for three-dimensional objects such as 3D-CAD drawings.
  - **TOPOLOGY_2D and TOPOLOGY_3D:** This type of similarity search focuses on the topology of the described objects (refer to [Fonseca and Jorge, 2003] for further information).

Further details on the methods used to conduct similarity searches in the product development domain is given for 3D objects for instance in [Bustos et al., 2005] and for 2D objects in [Weber and Henrich, 2007].

For cardinal scale facets it can be defined whether the ranking functionality from Section 4.3.4 on page 110 is enabled or disabled for users with the attribute `weightingEnabled`.

Additionally, a facet can also be of type **DATE** which allows time-aware filtering.

The aspects of facets described above are necessary for the indexing and the query handling in the LFRP-search framework. Additionally, the schema contains information about the visual representation of each facet. As shown in Listing 4.1 on the next page each `facet` element in the schema contains a child element `visual` which defines clear text names and an icon path for the visual representation in the user interface. Additionally, it is defined whether a facet allows multi-valued selection and if so, which combination operator should be used (disjunction or conjunction).

As mentioned in Section 4.4 on page 118 it is not always reasonable to display all available facet values to the user due to screen space limitations and the

added feeling of being overwhelmed. Each facet can be parameterized by multiple `comparators` which are used in the result computation to determine how many facet values are included for a facet and how they are sorted. The schema allows the definition of a default comparator for each facet. Each comparator is defined by its JAVA classname which is instantiated during the result computation and assigned to each facet. If multiple comparators are instantiated, the user interface can allow users to switch the sorting of the facets based on these comparators. Exemplary comparators are:

- **Numerical Comparator** This comparator is mainly used for cardinal scale facets and sorts values numerically.
- **Facet Count Comparator** This comparator sorts facet values bases on its cardinality of the set of artifacts assigned to this value.
- **Alphanumerical Comparator** This comparator is used for textual facets and sorts the facet values alphanumerically.

The framework is not limited to these comparators, but allows easy extensibility by offering a comparator interface for the development of custom comparators.

As outlined in Section 4.1 on page 92 each facet is visualized as an axis in a parallel coordinates plot in the user interface. Although, the framework harmonizes the query statement along different types of search criteria, some distinctions need to be made. The child element `visualizationtype` defines the display of each facet in the visualization due to the order of the displayed facet values, the integration of distances between facet values and the application of user preference function. Thus, the schema distinguishes the following visualization types:

- **NOMINAL**
- **ORDINAL**
- **FUNCTION_INTERVAL** This type is used for cardinal scale facets and allows the use of user preference functions.
- **DATE** This facet type allows users to conduct interval queries on temporal data and support hierarchical queries where the facet values can be aggregated along different temporal granularities (year, quarter, month, day, etc.).

Listing 4.1 shows an example facet which handles 3D-geometry similarity searches.

```
1 <facet id="3dGeometry" type="QUERY_BY_EXAMPLE" similarityType="
    GEOMETRY_3D" documentType="CAD_MODEL" multiIndexable="false"
    weightingEnabled="true">
   <visual name="3D GEO-Similarity" iconPath="images/facets/
      qbe_cad_geometry.png" multiValued="true" multiValueOp="OR">
3    <comparators>
       <comparator default="true"
5        classname="org.forflow.search.facet.comparator.
            NumericalComparator" />
     </comparators>
7    <visualizationtypes>
```

```
      <visualizationtype facetType="FUNCTION_INTERVAL"
9           default="true" />
    </visualizationtypes>
11  </visual>
</facet>
```

Listing 4.1: Facet description for a 3D-geometry similarity facet.

In contrast Listing 4.2 describes an attribute facet for the type of a document.

```
<facet id="documenttype" type="NOMINAL" multiIndexable="false">
2  <visual name="Document Type" multiValued="true" multiValueOp="OR">
     <comparators>
4      <comparator default="true" classname="org.forflow.sortable.
         FacetCountComparator" />
       <comparator classname="org.forflow.sortable.AlphaNumComparator"
         />
6    </comparators>
     <visualizationtypes>
8      <visualizationtype facetType="NOMINAL" default="true" />
     </visualizationtypes>
10 </visual>
</facet>
```

Listing 4.2: Facet description for a document type attribute facet.

The second section of the LFRP-schema covers the artifact type hierarchies where the different artifacts are defined along with their facets, inheritance hierarchy and dependencies to specialized artifacts of the same root artifact or to other artifacts. These definitions are found in the `entities` element. Each entity is described by a set of `facet` elements which reference the facet descriptions from above. Furthermore, it is defined whether that facet is mandatory or optional by the `required` attribute. This information is used during indexing to ensure data quality. As shown in Listing 4.3 each artifact has at least three required facets *artifactId*, *artifactPath* and *artifactType*.

```
1 <entity id="artifact" name="Artifact" root="true">
    <facet required="true" id="artifactId" />
3   <facet required="true" id="artifactPath" />
    <facet required="true" id="artifactType" />
5   <facet required="false" id="embeddedSystemPath" />
    <facet required="false" id="artifactHierarchyPath" />
7   <facet required="false" id="guiDescription" />
    <dependency>
9     <from id="artifactType" />
      <mapping>
11      <value>PRODUCT</value>
        <to>product</to>
13    </mapping>
      <mapping>
15      <value>DOCUMENT</value>
          <to>document</to>
17    </mapping>
      <mapping>
```

```
19        <value>PERSON</value>
          <to>person</to>
21      </mapping>
        <mapping>
23        <value>MATERIAL</value>
          <to>material</to>
25      </mapping>
        <mapping>
27      <value>PROJECT</value>
          <to>project</to>
29      </mapping>
    </dependency>
31  <parent-entity id="artifact" />
</entity>
```

Listing 4.3: Entity description for the root artifact.

The *dependency* element defines a dependency of the currently described artifact to a child artifact by providing the source facet, its value and the target artifact. Listing 4.4 shows the *person* entity (which itself is inheriting from the *artifact* entity) and its child entity *employee*. The dependency between these two artifacts is modeled by the facet *role*. Persons for whom the role matches "Employee" are specialized *Employee* artifacts.

```
<entity id="person" name="Person">
2   <facet required="true" id="role" />
    <facet required="false" id="surname" />
4   <dependency>
      <from id="role" />
6     <mapping>
        <value>Employee</value>
8       <to>employee</to>
      </mapping>
10  </dependency>
    <parent-entity id="artifact" />
12</entity>

14<entity id="employee" name="Employee">
    <facet required="false" id="department" />
16  <!-- relations to other layers -->
    <relationfacet required="false" id="works_in_project" toEntityId="
        project" toEntityFacetId="project_name"/>
18  <parent-entity id="person" />
</entity>
```

Listing 4.4: Entity description for the person artifact and its child artifact employee.

A special type of facets are relation facets which describe relationships between artifacts of different sub-hierarchies and are derived by traversing to other layers. A relation facet of an artifact type A refers to a facet or to a derived facet of an artifact type B but assigns this attribute a new semantic. In general, they are defined by a source layer (e. g. documents), a set of combinable target attributes (i. e. facet names) such as the surname or a combination of first and last name ("first name" +

”last name”) and an identifying relationship which defines the name of the relation facet.

```
1 <relationfacet required="false" id="author" toEntityId="person"
    toEntityFacetId="surname"/>

3 <relationfacet required="false" id="works_in_project" toEntityId="
    project" toEntityFacetId="project_name"/>
```

Listing 4.5: Examples for definitions of relation facets.

The facet name of a person artifact gets the semantic of creator or creating person or author for the document artifact by the assignment of a relationship description (e. g. has created). In the example shown in Listing 4.5, the surname of a person artifact is assigned the semantic of being the author of a document. Although, that facet is derived from another artifact (in this case from person artifacts) that facet belongs to the set of document facets. Similarly, the name of a project is assigned to a facet of a person which works in this project.

Appendix B.2 on page 181 shows a complete example of the schema file with several entity types and according facets.

4.6 Related Work

Ross and Janevski [2005] present work on searching faceted databases and aim for the definition of a general query language for hierarchically classified data, called the *entity algebra*. The authors show that their query language possesses low data complexity by having linear data complexity in terms of space and quadratic data complexity in terms of time. Additionally, the authors introduce a query engine for the domain of archeological databases. A query in their entity algebra takes entity sets as input and produces an entity set as output. The query language is compared to the relational algebra and maps its functionality excluding projections and cyclic hypergraphs. Similar to the LFRP-query concept, the entity algebra resorts to a pre-defined schema which defines the artifact hierarchies and the respective facets which are available. Similarity searches as well as ranking functionality are not considered in the entity algebra.

Clarkson et al. [2009] review various faceted search engines and tries to derive generalized formal models to describe these systems and find possible extensions to this query paradigm. One aspect the authors discuss are faceted data models to represent the faceted information properly. They start by using the Entity Relationship (ER) model and describe entities and their attributes / facets with and without relationships to other entities. Additionally, they define formalisms to represent queries (containing flat or hierarchical data) for faceted search. The authors introduce the notion of single- vs. multi-focus items which is slightly comparable to the multi-layer functionality of the LFRP-search framework but focuses more on relations between facets (in contrast to the relationships between artifacts which are covered in this publication).

Chapter 5

Prototypical Realization of the LFRP–Search Framework

This section covers various aspects of the implementation of the LFRP-search framework which was formally described in Chapter 4 on page 91. First, in Section 5.1 the visualization type of parallel coordinates is introduced which is suited for the display of multi-dimensional data. This is followed by the description of the architecture of the LFRP-search framework implementation in Section 5.2 on page 131. A detailed characterization of the user interface and the usage of parallel coordinates for the visualization of search queries and search results is given in Section 5.3 on page 134. Finally, Section 5.4 on page 152 compares the LFRP approach for handling multi-dimensional data with concepts from DWH systems.

5.1 Parallel Coordinates

In Section 2.5 on page 33 the advantages of visualizations in search interfaces were motivated and highlighted. Since the data found in the portrayed search scenarios is multi-dimensional, appropriate forms of visualizations need to be found.

During his studies of multi-dimensional geometries, Alfred Inselberg noticed the lack of appropriate visualizations and started to research how this kind of data can be visualized with the concept of parallel coordinates [Inselberg, 1985; Inselberg and Dimsdale, 1990][1].

The main idea of parallel coordinates is to project multiple dimensions of a data set into the two-dimensional space by drawing each dimension as an axis of equal height parallel to each other. Figure 5.1 on the following page shows an example of a visualized tuple with five dimensions. Along the x-axis, n parallel axes x_i (with $i = 1, \ldots, n$) are drawn with equal distance to each other. Each value of a tuple is marked on a location on the according axis which indicates the value relative to the other values of the attribute. The points of a tuple are then connected by a polyline.

[1] Alfred Inselberg is currently working at the Computer Science and Applied Mathematics Department at the Tel Aviv University, Israel. On his website he provides additional information about parallel coordinates (`http://www.math.tau.ac.il/~aiisreal/` last accessed 08/01/2010).

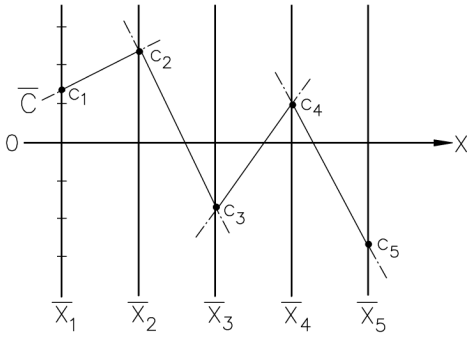


Figure 5.1: Visualization of a data tuple in parallel coordinates [Inselberg, 2005].

The lower and upper end of an axis represents the minimum and maximum value of the available attributes for the current record set for sortable attributes. The plot of the data in the two-dimensional space gives users a visual representation of the relationships between different variables.

The parallel coordinates visualization helps users to easily understand relations in the data by transforming multi-dimensional data into patterns which can be easily seen in the two-dimensional space. For instance, a set of points on a line in the Cartesian coordinate system is represented as a set of lines in the two-dimensional parallel coordinates which all intersect in one point. This linear relationship is denoted as line $\leftrightarrow$ point duality. The study of these kind of dependencies help users to make conclusions about relationships in multi-dimensional data [Inselberg and Dimsdale, 1990].

Depending on the scale of the shown attribute, the distances between the attribute values differ. Values of discrete attributes with a nominal or ordinal scale are usually drawn on the axes with equal distances. Cardinal scale attributes typically are represented linearly on the axis, although a logarithmic display might be applicable for certain kinds of attributes. The interval of the values of a cardinal scale attribute is distributed onto the whole length of an axis. In situations where attributes of the same type are analyzed such as temperatures at different times of day as illustrated in Figure 5.2 on the next page the different scaling based on minimum and maximum values for the current data set might hide relations between the data. The right side of Figure 5.2 shows the parallel coordinates where each axis is set to the same minimum value of 9.5 and it is immediately obvious that the temperature for each tuple at 6:00am is the lowest which was not visible before. Thus, users should be given the option to explicitly enable the same scaling of similar attribute types in the user interface.

Parallel coordinates cannot only be used for geometrical visualizations, but were later transfered to the domain of exploratory data analysis for multi-dimensional data [Wegman, 1990; Unwin et al., 2006]. For instance, Edsall [2003] used parallel

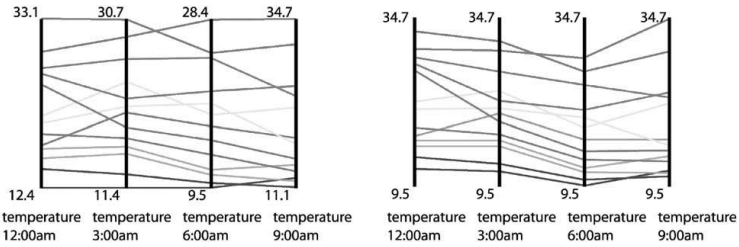


Figure 5.2: Scaling problems with cardinal scale attributes [Edsall, 2003].

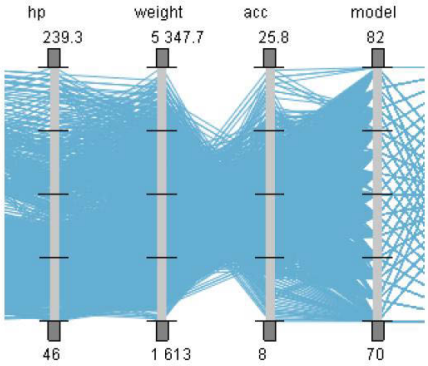


Figure 5.3: Overplotting in a parallel coordinates plot [Ericson et al., 2005].

coordinates to represent spatial and spatio-temporal data and linked it with other visualization forms such as maps and scatterplots. Tory et al. [2005] used parallel coordinates as a means for exploratory volume visualization in the field of medical imaging.

With large data sets the static parallel coordinates visualization does not reveal all contained information about the data set. Additionally, parallel coordinates can get very cluttered when visualizing large data sets. Ericson et al. [2005] call this effect *overplotting* which complicates the recognition of dependencies. Figure 5.3 gives an example of the effect of showing too many lines in the parallel coordinates. To overcome this cluttered view and to reveal additional information about the shown data several interactive features were introduced.

In general, it is necessary to provide users with the possibility to choose which attributes should be added to the parallel coordinates. Furthermore, they should be

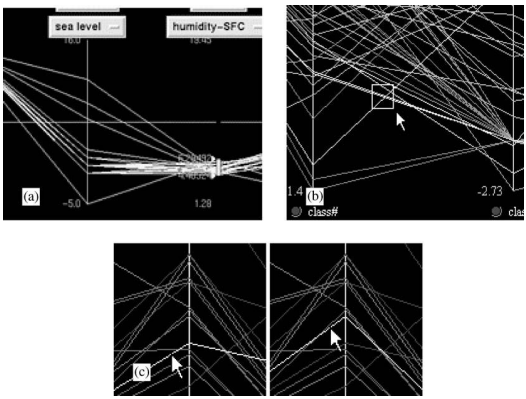


Figure 5.4: Interactions with parallel coordinates. (a) Focusing (b) Brushing (c) Strumming [Edsall, 2003].

allowed to hide attributes from the parallel coordinates so that they can focus on their task and are not distracted by too many axes. The parallel coordinates visualization only shows dependencies between attributes which are drawn on adjacent axes. By allowing users to switch axes they can easily change the order of the axes. Thus, dependencies and trends which exist in the data set can be better observed by users.

Edsall [2003] introduces several features which allow a deeper analysis of a subset of records drawn in the visualization. *Focusing* helps users to select a value or a range of the values of an attribute. Polylines of records which are not included in these selections are removed from the view. By that an easy reduction of a large dataset can be achieved. By *brushing* users are able to select a set of polylines by a selection method. Various prototypes allow the selection of polylines by click-dragging a box around the desired records. The chosen polylines then are highlighted to support the analysis of multi-dimensional data. A specialization of brushing is *strumming* where users can highlight a single record by hovering the mouse pointer over the specific polyline. The name is chosen since this action reminds of strumming a string of a guitar where the frets equal the axes of the parallel coordinates. A polyline is highlighted in real time along all axes of the parallel coordinates. Figure 5.4 shows an example for each of these three interaction types from Edsall [2003].

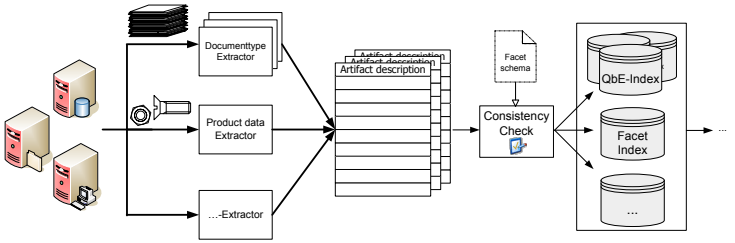


Figure 5.5: Architecture of the LFRP-indexing framework.

5.2 Architecture of the Framework

The prototypical implementation of the LFRP-search framework can be divided in two main parts which are necessary for the search engine. Figure 5.5 shows the architecture of the LFRP-indexing framework. Its main purpose is to index data from different source systems and to generate artifact descriptions. Due to data which might exist in multiple source systems it is necessary to transform the data in a well-defined format and remove redundancies. This transformation step is based on the schema containing the artifact type hierarchies described in Section 4.5 on page 121. Additionally, this step helps to correct different syntaxes and semantics and maps the data onto a uniform data schema. Finally, the data is stored in appropriate index structures which allow efficient retrieval during query time. The artifacts are specified by different features which can be used for querying. Indexing in the LFRP-search framework is not a singular process, but can be accomplished by two different approaches. On the one hand, the indexing framework can be notified from source systems when changes are made on artifacts which triggers the indexing. On the other hand, a crawler can—also in addition to the notification approach—search for new or changed information in the various source systems from the application landscape. With respect to this publication, the details of the indexing framework are out of scope. Further details about the indexing can be found in [Weber et al., 2009].

Figure 5.6 on the following page shows the general structure of the LFRP-query framework and illustrates the flow of a search query. As described in the sections above, the query elaboration process with the LFRP-search framework is of an interactive nature, i. e. users do not formulate the complete query in one step, but refine their queries in several steps being supported by the search engine.

After each change the query is handed over to the query framework. The central module is the *Search Handler*, whose main task is the control and delegation of the sub-tasks necessary for the evaluation of the query. Each sub-query is forwarded to the responsible *search module*. For instance, the faceting module is used for the determination of the query previews of simple attribute facets. For similarity facets specialized modules are employed which conduct the similarity computation.

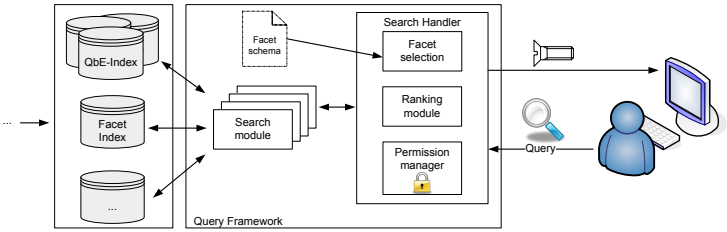


Figure 5.6: Architecture of the LFRP-query framework.

For instance, a module can determine the 3D-geometry similarity or conduct a full text search for textual contents. Each of these modules return a ranking which is necessary for the axis visualization in the parallel coordinates.

The ranking module computes the overall artifact ranking based on the different single rankings of the sub-queries and their according preference functions. Based on the current query and the artifact type hierarchies (including the facet descriptions), the facet selection module determines the list of currently available and eligible facets. Finally, the permission manager ensures that users only are returned artifacts in the ranking for which they have the necessary permissions.

The prototype itself was developed for the Java Platform[2] due to its interoperability and availability for cross-platform environments. The implementation of the framework is based on the client-server model.

The framework uses the Spring Framework[3] for the front and the back end to allow simple extensibility of the framework. By declaratively adding additional index storages, indexing components and search modules in the XML configuration supported by Spring, the implementation can be enhanced to support different domains and use cases.

Currently the back end of the framework resorts to two different index structures. Textual contents and attribute facet information are stored in an instance of the Apache Solr enterprise search platform[4]. Apache Solr is based on the open source library Apache Lucene which mainly focuses on full-text search. Solr is built on top of Lucene and provides a full text search server with enterprise features such as scalability, distributed search and index replication. Additionally, it supports built-in faceted search, clustering, database integration and hit highlighting. This solution was chosen as the base for the LFRP-search back end due to its stability,

[2] `http://www.oracle.com/us/technologies/java/index.html` (last accessed 08/01/2010)

[3] The Spring framework is a layered open source application framework for the Java/J2EE platform. Its primary goal is to simplify enterprise software development by providing a lightweight inversion of control container, an aspect-oriented programming framework, a data access framework and a Model-View-Controller (MVC) framework amongst others. More information about the framework can be found at `http://www.springsource.org/about` (last accessed 08/01/2010).

[4] `http://lucene.apache.org/solr/` (last accessed 08/01/2010)

functionality, and open source availability. The latter easily allowed the addition or modification of aspects of the framework which were needed for LFRP functionality[5].

Internally, the Lucene library uses an inverted index for the storage of the textual features which is suitable performance-wise for the determination of textual similarity as well as the faceted search computations[6].

The relation information between different artifacts and the artifact features for geometry and topology information are stored in an object-oriented database. The decision for this type of index structure was chosen due to its simplicity for the implementation and not performance-wise since this aspect was not the primary focus of the prototypical implementation.

The LFRP-search framework abstracts from the used index structures and offers a higher-level API to access both the indexing as well as the query functionality.

Currently, the communication between front end and back end is based on the HTTP invoker mechanism[7] from the Spring framework, but could easily be configured to use other technologies such as Web Services (with Java API for XML-based RPC (JAX-RPC) or Java API for XML-Web Services (JAX-WS), Remote Method Invocation (RMI), and Java Message Service (JMS).

The used approach depends on standard Java serialization. After each query refinement step, the changed search query is sent to the back end where the query framework determines the search results. The following information is sent to the user interface:

- The search result list with result previews for each artifact such as textual snippets or preview thumbnails for images.
- For each search result artifact the respective facet values of the chosen facets are included for the polylines in the parallel coordinates visualization.
- The currently chosen facets including the query previews for the current search results.
- The list of available facets from which users can choose in the next query refinement step. These facets are constrained by the artifact type hierarchies. Additionally, the relationships of the search results to related artifact layers are evaluated and when available provided to the user for multi-layer queries.

[5] Although Section 4.3 on page 100 introduced and explained the LFRP-query model with SQL statements, the current implementation in the back end prototype resorts to built-in functionality provided by Apache Solr to evaluate queries.

