Università di Pisa

Dipartimento di Ingegneria dell'Informazione
Dottorato di Ricerca in Ingegneria dell'Informazione
ING-INF/05

Ph.D. Thesis

Virtual Digital Libraries

Leonardo Candela

Supervisor
Dr. Donatella Castelli

Supervisor
Prof. Giuseppe Alia

2003

Abstract

Digital libraries represent the meeting places for knowledge providers and knowledge consumers supporting and enhancing the process through which knowledge is created, used, and discovered. The demand for digital libraries is worldwide strong. These complex systems can offer a richer set of functionality than that initially expected and are able to transform the way in which joint research is conducted, thus responding to the requirements of a great number of research communities. In fact, nowadays research is a collaborative and multidisciplinary effort conducted by virtual research organisations whose components are spread worldwide. Despite this large and innovative demand the current digital library development models remain unchanged and so they are not able to match the emerging requirements. In this dissertation we propose a novel approach based on *virtual digital libraries*, i. e. digital libraries built by dynamically aggregating and appropriately presenting the pool of shared resources needed to fulfil the requirements of digital library communities. To support such an approach we introduced (*i*) a *reference model* for understanding significant relationships among the components of digital libraries and for developing consistent services that support them and (*ii*) a set of approaches and services able to provide *virtual views* over the heterogeneous information space resulting from reusing shared information sources. In particular, three approaches to information space virtualization are presented that provide profitable usage of the shared resources: *information objects virtualization*, *collections virtualization*, and *distributed semantic search*.

Acknowledgments

First and foremost, I wish to thank my supervisor, Dr. Donatella Castelli, who allowed me to undertake my Ph.D. research at the Institute of Information Science and Technologies (ISTI) of the Italian National Research Council (CNR). Donatella's constant encouragement, inspiration, and support led me to believe in the importance of this work.

I'm indebted with my supervisor Prof. Giuseppe Alia for the advices and the support he provided me during this work.

Thanks to the members of the Networked Multimedia Information Systems Laboratory of the ISTI - CNR, I benefited from useful discussions and friendly collaborations with each of them during the day-by-day work. Special thanks go to Pasquale Pagano, Umberto Straccia and Maria Bruna Baldacci for the stimulating discussions we had on various and heterogeneous research topics ranging from the information retrieval to description logics and library sciences. Each of them enriched my working life and knowledge with their personal experiences.

I also profited from the various IST projects I participated in during these years. These experiences gave me the opportunity to meet many research groups and thus to grow as a researcher by facing with concrete problems. The financial support from the IST projects CYCLADES (IST-2000-25456), Open Archives Forum (IST-2001-320015), DELOS (G038-507618), and DILIGENT (IST-2003-004260) is gratefully acknowledged. Moreover, I'm specially indebted with Prof. Heiko Schuldt and Prof. Hans-Jörg Schek which provided me the opportunity and the support during the period I spent as guest researcher in the Information and Software Engineering Group at the University for Health Sciences, Medical Informatics and Technology (UMIT) in the context of the DELOS Network of Excellence for Digital Libraries Research Exchange Program.

Final thanks go to Ermelinda and Lorenzo, my parents, and to Mariarosaria and Maria. They represent my family and supported me with love and patience during these years. This achievement belongs also to them.

to Maria

to my parents

"The future belongs to neither the conduit or content players, but to those who control filtering, searching, and sense-making tools we will rely on to navigate through the expanses of cyberspace."
Saffo, Paul. "It's the Context, Stupid.". 1994

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The understanding and expectations of digital libraries (DLs) [Arm01, FAFL95, FM98] have evolved considerably since the nineties, when the first digital library systems were built. The initial DLs were mainly intended as digital entities analogous to the physical libraries [DL96, ABB+99, SAG+01]. They were providing mechanisms to maintain collections of documents, and to search through these collections by exploiting the metadata records associated with the documents, presenting the results in a suitable format to meet the needs of the specific DL audience. Much effort from experts in the field was required to prepare the digitised content and to develop the software that implements the DL functionality. Dedicated computers, sometimes quite powerful, as in the case of DLs for audio/visual resources, had to be acquired in order to store and process the documents. These dedicated resources had to be sufficient to support the highest peak of activities, even if these were executed only rarely, e. g. at start-up of the DL or periodically for preservation purposes. As a consequence, DLs were only created to serve large research communities or important institutions since these were the only ones that could afford the cost of such products.

After approximately ten years of study and development it has now become clear that DL systems can actually offer much richer functionality than initially expected and that, if they become more widely employed, have the potentiality to transform the way in which joint research is conducted. Digital information objects are more versatile than physical documents. They offer the possibility of creating many multitype object formats by combining multimedia components in an unlimited variety of ways. A DL can thus, for example, manage information objects that mix texts, scientific data and satellite images, or information objects that integrate images, annotations and videos. The operations on these objects can be extended in any direction without the limits imposed by the physical manifestation of the document. These operations, in turn, can generate new information objects that may convey

different semantic information. Furthermore, a DL can support the work of its users by providing functionality that may range from general utilities, like search, annotation, summarisation or co-operative work support, to very audience-specific functions, like processing of maps, semantic analysis of images, simulation, etc.

DLs are thus now moving far beyond any connotation of the term "library", and are rapidly shifting towards more general systems, now termed *Dynamic Universal Knowledge Environments* [dBGI04][1]. Through these environments, groups of individuals, collaborating towards a common goal, can be authorised to access, discuss and enhance on-line shared information. For example, in such an environment, a scientist can be enabled to annotate the article of a colleague with a program that extracts useful information from a large amount of data collected by a specific observatory. This annotation, executed on-demand when the annotation is accessed, will complement the content of the paper with continuously updated new information.

In parallel with the above evolution of the role of DL systems, we are now observing *a large expansion of the demand for DLs*. Research work today is often a collaborative and multidisciplinary effort carried out by groups belonging to different organisations distributed worldwide. Motivated by a common goal and funding opportunities, these groups dynamically aggregate into *virtual research organisations* that share their resources, e. g. knowledge, experimentation results, instruments, for the duration of their collaboration, creating new and more powerful virtual research environments. These virtual research organisations, set-up by individuals that do not necessarily have a great economic power and technical expertise, increasingly frequently require DLs as tools for accelerating the achievements of their research results. These new users demand less expensive and more dynamic DL development models. They want to be able to set up new DLs that serve their needs for the duration of their collaborations within an acceptable time frame and with an acceptable cost.

The current DL development model is not able to satisfy this large demand; a radical change is needed if we want to be able to address these new emerging requirements. New technologies must be investigated to support the implementation of novel functionality on the more versatile digital information objects. New organisational, development and maintenance models must be introduced to reduce their cost and to speed up their development time.

In this thesis we envision a new DL development model based on two main mechanisms: (*i*) *controlled sharing* of resources among multiple DLs and (*ii*) *virtualisation* of these resources in order to offer views of them that meet the specific needs of different application frameworks. These mechanisms enable the construction of *Virtual Digital Libraries*(VDLs), i. e. DLs built by dynamically aggregating and appropriately presenting the pool of resources needed to fulfil the user requirements. This scenario is depicted in Figure 1.1. By exploiting these mechanisms the cost of

[1]This expression was coined during the DELOS [DEL] brainstorming meeting held in Corvara on July 2004.

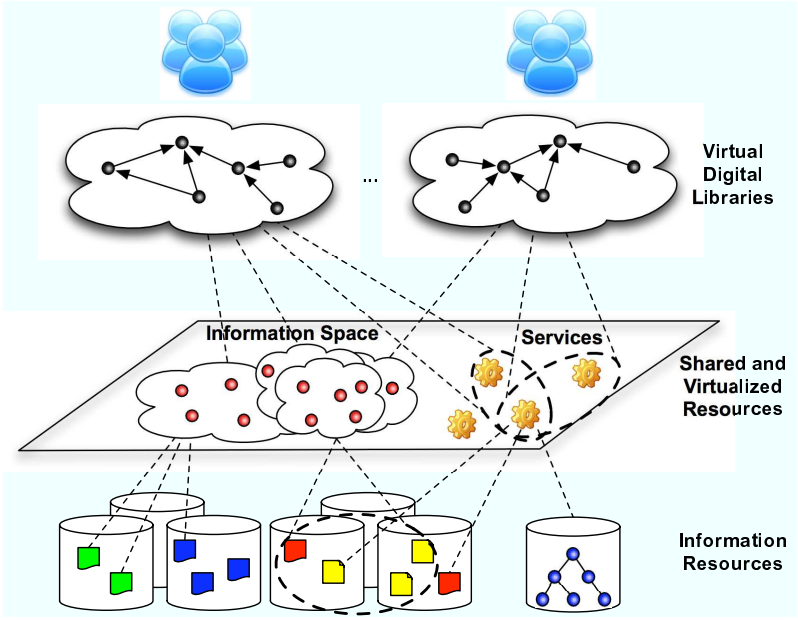

Figure 1.1: The Virtual Digital Libraries Scenario

DLs can be heavily reduced and a good level of user satisfaction can be achieved. The implementation of these mechanisms requires the design of appropriate DL architectures and the provision of a specific functionality. In fact, sharing implicitly introduces the need for components able to deal with many different heterogeneous resources while virtualization requires being able both to provide different views over these resources and to aggregate them differently. The nature and the complexity of such services depends to a large extent on the type of resources being shared. This can vary from the more traditional content sources, to services, and even to processing and storage capabilities.

This thesis proposes systematic solutions to some of the main issues involved in sharing and virtualisation and shows how these solutions have been embedded and validated in real DL systems.

1.2 Research Contributions

This dissertation introduces a framework for systemizing the design and the construction of Virtual Digital Libraries and presents a number of services that have been developed within real DL systems to support this view. The former group of illustrated services introduces mechanisms for implementing the notion of VDLs by sharing information sources, the latter supports a more general notion of shared resources and introduces the new concept of *on-demand transient VDLs*. In particular, the main innovative research contributions presented in this dissertation are:

- *A Reference Model for Digital Libraries*, i. e. an abstract framework for understanding significant relationships among the components of DLs and for developing consistent services that support them.

  Despite DLs have been introduced more than a decade ago, there is not yet a consensus on their main entities and expected functionality. In order to systematically design mechanisms for implementing VDLs, we made an effort to define the core of a model that represents the significant entities and relationships of a DL [CCP06a, CCP06b]. This model is currently being extended, validated and consolidated as part of an international activity funded by the DELOS Network Of Excellence on DLs [DEL].

- An approach to *information objects virtualization* that relies on set of mechanisms for hiding the heterogeneity of the information objects maintained in multiple shared information sources and for presenting them according to the needs of the DL users.

  The representation of information objects can be very heterogeneous in different sources. An information object can be a single element or a complex aggregation of parts, it can have versions and be disseminated in multiple physical manifestations. Furthermore, it can be associated with one or more metadata in different formats. This dissertation presents an approach to the virtualization of information objects extracted from/maintained in shared information sources. This approach is based on the introduction of a particular document model and on a number of components able to manage it. The dissertation describes also how this approach has been embedded in both the OpenDLib DL System [CP02, CP03] and in an extended version of it, OpenDLibG, and it shows the effects of the virtualization in two real DL application cases.

- An approach to the *collections virtualization* based on a set of mechanisms for supporting the dynamic construction of virtual collections that are built by exploiting the content maintained in multiple, shared information sources.

  The content of an information source is usually structured in a fixed set of collections that reflect the organisational choices of its specific application area. Users of a DL have different needs that often do not match the needs the original sources are built for. The innovative approach presented in this dissertation is based on a number of mechanisms that abstract from the heterogeneous and specific collections published by the information sources and support the on-demand creation of virtual collections. Authorised users can dynamically build these collections by specifying a set of characterisation criteria over the content of the shared information sources. Virtual collections have been implemented in both the OpenDLib and Cyclades [CYC] DL systems as part of this dissertation work. Other on-going projects, like BRICKS [BRI], are also exploiting the results presented here.

- An approach to *distributed semantic search* across a set of heterogeneous information sources whose objects are described and annotated with different metadata formats and ontologies.

  Search services are based on indexes that exploit the information objects metadata. These indexes are usually heterogeneous since the shared information sources support different metadata formats and ontologies, e. g. different metadata fields and different term vocabularies for the same field. The approach to distributed semantic search introduced in this dissertation supports a transparent and uniform search across multiple heterogeneous information sources. The novelty of the approach is its ability to exploit the ontologies of the shared information sources to allow the formulation of more semantically expressive queries. This approach has been experimented in an enhanced version of the OpenDLib DL system [CCP04].

- An approach to a *dynamic and (semi-)automatic generation of Virtual Digital Libraries* based on a shared pool of resources.

  By relying on a description of the resources and on a language for a declarative specification of DLs, both derived from the Reference Model conceptualisation, a service capable to collect the requirements of different communities and to identify the optimal pool of resources needed to fulfil the expressed requirements is introduced. The design and development of this service is being carried out as part of the currently on-going DILIGENT project [DIL] which aims at building a digital library infrastructure on Grid enabled technologies. In this project the notion of shared resources is generalised and comprises not only information resources but also applications, processing and storage resources.

1.3 Outline of Dissertation

This dissertation is organised in seven chapters and an appendix.

Chapter 1 introduces this dissertation by outlining the problem space, the motivations and the research contributions.

Chapter 2 presents the Reference Model for Digital Libraries. The chapter contains an overview of the model reporting also the principles adopted, presents the concepts and the relationships identified with the appropriate definitions. Throughout the chapter formal definitions are enriched with concrete examples explaining the definition in an intuitive manner. Finally, examples of exploitation and usage of the models with different goals are reported in order to prove its feasibility.

Chapter 3 elaborates the concept of *document model* presented into the Reference Model by presenting a Document Model for Digital Library (DoMDL). We report the requirements constraining the design of this model, present the OpenDLib Repository, i. e. a type of *Document Virtualizer* the OpenDLib system is equipped

with, and report concrete exploitation of this service. A further improvement obtained by combining the service with the Grid facilities is also reported. Finally, a survey on concrete document and data models used in various Digital Library Systems concludes the chapter.

Chapter 4 elaborates the concept of *collection* presented into the Reference Model. We report the design of the Collection Service, i. e. a service able to support the mechanism of collection, by presenting the architecture and the developed algorithms and techniques, i. e. *query sampling* and *source selection* needed to deal with non co-operative information sources. Moreover, we present the implementation of this service in the context of two concrete digital library systems, i. e. OpenDLib and CYCLADES. We conclude the chapter by presenting a survey on the exploitation of the concept of collection in concrete Digital Libraries and Digital Library Systems.

Chapter 5 elaborates on the concepts of *metadata mediator* and *services mediator* presented into the Reference Model by presenting the *Distributed Semantic Search*. The architecture and the underlying developed formal theory are described as well as the implementation is presented. Finally, a survey on approaches for distributed search implemented in concrete Digital Libraries and Digital Library Systems is presented.

Chapter 6 introduces the framework for implementing Virtual Digital Libraries. In particular, the *Virtual Digital Library Generator* service is presented by providing details about the exploitation of the concepts introduced into the Reference Model for defining a DL definition language, a DL component description language, and the matchmaking algorithm needed to identify the pool of components needed to fulfil a DL definition. The exploitation of this framework into the context of the IST EU project named DILIGENT completes the chapter.

Chapter 7 concludes this dissertation and presents future works proposals.

Appendix A presents an overview of OpenDLib, a digital library service system developed at ISTI-CNR and equipped with many of the services presented in this dissertation.

Related work is covered in the context of each chapter.

Chapter 2

Virtual Digital Libraries and the Digital Library Reference Model

In order to be able to build a DL by appropriately and dynamically aggregating a pool of shared resources providing both the content and the functionality required, i. e. to implement the Virtual Digital Library, a common understanding on what a DL is and what its characteristics are is needed. This chapter introduces a Reference Model for Digital Libraries [CCP06a, CCP06b], i. e. an abstract framework for understanding and explaining the significant entities and relationships among those entities in the DL environment.

The chapter is organised as follows. Section 2.1 introduces the Reference Model and describes the main principles underlying its organisation in incremental views, each perceived by one of the four typologies of actors identified. Section 2.2 presents the concepts and the relationships identified to fulfil the information needs of the DL End-users. Section 2.3 enriches the set of concepts and relationships by reporting DL Designer perspective. Section 2.4 introduces the DL System Administrator and the related model. Section 2.5 presents the concepts and relationships of interest with respect to the DL Application Developers. Section 2.6 introduces the Digital Library System Reference Architecture, a blueprint reporting the mapping of the concepts identified into software components that implement them, with the aim to promote loosely coupled development of these components and encourage reuse and integration. Finally, section 2.7 introduces and discusses related works.

2.1 Introduction

Until now digital libraries have evaded any definitional consensus. The main reason resides in DLs itself, i. e. digital libraries are complex systems, the underlying sciences are highly multidisciplinary and each community has its own perspective. As a consequence a plethora of DL definitions have been coined but none of them is comprehensive enough to represent DLs in all their flavours. For instance, Fox et.

Al. [FAFL95] observed that the phrase "digital library" evokes a different impression in each reader ranging from the simple computerisation of traditional libraries to space in which people communicate, share, and produce new knowledge and knowledge products. Belkin [Bel99] states that a DL is an institution in charge to provide at least the functionality of a traditional library in a context of distributed and networked collections of information objects. Lesk [Les99] analyses and discusses the importance of the terms "digital" and "library" in the expression "digital library", where the former term mainly corresponds to the software for searching text while the latter term corresponds to the scanning of existing material for online access, and concludes that the research effort in the field are not usually associated with the users' needs. Borgman [Bor99] notices that at least two competing visions of the expression "digital library" exist: researchers view digital libraries as content collected on behalf of user communities, while practising librarians view digital libraries as institutions or services. Kuny and Cleveland [KC96] discussed four myths about digital libraries with the aim to explode them, i. e. (*i*) the Internet is the digital library, (*ii*) the myth of a single digital library or one-window view of digital library collections, (*iii*) digital libraries will provide more equitable access, anywhere, any time, and (*iv*) digital libraries will be cheaper than print libraries; and concludes that digital libraries impose reinventing the role of librarians and the library models.

Our intention here is not to propose another DL definition comprehensive enough to cover any need simply because it is not feasible. Instead, we decided to propose a digital library reference model, i. e. an abstract framework for (*i*) understanding the significant entities and relationships between them within a DL environment and (*ii*) developing systems for supporting that environment.

Before to introducing this model, an informal clarification of what is our understanding about a DL and a system for supporting it is needed in order to make the context clear. We define a digital library as follows.

Definition 2.1.1 (Digital Library) *A networked entity with the aim to provide at least the functions of a library in the context of distributed, networked collections of information objects in digital form representing the DL information space.*

In order to identify the system in charge to operate and provide DLs we introduce the concept of *Digital Library System.*

Definition 2.1.2 (Digital Library System) *A software system providing the digital library functionality on a set of information objects.*

In many contexts the concepts of DL and DLS tends to collapse into the same entity, mainly because the first entity is an abstract entity that is perceives when implemented via the second one[1].

[1]Throughout this dissertation, where no confusion arises, we will use the terms DL and DLS as synonyms and tend to prefer the first for both. In particular, all the concept maps we present are rooted by the Digital Library concept that represents both the entities.

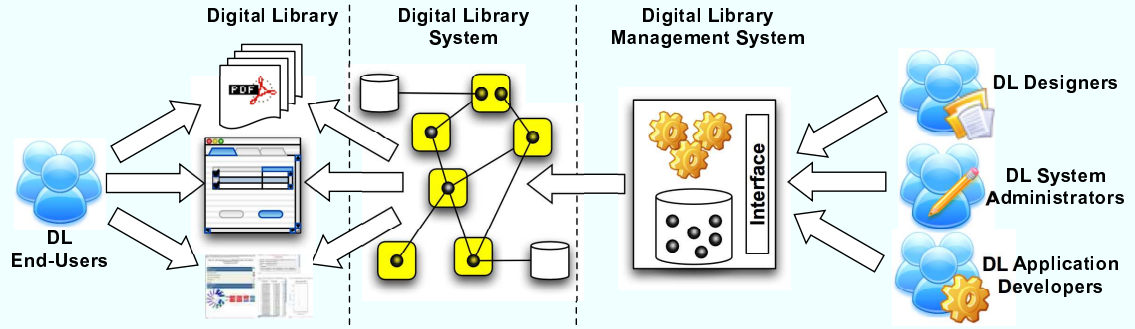

Figure 2.1: DL, DLS, and DLMS – A three-tier framework

A second type of system is needed to manage DLs. In accordance with [IMA+05] we call it *Digital Library Management System* (DLMS) and define it as follows.

Definition 2.1.3 (Digital Library Management System) *A software system in charge to creating and managing DLs.*

The three-tier framework arising from these definitions is depicted in Figure 2.1. The characteristic that differentiate the two systems resides into the class of objects objective of the management activity: in the case of the DLS it is the information space of the digital library while in the case of the DLMS is the digital library itself[2]. It is also important to notice that none of the nowadays existing systems for digital libraries is a DLMS and that a digital library can be created also without a DLMS. These systems represent what the digital library research community has established to be created into the near future in order to overcome the drawbacks arising from the current digital libraries development process [IMA+05].

In order to describe digital libraries, digital library systems and digital library management systems and characterise them appropriately, we decided to look at them from the perspectives of the actors that operate with them. These perspectives vary according to the role played by the actor. We focus our attention on the modelling needs of four main actor roles and we introduce an appropriate perspective for each of them as described in the following section.

2.1.1 The Perspectives

The four roles taken into account in analysing and describing digital libraries are: *DL End-user*, *DL Designer*, *DL System Administrator*, and DL *Application Developer*

[2]For those familiar with Data Base (DB) and Data Base Management Systems (DBMS) it is worth noting the existing differences to avoid misunderstanding due to similar names. In particular, in the DB area the goal is to manage a pool of data and the DBMS is in charge to maintain the data and provide the functionality to manage them. In the DL area the goal is to manage a pool of information objects and the DLS is in charge to provide the functionality to manage them, while the DLMS is a system capable to instantiate DLSs and do not have any counterpart in the DB field.

and are described as follows.

DL End-users

The actors that exploit the digital library functionality for providing, consuming and managing the DL content. They perceive the DL as a stateful entity which serves their functional needs through the interaction with an instance of a Digital Library System. It is worth noting that the behaviour and the outcomes of the functionality depend on the state of the digital library at the time of the request, where the state of the digital library meant here is represented by the set of collections of information objects available plus the pool of users authorised to access the digital library. This state is continuously evolving accordingly to the functionality activated by the various users.

DL Designers

The actors that, by exploiting their knowledge of the application domain semantics, customise and maintain the DL aligned with respect to the information and functional needs of its end-users. These actors correspond to the chief librarians in a tradition library, they are in charge to define the rules and the policies holding into the digital library with respect to the information space composition and organisation, the functionality to be provided, the typologies of users entitled to have access to the digital library, the quality of the services offered and any other aspect related to the arrangement of the digital library. These actors perform their task by interacting with the Digital Library Management System. By using the operations provided by this system, they identify, among the set of possible DLs that can be realised with the given DLMS, the one that better satisfies the application needs of the end-users. Then, they define a number of (*i*) functional configuration parameters, i. e. parameters that characterise the format of a specific DL functionality as perceived by the end-user, like metadata formats, query language, user profile formats, (*ii*) content configuration parameters, i. e. parameters that characterise the accessible content, like the information sources to be harvested, and (*iii*) quality configuration parameters, i. e. aspects characterising qualitative behaviour of the system functionality, e. g. the response time of a search task. The value of these parameters can be modified during the digital library lifetime. Any change at this level results in a change of the digital library state that determines the features perceived by the end-users.

DL System Administrators

The actors that select the Digital Library System software components to install in order to implement the required digital library and decide where and how to deploy them. They interact with the DLMS by invoking specific operations that require

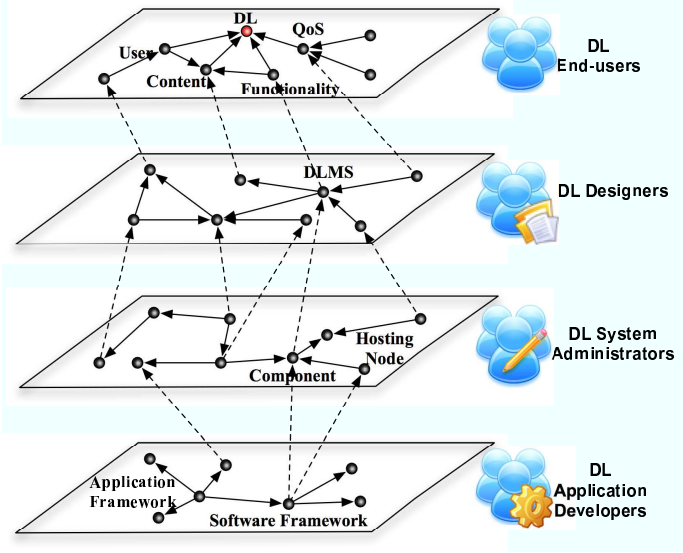

Figure 2.2: DL perspectives hierarchy

to provide architecture configuration parameters, like the selected software components, the hosting nodes, the components allocation, etc. Their task is to identify the architectural configuration, among those supported by the chosen DLMS, that better implements the digital library configuration established by the DL Designer and ensure the required quality of service. The value of the architecture configuration parameters can be changed over the DL lifetime. The change of these parameters may result in a different DL functionality when a new software component is added and, more generally, in a different level of quality of service provided by the digital library.

DL Application Developers

The actors in charge to develop the digital library management system components or extend the digital library system software components made available by a DLMS in order to satisfy the additional needs of a DL specific application framework. This means that both DLMS and DLS are open systems whose set of functionality can be easily enlarged or improved by adding novel components. Thanks to this capability and to the envisaged configuration aspects we are designing a software system typology capable to evolve in accordance to the users requirements and thus able to fulfil the needs arising in any application scenario with the minimum effort as possible.