[6] Alternatively, the back end could have been realized with relational databases. There exist several approaches which combine the relational model with keyword based search, for instance [Agrawal et al., 2002], [Hristidis et al., 2003], and [Liu et al., 2006].

[7] For documentation refer to `http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/remoting.html#remoting-httpinvoker` (last accessed 08/01/2010) and for the API to `http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/remoting/httpinvoker/package-summary.html` (last accessed 08/01/2010).

5.3 Description of the User Interface

Hearst [2009] defines the function of a search user interface "to aid users in the expression of their information needs, in the formulation of their queries, in the understanding of their search results, and in keeping track of the progress of their information seeking efforts". In Section 4.1, the search user interface of the LFRP-search framework was introduced shortly. In this section several important aspects and funtions of it are highlighted and elaborated.

The introduced LFRP-search framework heavily relies on visualization techniques to support users in understanding the search results better. Due to the graphic nature of the visualization and the requirement to give users instant guidance after each query refinement step, the decision was made that the user interface prototype is realized in Java Swing[8] and is deployed via Java Web Start[9]. Java Swing provides an API for building graphical user interface for desktop applications. Although, the development of modern Web browsers is heavily focused on improving performance to provide "desktop application experience" for web applications, the choice of the Swing framework was chosen due its maturity and stability.

The user interface is divided into three main areas which are described subsequently. The partition of the user interface was chosen to prevent users from having to scroll down to see the search results which is recommended for better usability of search user interfaces [Hearst, 2009].

5.3.1 Controlling the User Interface

The top section of the user interface contains a ribbon component also known from current Microsoft Office[10] applications. The use of ribbons for controlling the user interface supports users in finding the options necessary for completing search tasks efficiently without having to refer to program documentation or help sections. Ribbons allow the placement of many different actions in a space-saving way by providing user with different tabs which can be populated with various different elements[11].

As shown in Figure 5.7 on the facing page, the ribbon contains three different tabs. The *Facets* tab is the primarily used tab and provides users with the possibility to add search criteria to the current query. The in-ribbon gallery in the *Add facet* ribbon group contains a set of currently available and valid facets from which uses can choose. The graphical representation of the facet in form of an icon can be set in LFRP-schema. The set of valid facets comprises attribute facets as well as similarity

[8] `http://download.oracle.com/docs/cd/E17409_01/javase/tutorial/uiswing/` (last accessed 08/01/2010)

[9] `http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136112.html` (last accessed 08/01/2010)

[10] `http://office.microsoft.com/en-us/` (last accessed 08/01/2010)

[11] Refer to `http://msdn.microsoft.com/en-au/library/cc872782.aspx` (last accessed 08/01/2010) for a detailed description of the ribbon user interface concept provided by the Microsoft Developer Network.

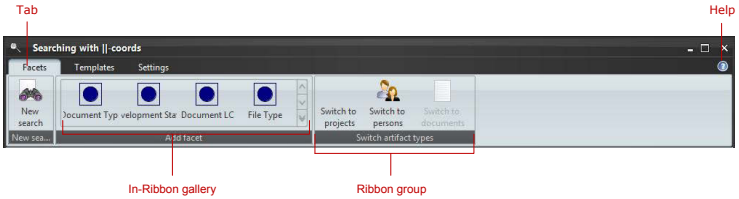


Figure 5.7: Ribbon component which is used for controlling the user interface.

facets. By adding an attribute facet to the parallel coordinates plot, the facet values and their respective facet counts are computed in the back end of the framework and are instantly visualized as an axis. When a similarity facet is added to the plot, initially, an empty placeholder axis is added to the parallel coordinates which offers the users the possibility to specify the search criterion for this facet. This search criterion can consist of a keyword query or an example file for QbE queries. In the former case, users are provided with an input field below the respective axis where they can enter their keywords. In the current prototype, the keyword search is applied to all textual contents including document contents and textual attributes. In the latter case, users are provided with the possibility to upload a file to the search engine back end which then computes the search results. Another approach to begin a similarity search is the invocation based on a search result artifact. For instance, users could choose the similarity facet by accessing a context-menu from the search result list which offers the applicable similarity methods.

The elements of the in-ribbon gallery are sorted based on artifact types. This functionality is necessary for facets which are referencing a related artifact layer and is detailed in Section 5.3.4 on page 144. Users simply select the desired search criterion and the appropriate facet is immediately added to the parallel coordinates. The ribbon band *Switch artifact types* is adjusted after each query refinement step and contains facets which initiate a layer switch (cf. Section 5.3.4 on page 144). Additionally, users can reset their search and remove all facets and facet selections to start a new query.

5.3.2 Parallel Coordinates for Search Query Formulation

The central part of the user interface is the parallel coordinates visualization which is responsible for two different tasks. One the one hand, it provides a visualization of the query and allows users to refine the query by conducting (additional) selections visually. On the other hand, the search results are visualized to help users understanding the search results better and to identify relations which exist between the different facets and between the artifacts. This is achieved by visualizing each search result as a polyline representing the facet values of this artifact in the parallel coordinates.

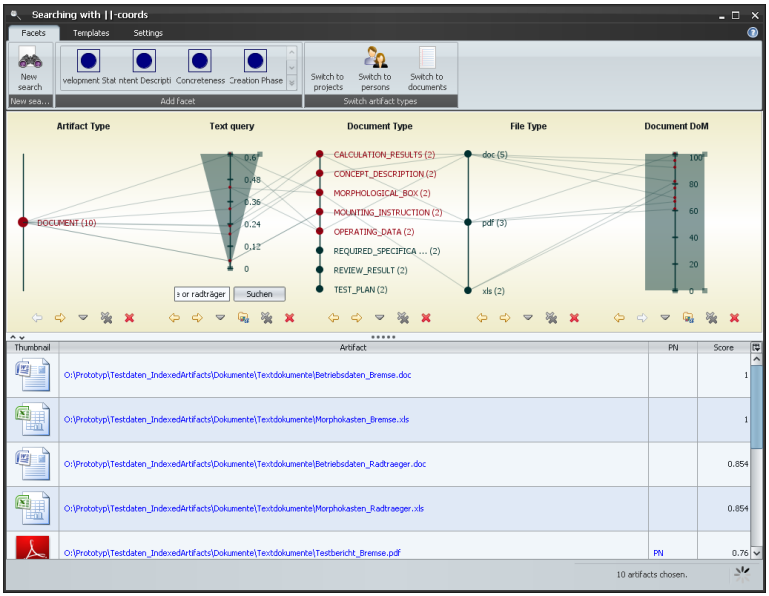


Figure 5.8: The search user interface of the LFRP-search framework with an exemplary search request.

Each chosen facet is added to the parallel coordinates and represented as an axis with marker points representing the different available facet values. As shown in Figure 5.8 each axis is labeled with its name. Below each axis users find a control panel which allows them to apply different actions concerning the axis.

Visualization of the Different Facet Types

In Section 4.3 on page 100 the different supported facet types were introduced along with the available query options. One of the primary goals of the LFRP-search framework is the harmonization of the query statement across different search paradigms such as text, geometry and topology similarity as well as faceted filtering. Thus, each sub-query—independent of the applied query paradigm—is visualized as an axis in the parallel coordinates. The presentation of the different axes based on the distinction of the scale of a facet differs slightly to be able to express all potentials offered by the specific facet type.

A scale is composed of an axis of fixed height. Based on that height, the facet value distribution for the facet on the axis has to be calculated so that on the one hand all available facet values can be displayed and on the other hand the distances

between the values carry a meaning. For example, if a cardinal scale is assumed the values could be arranged proportionally, whereas with a nominal scale a constant distance between values is more appropriate.

In the following, the possibilities are described how to represent the different facet types visually.

Facets with a **nominal scale** are characterized by facet values which cannot be put into an ordered sequence, e. g. the facet *project* addressing the project where a document was created. This fact raises a few questions. Though, no natural order exists, the facets still have to be aligned in an intuitive way for the user in the user interface. As each facet value "carries" a facet count which describes the number of artifacts which belong to that value, a frequency-based approach can be used which sorts the facet values decreasingly based on the facet counts. Alternatively, a lexicographic order is imaginable. Of course, other heuristics are imaginable for the sorting of the facet values. For example, the system can record the choices of the users and then order the facet values according to the most often chosen values. Since this approach relies on previous user selections, the system needs a certain amount of training data until it can work reliably.

In contrary to the just described facet type, **ordinal scale** facets include a natural order of the facet values. But still, the visualization of the distances between the facet values cannot be expressed by the facet value names as they do not carry a distance semantic. Thus, the facet values are displayed with equal distance on each axis.

For both types, each displayed facet value is marked on the axis and is described by a label containing its value and the number of artifacts in the search result which are assigned this value. Another visual aid for the users is the size of the facet value markers drawn in each axis. They vary in size proportional to the facet counts, so that users easily can understand for which facet values more search result artifacts are available based on the current selections. This is especially helpful in situations where the facet labels are not sorted based on the facet counts, but lexicographically for instance.

The configuration of the available visualizations and sorting of the facet values for these facets is done in the schema as introduced in Section 4.5 on page 121. If multiple comparators are given, users can easily switch the sorting on demand. Thus, users are given various options how the facets can be sorted and displayed in the user interface. Nevertheless, an approach which results in a known order, such as a frequency-based approach should be chosen by default, since various studies indicate that users prefer the predictability of a known order of the facet values (e. g. [Pratt et al., 1999]).

Both the nominal and the ordinal scale visualizations can suffer from too many facet labels which need to be displayed on one axis. In the worst case the axis labeling is so crammed that users cannot read the labels anymore. Here several solutions are imaginable. For instance, only the most probable ones (e. g. based on the interestingness measure of [Dash et al., 2008]) or the ones with the highest facet count could be displayed initially. The user interface should provide an option to include the facet values with lower counts using a dialog window. Those low occurrence

facet values could also be abstracted to an "Other" category. Alternatively, an aggregation can be made which clusters the entries lexicographically (e. g. A–E; F–R; S–Z). An important requirement for the realization of the search interface is that the entries in the axes are not visually moving too much. If the user restricts facets, the order and amount of the facet values can change, which might lead to a "fluttering" of the entries where the position of the facet labels change when additional selections are conducted.

These two facet types support single-value and multi-value selections. Selections are simply conducted by mouse-clicking the marker on the axis or the describing label. The refined query is then evaluated and the visualization is getting updated. For selections of multiple facet values, the user can initially select multiple values by holding the *CTRL* key and clicking the desired values similar to selections in file managers.

After the selection all unselected facet values are usually hidden to help users focus on their selections. In situations where users want to broaden their search by selecting additional facet values for a facet, the display behavior of multi-valued facets can be changed. For multi-valued facets users can decide whether the unselected facet values should be displayed in the parallel coordinates plot or not in the *Settings* tab. If unselected values are displayed users might be provided with additional insight about the values in the facet. An alternative realization of this requirement is the provision of a filter dialog which can be invoked from the control panel below each facet axis to add additional values.

For multi-valued facets which are enabled for multi-selections in the schema, users should be able to define whether a multi-selection is combined by a conjunction or disjunction. For instance, this can be implemented by providing a button in the control panel below the according axis which offers the available combination operators.

The visualization of the **cardinal scale** includes several key points which need to be addressed. In general, it is not reasonable to visualize all facet values because that would lead to a cluttered user interface. The front end has to calculate some marker values to guide the users so that they can comprehend the approximate values of a data record for that facet. Additionally, the actual facet values are marked by small red circles to show users the exact position of the facet value of an artifact. For analysis purposes of the search results, users are able to change the minimum and maximum value which is used for the display of an axis. Users can switch between a display of the axis which is restricted to the current minimum and maximum value of the current search result. Alternatively, they can display the whole codomain for a better understanding of the actual distribution of these facets. A product part search might be constrained by certain size restrictions. Users might be interested how the current parts in the result set are distributed on the complete scale and therefore display the complete range from zero to the current maximum. This functionality helps users to examine absolute and relative differences between several artifacts in the search result.

An alternative representation of this facet can be achieved by transforming the continuous values into an interval model by aggregating them to several intervals

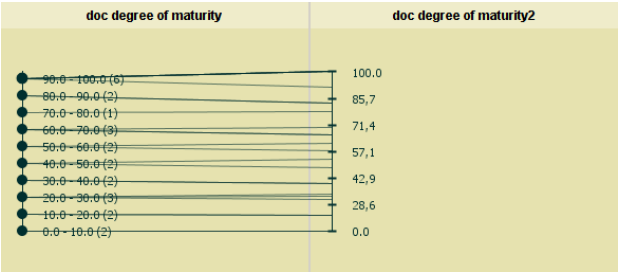


Figure 5.9: Displaying a document's degree of maturity as ordinal and as cardinal scale facet.

which represent the original values. The scale is then transformed into an ordinal scale because the order of the intervals can still be derived. Figure 5.9 illustrates this approach with the facet of the document's degree of maturity which shows users a quantitative measure of the completion of a specific document. The left part shows the interval visualization where the continuous values were aggregated to intervals of 10 percent spans. For instance, the interval *90.0–100.0* now comprises six documents, whereas in the right visualization the data points can be seen at *92.0* and *100.0*. The downside of this interval visualization is the restriction on the predefined intervals when users want to select ranges and might lead to the situation where users cannot select their desired interval.

Selections on cardinal scale facets are conducted using user preference functions and work for cardinal scale attribute facets as well as similarity facets. The functionality of user preference functions allows the easy statement of user preferences by simply "drawing" a function onto the axis of a facet. Obviously, for certain user groups this is a very helpful feature, but can overwhelm standard users. Thus, the user interface needs to support function templates from which users can choose. These function templates can be kept in the *Templates* tab in the ribbon component and show users the available function templates (cf. Section 5.3.5 on page 150).

Attribute facets which are added to the parallel coordinates initially show all values of the facet. Multi-value selections are conducted by "drawing" an interval with the mouse on the axis to define the upper and lower boundary of the selection. An example is given in Figure 5.10 on the next page in the first and the second axis. This constant function can be modified to give higher preferences on certain ranges. The user preference functions are changeable by the small rectangular handles in each corner. The user can simply drag a handle and change the function to the desired position. The current implementation of the user interface prototype realizes the example functions introduced in Section 4.3.4 on page 110 above which constrain the degrees of freedom for users. The placement of the handles of the functions are limited to two values. The handle can be positioned on the axis itself (denoting a weight of 0) or on the maximum to the right (denoting the maximum weight of 1

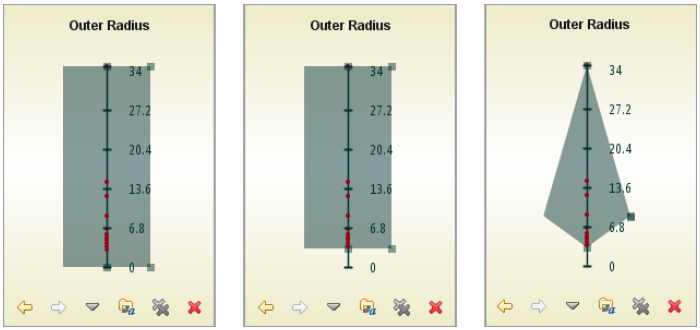


Figure 5.10: Several function overlays of the cardinal scale attribute facet outer radius.

for this facet value). If users move the handle to other positions in the function area of the axis, the user interface automatically moves the handle to the nearest point of these two extremes. This approach was chosen, to simplify the user preference functions. The relative weighting of certain facet values by giving a different width is not allowed, but could easily be implemented. The weights for the facet values between two chosen handles are determined by the defined linear functions between the handles. User can add additional handles on the functions by double-clicking on the desired position on the frame of the function. After each change of the function the search results are immediately updated. The options for users include the pruning of the search results by changing the boundaries of the interval to filter artifacts where the facet value lies below and/or above a certain threshold. The use of linear functions allows the statement of intervals where certain facet values are preferred and the artifacts are promoted in the search results.

Figure 5.10 shows different functions for the attribute facet *outer radius* of a product part. The first axis shows the view where a default interval function is overlaid covering all values. In the second axis users chose to exclude parts whose outer radius is too small by excluding the range from 0 to 3 by setting the lower boundary to a higher value. The constant interval function is still used. The right axis sets the preference on artifacts with an outer radius of 8 by applying a fuzzy interval which still considers artifacts with other radiuses. The result of the application of the shown function is that artifacts with an outer radius more close to the value of 8 are ranked higher in the search result ranking. This type of function is helpful in situations where users search for artifacts with an attribute in a certain interval where several values are preferred.

Although a rare use case, users can select single values by clicking on the desired data point on the axis.

Initially, similarity facets are overlaid by a linear function which covers the complete spectrum of the similarity scores to ensure the normalization of the similarity scores returned from the respective search module. The normalization makes sure

that the similarity scores which are used for the computation of the result ranking lie in the interval $[0, 1]$. Visually, the facet axis displays the similarity scores returned by the respective search module to the users graphically, i. e. the artifacts are shown on the axis according to their score. Similar to attribute facets with a cardinal scale, the function can be adjusted to the users' preferences. For instance, they can adjust the boundaries of the interval to prune the search results lying below a certain threshold.

If the default interval function is changed, the axis can be displayed in two ways. First, the scaling of the axis can remain as it was before the selection to illustrate the minimum and maximum value of that facet for the complete data (as introduced above). Second, the axis could be "zoomed" which restricts the axis to the current minimum and maximum of the chosen interval. Therewith, the single facet values are better viewable because they can be arranged on the complete height of the axis. The current prototype follows the first approach by default with the option to enable zoom.

The polylines in the parallel coordinates visualization illustrate dependencies between the chosen artifacts and their facets. In Section 2.4.2 on page 27 the functionality of backward highlighting was introduced which helps users in directional faceted browsers to understand how the current selections affect the facets to the left of the current selection which is summarized by the notion of "added facts" by Wilson et al. [2008]. The polylines also support this information since they connect the chosen facet values of all facet axes and help users to assess inter-column dependencies. This is especially helpful, when users chose to display all facet values (not only those for the current search result) of a facet in the *Settings* tab. Nevertheless, users can change the opacity of the polylines or hide them from the *Settings* tab in the ribbon component on top.

As described above, the search result comprises the search results and their facet values for the currently chosen facets in the visualization. The front end then determines the polylines which can affect the performance of the user interface in addition to the visual problem of overplotting which was introduced above. In two projects which were conducted during the research for this publication, it was shown that this front end centered approach can result in noticeable delays in the use of the user interface [Mechnich, 2008a,b]. This effect was the result of the calculation of the positions of the polylines as well as the time necessary to draw the lines in the parallel coordinates[12]. A solution to ensure the performance of the user interface can be prior calculations on the back end to ensure that no duplicate lines need to be drawn. Alternatively, the drawing of the polylines can be omitted until the number of search results is in a magnitude where the delay-free drawing and usability can be ensured. In the project works, the user interface was drawing the polylines when the search result comprised 1000 or less artifacts.

[12]It has to be noted that the realization of the faceted search user interface which utilizes parallel coordinates was realized in Adobe Flash in these two projects.

Interactions with the Parallel Coordinates

One main benefit of the search interface visualization is the attained insight and knowledge which can be acquired about the underlying search results. To tap the full potential of the visualizations, users are provided with a set of different operations which help them controlling the parallel coordinates. This supports the navigation in the artifacts [Siirtola, 2000]. A control panel for each facet in the parallel coordinates is situated below each axis and shown exemplarily in Figure 5.10 on page 140. Therewith, users can easily conduct changes on each axis.

A characteristic of parallel coordinates is that dependencies can only be determined for adjacent axes. Below each axis users find a left- and a right-pointing arrow which allows users to move this axis to the desired position. The polyline representation is immediately updated to reflect the changes. The search result which is transferred to the front end includes the complete information about the polylines of the current search result. Thus, no new query needs to be submitted and evaluated. Additionally, this functionality lets users determine how they wish to organize their search space to best support their focus of interest.

In queries comprising many different search criteria, the screen space for the visualization can get scarce. Users can minimize an axis horizontally by choosing the third button in the control panel, which hides the labels from the view and only displays the marker points on the respective axis. The control panel is also minimized and shows only the button to maximize the axis again. Additionally, the selections on minimized axes are locked so that no unintentional selection or deselections of facet value constraints can be conducted. Users are notified visually of this lock by a change of the cursor when they hover over the facet.

In addition to removing selected facet labels by clicking on them in the axis, users can also resort to the fourth button (with the two gray crosses) in the control panel to quickly remove all current selections on the respective facet axis. The button with the red cross to the right removes the facet along with the axis and the selections completely from the plot and the query.

For cardinal scale facets an additional button is found in the control panel which allows to set the relative weight between facets. Initially, all cardinal scale facets are weighted equally with respect to the computation of the search result ranking. By changing the weight slider users can influence the relative weights of each ranking facet as introduced formally in Section 4.3.4 on page 110. If a less granular weighting between facets is desired, this slider could be for instance replaced by a number of stars, where users could easily set their preferences between facets.

The polyline visualization is interactive as well. The common parallel coordinates features such as strumming and brushing are implemented. User can hover over the polylines with the mouse and the corresponding polyline is immediately highlighted in a striking color. Hovering over a facet value in an axis highlights all records which belong to this value, which allows user to better analyze dependencies in the plot. Alternatively, it is possible to select multiple records by clicking the corresponding polylines while holding the *CTRL* key. Simultaneously, the records in the result list are marked. Vice versa, by selecting search results in the list at the

bottom of the user interface the corresponding polylines are highlighted as well.

Further control functionality should be provided to the users. The configuration of these features was introduced in the 4.5 on page 121 and is summarized shortly below:

- Switch between multiple visualization of a facet if enabled (such as the cardinal vs. the interval facet display introduced above).
- Switch through the different configured comparators to change the sorting of the displayed facet values.
- Apply the zoom functionality for cardinal scale facets to either display the whole interval of the data set or the interval valid for the current search result set.
- Provide users with information about the current artifact when hovering over a polyline in the parallel coordinates. This could be realized with the display of a pop-up window. For instance, it may contain the artifact preview and the facet values to give users an instant feedback about the search results.

5.3.3 Presentation of Search Results

Whereas the parallel coordinates help to understand the search results and support query statement for users, the bottom part of the user interface contains the search result list and provides users with the possibility to open the found artifacts in the source systems. After each query refinement step, the list is constantly updated so that only selected artifacts are displayed. Studies showed that it is important to present users immediate results after each query refinement step [Plaisant et al., 1997]. This helps users to immediately see if their chosen search path is right or if adjustments to the query are necessary.

The current prototype shows the result in a tabular structure. Each search result is represented with a preview, the artifact name, and the respective RSV. Depending on the artifact type, different types of previews are used. For textual documents short textual previews are chosen if the query contained a search for keywords. Considering 2D- or 3D-CAD drawings, small visual representations are used to give users an instant idea about the represented artifact. The use of artifact previews helps users to find the searched artifacts better and faster in a search result list when they are provided with thumbnails [Woodruff et al., 2001].

The tabular display of search results allows users to add the currently chosen facets as columns. Additionally, the table representation allows users to sort the results column-wise if necessary. This is useful for situations where users did not utilize the ranking functionality, but simply want to order the search result artifacts based on certain ordinal or cardinal values. The search result table also offers sorting based on several columns.

Additionally, the table supports the grouping of its entries. For instance, if several artifact types are contained in the search result list, users can group the search results based on the artifact type.