DL Perspectives Hierarchy

These four roles[3], taken in the order, identify four different perspectives each of which is an extension of the previous one (see Figure 2.2) both in terms of the number of concepts and relationships identified and in terms of the details associated to each entity/relationship.

The DL End-users only perceive what a specific digital library provides them, therefore they need a model of the DL that comprises the concepts and relationships required to interact with a single DL, e. g. the structure of the information objects, the organisation of the information space, the functionality provided.

The DL Designer uses a DLMS to configure a DL serving a specific class of end-users. Therefore, they need a model that represents the configuration functionality offered by the DLMS and the set of resulting DLs which can be created through this system, as perceived by the end-users. For instance, their model must be able to represent characteristics of the digital library information space like the allowed information objects structures and the allowed metadata formats, the operational and qualitative characteristics of the offered functionality, the policies regulating the activities performed by the DL End-users.

The DL System Administrators, which are responsible for selecting and deploying the digital library system software components that implement the functional (e. g. a type of search, the publishing procedure) and content choices (e. g. import of a collection from an information source, the metadata schema to be used for cataloguing information objects) established by the DL Designers, need a more complete model of both the DLMS and DL. This model must be suitable to represent not only the functional aspects of the supported DLs but, also, the components that implement the functionality and the hosting nodes where these components can be deployed.

Finally, the DL Application Developers require the most complete representation of the digital library system, namely from the system's architecture point of view because they are going to produce new components that have to co-operate with the already existing ones in order to deliver novel functionality. In particular, they need a model that specifies the underlying software and application frameworks[4], the relationships and dependencies among the software components, and how these are related to the end-user functionality.

For each of the identified perspective we need to cluster the pool of concepts and relationships characterising the model according to few and well established dimensional aspects that are identified and described in the following section.

[3]It is worth noting that the actors meant here are not necessarily humans. They can be software agents that operate as dictated by the corresponding role. In this case the model is a necessary key element since it establishes the context for the definition of the algorithms that implement the automatic actor. A concrete example of such agents is the VDL Generator that replaces the DL System Administrator, as presented in Chapter 6.

[4]These concepts are explained in detail in Section 2.5.

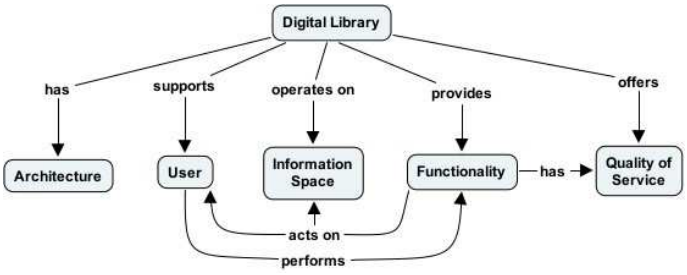

Figure 2.3: The Digital Library main concepts

2.1.2 The Digital Library Main Concepts

Figure 2.3 presents the *concept map*[5] [NG84] of the most important and high level concepts characterising a digital library, i. e. the *Architecture*, the *User*, the *Information Space*, the *Functionality*, and the *Quality of Service*. In accordance to the map, each digital library supports users, provides functionality, operates on an information space and offers a level of quality. Moreover, it is characterised by an architecture. These five ingredients allow us to cover the whole spectrum of characteristics and aspects related to digital libraries.

Information Space

The information space represents the most important resource of any digital library, i. e. it corresponds to the whole set of information the digital library makes available to its users. It is usual to consider the information space as composed by a pool of *information objects* organised into *collections* especially from a DL End-user perspective but under this umbrella we aggregate all the concepts related to and needed for dealing with the management of any type of information the digital library is going to offer to its users. In particular, the metadata play an important role. They can be used in different contexts and for different purposes, e. g. they describe the content of an information object, they express the structure of a complex object, the characterise the policies regulating the usage, etc. Finally, it is important to remark how the understanding of the concepts classified into this area varies in accordance to the perspectives introduced previously. For instance, an information object perceived by the End-user is usually the unit of information she/he is looking for; an information object for the DL Designer represents a unit of information to make available to the DL End-users; an information object for the System Administrator

[5]A concept map is an informal graphical way to illustrate key concepts and relationships among them. A concept identifies a class of objects that we expect to be able to identify in the proposed context. The precise "form" of concept c may be different in diverse implementations, but the presence of the concept tells us what to look for in a given concrete scenario rather than prescribing its precise form. This formalism offers a practical method to represent complex information in a compact and easy-to-understand way.

is a part of a component that must be maintained up and running into the digital library system; finally, an information object for the DL Application Developer is composed by a set of files stored somewhere that must be manipulated by the components she/he is developing.

User

The user dimension represents the second most important resource of any digital library, i. e. the actors entitled to interact with it. In fact, the role of the digital libraries is to connect people with information and support the users in performing their tasks, namely, producing new information objects or consuming already available information objects. As in the case of the previous dimension, the umbrella of User covers all the concepts and relationships related to representation and management of human entities into the system, e. g. it contains the digital entity representing the human actor, the rights the entity has into the system, the characteristics the human decides to be represented with for collaboration aspects. Also in this case, the perception of the same concept varies in accordance of the user role perspectives. For instance, in the case of DL End-users a user is a human person the system provides with information; the user perceived by a DL Designer can still be a human person or a class of human persons she/he is going to provide rights; in the case of the System Administrator a user is a digital item stored into a digital library system component she/he is in charge to equip the digital library with; finally, the user perceived by a DL Application Developer is a data structure reporting for instance an identifier and an email address representing a user she/he must take into account in designing and developing the component providing a certain functionality.

Functionality

A digital library is composed by an information space and a set of users aggregated with the aim to give the users a pool of processes to operate on these elements, i. e. the functionality. The functionality expected from digital libraries are not fixed neither in type nor in form, e. g. there exists a set of core functionality each digital library must provide like submission, search, browse, but as previously explained each user community may ask for a certain type of a core functionality or for a novel type of functionality for using and profiting of digital library resources. Moreover, functionality are perceived differently from the four identified user roles. For instance, the search functionality is perceived as the mechanism to retrieve the information objects she/he is interested in from the point of view of the DL End-user; in the case of the DL Designer, a search functionality represents the access path to the content available into the digital library and must be customised to fulfil the user requirements; the same functionality is instead perceived as one or more software modules to be deployed and made available from the DL System Administrator that

must take care of the performance offered by such components and eventually study replication and distribution strategies for improving them; finally, the search functionality is a complex process the DL Application Developer must take care of when implementing the digital library system software component in charge to provide it or participating in providing it.

Quality of Service

The Quality of Service concept groups a pool of qualitative parameters characterising the digital library behaviour, in particular it represents measurable and non-functional characteristics of a functionality within a given operational domain. It is usual to have different characteristics for diverse functionality and a measurement of each of them, whether it is objective or subjective. The objective measure is performed automatically and can be repeated while the subjective measure involves humans and their personal feeling. Among the other aspects considered in evaluating digital libraries, quality of services measurement plays an important role in characterising these complex systems. Unlike the previous concepts, quality of service concepts by themselves have no different perceptions when moving from one user role to another, the differences of perceptions can reside on the object the user role is going to measure, on the measure the user role is interested in and on the process behind the observed value. For instance, the response time of a given functionality is measured in average time per request from the point of view of all the user roles with the following differences. The End-users are passive subjects and usually cannot do much more than ask the DL Designers for an improvement. The DL Designer establishes the threshold, i. e. the maximum average time tolerated into the digital library she/he is responsible for. The DL System Administrator is in charge to deploy and configure the components of the digital library system providing the functionality by ensuring the required quality of service level. Finally, the DL Application Developer is in charge to carefully evaluate the consequences of its design and implementation choices in terms of the quality of service offered by the component she/he is going to develop.

Architecture

A common understanding of digital libraries is that they are among the most complex and advanced forms of information system [FM98]. Moreover, as thousands of digital libraries exist around the world and new ones are emerging, one of the biggest issue of the research community is to make these heterogeneous digital libraries interoperable. The architectural understanding of the digital library systems behind the digital libraries becomes thus a foundational dimension in reaching this goal. Unlike the other main four digital library concepts, the architecture becomes meaningful and of pertinence of the DL System Administrators and DL Application Developers only. The DL End-users do not take care of this characteristic because

they usually perceive the digital library by interacting with a graphical user interface provided in a web browser. The DL Designers are usually expert librarians and do not have the technical skills to deal with the architecture of the system that is in the hand of the DL System Administrators. From our point of view and for the sake of this reference model, the most comprehensive definition of the architecture concept is "A representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components"[6]. One of the contributes of this model is the reference architecture for digital library systems reported in Section 2.6.

2.2 The DL End-user Perspective

End-users are the actors interacting with the digital library in order to exploit the resources and the facilities offered by this entity. They perceive the DL as the stateful entity capable to fulfil their information and functional needs and do not take care about the underlying digital library and digital library management systems. In describing this perspective we have identified three classes of actors, i. e. *Content Consumer*, *Content Provider*, and *Librarian*, and four classes of functionality, i. e. *Content Management*, *Access*, *Personalisation*, and *DL Management* as reported in Sections 2.2.2 and 2.2.3 respectively. Figures 2.4, 2.6, 2.5, and 2.7 reports concepts and relationships modelling DLs from these users perspectives clustered according to the five main DL concepts. Actually, as these users do not perceive the Architecture facet, the dimensions taken into account are the remaining four. These modelling elements are described in detail in the next subsections and illustrated by giving examples.

2.2.1 Information Space

This section introduces the main concepts that characterise the information space of a specific DL as perceived by the DL End-users. Note that this representation of the information space, and even its instances, may be different from the concrete representation manipulated by the software components that implement the DL. The representation meant here is the one that is disseminated by the functionality of the DL that support content creation, access and management.

The main concept of the digital library information space is the information object.

Definition 2.2.1 (Information Object) *The main unit of information which is managed by the DL. An information object has an Information Object Identifier for*

[6]Glossary of the Carnegie Mellon University's Software Engineering Institute. `http://www.sei.cmu.edu/opensystems/glossary.html`

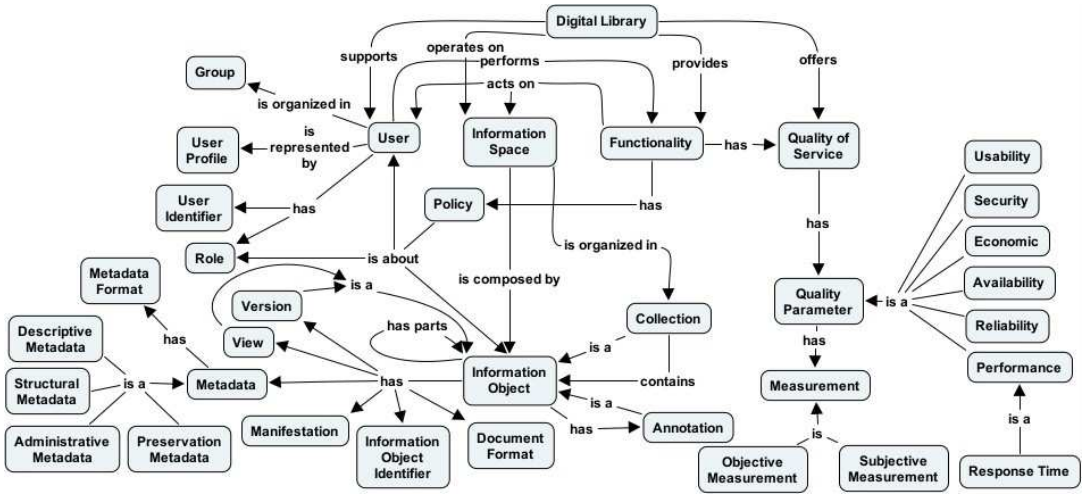

Figure 2.4: The DL End-user concept map – Main concepts

identification purposes and Metadata for various management purposes. Moreover, each Information Object can: (i) be structured, i. e. it can be composed by other information objects, (ii) have multiple versions, views, and manifestations, and (iii) be annotated.

The definition above contains the motivations convinced us to avoid the classical term "document" to represent the unit of data managed by the digital library. Nowadays, we are moving far from the digital objects as counterparts of the physical documents stored into a traditional library; even if great effort is spent in digitalise existing material, most information "born digital" today results in novel type of "documents" more flexible and informative than the classical ones. Examples of such information objects are: sound recording of voices equipping a set of slides, sheet music whose content can be eared rather than observed in accessing it, politic and economic data equipped with interactive simulations. Also the classic documents take advantages in being digital, for instance a Ph.D. thesis can be represented via an information object composed by one information object for each chapter as well as an information object containing a simulation of one of the experiments conducted and an information object containing the experimental data set adopted. The message here is: information objects are complex objects and digital libraries should be prepared to manage them even if their complexity is not known a-priory and can evolve during the digital library lifetime. However, in order to manage them the digital library must be able at least to identify them, thus in our model each information object has an information object identifier.

Definition 2.2.2 (Information Object Identifier) *The minimal information enabling to distinguish one Information Object from all the others within an identification scope.*

Various forms of information objects identifiers can be envisaged each having the aim to univocally recognise an entity a certain context. These can vary from simple sequential numbers to URIs and Digital Object Identifiers[7] (DOIs). Clearly, each of them has a different discriminating power when considered in the context of digital libraries whose content can be gathered from worldwide located information sources. Therefore carefully attention must be posed in detecting duplicate copies of the same object. In the context of virtual digital libraries universally unique identifiers becomes of vital importance. In particular, a good candidate is represented by DOI, i. e. a name (not a location) for an entity on digital networks. It provides a system for persistent and actionable identification and interoperable exchange of managed information on digital networks.

As stated by Def. 2.2.1 an information object has multiple versions.

Definition 2.2.3 (Version) *An expression of an information object along the time dimension.*

The concept of versioning is well known and applies successfully in traditional libraries. As for any other concept we inherit from the library area, the goal is to maintain the semantic as much as possible unchanged even if the models can be revised. For instance, the draft version, the version submitted, and the version published in the proceedings are different versions of the same information object representing a paper submitted to a conference. Another example is the information object reporting the graph presenting the growth of the material available into a digital library, the per year or per month graphs represent different versions of the same information object. Our model introduces the concept of version but does not constrain it allowing each community to use it in the most profitable form. Moreover, It is worth noting that versions of an information object are themselves other information objects populating the digital library information space.

Besides different versions, the same information object can have different views.

Definition 2.2.4 (View) *A way through which the information object is perceived.*

The concept of view is useful to represent the diverse expressions an information object may assume, where the term "expression" is taken from [IFL]. This aspect becomes particularly important in the digital era where diverse expressions of the same object can be easily created. It is worth to remark that physical aspects do not contribute to the generation of different views that thus are mechanisms to differentiate the other information object perceptions. For instance, considering an information object representing the outcomes of a workshop, three different views of this object can be envisaged: (*i*) the "full view" containing a preface prepared by the conference chair and the whole set of papers accepted and organised thematically, (*ii*) the "handbook view" containing the conference program and the slides of each

[7] `http://www.doi.org/`

lecturer accompanied with the abstract of the paper the talk is about and organised per section, and (*iii*) the "informative view" reporting the goal of the workshop and the list of the accepted papers' title and the relative abstract. The mechanism offered by views allow to tailor the information object expression to different user needs. As in the case of versioning, it is worth noting that views are information objects populating the DL information space and the model does not constrain them in any form but limits to introduce them as important concept in future DLs.

In accordance with Def. 2.2.1, an information object has multiple versions, multiple views, and multiple manifestations.

Definition 2.2.5 (Manifestation) *The physical embodiment of an Information Object.*

The concepts introduced until now deal with the logical organisation of information objects, while the manifestation deals with the physical presentation of its informative content. This concept probably is the most important one as regards how the users are provided with the information they are looking for. It is worth noting that we are dealing with digital objects and thus the manifestation is itself a digital object. Examples of manifestations are the PDF file of a paper, the MPEG file containing the video recording of a lecture, a text file containing the raw data observed by a sensor, an xml file reporting the results of a certain elaboration. These pieces of information can be physically stored into the digital library somewhere as well as dynamically generated, from the DL End-users perspective it is only important that they can be being, to have access to them.

Until now we have introduced the main ingredients an information object is made with. However, a digital library usually provides DL End-users with information objects compliant with a document format.

Definition 2.2.6 (Document Format) *The abstract model through which an information object is perceived by the user.*

To make information objects more useful for the DL End-users it is usual to establish and organise them in accordance with a set of document formats. For instance, a digital library containing Ph. D. thesis can be designed to adopt the document format "thesis" that constrain information objects to be organised in a part representing the cover page, a part representing the preface, and a part for each chapter. It is matter of the DL Designers to define the document model the digital library adopts as explained in Section 2.3.

An important concept related to information objects is metadata, each information object has its own metadata.

Definition 2.2.7 (Metadata) *Additional data about an information object.*

The "classic" definition of metadata is "data about data". These additional data can be used in different contexts and with different purposes; our model captures the needs to have them associated to an information object as a means for enhancing the functionality and in general the management of the object. Examples of metadata of an information object are its bibliographic record, the policies regulating the access to its content, a description of the structure the object is organised in, a description of the preservation process adopted. It is worth noting that the model does not constrain the application of metadata in fixed contexts but introduces a placeholder the DL End-user are interested in when use a digital library.

To enable DL End-users to fruitfully use the information stored into metadata it is mandatory to share a common understanding of them with their producer. Usually, this common understanding is reached by relying on the metadata format as a mechanism for abstractly describe the "shape" metadata will have into the digital library.

Definition 2.2.8 (Metadata Format) *The way in which metadata is arranged or set out.*

Each digital library is entitled to define and adopt its own metadata format but in order to improve interoperability and reuse of already existing stuffs, it is usual to adopt well known metadata format. A common format for bibliographic records is represented by the Dublin Core [DC], probably due to its simplicity on the contrary of the MARC format [MAR05] that is much more articulated. Another example of metadata format is represented by the Metadata Encoding and Transmission Standard (METS) [The02], i. e. a standard providing and encoding format for descriptive, administrative, and structural metadata designed both to support the management of the information objects and the delivery and exchange of them across systems. However, it is also usual to combine multiple metadata formats into novel metadata format named *application profile* in order to fit them with the digital library needs.

Having clarified the role metadata plays into the digital library and the needs to describe them according to a common format, in the following definitions we report the main categories of metadata adopted in digital libraries, i. e. descriptive, administrative, structural, and preservation metadata.

Definition 2.2.9 (Descriptive Metadata) *Metadata for discovery and interpretation of an information object.*

As stated, metadata are used to support the functionality acting on the information objects. In particular, the umbrella of descriptive metadata covers the additional information supporting information object finding and interpretation. Examples of descriptive metadata are a bibliographic record, a list of keywords characterising the information object content, a summary describing the object.

Definition 2.2.10 (Administrative Metadata) *Metadata for managing the digital object and providing more information about its creation and any constraint governing its use.*

This category of metadata might include: metadata describing technical characteristics of an information object, metadata describing the object from which the information object was produced, metadata describing the history of the operations performed on the information object since its creation/capture, metadata describing copyright, use restrictions and license agreements constraining the use of the object.

Definition 2.2.11 (Structural Metadata) *Metadata describing the logical or physical relationships between the information object parts.*

Structural metadata represent the glue enabling to deal with complex information objects composed by aggregating in novel ways pre-existing information objects. Even if these metadata are not directly seen by the content consumers and content creators which perceive the structure of the objects by directly accessing them or using the fixed document format, they are matter of the library operators that may need to have access to the internal structure of the objects to perform their content management tasks. Moreover, this kind of metadata is usually of interest of the technical actors, namely the DL Application Developer.

Definition 2.2.12 (Preservation Metadata) *Metadata for supporting preservation tasks.*

Many work have been conducted to investigate the most appropriate form and content this type of metadata should have. For instance the PREMIS, an OCLC and RLG international working group, prepared a Data Dictionary with the aim to define a set of "core" preservation metadata elements [PRE05]. PREMIS defines "preservation metadata" as the information a repository uses to support the digital preservation process. Specifically, this group focussed on metadata supporting the functions of maintaining viability, renderability, understandability, authenticity, and identity in a preservation context. Preservation metadata thus spans a number of the categories typically used to differentiate types of metadata: administrative (including rights and permissions), technical, and structural. Particular attention was paid to the documentation of digital provenance (the history of an object) and to the documentation of relationships, especially relationships among different objects within the preservation repository.

Having clarified the concepts of information objects and their related metadata we come back to Figure 2.4 in order to recall that the DL End-users perceive the digital library as composed by information objects and organised in collections.

Definition 2.2.13 (Collection) *An information object representing a set of information objects grouped according to a characterisation criterion.*

Collections represent the "classic" mechanism to organise a huge amount of information objects and to provide focused views. Thanks to these focused views the DL End-users are entitled to have access to thematic parts of the whole information space and avoid to deal with the volume of data the digital library makes available. These focused views can be created by the library operators in order to support content consumers and content creators to keep the library information space organised and improve it access and usage; further, they can be created by authorised content consumers in order to implement their own personal views of the digital library information space. The definition and identification of the objects constituting each collection is based on a set of characterisation criteria. These criteria can range from the enumeration of the information objects belonging to the collection to membership conditions that specify which properties information objects must share to become collection members. Finally, it is worth noting that collections themselves are considered information objects of the digital library information space. The main consequences of this aspect of definition are: (*i*) a collection is the object of the same functionality as the information objects, e. g. the search, the browse, the annotate, the preserve, and (*ii*) it is possible to organise collections hierarchically via the "has part" relationship.

In Chapter 4 we present the exploitation of the collections as the mechanism to organise the information space of a virtual digital library and provide concrete examples of exploitation of this mechanism in concrete scenarios like the CYCLADES project and the OpenDLib based digital libraries.

The last concept related to the digital library information space is annotation.

Definition 2.2.14 (Annotation) *An information object representing extra information associated with another information object.*

Annotations represent an important type of information objects used especially in digital libraries supporting co-operative work. This information can be used in various contexts, e. g. to express a personal opinion about an information object, to enrich an information object with references to related works or contradictory information objects, to add personal notes about a retrieved information object for future usage. It is also important to notice that we decided to give annotations the importance of information objects for two reasons, (*i*) due to their nature annotations themselves are information objects, (*ii*) to be effective instrument of co-operation they must have the same expressive power as information objects. As a consequence, annotations managed in digital libraries can assume different formats and be expressed in different media. Moreover, they have assigned metadata reporting additional information enabling the content consumer as well as the content creator to assign the appropriate weight to this type of information.

2.2.2 User

The user dimension is used to introduce and present the concepts needed to represent and support the human actors in charge of interacting with the digital library. The minimal set of concepts and relationships needed for this purpose is reported in Figure 2.4.

First of all we are interested in being able to univocally identify these users and thus we need a User Identifier.

Definition 2.2.15 (User Identifier) *The minimal information enabling to distinguish one user from all the others within an identification scope.*

As in the case of Information Object Identifiers, this identifier must be unique within the digital library and in the case of interoperation among different DLs a mechanism to exchange these identifiers and resolve possible conflicts is needed.

Having univocally identified users, we need also user profiles in order to maintain information representing them.

Definition 2.2.16 (User Profile) *The descriptive information the digital library maintains about a single user.*

The user profile is the container of any characterising information of digital library users except information related to the policies. This is covered separately, via the usage of the user role. Examples of information usually gathered about users are: (*i*) the contact information like name, mail and e-mail addresses, (*ii*) expression of interests regarding the digital library information space; this can be advertised by the user himself or dynamically computed by the digital library system in accordance to the usage and the behaviour of the user in the DL context, (*iii*) various statistics like the number of information object accessed, their type, the number of logs in, (*iv*) personalisation and customisation of the digital library environment like look and feel of the user interface, preferred services of the digital library to interact with. No standard for users profiles exists. Moreover, which information the user profile contains depends on the functionality the digital library is going to provide and vice versa, i. e. the type and the behaviour of functionality that relies on the user profile data are a direct consequence of the information these data contain. For instance the digital library could provide a per-user personalised recommendation functionality if and only if the user profile contains data about the topics of interest.

As in any other system dealing with many users, it is usual to introduce the concept of group in order to manage a pool of users as a single entity.

Definition 2.2.17 (Group) *A number of users that are considered or classed together.*

Groups play, with respect to users, the same role as collection do with respect to information objects. As a consequence, a group is considered as a meta-user and thus can be the object of the same functionality, e. g. advertise an information object, grant or remove access rights.

The latter aspect, i. e. that related to access rights, is actually regulated by relying on the concept of role.

Definition 2.2.18 (Role) *A job function within the context of an organisation, i. e. the DL, with some associated semantics regarding the authority and responsibility conferred on the user assigned role.*

The above definition comes from [FKC03] and works in accordance with the policy mechanism described in Section 2.2.3.

As anticipated, our model introduces three roles representing three types of actors interacting with the digital library, i. e. the content creator, the content consumer, and the librarian.

Definition 2.2.19 (Content Creator) *The role associated with the users in charge of providing new information objects to be stored into the digital library or updating already existing information objects.*

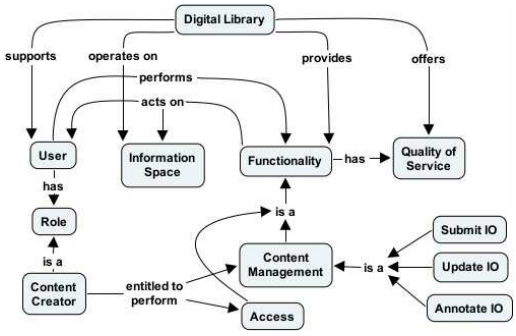

Figure 2.5: The DL End-user concept map – Content Creator functionality

Figure 2.5 depicts the Content Creator concept map and in particular reports the functionality the users having this role are entitled to perform, i. e. submit a novel information object, update an existing information object, and annotate an information object. For a description of them please refer to Section 2.2.3.

Definition 2.2.20 (Content Consumer) *The role associated to the users accessing the digital library in order to consume its information objects.*

Figure 2.6 depicts the Content Consumer concept map and in particular reports the functionality the users having this role are entitled to perform, namely access DL information objects. For a description of them please refer to Section 2.2.3.

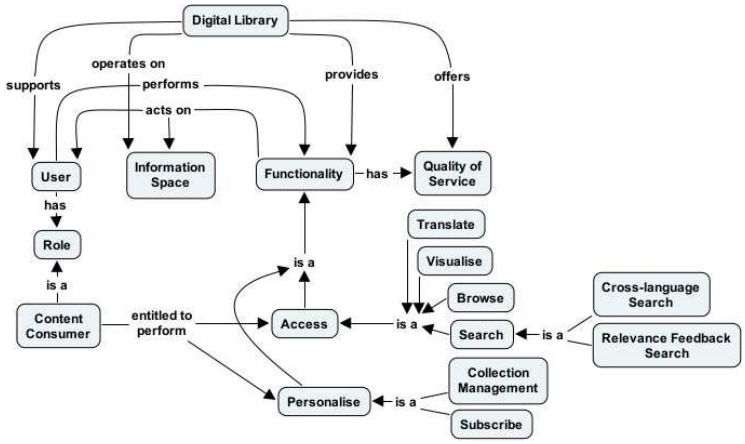

Figure 2.6: The DL End-user concept map – Content Consumer functionality

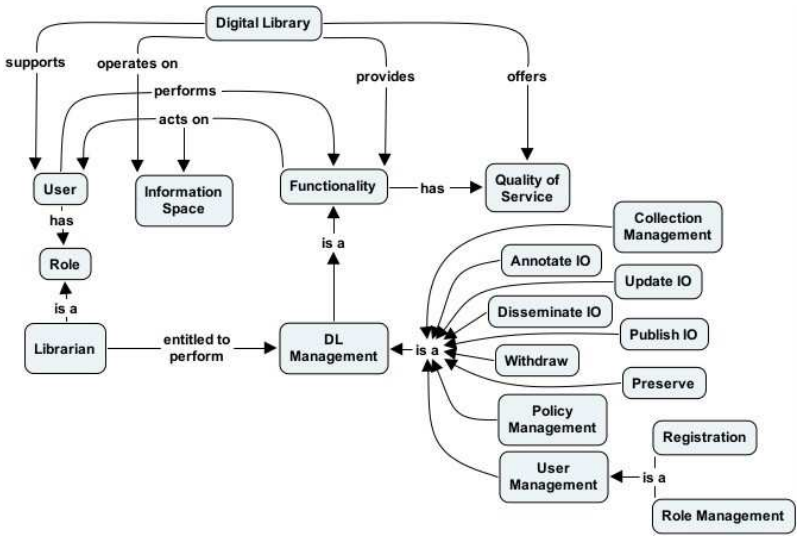

Figure 2.7: The DL End-user concept map – Librarian functionality

Definition 2.2.21 (Librarian) *The role associated to the users of the digital library in charge of managing it by performing the day by day librarian tasks.*

Figure 2.7 depicts the Librarian concept map and in particular reports the functionality the users having this role are entitled to perform, namely the management of information objects constituting the digital library and users having access to them. For a description of them please refer to Section 2.2.3.

2.2.3 Functionality

Below we list the mandatory functionality that the DL End-users of digital libraries expect from such systems. Note that DLs may also provide additional functionality

to cover the peculiar needs of specific application frameworks.

Before going in detail and describe this list, we recall that the usage of the digital library functionality is usually regulated by policies as reported in our model (Figure 2.4, page 17).

Definition 2.2.22 (Policy) *A triple (role,functionality,object) where the object is an information object or a user regulating the usage of the functionality in the digital library context.*

Policies are the mechanism used to regulate and restrict the DL system access and usage to authorised users. Various approaches exist in implementing access control mechanisms, e. g. Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role-Based Access Control (RBAC). In modelling the access control we used the RBAC approach [FKC03]. This is the main reason for introducing the role concept. Coming back to the model reported in Figure 2.4, a policy is associated to functionality and is used to restrict the usage of the single functionality to an established role. Moreover, as functionality may acts on both users and information objects and thus the policy must be able to identify the objects that are affected by the functionality in order to provide an effective and fine grained access control mechanism.

In order to identify, describe and organise this set of functionality each digital library is in charge of providing, we proceed analysing the functional needs of the three roles introduced in Section 2.2.2, i. e. Content Creator, Content Consumer, and Librarian. These functionalities are reported in Figures 2.5, 2.6, and 2.7 respectively. From this analysis and thus from the DL End-user point of view it arises that a digital library must provide four types of high level functionalities: Content Management, Access, DL Management, and Personalisation.

The content creator role must be entitled to perform content management functionality.

Definition 2.2.23 (Content Management) *The functionality acting on the digital library information space to produce new information objects or updating already existing information objects.*

Actually, this functionality represents a family of functionalities because the tasks to be performed in managing a set of objects are numerous. In the following, we identify and describe the main types of content management functionalities, i. e. submit, update, and annotate an information object.

Definition 2.2.24 (Submit Information Object) *The functionality allowing the user to define and provide a new information object to be stored and made available into the digital library.*

It is worth noting that this functionality contains the authoring phase supporting the content creator to define the information object to be submitted. During the authoring phase the content creator must be enabled to create information objects accordingly to one of the document formats (Def. 2.2.6) supported by the digital library as well as to identify and reuse already existing information objects in order to build complex objects. It is also important to notice that according to the digital library policies established by the DL Designer (Section 2.3), the submit functionality usually adds the newly created information object to the incoming information space, i. e. a temporary area that contains all the objects waiting for published (Def. 2.2.38) by the librarians, or it may add the object directly to the DL information space.

Definition 2.2.25 (Update Information Object) *The functionality acting on the digital library information space which allows users to modify an already existing information objects.*

This functionality is similar to the submission functionality because it implies authoring capabilities in order to rearrange the information object and usually produces a novel information object that may be a new version (Def. 2.2.3), a new view (Def. 2.2.4), or a new manifestation (Def. 2.2.5) of an already existing information object.

It is important to notice that we have modelled collections as information objects (Def. 2.2.13) and thus the functionality described applies also to this type of objects. It is clear that the authoring mechanisms should be different, however they produce information objects compliant with a particular document format. Another type of information object is represented by annotations (Def. 2.2.14).

Definition 2.2.26 (Annotate Information Object) *The functionality acting on an information object of the digital library information space to produce an additional piece of information, i. e. the annotation, to be assigned to the information object itself.*

The content consumer as well as the content creator roles must be entitled to perform access functionality in order to discover and use the information objects populating the digital library they are interested in.

Definition 2.2.27 (Access) *The functionality acting on the digital library information space and providing users with mechanisms for discovering and using the information objects they are interested in.*

Actually, this functionality represents a family of functionalities because the tasks for accessing and using the information objects are numerous and dependant from the type of objects the digital library deals with. In the following, we identify and describe the main types of access functionalities, namely the visualisation, the translation, and the various forms of search and browse.

Definition 2.2.28 (Visualise) *The functionality enabling the users to perceive an information object in a graphical way on an appropriate device.*

As previously stressed, information objects may be complex objects and they may combine information manifested in different media. Thus, this functionality must be tailored to the end-user characteristics, like the device used or their personal setting (usually this information belongs to the profile – Def. 2.2.16), as well as to the characteristics of the object to be rendered. Actually, these characteristics hold and apply to the whole user interface of the digital library, because it has to provide the users with a comfortable environment, giving intuitive and easy access to the object they needs as well as to the functionality they are interested in.

Besides offering a graphical interface, digital libraries are in charge of facilitating the access to the information objects in any possible way, as they can suffer from a limitation of usage of the knowledge they provide. For example, this may happen because of the language the objects are expressed in. In particular, this holds for virtual digital libraries, as they are built by aggregating information objects collected from different information sources. A translation functionality alleviate this drawback and improve the knowledge sharing.

Definition 2.2.29 (Translate) *The functionality enabling end-users to perceive an information object in a language different from the native one. In this context languages can range from country languages, e. g. Italian, English, to community and cultural languages, e. g. Muslim culture.*

Before visualising or asking for an information object translation it is needed to identify it. The "classic" functionality for discover digital library information objects are the search and the browse.

Definition 2.2.30 (Search) *The functionality enabling the users to discover the information objects, if any, capable to fulfil the information need expressed by a user, usually named "query".*

This definition of search functionality is general and does not constrain the forms this functionality may assume in a concrete digital library. These forms vary usually in the way through which the query is expressed. For instance, we can have digital libraries providing a simple keyword based search functionality (the Google [8] style) or providing what is usually called "advanced search" where more sophisticate conditions can be expressed, e. g. "all the information objects on a given research topic having a certain author and published in a period of time". As the objects the digital library deals with are of different types also the query content can be something different from a text. For instance it should be possible to search for

[8] `http://www.google.com`

information objects similar to a provided sample image representing the user information need as well as to search for those deemed similar to an excerpt of an audio. The form of the query does not constrain the type of object retrieved, e. g. a textual query can be used to retrieve information objects whose manifestation are videos or audio files. Finally, it is also worth noting that this functionality usually relies on the metadata associated to the information objects, in particular on the descriptive metadata (Def. 2.2.9).

The access mechanism envisaged in Def. 2.2.30, where the user expresses her/his information need and the digital library presents the information objects deemed as relevant, can be improved with more sophisticated features as in the case of the relevance feedback search.

Definition 2.2.31 (Relevance Feedback Search) *The search functionality which supports the iterative improvement of the search result set by allowing the user to express a relevance judgement on the retrieved objects at each iteration step.*

This mechanism has been proved to effectively improve the discovery mechanism and the user satisfaction because reduces the draw back related to the expressive power of the query language supported by the digital library.

As previously stated, the information objects populating the digital library information space can be expressed in different languages. In order to improve the discovery phase and make it effective, a cross-language search functionality is needed.

Definition 2.2.32 (Cross-language search) *The search functionality supporting the users in expressing queries in a certain language and retrieving information objects expressed in whatever other language.*

Many efforts exist in making this functionality an effective discovery mechanism [CLE].

Another "classic" discovery mechanism is represented by browsing functionality.

Definition 2.2.33 (Browse) *The functionality providing access to the digital information objects by listing them accordingly to a certain characteristic.*

The browse represents a functionality allowing the user to explore the digital library information space exhaustively. It may be considered a pre-search mechanism, aiming at finding information useful for searching. As in the case of the search, a digital library can be equipped with various types of browse functionality. For instance, it is possible to have a digital library information space depicted by using bubbles or areas of different size each representing a certain topic and then navigating among those bubbles in order to investigate on the content of each. Another common form of browsing is represented by the per-author browse, where the information space is explored for searching the correct form of the name of an author.

Moreover, in order to enhance the perception consumers have of the digital library and of its information space, the personalisation functionality becomes fundamental.

Definition 2.2.34 (Personalise) *The functionality allowing users to customise both the digital library information space and the way through which functionalities behave and are perceived.*

The personalise is a family of functionalities, as many aspects of a digital library can be customised to adapt to the content consumer needs. These aspects may range from the customisation of the look and feel digital library exhibits to the personalised organisation of the digital library information space so that it highlights the personal interest of its users. In particular two examples of personalisation of the digital library information space are related to the collection management and to the subscription functionality.

Definition 2.2.35 (Collection Management) *The functionality allowing users to create, update, and remove collections.*

The importance of collections as a mechanism to organise the digital library information space has been presented in Section 2.2.1 and in particular in Def. 2.2.13. The collection management functionality represents the family of functionalities needed to deal with collections. Thanks to this family of functionalities each content consumer is enabled to build her/his own virtual organisation of the digital library information space. This organisation may appear similar to the file system folder paradigm, with the difference, however, that it is a virtual one and evolves dynamically following the dynamism of the digital library. For instance, if a new document matching the definition criteria of a content consumer collection is added to the digital library this automatically becomes part of that collection.

Definition 2.2.36 (Subscription) *The functionality allowing users to express their interest in a certain topic.*

This functionality represents a personalisation functionality related to the dissemination of the digital library information objects (Def. 2.2.40). In particular the user expresses her/his interest in certain topics and each time new information objects about such topics become available into the digital library the digital library system alerts the user. Thanks to this characteristics digital libraries become proactive systems instead of being just passive systems in charge of replying to content consumer queries.

Until now we have introduced and described the functionality characterising the content creators and the content consumers. What actually differentiates a digital library from the Web is that in a digital library there exists a "control" on and the management of such resources by the librarians (Def. 2.2.21).

Definition 2.2.37 (DL Management) *The functionality acting on both the digital library information space and the digital library users for maintaining the digital library up and running and in line with the characteristics established by the DL Designer.*

Actually this functionality represents a class of functionalities because the management of a digital library is a complex task. The main functionalities have been depicted in Figure 2.7 on page 25. From this picture it arises that a librarian is in charge of performing functionality dealing with the management of information objects (annotate and update), with their publishing and advertising (publish, withdraw, and disseminate), with their maintenance (preserve), with the establishment of the rules regulating their usage (policy management), and with the management of users.

The "annotate information object" and "update information object" have already been described. In the context of the Librarian they represent mechanisms to co-operate with the content creators in enriching the digital library information space. For instance, annotations can be used to communicate to the content creator the improvement that must be performed on a submitted information object in order to make it publicly available into the digital library information space. The update information object instead can be used when minor changes of the information objects are needed, e. g. the descriptive metadata are incomplete or contains typos, or for other DL management reasons, e. g. the submitted format is not suitable for whole preservation and therefore is to be transformed in another, fully fledged format.

Definition 2.2.38 (Publish Information Object) *The DL management functionality allowing the users making the information object "publicly" available into the DL. With publicly here we intend that the object becomes available within a DL in accordance with the policies assigned to it.*

Definition 2.2.39 (Withdraw) *The DL management functionality allowing users to draw back an information object from the DL information space.*

The publishing and withdrawing of information objects from the digital library information space have been assigned to the librarians in order to highlight that the information objects made available by a digital library have been submitted to an evaluation process and thus fit with the rule regulating the quality of the information space. Thus a digital library guarantees certain characteristics on the content it offers. This process presents similarities with the traditional library scenario. However, the digital library may introduce novel paths in traditional library tasks, e. g. a digital library may defer this control to the content creator that is thus enabled to autonomously add its information object to the digital library information space.

The management of a digital library involves also the dissemination of its information objects.

Definition 2.2.40 (Disseminate Information Object) *The DL management functionality allowing digital library users to the advertise information objects available into the DL information space.*

This management functionality can have multiple manifestations, e. g. it can be performed by a human actor (the librarian) or it can be automatically done by the system (Recommender Systems [ACS, FFS+01, HCOC02]), can give notice of a new acquisition of information objects or of material deemed pertinent to a user profile (Def. 2.2.16), can be executed by advising all the DL Users or a subset of them, chosen by relying on the user profile preferences, can be performed by sending an email, an SMS or by any other messaging mechanisms.

Information objects represent the most important resource of any digital library and thus one of the functionality for digital library management is the preservation of such resources.

Definition 2.2.41 (Preserve) *The DL management functionality allowing librarians to extend the usable life of information objects and protect them from failure and technological obsolescence.*

For the sake of the reference model we do not add details about the technologies and the modalities to provide this functionality that can be found elsewhere [Gla06, PRE05, DEL]. It is worth noting that this functionality relies on preservation metadata (Def. 2.2.12) digital objects are equipped with.

Besides ensuring that information objects be available for the long term period, it is important to regulate their usage and prevent unauthorised accesses. In our model, this latter point is ensured by policies (Def. 2.2.22). As a consequence the digital library must provide the functionality for managing policies.

Definition 2.2.42 (Policy Management) *The DL management functionality allowing librarians to define and manage policies in order to regulate the usage of the digital library.*

It is worth noting that this functionality depends on the rules and constraints established by the DL Designer via the Establish Policy functionality (Def. 2.3.9).

This long list of functionalities concludes with those dedicated to the management of users, one of the important components of any digital library.

Definition 2.2.43 (User Management) *The DL management functionality allowing librarians to administer the pool of users having access to the digital library.*

This functionality represents a family of functionalities allowing the librarian to deal with the DL users management. In particular, the librarian must be enabled to create new users, remove already existing ones, and regulating the rights they hold into the system, i. e. the tasks they are entitled to perform and the information objects they are entitled to use. For the sake of the reference model we do not provide further details of this functionality but introduce two of the functionalities classified under this umbrella, the registration and the role management in order to explain and clarify two processes.

Definition 2.2.44 (Registration) *The user management functionality allowing the librarian to effectively adding a new user to those the digital library manage and recognise.*

This process is responsible for populating the digital library user community. Even if it can easily be automatised, it is preferable to be regulate it somehow. The less constraints are imposed on the registration of novel users, the less the system is capable to ensure the identity of a user. Moreover, the constraints imposed at registration time are a direct consequence of the audience the digital library is designed for. All these aspects are decided at the DL design time by the DL Designer (Section 2.3).

In order to provide users with rights enabling them to have access to the digital library information objects or use digital library functionality the librarians have to deal with the management of roles.

Definition 2.2.45 (Role Management) *The user management functionality for creating and administrating the associations between users and roles.*

As previously explained, our model relies on the Role-Based Access Control [FKC03] where the management of roles is of fundamental importance.

This section has introduced the main functionalities DL End-users expects from a digital library system. This list is not exhaustive, i. e. new functionalities can be envisaged in observing or designing a digital library system. However, the functionalities possibly missing are classifiable under one of the four main functionalities, i. e. access, content management, DL management, and personalise.

In the following section we introduce the last dimension completing the DL End-user model, i. e. the aspects related to the quality of service of the digital library.

2.2.4 Quality of Service

The quality of service dimension captures the qualitative characteristics of a digital library. In this section we present those of interest with respect to the DL End-users perspective. These quality characteristics are expressed by a specific metrics as suggested in [MS04]. Then before introducing the characteristics we introduce the concepts needed to describe and capture them.

As depicted in Figure 2.4 on page 17, the quality of service is composed by a set of *quality parameters* each having a *measurement* that can be *objective* or *subjective*.

Definition 2.2.46 (Quality Parameter) *An aspect of the quality of the service that we are going to express via a measurement.*

Examples of such parameters are reported in the following.

Definition 2.2.47 (Measurement) *The action of and the value obtained by measuring a quality parameter in accordance with a selected process and a unit of measurement.*

Measurement are further classified in objective and subjective measurement.

Definition 2.2.48 (Objective Measurement) *A measurement of a quality parameter obtained via a well defined process that does not depend on individual perception.*

Definition 2.2.49 (Subjective Measurement) *A measurement of a quality parameter based on or influenced by personal feelings, tastes, or opinions.*

The DL End-users are interested in having a perception of the following quality parameters: security, economic, usability, availability, reliability, performance, and response time. For each of them we provide a definition.

Definition 2.2.50 (Security) *The quality parameter which measures the level and kind of security of a given functionality.*

Definition 2.2.51 (Economic) *The quality parameter which measures the economic conditions of the usage of a given functionality.*

Definition 2.2.52 (Usability) *The quality parameter which measures the easiness of use of a given digital library functionality.*

Definition 2.2.53 (Availability) *The quality parameter which measures the probability that a functionality responds to a user request.*

Definition 2.2.54 (Reliability) *The quality parameter which measures the likelihood of successfully using a functionality, typically, it parallels availability.*

Definition 2.2.55 (Performance) *The quality parameter which measures the performance of the functionality from the users perspective.*

This parameter can be expressed by means of a series of other parameters, one of them is the response time.

Definition 2.2.56 (Response Time) *The quality parameter capturing the delay from the functionality request to the reception of the response.*

2.3 The DL Designer Perspective

In accordance with the three-tier framework, the DL Designer is in charge of defining and maintaining the digital library in line with the requirements of the DL End-users. In order to perform this task she/he interacts with the digital library management system that provides a set of digital library management functionality mainly related to the configuration of the digital library. It is worth noting the differences existing among these management functionalities and those performed by the librarians, as described in the previous section (Def. 2.2.37 on page 30). The DL management functionalities used by the librarians are provided by the digital library system and their goal is to daily operate a digital library while those used by the DL Designer are provided by the digital library management system and their goal is to define and manage the digital library as a whole, and in particular establish the processes and the rules the librarian must follow in operating the DL.

The second point to recall is that in accordance with the perspective hierarchy, the DL Designer inherits the concepts and relationships of the DL End-users and thus in this section we introduce only the new concepts and relationships needed to fulfil additional modelling needs. These concepts and relationships are depicted in Figure 2.8 and described below according to the five main DL concepts. Actually, as these users do not perceive the Architecture facet, the dimensions taken into account are the remaining four. It is also important to notice the concept of DLMS already described (Def. 2.1.3) and the relationship existing between this system and the digital library.

2.3.1 Information Space

The perception DL End-users and DL Designer have are exactly the same when expressed in terms of the information objects composing it and the collections it is organised in. In adjunction to this view the DL Designer needs to consider the *configuration aspects* she/he can acts on in order to arrange the digital library information space.

Definition 2.3.1 (Configuration Aspect) *A feature characterising the information space or the functionality that can be the object of a customisation.*

The customisation meant here is performed via the configure information space and configure DL functionality described in Section 2.3.3. The aspects of the information space that can be customised by the DL Designers are:

- the organisation and structure of the information space, i. e. the set of collections and their hierarchy;
- the set of third-party information sources from where the content is harvested;
- the DL supported information object formats (Def. 2.2.6);

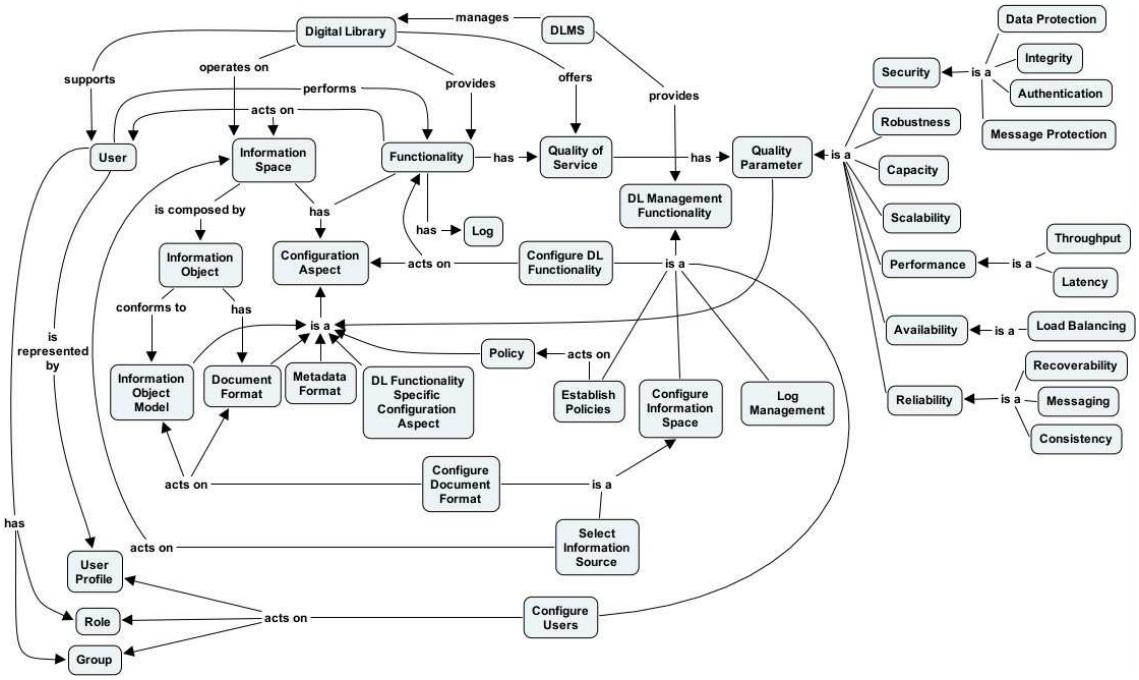

Figure 2.8: The DL Designer concept map – Main concepts

- the DL supported metadata formats (Def. 2.2.8).

In order to define the document format the following concept of information object model is required.

Definition 2.3.2 (Information Object Model) *The model characterising the class of all the possible information objects supported by the digital library system.*

This model presents the way trough which the concepts about information objects introduced in previous section, i. e. version, view, metadata, and manifestation, are modelled concretely into the digital library system. Thus the DL Designer, by appropriately instantiating this model is enabled to establish the document formats that are adopted by the digital library and perceived by its DL End-users. Examples of information object models are DoMDL [CCPS05a], the DSpace data model [TBS03a] and the Fedora Object Model [LPSW05]. A detailed description and a comparison of them is reported in Chapter 3.

2.3.2 User

As in the case of the information space, DL Designers and DL End-users are enabled to model the users with the same set of concepts and relationships among them. The only difference is the perception of these concepts, i. e. in the case of the DL End-users the concepts are the way through which concrete users can be perceived

and managed while in the case of the DL Designers these concepts are used to characterise the classes of users having access to the digital library. For instance, the DL End-users perceive the instance of the user profile as a representative of a human user while the DL Designer decides how the user profile instances are composed, the information they must provide, the process adopted to fill it with appropriate information, etc.

The aspects of the user dimension that can be customised by a DL Designer are:

- the user profile type (Def. 2.2.16) supported.
- the supported set of groups (Def. 2.2.17);
- the set of roles (Def. 2.2.18).

2.3.3 Functionality

DL Designers inherit the same perception of the digital library functionality, they need only to introduce the concepts related to the customisation, i. e. the configuration aspect, and the logging.

The definition of what a configuration aspect is has already been provided. It is worth noting that (*i*) each functionality has its own configuration aspects, (*ii*) some configuration aspects are known others depend on the functionality itself, (*iii*) configuration of a functionality may constrain the configuration of other DL functionalities, and (*iv*) each digital library system supports a set of configuration aspects regarding the functionality it provides. These reasons convinced us to highlight, in our model, the common configuration aspects, i. e. information object model, document format, metadata format, and policy, as well as to introduce the concept of *DL functionality specific configuration aspect*. Job of the DL Designer is to configure appropriately the digital library functionality, both in terms of the set of the provided functionalities and the behaviour of each of them, in order to fulfil the requirements arising from the audience the digital library is designed for. For instance, if the DL Designer is preparing a digital library for supporting information objects compliant with the document format F, she/he has to decide the type of functionality allowed on such an object and configure each functionality to deal with objects compliant with F.