5.3.4 Layer Switching on the User Interface Level

Section 4.3.3 on page 107 introduced the multi-layer functionality of the LFRP-search framework. Above two use cases were introduced which make use of the relationships which exist between different artifact layers. This section introduces various possibilities how the user interface can be realized for these two cases.

In the first use case facets from related artifact layers—the so-called relation facets—are used for the filtering of artifacts of the initial artifact type. To keep the current way of adding facets to the parallel coordinates plot, the relation facets are added similar to facets of the same artifact type to the plot. The currently valid relation facets are added to the list of available facets. These facets are placed in a separate area of that list to point out that those facets are from a related layer. This separation is necessary for better comprehension of the origin of each facet. So, users can see easily which facets belong to the current artifact layer and which originate from connected layers. This separation is natively supported by the in-ribbon gallery in the *Add facet* ribbon group by labeled separators. An example for this type of query is the search for documents which are be filtered by certain project names. Here the result artifact type remains on the document layer, but the filtering of the search results is done with a relation facet from the project layer.

The second use case includes the actual transition from one artifact type to another under the consideration of the current search results as initial constraint for the artifacts of the target layer. To visually separate this type of layer traversal more from the first use case, it is invoked differently. The list of artifact layers to which users can switch—based on the current selections—resides next to the list of the available facets in the user interface (cf. the ribbon group *Switch artifact types* in Figure 5.8 on page 136). This spatial separation to the facets from the first use case is necessary as the change of the artifact layer involves a more complex procedure in the user interface. If the current search result is of one type and has relationships to multiple artifact layers, users can choose which layer is appropriate for their current information need. Exemplary relations between artifact types were given in Section 4.2.1 on page 94.

The visual realization of a layer transition can be conducted in two different ways which are introduced below.

The first solution seems to be the more intuitive way as the parallel coordinates are simply kept with the current selections on the initial artifact layer and the layer switch to the connected artifact layer is conducted in the current plot. When new facets are added from related layers the current selections have to be marked as locked. Then, new facets from the related layer can be added to the current plot. When many facets are chosen on the first layer, the adding of further facets can introduce screen space problems. These can be partially counteracted by the minimization of some axes so that only the axis is displayed—omitting the facet values. Of course that reduces the visibility of the relationships between the facets and the obtained search results.

This implementation deviates from the classic visualization of the parallel coordinates where each artifact is represented by a polyline. Since this approach now

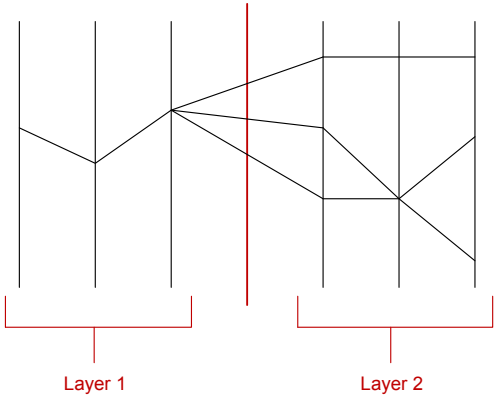


Figure 5.11: Example switch of the artifact layer in one parallel coordinates plot.

visualizes two artifact types which are connected by a relationship in one plot, the semantic slightly changes. The relationships between artifact types are usually modeled with a one-to-many cardinality. Thus, the representation of an artifact as a polyline has to be split in several polylines after the layer switch when several artifacts from the target layer are connected with the initial artifact. Taking the example of *product* artifacts as the origin layer of the transition and *document* artifacts as the target layer. In this case, the initial facets in the plot will represent product facets and the further facets will be document facets. Since a product can be described by multiple documents, the line in the plot representing a certain product will be split into multiple lines representing the associated documents for the document facets. Figure 5.11 shows an exemplary parallel coordinates plot for this issue with one artifact of the initial layer 1. The red line shows where the layer switch is conducted. It can be seen that there exist three related artifacts in layer 2 for the one artifact from the initial layer.

The second approach separates the view of the different artifact layers in multiple tabs containing both the parallel coordinates plot with the facets of one artifact layer and the corresponding search result list. The notion of a tab can be compared to the concept modern web browsers use to enable users to have multiple web pages open in one window. The user interface notifies the users about the available options for the artifact layer switch by showing icons for the respective layers in the ribbon group *Switch artifact type* next to the *Add facet* ribbon group. These icons do not represent facets as in the first solution but are triggers which initiate the layer switching. When users invoke the layer switching, a new tab is opened for the new parallel coordinates plot and the result list with the mapped search results based on the results from the first tab. Initially, the parallel coordinates plot shows

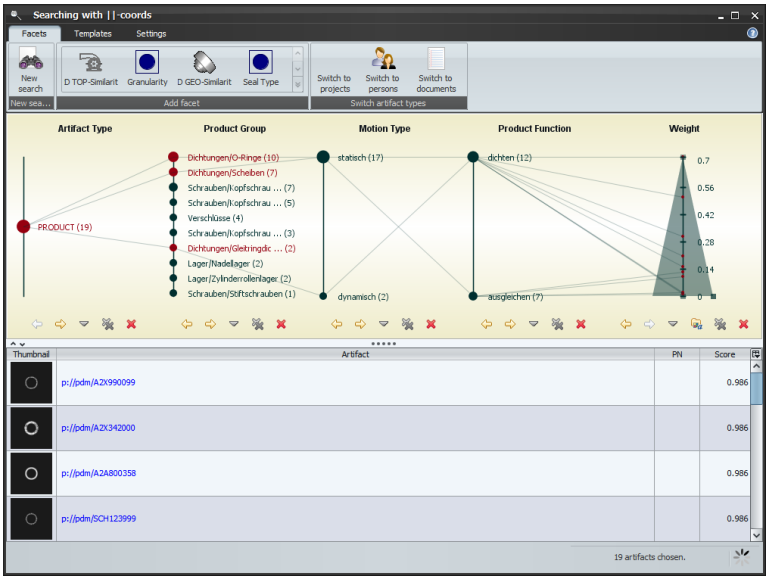


Figure 5.12: The user conducts a query on the product layer and constrains the search results by the artifact type, product group and weight facet resulting in 19 product artifacts in the search result. For the next step, the user wants to retrieve all documents which are linked to these products.

a marker axis which defines from which layer the users came to this layer to provide additional explanation on how the transition was conducted. Analog to a similarity facet this axis should show the calculated RSVs from the initial layer according to the applied fusion semantic to provide users with a better understanding of the propagated ranking values. Users can add facets of the second layer to the new parallel coordinates plot to further restrict the search results. Furthermore, users can switch between layers using the tabs. Selections and modifications made in one tab will reflect in the other tab.

Figure 5.12 and 5.13 on the next page show a visual prototype of the user interface for a search task which initially begun with a query on the *product* layer and then was continued on the *document* layer. Users can close the target artifact tab, when the results of a layer switch did not lead to the results they were looking for.

5.3.5 Potential Enhancements

Whereas the sections before described the general functionality of the LFRP-search user interface, this section presents several enhancements for the LFRP-search frame-

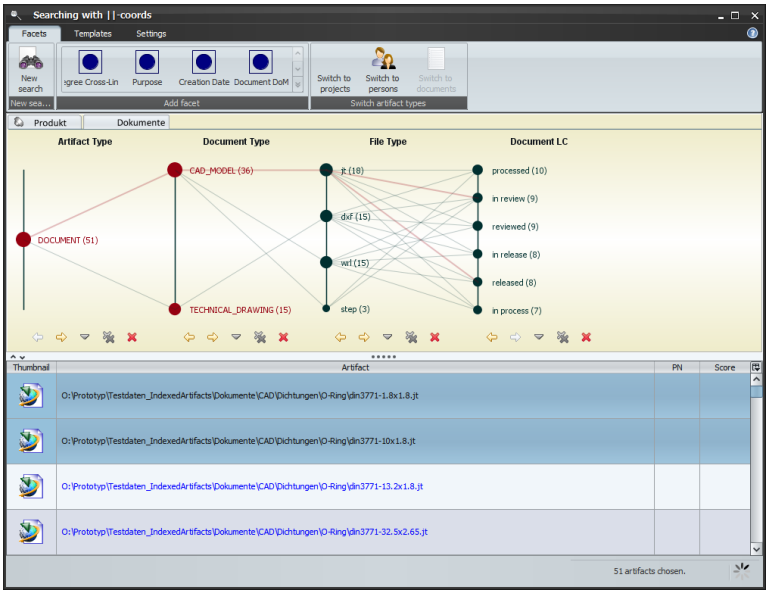


Figure 5.13: After choosing the *Switch to documents* button in the *Switch artifact types* ribbon group, a new parallel coordinates plot is opened in a second tab shown directly below the ribbon component. The figure shows user selections for the document layer on the document type facet. This search query results in 51 document artifacts which were based on the initial 19 product artifacts.

work which aim for a higher usability for the users.

Extensions to the Parallel Coordinates Visualization

In addition to overplotting issues, the parallel coordinates visualization can hide relations between different facets because of the polyline representation. For users it is difficult to follow the polyline of an artifact if there exist different artifacts which share the same facet value. Figure 5.14 on the following page shows this “crossover-problem” with two examples where users cannot tell which part of the line belongs to which artifact. Graham and Kennedy [2003] propose the use of curve segments instead of line segments for the polylines. In cases where the number of artifacts is not too large, users can more easily follow the line of an artifact. Figure 5.15 illustrates this solution for the example above and shows that the identification of the according line for an artifact is easier with this type of visualization. The authors use quadratic or cubic curves for the different segments between the axes.

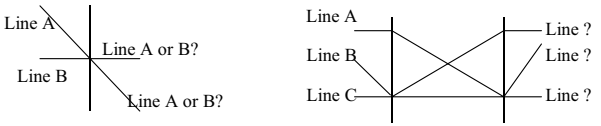


Figure 5.14: Using lines to connect facet values [Graham and Kennedy, 2003].

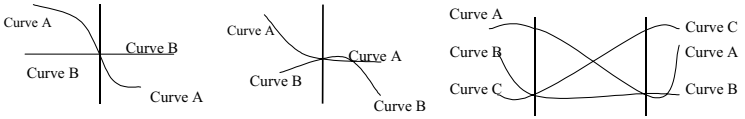


Figure 5.15: Using curves to connect facet values [Graham and Kennedy, 2003].

Additional information is hidden by the polyline visualization when a connection between two facet values of two adjacent facet axes is shared by several artifacts. In this case all artifacts are represented by a single line segment which does not allow to see how many artifacts are assigned to this facet value combination. Here the use of the curve visualization helps to reveal this information. Figure 5.16 on the next page shows the difference based on a prototype developed in the project work described in [Mechnich, 2008a]. The prototype was developed for the scenario of a search in the product range of an e-commerce system for a wine merchant. The visualized relationship in Figure 5.16 on the facing page shows the facets color (in German: Farbe) and type (in German: Art). The left part visualizes the relationship of wines with a red color and those which are of the type red wine by a single visible line. By changing the visualization to curve segments this relation is much more visible since all segments are shown.

Another approach to provide users with additional information about the distribution of facet values of a facet for the current search result is proposed by Hauser et al. [2002]. They compute histograms for each axis in the parallel coordinates which then are plotted over the axes. In situations where many artifacts are visualized, users still can easily see which facet values are more often represented in the current data. This substitute visualization could for instance also be used in situations where the search result is too large to show the polylines as pointed out in Section 5.3.2 on page 136.

Exposing sub-hierarchies of facets.

In Section 4.3 on page 100 the concept of hierarchical facets was introduced. From a user interface viewpoint it should be possible to easily drill-down into a hierarchy of facet labels. An approach to expose a facet sub-hierarchy of a hierarchical facet in the user interface is the use of a pop-up window which is automatically shown when users hover over a hierarchical facet label similar to the realization in the Fla-

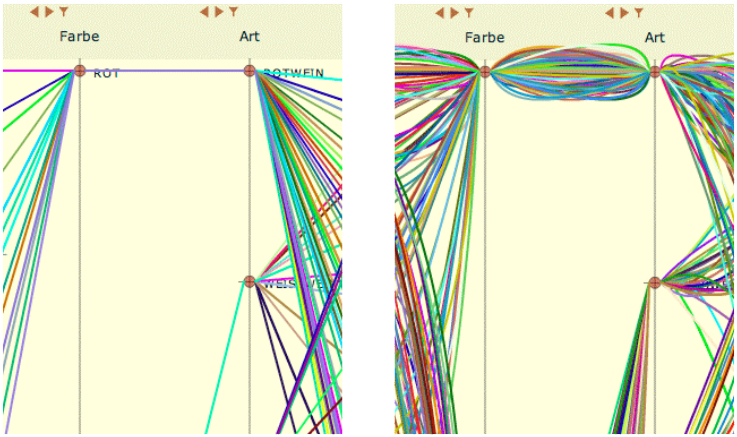


Figure 5.16: The left parallel coordinate visualization is based on line segments and omits the information that there are many artifacts with the same facet value combination. The right visualization shows the relation between the two facet values by using curve segments (Screenshots based on the prototype developed in [Mechnich, 2008a]).

menco project (cf. Section 2.4.1 on page 24). Users are then shown the labels of the sub-hierarchy from which they can conduct selections on these labels. Disadvantages of this approach are the possibility of a crowded visualization resulting from a very large number of facet labels and the complication of multiple selections. The latter problem arises since a selection would result in a facet axis which displays the chosen facet label from the sub-hierarchy omitting the labels from the level above.

Figure 5.17 on the following page shows a possible realization of a hierarchical facet axis in the user interface for a facet containing hierarchical geographic information. In contrast to flat facets, the facet description above the axis contains the current hierarchy level selection. In this case, users selected artifacts in the left axis which belong to the continent of *Europe* in the previous selection resulting in five different facet labels describing countries. The figure also shows query previews for further drill-down operations when users mouse-hover over a facet label. Users can conduct (single) selections directly from the shown pop-up window. Alternatively they can select a complete facet label of the current hierarchy level (in the shown case *Germany*) and the axis is adjusted to only showing the German states for which artifacts exist in the search result as shown in the right axis. The facet axis description is then adjusted to reflect this additional selection in the hierarchy. The latter selection approach can provide additional information about the search result, since the polyline visualization of the parallel coordinates is shown for the complete sub-hierarchy of the facet.

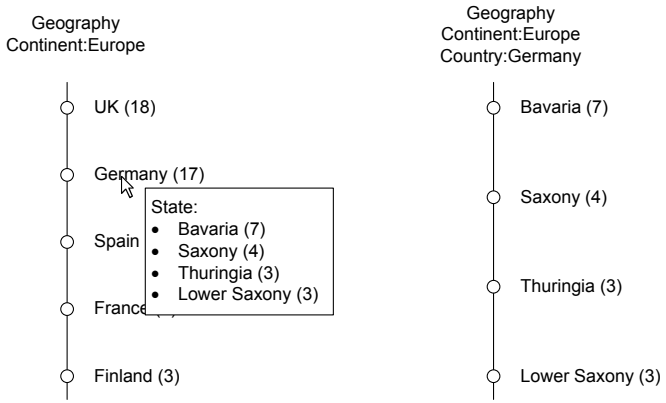


Figure 5.17: Potential visualization of hierarchical facets.

Templates

The LFRP-search framework offers a wide spectrum of functionality to conduct complex search queries. Nevertheless, this added functionality introduces additional complexity for the users which is not desired in every search situation. One approach to support users to work with the LFRP-search framework is the use of templates which are offered to the users. Templates are a more coarse-grained way of guidance for users.

The concept of the LFRP-search framework supports different types of templates. The most complete type are query templates. Expert users predefine faceted queries for certain search tasks. The definition of templates like this is done by specifying which facets are initially included in (i. e. relevant for) the search query and whether several facet values are preselected. This approach minimizes the degrees of freedom for users that are available during query-time. For each sub-query in the template it is defined whether the users can change the predefined selections. It is helpful for users to see and analyze dependencies for all chosen facets even if no changes to the facet itself are allowed. Additionally, the query options for each facet need to be defined. That comprises the use of keyword or QbE queries as well as the selection functionality itself. The availability of user preference functions for ranking facets can be limited to simplify query statement (by sacrificing some of the functionality of the framework).

The definition of these query templates should be conducted in the search user interface. Expert users should be given the option to save their current query as a template which can be customized in a new dialog window based on the description above. End users can access the templates in different ways. The user interface itself

offers the tab *Templates* which contains the available templates from which users can choose from. This approach still expects users to understand the purpose of each template. This tab also includes the saving functionality for queries.

The introduced approach of invoking search templates was restricted to starting searches directly from the search user interface. Often certain artifacts are needed in the application users are currently working in. Therefore, the search engine or queries itself should be invokable from these source systems. Assuming the use of a project-planning portal software which helps users to be guided through development processes, search templates which are typically used in certain process steps could be invoked directly from this system. This mapping of search templates to process steps or specific work tasks needs to be assigned beforehand. Additionally, theses application systems offer additional contextual information which might be usable for preselecting various facets in the query.

Another type of template affects the user preference functions. The possibility to change functions arbitrarily by "drawing" them in the parallel coordinates on the one hand is a very powerful feature but on the other hand introduces additional complexity for users. Thus, it is recommended that the user interface offers function templates which can simply be added to an axis. The simple functions described in Section 4.3.4 on page 110 should be considered for inclusion in the set of templates. The implementation of this type of templates could be realized by adding these templates to the *Templates* tab in the ribbon component where users could choose a template and drag and drop it on a ranking facet.

Search Dashboard

The idea of providing different types of templates to lower the barriers to use the LFRP-search framework can be broadened by expanding the search user interface to a search dashboard. Dashboards typically are used in executive information systems and provide aggregated visualizations of information from various sources. Transferring this concept to a search engine, users can be provided with advanced analysis capabilities.

In the following several functions for a search dashboard are proposed:

- **Saved searches.** Users should be able to save current searches in the dashboard to be able to simply access them again in similar situations in future projects.
- **Saved search results.** Especially in enterprise scenarios search tasks can be fragmented throughout the day (cf. Chapter 3 on page 69). Therefore, users should be supported in being able to interrupt a search task and resume it later without big efforts. The search engine should be personalizable, i. e. users have a user account for the search engine and can store their search history. When users end a search session, the current state of the search can be stored and resumed easily when logging into the system again. This functionality is also necessary when users want to save interim search results for later review. For instance for the domain of product development it can be

useful to search for different existing solution for a certain design problem. The different retrieved solutions can be stored in a digital notebook in the dashboard, which also allows annotating notes. After having identified useful solutions the engineer can compare the solutions collectively and choose the appropriate one.

- **Notifications/Alerts.** This function is related to saved searches. Users might be interested in being notified when artifacts are created or changed which fit certain search criteria. The search engine then constantly monitors the artifacts which are changed and indexed and notifies users when appropriate artifacts are identified. This notification could be realized by digests which are sent by email or are distributed by RSS feeds in certain configurable time intervals (hourly, daily, weekly, or so on). The configuration could be an additional option when users save searches.
  A similar functionality for web results is provided by Google Alerts[13] where users can specify keywords for which they want to be notified.
- **Additional visualizations.** To provide users with additional options to analyze the search results, a search dashboard should include other types of visualizations which can provide insights which are not possible to identify with parallel coordinates.
  Applicable visualizations can for instance be the examples described in Section 2.5.2 on page 36. However, additional visualizations in the search user interface can lead to difficulties, when users are not able to comprehend anymore which changes in a visualization affects the current query. When users are formulating their queries by using the parallel coordinates and later interact with other visualizations, the (currently) clear definition of the query statement gets blurry, i. e. the parallel coordinates are not anymore the single way to adjust the query.

5.4 Comparison of the LFRP–Search Framework and Data Warehouse Systems

Several aspects of the structure and the goals of the LFRP-search framework resemble those of DWH systems. This section will outline these similarities and will show differences with a focus on the application of both systems in an enterprise environment.

One of the first definitions of a data warehouse was made by Inmon:

> "A data warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management's decision-making process." [Inmon, 2002]

According to this definition, a DWH is characterized by four criteria which are necessary for decision-making processes and will briefly be detailed.

[13] `http://www.google.com/alerts` (last accessed 08/01/2010)

- **Subject orientation.** The selection of the data which is transfered to the DWH is chosen based on the necessary subjects for the analysis of indicators for decision processes and not for specific areas of operation.
- **Integration.** The used data is integrated based on several operative source systems of different areas of operation. The source data usually is stored in heterogeneous structures in the source systems.
- **Time Variance.** Data about a subject is not only stored at one point in time but stored so that analyses can be made which compare the data in a timely fashion. Thus, changes in time on the data can be evaluated.
- **Non-Volatility.** Data which is stored in a DWH is neither changed nor deleted (which is important for the time-variant characteristic).

Both DWH systems and the LFRP-search framework aim at providing users with insight about multi-dimensional data in which they can navigate arbitrarily. Both systems provide users with a toolset for the analysis of the data in the respective data structures. Therewith, users can conduct flexible searches in the data set.

The data set itself is built by integrating information from different source systems. DWH systems usually access systems of different areas of operation in an organization whereas the LFRP-search framework primarily accesses management systems for the different supported artifacts in the organization.

The analysis of data in a DWH is a dynamic process involving different operators which help navigating the multidimensional data structure and follows the principle of OLAP which was coined in [Codd et al., 1993]. The principle describes the interactive, exploratory analysis based on multidimensional data structures called hypercubes. The elements of a hypercube consist of different *measures* which are classified based on different dimensions. An example are sales figures of a company which are described by dimensions such as time and geography. A general overview of DWH and OLAP technology is given in [Chaudhuri and Dayal, 1997]. For more detailed information refer to [Barquin and Edelstein, 1997] or [Kimball and Ross, 2002].

Besides the multidimensional data model, OLAP also defines a set of specific operations for the analysis of the data.

- **Drill Down and Roll Up.** Both operators navigate along a dimension hierarchy in the hypercube. *Drill Down* navigates deeper in the hierarchy, for instance by specializing the view of the sales figures from a monthly aggregation level to a weekly view. *Roll up* is the opposing operator which aggregates the dimensions. Both operators allow the hierarchical navigation of the information in the DWH. Based on the example above, an analysis can begin with an aggregated sales figure for a calendar year. For instance, by drilling down the sales figures can be compared on a quarterly, monthly or weekly basis. Alternatively, the sales figure can be compared based on the different countries or on the subsidiaries a company is distributing their products.
- **Slice and Dice.** Both operators create individual views on the multidimensional data. The *slice* operator selects a slice of the hypercube by aggregating

a measure along one value of a dimension. An example is the analysis of the sales figure along the dimensions time and geography for one specific product. Dicing consists of slicing on more than two dimensions of the hypercube such as analyzing the sales figures for a specific product and a specific time span.

- **Rotate.** Rotation (or Pivoting) allows users to rotate the hypercube by transposing two dimensions to analyze arbitrary perspectives.

These hierarchical dimensions of the different measures equal the hierarchical facets found in the LFRP-search framework. By selecting facet labels of hierarchical facets users can conduct *drill down* and *roll up* operations similarly to the OLAP functionality. The *slice and dice* operators equal the selection of a faceted category allowing the analysis of the remaining facet dimensions.