Actually, the decisions taken by the DL Designer are concretely realised by the DL System Administrator via the deployment and configuration of the software components constituting the digital library system as described in Section 2.4. Moreover, this activity is nowadays conducted without any automatic support.

The model in Figure 2.8 contains other concepts that will be described in this area. In describing the principles characterising our model we stated that the functionality umbrella is used to model and characterise the processes the digital library provides in order to operate the digital library users and information objects. However, as the DL management functionality offered by the DLMS represents a type

of functionality we decided to describe and introduce it under the same umbrella of the digital library functionality because no confusion arises.

Definition 2.3.3 (DL Management) *The functionality acting on the digital library information space, the digital library users, and the digital library functionality for selecting and configuring the entities composing the digital library.*

This functionality represents the family of functionalities the DL Designer expect the DLMS provides in order to support the digital library definition and configuration tasks. In our model we envisaged five types of DL management functionality, i. e. *configure information space*, *configure DL functionality*, *configure users*, *establish policies*, and *log management*.

Definition 2.3.4 (Configure Information Space) *The DL management functionality provided by the digital library management system allowing the DL Designer to customise the digital library information space both in quantitative and qualitative terms.*

This functionality represents the family of functionalities supporting the various configuration aspects related to the digital library information space. As anticipated, they can be classified into two categories, those dealing with the quantitative aspects - like the number of information objects constituting it, e. g. *select information sources* - and those dealing with qualitative aspects like the typology of the information object perceived by the DL End-users, e. g. *configure document format*.

Definition 2.3.5 (Select Information Source) *The DL management functionality provided by the digital library management system allowing the DL Designer to select the information sources from which the digital library information objects are to be gathered.*

The implementation of this functionality can be various, depending on the typology of ingestion mechanism the digital library management system supports. For instance, if the digital library management system contains a module able to harvest metadata records via the OAI-PMH [OAI], the *select information source* allows specifying the url of the information source, possibly the set the information objects have to be taken from, and the start and end date. On the contrary, this functionality has to support a more advanced behaviour if the digital library management system provides a module supporting advanced mechanisms, for example a mechanism enabling a set of transformation rules to be applied in producing digital library information objects from those actually provided by the information source.

Definition 2.3.6 (Configure Document Model) *The DL management functionality provided by the digital library management system allowing the DL Designer to design the typologies of information objects the digital library has to deal with.*

As previously explained, this functionality supports the DL Designer in defining the document formats (Def. 2.2.6) in terms of the general information object model (Def. 2.3.2).

Definition 2.3.7 (Configure DL Functionality) *The DL management functionality provided by the digital library management system allowing the DL Designer to customise the digital library functionality both in quantitative and qualitative terms.*

This functionality represents the family of functionalities dealing with the customisation of the functionality the digital library provides. The possible implementations of this functionality depend on the characteristics of the digital library management system and in particular on the characteristics of the modules implementing the digital library system it provides. However, this functionality must allow identifying the set of functionalities the digital library has to provide among those allowed and to customise their behaviour in terms of the allowed configuration aspects. It is worth noting that the broader the range of customisations supported by a digital library management system is, the greater its capability to adapt to different scenarios is and hence its success in implementing digital libraries.

Definition 2.3.8 (Configure Users) *The DL management functionality provided by the digital library management system allowing the DL Designer to customise the digital library users both in quantitative and qualitative terms.*

This functionality represents the family of functionalities supporting the personalisation of the user related aspects. In particular, it is expected that by interacting with this functionality the DL Designer is enabled to define at least the composition of the user profile, the roles, and the groups the digital library supports.

Definition 2.3.9 (Establish Policies) *The DL management functionality provided by the digital library management system allowing the DL Designer to set up the family of rules regulating the usage of the digital library resources.*

This functionality is the highest level functionality with respect to the management of policies, i. e. all the other functionalities dealing with policies are constrained by the outcome and the choices of it. For instance, the policy management functionality performed by the librarians (Def. 2.2.42) takes care of assigning the policies in accordance with the constraints imposed by the *establish policies* functionality.

Another aspect introduced by the DL Designer perspective is the presence of *logs* in order to record the events occurring for each functionality. Thanks to them, the DL Designer is enabled to investigate on the usage of the various digital library functionalities and decide the appropriate customisation in order to improve the service offered or tailor it to the actual community the digital library is serving. The digital library management system must be equipped with a functionality allowing to manage such logs.

Definition 2.3.10 (Log Management) *The DL management functionality provided by the digital library management system allowing the DL Designer to manage the digital library functionality logs.*

2.3.4 Quality of Service

The DL Librarian is interested in additional and finer grained quality parameters with respect to those introduced for the DL End-users. In particular, as reported in Figure 2.8, new quality parameters are added, i. e. *robustness*, *capacity*, and *scalability*, and specialisation of already presented parameters are introduced, i. e. *integrity*, *authentication*, *message protection*, and *data protection* with respect to security, *throughput* and *latency* with respect to performance, *load balancing* with respect to availability, and *recoverability*, *messaging*, and *consistency* with respect to reliability.

Definition 2.3.11 (Robustness) *The quality parameter measuring the resilience of the functionality to ill-formed input and incorrect invocation sequences.*

Definition 2.3.12 (Capacity) *The quality parameter measuring the limit on the number of actions a functionality can perform.*

This characteristic influences also the availability and reliability qualities, i. e. when a functionality operates beyond its capacity these other qualities parameters are negatively affected.

Definition 2.3.13 (Scalability) *The quality parameter measuring the capability to increase the functionality capacity as needed.*

Definition 2.3.14 (Integrity) *The quality parameter measuring the ability of the functionality to prevent unauthorised access and preserve its data integrity.*

Definition 2.3.15 (Authentication) *The quality parameter capturing whether the functionality requires user authentication or it accepts anonymous users.*

Definition 2.3.16 (Data Protection) *The quality parameter measuring the capability of a functionality to prevent unauthorised access to the data the functionality is in charge for.*

Definition 2.3.17 (Message Protection) *The quality parameter measuring the capability of a functionality to protect the interaction with the users in terms of the messages exchanged.*

Definition 2.3.18 (Latency) *The quality parameter measuring the delay interval spent between the time when the user invokes the functionality and the time when the functionality effect begins.*

Definition 2.3.19 (Throughput) *The quality parameter measuring the rate at which a functionality sends and receives data, i. e. the number of requests it is capable to serve in a certain interval of time.*

The difference between throughput and latency is the following. The unit of latency is time. It measures the interval between the time a request leaves the client and the time the response arrives back at the client from the serving functionality. Throughput is the amount of data that is transferred over a period of time.

Definition 2.3.20 (Load Balancing) *The quality parameter measuring the capacity of a functionality to spread work between many resources.*

Definition 2.3.21 (Recoverability) *The quality parameter measuring the capacity of a functionality to recover from failures.*

Definition 2.3.22 (Messaging) *The quality parameter measuring the reliability of a functionality in terms of the likelihood of using the functionality by interacting via message exchange.*

Definition 2.3.23 (Consistency) *The quality parameter measuring the reliability of a functionality in terms of the likelihood of using the functionality avoiding contradictory results.*

2.4 The DL System Administrator Perspective

The DL System Administrator is the actor in charge to interact with the *digital library management system* in order to (*i*) select the *digital library system* software components to be installed in order to implement the digital library and (*ii*) decide where and how to deploy them in order to fulfil the requirements expressed by the DL Designer in terms of the information space, the user, the functionality, and the quality of service. Thus the implementation of the digital library is a matter of this actor.

Figure 2.9 reports the concepts and the relationships modelling DLs from these users' perspective clustered accordingly to the five main DL concepts. It is important to notice the main difference between this map and the previous ones; this map introduces the architecture concepts and focus on them and their relationships inheriting all the previous concepts with respect to user, information space, functionality and quality of service.

2.4.1 Information Space

All the aspects related to the digital library information space have already been introduced in the previous sections. As a consequence this model does not need to

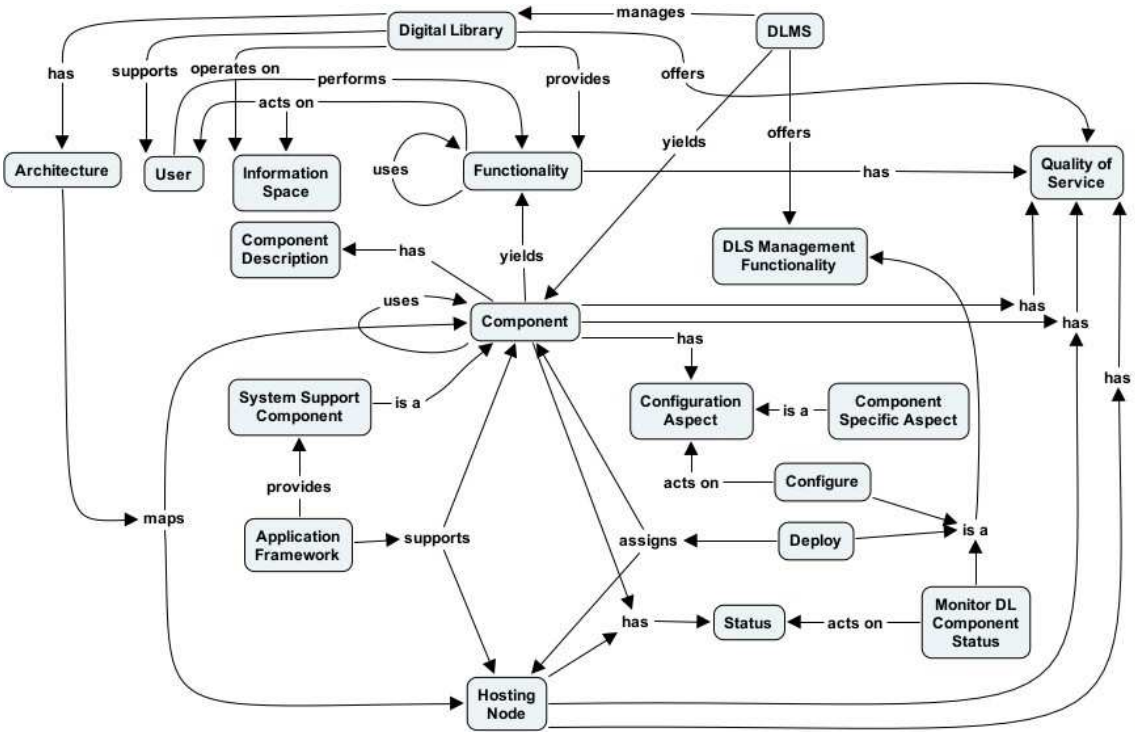

Figure 2.9: The DL System Administrator concept map – Main concepts

introduce any new concept. Here we want just to recall how the information space comes into play from the DL System Administrator point of view.

Since the first view of our model (Fig. 2.3 on page 13) we reported the concepts of information space and functionality modelling that a functionality acts on the DL information space. Moreover it should be clear that the DL information space is actually perceived by the DL End-users via the functionality the DL offers. From the DL System Administrator perspective we introduce the concept of *component* as the element holding and thus implementing a digital library functionality. Thus, the digital library information space is actually realised by the set of components appropriately configured and deployed to provide the digital library functionality over the DL information objects.

It is a matter of the DL System Administrator to map the requirements expressed by the DL Designer about the information space into component configurations or deployment strategies. For instance, the system administrator configures the document conformer component (Section 2.6) in order to produce the information objects the digital library must provide from the concrete information objects stored in third party information sources, as well as it is a matter of the administrator to configure the OAI-PMH Harvester (Section 2.6) in order to gather the information objects from already existing OAI compliant information sources.

2.4.2 User

With respect to the user dimension, the same considerations made for the information space are valid. This means that (*i*) in order to model the users from the DL System Administrator perspective the concepts previously presented are sufficient, and (*ii*) the user are perceived via the functionality acting on them and thus they are realised by the set of components in charge of providing such a functionality.

It is a matter of the DL System Administrator to map the requirements expressed by the DL Designer about the users into component configurations or deployment strategies. For instance, the system administrator configures the module in charge of managing the user profile with the DL Designer requirements and, as a consequence, configures all the other components constituting the digital library to deal with these pieces of information.

2.4.3 Architecture

As introduced in Section 2.1.2, the architecture is a representation of the system dealing with mapping functionality onto hardware and software components. Our model is based on such understanding and thus the characterising concepts are *component* and *hosting node*.

Definition 2.4.1 (Component) *A software module providing a well defined set of digital library functionality such that (i) it is autonomously configurable and (ii) it is deployable on one or more hosting nodes.*

It is also worth noting that each component has a component description characterising it and promoting the correct usage. This description may assume diverse forms ranging from human oriented description, e. g. a textual description in natural language, to a machine understandable one, e. g. the WSDL as in the case of web services. This description must be tailored on the needs of the DL System Administrator, e. g. in Chapter 6 we present a service in charge of partially replacing the system administrator and thus we need a component description usable by the implemented automatic reasoning algorithm. The second characteristic to highlight is the use relationship among components, as in a component based system components rely on others components to provide their functionality.

Definition 2.4.2 (Hosting Node) *An hardware device providing computational and storage capabilities such that it (i) is networked, (ii) is available, and (iii) is capable to host components.*

Components and hosting nodes are the building blocks of the digital library system. However, in order to allow them to operate as an application, the application framework is needed.

Definition 2.4.3 (Application Framework) *A set of library and subsystems supporting the operation of the digital library system components.*

The application framework influences the way through which components are realised as well as imposes constraints on the DL System Administrator that is in charge of also deploying and configuring the component constituting this framework. For instance, in the case the components have been realised by relying on an application framework providing a SOAP library, it is up to the DL System Administrator to provide them with such a library.

However, in order to model the library and the subsystems constituting the application framework as any other digital library component and thus leverage on the management functionality provided by the DLMS, we introduced the concept of system support component.

Definition 2.4.4 (System Support Component) *A software component providing digital library component support functionality.*

Having clarified that the DL System Administrator is in charge of managing the pool of components and the pool of hosting nodes constituting the instance of digital library system, it is necessary to introduce the concept of status.

Definition 2.4.5 (Status) *An information characterising the current standing of a component deployed on an hosting node.*

This information is fundamental in order to allow the DL System Administrator to monitor the status of the system supporting the digital library and to act for ensuring the characteristics and behaviour required by the DL Designer.

2.4.4 Functionality

In order to describe the digital library functionality the DL System Administrator relies on the same concepts and relationships already introduced for the DL Enduser and DL Designer perspectives. Moreover, from this point of view it is enough to know which software component has to provide a certain functionality and any other detail useful for appropriately tuning the component in order to obtain the expected behaviour and match the quality of service required.

Thus in the remaining part of this section we describe the DLMS expected functionality supporting the DL System Administrator tasks.

Definition 2.4.6 (DLS Management) *The functionality acting on the digital library system components for selecting, configuring and monitoring those composing the digital library.*

This functionality represents the family of functionalities a digital library management system must provide in order to allow the DL System Administrator to create and manage the digital library system instances in charge to provide the digital libraries required by the DL Designers. In particular, we have identified three main functionalities, respectively corresponding to the configuration, deployment, and monitoring phases of a digital library development process.

Definition 2.4.7 (Configure) *The functionality provided by the digital library management system allowing the DL System Administrator to customise a digital library system component.*

The model does not constrain the implementation of this functionality in a fixed form. For instance, it is possible to do the configuration of a component by manually editing configuration files as well as to envisage a graphical configuration environment driving the DL System Administrator during this complex task and capable to verify and maintain the consistency of the configured aspects. As in the case of functionality, the model takes care for introducing the concept of configuration aspect but does not present the list of allowed aspects for the following reason: each component may expose its own configuration aspects, some of them may be common to all the components (e. g. the component name, the communication protocol) while others are component specific (e. g. the metadata format to support, the document format to support); our model is comprehensive enough to represent all of them under the common umbrella.

The configuration of a single component is one of the aspects of customisation of a digital library system, the second aspect is represented by the deployment.

Definition 2.4.8 (Deploy) *The functionality provided by the digital library management system allowing the DL System Administrator to install a digital library system component on a hosting node and make it operative.*

As for configuration, the model does not constrain the implementation of this functionality. It is possible to build digital library management systems where this functionality is executed by the DL System Administrator by hand as well as to envisage sophisticated mechanisms supporting the dynamic deployment (an attempt to perform this dynamic deployment is under realisation in the context of the DILIGENT project as described in Chapter 6).

The deployment phase consists of assigning components to hosting nodes with the aim to ensure the quality of service parameter values required by the DL Designer. In order to evaluate the effectiveness of the allocation choices as well as to ensure the digital library operation, a monitoring functionality is needed.

Definition 2.4.9 (Monitor DL Component Status) *The functionality provided by the digital library management system allowing the DL System Administrator to perceive the current standing of a deployed digital library system component.*

Such functionality relies on the status information reporting the standing of a digital library component deployed on a hosting node. The implementation of it thus depends on the information made available and can be sophisticated as we like. For instance, a mechanism can be envisaged allowing the DL System Administrator to manually look for such information as well as graphical interface reporting the graphs of certain characteristics of the components or an automatic advertising mechanism alerting the DL System Administrator when certain characteristics of the deployed components exceed an established threshold.

2.4.5 Quality of Service

The DL System Administrator takes care of managing the components and the hosting nodes. As a consequence she/he inherits all the quality of service parameters previously introduced for the other user perspectives and needs to introduce new relationships that make it possible to assign such parameters to the entities she/he is responsible for. Thus in Figure 2.9 we assigned the quality of service characteristics to a component, an hosting node, and to the pair (component,hosting node) in order to capture aspects related to each of them. In particular, the quality of services assigned to a component supports the DL System Administrator during the component selection and configuration phase, those assigned to the hosting node support the deployment phase, and finally, those assigned to the pair supports the monitoring and maintenance phase.

2.5 The DL Application Developer Perspective

The DL Application Developers are the actors in charge to develop the components a digital library management system provides for building digital library systems realising digital libraries. They come in play both during the realisation of the DLMS and during the usage of the DLMS in building DLSs. In particular, during the first phase they are in charge to develop the "standard" components, i. e. components fulfilling the requirements of "classic" digital libraries, while during the second phase are in charge to develop personalised components, i. e. components fulfilling additional needs arising in specific application framework.

Due to this characteristic, the perspective of these users is focused on the technicalities needed to operate a digital library system realised in terms of components. Figure 2.10 presents the concept map to fulfil their perspective. It is manifest that these actors inherit a great amount of concepts from the previously presented perspectives and need just to add (*i*) a set of expected *enabling functionality* the components they are going to develop can rely on and (*ii*) the concept of *software framework* needed to provide them a blueprint of the digital library system they are going to introduce a new component or replace an existing one.

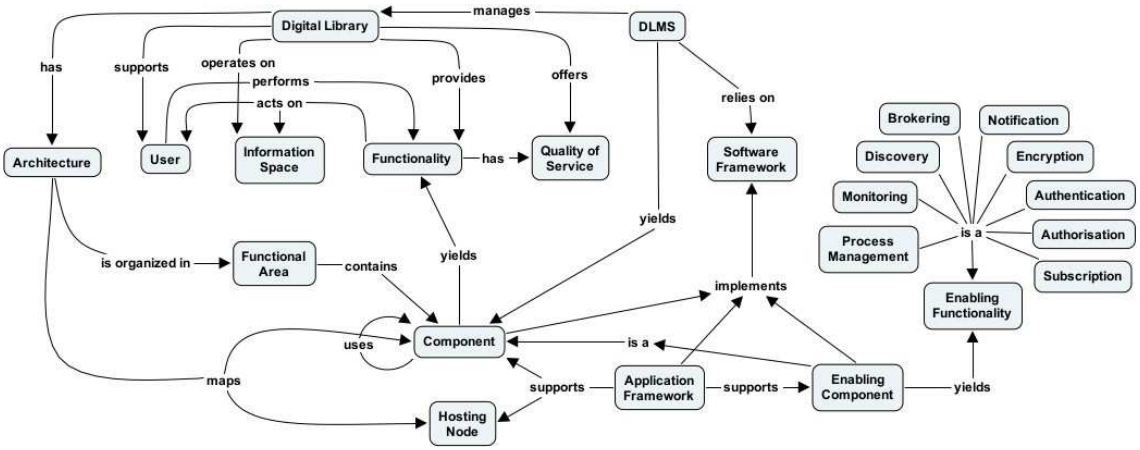

Figure 2.10: The DL Application Developer concept map – Main concepts

2.5.1 Information Space

For the sake of this model we do not need to introduce any further concept with respect to the information space. In fact, these actors are usually interested in realisation details of the concepts already introduced and these details, e. g. file, data structure, XML schema, are too fine grained with respect to the reference model; thus we decide to make them implicitly contained within the concept of the information space entity they are related to, e. g. these actors perceive the various metadata formats (Def. 2.2.8 on page 20) through the XML schemes defining them, an information object (Def. 2.2.1 on page 16) through the set of files constituting it, a collection (Def. 2.2.13 on page 21) through the metadata file defining it.

2.5.2 User

With respect to the user dimension the same considerations made for information space holds, i. e. the DL Application Developer needs to model exactly the same concepts and relationships previously introduced in terms of realisation details. For instance, from their perspective the user profile (Def. 2.2.16 on page 23) is both the entity representing a human actor as well as it is the data structure they can rely on for realising a functionality of the software component under development.

2.5.3 Architecture

The architecture dimension is of fundamental importance for the understanding of the digital library system and to allow DL Application Developers to produce software components capable to co-operate with the others system components in realising the expected functionality.

These users perceive the digital library architecture as in the case of the System Administrators, i. e. a mapping of the system functionality onto *components*

(Def. 2.4.1 on page 43) and *hosting nodes* (Def. 2.4.2 on page 43). Moreover, they must be aware of the presence of an *application framework* (Def. 2.4.3 on page 44) providing its supporting functionality via a set of components.

Actually, they must know the *software framework* underlying the family of digital library systems supported by the digital library management system they are interacting with and for which they are implementing a novel component.

Definition 2.5.1 (Software Framework) *A blueprint of the software system the digital library system is built accordingly.*

The software framework is a reusable design for a software system expressed in terms of abstract classes and explaining the way through which their instances cooperate [JF88, Deu89, Gac03].

This abstract and general picture of the system is realised by the application framework components, by the component constituting the digital library system in charge to provide the digital library functionality, and by the *enabling components*.

Definition 2.5.2 (Enabling Component) *A software component providing the enabling functionality a digital library system component rely on in interacting with others digital library system components.*

It is worth highlighting the differences existing between these enabling components and the system support components (Def. 2.4.4 on page 44). The former are components that are not associated with any specific resource but rather are global in nature, manage and support the interactions across collections of components. They are not built on top of, nor they are intended to replace or hide, the system support components; they are intended to exploit the capabilities of the application framework in order to extend it and cover the needs of the specific application context. In particular, they are in charge to provide the *enabling functionality* described in Section 2.5.4.

The last concept we decided to introduce is related to the organisation of components and named *functional area*.

Definition 2.5.3 (Functional Area) *A packaging of components in charge to provide a well defined set of functionality.*

This concepts become a foundational concept in presenting the digital library system reference architecture reported in Section 2.6.

2.5.4 Functionality

With respect to this DL main concept, the DL Application Developers inherit the same concepts already introduced even if with a diverse and more rich understanding. For instance, in the case of the search functionality defined in Section 2.2.3 (i)

DL End-users, DL Designers, DL System Administrators and DL Application Developers share the same functional understanding, (*ii*) DL System Administrators and DL Application Developers share the same architectural understanding, and (*iii*) only DL Application Developers must be aware of the complex process needed to provide such functionality.

Moreover, the presence of the enabling components introduce a family of novel functionality named *enabling functionality* the DL Application Developers can rely on in implementing their components, i. e. *process management*, *monitoring*, *discovery*, *brokering*, *notification*, *encryption*, *authentication*, *authorisation*, and *subscription*. It is important to remark that the actors profiting from the presence of such facilities are the digital library components when interacting with other components and that this communication is based on messages exchange, as usual.

Definition 2.5.4 (Process Management) *The functionality allowing to define and provide a novel functionality as composition of already existing functionality.*

This functionality is a family of functionality providing the *process* mechanism, i. e. a powerful and flexible way for building and delivering novel functionality by "simply" combining along novel paths already existing functionality also known as "programming in the large" or "mega programming" [WWC92]. This functionality provides support for their definition, management, verification, and execution. Thanks to it part of the logic needed to provide the component functionality the DL Application Developer is realising can be implemented by defining a process whose execution is up to the component providing process management facilities.

Definition 2.5.5 (Monitoring) *The functionality allowing to periodically and automatically check for certain events or conditions.*

This functionality provide the component with facilities for being automatically notified when the event the component expresses interest in happens.

Definition 2.5.6 (Discovery) *The functionality allowing to identify another component compliant with the specification of the request performed.*

This functionality is particularly useful in the case of dynamic systems whose components change or evolves without advices, and each time a component that needs to interact with another component must first identify the most appropriate among those available.

Definition 2.5.7 (Brokering) *The functionality supporting the component-to-component communication by hiding communication details.*

This functionality is useful in case of distributed and dynamic system allowing to easily the communication between components because take care of transforming the messages of the sender components in terms of messages the receiver can interpret by ensuring their delivery as well.

Definition 2.5.8 (Subscription) *The functionality allowing to express an interest with respect to an event in order to be automatically notified when the event happens.*

This functionality is paired with the notification.

Definition 2.5.9 (Notification) *The functionality alerting the subscriber entity when the event it is interested in happens.*

The remaining functionality are related to the security aspect in messages exchange.

Definition 2.5.10 (Authentication) *The functionality allowing to verify the validity of the provided sender identity.*

Definition 2.5.11 (Authorisation) *The functionality allowing to establish whether the sender is entitled to send the message.*

Actually, the message is the mechanism to use a functionality provided by the component and thus being entitled to send the message corresponds to being entitled to use the functionality.

Definition 2.5.12 (Encryption) *The functionality ensuring privacy in messages exchange, i. e. codifying appropriately the message in order to prevent its understanding by unauthorised entities.*

2.5.5 Quality of Service

With respect to the quality of service concept, the DL Application Developer does not need to add any further concept than those already provided. In particular, this actor perceive the quality of service parameters as a set of constraints and characteristics she/he must carefully take care when designing and implementing the component.

2.6 The Digital Library System Reference Architecture

In accordance with MacKenzie et. A. [MLM+06], a reference architecture is an architectural design pattern that indicates how an abstract set of mechanisms and relationships implements a predetermined set of requirements. We appreciate their explaining example on reference architecture for housing and therefore we report it in below in order to ease the understanding and to remark the importance of the role this entity plays in the digital library context as well.

> The role of a reference architecture for housing would be to identify abstract solutions to the problems of providing housing. A general pattern for housing, one that addresses the needs of its occupants in the sense of, say, noting that there are bedrooms, kitchens, hallways, and so on is a good basis for an abstract reference architecture. The concept of eating area is a reference model concept, a kitchen is a realisation of eating area in the context of the reference architecture.
>
> There may be more than one reference architecture that addresses how to design housing; for example, there may be a reference architecture to address the requirements for developing housing solutions in large apartment complexes, another to address suburban single family houses, and another for space stations. In the context of high density housing, there may not be a separate kitchen but rather a shared cooking space or even a communal kitchen used by many families.
>
> An actual – or concrete – architecture would introduce additional elements. It would incorporate particular architectural styles, particular arrangements of windows, construction materials to be used and so on. A blueprint of a particular house represents an instantiation of an architecture as it applies to a proposed or actually constructed dwelling.
>
> The reference model for housing is, therefore, at least three levels of abstraction away from a physical entity that can be lived in.[9]

In the light of this example it becomes clear that (*i*) the reference model introduced in previous sections provides a common conceptual framework that can be used consistently across and between different implementations and is of particular use in modelling specific solutions, (*ii*) the architecture of any digital library system is based on a component-oriented approach, and (*iii*) the software framework (Def. 2.5.1 on page 48) represents a particular instance of a digital library system.

In Figure 2.11 we present a digital library system reference architecture organised in functional areas (Def. 2.5.3 on page 48) that makes it possible to realise both "classic" digital libraries as well as the virtual digital libraries that are the subject of this dissertation. Thus, in designing this architecture our goal is twofold. On the one hand we aim at promoting the development of digital library systems not as single entities with a fixed business logic rather as entities composed by loosely coupled components each providing a set of functionality that (*i*) can be developed separately and independently, (*ii*) can be arranged in a variety of ways, (*iii*) can be reused in different domains, (*iv*) can easily be replaced by novel versions, and (*v*) provides help in dealing with heterogeneity issues since the adjunction of a component dealing with the heterogeneity allows the reuse of an already existing one. On the other hand we want to promote the implementation of an infrastructure

[9] The reference architecture for housing example is reproduced from [MLM+06]

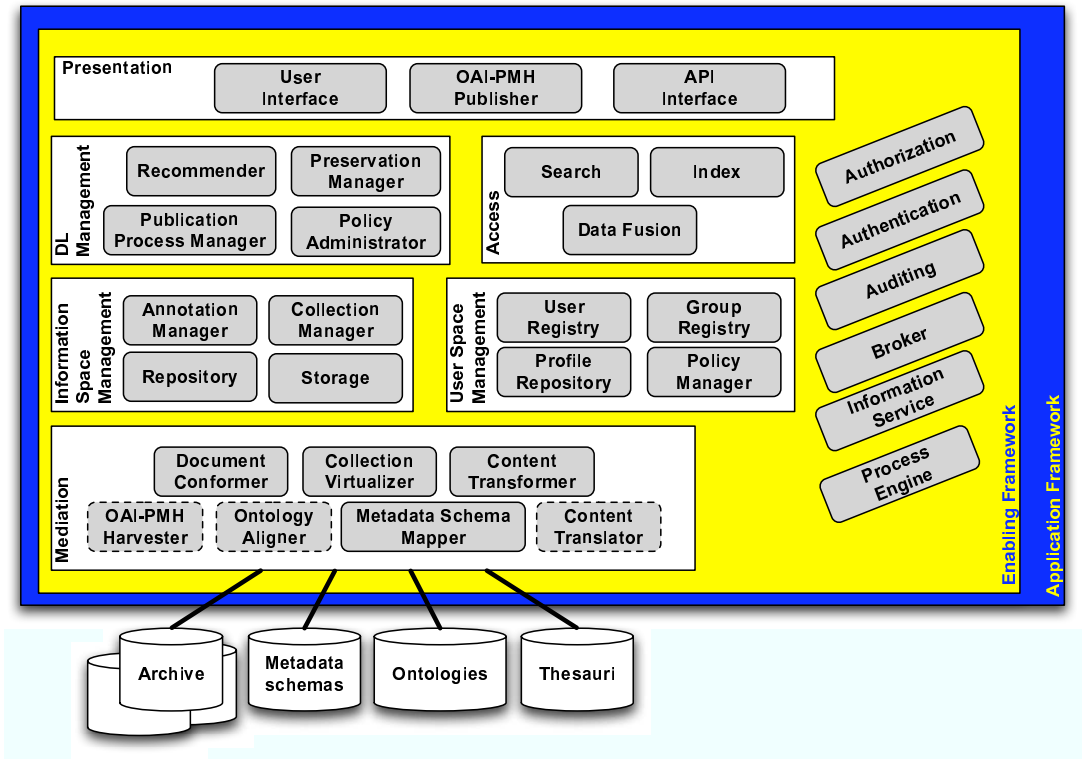

Figure 2.11: The Digital Library Systems Reference Architecture

for supporting the discovery and management of reusable components in order to build virtual digital libraries.

Before going in detail some clarifications are needed.

The first is related to the usage of a layered architecture. Actually, modern software architectures go ahead a mono dimensional layered vision even if the main achievements are maintained and exploited. In such architecture as well in in our reference architecture the use relationships may exist a priori among any subset of the components; the components can be combined in different ways to support different functionality; and the same components may be used in different ways, in accordance with the restrictions placed on their use and goal.

The second clarification is related to the components depicted into each functional area. For the sake of this dissertation we concentrate on those constituting the mediation area since they are of fundamental importance in implementing virtual digital libraries. For an exhaustive and full description please refers to [CCP06b][10].

The last clarification is related to the components constituting the enabling framework, i. e. the rightmost components that are outside the other functional

[10]The lack of the description of some components does not invalidate the presentation of the reference architecture nor the completeness of the dissertation since the functionality they are in charge to provide have been described in the previous sections.

areas. These components are the enabling components (Def. 2.5.2 on page 48) in charge of providing the functionality presented in Section 2.5.4.

In the following we introduce each of the identified functional area by briefly reporting the functionality they are in charge of providing.

2.6.1 The Presentation Area

The presentation area collects all the components that expose the digital library functionality to end-users and third-party applications as well. Thus this area represents the entry point for using digital libraries.

In considering the components constituting this area it must be taken into account that a comprehensive user interface able to cover all user needs despite their profile and their access device, the functionality, the information object types, formats, and media a digital library provides is not realisable yet. Thus this component must provide a strong customisation capability in order to be easily adapted to the diverse contexts.

Even if the user interface constitutes the main element of this area, the presentation is not limited to human oriented aspects, rather it includes as many public interfaces as required to improve the accessibility and the usability of the content managed by the digital library via the functionality offered.

2.6.2 The Access Area

The access area contains the components in charge of providing the access functionality (Def. 2.2.27 on page 27). In particular, in Figure 2.11 we have identified three components in charge of providing the search (Def. 2.2.30 on page 28) and the browse (Def. 2.2.33 on page 29) functionality.

The implementation of such components vary according to the characteristics of the digital library information space where to search in as well as to the DL End-user access requirements. An implementation capable to deal with distributed and heterogeneous information objects capable to support an enhance semantic matching between users queries and the retrieved information objects is presented in Chapter 5.

2.6.3 The DL Management Area

The DL management area groups the components in charge of providing and supporting the DL management functionality (Def. 2.2.37 on page 30).

In particular, the *Preservation Manager* is a component that, by exploiting the preservation facilities provided by the Repository components, allows users to perform or automatically perform preservation actions according to the digital library rules. The *Policy Administration* supports librarians in dealing with policy creation, assignment, and withdrawal. The *Publication Process Manager* component supports

the librarian in implementing the publishing process, i. e. the task ending with the publication of a new information object into the digital library information space. The *Recommender* component implements the disseminate functionality through which any type of information object is advertised to the users according to their profile. Recommendations can be directly managed by the librarian through this component as well as be automatically generated by the component itself.

2.6.4 The User Space Management Area

The user space management area contains the digital library system components in charge of supporting and providing the user management functionality (Def. 2.2.43 on page 32).

In particular, the *User Registry* component provides the mechanisms for dealing with user profiles compliant with a profile format. The *Group Registry* component provides the mechanisms for dealing with groups, e. g. their creation and the adjunction/withrawal of a user. The *Profile Repository* component supports the storage, maintenance, and retrieval of profiles manifestations compliant with one of the supported profile formats. The *Policy Manager* component provides the mechanisms dealing with roles and co-operates with the *Policy Administration* in implementing the digital library policy mechanism.

2.6.5 The Information Space Management Area

The information space management takes care of providing the functionality for dealing with the digital library information objects (Def. 2.2.1 on page 16), the most important resource of any digital library. In particular, these components cooperate in supporting the content management functionality (Def 2.2.23 on page 26) such as submission, updating, and annotation of information objects.

In particular, the *Storage* component provides the mechanisms and functions for the storage, maintenance and retrieval of manifestations. The *Repository* component provides the mechanisms to manage information objects compliant with a document format (Def. 2.2.6 on page 19). The *Annotation Manager* component supports the creation and management of annotations (Def. 2.2.14 on page 22). The *Collection Manager* component provides the mechanisms for supporting collections. In particular, an example of implementation of the latter component is given in Chapter 4.

The Open Archival Information System (OAIS) Reference Model [Con02] presents a comprehensive logical model describing all of the functions required in a digital repository. It outlines how digital objects can be prepared, submitted to an archive, stored for long periods, maintained, and retrieved as needed. In implementing the functionality of this area, and in particular the repository component, these recommendations must be carefully taken into account.

2.6.6 The Mediation Area

The goal of this dissertation is to provide a mechanism for building virtual digital libraries. One of the enabling factors is the capability to reuse already existing information objects. As a consequence, digital library systems need a set of components that makes it possible to abstract from the organisation and structuring of the concrete underlying information space, thus making external objects accessible as collections of virtual information objects tailored to the DL needs. The components constituting the mediation area are responsible for providing this functionality.

In particular the *Document Conformer* component, exploiting encoded knowledge of the structure of the documents whose manifestations are maintained by external sources, creates a representation matching the document model expected by the other DL components. An example of implementation of such component is provided in Chapter 3.

Other envisaged components needed to mainly support the document conformer are: (*i*) the *Content Transformer* that is capable to generate alternative manifestations of a given one, (*ii*) the *Content Translator* that is capable to generate a manifestation in a language different from the original one. Examples of such components are represented by the plug-ins provided by Greenstone [WBB01] system, the Fedora *disseminators*, and the OpenDLib *Adapters* described in Chapter 3.

Dealing with heterogeneous information sources implies also managing multiple metadata formats. This requirement is met through the *Metadata Schema Mapper* component that generates alternative metadata representations according to given metadata schemas, and through exploiting the features of the *Ontology Aligner* component that identifies and suggests semantic correspondences between the representational elements of heterogeneous ontologies. Both components rely on a set of mapping rules that can be injected as configuration parameters or be dynamically derived by the mediators that can express the same metadata into different schemas, or equivalent terms into different languages. An exploitation of such facilities in implementing a semantic based search service is described in Chapter 5.

Finally, the *Collection Virtualiser* component is the entity enabling other services to build virtual views, tailored on their needs, over the external information space. It also guarantees the correct management of the mapping between the internal information objects and the constituent items physically stored in external archives. An implementation of such component is introduced in Chapter 4.

2.7 Related Work

Despite the importance of formal theories and models as mechanisms to specify and understand complex systems in a clear and unambiguous fashion, Digital Libraries researchers and developers have preferred to have a pragmatic approach until now. Unfortunately, this approach introduces a series of drawbacks. Namely, DLs are de-

veloped from scratch, have scarce possibility of reuse, few possibility to interoperate, and difficulties to evolve in order to meet new requirements.

In our best knowledge, apart from the model presented in this dissertation, in literature there exists just one work that propose a formal model for DLs, i. e. the 5S framework that is described in the section 2.7.1. We describe it for highlighting the similarities with and the differences from our Reference Model, as well as for identifying the strength and the weakness of that framework with respect to it.

Another attempt to model digital libraries was made in the context of the DELOS working group on DL evaluation [11]. This is discussed in section 2.7.2.

2.7.1 The 5S Framework

The 5S framework [GFWK04, Gon04] is based on five fundamental abstraction, i. e. *Streams*, *Structures*, *Spaces*, *Scenarios*, and *Societies*, to define digital libraries rigourously and usefully.

These five concepts are informally defined as follows:

- Streams are sequences of elements of an arbitrary type (e. g. bits, characters, images) and thus they can model both static and dynamic content. Static streams correspond to an information content represented as basic elements, e. g. a simple text is a sequence of characters while a complex object like a book may be a stream of simple text and images. Dynamic streams are used to model any information flow and thus are important for representing any communication that take place in the digital library. Finally, streams are typed and the type is used to define their semantics and application area.

- Structures are the way through which parts of a whole are organised. In particular, they can be used to represent hypertexts and structured information objects, taxonomies, system connections, and user relationships.

- Spaces are sets of objects together with operations on those objects obeying to certain constraints. Despite the generality of this definition, this kind of construct is powerful and, as suggested, when a part of a DL cannot be well described using another of the 5S concepts, space may well be applicable. Document spaces are the key concepts in digital libraries. However, spaces are used in various contexts – e. g. indexing and visualising – and different types of spaces are used, e. g. measurable spaces, measure spaces, probability spaces, vector spaces, and topological spaces.

- Scenarios are sequences of events that may have parameters where events represent state transitions. The state is determined by the content in a specific location but the value and the location aren't further investigated because these aspects are system dependent. Thus scenario tells what happens to the

[11] `http://www.delos.info/WP7.html`

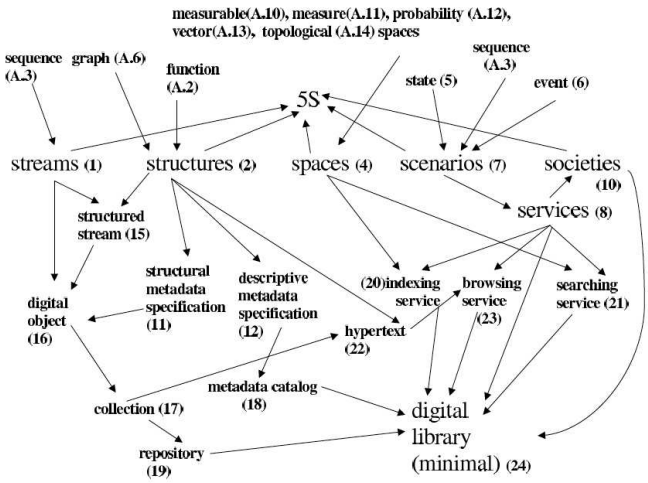

Figure 2.12: 5S – Map of formal definitions

streams in spaces and through the structures. When considered together, the scenarios describe the services, the activities, and the tasks representing the digital library functionality. Thus DL workflows and dataflows are examples of scenarios.

- Societies are sets of entities and relationships between them. The entities are both humans and software and hardware components, which either use or support digital library services. Thus society represent the highest-level concept of a digital library, which exists to serve the information needs of its societies and to describe the context of its use.

These concepts are of general purpose and very low level constructors. Based on them, Gonçalves et. Al. introduced the whole framework taking care to formally define the DL concepts reported in Figure 2.12[12]. In accordance to this framework, they define a Digital Library in the following way.

Definition 2.7.1 *A digital library is a 4-tuple ($\mathcal{R}$,Cat,Serv,Soc) where:*

- *$\mathcal{R}$ = (R,get,store,del) is a repository where $R \subset 2^C$ is a family of collections and get, store, and del are functions acting on them;*
- *$Cat = \{DM_{C_1}, DM_{C_2}, \ldots, DM_{C_k},\}$ is a set of metadata catalogs for all collections $\{C_1, C_2, \ldots, C_k,\}$ in the repository;*
- *Serv is a set of services containing at least services for indexing, searching, and browsing;*
- *Soc is a society.*

[12]Figure 2.12 is extracted from [Gon04]

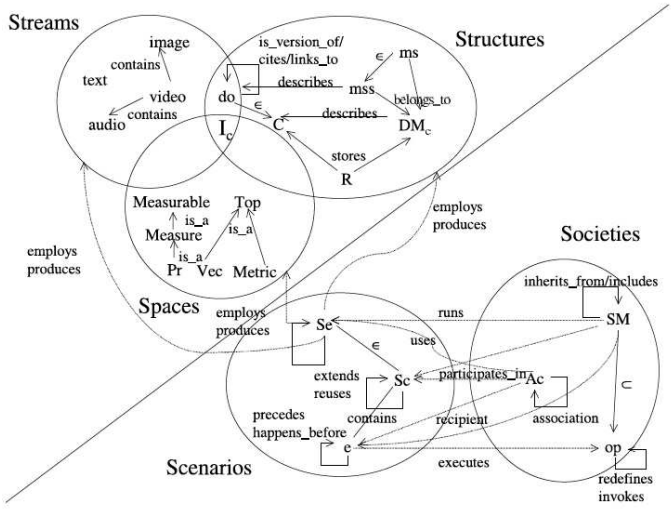

Figure 2.13: 5S – DL ontology

On top of this framework they proposed an ontology with the aim of arranging the concepts of the framework and identifying the relationships among them. Figure 2.13[13] presents graphically this ontology.

The main differences arising with respect to our Reference Model are:

- The Ss are very general purpose constructs and, in our opinion, result less immediate than a pragmatic approach as that proposed in our model. However, based on a mathematical formalisation of such constructors Gonçalves et. Al. have been able to provide a formal theory defining the identified concepts. Our approach is different, we have introduced and identified a broaden set of concepts and relationships among them and one of the future work is to define mathematically such concepts.

- In our framework we decided to provide different perspectives of the same entity, i. e. the digital library, because different users have diverse perceptions of this complex entities as commonly observed [FAFL95]. Moreover, we introduced and stressed the presence of systems realising the digital libraries.

- By relying on the concept of space, Gonçalves et. Al. introduced as first-class citizens probability spaces, vector spaces, topological spaces, etc. We considered such elements too fine grained with respect to the goal of our model and left them out.

- In the 5S Framework the modelling of services, the counterpart of our components, is provided in terms of scenarios and thus focuses on their behavioural

[13]Figure 2.13 is extracted from [Gon04].

description. Moreover, via the structure concept it is possible to model co-operating services. We consider such an approach valid for descriptive and modelling purposes, therefore we preferred to introduce the concept of reference architecture in order to suggest and constrain a pattern to follow in building digital library systems as well as we stressed the implementation and deployment details in order to being able to cover such concepts.

It is also important to notice the similarity existing between our concept of information object and the concept of digital object present into the 5S framework. This is to confirm that digital libraries objects have been more investigated and probably better understood than the other elements constituting such complex systems.

To conclude, the two models have the common goal to introduce a formalisation of the complex framework of the digital library world. While the 5S concentrates on the mathematical foundations and takes care of capturing behavioural aspects, our reference model derives from a pragmatic description of the digital libraries and takes care of introducing the systems needed to realise concretely such entities. Moreover, by providing different perspectives our model allows the consumers and the providers of such system content to share a common understanding of the system itself, thus promoting a fruitful co-operation.

2.7.2 The DELOS Classification and Evaluation Scheme

In the context of the DELOS Network of Excellence on Digital Libraries [DEL], the working groups dealing with the *evaluation of digital libraries* problem proposes a model [FHM+01] broader in scope than those usually adopted in their context with the aim to be able to satisfy the needs of all kinds of DL researchers, i. e. research community and traditional library community.

They starts from a general purpose definition of digital library and identify three non-orthogonal components within this *digital library domain*: the *users*, the *data/collection*, and the *system/technology* used as reported in Figure 2.14[14]. These entities are related and constrained by means of a series of relationships, namely (*i*) the definition of the set of users predefines the range and the content of the collection relevant and appropriate for them, (*ii*) the nature of the collection predefines the range of technologies that can be used, and (*iii*) the attractiveness of the collection content with respect to the user needs and the ease of use of the technologies by these users determines the extent of *usage* of the DL. By relying on these core concepts and relationships it is possible to move outwards to the DL Researcher domain and create a set of researcher requirements for a DL test bed.

Recently [TKP04], this model has been enriched by focusing on the inter-relationships between the basic concepts, i. e. the User-Content relationship is related to the *usefulness* aspects, the Content-System relationship is related to the *performance* attributes, while the User-System is related to *usability* aspects. For each of these

[14]Figure 2.14 is extracted from [FHM+01].

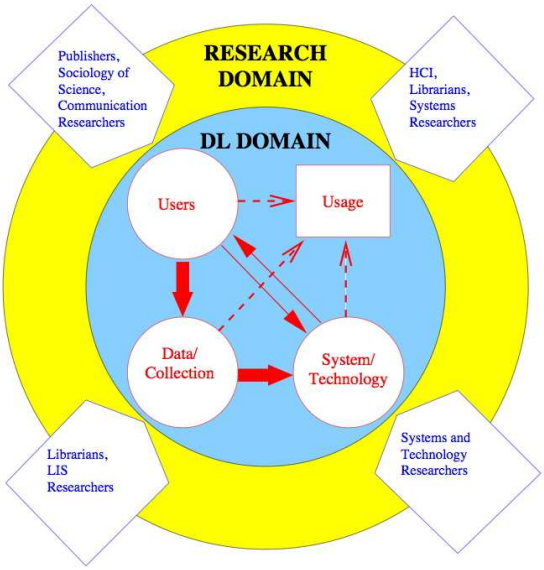

Figure 2.14: DELOS generalised schema for a Digital Library

three aspects they introduce techniques and principles for producing quantitative data and implementing their evaluation.

The comparison between this model and the Reference Model suffers from the different goal underlying them. In particular, the DELOS framework localises on introducing measures of the characteristics of the DL systems in order to evaluate them. On the contrary, the Reference Model introduces a great amount of concepts and relationships and is capable to support the evaluation via the quality of service dimension assigned to their functionality. Thanks to that our framework deals with different perspectives of the digital library entity we consider it able to integrate all the aspects covered by the DELOS model in a fine grained fashion. The problem of how to obtain measurements is out of the scope of the proposed reference model, however we have the construct to also express the "quality" of the quality parameter we are going to evaluate and thus provide an effective information about the observed value.

Chapter 3

Virtual Information Objects

A key aspect in building a Virtual Digital Library is to mediate between the representation of the information objects provided by the shared heterogeneous information sources and that manipulated by the DL services that implement the end-user functionality. This mediation may involve not only a change of the information object representation but also either splitting or aggregating them to produce either simpler or more complex new ones, respectively. This chapter presents an approach for supporting this mediation based on the introduction of a powerful document model, named *DoMDL*, and a *Repository* service capable of maintaining and disseminating DoMDL-compliant objects. The chapter also describes an implementation of such service in two concrete systems, i. e. OpenDLib [CP02, CP03] and OpenDLibG, a modified version of OpenDLib which exploits Grid technologies [FKT01, FKNT02] for broadening the types of managed information objects.

The chapter is organised as follows. Section 3.1 introduces the problems underlying the heterogeneous information space in the context of Virtual Digital Libraries and highlights the needs and the opportunities arising from the presence of virtual information objects. Section 3.2 presents the *Document Model for Digital Libraries (DoMDL)*, the model we propose to represent structured, multilingual and multimedia information objects whose parts may be dynamically generated or gathered from external sources. Section 3.3 introduces the architecture of a repository service capable of supporting the virtualization through DoMDL. Section 3.4 reports details on the implementation of the Repository service and of DoMDL in the context of the OpenDLib system. Moreover, it presents examples of the exploitation of the proposed document model, and related services, for fulfilling the desiderata of real community scenarios. Section 3.5 reports on the implementation of an enhanced version of the repository service capable to profit from grid facilities and shows an exploitation of such novel service. Finally, Section 3.6 discusses related research work by presenting the solutions implemented in concrete DL systems as well as comparing DoMDL with various standards for representing and exchanging information objects.

3.1 Dealing with Heterogeneous Information Objects

Digital libraries mediate between content providers and content consumers. Information objects gathered from the providers' different information sources may vary in their structure, format, media, and physical representation; they may be described by different metadata formats and their access may be regulated by different policies; they may either be physically copied from proprietary repositories into the DL own repositories or they may be accessed on demand following the link stored into the corresponding metadata records.

Content consumers want to search, retrieve, access and manipulate information objects semantically meaningful in their application domain. For instance, some communities may want to see the whole information space as composed of journals structured in articles, while others may want to work with information objects containing the papers produced by an author or with composite objects made by a text and all the images that illustrate that text. Such information objects may not correspond to the objects actually submitted to the DL nor to the objects maintained in the shared information sources, rather they may be virtual information objects created by reusing or by processing real objects or parts of them.

In order to fulfil this mediation role a digital library system must be equipped with mechanisms to transform the particular model of the information objects stored in the shared sources into the model required by the services that make them accessible to the DL users.

A key mechanism for supporting such mediation is the document model[1] that is supported by the DL system. Information objects collected from different sources are logically represented to and known by all the digital library services as objects compliant with this model. The services thus provide functionality that acts at the level of abstraction specified by the model. The selection of an appropriate model is therefore of key importance in determining which level of virtualisation can be reached by the informations objects through a DL system. The next section presents DoMDL, the document model we propose for covering this important role.