Differences between the two approaches lie in the different target groups. Whereas the LFRP-search framework targets search situations with complex information needs, applications of DWH systems often target business-management problems such as planing the pricing of products and the planing of a range of products. DWHs are often part of Executive Information Systems (EISs) which aim for supporting the management of an organization. The LFRP-approach is mainly targeted at search tasks at the operational level such as supporting design engineers to retrieve information which helps them to develop better products. This also mirrors in the results the different systems return. The approach introduced in this publication uses artifacts which are classified based on different dimension (i. e. the facets describing attributes of an artifact). The framework mainly tries to help users find certain artifacts necessary for their current work tasks. Here the multi-dimensional data and its visualization and query options help users by giving them a better understanding about the search results. In addition, the framework offers the use case of stating QbE queries where users search for artifacts based on an example object which triggers a similarity search. This type of searches is out of scope of DWH systems.

In contrast, the goal of DWHs is the identification of connections by aggregating and analyzing data to help controlling business processes. Thereby, quantitative data (i. e. the measures) which is classified with qualitative data representing the dimensions is used. The results are usually standard or ad hoc reports which provide aggregated views of enterprise data.

Considering the data which forms the basis of these two approaches, the index structure of the LFRP-search framework is updated when the source artifacts in the source systems are changed and primarily current data is stored in the index. DWHs allow the access to historical data to analyze variations in time of the data and therefore keep historical information which is never altered (as described above with the non-volatility and time variance characteristic) after loading it in the DWH.

Both approaches aim for the integration of data with different structure from heterogeneous source systems. The process of integrating data from different source systems in a DWHs is called ETL [Kimball and Caserta, 2004] and solves difficulties which also appear in the indexing process in the LFRP-search framework.

- **Extract.** This initial part of the process extracts data from the various source systems. In this step, it is decided which parts of the data are included in the DWH. This comprises internal and external sources which are analyzed to be able to decide which objects and attributes need to be included for the intended analyses. Furthermore, update strategies for the data needs to be defined. Periodically, changes in the data needs to be determined and transfered into the *staging area* of the DWH.
- **Transform.** The data integration from different systems necessitates the transformation of the individual schemes of the source systems into the DWH-schema. Due to different utilized database systems and data models as well as conflicts in the original schema, a *schema integration* is necessary for the transformation. Based on [Batini et al., 1986] an integrated schema has to adhere to the criteria of completeness, correctness, minimality, and understandability. They also provide a comparative analysis of methods for database schema integration in [Batini et al., 1986]. Spaccapietra et al. [1992] propose an assertion-based approach for schema integration.
  Schema conflicts can occur due to aspects concerning different data types, table structure conflicts, naming conflicts or missing attributes amongst others [Kim and Seo, 1991]. An exemplary problem is the consistency between objects which exist in multiple data sources. As the identifying keys of the same object often differs in the different systems, surrogate keys must be created and managed to ensure the identification of the same object from multiple systems. Solutions to solve these difficulties are shown for instance in [Sheth and Larson, 1990]. The conducted data transformations comprise syntactic as well as semantic aspects of the data.
- **Load.** The final stage of the ETL process loads the transformed data in appropriate data structures in the DWH. Initially, the complete data needs to be loaded in the DWH which then is updated periodically to reflect updated data in the source systems. In particular, the initial loading is very time-consuming which might be problematic due to the induced system load.

These difficulties also are existent when a search engine such as the LFRP-search framework is deployed in an organization. As the framework also supports unstructured content in addition to structured data, additional issues need to be solved. More information about the indexing part of the LFRP-search framework which covers some of these difficulties can be found in [Weber et al., 2009].

Recently both enterprise search vendors as well as academic researches recognized the similarities and the relationship between DWHs and enterprise search solutions. Evelson and Brown [2008] emphasize the requirement to bring search technology closer to BI applications to provide better insights of the information in an organization. In Section 2.7 on page 53 several enterprise search solutions were introduced whereas for instance the solutions of Endeca and Exalead are trying to include BI functionality such additional visualizations and better aggregation operators.

Ben-Yitzhak et al. [2008] pursue an interesting approach which extends traditional faceted search by "flexible, dynamic business intelligence aggregations" which offer more (qualitative) insight about the data than the classic quantitative query previews for artifacts. The authors apply aggregations on sub-categories of arithmetic and Boolean functions over numerical attributes of the artifacts. Their model supports the *min*, *max*, *average*, and *sum* functions in addition to the *count* function for the quantitative query previews. The authors extended the Lucene search library to support their search model.

The authors use the scenario of a book catalog which is accessed by their extended faceted search model. The application of their aggregation functions allows the determination of criteria such as the average price of books in a faceted category, the number of best selling books of an author (by counting the number of books in the 5000 best-selling books). The approach allows the inclusion of the similarity score of the initial text query in the computation of the aggregated previews which can be used to show users the maximum score for a faceted category.

Additionally, the authors introduce the notion of *dynamic facets* which are not yet known at indexing time and have to be determined at query time. An example are temporal facets in the form of "artifacts dated in the last day/week" or spatial facets which represent the count of documents which lie in certain radii.

Chapter 6

Evaluation

This chapter provides an assessment of the applicability of the LFRP-search framework for complex information needs. The evaluation is two-fold. First in Section 6.1, the assessment criteria from Tunkelang [2009] are used to evaluate the complete LFRP-search framework as a search engine operating based on the HCIR model. Second in Section 6.2 on page 160, the user interface is evaluated based on the eight characteristics of search user interfaces defined by Shneiderman et al. [1997].

6.1 Evaluation of the LFRP–Search Framework

In Section 2.1 on page 11 three goals were introduced for search engines which are based on the HCIR paradigm: *Transparency*, *Control* and *Guidance*.

Transparency

The characteristic of *Transparency* shall ensure that users understand why the search engine returned a particular search result. The LFRP-search framework defines a concise query model formally. This model strongly relies on users clearly specifying their queries visually. In the current implementation no supplemental information is added to the user query. At all times users can see and review their queries in the parallel coordinates visualization.

For faceted queries in the user interface the transparency characteristic reflects in the parallel coordinates visualization. Thus, users need to understand the metaphor of an axis in the plot to equal an attribute of the artifacts with certain selected values.

As described in Section 5.2 on page 131, similarity queries are handled in specific search modules which implement algorithms to compute the similarity between an example object and the artifacts in the index. These computations usually are more complex to understand than the Boolean filtering which takes place for attribute facets. Additionally, the applied algorithms often are not exposed to end users. Still, a basic understanding of the concept of a similarity facet is assumed,

i. e. for the domain of product development users should be aware of the difference of geometry and topology similarity whereas thorough knowledge of the algorithms is not necessary. Thus, they need to be provided with more details why the artifacts in the search result are similar to the query object. Depending on the used similarity method, users need to be informed visually—if possible—how each result relates to the query object. Marchionini and White [2007] propose the use of *document surrogates*[1] which are summarized information about the document to provide users with a better understanding of the search artifact. This can be information based on the content of an artifact or metadata summaries. For instance, an artifact preview in form of a thumbnail can help users to see the similarity of a product to the given QbE object. Taking the example of a 3D-similarity query, users are shown a thumbnail of the three-dimensional representation of the product for each object in the search result list. For text queries, it is helpful to provide users with text summaries of the artifacts where the query terms are highlighted [Marchionini, 1995]. The current implementation of the prototype realizes these previews for textual queries and for 2D- and 3D-geometry queries. For the latter a small graphical representation is shown for each search result in the result list. The same representation for the query object is displayed in the respective similarity facet in the parallel coordinates.

Whereas the transparency of queries which focus on one artifact type is ensured by displaying the current query in one parallel coordinates plot, the layer switching functionality complicates the compliance with this characteristic. By moving the query of the target artifact layer to a second tab in the user interface users lose some transparency as the query of the initial layer is not shown. The user interface provides users with a marker axis on the second tab which visualizes the search results from the initial layer. This axis shows the initial search results as a facet with a cardinal scale and provides the link to the query on the original artifact layer.

The user preference functions which are used for specifying preferences of facet values for the ranking computation are an advanced feature which tries to simplify the statement of preferences. Although the LFRP-query model supports arbitrarily complex continuous functions, users are restricted to linear functions to reduce the complexity of the user interface.

A means which could be integrated for users is a natural language query visualization which provides users with an easier understanding of the selected search criteria (e. g. “Search for any products from the product group *screws* and the material *steel*” similar to the implementation in the VQuery system [Jones, 1998]).

In summary, the LFRP-search framework provides users with transparency of the search engine as the complete query is visualized in the user interface and no automatic query augmentations takes place. The functionality of the layer switching and the user preference functions can lead to diminished transparency for users.

[1] For the case of this publication the term *artifact surrogate* is more appropriate to reflect the different supported artifact types in the LFRP-search framework.

Control

The characteristic of *Control* aims for giving users tools to involve them more in the information-seeking process. These tools comprise options to influence the filtering as well as the ranking of the search results.

The LFRP-search framework allows users at each point in the search process to change sub-queries to reflect the obtained additional insights about the results. Users have control over adding additional facets to the query to restrict the search results or to remove facets to broaden the search scope again. Additionally, they can adjust selections on a facet by adding or removing facet values. Facets without selections also can be added to the parallel coordinates to allow additional analysis of the search results.

The feature of user preference functions gives users a fine-grained control over the ranking. Users can adjust their preference for the chosen facets as well as for specific value (ranges) by applying these functions visually in the user interface. Function templates help users to keep the complexity which is introduced by these functions low.

All these changes to the query are immediately reflected in the user interface which behaves according to the concept of dynamic queries [Shneiderman, 1994]. Both the parallel coordinates visualization as well as the search result list is updated to reflect the current query.

Guidance

Although helpful, the given control can lead to additional complexity of the user interface. Thus, users need to be *guided* through the search process. To cover the complexity which is introduced by the support of multiple artifact types where each offers different search criteria, the LFRP-search framework can be customized to work only on a subtree of the artifact type hierarchy. This is useful in situations where the information needs are more concise. The restriction to the subtree helps users by reducing their search options.

Additionally, the currently available facets for the current state of the query are presented to the users. This feature called *dynamic facet provision* prevents users from choosing facets which would lead to an inconsistent state of the query.

Furthermore, the framework supports the inclusion of specialized comparator methods which influence the display of the facet values of each facet. This functionality is necessary to provide users with facets which might help them in their current search task. For instance, the order of the facet values in the display could be arranged based on the “interestingness” measure introduced in [Dash et al., 2008].

However, an aspect which currently is not under consideration of the framework is the inclusion of contextual information which is automatically evaluated and used in the search interface. Conceivable approaches could comprise the use of contextual information to preselect certain facets along with their values to give users an initial search query. Furthermore, the search could restrict the search options automatically to a sub-tree of the artifact type hierarchy based on the current work or

search task.

6.2 Evaluation of the User Interface

Shneiderman et al. proposed eight golden rules of interface design in [Shneiderman and Plaisant, 2005] which have been rephrased in [Shneiderman et al., 1997] for the context of information retrieval. A search user interface should have the following characteristics:

- **Strive for consistency.** User interfaces should be consistent in the use of the terminology, the instructions, and the layout including colors and fonts.
- **Provide shortcuts for skilled users.** All functionality should be reachable easily, for instance by providing keyboard shortcuts.
- **Offer informative feedback.** Users need to be informed about all aspects of the search process including the used sources, query fields, etc. After the search is complete, users should be aware of what was searched and why the search results were returned.
- **Design for closure.** Users should easily notice when they searched the complete data set or the result list.
- **Offer simple error handling.** When errors occur during the search process users should be presented with comprehensible support explaining the occurred error. Additionally, changes to the query should be easy to apply.
- **Permit easy reversal of actions.** User interfaces should give users the ability to reverse single actions in their search process, for instance by allowing them to revert selections by accessing a search query history.
- **Support user control.** Users should be able to specify queries in any order. Thus, a search engine should not force users to follow a strict sequence of steps during query refinement. Users can also get additional control over the search by being presented visual overviews over the data set. This helps to gain a better understanding of the data and make more controlled search query refinements.
- **Reduce short-term memory load.** Search interfaces should be designed that users always are shown the relevant information avoiding situations where they need to remember or keep track of it.

Below, the search user interface of the LFRP framework is checked for the fulfillment of these characteristics.

Strive for consistency

The conception of the user interface for the LFRP-search framework especially focused on providing users with a consistent metaphor to state queries that combine different types of sub-queries. The use of an axis representing a facet in the user

interface as the general instrument to conduct selections helps users to understand the implications of their selections better. All supported types of facets operate in a similar fashion.

The concept of similarity facets allows users to provide the search engine with additional information as input for the query. Dependent on the type of similarity, users provide different types of input such as keywords or example files. The presentation of this query stays the same, i. e. users are presented with an axis which shows the search results sorted based on the used similarity measure in the axis.

Although an advanced feature, the user preference functions introduce a concept which allows users to state preferences about facet values of a facet. The graphical representation of preferences which can be easily adjusted by the mouse shows users visually which values of their selections are considered more important for the determination of the search results. This feature works for different types of facets such as similarity facets and attribute facets.

Provide shortcuts for skilled users

Users can control the user interface by using the keyboard and a pointing device such as a mouse. The search engine is mainly controlled through the use of the ribbon component on the top part of the user interface. The use of this interface component helps to keep the length of the menus short. For skilled users, the ribbon component is completely controllable by keyboard shortcuts which makes operating the user interface more efficient.

The enhanced functionality of the LFRP-search framework such as the multi-layer functionality and the user preference functions can be offered to expert users. As these are more complicated to handle, a plain keyboard control is hard to achieve and thus, it is primarily resorted to mouse input. To simplify these features, the user interface provides different templates for user preference functions which can be overlaid on an axis by choosing them from the ribbon component.

Offer informative feedback

One goal of the LFRP-search framework is to simplify the statement of difficult search queries in situations with complex information needs. As described in previous chapters, complex information needs comprise multiple search criteria and may concern various artifact types.

The LFRP user interface shows each chosen criterion as an axis to visualize the current query to the users and to provide them with additional transparency. The polylines which connect the different axes (i. e. the chosen criteria) help users to gain a better understanding of the chosen search criteria as well as the search results by visualizing dependencies. This analysis functionality also allows users to add facets without conducting selections on them to provide additional insight. After each change to the query, the result is instantly updated which helps to understand how the changed query affects the search results. The feature of the user preference

functions help to better comprehend the weighting between different preferred facet values visually.

Apart from this query and search result visualization, users are not provided with information why the result was returned by the search engine. Users could be supported by showing them a text representation of the query similar to the VQuery system [Jones, 1998] as introduced in Section 2.5.1 on page 34. Additionally, artifact previews such as thumbnails or textual snippets support users in better understanding why the results were returned.

A requirement for offering informative feedback lies in conveying which sources were used to compute the search result. The current implementation of the user interface only marginally realizes this. The result list allows users to access the result artifact by accessing the source system. Here, users should be supported better by the search framework. Solutions could comprise a source selection features where users can initially choose the source systems they want to access with their search. Alternatively, different sources could be automatically chosen by the search framework based on the users' context. For instance, the search engine could choose different initial sources based on the place where the search was started. Considering the integration of the LFRP-search framework in a CAD software, the searches mainly circle around products. Thus, a search in the company's PDM system is likely.

Design for closure

This characteristic demands that users are able to notice when they searched the whole dataset. With respect to this requirement, the LFRP-user interface only partially fulfill this characteristic. The query previews which are automatically computed for each facet value give users an overview of how many artifacts are available for a query refinement with a specific facet value.

Offer simple error handling

The LFRP-search framework tries to prevent users from making errors during the statement of their queries. By relying on the paradigm of faceted search users are prevented from making query selections which would result in empty results. Additionally, conflicting sub-queries are prevented since users are only offered available facets and facet values by the feature of *dynamic query provision*. Thus, only similarity facets can lead to difficulties during query statement. For instance, a similarity query can fail if users provide the wrong example object which is not applicable for the chosen similarity type. In such cases, the search engine should notify the users of this error and provide them with an explanation of which types of data are applicable for the facet type. Furthermore, the advanced feature of the user preference functions can lead to erroneous usage and thus should be kept simple.

A difficulty which should be communicated to the users by the search engine is the fact, that similarity facets can lead to zero results.

Permit easy reversal of actions

All user selections can be reverted independently from all other sub-queries. This includes the removal of single facet selections as well as complete sub-queries by removing the facet axis.

To simplify this process, the user interface should be enhanced by a search history functionality. Apart from being able to execute past queries again, this function would allow users to access any snapshot from the query statement process to revert recent steps.

Currently, the LFRP-search framework does not include contextual knowledge. Thus, users create their queries completely on their own and have an understanding of the consequences of each sub-query. When the framework is used with templates or contextual information which automatically preselects facets and facet value selections, users should be able to remove those selections if they do not fit the current information need. This also should be feasible with the search history functionality.

Support user control

This requirement is similar to the above described characteristic of *Control* defined by Tunkelang and thus it is referred to the description in Section 6.1 on page 157.

Reduce short-term memory load

Search queries which apply to only one artifact layer are displayed completely on one screen. This directly helps users to keep the memory load small since they can comprehend the complete query. In contrast, multi-layer queries raise the users' memory load since each layer query resides on a different (but still logically connected) tab. Hence, the deployment of this feature must be weighed up against the additional power of the available search features.

The current search result display in the list view allows users to easily access the search results, but for a larger result sets, users would have to scroll to see all results. When users do not want to restrict the search result any further by facet selections, additional visualizations might support them in the task of choosing the right result.

The above described search history functionality can also support the requirement to reduce the short-term memory load. This feature allows users to continue their search sessions where they left off before or which they aborted.

6.3 Summary of the Evaluation

The evaluation based on the two different viewpoints from the previous sections showed that the LFRP-search framework qualifies for the support of users in complex search situations with the help of the search paradigm of HCIR.

An aspect which should be supported stronger is the use of additional visualizations to provide users with an even more comprehensive understanding of the data

collection under consideration. A pivotal point is the distinction of a visualization as a visual instrument to provide additional insights and the use of a visualization as a query instrument. Currently, the visualization type of parallel coordinates is used as a single tool to formulate a query. By introducing additional visualizations which allow selections on the current search result, the problem is introduced that users may not be able anymore to focus on which visualization is responsible for a change in the search results. Thus, either the different visualizations are synchronized, i. e. both change when a selection is conducted, or only one visualization can be used for interactions which apply to the query.

Lastly, the user interface should be better integrated in the application systems used by the target users. This leads to a closer integration which helps users to focus on the queries when they occur during their work tasks.

Part III

Outlook

Chapter 7

Conclusion

The main goal of this thesis was the analysis of difficulties and problems of retrieving information in an enterprise setting and the conception of a search framework to support users in coping with these complex search situations.

The analysis covered various search scenarios from the industrial partners participating in the joint research project FORFLOW as well as multiple studies that analyzed user search behavior. By examining these search scenarios, requirements for an enterprise search engine could be identified that support interactive search processes in complex work situations.

Furthermore, the thesis introduced modern search paradigms such as exploratory search and faceted search and gave examples of implementations. Additionally, an overview about the topic of enterprise search and its challenges was given. Different types of currently available commercial enterprise search systems were introduced along with their abilities and limitations.

The main part of this thesis focused on the introduction of the LFRP-search framework which aims for a better provision of existing information by combining multiple known search paradigms. Thus, the acronym LFRP defines the four constituent parts on which the framework is built on. The *multi-layer functionality* supports users in stating queries which comprise multiple different artifact types. Therewith, users can query for artifacts by filtering the result set by facets defined in related artifact layers or switch the search result artifact type during the query based on a result set of related artifacts.

The *faceted search* paradigm provides users with an intuitive way of composing queries iteratively based on facets which are describing the search results. By providing users with a clear metaphor on how filter criteria can be added and removed from the query, users can easily comprehend which effect a specific search criterion will have on the search result if chosen.

The *ranking* functionality enhances the search options by providing users with a tool to add preferences on facet values based on user preference functions which are integrated with the other major parts of the framework. This functionality allows users to easily add and remove importance from certain search criteria in comparison to the binary view of boolean selections.

The concept of *parallel coordinates* is used to represent the search space in show-

ing both the query as well as the search results in a graphical way which allows users to discover dependencies between facets and artifacts. In addition, the visualization acts as the primary input to create and change the user query.

In this thesis a two-step approach was chosen to demonstrate the functionality and expressiveness of the framework. First, the framework was described formally by introducing the notion of artifacts and their characteristics as well as the formalized query model of the framework. For that, a representation of SQL statements was chosen to describe the available query options.

Second, the prototypical framework implementation was described to show the feasibility of the query model. Here, the focus lay on the query framework and the respective user interface using the parallel coordinates visualization.

The combination of these two parts offers knowledge workers a toolset to formulate complex queries in an interactive fashion to effectively and efficiently retrieve information in an enterprise setting.

The evaluation of the framework based on the criteria of Tunkelang [2009] showed a general applicability of the framework for search situations which demand a higher user involvement as propagated by the HCIR community. The user interface of the framework was mirrored against general characteristics for search user interfaces [Shneiderman et al., 1997] which were mainly fulfilled.

The LFRP-search framework offers a comprehensive set of functions to improve information access and retrieval. Nevertheless, the conception of the framework only marks the beginning of further areas which have to be addressed. In the following, several promising fields for research are proposed.

- **Assessment of the applicability of the framework in other domains.** The scenarios which were examined in this publication covered the engineering domain. The framework itself was developed generically so that it can be applied to different domains. Thus, other information-oriented domains such as medical imaging can benefit from a search solution such as the LFRP-search framework. To support the diagnosis of illnesses, a search scenario could deal with the analysis of computed tomography images. Doctors would be able to search for similar images from earlier injuries from other patients based on an example image from the current patient. The framework then would support doctors in accessing past indications and diagnoses from former courses of illnesses.
  The applicability in this exemplary domain as well as other should be evaluated further.
- **Collaborative information seeking.** An interesting research field is the collaborative information seeking domain which examines situations where multiple people are occupied with searching information together. Current search systems lack the support for these types of scenarios and users often resort to sending single search results back and forth between the participating users. To prevent this cumbersome exchange of results, the search user interface of the LFRP-framework could be enhanced by groupware functionality tailored to search needs. This could include project spaces where collaborating users

can save search results as well as queries in order to share them with each other as sketched above with the dashboard enhancement in Section 5.3.5 on page 146. This approach would help in situation where users are co-located or distributed. Golovchinsky et al. [2009] introduce a taxonomy of collaboration in online information seeking along four dimension to provide an initial classification of current search systems.

- **Integration of context-based information.** The integration of contextual information about users can help to improve search results. Currently, the LFRP prototype only resorts to using information which is solely provided by the users themselves. Due to the aforementioned *label effect*, users are often not stating all the information which would be necessary to satisfy their information need. Context-based IR can help to include information about the users and their search tasks by automatically augmenting the search query with additional information. Section 2.6.6 on page 51 gave a brief overview of which contextual information can be considered in search scenarios in enterprise related settings, an examination for the field of product development was given in [Eckstein and Henrich, 2008a].
  One downside of such an automatism is the reduction of *transparency* of the search engine in providing results whose determination is more difficult to comprehend for the users. A graduated approach can be the display of these determined factors to the users which then can decide whether they include the additional criteria or not.
- **Integration in application landscape.** An important aspect which was only marginally touched is the integration of the framework in an existing application landscape. Both the integration of the search tool into existing applications to make searching easier for end users as well as providing a standalone search tool which allows users to access the host systems that contain the found search results has to be considered.
- **Performance improvements.** The current realization of the LFRP prototype for the front and the back end mainly aimed for the demonstration of the capabilities of the framework, neglecting in-depth performance considerations. The current implementation relies on a combined approach of an object-oriented database and a full text index (which internally uses an inverted index for the storage and retrieval of textual contents). Dash et al. [2008] show how faceted search can be achieved with good performance using Apache Solr.
  Further efforts should be made in evaluating whether other types of access structures would improve the retrieval performance of the framework.
  Another significant aspect are the used similarity search algorithms which can negatively affect performance. This effect can be reduced by providing users with initial faceted filtering options which reduce the amount of potential results which have to be considered for similarity search.
- **Additional user evaluation of the search interface.** This publication introduced an approach to realize the LFRP-search framework in a user interface

and showed its compliance with formal evaluation criteria. Additionally, a user evaluation is recommended to discover further areas of improvement of the whole framework.