3.2 The Document Model for Digital Library

The *Document Model for Digital Library* (DoMDL) has been designed to represent structured, multilingual and multimedia information objects and can be customised according to the DL content to be handled. For example, it can be used to describe a lecture as the composition of the teacher presentation together with the slides, the video recording and the summary of the talk transcript. However, the same lecture

[1]We use the term "document model" for historical reasons. In the reference Model introduced in Section 2 this model is named "Information Object Model".

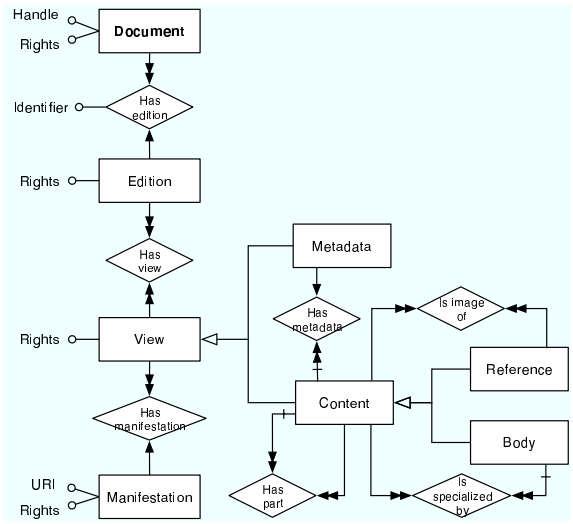

Figure 3.1: DoMDL – Document Model for Digital Library

can be disseminated as the MPEG3 format of the video or the SMIL document synchronising its parts.

In order to be able to represent information objects with completely different structures DoMDL distinguishes four main aspects of document modelling, in accordance to the reference model introduced in Chapter 2. These aspects are represented, using terms and definitions very similar to the IFLA FRBR model [IFL], through the following entities: Document, Edition, View, and Manifestation (see Figure 3.1).

The *Document* entity, representing the information object as a distinct intellectual creation, captures the more general aspect of it. For example, a book such as "Digital Libraries and Electronic Publishing" by W. Arms or a lecture such as "Introduction to Mixed Media Digital Libraries", by C. Lagoze, can all be modelled as Document entities. Each entity of this type is identified via the *Handle* attribute.

The *Edition* entity, representing a specific expression of a distinct intellectual creation, models an information object instance along the time dimension. The preliminary and draft version of a paper, its version submitted to a conference, and its version published in the conference proceedings, are all examples of editions of the same information object. Any Edition is related to the appropriate Document with an *Identifier* whose value is linear and numbered.

The *View* entity, modelling a specific intellectual expression, is the way through which an edition is perceived. A view excludes physical aspects that are not related to how a document is to be perceived. For example, the original edition of the proceedings of a workshop might be disseminated under three different views: (*i*) a "structured textual view" containing a "Preface" created by the conference chairs, and the list of thematic sessions containing the accepted papers, (*ii*) a "presentation view", containing the list of the ppt slides used in the presentations, and (*iii*) a

"metadata view", containing a structured description of the proceedings.

The *Manifestation* entity models the physical formats by which an information object is disseminated. Examples of manifestations are: the MPEG file containing the video recording of a lecture made at a certain summer school, the AVI file of the same video, the postscript file of another lecture given at the same school, etc. Physical formats are accessible via *URIs*, used to associate local or networked file locations.

These entities are semantically connected by a set of relationships. The relationships *Has edition*, *Has view*, and *Has manifestation* link the different aspects of an information object. Note that these relationships are multiple, i. e. there can be several objects in the range associated with the same object in the domain. This means that there can be multiple editions of the same information object, multiple views of the same edition and multiple manifestations of the same view.

The View entity is further specialised in two sub-entities: *Metadata* and *Content*. The former allows a document edition be perceived through the conceptualisation given by its metadata representations. These may be a flat list of pairs (fields, values), as in the Dublin Core metadata records [Dub], or more complex conceptual structures, such as in the IFLA-FRBR records. Typically, this metadata view is indexed to support attribute-based querying and browsing operations, but it may otherwise be used. For example, it may be disseminated free of charge while the document contents are regulated by fee access, or disseminated on a mobile device. By using the *Has metadata* relationship it is possible to model the fact that also content views can be described by one or more metadata records in different formats.

The Content view has two sub-entities: *Body*, and *Reference*. The former is a view of the information object content when it is to be perceived either as a whole or as an aggregation of other views. For example, a textual view of the proceedings a workshop is built as the aggregation of the textual views of its component articles. The relationship *Has part* links a Body view with its component views. A Body view may be specialised by other views that represent more detailed perceptions of the same content. For example, an article of the cited proceedings may be specialised by two views related to the French and English version of that document, respectively. A view is related to all its specialisation through the relationship *Is specialised by*.

The Reference entity represents a view that does not have associated manifestations because it is linked with an already registered manifestation. This entity has been introduced to represent the relationship between views of different information objects editions. Articles presented at the same workshop, for example, can be modelled as single information objects and grouped together by the workshop proceedings information object that contains only the references to them. It is worth noting that this entity, bringing together parts of real or virtual information objects, makes it possible to manage virtual objects that are not explicitly maintained by the DL storage system. For example complex reports, or training lectures, can easily be modelled as composition of parts extracted from real objects. A reference view is linked with another view via the relationship *Is image of*.

Each of the entities described above has a set of attributes that specify the rights on the modelled information object aspects. This makes it possible, for example, to model possibly different rights on different editions, different access policies on different views or on different parts of the same view, and so on.

3.3 Virtualization through DoMDL

From the architectural point of view, the virtualization of information objects requires the introduction of a number of components. Figure 3.2 presents the main modules constituting the internal architecture of a Repository service capable of supporting such a virtualization.

The main module is represented by the *Document Manager* that is in charge of mapping the operations expressed in terms of DoMDL objects (i.e., objects supported by the document model) into operations on the storage back-end or on the *Document Conformer*. This component is a type of Information Mediator and plays two roles: it contains a module in charge of injecting external information objects (acquired via either harvesting or ad-hod developed wrappers) into the Repository by representing them as objects compliant with the supported model. Moreover, this component is equipped with a set of *Content Transformers* that are capable to generate alternative manifestations. It is worth noting that (*i*) the injection process is guided by a set of configuration parameters, i. e. the way through which the external information objects are mapped into common model objects is customisable, and (*ii*) alternative manifestation can be generated at publishing time, i. e. when the information object is created into the repository and thus the new manifestation is stored, or at access time, i. e. when the particular manifestation is accessed via the API.

3.4 The OpenDLib Implementation

The DoMDL model and the above described architectural components have been successfully validated by implementing them in the OpenDLib system [CP02, CP03]. OpenDLib has been, at our knowledge, the first system to offer the functionality of DLMS. It supports the creation and maintenance of very different DLs thanks to its high capability of being customised and to the adoption of the DoMDL document model. In particular, by exploiting the features of this model it is capable of handling digital information objects that not only may be the analogous of reports, books, journals, videos, archival records, but can also consist of scientific data, programs and any other kind of multimedia documents the user communities may consider as appropriate instruments for supporting their communication. OpenDLib has been designed to interoperate with existing archives and digital libraries: it supports the management of external resources located anywhere, e.g. papers maintained by

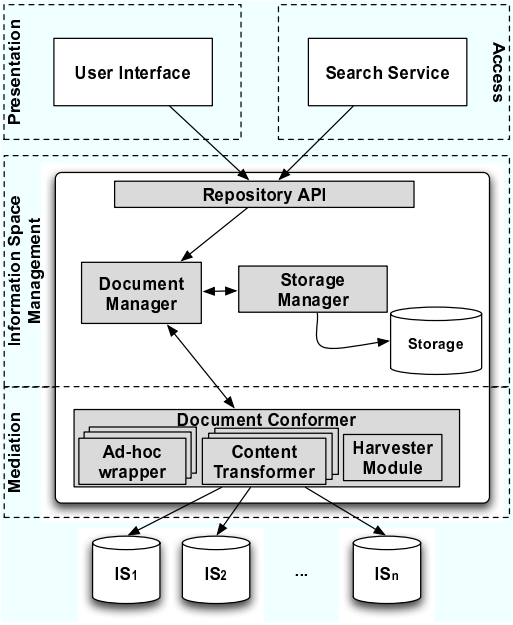

Figure 3.2: The Repository architecture

their authors on personal web pages; the harvesting of document representations, e.g. archive records, and of their manifestations through the Open Archive Initiative protocol; the loading of documents stored on local file-system or FTP servers, e.g. documents not yet published world wide. A more detailed description of this system is reported in Appendix A.

Supporting DoMDL in OpenDLib has brought a number of benefits but it has also implied a number of design and development challenges not only in the implementation of the Repository Service but also in the realisation of the other functions that exploit the richness of DoMDL. In the rest of this section, we first describe the OpenDLib representation of DoMDL and then analyse and describe how the introduction of this model impacts on the realisation of the following functionality: (*i*) information object storage, (*ii*) information object discovery, (*iii*) information object access, and (*iv*) information object visualisation. At the end of the section we also briefly illustrate two OpenDLib DLs serving different application areas by showing which level of virtualization has been obtained.

3.4.1 DoMDL Representation

The representation of a complex object compliant with the proposed document model usually deals with two issues: (*i*) the description of the internal relations among information object entities, and (*ii*) the management of the related physical parts of each entity.

The OpenDLib solution to these problems is to decouple the definition of the

document model instance from its real data. With this approach an information object is really composed by several files. The instance of the document model for a given information object is described in a separate file, named *Structure file*, which is the only mandatory element that must be provided. The goal of this file is to explain the composition and the relations among the other files that compose the document. Thus, the repository file is the concrete placeholder the Repository maintains for each virtual information object.

The natural way to express such structured data is through an XML document. Therefore, a major design issue was to define an appropriate XML Schema, capable to cover all the DoMDL features. XML Schemas provide a standard means to specify which elements may occur in an XML document and in which order, and to constrain certain aspects of these elements. The result of this effort is the DoMDL XML Schema[2]. An XML document validated against this Schema describes a particular edition of a document; main entities (views and manifestations) belonging to an information object are represented with tags while relationships among them are expressed by nesting these tags. As well, a number of attributes on the entity tags allows their type and the related behaviour be specified. In this way, a Structure file can put together different physical components to form an unique and coherent structured information object. Different editions of the same information object are not physically linked together, rather they are logically grouped by the storage model in order to obtain a higher flexibility of the system. The storage model, in fact, is able to manage editions as a single entity since they share the same document identifier.

Finally, according to the document model specification, it is also possible to express a set of rules that regulate the rights on the information object views via the properties child tag; in this implementation the rights to download, deliver, transcode or display a view may be, or not be, granted.

3.4.2 Information Object Storage

At the hearth of any digital library system, as in any information system, there is its storage capability and the related model, that is how the system maintains the information it needs. Here we do not argue about the physical storage manager implemented by OpenDLib since, traditionally, a storage model decouples the document model adopted from the underlying technologies used to store documents. Rather we present both the constraints and opportunities that the utilisation of DoMDL has introduced in the system. Some of them, namely those that conducted us to investigate Grid technologies, are presented in Section 3.5.

Primarily, according to the DoMDL specification, the storage model must be able to manage multiple metadata formats for the same information object and multiple physical manifestations for the same view of an information object. This allows

[2] `http://www.opendlib.com/resources/schemas/domdl.xsd`

OpenDLib to be enriched with the capability to: (*i*) automatically move from one metadata format to another one by using appropriate XSLT stylesheets, and (*ii*) automatically migrate from one physical manifestation (e. g. a pdf file) to another one using a transformation procedure provided by the system or configuring the system to use a third party tool.

Among the others, two major advantages rising from these capabilities are (*i*) to make it easy to create new information objects for populating new digital libraries starting from existing and heterogeneous information sources and (*ii*) to preserve documents from the technological obsolescence. Regarding manifestations, they are identified by URIs. As a consequence, a manifestation can be stored inside or outside the system, depending on the time in which the URI is dereferenced.

When a new information object is created, a number of solutions are offered in order to support a range of different needs. In fact, a manifestations can be: (*i*) directly uploaded with the document, (*ii*) automatically retrieved from an external location and locally stored, (*iii*) maintained as an external manifestation and dynamically retrieved at the access time, or (*iv*) maintained as an external manifestation and displayed through its original location at the access time.

These options are made available by properly combining the values of the attributes of the manifestation tags in the document Structure file. The combination of these options at the information object level makes it possible to build new structured documents that enrich the original ones by aggregating multiple parts of different objects from different heterogeneous information sources. Moreover, these choices promote the optimal utilisation of the storage resources. For instance, if a manifestation requires too many storage resources to be stored internally, it can simply be referred to its external original location. This optimisation is also supported by the reference view mechanism. Following the model specification, a view can be a reference to another view of a different document; by implementing this mechanism, data duplication can be avoided.

The last advantage we mention here is the possibility to submit and manage documents that are modelled in very different fashions in the same OpenDLib instance, if they are compliant with the DoMDL XML Schema. This introduces a high level of flexibility and promotes a full integration among heterogeneous information sources with different types of documents or metadata.

Finally, let us mention addressability. The basic addressable unit is the single manifestation. Moreover, the list of all views or manifestations as well as the list of editions of a information object can also be addressed.

3.4.3 Information Object Access

The access granularity, i. e. how an information object or its components can be accessed, is closely tied to the storage model. Possible options include: (*i*) to expose data according to the document model representation, and (*ii*) to hide the representation and provide an interface to query the model in order to obtain the document

parts.

OpenDLib implements both solutions by providing direct access to the Structure file and also an interface to query a given information object. This design choice allows users (either humans or services) to select the option that fulfils their needs at best. For instance, to speed up the operations, other OpenDLib modules retrieve from the storage subsystem the Structure files and then manage the corresponding information objects. Other services should instead be interested in requiring the information object entities (e. g. all the editions of a document, all the manifestations, etc.) in order to manipulate and rearrange them independently from the DoMDL representation.

3.4.4 Information Object Discovering

Information object discovering is a crucial component of any distributed digital library system. This feature is usually achieved through indexing and search services. OpenDLib provides these functionalities both on the information object metadata and, when possible, on the information object themselves (full-text indexing) via its search subsystem. The adoption of DoMDL had a great impact during the design of this subsystem because information objects can be expressed in any format and thus no assumptions could be made about the presence of any field or structure of the indexed information. The result is a highly customisable search subsystem based on: (*i*) a complete configuration of any index concerning the metadata or manifestation format, of the elements to be indexed and the set of elements to be returned after a query, (*ii*) an abstraction layer between the query engine and the format-independent query language supported, and (*iii*) inspection mechanisms that support the discovery of which indexed format, which query operators and which result sets are supported by a particular instance of an index. Therefore, thanks to the document model, an OpenDLib instance can have multiple indexes able to index any format independently of their number or location. Also the graphical user interface provided to interact with the search subsystem has the capability to configure itself, depending on which index it currently interacts with, by automatically adding, removing or changing both its components and its look and feel. In addition, the search subsystem offers the very new possibility to execute queries across documents handled by different information sources and expressed in different formats.

3.4.5 Information Object Visualisation

The visualisation of information objects is the last main issue strictly related with the document model. DoMDL gives a great number of opportunities for the presentation of complex and structured objects. For instance, it allows information object visualisation be personalised by deciding who has the rights to view what.

OpenDLib provides two kinds of information object visualisation, one tab-based (Figure 3.3) and one window-based (Figure 3.4), both able to display information

Figure 3.3: DoMDL tab-based visualisation

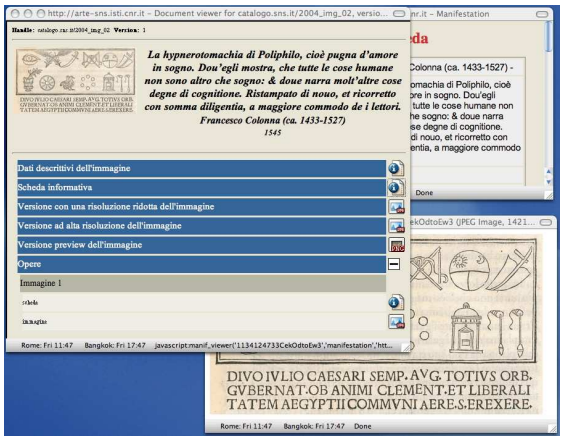

Figure 3.4: DoMDL window-based visualisation

objects compliant with the DoMDL model. In either mode, a graphical rendering of the document structure is visualised and manifestations are retrieved on demand next to the user requests. As well, OpenDLib can easily be extended with additional visualisation features; this is specially useful for OpenDLib instances that manage classes of documents with the same structure, e. g. papers or talks, to better exploit the specific structure of those documents. The mechanisms above make it possible to present the same document in different ways by making the concept of *virtual document* concrete.

3.4.6 DELOS Exploitation

The first example we report is extracted from the DELOS DL[3]. This DL handles documents published by the homonymous Network of Excellence on Digital Libraries. It stores, maintains, and disseminates, among the others, the proceedings of several DELOS events like the ECDL conferences, a number of thematic and brainstorming workshops, and the documents of the international summer schools.

These documents are characterised by a large number of inter-relationships that are emphasised to improve the accessibility and readability of semantically related information objects.

Figure 3.5 depicts a typical edition of an ARTICLE maintained in this DL. Each edition has the following views: *Metadata*, *Abstract*, and *Content* which are related to manifestations in different formats; *Related Talk*, which links with the presentation of the article made by its author during the related event; and *In-Proceedings*, which links with the document that represents the proceedings where the article has been published. Reference views are also used to link a TALK document with the content of the edition of the respective article. It is also important

[3]DELOS Digital Library Web site `http://delos-dl.isti.cnr.it`

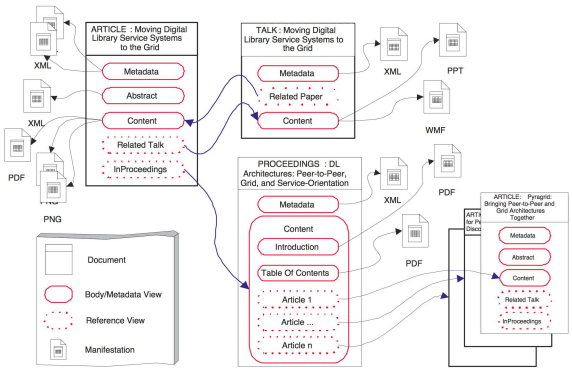

Figure 3.5: DELOS Digital Library documents

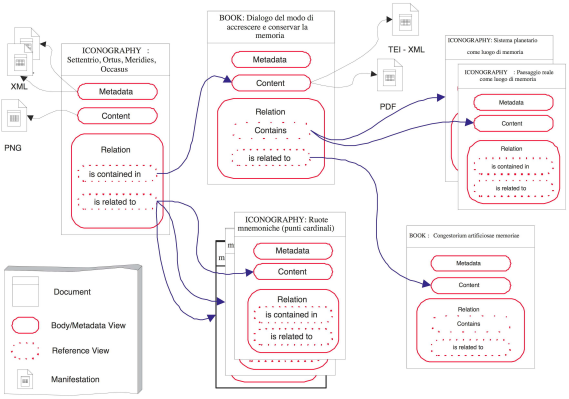

Figure 3.6: ARTE Digital Library Documents

to point out that in the DELOS DL different metadata formats are used to represent the description of an article, that multiple manifestations in different formats are associated with the same view, and that the video manifestations are stored on video streaming servers able to improve their fruition. Finally, we highlight that the end-user perceives an intellectual creation via the homogeneous and coherent presentation of a virtual information object that, instead, is obtained collecting parts of different and heterogeneous stored information objects.

3.4.7 ARTE Exploitation

The second example is extracted from the ARTE DL[4]. This DL stores, maintains, and disseminates the digitised versions of ancient texts and images linked by relationships that express semantic associations among them, such as the *contains*, *is contained in*, *is related to*, and *has authored by* relationships. The original documents are collected from very heterogeneous information sources, that (*i*) range from different types of database to file-system based storage systems, and (*ii*) are

[4]ARTE Digital Library Web site `http://arte-sns.isti.cnr.it`

based on proprietary content representations. A typical edition of an ICONOGRAPHY is represented by its metadata and related picture. The value added by using DoMDL is perceived by analysing the information objects relationships. In fact, using reference views it has been possible to model virtual information objects allowing end-users to navigate the relationship from an iconographic object to the book that contains it, analyze the textual part before and after the mentioned picture, browse the book to see other similar information objects, and also immediately access the other related iconographic objects.

Finally, it is important to note other specific characteristics of the documents managed by this digital library, namely: the wide heterogeneity of the representation and description formats; the existence of access policies regarding many parts of the documents; and the variety of new documents that are created by the users of the digital library by composing parts of existing documents.

3.5 The OpenDLibG Implementation

DoMDL has also been used as a key element in OpenDLibG, an enhanced version of the OpenDLib that is capable of exploiting the storage and processing capabilities offered by a Grid infrastructure[5]. Thanks to this new feature OpenDLibG can manage documents requiring huge storage capacities, like particular types of images, videos, and 3D-objects, and can also support on-demand creation of them as the result of a computational intensive elaboration on a dynamic set of data.

The main extension introduced in OpenDLibG with respect to OpenDLib is a new Repository service, named *Repository++*, described in the next section.

3.5.1 Repository++

Repository++ acts as a *virtual repository*, i. e. it is capable of the same operations as those required to store and access information objects but its logic is completely different because it does not store any content locally but it relies on the storage facilities provided by both the OpenDLib Repository and the gLite infrastructure via the *gLite SE broker*, i. e. a service interfacing with a gLite [EGEb] based grid infrastructure in order to store and access files.

In designing this component we decided to make it configurable with respect to the strategy to be adopted in distributing content on the two kinds of storage systems.

The configuration aspects exploit the DoMDL management functionality that allows any supported manifestation type to be associated with a predefined workflow tailored to deliver customised storage, access, and retrieve capabilities. Thanks to this characteristic it is possible to design and implement the most appropriate

[5]This work is partially funded by the European Commission in the context of the DILIGENT project, under the 2nd call of FP6 IST priority.

processes for each new type of raw data managed by the DL and easily associate these processes with the related manifestation type without taking care of the logic of the Repository itself. In the current version, one of these workflows has been implemented. It allows storing, accessing, and retrieving files maintained in storage elements accessible through the described gLite wrappers. For instance, it is possible to configure the Repository++ service in order to maintain all metadata manifestations on a specific OpenDLib Repository instance, a certain manifestation type on another OpenDLib Repository, while data and other manifestations accessed less frequently and requiring a huge amount of storage can be stored, with the help of the gLite SE broker, on a storage element provided by the gLite Infrastructure. The characteristics of the content to be stored should drive the designer in making the configuration. Usually, manifestations that require to be frequently accessed, or that need to be maintained under the physical control of a specific library institution, should be stored on standard OpenDLib Repository services. On the contrary, content returned by processes, that is either not directly usable by the end-user, or that can be freely distributed on third-party storage devices should be stored on gLite storage elements.

Cryptography capabilities are under investigation to mitigate the problems mostly related to the copyright management for storing content on third-party devices. The envisaged mechanism is based on splitting the file in parts and encrypting them with a standard key based encryption algorithm. The key consists of 64 binary digits and it is preserved on the Repository++ service. In this way, any single part of the file is protected and, anyhow, it is not meaningful without the other parts. Obviously, in this case the Repository++ service must collect all the file parts, decrypt them, and reconstruct the whole file before responding to a request for accessing a manifestation.

Another important feature added to the enhanced repository is the capability of associating a job or a DAG of jobs with a manifestation[6]. This feature allows the management of a new type of document manifestation that actually is dynamically generated by running a process at access time. Thanks to this functionality novel types of documents, such as documents with automatically changing manifestations computed on demand from raw data, can be supported by the digital library. From a technical point of view, this extension has a substantial impact on the DL features, however, in the framework provided by OpenDLib, its implementation has been quite simple. This is mainly due to the characteristics of the DoMDL model and its related management functionality. In fact, DoMDL is able to associate the URI of a specific task with a manifestation. In this case, this task uses the gLite WMS wrapper[7] in order to execute a process customised with the information identifying the job/DAG to be executed and the appropriate parameters.

[6]In gLite terminology jobs are an application that can run on a Computing Element, and DAGs are direct acyclic graphs of dependent jobs.

[7]The OpenDLibG service interfacing with a gLite based grid infrastructure in order to execute jobs on third party computing elements.

A concrete example of exploitation of this functionality is provided in the following section.

3.5.2 OpenDLibG and the Environmental DL

The third example is related with the exploitation of OpenDLibG in the context of the experimentation activity conducted into the DILIGENT project [DIL]. In this context there is the needs to support a group of agencies working together to define environmental conventions. By exploiting their rich information sources, that range from raw data sets to maps and graphs archives, these organisations periodically prepare reports on the status of the environment.

To demonstrate the potentialities of this DL powered by the Grid and the feasibility of on-demand reports generation, level two ENVISAT-GOMOS products[8] are stored in the digital library in order to be elaborated on demand. In particular, the GOMOS (Global Ozone Monitoring by Occultation of Stars) sensor measures ozone, temperature, moisture, NO_2, NO_3, $OClO$, O_3, and a specific application capable to elaborate these data, named BEAT2GRID, was provided by the ESA organisation and adapted to run on a gLite based infrastructure by the CNR team.

Moreover, a specialised user interface for visualising GOMOS virtual information objects is produced. It allows end-users to ask for the elaboration of these products in order to access to a number of human readable outputs. This interface progressively shows the status of the workflow execution and at the end gives access to the generated information. In particular the following output can be generated: geolocation information extracted from the data file; the NO_2/NO_3 image profile information showing the density with respect to the altitude; the NO_2/NO_3 profile information comprising date, time, longitude of tangent point, latitude of tangent point, longitude of satellite, latitude of satellite; the ozone density related to the altitude and the ozone density covariance.