In summary, this publication tried to sensitize the reader that complex search situations as found in organizations can be supported by search user interfaces, but for certain information needs additional user efforts are necessary to specify their information needs. The LFRP-search framework tries to bring users and search user interfaces closer together by using each other's comparative advantage. The search engine can process large amounts of data quickly and present it in a way which provides users with additional information about correlations and dependencies. The users then can make their conclusions about the given results and make decisions on how to refine the search query to satisfy their information needs.

On a broader scale, the chosen approach will help to improve the real-time access of large data sets in organizations providing the ability to get more detailed insights into crucial data. Connecting multiple streams of data help organizations to make better, informed decision leading to a more "intelligent" organization. The LFRP-search framework provides a building block of a comprehensive information access strategy for the enterprise. This thesis should not mark an ending but a starting point for further research to make Bush's vision of the *memex* come true in the enterprise.

Part IV

Appendix

Appendix A

List of Abbreviations

ACL	Access Control List
API	Application Programming Interface
ATC	Air Traffic Control
BI	Business Intelligence
BPEL	Business Process Execution Language
CAD	Computer-Aided Design
CADK	Content Adapter Development Kit
CAS	Content Acquisition System
CBIR	Content-Based Image Retrieval
CLIR	Cross-Language Information Retrieval
CM	Content Management
CRM	Customer Relationship Management
CSS	Cascading Style Sheet
CSV	comma-separated values
DBM	Database Management
DDC	Dewey Decimal Classification
DfX	Design for X
DLS	document level security
DM	Document Management
DWH	Data Warehouse
ECM	Enterprise Content Management
EIS	Executive Information System
ER	Entity Relationship
ERP	Enterprise Resource Planning
ESS	Exploratory Search System
ETL	Extract, Transform, and Load
EVE	Endeca Virtual Engine
GSA	Google Search Appliance
HCI	Human-Computer Interaction
HCIR	Human-Computer Information Retrieval
HFC	Hierarchical Faceted Categories
HITS	Hyperlink-Induced Topic Search

HTML	Hyper Text Markup Language
IA	Information Access
IAP	Information Access Platform
IDE	integrated development environment
IGES	Initial Graphics Exchange Specification
IR	Information Retrieval
ITL	Information Transformation Layer
JAX-RPC	Java API for XML-based RPC
JAX-WS	Java API for XML-Web Services
JDBC	Java Database Connectivity
JMS	Java Message Service
JT	Jupiter Tessellation
HTTP	Hypertext Transfer Protocol
LDAP	Lightweight Directory Access Protocol
LFRP	Multi-Layer Faceted Search with Ranking using Parallel Coordinates
LM	language modeling
MAMI	Monitoring and Management Interface
MDM	Master Data Management
MVC	Model-View-Controller
NLP	Natural Language Processing
NTLM	NT LAN Manager
ODBC	Open Database Connectivity
OEM	Original Equipment Manufacturer
OLAP	Online Analytical Processing
OWA	Ordered Weighted Averaging
OWL	Web Ontology Language
PDF	Portable Document Format
PDM	Product Data Management
PLM	Product Lifecycle Management
PKI	Public Key Infrastructure
POS	Part-of-speech
QbE	Query-by-Example
RDBMS	Relational database management system
RDF	Resource Description Framework
REST	Representational State Transfer
RMI	Remote Method Invocation
RSS	Really Simple Syndication
RSV	Retrieval-Status-Value
SAML	Security Assertion Markup Language
SBA	Search-Based Applications
SCA	Service Component Architecture
SMILA	SeMantic Information Logistics Architecture
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SCM	Supply Chain Management

SPI	Service Provider Interface
SQL	Structured Query Language
SRM	Supplier Relationship Management
SSO	Single Sign-On
STEP	STandard for the Exchange of Product model data
SVM	Support Vector Machine
TREC	Text REtrieval Conference
UDC	Universal Decimal Classification
UIMA	Unstructured Information Management Architecture
UML	Unified Modeling Language
URL	Uniform Resource Locator
VDI	Verein Deutscher Ingenieure
WSDL	Web Services Description Language
XML	Extensible Markup Language
XML-RPC	XML-Remote Procedure Call

Appendix B

LFRP–Search Framework Schema

B.1 LFRP XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.forflow.org"
  xmlns="http://www.forflow.org"
  elementFormDefault="qualified">

  <xs:annotation>
    <xs:documentation xml:lang="en">
      Schema for the LFRP-search framework.
    </xs:documentation>
  </xs:annotation>

  <xs:element name="facetdescription" type="FacetDescriptionType" />

  <xs:complexType name="FacetDescriptionType">
    <xs:sequence>
      <xs:element name="facets" type="FacetsType" />
      <xs:element name="entities" type="EntitiesType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="FacetsType">
    <xs:sequence>
      <xs:element name="facet" type="FacetDefinitionType"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="FacetDefinitionType">
    <xs:sequence>
      <xs:element name="visual" type="VisualType" minOccurs="0"
          maxOccurs="1" />
      <xs:element name="rule" type="CreationRule" minOccurs="0"
          maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
```

```
35      <xs:attribute name="type" use="required" type="FacetType"/>
        <xs:attribute name="multiIndexable" type="xs:boolean" use="required"
            />
37      <xs:attribute name="similarityType" type="SimilarityType"
          use="optional" />
39      <xs:attribute name="documentType" type="DocumentType"
          use="optional" />
41      <xs:attribute name="weightingEnabled" type="xs:boolean" />
    </xs:complexType>
43
    <xs:complexType name="VisualType">
45      <xs:sequence>
          <xs:element name="comparators" type="ComparatorsType" />
47        <xs:element name="visualizationtypes" type="VisualizationTypes" />
        </xs:sequence>
49      <xs:attribute name="name" type="xs:string" use="required" />
        <xs:attribute name="iconPath" type="xs:string" />
51      <xs:attribute name="multiValued" type="xs:boolean" use="required"/>
        <xs:attribute name="multiValueOp">
53        <xs:simpleType>
            <xs:restriction base="xs:string">
55            <xs:enumeration value="AND" />
              <xs:enumeration value="OR" />
57            <xs:enumeration value="AND_OR" />
            </xs:restriction>
59        </xs:simpleType>
        </xs:attribute>
61  </xs:complexType>

63  <xs:complexType name="ComparatorsType">
      <xs:sequence>
65      <xs:element name="comparator" type="ComparatorType"
          maxOccurs="unbounded" />
67    </xs:sequence>
    </xs:complexType>
69
    <xs:complexType name="ComparatorType">
71    <xs:attribute name="default" type="xs:boolean" />
      <xs:attribute name="classname" type="xs:string" use="required"/>
73  </xs:complexType>

75  <xs:complexType name="VisualizationTypes">
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
77      <xs:element name="visualizationtype" type="VisualizationType" />
      </xs:sequence>
79  </xs:complexType>

81  <xs:complexType name="EntitiesType">
      <xs:sequence>
83      <xs:element name="entity" type="entitytype" maxOccurs="unbounded"
          />
      </xs:sequence>
85  </xs:complexType>
```

```
87  <xs:complexType name="entitytype">
      <xs:sequence>
89      <xs:element name="facet" type="FacetRefType" minOccurs="0"
            maxOccurs="unbounded" />
        <xs:element name="relationfacet" type="RelationFacetType"
            minOccurs="0" maxOccurs="unbounded" />
91      <xs:element name="dependency" type="DependencyType" minOccurs="0"/
            >
        <xs:element name="parent-entity">
93        <xs:complexType>
            <xs:attribute name="id" type="xs:string" use="required" />
95        </xs:complexType>
        </xs:element>
97    </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required" />
99    <xs:attribute name="name" type="xs:string" use="required" />
      <!--
101     declares this entity as root element of the hierarchy. Only one
        entity should be root
103   -->
      <xs:attribute name="root" type="xs:boolean" default="false" />
105 </xs:complexType>

107 <xs:complexType name="FacetRefType">
      <xs:attribute name="required" type="xs:boolean" use="required"/>
109   <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>
111
    <xs:complexType name="RelationFacetType" >
113   <xs:complexContent>
        <xs:extension base="FacetRefType">
115         <xs:attribute name="toEntityId" type="xs:string"/>
            <xs:attribute name="toEntityFacetId" type="xs:string"/>
117     </xs:extension>
      </xs:complexContent>
119 </xs:complexType>


121
    <xs:complexType name="DependencyType">
123   <xs:sequence>
        <xs:element name="from">
125       <xs:complexType>
            <xs:attribute name="id" type="xs:string" use="required"/>
127       </xs:complexType>
        </xs:element>
129     <xs:element name="mapping" type="DependencyMappingType"
          maxOccurs="unbounded" />
131   </xs:sequence>
    </xs:complexType>
133
    <xs:complexType name="DependencyMappingType">
135   <xs:sequence>
        <xs:element name="value" type="xs:string" />
137     <xs:element name="to" type="xs:string" />
```

```
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="FacetType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="NOMINAL"></xs:enumeration>
      <xs:enumeration value="ORDINAL"></xs:enumeration>
      <xs:enumeration value="FUNCTION_INTERVAL"></xs:enumeration>
      <xs:enumeration value="DATE"></xs:enumeration>
      <xs:enumeration value="TEXT_QUERY"></xs:enumeration>
      <xs:enumeration value="QUERY_BY_EXAMPLE"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="DocumentType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="TEXT"></xs:enumeration>
      <xs:enumeration value="CAD_MODEL"></xs:enumeration>
      <xs:enumeration value="TECHNICAL_DRAWING"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="SimilarityType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="TEXT"></xs:enumeration>
      <xs:enumeration value="GEOMETRY_2D"></xs:enumeration>
      <xs:enumeration value="GEOMETRY_3D"></xs:enumeration>
      <xs:enumeration value="TOPOLOGY_2D"></xs:enumeration>
      <xs:enumeration value="TOPOLOGY_3D"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="VisualizationType">
    <xs:attribute name="facetType" type="VisualFacetType" />
    <xs:attribute name="default" type="xs:boolean" default="false" />
  </xs:complexType>

  <xs:simpleType name="VisualFacetType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="NOMINAL"></xs:enumeration>
      <xs:enumeration value="ORDINAL"></xs:enumeration>
      <xs:enumeration value="FUNCTION_INTERVAL"></xs:enumeration>
      <xs:enumeration value="DATE"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Listing B.1: XML Schema for the LFRP-search framework.

B.2 Example Schema for the Domain of Product Development

This exemplary schema for the LFRP-search framework describes exemplary facets and an entity hierarchy of artifacts for the domain of product development. Many of these artifacts and facets were identified in the different sub-projects of the joint research project FORFLOW[1].

```
<?xml version="1.0"?>
<facetdescription xmlns="http://www.forflow.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.forflow.org facetSchema.xsd">

  <!-- <facetdescription>-->
  <facets>
    <!-- input similarity facets start -->
    <facet id="3dGeometry" type="QUERY_BY_EXAMPLE" similarityType="
        GEOMETRY_3D"
      documentType="CAD_MODEL" multiIndexable="false" weightingEnabled="
          true">
      <visual name="3D GEO-Similarity" iconPath="images/facets/
          qbe_cad_geometry.png" multiValued="true" multiValueOp="OR">
        <comparators>
          <comparator default="true"
            classname="org.forflow.search.facet.comparator.
                NumericalComparator" />
        </comparators>
        <visualizationtypes>
          <visualizationtype facetType="FUNCTION_INTERVAL"
            default="true" />
        </visualizationtypes>
      </visual>
    </facet>

    <facet id="2dGeometry" type="QUERY_BY_EXAMPLE" similarityType="
        GEOMETRY_2D"
      documentType="TECHNICAL_DRAWING" multiIndexable="false"
      weightingEnabled="true">
      <visual name="2D GEO-Similarity" iconPath="images/facets/
          qbe_drawing_geometry.png" multiValued="true" multiValueOp="OR">
        <comparators>
          <comparator default="true"
            classname="org.forflow.search.facet.comparator.
                NumericalComparator" />
        </comparators>
        <visualizationtypes>
          <visualizationtype facetType="FUNCTION_INTERVAL"
            default="true" />
        </visualizationtypes>
      </visual>
    </facet>
```