Figure 3.7 shows an example of report illustrating (*i*) the status of the workflow, (*ii*) the graphs resulting from a completed elaboration, (*iii*) the NO_2 profile information, and (*iv*) the derived report metadata.

3.6 Related Work

A lot of prior works exists with respect to the representation of information objects both in the field of digital library than in other research area. In this section we analyse the models and the related functionality offered by the three most important and significant digital library systems, i. e. DSpace and Fedora, as well as we compare

[8]ENVISAT (Environment Satellite) is an ESA Earth observation satellite launched in March 2002. Its purpose is to collect earth observations: it is fitted with 10 sensors ASAR, MERIS, AATSR, RA-2, MWR, DORIS, GOMOS, MIPAS, SCIAMACHY, LRR and other units. Detailed information about the ENVISAT satellite can be found at `http://envisat.esa.int/`

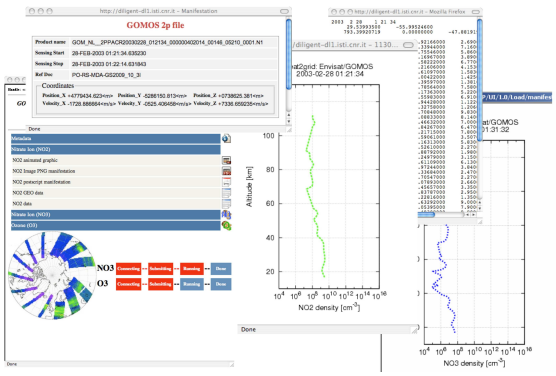

Figure 3.7: A GOMOS Virtual Information Object

the object representation models underlying the MPEG21 standard and the METS format.

3.6.1 The DSpace Data Model

DSpace[9] [TBS+03b, TBS03a] is an open source digital library system designed to operate as a centralised system for capturing, storing, indexing, preserving, and redistributing documents in digital formats. It has been initially designed to manage intellectual outputs of a university research faculty.

Item is the basic archival element in DSpace, as such it corresponds to the DoMDL Document entity. An item is organised into bundles of bitstreams, where a *bundle* is a set of somehow closely related bitstreams, partially corresponding to our View entity, and a *bitstream* is a stream of bits. Usually, a bitstream is a computer file, and it is therefore close to the physical part of our Manifestation entity. For example, a document having two different manifestations, a PDF and an HTML one, is modelled in DSpace as an item having two bundles: the PDF bundle, which has a bitstream representing the PDF file, and the HTML bundle, which has a set of bitstreams representing the component HTML files and images.

The ordered sequence is the only type of relationship that can be expressed among bitstreams of the same bundle. The concept of edition is not explicit within this model, even if it may be modelled via particular structural metadata by adapting some of the DSpace components. Furthermore, references, that enable the construction of documents as aggregation of already existing ones, are not explicitly supported.

DSpace manages descriptive, administrative and structural metadata. Each item is associated with one qualified Dublin Core descriptive metadata record. The used metadata schema can be changed but the system search and submission operations must be explicitly updated as they are not capable to automatically adapt them-

[9] `http://www.dspace.org`

selves to these changes. The system manages only one descriptive metadata record for each item. When multiple metadata records are available for a same document, one is handled as a proper metadata record used in the discovery phase, the others are stored into the system by using the bundle concept. Administrative metadata include preservation metadata and authorisation policy metadata expressed in a proprietary format. DSpace structural metadata can be considered as being fairly basic, i. e. bitstreams of an item can be arranged into separate bundles as described above. A plan exists to improve this aspect of metadata in future DSpace releases.

3.6.2 The Fedora Object Model

Fedora[10][PT02, LPSW05] is a repository service for storing and managing complex objects.

A Fedora digital object is composed by (*i*) a *unique identifier*, (*ii*) a set of *descriptive properties*, (*iii*) a set of *datastreams*, and (*iv*) a set of *disseminators*. Descriptive properties maintain the information that is needed for managing the objects within the repository, i. e. the object type, its state, its creation, and last update date. The type is used to distinguish among the primitive Fedora objects, while the state is used to distinguish among active, inactive and deleted objects.

Datastreams are containers used to maintain both data and metadata belonging to an object. Therefore the same concept is used to model bytestreams representing the document as well as metadata to express relationships with other objects, policies and audit data. Moreover, a datastream is used both to encapsulate any type of bytestream internally as well as to reference to it externally. Datastreams are thus a very flexible mechanism that makes it possible to aggregate the local content with the external content and the document structure with its content. The broad concept of datastream, which is equivalent to the DoMDL view, uses reserved datastreams to differentiate between its types. For instance, a datastream of type DC is used to express the DC Metadata record [DC], while a datastream of type REL-EXT is used to express object-to-object relationships following a well established relationships ontology[11].

Disseminators are components that associate an external service with the object in order to supply a virtual view of the object itself or of its datastream content. The Fedora repository produces this virtual view by interoperating with the service. The Fedora approach to virtual views is thus object centric, i. e. in order to offer a new view over a set of documents a service capable to disseminate this view must be created and all the objects of the set must be updated to include the new service.

Fedora object model covers also versioning related to components, i. e. datastreams and disseminators. The system automatically creates a new version of them whenever they are modified, while maintaining also the former representation with-

[10] `http://www.fedora.info`

[11] `http://www.fedora.info/definitions/1/0/fedora-relsext-ontology.rdfs`

out changing the document structure. The same component maintains then all the versions and each of them is identifiable via its own identifier internal to the component.

DoMDL and Fedora digital object models have many commonalities and both aim at managing complex structured documents. Main differences concern mechanisms for offering virtual views as well as for decoupling structural information from content information.

3.6.3 MPEG 21 and the DIDL

Although MPEG21 [BdWH+03] originates within the multimedia community, its framework is general-purpose and can accommodate any kind of complex digital objects, including electronic texts, electronic journals, scientific datasets, etc. . There is a clear overlap between the problem domain addressed by the MPEG21 effort and those related to the management of information objects in the digital library community.

Despite of such similarities, MPEG21 has received little attention from the DL community, partly for the seemingly different areas of application and partly because MPEG standards must be purchased. To our knowledge, Los Alamos National Laboratory (LANL) Research Library is the only example of Digital Library technology adopting MPEG21 DIDL as the reference language for its information objects representation [BHdS03].

The standard puts the bias on two essential concepts: *Digital Items* (DIs), which are the units of exchange (the "what"), intended as hierarchical containers of resources, metadata, and other digital items; and *Users*, which are the producers and consumers of DIs (the "who"). Currently twelve high-level, modular parts of MPEG21 are defined.

The crucial part of MPEG21 is the Digital Item Description Language (DIDL), an XML Schema for the definition of Digital Item Declarations (DIDs). DIDs declare DIs, meaning both their content and behavior, i. e. relations with users and with the processing environment.

The DIDL XML schema reflects a set of abstract concepts according to which a DI can be described as a combination of nested *containers*, *items*, *components*, and *resources*[12].

DID is a container, which contains a sequence of other containers or items; an item can include other items or a sequence of components, which in turn include a sequence of resources. Resources, i. e. their content, within the same component are considered equivalent, thus an agent may use any of them. Furthermore, *descriptor* elements can be included in elements of a DID to describe properties and behavior of the including elements, such as identification, processing information (code or

[12]Actually the data model includes other primitives, but they are outside the aim of this comparison

pointers to code for the processing of an element), and rights and permissions. Other, personalized, descriptors can be customized by users to describe element context-specific properties.

From the digital library perspective, DoMDL and DIDL are high-level approaches for complex digital object description and representation, in that they both accommodate any kind of metadata formats and object datastreams. However, the two models propose different level of abstraction and their comparison resorts to the well-known trade-off between high-level and low level models. DoMDL is specifically designed to model digital library information objects. Thus its primitives represent components, properties, and relationships, targeting the application context. DIDL is attractive because of the generality and flexibility offered by its data model, plus the extensibility supported by the descriptors approach. Indeed, any system of interrelated components can be described as a DID.

3.6.4 METS

The Metadata Encoding and Transmission Standard (METS) [The02] provides an XML document format for encoding metadata necessary for both management of digital library objects within a repository and exchange of such objects between repositories.

A METS document consists of seven major sections: (*i*) *METS Header*, contains metadata describing the object itself; (*ii*) *Descriptive Metadata*, reports the descriptive metadata, possibly in multiple manifestations, that can either be internally embedded or point to an external source; (*iii*) *Administrative Metadata*, reports administrative metadata (e. g. how object is created and stored, its intellectual property rights, its provenance) and can be internally stored or reference to external sources; (*iv*) *File Section*, contains the list of all files representing the object content grouped with respect to the version of the digital object they belong to; (*v*) *Structural Map*, contains the hierarchical structure for the digital object, and links the elements of that structure to content files and metadata that pertain to each element; (*vi*) *Structural Links*, contains the hyperlinks between the nodes of the Structural Map; and (*vii*) *Behaviour*, contains the metadata used to associate executable behaviour with the information object.

Many similarities exists between such rich and complex model and the DoMDL. In particular, the structural map that allows the components of a complex object be organized hierarchically corresponds to the DoMDL structure file. Notable is the behaviour section which allow the object be equipped with the logic needed to dynamically generated alternative manifestations. These similarities validate the DoMDL modelling choices. The transformation of the objects from one model to the other is an easy task and improves the interoperability between diverse systems.

Chapter 4

Information Space Organisation: the Collection Service

In this chapter we present the design and implementation of the *Collection Service*, a service introduced into the Reference Model that is mandatory for efficiently building Virtual Digital Libraries. In particular this service provides the mechanism of collections, i. e. virtual views over the information space. Thanks to this mechanisms users and services are enabled to tailor the information space to their needs and thus improve the discovery of and the access to the information objects they need, e. g. identify the portion relevant for a community, organise the content into novel and unpredictable sets, reduce the space where to search in.

This chapter is organised as follows. Section 4.1 introduces the service and the collection mechanism as instruments to dynamically customise and organise the information space of a Virtual Digital Library. Section 4.2 reports a functional view of the Collection Service and details of the metadata and the definition language needed for managing collections. Section 4.3 presents the architecture of the service by reporting on the main components, their functionality and the relationships among them. Section 4.4 reports on the *query-based sampling* technique as the mechanism used to acquire a description of the information sources constituting the information space. Section 4.5 introduces the *source selection* technique as the mechanism used by the collection service in order to find the appropriate information sources where to search for identifying the objects belonging to the collection. Section 4.6 reports the results of the experimental evaluation of the proposed techniques that has been conducted to prove the effectiveness of the approach. Section 4.7 presents the implementation of a collection service in the context of the Cyclades project (IST-2000-25456). Here, the central role such type of service plays and the improvement it produces are highlighted in the context of a DL built by aggregating third-party information sources. Finally, Section 4.8 reports on related research.

4.1 Introduction

Building a Virtual Digital Library by aggregating content from a set of different heterogeneous information sources provided by third parties, that grow independently along the time, presents many advantages but also introduces problems. One of the advantages is that thanks to these type of DLs the content produced to serve the information needs of those communities for which the information sources were originally set up becomes available to meet also the needs of other multidisciplinary communities whose interests span across various information sources. On the contrary, one of the main problems encountered in designing these DLs is the implementation of an efficient and effective resource discovery mechanism. The heterogeneity of the content and the huge dimension of the stored information render this problem hard to solve.

In the past, the most common way to deal with resource discovery consisted in structuring the whole information space into a number of well established content classes, possibly organised hierarchically. Before formulating their queries, user are asked to navigate in the hierarchy and to locate the class that best satisfies their needs. This organisation is based on some fixed set of criteria – e. g. subject, date, location – that reflect both the typology of the underlying information sources and the needs of the expected user communities. This solutions fails for Virtual Digital Libraries. Each new document added to the information sources constituting the information space, or stored into the new sources must be explicitly indexed according to the terms of the established organisation. This organisation may over the time become obsolete and not capable to satisfy anymore the needs of the new and heterogeneous communities of users asking for Virtual Digital Libraries.

We propose a novel approach to dealing with information space organisation. This is based on the *Collection Service* (CS), that is a dynamic information space mediator that mediates between the real organisation of the set of information sources constituting the information space, and the virtual organisation of information objects into virtual sets, named *collections*, that are meaningful from the perspective of the user communities as well as of the services constituting the Virtual Digital Library. Via a collection, a set of information objects logically correlated can be grouped together in order to satisfy an information need and to be referred as an information unit. The most important characteristic is that this set of objects is characterised via logical criteria and thus it is dynamic, e. g. if a new information object meets the collection definition criteria then it automatically becomes a member of the collection. The CS accepts requests for the creation of new collections, expressed in term of a set of criteria and, by exploiting the information about the underlying information space configuration, dynamically generates collection descriptive metadata that are disseminated to the other services on request.

4.2 The Collection Service Functionality

The proposed service introduces a novel path in accessing an information space. In particular, it allows users to follow the search strategy proposed in [Bla02]. Here a two-stage search process is presented in order to improve the document retrieval from *large* information spaces. The first phase of this process consists of the *partitioning* of a large information object collection into small collections (partitions), while the second phase consists of submitting the query representing the information needs to the right partition, i. e. the partition which is likely to contains the desired documents. The Collection Service supports the partitioning mechanism via the definition of *virtual collections*. A collection is usually defined as a statically identified set of documents. Our service, instead, implements *virtual* collections as it does not gather nor stores the documents belonging to a collection, as other solutions do – e. g. [WBB01, Ber02], but it characterises and identifies them via a set of definition criteria. This means that CS collections are capable to adapt to and follow the dynamism of the underlying information space. If a new document meets the collection definition criteria then it is automatically included in the collection. The requests for the creation of new collections are submitted to the CS. These are formulated via a declarative collection definition language named *Membership Condition language* that is presented in Section 4.2.2.

Collection definitions are stored by the CS and information about them is disseminated to the other services upon request. A collection is described by *Collection Metadata*, i. e. a set of data about the collection that comprises identification and managing information. The format and and the semantics of this metadata are described more in detail in Section 4.2.1.

4.2.1 Collection Metadata

Collection Metadata is the information that the system stores about a collection and disseminates upon request. The collection metadata are generated by a stepwise process that is composed by the following phases:

1. Via the CS GUI (Section 4.7.2) the user expresses his own information need using a definition language. Note that this kind of information need is not a one-time request, i. e. it is not intended for the identification of the single information object the user is interested in, but it represents an expression of interest about a set of information objects with certain characteristics where further to search in for an information object;

2. The system processes the request of the user in order to *generate* the collection. During this phase detailed data about the collection (see Section 4.2.1) are derived by the system using a set of internal and automatic procedures. The most important procedure identifies the information objects that belong to

the collection. These objects are characterised by the set of characterisation criteria that forms the *Retrieval Condition* (see Section 4.5 for details);

3. The collection is now ready to be *consumed* by other DL services. Example of exploitation can be found in Section 4.7 where we present the implementation of the Collection Service in the context of the CYCLADES project and describe its central role in supporting an appropriate search and an advanced recommendation functionality in a context of heterogeneous information sources.

Collection metadata is composed by the following fields:

- *Identifier* - the unique identifier of the collection;
- *Name* - the name of the collection;
- *Description* - the textual description associated with the collection;
- *Membership Condition (MC)* - the condition that the creator has used to define the collection. It is maintained as a formal specification of the collection;
- *Retrieval Condition (RC)* - the condition that specifies how to retrieve, effectively and efficiently, the documents belonging to the collection;
- *Parent* - the identifier of the parent collection. It is used to maintain the hierarchical organisation in the set of collections.

This is the minimal set of fields required to manage collections. It contains identification information (Identifier, Name and Description), information on how to formally (MC) and operationally (RC) retrieve the content of the collection and information (Parent) about its position in the the hierarchical organisation of the set of collections. This set of fields can be extended with other type of information – e. g. statistics about content, policies to regulate the access, and so on – in order to allow other services having a more rich and detailed description of the collection. The richer this set of fields is the more accurate is the functionality that the other services can supply building over collections.

The main issue that the CS comes up against is the automatic derivation and generation of these metadata fields. A lot of them, e. g. Name, can be derived directly from the definition criteria expressed by the user, others are generated by the system, e. g. Identifier, whereas others require supplementary knowledge that the CS must either receive as input or acquire automatically. Sections 4.4 and 4.5 discuss the latter case in more detail.

4.2.2 Membership Condition Language

The CS allows users to specify their own information needs via a declarative collection definition language named *Membership Condition Language*. This language

must be simple, expressive and quite powerful to capture any kind of information need arising from users. On the other hand, the definitions given in this language must be translated into a condition that all the information sources constituting the information space understand.

The syntax of the language that has been used in the prototype implementation of the CYCLADES CS (Section 4.7) is given below using the Backus-Naur Form (BNF):

```
query       ::= condition* [, (archiveList)]
condition   ::= ([weight,] field, predicate, value)
weight      ::= + | - | 1..1000
field       ::= [schemaName":"]attributeName
predicate   ::= cw | < | <= | >= | > | = | !=
archiveList ::= archiveName | archiveName, archiveList
```

This is an ALTAVISTA-style language where a query is a set of conditions, which are either optional, *mandatory* (+) or *prohibitive* (-). In addition, it allows for weighting of optional conditions. With respect to the structure of metadata records it assumes that they have a one-level structure and allows for the use of namespace (`schemaName`). The set of predicate supported is composed from the classical comparison operators ($<$, $<=$, $>=$, $>$, $=$ and $!=$) plus `cw` operator used to specify a condition on the content of a text field, e. g. `(description, cw, library)` stands for "the field `description` contains the term `library`".

This language has pros and cons:

- it is quite simple and intuitive as it is similar to others, well known query languages;
- it is quite general, the assumption about the one-level metadata record structure can be simply removed using the attribute name path instead of the attribute name;
- it is not expressive as others query languages are, e. g. SQL. We are currently working at the evaluation of the *right grade of expressive power* required in order to define collections.

4.3 The Collection Service Architecture

Figure 4.1 shows the logical architecture of the Collection Service expressed in terms of the Digital Library Service Reference Architecture (Section 2.6). This picture shows how the initial user description of the collection, i. e. the membership condition *MC*, is manipulated in order to produce the collection metadata *MD* that are stored into the system and disseminated upon request via the CS API, in accordance with the process presented in Section 4.2.

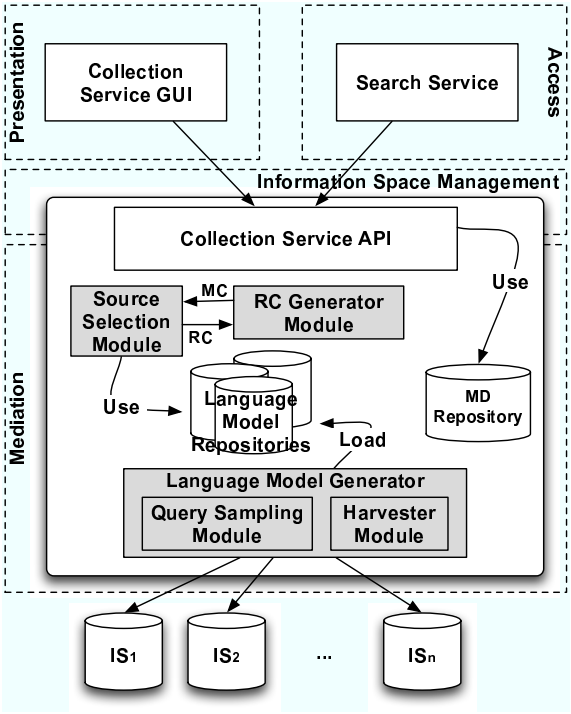

Figure 4.1: The Collection Service Logical Architecture

The CS contains a module, the *RC Generator Module*, that is responsible for the generation of the retrieval condition *RC*, i. e. the condition that is used in order to find the documents belonging to the collection. The RC consists of the membership condition plus a set of automatically selected information sources[1] that are relevant to the conditions specified via the membership condition.

One of the functionality provided by the Collection Service is the translation of the Membership Condition, i. e. the collection definition criteria, in terms of Retrieval Condition, i. e. the query actually used to identify the information objects constituting the collection. The purpose of this translation is twofold: on the one hand it is necessary to rewrite the MC into the query language supported by the information sources in order to render them able to reply; on the other hand it is important to identify in advance the information sources to query for collecting collection objects in order to reduce the number of sources to interact with. The latter aspect is also known as source selection: the CS generates a Retrieval Condition that contains the list of information sources to be queried in order to identify the information objects belonging to the collection.

In order to identify this set of archives the *RC Generator* module uses the *Source Selection Module*. This module is responsible for solving the source selection problem, i. e. the selection of the subset of information sources relevant to a given query

[1] In the follow we will use the terms archive and information source as synonyms.

among the set of accessible sources (see Section 4.5).

In order to choose the *right* information sources, the CS must *know* them, i. e. it must have an appropriate knowledge of the content of each information source. How to best represent an information source content is an open problem. The current approach to this problem is based on the use of a *language model*. This approach will be presented in detail in Section 4.4 where a technique used for acquiring the language model from a set of non co-operating information sources is reported.

Finally, it is worth noting that due to the mediation role that the logical modules play, some of them must interact directly with heterogeneous information sources and, therefore, part of their implementation is strictly dependent on that environment.

4.4 Language Model and Query-Based Sampling

In order to select the information sources relevant to a query, the CS must have a description of their content. From our point of view an information source is a set of information objects. The issue of how to best describe this set is still open. The most widely used approach consists in using a *language model*, i. e. a list of terms with their term frequency or term weight information. As it will be clarified in the next section, this knowledge is sufficient for the source selection technique that we have adopted.

However, we can assume that the language model is provided by the single information sources only in a federated environment where the participating institutions agree on a number of rules and thus collaborate in providing system functionality. Instead, in the context of Virtual Digital Libraries where the information sources are pre-existent and it is required the minimal effort in participating to the system, we cannot assume that the information sources supply their own language model and thus we need to envisage a mechanism to acquire it.

In our Collection Service we have envisaged two mechanisms for learning the language model. The first assumes that the Information Sources act as data providers as in the context of the Open Archive Initiative [LV01], i. e. adopt the OAI-PMH technical framework as a means of exposing metadata records about their content. Then the Collection Service collects all the metadata records and extract from them the needed information. The quality of the language model acquired via this approach depends on the quality of the metadata records exposed. The second methodology comes from the distributed information retrieval area and is known as query-based sampling requiring query-response capabilities by the Information Sources.

The query-based sampling technique has been proposed by Callan and Connell [CC01] for acquiring accurate resource description[2] in a context where information sources are text databases. This technique does not require the co-operation of

[2]Resource description is a kind of knowledge about the content of an information source.

source providers, nor does it require that source providers use a particular search engine or presentation technique. Resource descriptions are created by running queries and examining the information objects returned. At the end of this process a sample of the records of the information source that represent its content is acquired. This set is called resource description and using it the language model of the archive can be derived.

```
1: query = generateInitialTrainingQuery();
2: resultSet = run(query);
3: if(|resultSet| < L_tr){
4:    go to 1;
5: }else{
6:    updateResourceDescription(resultSet);
7:    if(NOT stoppingCriteria()){
8:       query = generateTrainingQuery();
9:       resultSet = run(query);
10:      go to 6;
11:   }
12: }
```

Figure 4.2: Query-based Sampling Algorithm

Figure 4.2 reports the query-based sampling algorithm that we have appropriately extended in order to deal with information sources presenting information objects annotated with multiple text attributes – e. g. bibliographic records – (similar to [XCLN98]). This algorithm uses the functions explained below:

generateInitialTrainingQuery() generates the *start* training query. In order to generate a query we need: (*i*) a set of words among which randomly choose the ones to build the condition and (*ii*) a set of attributes among which randomly choose the ones to build the condition. For each selected attribute we randomly select 1 to max_t distinct terms and for each pair we choose an operator to relate attribute and term into the condition.

This function, like the generateTrainingQuery(), is dependent on the information source query language and from other parameters. Here we assume that each query language supports at least conditions on single attribute of a bibliographic record. All the other aspects are configurable.

In the CYCLADES CS prototype (Section 4.7) we have taken the following design choices: (*i*) the words belongs to the set of terms that characterise the second and the third level of the Dewey Decimal Classification [DDC] system, (*ii*) the attributes that we have used belongs to the Dublin Core [DC] fields, (*iii*) $max_t = 4$, and (*iv*) the operator used is always the `cw` operator.

updateResourceDescription() updates the set of information objects that represent the resource description. Note that a query must return at least L_{tr}

objects before the objects collected (the top L_{tr}) can be added to the resource description data set. This minimum result size is required because query returning small results do not capture source content well.

In our prototype we have used L_{tr} equals to 4 as proposed in [XCLN98], this is just another configuration aspect.

stoppingCriteria() evaluates if the stopping criteria was reached. In our best knowledge no one has proposed significantly stopping criteria.

Callan and Connell [CC01] experiments have been conducted stopping the sampling after examining 500 documents, a stopping criteria chosen empirically observing that augmenting the number of documents examined the language model does not improve significantly.

In our prototype implementation the stopping criteria is reached when the system runs 10 queries, each ones returns at least L_{tr} records without resource description records set changes. This is an aspect that we plan to further investigate in the future.

generateTrainingQuery() generates the *next* training query. Training queries are generated as follows:

1. randomly select an object R from resource description data set;
2. randomly select a set of attribute of R to use in training query;
3. for each attribute to be included in the training query, construct a predicate on it by randomly selecting 1 to max_t distinct terms (stopwords are discarded) from the corresponding attribute value and using the `cw` operator.

In order to investigate the accuracy of the learned resource description acquired via the sampling technique, we have conducted some experiments whose results are reported in Section 4.6.

4.5 Source Selection Technique

As already stated, source selection is the technique allowing to identify from a large set of accessible information sources the ones relevant to a given query. In our case the query is the MC, i. e. the collection characterisation criteria, while the selected information sources are used in the generation of the Retrieval Condition in order to allow a faster discovery of the information objects belonging to the collection.

The source selection problem can be formally defined as follows.