[1] For more information refer to Section 1.5 on page 10.

```
37
    <facet id="2dTopology" type="QUERY_BY_EXAMPLE" similarityType="
        TOPOLOGY_2D"
39      documentType="TECHNICAL_DRAWING" multiIndexable="false"
        weightingEnabled="true">
41      <visual name="2D TOP-Similarity" iconPath="images/facets/
            qbe_drawing_topology.png" multiValued="true" multiValueOp="OR">
          <comparators>
43          <comparator default="true"
              classname="org.forflow.search.facet.comparator.
                  NumericalComparator" />
45        </comparators>
          <visualizationtypes>
47          <visualizationtype facetType="FUNCTION_INTERVAL"
              default="true" />
49        </visualizationtypes>
        </visual>
51    </facet>

53    <facet id="text_query" type="TEXT_QUERY" weightingEnabled="true"
        documentType="TEXT" multiIndexable="false">
55      <visual name="Text query" iconPath="images/facets/text_query.png"
            multiValued="true" multiValueOp="OR">
          <comparators>
57          <comparator default="true"
              classname="org.forflow.search.facet.comparator.
                  NumericalComparator" />
59        </comparators>
          <visualizationtypes>
61          <visualizationtype facetType="FUNCTION_INTERVAL"
              default="true" />
63        </visualizationtypes>
        </visual>
65    </facet>

67
      <!-- attribute facet definition -->
69    <facet id="artifactId" type="NOMINAL" multiIndexable="false">
      </facet>
71
      <facet id="artifactType" type="NOMINAL" multiIndexable="false">
73      <visual name="Artifact Type" multiValued="true"
          multiValueOp="AND">
75        <comparators>
            <comparator default="true"
77            classname="org.forflow.sortable.FacetCountComparator" />
            <comparator classname="org.forflow.sortable.AlphaNumComparator
                " />
79        </comparators>
          <visualizationtypes>
81          <visualizationtype facetType="NOMINAL" default="true" />
          </visualizationtypes>
83      </visual>
      </facet>
```

```
85
    <facet id="artifactPath" type="NOMINAL" multiIndexable="false">
87  </facet>

89  <!-- path to an overview page of the artifact contained in the
    system the search framework is embedded in -->
91  <facet id="embeddedSystemPath" type="NOMINAL" multiIndexable="false"
        >
    </facet>
93
    <facet id="artifactHierarchyPath" type="NOMINAL" multiIndexable="
        false">
95  </facet>

97  <facet id="guiDescription" type="NOMINAL" multiIndexable="false">
    </facet>
99
    <!-- document facets -->
101 <facet id="documenttype" type="NOMINAL" multiIndexable="false">
      <visual name="Document Type" multiValued="true" multiValueOp="OR">
103     <comparators>
          <comparator default="true"
105         classname="org.forflow.sortable.FacetCountComparator" />
          <comparator classname="org.forflow.sortable.AlphaNumComparator
              " />
107     </comparators>
        <visualizationtypes>
109       <visualizationtype facetType="NOMINAL" default="true" />
        </visualizationtypes>
111   </visual>
    </facet>
113
    <facet id="file_name" type="NOMINAL" multiIndexable="false">
115 </facet>

117 <facet id="filetype" type="NOMINAL" multiIndexable="false">
      <visual name="File Type" multiValued="true" multiValueOp="OR">
119     <comparators>
          <comparator default="true"
121         classname="org.forflow.sortable.FacetCountComparator" />
          <comparator classname="org.forflow.sortable.AlphaNumComparator
              " />
123     </comparators>
        <visualizationtypes>
125       <visualizationtype facetType="NOMINAL" default="true" />
        </visualizationtypes>
127   </visual>
    </facet>
129
    <facet id="creation_date" type="DATE" multiIndexable="false">
131   <visual name="Creation Date" multiValued="true" multiValueOp="OR">
        <comparators>
133       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
```

```
135           <comparator classname="org.forflow.sortable.DateComparator" />
              </comparators>
137           <visualizationtypes>
                <visualizationtype facetType="DATE"
139               default="true" />
              </visualizationtypes>
141         </visual>
            </facet>
143
            <facet id="creating_phase" type="ORDINAL" multiIndexable="false">
145         <visual name="Creation Phase" multiValued="true" multiValueOp="OR"
                >
              <comparators>
147             <comparator default="true"
                  classname="org.forflow.sortable.FacetCountComparator" />
149             <comparator classname="org.forflow.sortable.AlphaNumComparator
                    " />
              </comparators>
151           <visualizationtypes>
                <visualizationtype facetType="ORDINAL" default="true" />
153           </visualizationtypes>
            </visual>
155         </facet>

157         <facet id="degreeOfConcreteness" type="ORDINAL" multiIndexable="
              false">
            <visual name="Concreteness" multiValued="true" multiValueOp="OR">
159           <comparators>
                <comparator default="true"
161               classname="org.forflow.sortable.FacetCountComparator" />
                <comparator classname="org.forflow.sortable.AlphaNumComparator
                    " />
163           </comparators>
              <visualizationtypes>
165             <visualizationtype facetType="ORDINAL"
                  default="true" />
167           </visualizationtypes>
            </visual>
169         </facet>

171         <facet id="contentDescription" type="ORDINAL" multiIndexable="false"
              >
            <visual name="Content Description" multiValued="true" multiValueOp
                ="OR">
173           <comparators>
                <comparator default="true"
175               classname="org.forflow.sortable.FacetCountComparator" />
                <comparator classname="org.forflow.sortable.AlphaNumComparator
                    " />
177           </comparators>
              <visualizationtypes>
179             <visualizationtype facetType="ORDINAL" default="true" />
              </visualizationtypes>
181         </visual>
```

```
    </facet>
183
    <facet id="purposeOfUse" type="ORDINAL" multiIndexable="false">
185   <visual name="Purpose" multiValued="true" multiValueOp="OR">
        <comparators>
187       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
189       <comparator classname="org.forflow.sortable.AlphaNumComparator
              " />
        </comparators>
191     <visualizationtypes>
          <visualizationtype facetType="ORDINAL" default="true" />
193     </visualizationtypes>
      </visual>
195 </facet>

197 <facet id="degreeOfCrossLinking" type="ORDINAL" multiIndexable="
      false">
    <visual name="Degree Cross-Links" multiValued="true" multiValueOp=
        "OR">
199     <comparators>
          <comparator default="true"
201         classname="org.forflow.sortable.FacetCountComparator" />
          <comparator classname="org.forflow.sortable.AlphaNumComparator
              " />
203     </comparators>
        <visualizationtypes>
205       <visualizationtype facetType="ORDINAL" default="true" />
        </visualizationtypes>
207   </visual>
  </facet>
209
  <facet id="developmentStatus" type="ORDINAL" multiIndexable="false">
211   <visual name="Development Status" multiValued="true" multiValueOp=
        "OR">
        <comparators>
213       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
215       <comparator classname="org.forflow.sortable.AlphaNumComparator
              " />
        </comparators>
217     <visualizationtypes>
          <visualizationtype facetType="ORDINAL"
219         default="true" />
        </visualizationtypes>
221   </visual>
  </facet>
223
  <facet id="document_lifecycle_state" type="ORDINAL" multiIndexable="
      false">
225   <visual name="Document LC" multiValued="true" multiValueOp="OR">
        <comparators>
227       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
```

```
229           <comparator classname="org.forflow.sortable.AlphaNumComparator
                " />
              </comparators>
231           <visualizationtypes>
                <visualizationtype facetType="ORDINAL" default="true" />
233           </visualizationtypes>
            </visual>
235       </facet>

237       <facet id="document_degree_of_maturity" type="FUNCTION_INTERVAL"
            multiIndexable="false">
            <visual name="Document DoM" multiValued="true" multiValueOp="OR">
239           <comparators>
                <comparator classname="org.forflow.sortable.
                    NumericalComparator" />
241           </comparators>
              <visualizationtypes>
243             <visualizationtype facetType="FUNCTION_INTERVAL"
                  default="true" />
245           </visualizationtypes>
            </visual>
247       </facet>

249       <facet id="drawing_number" type="NOMINAL" multiIndexable="false">
            <visual name="Drawing No" multiValued="true" multiValueOp="OR">
251           <comparators>
                <comparator default="true"
253               classname="org.forflow.sortable.FacetCountComparator" />
                <comparator classname="org.forflow.sortable.AlphaNumComparator
                    " />
255           </comparators>
              <visualizationtypes>
257             <visualizationtype facetType="NOMINAL" default="true" />
              </visualizationtypes>
259         </visual>
          </facet>
261
          <!-- person facets -->
263       <facet id="surname" type="NOMINAL" multiIndexable="false">
            <visual name="Surname" multiValued="true" multiValueOp="OR">
265           <comparators>
                <comparator default="true"
267               classname="org.forflow.sortable.FacetCountComparator" />
                <comparator classname="org.forflow.sortable.AlphaNumComparator
                    " />
269           </comparators>
              <visualizationtypes>
271             <visualizationtype facetType="NOMINAL" default="true" />
              </visualizationtypes>
273         </visual>
          </facet>
275
          <facet id="role" type="NOMINAL" multiIndexable="false">
277         <visual name="Role" multiValued="true" multiValueOp="OR">
```

```
        <comparators>
279       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
281       <comparator classname="org.forflow.sortable.AlphaNumComparator
            " />
        </comparators>
283     <visualizationtypes>
          <visualizationtype facetType="NOMINAL" default="true" />
285     </visualizationtypes>
      </visual>
287 </facet>

289 <facet id="department" type="NOMINAL" multiIndexable="false">
      <visual name="Department" multiValued="true" multiValueOp="OR">
291     <comparators>
          <comparator default="true"
293         classname="org.forflow.sortable.FacetCountComparator" />
          <comparator classname="org.forflow.sortable.AlphaNumComparator
            " />
295     </comparators>
        <visualizationtypes>
297       <visualizationtype facetType="NOMINAL" default="true" />
        </visualizationtypes>
299   </visual>
    </facet>
301
    <!-- product facets -->
303 <facet id="product_name" type="NOMINAL" multiIndexable="false">
    </facet>
305
    <facet id="product_group" type="NOMINAL" multiIndexable="false">
307   <visual name="Product Group" multiValued="true" multiValueOp="OR">
        <comparators>
309       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
311       <comparator classname="org.forflow.sortable.AlphaNumComparator
            " />
        </comparators>
313     <visualizationtypes>
          <visualizationtype facetType="NOMINAL" default="true" />
315     </visualizationtypes>
      </visual>
317 </facet>

319 <facet id="product_granularity" type="NOMINAL" multiIndexable="false
      ">
      <visual name="Granularity" multiValued="true" multiValueOp="OR">
321     <comparators>
          <comparator default="true"
323         classname="org.forflow.sortable.FacetCountComparator" />
          <comparator classname="org.forflow.sortable.AlphaNumComparator
            " />
325     </comparators>
        <visualizationtypes>
```

```
327           <visualizationtype facetType="NOMINAL" default="true" />
              </visualizationtypes>
329         </visual>
          </facet>
331
          <facet id="procurement_type" type="NOMINAL" multiIndexable="false">
333         <visual name="Procurement Type" multiValued="true"
              multiValueOp="OR">
335           <comparators>
                <comparator default="true"
337               classname="org.forflow.sortable.FacetCountComparator" />
                <comparator classname="org.forflow.sortable.AlphaNumComparator
                    " />
339           </comparators>
              <visualizationtypes>
341             <visualizationtype facetType="NOMINAL"
                  default="true" />
343           </visualizationtypes>
            </visual>
345       </facet>

347       <facet id="function" type="NOMINAL" multiIndexable="true">
            <visual name="Product Function" multiValued="true"
349           multiValueOp="OR">
              <comparators>
351             <comparator default="true"
                  classname="org.forflow.sortable.FacetCountComparator" />
353             <comparator classname="org.forflow.sortable.AlphaNumComparator
                    " />
              </comparators>
355           <visualizationtypes>
                <visualizationtype facetType="NOMINAL" default="true" />
357           </visualizationtypes>
            </visual>
359       </facet>

361       <facet id="weight" type="FUNCTION_INTERVAL" multiIndexable="false">
            <visual name="Weight" multiValued="true" multiValueOp="OR">
363           <comparators>
                <comparator classname="org.forflow.sortable.
                    NumericalComparator" />
365           </comparators>
              <visualizationtypes>
367             <visualizationtype facetType="FUNCTION_INTERVAL"
                  default="true" />
369           </visualizationtypes>
            </visual>
371       </facet>

373       <facet id="seal_type" type="NOMINAL" multiIndexable="false">
            <visual name="Seal Type" multiValued="true" multiValueOp="OR">
375           <comparators>
                <comparator default="true" classname="org.forflow.sortable.
                    FacetCountComparator" />
```

```
377           <comparator classname="org.forflow.sortable.AlphaNumComparator
                " />
            </comparators>
379         <visualizationtypes>
              <visualizationtype facetType="NOMINAL" default="true" />
381         </visualizationtypes>
          </visual>
383     </facet>

385     <facet id="outsideRadius" type="FUNCTION_INTERVAL" multiIndexable="
            false">
          <visual name="Outer Radius" multiValued="true" multiValueOp="OR">
387         <comparators>
              <comparator classname="org.forflow.sortable.
                  NumericalComparator" />
389         </comparators>
            <visualizationtypes>
391           <visualizationtype facetType="FUNCTION_INTERVAL"
                default="true" />
393         </visualizationtypes>
          </visual>
395     </facet>

397     <facet id="motion_type" type="NOMINAL" multiIndexable="false">
          <visual name="Motion Type" multiValued="true" multiValueOp="OR">
399         <comparators>
              <comparator default="true"
401             classname="org.forflow.sortable.FacetCountComparator" />
              <comparator classname="org.forflow.sortable.AlphaNumComparator
                " />
403         </comparators>
            <visualizationtypes>
405           <visualizationtype facetType="NOMINAL" default="true" />
            </visualizationtypes>
407       </visual>
        </facet>
409
        <facet id="insideRadius" type="FUNCTION_INTERVAL" multiIndexable="
            false">
411       <visual name="Inner Radius" multiValued="true" multiValueOp="OR">
            <comparators>
413           <comparator classname="org.forflow.sortable.
                  NumericalComparator" />
            </comparators>
415         <visualizationtypes>
              <visualizationtype facetType="FUNCTION_INTERVAL"
417             default="true" />
            </visualizationtypes>
419       </visual>
        </facet>
421
        <facet id="thickness" type="FUNCTION_INTERVAL" multiIndexable="false
            ">
423       <visual name="Thickness" multiValued="true" multiValueOp="OR">
```

```
        <comparators>
425       <comparator classname="org.forflow.sortable.
              NumericalComparator" />
        </comparators>
427     <visualizationtypes>
          <visualizationtype facetType="FUNCTION_INTERVAL"
429         default="true" />
        </visualizationtypes>
431   </visual>
    </facet>
433
    <facet id="height" type="FUNCTION_INTERVAL" multiIndexable="false">
435   <visual name="Height" multiValued="true" multiValueOp="OR">
        <comparators>
437       <comparator classname="org.forflow.sortable.
              NumericalComparator" />
        </comparators>
439     <visualizationtypes>
          <visualizationtype facetType="FUNCTION_INTERVAL"
441         default="true" />
        </visualizationtypes>
443   </visual>
    </facet>
445
    <facet id="threading_type" type="NOMINAL" multiIndexable="false">
447   <visual name="Threading Type" multiValued="true" multiValueOp="OR"
        >
        <comparators>
449       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
451       <comparator classname="org.forflow.sortable.AlphaNumComparator
            " />
        </comparators>
453     <visualizationtypes>
          <visualizationtype facetType="NOMINAL" default="true" />
455     </visualizationtypes>
      </visual>
457 </facet>

459 <facet id="nominal_thread_diameter" type="FUNCTION_INTERVAL"
      multiIndexable="false">
    <visual name="nom. Thread Diameter" multiValued="true"
        multiValueOp="OR">
461   <comparators>
        <comparator classname="org.forflow.sortable.
            NumericalComparator" />
463   </comparators>
      <visualizationtypes>
465     <visualizationtype facetType="FUNCTION_INTERVAL"
          default="true" />
467   </visualizationtypes>
    </visual>
469 </facet>
```

```
471     <facet id="screw_type" type="NOMINAL" multiIndexable="false">
          <visual name="Screw Type" multiValued="true" multiValueOp="OR">
473         <comparators>
              <comparator default="true"
475             classname="org.forflow.sortable.FacetCountComparator" />
              <comparator classname="org.forflow.sortable.AlphaNumComparator
                  " />
477         </comparators>
            <visualizationtypes>
479           <visualizationtype facetType="NOMINAL" default="true" />
            </visualizationtypes>
481       </visual>
        </facet>
483
        <facet id="head_shape" type="NOMINAL" multiIndexable="false">
485       <visual name="Head Shape" multiValued="true" multiValueOp="OR">
            <comparators>
487           <comparator default="true"
                classname="org.forflow.sortable.FacetCountComparator" />
489           <comparator classname="org.forflow.sortable.AlphaNumComparator
                  " />
            </comparators>
491         <visualizationtypes>
              <visualizationtype facetType="NOMINAL" default="true" />
493         </visualizationtypes>
          </visual>
495     </facet>

497     <facet id="head_height" type="FUNCTION_INTERVAL" multiIndexable="
            false">
          <visual name="Head Height" multiValued="true" multiValueOp="OR">
499         <comparators>
              <comparator classname="org.forflow.sortable.
                  NumericalComparator" />
501         </comparators>
            <visualizationtypes>
503           <visualizationtype facetType="FUNCTION_INTERVAL"
                default="true" />
505         </visualizationtypes>
          </visual>
507     </facet>

509     <facet id="length_threaded_end" type="FUNCTION_INTERVAL"
            multiIndexable="false">
          <visual name="Threaded End Length" multiValued="true" multiValueOp
              ="OR">
511         <comparators>
              <comparator classname="org.forflow.sortable.
                  NumericalComparator" />
513         </comparators>
            <visualizationtypes>
515           <visualizationtype facetType="FUNCTION_INTERVAL"
                default="true" />
517         </visualizationtypes>
```

```
      </visual>
519   </facet>

521   <facet id="head_diameter" type="FUNCTION_INTERVAL" multiIndexable="
        false">
        <visual name="Head Diameter" multiValued="true" multiValueOp="OR">
523       <comparators>
            <comparator classname="org.forflow.sortable.
                NumericalComparator" />
525       </comparators>
          <visualizationtypes>
527         <visualizationtype facetType="FUNCTION_INTERVAL"
              default="true" />
529       </visualizationtypes>
        </visual>
531   </facet>

533   <facet id="knurl_type" type="NOMINAL" multiIndexable="false">
        <visual name="Knurl Type" multiValued="true" multiValueOp="OR">
535       <comparators>
            <comparator default="true"
537           classname="org.forflow.sortable.FacetCountComparator" />
            <comparator classname="org.forflow.sortable.AlphaNumComparator
                " />
539       </comparators>
          <visualizationtypes>
541         <visualizationtype facetType="NOMINAL" default="true" />
          </visualizationtypes>
543     </visual>
      </facet>
545
      <facet id="rolling_element_type" type="NOMINAL" multiIndexable="
        false">
547     <visual name="Rolling Element" multiValued="true" multiValueOp="OR
          ">
        <comparators>
549       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
551       <comparator classname="org.forflow.sortable.AlphaNumComparator
            " />
        </comparators>
553     <visualizationtypes>
          <visualizationtype facetType="NOMINAL" default="true" />
555     </visualizationtypes>
      </visual>
557   </facet>

559   <!-- project facets -->
      <facet id="project_name" type="NOMINAL" multiIndexable="false">
561     <visual name="Project Name" multiValued="true" multiValueOp="OR">
        <comparators>
563       <comparator default="true"
            classname="org.forflow.sortable.FacetCountComparator" />
565       <comparator classname="org.forflow.sortable.AlphaNumComparator
```

```
            " />
        </comparators>
        <visualizationtypes>
          <visualizationtype facetType="NOMINAL" default="true" />
        </visualizationtypes>
      </visual>
    </facet>

    <facet id="start_date" type="DATE" multiIndexable="false">
      <visual name="Start Date" multiValued="true" multiValueOp="OR">
        <comparators>
          <comparator classname="org.forflow.sortable.DateComparator" />
        </comparators>
        <visualizationtypes>
          <visualizationtype facetType="DATE" default="true" />
        </visualizationtypes>
      </visual>
    </facet>

    <facet id="finish_date" type="DATE" multiIndexable="false">
      <visual name="Finish Date" multiValued="true" multiValueOp="OR">
        <comparators>
          <comparator classname="org.forflow.sortable.DateComparator" />
        </comparators>
        <visualizationtypes>
          <visualizationtype facetType="DATE" default="true" />
        </visualizationtypes>
      </visual>
    </facet>

    <facet id="project_degree_of_maturity" type="FUNCTION_INTERVAL"
        multiIndexable="false">
      <visual name="Project DoM" multiValued="true" multiValueOp="OR">
        <comparators>
          <comparator classname="org.forflow.sortable.
              NumericalComparator" />
        </comparators>
        <visualizationtypes>
          <visualizationtype facetType="FUNCTION_INTERVAL" default="true
              " />
        </visualizationtypes>
      </visual>
    </facet>

    <facet id="project_description" type="NOMINAL" multiIndexable="true"
        >
    </facet>

    <!-- material facets -->
    <facet id="material_name" type="NOMINAL" multiIndexable="false">
    </facet>

    <facet id="material_group" type="NOMINAL" multiIndexable="false">
      <visual name="Material Group" multiValued="true" multiValueOp="OR"
```

```
        >
615       <comparators>
            <comparator default="true"
617           classname="org.forflow.sortable.FacetCountComparator" />
            <comparator classname="org.forflow.sortable.AlphaNumComparator
                " />
619       </comparators>
          <visualizationtypes>
621         <visualizationtype facetType="NOMINAL" default="true" />
          </visualizationtypes>
623     </visual>
      </facet>
625
      <facet id="density" type="FUNCTION_INTERVAL" multiIndexable="false">
627     <visual name="Density" multiValued="true" multiValueOp="OR">
          <comparators>
629         <comparator classname="org.forflow.sortable.
                NumericalComparator" />
          </comparators>
631       <visualizationtypes>
            <visualizationtype facetType="FUNCTION_INTERVAL" default="true
                " />
633       </visualizationtypes>
        </visual>
635   </facet>

637   <facet id="eModulus" type="FUNCTION_INTERVAL" multiIndexable="false"
        >
      <visual name="E Modulus" multiValued="true" multiValueOp="OR">
639     <comparators>
          <comparator classname="org.forflow.sortable.
              NumericalComparator" />
641     </comparators>
        <visualizationtypes>
643       <visualizationtype facetType="FUNCTION_INTERVAL" default="true
              " />
        </visualizationtypes>
645   </visual>
    </facet>
647
    <!-- facets which represent relations -->
649
    <!-- author facet for documents -->
651 <facet id="author" type="NOMINAL" multiIndexable="false">
      <visual name="Author" multiValued="true" multiValueOp="OR">
653     <comparators>
          <comparator default="true"
655         classname="org.forflow.sortable.FacetCountComparator" />
          <comparator classname="org.forflow.sortable.AlphaNumComparator
              " />
657     </comparators>
        <visualizationtypes>
659       <visualizationtype facetType="NOMINAL" default="true" />
        </visualizationtypes>
```

```
661       </visual>
        </facet>
663
        <!-- Works in facet for persons -->
665     <facet id="works_in_project" type="NOMINAL" multiIndexable="false">
          <visual name="Works in" multiValued="true" multiValueOp="OR">
667         <comparators>
              <comparator default="true"
669             classname="org.forflow.sortable.FacetCountComparator" />
              <comparator classname="org.forflow.sortable.AlphaNumComparator
                  " />
671         </comparators>
            <visualizationtypes>
673           <visualizationtype facetType="NOMINAL" default="true" />
            </visualizationtypes>
675       </visual>
        </facet>
677
        <!-- Used in for materials pointing to products -->
679     <facet id="is_used_in_product_group" type="NOMINAL" multiIndexable="
            false">
          <visual name="Used in Product Group" multiValued="true"
              multiValueOp="OR">
681         <comparators>
              <comparator default="true"
683             classname="org.forflow.sortable.FacetCountComparator" />
              <comparator classname="org.forflow.sortable.AlphaNumComparator
                  " />
685         </comparators>
            <visualizationtypes>
687           <visualizationtype facetType="NOMINAL" default="true" />
            </visualizationtypes>
689       </visual>
        </facet>
691
        <!--  Is made of (material) for products -->
693     <facet id="is_made_of_material_group" type="NOMINAL" multiIndexable=
            "false">
          <visual name="Is made of Material Group" multiValued="true"
              multiValueOp="OR">
695         <comparators>
              <comparator default="true"
697             classname="org.forflow.sortable.FacetCountComparator" />
              <comparator classname="org.forflow.sortable.AlphaNumComparator
                  " />
699         </comparators>
            <visualizationtypes>
701           <visualizationtype facetType="NOMINAL" default="true" />
            </visualizationtypes>
703       </visual>
        </facet>
705   </facets>

707   <!-- entity definitions -->
```

```
  <entities>
709     <entity id="artifact" name="Artifact" root="true">
          <facet required="true" id="artifactId" />
711       <facet required="true" id="artifactPath" />
          <facet required="true" id="artifactType" />
713       <facet required="false" id="embeddedSystemPath" />
          <facet required="false" id="artifactHierarchyPath" />
715       <facet required="false" id="guiDescription" />

717       <dependency>
            <from id="artifactType" />
719         <mapping>
              <value>PRODUCT</value>
721           <to>product</to>
            </mapping>
723         <mapping>
              <value>DOCUMENT</value>
725           <to>document</to>
            </mapping>
727         <mapping>
              <value>PERSON</value>
729           <to>person</to>
            </mapping>
731         <mapping>
              <value>MATERIAL</value>
733           <to>material</to>
            </mapping>
735         <mapping>
              <value>PROJECT</value>
737           <to>project</to>
            </mapping>
739       </dependency>
          <parent-entity id="artifact" />
741     </entity>

743     <entity id="person" name="Person">
          <facet required="true" id="role" />
745       <facet required="false" id="surname" />

747       <dependency>
            <from id="role" />
749         <mapping>
              <value>Employee</value>
751           <to>employee</to>
            </mapping>
753       </dependency>
          <parent-entity id="artifact" />
755     </entity>

757     <entity id="employee" name="Employee">
          <facet required="false" id="department" />
759
          <!-- relations to other layers -->
761       <relationfacet required="false" id="works_in_project" toEntityId="
```

```
        project" toEntityFacetId="project_name"/>

763     <parent-entity id="person" />
    </entity>
765
    <entity id="project" name="Project">
767     <facet required="false" id="project_name" />
        <facet required="false" id="start_date" />
769     <facet required="false" id="finish_date" />
        <facet required="false" id="project_degree_of_maturity" />
771     <facet required="false" id="project_description" />

773     <parent-entity id="artifact" />
    </entity>
775
    <entity id="material" name="Material">
777     <facet required="false" id="material_name" />
        <facet required="false" id="material_group" />
779     <facet required="false" id="density" />
        <facet required="false" id="eModulus" />
781
        <!-- relations to other layers -->
783     <relationfacet required="false"  id="is_used_in_product_group"
            toEntityId="product" toEntityFacetId="product_group"/>

785     <parent-entity id="artifact" />
    </entity>
787
    <entity id="document" name="Document">
789     <facet required="true" id="documenttype" />
        <facet required="false" id="file_name" />
791     <facet required="false" id="filetype" />
        <facet required="false" id="creation_date" />
793     <facet required="false" id="creating_phase" />
        <facet required="false" id="degreeOfConcreteness" />
795     <facet required="false" id="contentDescription" />
        <facet required="false" id="purposeOfUse" />
797     <facet required="false" id="degreeOfCrossLinking" />
        <facet required="false" id="developmentStatus" />
799     <facet required="false" id="document_lifecycle_state" />
        <facet required="false" id="document_degree_of_maturity" />
801
        <!-- relations to other layers -->
803     <relationfacet required="false" id="author" toEntityId="person"
            toEntityFacetId="surname"/>

805     <dependency>
          <from id="documenttype" />
807       <mapping>
            <value>TECHNICAL_DRAWING</value>
809         <to>drawing</to>
          </mapping>
811     </dependency>
        <parent-entity id="artifact" />
```

```
813     </entity>

815     <entity id="drawing" name="Technical Drawing">
          <facet required="false" id="drawing_number" />
817       <parent-entity id="document" />
        </entity>
819
        <entity id="product" name="Product">
821       <facet required="true" id="product_group" />
          <facet required="true" id="procurement_type" />
823       <facet required="false" id="product_name" />
          <facet required="false" id="product_granularity" />
825       <facet required="false" id="function" />
          <facet required="false" id="weight" />
827
          <!-- relations to other layers -->
829       <relationfacet required="false"  id="is_made_of_material_group"
              toEntityId="material" toEntityFacetId="material_name"/>

831       <dependency>
            <from id="product_group" />
833         <mapping>
              <value>Seal</value>
835           <to>seal</to>
            </mapping>
837         <mapping>
              <value>Bearing</value>
839           <to>bearing</to>
            </mapping>
841         <mapping>
              <value>Screw</value>
843           <to>screw</to>
            </mapping>
845       </dependency>

847       <parent-entity id="artifact" />
        </entity>
849
        </entity>
851
        <entity id="seal" name="Seal">
853       <facet required="false" id="motion_type" />
          <facet required="false" id="outsideRadius" />
855       <facet required="true" id="seal_type" />

857       <dependency>
            <from id="seal_type" />
859         <mapping>
              <value>O-Ring</value>
861           <to>oRing</to>
            </mapping>
863         <mapping>
              <value>Disk</value>
865           <to>disk</to>
```

```
        </mapping>
        <mapping>
          <value>Floating Ring Seal</value>
          <to>floatingRingSeal</to>
        </mapping>
      </dependency>
      <parent-entity id="product" />
    </entity>

    <entity id="oRing" name="O-Ring">
      <facet required="false" id="insideRadius" />
      <facet required="false" id="thickness" />

      <parent-entity id="seal" />
    </entity>

    <entity id="disk" name="Disk">
      <facet required="false" id="insideRadius" />
      <facet required="false" id="height" />

      <parent-entity id="seal" />
    </entity>

    <entity id="floatingRingSeal" name="Floating Ring Seal" >
      <parent-entity id="seal" />
    </entity>

    <entity id="bearing" name="Bearing">
      <facet required="true" id="rolling_element_type" />

      <dependency>
        <from id="rolling_element_type" />
        <mapping>
          <value>Needle Bearing</value>
          <to>needleBearing</to>
        </mapping>
        <mapping>
          <value>Cylinder Roller Bearing</value>
          <to>cylinderRollerBearing</to>
        </mapping>
        <mapping>
          <value>Ball Bearing</value>
          <to>ballBearing</to>
        </mapping>
      </dependency>

      <parent-entity id="product" />
    </entity>

    <entity id="needleBearing" name="Needle Bearing" >
      <parent-entity id="bearing" />
    </entity>

    <entity id="cylinderRollerBearing" name="Cylinder Roller Bearing" >
```

```
    <parent-entity id="bearing" />
  </entity>

  <entity id="ballBearing" name="Ball Bearing" >
    <parent-entity id="bearing" />
  </entity>

  <entity id="screw" name="Screw">
    <facet required="false" id="threading_type" />
    <facet required="false" id="nominal_thread_diameter" />
    <facet required="true" id="screw_type" />

    <dependency>
      <from id="screw_type" />
      <mapping>
        <value>Headscrew</value>
        <to>headScrew</to>
      </mapping>
      <mapping>
        <value>Stud Screw</value>
        <to>studScrew</to>
      </mapping>
    </dependency>

    <parent-entity id="product" />
  </entity>

  <entity id="headScrew" name="Head Screw">
    <facet required="true" id="head_shape" />
    <facet required="false" id="head_height" />

    <dependency>
      <from id="head_shape" />
      <mapping>
        <value>Hexagon Head Screw</value>
        <to>hexagonHeadScrew</to>
      </mapping>
      <mapping>
        <value>Cylinder Head Bolt</value>
        <to>cylinderHeadBolt</to>
      </mapping>
      <mapping>
        <value>Knurled Thumb Screw</value>
        <to>knurledThumbScrew</to>
      </mapping>
    </dependency>

    <parent-entity id="screw" />
  </entity>

  <entity id="studScrew" name="Stud Screw">
    <facet required="false" id="length_threaded_end" />
    <parent-entity id="screw" />
  </entity>
```

```
975     <entity id="hexagonHeadScrew" name="Hexagon Head Screw">
          <parent-entity id="headScrew" />
977     </entity>

979     <entity id="cylinderHeadBolt" name="Cylinder Head Bolt">
          <facet required="false" id="head_diameter" />
981       <parent-entity id="headScrew" />
        </entity>
983
        <entity id="knurledThumbScrew" name="Knurled Thumb Screw">
985       <facet required="false" id="head_diameter" />
          <facet required="false" id="knurl_type" />
987       <parent-entity id="headScrew" />
        </entity>
989
      </entities>
991
    </facetdescription>
```

Listing B.2: Example schema for the LFRP-search framework applied to the domain of product development.

Bibliography

Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From relations to semistructured data and XML*. Kaufmann, San Francisco, Calif., 2001.

Man Abrol, Neil Latarche, Uma Mahadevan, Jianchang Mao, Rajat Mukherjee, Prabhakar Raghavan, Michel Tourn, John Wang, and Grace Zhang. Navigating large-scale semi-structured data in business portals. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 663–666, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *Proceedings of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA*, pages 5–16. IEEE Computer Society, 2002.

Saeema Ahmed. *Understanding the use and reuse of experience in engineering design*. PhD thesis, University of Cambridge, Cambridge, 2001.

Saeema Ahmed, Ken M. Wallace, and Luciënne Blessing. Understanding the differences between how novice and experienced designers approach design tasks. *Research in Engineering Design*, 14(1):1–11, 2003.

Maryam Alavi and Dorothy E. Leidner. Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly*, 26 (2):107–136, 2001.

Thomas J. Allen. Distinguishing engineers from scientists. In Ralph Katz, editor, *Managing professionals in innovative organizations*, pages 3–18. Harper Business, New York, 1988.

Thomas J. Allen. *Managing the flow of technology: Technology transfer and the dissemination of technological information within the R&D organization*. MIT Press, Cambridge, Mass., 7. print. edition, 1995.

Omar Alonso, Ricardo Baeza-Yates, and Michael Gertz. Exploratory Search Using Timelines. In *SIGCHI 2007 Workshop on Exploratory Search and HCI Workshop*, 2007.

Whit Andrews. Magic Quadrant for Information Access Technology, 2009. URL `http://www.gartner.com/technology/media-products/reprints/vivisimo/169927.html`.

Whit Andrews. MarketScope for Enterprise Search, 2010.

Peter Anick, J. D. Brennan, Rex A. Flynn, David R. Hanssen, B. Alvey, and Jonathan M. Robbins. A direct manipulation interface for boolean information retrieval via natural language query. In *SIGIR '90: Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 135–150, New York, NY, USA, 1990. ACM.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, Harlow, 2008.

Peter Bailey, David Hawking, and Brett Matson. Secure search in enterprise webs: tradeoffs in efficient implementation for document level security. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 493–502, New York, NY, USA, 2006. ACM Press.

Peter Bailey, Nick Craswell, Arjen P. de Vries, and Ian Soboroff. Overview of the TREC 2007 Enterprise Track. In *Proceedings of the 16th Text Retrieval Conference (TREC 2007)*, Gaithersburg, USA, 2007.

Krisztian Balog. *People Search in the Enterprise*. PhD thesis, University of Amsterdam, 2008.

Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In Susan Dumas, editor, *SIGIR Seattle 2006*, Special issue of the SIGIR forum, pages 43–50, New York, NY, 2006. Association for Computing Machinery.

Krisztian Balog, Ian Soboroff, Paul Thomas, Peter Bailey, Nick Craswell, and Arjen P. de Vries. Overview of the TREC 2008 Enterprise Track. In *Proceedings of the 17th Text Retrieval Conference (TREC 2008)*, Gaithersburg, USA, 2008.

Krisztian Balog, Arjen P. de Vries, Pavel Serdyukov, Paul Thomas, and Thijs Westerveld. Overview of the TREC 2009 Entity Track. In *Proceedings of the 18th Text Retrieval Conference (TREC 2009)*, Gaithersburg, USA, 2009.

Ramon C. Barquin and Herb A. Edelstein. *Building, using, and managing the data warehouse*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

Madeleine Bates. Models of natural language understanding. *Proceedings of the National Academy of Sciences of the United States of America*, 92(22):9977–9982, 1995.

Marcia J. Bates. The Design of Browsing and Berrypicking Techniques for the Online Search Interface. *Online Review*, 13(5):407–424, 1989.

Marcia J. Bates. What is browsing-really? A model drawing from behavioural science research. *Information Research*, 12(4):paper 330, 2007.

Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys (CSUR)*, 18(4): 323–364, 1986.

Matthias Beck and Burkhard Freitag. Weighted Boolean conditions for ranking. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering Workshop*, pages 568–571, Washington, DC, USA, 2008. IEEE Computer Society.

Wolfgang Beitz, Klaus Ehrlenspiel, Jens Erb, Alexander Fink, Jürgen Gausemeier, Kerstin Geiger, Hans Grabowski, Dirk Helbig, Johannes Klose, Lothar Krause, Udo Lindemann, Werner Mattes, Robert Mertens, Robert Stößer, and Gunther Storz. *New ways for product development (Neue Wege zur Produktentwicklung)*. Raabe Fachverlag für Wissenschaftsinformation, 1997.

Nicholas J. Belkin. Interaction with texts: Information retrieval as information-seeking behavior. In *Information Retrieval '93. Von der Modellierung zur Anwendung.*, pages 55–66. Universitätsverlag Konstanz, 1993.

Nicholas J. Belkin. (Somewhat) Grand Challenges for Information Retrieval. *SIGIR Forum*, 42(1):47–54, 2008.

Nicholas J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management*, 31 (3):431–448, 1995.

Nicholas J. Belkin, Michael Cole, and Jingjing Liu. A Model for Evaluation of Interactive Information Retrieval. In Shlomo Geva, Jaap Kamps, Carol Peters, Tetsuya Sakai, Andrew Trotman, and Ellen Voorhees, editors, *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*, pages 7–8, Amsterdam, The Netherlands, The Netherlands, 2009. IR Publications.

Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In *WSDM '08: Proceedings of the International Conference on Web Search and Web Data Mining*, pages 33–44, New York, NY, USA, 2008. ACM.

Pia Borlund. The concept of relevance in IR. *Journal of the American Society for Information Science and Technology*, 54:913–925, 2003a.

Pia Borlund. The IIR evaluation model: a framework for evaluation of interactive information retrieval systems. *Information Research*, 8(3), 2003b.

Karl Bosch. *Grundzüge der Statistik: Einführung mit Übungen*. Oldenbourg, München/Wien, München, 2., erg. aufl. edition, 1999.

Martin Braschler, Norman Briner, Hans Fischer, Markus Häni, Thomas Mandl, Peter Schäuble, Markus Steinbach, and Jürg Stuker. Enterprise-Search-Systeme im internen Wissensmanagement: Ergebnisse einer Studie zu Perspektiven der Unternehmen. *Datenbank-Spektrum*, 9(30), 2009.

Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998.

Andrei Broder and Arthur Ciccolo. Towards the next generation of enterprise search technology. *IBM Systems Journal*, 43:451–454, 2004.

Vannevar Bush. As We May Think. *Atlantic Monthly*, (176):101–108, 1945.

Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan V. Vranic. Feature-based similarity search in 3D object databases. *ACM Comput. Surv.*, 37(4):345–387, 2005.

Katriina Byström and Kalervo Järvelin. Task complexity affects information seeking and use. *Information Processing & Management*, 31(2):191–213, 1995.

Jamie Callan. Distributed Information Retrieval. In *Advances in Information Retrieval. The Information Retrieval Series.*, volume 7, pages 127–150. Kluwer Academic Publishers, 2000.

Christopher S. Campbell, Paul P. Maglio, Alex Cozzi, and Byron Dom. Expertise identification using email communications. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 528–531, New York, NY, USA, 2003. ACM.

Robert Capra, Gary Marchionini, Jung Sun Oh, Fred Stutzman, and Yan Zhang. Effects of structure and interaction style on distinct search tasks. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 442–451, New York, NY, USA, 2007. ACM.

Robert G. Capra and Gary Marchionini. The relation browser tool for faceted exploratory search. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 420–420, New York, NY, USA, 2008. ACM.

Peter H. Carstensen. Towards Information Exploration Support for Engineering Designers. *Advances in Concurrent Engineering*, pages 26–33, 1997.

Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.

Eugene Charniak. Statistical Techniques for Natural Language Parsing. *AI Magazine*, 18(4): 33–43, 1997.

Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, 1997.

Hsinchun Chen, Andrea Houston, Robin R. Sewell, and Bruce R. Schatz. Internet Browsing and Searching: User Evaluations of Category Map and Concept Space Techniques. *JASIS*, 49(7):582–603, 1998.

Edward C. Clarkson, Shamkant B. Navathe, and James D. Foley. Generalized formal models for faceted user interfaces. In *JCDL '09: Proceedings of the 2009 joint international conference on Digital libraries*, pages 125–134, New York, NY, USA, 2009. ACM.

Cyril W. Cleverdon and Michael Keen. Aslib Cranfield research project - Factors determining the performance of indexing systems; Volume 2, Test results, 1966. URL `http://hdl.handle.net/1826/863`.

Cyril W. Cleverdon, Jack Mills, and Michael Keen. Aslib Cranfield research project - Factors determining the performance of indexing systems; Volume 1, Design; Part 1, Text, 1966a. URL `http://hdl.handle.net/1826/861`.

Cyril W. Cleverdon, Jack Mills, and Michael Keen. Aslib Cranfield research project - Factors determining the performance of indexing systems; Volume 1, Design; Part 2, Appendices, 1966b. URL `http://hdl.handle.net/1826/862`.

Edgar F. Codd, S. B. Codd, and C. T. Salley. *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*. Codd & Date, Inc, San Jose, 1993.

Thomas H. Cormen. *Introduction to algorithms*. MIT Press [u.a.], Cambridge, Mass., 2. ed., 8. printing. edition, 2007.

Andrew W. Court, Stephen J. Culley, and Christopher A. McMahon. Information Access Diagrams: A Technique for Analyzing the Usage of Design Information. *Journal of Engineering Design*, 7(1):55–75, 1996.

Nick Craswell, David Hawking, Anne-Marie Vercoustre, and Peter Wilkins. P@NOPTIC Expert: Searching for Experts Not Just For Documents. In *AusWeb*, pages 21–25, 2001.

Nick Craswell, Arjen P. de Vries, and Ian Soboroff. Overview of the TREC 2005 Enterprise Track. In *Proceedings of the 14th Text Retrieval Conference (TREC 2005)*, Gaithersburg, USA, 2005a.

Nick Craswell, Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Relevance weighting for query independent evidence. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 416–423, New York, NY, USA, 2005b. ACM.

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329, New York, NY, USA, 1992. ACM.

Wisam Dakka, Panagiotis G. Ipeirotis, and Kenneth R. Wood. Automatic construction of multifaceted browsing interfaces. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 768–775, New York, NY, USA, 2005. ACM.

Wisam Dakka, Rishabh Dayal, and Panagiotis G. Ipeirotis. Automatic Discovery of Useful Facet Terms. In *ACM SIGIR Workshop on Faceted Search*, 2006.

Raymond D'Amore. Expertise community detection. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 498–499, New York, NY, USA, 2004. ACM.

Debabrata Dash, Jun Rao, Nimrod Megiddo, Anastasia Ailamaki, and Guy Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 3–12, New York, NY, USA, 2008. ACM.

Francisco del Rey-Chamorro and Ken M. Wallace. A Study of Information Retrieval in the Aerospace Domain. In A. Folkeson, K. Gralen, M. Norell, and U. Sellgren, editors, *Proceedings of the 14th International Conference on Engineering Design (ICED '03)*, pages 271–272, 2003.

Francisco del Rey-Chamorro and Ken M. Wallace. Understanding the search for information in the aerospace domain. In A. Samuel and Lewis W., editors, *Proceedings of the International Conference on Engineering Design, ICED'05*, pages 81–82, 2005.

Melvil Dewey. *A Classification and Subject Index for Cataloguing and Arranging the Books and Pamphlets of a Library: Dewey Decimal Classification*. Kingsport Press, Inc., Kingsport, Tennessee, 1876.

Byron Dom, Iris Eiron, Alex Cozzi, and Yi Zhang. Graph-based ranking algorithms for e-mail expertise analysis. In *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 42–48, New York, NY, USA, 2003. ACM.

Daryl D'Souza, James Thom, and Justin Zobel. A Comparison of Techniques for Selecting Text Collections. In *ADC '00: Proceedings of the Australasian Database Conference*, page 28, Washington, DC, USA, 2000. IEEE Computer Society.

D. Dubois and Henri Prade. A review of fuzzy set aggregation connectives. *Information Sciences*, 36(1-2):85–121, 1985.

Raiko Eckstein and Andreas Henrich. An integrated context model for the product development domain and its implications on design reuse. In D. Marjanovic, editor, *10th International design conference*, pages 761–768, 2008a.

Raiko Eckstein and Andreas Henrich. Reaching the Boundaries of Context-Aware IR: Accepting Help from the User. In *Proceedings of the 2nd Int. Workshop on Adaptive Information Retrieval (AIR 2008)*, London, 2008b. BCS.

Raiko Eckstein and Andreas Henrich. Visual Browsing in Product Development Processes. In The Design Society, editor, *Proceedings of the 17th International Conference on Engineering Design*, pages 183–194, 2009.

Raiko Eckstein, Andreas Henrich, and Nadine Weber. Das LFRP-Framework zur Beherrschung komplexer Suchsituationen. In FG-IR, editor, *LWA 2009*, Darmstadt, 2009. TU Darmstadt.

Robert M. Edsall. The parallel coordinate plot in action: design and use for geographic visualization. *Computational Statistics & Data Analysis*, 43:605–619, 2003.

Richard Edwards. Technology Audit Google Search Appliance 6.0, 2009.

Endeca. Indexing Enterprise Content: White Paper, 2007. URL `http://www.endeca.com/resource-center-white-papers.htm`.

Endeca. Endeca Information Access Platform: Whitepaper, 2008.

Endeca. Search Applications on the Endeca Information Access Platform: White Paper, 2009a. URL `http://www.endeca.com/resource-center-white-papers.htm`.

Endeca. Endeca IAP 6 Performance and Scale: Technical Data Sheet, 2009b. URL `http://www.endeca.com/products-information-access-platform-mdex-engine.htm`.

Endeca. The Discovery for Manufacturing Suite: White Paper, 2009c. URL http://www.endeca.com/resource-center-white-papers.htm.

Daniel Ericson, Jimmy Johansson, and Matthew Cooper. Visual data analysis using tracked statistical measures within parallel coordinate representations. In *Proceedings of the Third International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 42–53, 2005.

Boris Evelson and Matthew Brown. Search + BI = Unified Information Access: Combining Unstructured And Structured Info Delivers Business Insight, 2008.

Exalead. Exalead CloudView Platform Highlights: White Paper, 2009a. URL http://www.exalead.com/software/forms/download.php?resourceid=22.

Exalead. Exalead CloudView Security: White Paper, 2009b. URL http://www.exalead.com/software/forms/download.php?resourceid=15.

Exalead. Exalead CloudView Architecture: White Paper, 2009c. URL http://www.exalead.com/software/forms/download.php?resourceid=2.

Exalead. Exalead CloudView Search edition Connectors & Formats: White Paper, 2009d. URL http://www.exalead.com/software/common/pdfs/products/cloudview/Exalead-Connectors-and-Formats.pdf.

Ronald Fagin, Ravi Kumar, Kevin S. McCurley, Jasmine Novak, D. Sivakumar, John A. Tomlin, and David P. Williamson. Searching the workplace web. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 366–375, New York, NY, USA, 2003. ACM.

Mohamed Farah and Daniel Vanderpooten. A Multiple Criteria Approach for Information Retrieval. In Fabio Crestani, Paolo Ferragina, and Mark Sanderson, editors, *String Processing and Information Retrieval, 13th International Conference SPIRE 2006, Glasgow, UK, October 11-13, 2006*, pages 242–254. Springer Verlag, 2006.

Mohamed Farah and Daniel Vanderpooten. An outranking approach for rank aggregation in information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 591–598, New York, NY, USA, 2007. ACM.

Susan Feldman. The high cost of not finding information. *KM World*, 13(3), 2004.

Susan Feldman and Chris Sherman. The High Cost of Not Finding Information: Technical Report #29127, 2003.

Christiane Fellbaum. *WordNet: An electronic lexical database*. Language, speech, and communication. MIT Press, Cambridge, Mass., 2. printing. edition, 1999.

Adam J. Ferrari, David Gourley, Keith Johnson, Frederick C. Knabe, Daniel Tunkelang, and John S. Walter. Hierarchical data-driven navigation system and method for information retrieval, 2006a.

Adam J. Ferrari, David J. Gourley, Keith A. Johnson, Frederick C. Knabe, Vinay B. Mohta, Daniel Tunkelang, and John S. Walter. Hierarchical Data-Driven Search and Navigation System and Method for Information Retrieval, 2006b.

Adam J. Ferrari, David Gourley, Keith Johnson, Frederick C. Knabe, Daniel Tunkelang, and John S. Walter. Hierarchical Data-Driven Navigation System and Method for Information Retrieval, 2007.

Raya Fidel and Maurice Green. The many faces of accessibility: engineers' perception of information sources. *Information Processing & Management*, 40(3):563–581, 2004.

Ken Fishkin and Maureen C. Stone. Enhanced dynamic queries via movable filters. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–420, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

University of California Flamenco Search Interface Project. Nobel Prize Winners (Flamenco), 2007. URL `http://orange.sims.berkeley.edu/cgi-bin/flamenco.cgi/nobel/Flamenco`.

Manuel J. Fonseca and Joaquim A. Jorge. Towards content-based retrieval of technical drawings through high-dimensional indexing. *Computers & Graphics*, 27(1):61–69, 2003.

Luanne Silvia Freund and Elaine G. Toms. Enterprise search behaviour of software engineers. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 645–646, New York, NY, USA, 2006. ACM.

Luanne Silvia Freund, Elaine G. Toms, and Charles L. A. Clarke. Modeling task-genre relationships for IR in the workplace. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 441–448, New York, NY, USA, 2005a. ACM Press.

Luanne Silvia Freund, Elaine G. Toms, and Julie Waterhouse. Modeling the Information Behaviour of Software Engineers Using a Work-Task Framework. In Andrew In Grove, editor, *Proceedings 68th Annual Meeting of the American Society for Information Science and Technology (ASIST) 42*, Charlotte (US), 2005b.

Hermann Friedrich. Herausforderungen im Umfeld Enterprise Search. In Jörg Eberspächer and Stefan Holtel, editors, *Suchen und Finden im Internet*, Springer-11775 /Dig. Serial], pages 181–186. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2007.

Eugene Garfield. *Citation indexing - its theory and application in science, technology and humanities*. Information sciences series. Wiley, New York, 1979.

William Goffman. On Relevance As A Measure. *Information Storage and Retrieval*, 2(3): 201–204, 1964.

Gene Golovchinsky, Jeremy Pickens, and Maribeth Back. A Taxonomy of Collaboration in Online Information Seeking. *CoRR*, abs/0908.0704, 2009.

Google Inc. Google Search Appliance 6 Datasheet, 2009a. URL `http://www.google.com/enterprise/pdf/gsa_datasheet.pdf`.

Google Inc. Google Search Appliance Product Models, 2009b. URL `http://www.google.com/enterprise/pdf/gsa_product_models.pdf`.

Martin Graham and Jessie Kennedy. Using Curves to Enhance Parallel Coordinate Visualisations. In *7th. International Conference on Information Visualization (IV'03)*, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

Sharon L. Greene, Steven J. Devlin, Philip E. Cannata, and Louis M. Gomez. No IFs, ANDs, or ORs: a study of databases querying. *Int. J. Man-Mach. Stud.*, 32(3):303–326, 1990.

Uri Hanani, Bracha Shapira, and Peretz Shoval. Information Filtering: Overview of Issues, Research and Systems. *User Modeling and User-Adapted Interaction*, 11(3):203–259, 2001.

Craig Harris, Alisdair Owens, Alistair Russell, and Daniel Alexander Smith. mSpace: Exploring The Semantic Web. A Technical Report in Support of the mSpace software framework, 2004. URL `http://eprints.ecs.soton.ac.uk/10359/`.

Yusef Hassan-Montero and Víctor Herrero-Solana. Improving Tag-Clouds as Visual Information Retrieval Interfaces. In Vincente P. Guerrero-Bote, editor, *Current research in information sciences and technologies*, 2006.

Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular Brushing of Extended Parallel Coordinates. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 127, Washington, DC, USA, 2002. IEEE Computer Society.

David Hawking. Challenges in enterprise search. In *ADC '04: Proceedings of the 15th Australasian database conference*, pages 15–24, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

David Hawking, Cécile Paris, Ross Wilkinson, and MingFang Wu. Context in Enterprise Search and Delivery. In Birger Larsen, editor, *Proceedings of the ACM SIGIR 2005 Workshop on Information Retrieval in Context (IRiX)*, pages 14–16, Salvador, Brazil, 2005.

Marti A. Hearst. TileBars: visualization of term distribution information in full text information access. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 59–66, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

Marti A. Hearst. Next Generation Web Search: Setting Our Sites. *IEEE Data Engineering Bulletin, Special issue on Next Generation Web Search, . Luis Gravano (Ed.)*, 23(3):38–48, 2000.

Marti A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, 2006a.

Marti A. Hearst. Design Recommendations for Hierarchical Faceted Search Interfaces. In *ACM SIGIR Workshop on Faceted Search*, 2006b.

Marti A. Hearst. *Search User Interfaces*. Cambridge Univ. Press, Cambridge, 2009.

Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, New York, NY, USA, 1996. ACM.

Marti A. Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Finding the flow in web site search. *Communications of the ACM*, 45(9):42–49, 2002.

Morten Hertzum and Annelise Mark Pejtersen. The information-seeking practices of engineers: searching for documents as well as for people. *Information Processing & Management*, 36(5):761–778, 2000.

Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. Efficient IR-style keyword search over relational databases. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 850–861. VLDB Endowment, 2003.

Peter Ingwersen. *Information Retrieval Interaction*. Taylor Graham, 1992.

Peter Ingwersen. Polyrepresentation of information needs and semantic entities: elements of a cognitive theory for information retrieval interaction. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 101–110, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

Peter Ingwersen and Kalervo Järvelin. *The Turn - Integration of Information Seeking and Retrieval in Context*. Springer, 2005.

William H. Inmon. *Building the data warehouse*. Wiley computer publishing. Wiley, New York, N.Y., 3. ed. edition, 2002.

Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, Volume 1, Number 4:69–91, 1985.

Alfred Inselberg. Visualization of concept formation and learning. *Kybernetes*, 34 Issue 1/2: 151–166, 2005.

Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In Arie E. Kaufman, editor, *Visualization '90*, pages 361–378, Los Alamitos, Calif., 1990. IEEE Computer Soc. Press.

Kalervo Järvelin and Peter Ingwersen. Information seeking research needs extension towards tasks and technology. *Information Research*, 10(1), 2004.

Brian Johnson and Ben Shneiderman. Tree-Maps: a space-filling approach to the visualization of hierarchical information structures. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

Steve Jones. Graphical query specification and dynamic result previews for a digital library. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 143–151, New York, NY, USA, 1998. ACM.

Mika Käki. Findex: search result categories help users when document ranking fails. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 131–140, New York, NY, USA, 2005. ACM.

Diane Kelly and Jaime Teevan. Implicit Feedback for Inferring User Preference: A Bibliography. *SIGIR Forum*, 37(2), 2003.

Werner Kießling and Gerhard Köstler. Preference SQL: design, implementation, experiences. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 990–1001. VLDB Endowment, 2002.

Won Kim and Jungyun Seo. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *Computer*, 24(12):12–18, 1991.

Ralph Kimball and Joe Caserta. *The data warehouse ETL toolkit: Practical techniques for extracting, cleaning, conforming, and delivering data*. Timely, practical, reliable. Wiley, Indianapolis, Ind., 2004.

Ralph Kimball and Margy Ross. *The data warehouse toolkit: The complete guide to dimensional modeling*. Wiley, New York, 2nd ed. edition, 2002.

Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5): 604–632, 1999.

Jonathan Koren, Yi Zhang, and Xue Liu. Personalized interactive faceted search. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 477–486, New York, NY, USA, 2008. ACM.

Bill Kules and Ben Shneiderman. Users can change their web search tactics: Design guidelines for categorized overviews. *Information Processing & Management*, 44(2):463–484, 2008.

Bill Kules, Robert Capra, Matthew Banta, and Tito Sierra. What do exploratory searchers look at in a faceted search interface? In *JCDL '09: Proceedings of the 2009 joint international conference on Digital libraries*, pages 313–322, New York, NY, USA, 2009. ACM.

Gloria J. Leckie, Karen E. Pettigrew, and Christian Sylvain. Modeling the Information Seeking of Professionals: A General Model Derived from Research on Engineers, Health Care Professionals, and Lawyers. *The Library Quarterly*, 66(2):161–193, 1996.

Joon Ho Lee. Properties of extended Boolean models in information retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 182–190, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

Joon Ho Lee. Analyses of multiple evidence combination. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–276, New York, NY, USA, 1997. ACM.

Christoph Marian Leszinski. *Ein Visualisierungs- und Navigationsassistent für Produktstrukturen in der Produktentwicklung*, volume 20006 of *Schriftenreihe / Ruhruniversität Bochum, Institut für Konstruktionstechnik*. Shaker, Aachen, 2001.

Chengkai Li, Kevin Chen-Chuan Chang, Ihab F. Ilyas, and Sumin Song. RankSQL: Query Algebra and Optimization for Relational Top-k Queries. In Fatma Özcan, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 131–142. ACM, 2005.

Hans Peter Litz. *Statistische Methoden in den Wirtschafts- und Sozialwissenschaften*. Oldenbourg, München/Wien, München, 3., vollst. überarb. und erw. aufl. edition, 2003.

Fang Liu, Clement Yu, Weiyi Meng, and Abdur Chowdhury. Effective keyword search in relational databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2006. ACM.

Craig Macdonald and Iadh Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396, New York, NY, USA, 2006. ACM.

Fritz Machlup and Una Mansfield. *The study of information: interdisciplinary messages*. Wiley, Chichester, 1984.

Wendy E. MacKay. Is paper safer? The role of paper flight strips in air traffic control. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 6(4):311–340, 1999.

Thomas M. Mann. *Visualization of search results from the World Wide Web*. PhD thesis, Universität Konstanz, Universitätsstr. 10, 78457 Konstanz, 2001. URL `http://kops.ub.uni-konstanz.de/volltexte/2002/751`.

Gary Marchionini. *Information Seeking in Electronic Environments*. Cambridge Univ. Press, NY, Cambridge, 1995.

Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49 (4):41–46, 2006a.

Gary Marchionini. Toward Human-Computer Information Retrieval. *Bulletin of the American Society for Information Science and Technology*, (June/July), 2006b.

Gary Marchionini and Ben Brunk. Toward a General Relation Browser: A GUI for Information Architects. *Journal of Digital Information*, 4(1), 2003.

Gary Marchionini and Ben Shneiderman. Finding Facts vs. Browsing Knowledge in Hypertext Systems. *Computer*, 21(1):70–80, 1988.

Gary Marchionini and Ryen White. Find What You Need, Understand What You Find. *International Journal of Human-Computer Interaction*, 23(3):205–237, 2007.

J. R. Marsh. *The capture and utilisation of design experience in engineering design*. PhD thesis, University of Cambridge, Cambridge, 1997.

David R. McGee, Philip R. Cohen, R. Matthews Wesson, and Sheilah Horman. Comparing paper and tangible, multimodal tools. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 407–414, New York, NY, USA, 2002. ACM.

Michael J. McGuffin and m. c. schraefel. A comparison of hyperstructures: zzstructures, mSpaces, and polyarchies. In *HYPERTEXT '04: Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, pages 153–162, New York, NY, USA, 2004. ACM.

Alistair McLean, Anne-Marie Vercoustre, and MingFang Wu. Enterprise PeopleFinder: Combining Evidence from Web Pages and Corporate Data. In David Hawking, editor, *Proceedings of the 8th Australasian Document Computing Conference (ADCS'03)*, Canberra, Australia, 2003.

Markus Mechnich. A User Interface for Faceted Search Based on Parallel Coordinates in Flash: Project work, 2008a.

Markus Mechnich. Design of a Backend Architecture for Faceted Search with Parallel Coordinates on Large Datasets: Project work, 2008b.

Harald Meerkamm, Andreas Henrich, Stefan Jablonski, Helmut Krcmar, Udo Lindemann, and Frank Rieg, editors. *Flexible Process Support in Product Development (Flexible Prozessunterstützung in der Produktentwicklung): Processes - Data - Navigation (Prozesse - Daten - Navigation); FORFLOW ; Final Report 01.10.2006 - 30.09.2009*. Shaker, Aachen, 2009. ISBN 9783832286408.

Marina Meila and David Heckerman. An Experimental Comparison of Model-Based Clustering Methods. *Machine Learning*, 42(1):9–29, 2001.

Weiyi Meng, Clement Yu, and King-Lup Liu. Building efficient and effective metasearch engines. *ACM Comput. Surv.*, 34(1):48–89, 2002.

A. Michard. Graphical presentation of boolean expressions in a database query language: design notes and an ergonomic evaluation. *Behaviour & Information Technology*, 1(3): 279–288, 1982.

Tom Michael Mitchell. *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill, Boston, Mass., 2008.

Calvin N. Mooers. Mooers' Law or Why Some Retrieval Systems Are Used And Others Are Not. *The Scientist*, 11(2):1–10, 1959.

Rajat Mukherjee and Jianchang Mao. Enterprise Search: Tough Stuff. *Queue*, 2(2):36–46, 2004.

Henning Müller, Nicolas Michoux, David Bandon, and Antoine Geissbuhler. A review of content-based image retrieval systems in medical applications–clinical benefits and future directions. *International Journal of Medical Informatics*, 73(1):1–23, 2004.

Meinard Müller. *Information retrieval for music and motion*. Springer-Verlag Berlin Heidelberg, 1st edition, 2007.

Wolfgang Müller, Markus Zech, Andreas Henrich, and Daniel Blank. VisualFlamenco: Dependable, Interactive Image Browsing Based on Visual Properties. In IEEE, editor, *Proceedings of 6th International Workshop on Content-Based Multimedia Indexing*, pages 568–575, London, UK, 2008.

Jack Muramatsu and Wanda Pratt. Transparent Queries: investigation users' mental models of search engines. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 217–224, New York, NY, USA, 2001. ACM.

Aaron Newman and Jeremy Thomas. *Enterprise 2.0 implementation: [incorporate cutting-edge Web 2.0 services within enterprise networks ; create internal social networks, blogs, wikis, and mashups ; disseminate corporate information via RSS/Atom feeds ; manage Enterprise 2.0 risk, security, compliance, backups, and disaster recovery]*. Network professional's library. McGraw-Hill, New York, 2009.

Ikujiro Nonaka and Hirotaka Takeuchi. *The knowledge creating company: How Japanese companies create the dynamics of innovation*. Oxford Univ. Press, New York, 1995.

Ragnar Nordlie. "User revealment"—a comparison of initial queries and ensuing question development in online searching and in human reference interactions. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, New York, NY, USA, 1999. ACM.

Henrik Nottelmann and Norbert Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 290–297, New York, NY, USA, 2003. ACM.

Leslie Owens. The Forrester Wave: Enterprise Search, Q2 2008: for Information & Knowledge Management Professionals, 2008.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web, 1998. URL `citeseer.ist.psu.edu/page98pagerank.html`.

Gerhard Pahl, Wolfgang Beitz, Jörg Feldhusen, Karl-Heinrich Grote, Ken M. Wallace, and Luciënne T Blessing. *Engineering design: A systematic approach*. Springer, London, 3. ed. edition, 2007.

Bo Pang and Lillian Lee. *Opinion mining and sentiment analysis*. Now Publishers, Hanover, MA, 2008.

Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object Exchange Across Heterogeneous Information Sources. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 251–260, Washington, DC, USA, 1995. IEEE Computer Society.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 1, New York, NY, USA, 2006. ACM.

Annelise Mark Pejtersen, Diane H. Sonnenwald, Jacob Buur, T. Govindaraj, and Kim Vicente. The Design Explorer Project: Using a Cognitive Framework to Support Knowledge Exploration. *Journal of Engineering Design*, 8(3):289–301, 1997.

Thomas E. Pinelli, Ann P. Bishop, Rebecca O. Barclay, and John M. Kennedy. The information-seeking behavior of engineers. In Allen Kent, editor, *Encyclopedia of Library and Information Science: Vol. 52*, pages 167–201. Dekker, New York, 1993.

Heinz-Jürgen Pinnekamp and Frank Siegmann. *Deskriptive Statistik: Mit einer Einführung in das Programm SPSS*. Oldenbourg, München/Wien, München, 5., vollst. überarb. und aktualisierte aufl. edition, 2008.

Peter Pirolli and Stuart K. Card. Information Foraging. *Psychological Review*, 106(4):643–675, 1999.

Catherine Plaisant, Gary Marchionini, Tom Bruns, Anita Komlodi, and Laura Campbell. Bringing treasures to the surface: iterative design for the Library of Congress National Digital Library Program. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 518–525, New York, NY, USA, 1997. ACM.

Catherine Plaisant, Ben Shneiderman, Khoa Doan, and Tom Bruns. Interface and data architecture for query preview in networked information systems. *ACM Trans. Inf. Syst.*, 17 (3):320–341, 1999.

Wanda Pratt, Marti A. Hearst, and Lawrence M. Fagan. A knowledge-based approach to organizing retrieved documents. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 80–85, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

Marc Quantrill. Oh no, not more Web 2.0 jargon: effective enterprise search. *Managing Information*, 15(8):56–58, 2008.

Sheizaf Rafaeli. Interactivity: From new media to communication. In J. M. Wiemann, S. Pingree, and Robert P. Hawkins, editors, *Sage Annual Review of Communication Research: Advancing Communication Science: Merging Mass and Interpersonal Processes*, volume 16 of *Sage annual reviews of communication research*, pages 110–134. Beverly Hills: Sage; Sage, Newbury Park, 1988.

Shiyali Ramamrita Ranganathan. *Colon Classification*. Madras Library Association, Madras, India, 1933.

Shiyali Ramamrita Ranganathan. *Classification, Coding, and Machinery for Search*. UNESCO, Paris, France, 1950.

Romaric Redon, Andreas Larsson, Richard Leblond, and Barthélémy Longueville. VIVACE Context Based Search Platform. In Boicho N. Kokinov, Daniel C. Richardson, Thomas Roth-Berghofer, and Laure Vieu, editors, *Modeling and Using Context, 6th International and Interdisciplinary Conference, CONTEXT 2007, Roskilde, Denmark, August 20-24, 2007, Proceedings*, volume 4635 of *Lecture Notes in Computer Science*, pages 397–410, Berlin / Heidelberg, 2007. Springer.

Harald Reiterer, Gabriela Tullius, and Thomas M. Mann. INSYDER: a content-based visual-information-seeking system for the Web. *International Journal on Digital Libraries*, 5(1): 25–41, 2005.

Joan M. Reitz. *Dictionary for library and information science*. Libraries Unlimited, Westport, Conn., 2004.

George Robertson, Kim Cameron, Mary Czerwinski, and Daniel Robbins. Polyarchy visualization: visualizing multiple intersecting hierarchies. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 423–430, New York, NY, USA, 2002. ACM.

Stephen E. Robertson. The Probability Ranking Principle in IR. *Jounal Of Documentation*, 33:294–304, 1977.

Kenneth A. Ross and Angel Janevski. Querying Faceted Databases. In Christoph Bussler, Val Tannen, and Irini Fundulaki, editors, *Semantic Web and Databases, Second International Workshop, SWDB 2004, Toronto, Canada, August 29-30, 2004, Revised Selected Papers*, volume 3372, pages 199–218. Springer, 2005.

Ian Ruthven and Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145, 2003.

Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science (JASIS)*, 41(4):288–297, 1990.

Gerard Salton, Anita Wong, and Chung-shu Yang. A vector space model for automatic index. *Communications of the ACM*, 18(11):613–620, 1975.

Gerard Salton, Edward A. Fox, and Harry Wu. Extended Boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213, New York, NY, USA, 1999. ACM.

Tefko Saracevic. Relevance: A Review of the Literature and a Framework for Thinking on the Notion in Information Science. Part II: Nature and Manifestations of Relevance*. *Journal of the American Society for Information Science and Technology*, 58(13):1915–1933, 2007.

Prabir Sarkar and Amaresh Chakrabarti. Understanding Search in Design. In Jean-Claude Bocquet, editor, *Proceedings of the International Conference on Engineering Design, ICED'07*, Paris, France, 2007.

Linda Schamber, Michael Eisenberg, and Michael S. Nilan. A re-examination of relevance: toward a dynamic, situational definition. *Information Processing & Management*, 26(6): 755–776, 1990.

Markus Schichtel. *Product data modelling in practice (Produktdatenmodellierung in der Praxis)*. Hanser, München u.a., 1st edition, 2002.

m. c. schraefel, Maria Karam, and Shengdong Zhao. Listen to the Music: Audio Preview Cues for Exploration of Online Music. In *Interact 2003 - Bringing the Bits Together*, 2003.

m. c. schraefel, Max L. Wilson, and Maria Karam. Preview Cues: Enhancing Access to Multimedia Content, 2004. URL `http://eprints.ecs.soton.ac.uk/9253/`.

m. c. schraefel, Max L. Wilson, Alistair Russell, and Daniel A. Smith. mSpace: Improving Information Access to Multimedia Domains with MultiModal Exploratory Search. *Commun. ACM*, 49(4):47–49, 2006.

Joseph A. Shaw and Edward A. Fox. Combination of Multiple Searches. In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 243–252. NIST, Maryland, 1993.

Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236, 1990.

Ben Shneiderman. Dynamic Queries for Visual Information Seeking. *IEEE Softw.*, 11(6): 70–77, 1994.

Ben Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proc. of the 1996 IEEE Symp. on Visual Languages*, pages 336–343, 1996.

Ben Shneiderman and Catherine Plaisant. *Designing the user interface: Strategies for effective human-computer interaction*. Pearson/Addison-Wesley, Boston, 4th edition, 2005.

Ben Shneiderman, Don Byrd, and Bruce Croft. Clarifying Search: A User-Interface Framework for Text Searches. *D-Lib Magazine*, January 1997, 1997.

Luo Si and Jamie Callan. A semisupervised learning method to merge search engine results. *ACM Trans. Inf. Syst.*, 21(4):457–491, 2003.

Harri Siirtola. Direct manipulation of parallel coordinates. In *4th. International Conference on Information Visualization (IV'00)*, pages 373–378, 2000.

Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

Vineet Sinha and David R. Karger. Magnet: supporting navigation in semistructured data environments. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 97–106, New York, NY, USA, 2005. ACM.

Ian Soboroff, Arjen P. de Vries, and Nick Craswell. Overview of the TREC 2006 Enterprise Track. In *Proceedings of the 15th Text Retrieval Conference (TREC 2006)*, Gaithersburg, USA, 2006.

Stefano Spaccapietra, Christine Parent, and Yann Dupont. Model independent assertions for integration of heterogeneous schemas. *The VLDB Journal*, 1(1):81–126, 1992.

Karen Sparck Jones, Stephen G. Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing & Management*, 36(6):779–808, 2000.

Robert Spence and Lisa Tweedie. The Attribute Explorer: information synthesis via exploration. *Interacting with Computers*, 11(2):137–146, 1998.

Amanda Spink, T. D. Wilson, Nigel Ford, Allen Foster, and David Ellis. Information seeking and mediated searching study. Part 3. Successive searching. *Journal of the American Society for Information Science and Technology*, 53(9):716–727, 2002.

Anselm Spoerri. InfoCrystal: a visual tool for information retrieval & management. In *CIKM '93: Proceedings of the second international conference on Information and knowledge management*, pages 11–20, New York, NY, USA, 1993. ACM.

Dirk Stelzer. Informationsbedarf. In Peter Mertens and Andrea Back, editors, *Lexikon der Wirtschaftsinformatik*, pages 238–239. Springer, Berlin, 2001.

Stanley Smith Stevens. On the Theory of Scales of Measurement. *Science*, 103(2684):677–680, 1946.

Emilia Stoica and Marti A. Hearst. Nearly-Automated Metadata Hierarchy Creation. In *The Companion Proceedings of HLT-NAACL '04, Boston*, pages 117–120, 2004.

Emilia Stoica, Marti A. Hearst, and Megan Richardson. Automating Creation of Hierarchical Faceted Metadata Structures. In *Proceedings of NAACL-HLT, Rochester NY*, 2007.

Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O'Neil, Patrick E. O'Neil, Alex Rasin, Nga Tran, and Stanley B. Zdonik. C-Store: A Column-oriented DBMS. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 553–564. ACM, 2005.

Alistair G. Sutcliffe, Mark Ennis, and J Hu. Evaluating the effectiveness of visual user interfaces for information retrieval. *International Journal of Human-Computer Studies*, 53(5): 741–763, 2000.

Arlene G. Taylor. *Introduction to cataloging and classification*. Library and information science text series. Libraries Unlimited, Englewood, Colorado, 8th ed. edition, 1992.

Robert S. Taylor. The process of asking questions. *American Documentation*, 13(4):391–396, 1962.

Melanie Tory, Simeon Potts, and Torsten Moller. A Parallel Coordinates Style Interface for Exploratory Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):71–80, 2005.

Edward Rolf Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, Conn., 1983.

Edward Rolf Tufte. *Envisioning information*. Graphics Press, Cheshire, Conn., 1990.

Daniel Tunkelang. Reconsidering Relevance and Embracing Interaction. *Bulletin of the American Society for Information Science and Technology*, 36(1):20–23, 2009.

Howard Turtle. Natural language vs. Boolean query evaluation: a comparison of retrieval performance. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 212–220, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

Anthony Unwin, Martin Theus, and Heike Hofmann, editors. *Graphics of Large Datasets - Visualizing a Million*. Springer Science+Business Media, LLC, 2006.

Pertti Vakkari. Task complexity, problem structure and information actions: Integrating studies on information seeking and retrieval. *Information Processing & Management*, 35 (6):819–837, 1999.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. Improving Accuracy in NLP Through Combination of Machine Learning Systems. *Computational Linguistics*, 27(2): 199–229, 2001.

VDI. Design engineering methodics; setting up and use of design catalogues (Konstruktionsmethodik - Methodisches Entwickeln von Lösungsprinzipien VDI 2222 Blatt 2), 1982.

VDI. Systematic approach to the development and design of technical systems and products (Methodik zum Entwicklen und Konstruieren technischer Systeme und Produkte VDI 2221), 1993.

VDI. Methodic development of solution principles (Konstruktionsmethodik - Methodisches Entwickeln von Lösungsprinzipien VDI 2222 Blatt 1), 1997.

Brian C. Vickery. *Faceted classification: A guide to construction and use of special schemes*, volume 5. Aslib, London, 1960.

Brian C. Vickery. *Faceted classification schemes*, volume 5 of *Rutgers series on systems for the intellectual organization of information*. Graduate School of Library Science Rutgers, New Brunswick, NJ, 1966.

Friedrich Vogel. *Formeln, Definitionen, Erläuterungen, Stichwörter und Tabellen*, volume 2 of *Beschreibende und schließende Statistik / von Friedrich Vogel*. Oldenbourg, München/Wien, München, 12., vollst. überarb. und erw. aufl. edition, 2000.

Peter von der Lippe. *Deskriptive Statistik: 25 Übersichten*. Fischer, Stuttgart, 1993.

Ken M. Wallace and Saeema Ahmed. How Engineering Designers Obtain Information. In Udo Lindemann, editor, *Human behaviour in design*, Engineering online library, pages 184–194. Springer, Berlin, 2003.

Nadine Weber and Andreas Henrich. Retrieval of technical drawings in DXF format - concepts and problems. In Alexander Hinneburg, editor, *LWA 2007*, Halle, 2007. Universität Halle.

Nadine Weber, Raiko Eckstein, and Andreas Henrich. Searching Multiple Artifacts: A Comprehensive Framework for Complex Search Situations. In *Proceedings of Eighth International Conference FQAS 2009 - Flexible Query Answering Systems, Roskilde, Denmark*, 2009.

Edward J. Wegman. Hyperdimensional Data Analysis Using Parallel Coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

David White. Enterprise Search: Discover the Next Opportunity for Growth, 2009. URL `http://www.aberdeen.com/launch/report/benchmark/6098-RA-enterprise-search-unstructured-data.asp`.

Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query–Response Paradigm*. Morgan & Claypool Publishers, 2008.

Rolf Wigand, Arnold Picot, and Ralf Reichwald. *Information, organization and management: Expanding markets and corporate boundaries*. Wiley series in information systems. Wiley, Chichester, 1st edition, 1997.

Max L. Wilson and m. c. schraefel. mSpace: What do Numbers and Totals Mean in a Flexible Semantic Browser. In *The 3rd International Semantic Web User Interaction Workshop at ISWC2006*, 2006.

Max L. Wilson and m. c. schraefel. A longitudinal study of exploratory and keyword search. In *Proceedings of the 2008 Joint Conference on Digital Libraries*, pages 52–56, New York, NY, 2008. ACM.

Max L. Wilson, Paul André, and m. c. schraefel. Backward highlighting: enhancing faceted search. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 235–238, New York, NY, USA, 2008. ACM.

Kent Wittenburg, Tom Lanning, Michael Heinrichs, and Michael Stanton. Parallel bargrams for consumer-based information exploration and choice. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, volume Vol. 3, Iss. 2 of *CHI letters*, pages 51–60, New York, NY, 2001. ACM Press.

Allison Woodruff, Andrew Faulring, Ruth Rosenholtz, Julie Morrsion, and Peter Pirolli. Using thumbnails to search the Web. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 198–205, New York, NY, USA, 2001. ACM.

Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.

Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti A. Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM Press.

Dawit Yimam and Alfred Kobsa. DEMOIR: A Hybrid Architecture for Expertise Modeling and Recommender Systems. In *Proceedings of the 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000. (WET ICE 2000)*., pages 67–74, 2000.

Degi Young and Ben Shneiderman. A graphical filter/flow representation of Boolean queries: a prototype implementation and evaluation. *J. Am. Soc. Inf. Sci.*, 44(6):327–339, 1993.

Oren Zamir and Oren Etzioni. Grouper: A Dynamic Clustering Interface to Web Search Results. *Computer Networks*, 31(11-16):1361–1374, 1999.

Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. Learning to cluster web search results. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA, 2004. ACM.

Pavel Zezula, Giuseppe Amato, Michal Batko, and Vlastislav Dohnal. *Similarity Search The Metric Space Approach*. Springer Science+Business Media Inc, Boston, MA, 2006.

Junliang Zhang and Gary Marchionini. Evaluation and evolution of a browse and search interface: Relation Browser++. In *dg.o 2005: Proceedings of the 2005 national conference on Digital government research*, pages 179–188. Digital Government Society of North America, 2005.

Lorri Zipperer. The Creative Professional and Knowledge. *Special Libraries*, 84(2):69–78, 1993.

Fritz Zwicky. *Discovery, Invention, Research Through the Morphological Approach*. Macmillan, 1969.

UNIVERSITY OF BAMBERG PRESS

In recent years information produced in organizations steadily increased. In order to stay competitive, companies have a growing interest in reusing existing knowledge from past projects. In contrast, the complexity of information in modern organizations often exceeds the capabilities of the currently deployed enterprise search solutions. The publication outlines difficulties and challenges for search engines in an enterprise setting and proposes the LFRP-framework that supports exploratory searches in an organizational setting which is built on top of four pillars.

1. The multi-layer functionality allows users to formulate complex search queries referring to more than one result type.

2. The search paradigm of faceted searching supports users in formulating search queries incrementally.

3. By combining the concept of faceted search with the capability to influence the search result order based on filter criteria, users can define weights for criteria values in a fine-grained way.

4. The visualization type of parallel coordinates is used for visual query formulation as well as for a representation of the search results enabling users to discover relationships between search results.

The framework enables users to access large linked data sets by navigation and constitutes a contribution to a comprehensive information strategy for organizations.

ISBN 978-3-86309-017-3
ISSN 1867-7401
21,70 Euro