Definition 4.5.1 (Source Selection Problem) *Let* $\overline{IS} = \{IS_1, IS_2, \ldots, IS_N\}$ *be a set of Information Sources. Let q be a query. The source selection problem consists in computing* $E \subseteq \overline{IS}$ *such that* $\forall\, F \subseteq \overline{IS}$ *Goodness(q,E)* $\geq$ *Goodness(q,F).*

Goodness is a function on the results returned by a set of IS E against a query q defined as follows:

$$Goodness(q, E) = \sum_{IS_i \in E} s_i \tag{4.1}$$

where s_i is the result size returned by IS_i for query q.

In order maximise the *Goodness* value for a query it is sufficient to rank the various information sources estimating the result size returned by each one. The weighting scheme that we propose has been obtained by appropriately extending the CORI schema [CLC95] in order to manage bibliographic records instead of plain text documents and to consider a richer query language than a keyword-based ones.

In accordance with the Membership Condition Language reported in Section 4.2.2 we consider a keyword-fielded-based query model, this mean that a query q is defined as a list of condition (w_i, a_i, o_i, v_i) where:

- w_i is the *weight* of this condition. + mean that the condition must be fulfilled, - that the condition must not be fulfilled (the boolean NOT);
- a_i is the field of the bibliographic record involved in the condition;
- o_i is the operator to use, e. g. `<=`, `=`, `cw`, etc.;
- v_i is the keyword.

For example, to retrieve all the records having author "Candela" and subject "Cyclades" we use the following query:

```
(+,author,cw,‘‘Candela’’)(+,subject,cw,‘‘Cyclades’’)
```

The technique exploits the discriminatory power of different conditions to increase the accuracy of archives selection. This is done by summarising the content of the information source IS via the language model LM. As stated in Section 4.4 the language model consists of a list of terms with their term frequency and can be acquired by a sample of the source content. Using it the CS is able to calculate the *document frequencies* (denoted by $df_{i,j}$) defined as the expected number of records in IS_i that match against the condition c_j plus other statistical values described in what follows.

Formally, the *Goodness* score $G(q, IS_i)$ for IS IS_i and query q is defined as follows:

$$G(IS_i, q) = \begin{cases} 0 & \text{if } \exists k \in [1..|q|] \,|\, w_k \in \{+, -\} \wedge p(c_k|IS_i) = 0 \\ \dfrac{\sum_{k=1}^{|q|} p(c_k|IS_i)}{|q|} & \text{otherwise} \end{cases} \tag{4.2}$$

where the "belief" $p(c_k|IS_i)$ in IS_i for condition c_k is defined as

$$p(c_k|IS_i) = \begin{cases} T_{i,k} \cdot I_k \cdot w_k & \text{if } w_k \in [1..1000] \\ T_{i,k} \cdot I_k & \text{if } w_k = \text{“+” or } w_k = \text{“–”} \end{cases} \quad (4.3)$$

$$T_{i,k} = \frac{df_{i,k}}{df_{i,k} + 50 + 150 \cdot \frac{cw_{i,k}}{\overline{cw_k}}} \quad (4.4)$$

$$I_k = \frac{\log\left(\frac{|D|+0.5}{cf_k}\right)}{\log\left(|D| + 1.0\right)} \quad (4.5)$$

where:

$df_{i,k}$	is the expected number (estimated via LM_i) of IS_i documents satisfying c_k,
$cw_{i,k}$	is the number of terms in attribute a_k in LM_i,
$\overline{cw_k}$	is the mean cw of the ISs being ranked,
cf_k	is the number of ISs satisfying c_k,
$\|D\|$	is the number of the ISs being ranked.

Note that the accuracy of the automatic source selection using this technique is promising, i. e. the RC that is generated approximates very well the MC as demonstrated by the experiments presented in Section 4.6.

4.6 Experimental Evaluation

Before proceeding with the description of the tests conducted to evaluate the technologies and the approaches proposed, we describe the test corpus we adopted for our experimentation.

4.6.1 Test Corpus

To the best of our knowledge, no corpus exists in the literature that fits to our settings. Thus we built a suitable corpus by taking the data from the Internet.

We decided to assemble two different corpus, one based on records gathered via the OAI-PMH protocol and another based on web documents selected from the *Open Directory Project*[3] (ODP or DMOZ).

The first corpus, named the *OAI Corpus*, is built on about 1000K records gathered from the 62 OAI compliant archives available in the context of the Cyclades project [CS04]. To evaluate the query-based sampling, we built two information sources:

- *Archive 1*, quite small and homogeneous information source containing 1616 records, 13,576 unique terms after stopwords removing and consisting in computer science papers published by the same authority;

[3]The Open Directory Project `http://dmoz.org`

- *Archive 2*, a larger and heterogeneous information source containing 16,721 records, 79,047 unique terms after stopwords removing and consisting in papers published by different authorities on different topics.

To evaluate the source selection technique we randomly generated 200 collections, i. e. the Membership Conditions, using Dublin Core fields. The collections generated are of two types, each type consisting of 100 collections:

- *T1*, generated using a combination of conditions on `description` and `title` fields;

- *T2*, generated using a combination of conditions on all fields of the Dublin Core schema.

The second corpus, named *DMOZ Corpus*, is built by relying on the largest human-edited directories of the Web made available by the homonymous project. These data include over 5.1 million sites, about 69,000 editors and over 590,000 categories and power the core directory services for the Web largest search engines, e. g. Google[4]. The document base we considered in our experiment is a subset of the categories under the Science umbrella consisting of 1415 folders and 18,091 documents. To evaluate both the query-based sampling and the source selection algorithms we aggregate the information objects according to three different information source distribution schemes whose characteristics are reported in Table 4.1:

- *ScienceI*, each of the 23 information sources corresponds to the first-level sub-category of Science into the DMOZ hierarchy. As a consequence, the information objects of an information source share the *same* topic;

- *Quasi-random*, each of the 85 information sources contains information objects, which have been selected randomly, from a subset of the first-level sub-category of the Science category. Thus these information sources are *more heterogeneous* than the previous ones because the set of objects contained into each source belongs to a different, but limited, set of categories. In this case we have *multi-topic* information sources;

- *Random*, each of the 100 information sources contains information objects randomly selected from the whole corpus. Thus the information sources are *highly heterogeneous*.

The creation of a pool of test collections is easy. In particular, we created 300 Membership Conditions by taking the terms from three categories of Science, i. e. Agriculture, Anomalies and Alternative Science, and Astronomy. A collection definition criteria corresponds to the set of top 100 terms belonging to the documents classified under the DMOZ category.

[4]Google web site `http://www.google.com`

Environment	Number of Information Sources	Source Size Max	Min	Avg
ScienceI	23	15,722	11	3,027.47
Quasi-random	85	5,660	1	809.67
Random	100	768	621	696.32

Table 4.1: The DMOZ Information Sources Experimental Environments

4.6.2 Query-based Sampling Evaluation

To evaluate the effectiveness of the query-based sampling mechanism as algorithm for approximating the information source content we compared the learned resource description of an information source with the real resource description for that information source. Resource descriptions are usually represented using two information, a vocabulary V of the set of terms appearing in the information source objects and a frequency information for each vocabulary term. This frequency, also called *document frequency* (*df*), represents the number of information objects containing that term. In accordance to [CC01] we have used two metrics to evaluate the quality of the resource description acquired by sampling, (*i*) the *ctf ratio* (CTF) to measure the correspondence between the *learned* (V') and the *real* (V) vocabulary and (*ii*) the *Spearman Rank Correlation Coefficient* (SRCC) to measure the correspondence between the learned and the real frequency information. These metrics are calculated using formulas 4.6 and 4.7 where:

- ctf_i is the number of times term i occurs in the resource description of an information source,
- δ_i is the rank difference of common term i where term rankings are produced by learned and actual *df* values,
- n is the number of terms.

$$\text{CTF} = \frac{\sum_{i \in V'} ctf_i}{\sum_{i \in V} ctf_i} \tag{4.6}$$

$$\text{SRCC} = 1 - \frac{6}{n^3 - n} \sum {\delta_i}^2 \tag{4.7}$$

OAI Corpus Experimentation

Five trials were conducted for each information source and for each trial a resource description consisting of a maximum of 500 records was acquired. The results reported here are the average of the results returned by the trials.

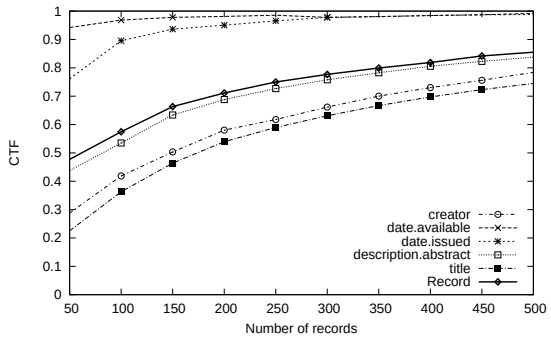

Figure 4.3: Archive 1 – CTF Graph

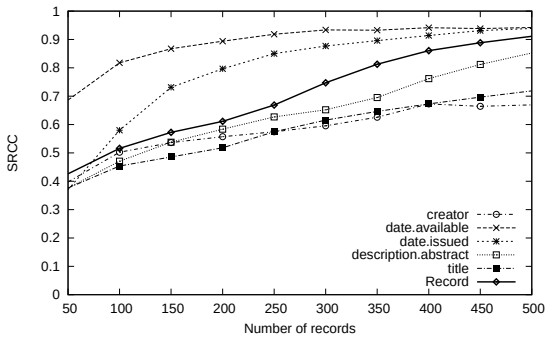

Figure 4.4: Archive 1 – SRCC Graph

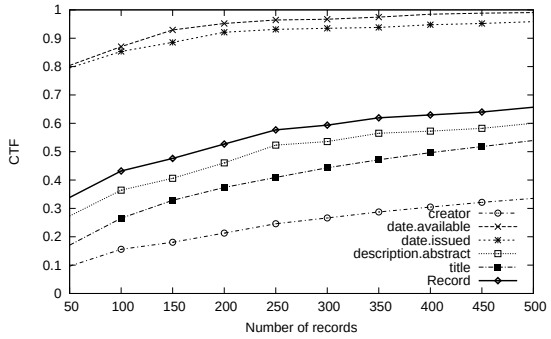

Figure 4.5: Archive 2 – CTF Graph

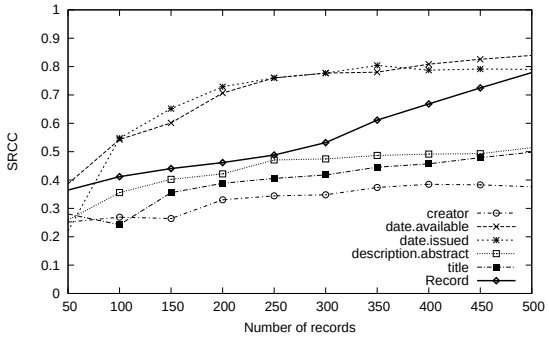

Figure 4.6: Archive 2 – SRCC Graph

Within this context we decided to compare the characteristics of the language model calculated with respect to the various fields with those obtained while considering the whole object as a blob of plain texts. In particular, we conducted the sampling by issuing per field queries with values taken from the set of terms assigned to that field in the case of the attribute curves, i. e. `creator`, `date.available`, `date.issued`, `description.abstract`, and `title`. In the case of the `Record` curve, the sample was conducted by issuing a query on the whole content with terms randomly taken from the whole set of acquired terms, independently from the field they came from.

Figures 4.3 and 4.4 show respectively the graphs of the CTF and the SRCC metrics calculated by field and by number of information objects for *Archive 1*, varying the number of records considered building the resource description, while Figures 4.5 and 4.6 report the same metrics in the case of *Archive 2*.

By observing the CTF graphics we note that the language model acquired for the first archive is generally better then the one acquired for the second one. Moreover we can note that the language model acquired on different fields has diverse characteristics and that those acquired for the whole record can be considered a good representative of the archive content because outperform many of the per field metrics. The reasons of this behaviour are twofold: on the one hand they are related to

the characteristics of the information source, *Archive 2* contains more information objects and is more heterogeneous in terms of content than *Archive 1*; on the other hand they depend on the intrinsic characteristics of the fields, i. e. certain fields are more heterogeneous than others, e. g. `creator` is characterised by more terms than e. g. `date`.

By observing the SRCC graphics we can notice that it suffers of the same drawbacks of the CTF curve related to the characteristics of the information sources and of the fields. However, in the case of the whole record considered as a blob of texts, the measures obtained are encouraging since are greater than 80% (see `Record` line).

The major outcome of these experiments is that the acquisition of the language model by issuing queries on the whole record is a good compromise between the quality of the information source description acquired and the number of queries performed. Moreover, the other interesting aspect is that even if the metrics on the quality of the language model are not impressive, i. e. are not equals to one, the source selection works well with the sample acquired using our algorithm as shown in the Source Selection Evaluation section.

DMOZ Corpus Experimentation

In the case of the DMOZ corpus it is not possible to investigate on the per field sampling because of the nature of the information objects, i. e. web documents. Thus we concentrated of the characteristics of the language model acquired with respect of the characteristics of the Information Sources. Thus in this case the sample is acquired by issuing terms based queries, similar to those we use in Google, and the terms are selected from the whole set of terms present into each information object constituting the partial sample.

In Table 4.2 we report the characteristics of the information sources and their approximations in terms of number of records gathered and number of terms in the samples, respectively.

	Avg Record			Avg Terms		
Environment	Source	Sample	Sample%	Source	Sample	Sample%
ScienceI	3,027.47	279.17	9.22%	102,375	17,707.2	17.29%
Quasi-random	809.67	238.3	29.43%	46,197.7	18,083.17	39.14%
Random	696.32	295.74	42.47%	48,571.31	27,121.01	55.83%

Table 4.2: The DMOZ Information Sources and their Samples – The Characteristics

In Table 4.3 we report the results of CTF and $SRCC$ effectiveness metrics. For example, observing the ScienceI case, we can note that acquiring just the 9% of the records of the source we are able to have a very close representation of the content of the information sources as we obtain a CTF of about 90% and an $SRCC$ of 80%. Moreover, note that the effectiveness of the approximations are quite independent from the content homogeneity of the information sources constituting the

environment. But, for instance, by observing the Random environment case, we can note that the number of records acquired by the sample process is much greater in percentage than that for the ScienceI case in order to get similar CTF and $SRCC$ values. Essentially, and quite intuitively, the more heterogeneous is an information source with respect to its content, the more information objects have to be gathered in its sample to reasonably approximate the information source content.

	CTF			$SRCC$		
Environment	Max	Min	Avg	Max	Min	Avg
ScienceI	98%	74%	87%	97%	53%	80%
Quasi-random	100%	71%	98%	100%	63%	85%
Random	92%	78%	87%	90%	75%	85%

Table 4.3: DMOZ – Statistics of Samples

4.6.3 Source Selection Evaluation

To evaluate the effectiveness of the source selection process, the idea is to compare the set of objects retrieved by query with the Membership Condition the set of selected sources against the set of objects obtained by querying all the information sources available. Due to the characteristics of our two test corpus and due to the diverse goal of the test conducted on each corpus different metrics are used to evaluate the effectiveness and are reported and discussed into the following subsections.

OAI Corpus Experimentation

We use the classic metrics of information retrieval,i. e. precision and recall, to evaluate the effectiveness of the source selection mechanism in the case of the OAI Corpus. In particular, given a collection definition MC_i and the relative RC_i obtained after source selection, $Precision_i$ is defined as the quantity:

$$Precision_i = \frac{|ret(RC_i) \cap ret(MC_i)|}{|ret(RC_i)|} \tag{4.8}$$

and $Recall_i$ is defined as the quantity

$$Recall_i = \frac{|ret(RC_i) \cap ret(MC_i)|}{|ret(MC_i)|} \ . \tag{4.9}$$

where:

- $ret(MC_i)$ is the set of records retrieved by submitting the MC_i query to all the archives of the dataset and for each taking the top-100 records. $ret(MC_i)$ is considered as the set of records effectively to be retrieved;

		Precision										
		0.00 – 0.10	0.11 – 0.20	0.21 – 0.30	0.31 – 0.40	0.41 – 0.50	0.51 – 0.60	0.61 – 0.70	0.71 – 0.80	0.81 – 0.90	0.91 – 1.00	
Recall	0.00 – 0.10	0.33%	0	0	0	0	0	0	0	0	8%	8.33%
	0.11 – 0.20	0	0	0	0	0	0.16%	0	0	0	5.83%	6%
	0.21 – 0.30	0	0	0	0	0	0	0	0	0	5.83%	5.83%
	0.31 – 0.40	0	0	0	0	0	0	0	0	0	7.5%	7.5%
	0.41 – 0.50	0	0	0	0	0	0.16%	0	0	0.16%	12.16%	12.5%
	0.51 – 0.60	0	0	0	0	0	0.16%	0	0	0	2.5%	2.66%
	0.61 – 0.70	0	0	0	0	0	0	0.16%	0	0	8.66%	8.83%
	0.71 – 0.80	0	0	0	0	0	0	0	0.5%	0.33%	8.83%	9.66%
	0.81 – 0.90	0	0	0	0	0	0	0	0	1.33%	9.83%	11.16%
	0.91 – 1.00	0	0	0	0	0	0	0	0	0	27.5%	27.5%
		0.33%	0	0	0	0	0.5%	0.16%	0.5%	1.83%	96.66%	

Table 4.4: Source Selection – Precision and Recall in OAI Corpus

- $ret(RC_i)$ is the set of records retrieved by submitting RC_i as query, i. e. submitting the MC_i query to the information sources mentioned in RC_i.

Table 4.4 reports the results of tests on the OAI Corpus composed by 200 collections. In it, each row/column pair (r, p), where r and p are intervals denoting respectively recall level and precision level, dictates the percentage of test pairs (MC_i, RC_i) such that $Recall_i \in r$ and $Precision_i \in p$. Furthermore, the right most column and the bottom row report the total amount w. r. t. a row and a column, respectively. For instance, from Table 4.4 we have that 27.5% of the test pairs (MC_i, RC_i) have recall and precision level in $[0.91, 1]$, while 96.66% of the tests have precision level in $[0.91, 1]$.

In Table 4.5 we report the response time needed to identify the records constituting each collection in the case of usage of Membership Condition, i. e. query all the information sources, compared with the usage of the Retrieval Condition, i. e. query a subset of information sources among those available. The improvement is terms of response time of the RC w. r. t. the MC is impressive. Thus we can conclude that with little loss in the set of records identified as members of the collection retrieved after automatic source selection we can obtain an high improvement in terms of the response time needed to identify them.

DMOZ Corpus Experimentation

The goal of the experimentation conducted with the DMOZ corpus is to investigate on the number of information sources to be inserted into the Retrieval Condition. In fact, even if the source selection is based on the ranking of the information sources based on the goodness value it is usually not appropriate to insert into the RC all

	T1	T2	Average
MC	162874 ms	186909 ms	174892 ms
RC	48469 ms	52253 ms	50361 ms
Improvement in ms	114405 ms	134655 ms	124530 ms
Improvement in %	70.24%	72.04%	71.20%

Table 4.5: Source Selection – Average Response Time

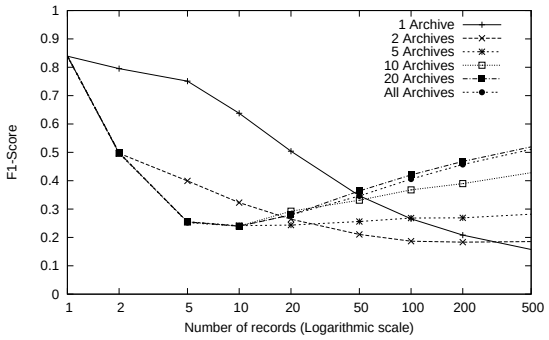

Figure 4.7: ScienceI – F1-Score Graph

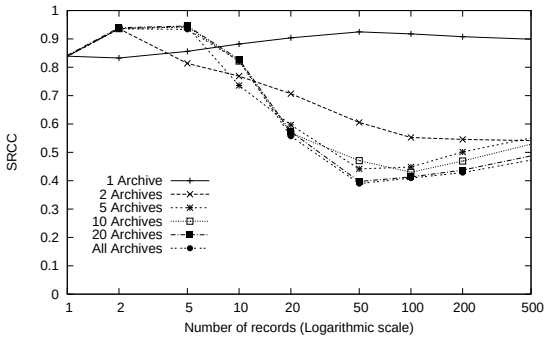

Figure 4.8: ScienceI – SRCC Graph

the information sources having a positive value because the number of these sources can be similar to the number of all those available and thus reduce the advantage of source selection. As a consequence a threshold is identified, e. g. the top-n ranked information sources, and the sources fulfilling this constraint are selected to be inserted into the RC.

For each of the 300 collections constituting the DMOZ corpus, we have evaluated the effect of varying the number of information sources to be used into the RC and the number of records (from 1 to 500) to be considered being part of the collection.

The metrics used to evaluate the effectiveness of our source selection algorithm are the *SRCC* (see equation 4.7 on page 91) and the *F1-score*, i. e. the harmonic mean of Precision (equation 4.8) and Recall (equation 4.9), defined as follows:

$$\text{F1-Score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.10}$$

The ranked list used to compute such metrics for each collection are obtained by issuing the query constituting the collection MC respectively to all the archives of the dataset (the baseline) and to the set of archive to be inserted into the RC (the test).

In Figures 4.7 and 4.8 we report the results for the ScienceI case. As previously observed, in this case an information source is homogeneous, i. e. its records belong to the same topic or set of topics. The selection of just the top-1 information source for each collection produces high F1 value, about 80%, and a high SRCC value. This means that our algorithm is able to find the most appropriate information

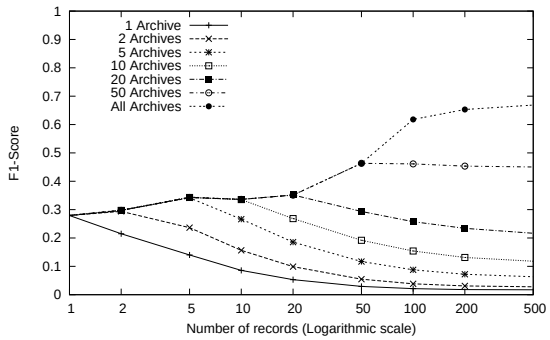

Figure 4.9: Quasi-random – F1-Score Graph

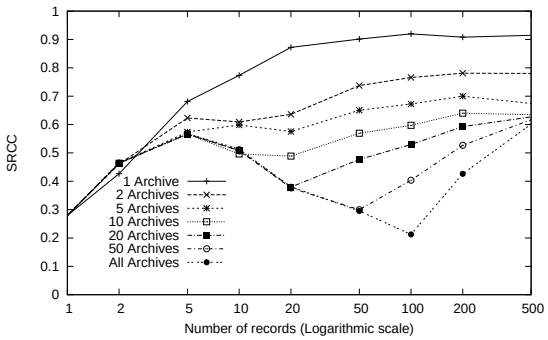

Figure 4.10: Quasi-random – SRCC Graph

source w. r. t. the collection in case the information source has a homogeneous topic. Selecting more than one information source produces a deterioration of the results as the selected information sources contain documents less pertinent to the collection topic.

In Figures 4.9 and 4.10 we report the results for the Quasi-random case. As previously observed these information sources are more heterogeneous than the previous ones, i. e. the records of a source may belong to a different but limited set of topics, and records about a topic are distributed among a limited number of information sources. In this case selecting just the top-1 information source to each collection produces a lower F1-Score than for the previous case. Moreover, the performance decreases if the number of records considered belonging to the collection increases. Selecting more than one source produces an improvement of the results as the selected information sources contain records relevant to the collection. Concerning the *SRCC* curve we note that increasing the number of records to be considered in a collection, each curve has a decreasing phase and finally it increases. The end of the decreasing phase coincides with the point where the F1-Score value starts to decrease. This indicates that the number of common records, between the baseline rank and the test rank, decreases and, thus this improves the SRCC. On the other hand, the numerator of both precision and recall decreases and, thus, F1-Score decreases.

In Figures 4.11 and 4.12 we report the results for the Random case. In this environment, the information sources are highly heterogeneous, i. e. the records of a source can belong to many different topics, and the records of a collection are distributed, potentially, among all sources. This is the worst case. We can note that the performance decreases if the number of records to be considered part of a collection increases, while it increases if the number of information sources to be included into the RC increases. However, our algorithm is still able to find the most appropriate information sources w. r. t. the collection.

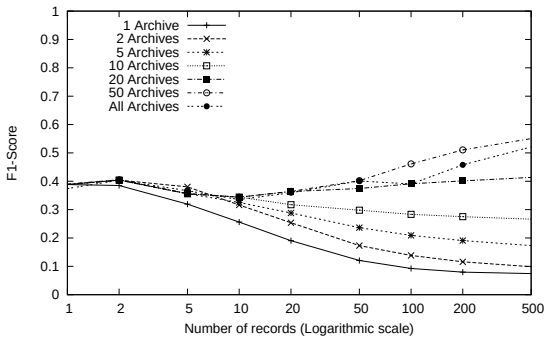

Figure 4.11: Random – F1-Score Graph

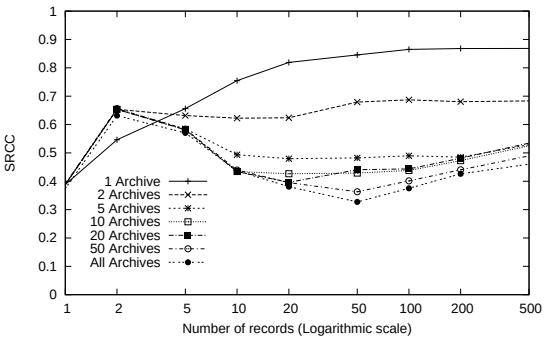

Figure 4.12: Random – SRCC Graph

4.7 Implementation: the Cyclades Collection Service

One of our main goals in designing and implementing the Collection Service has been reusability and adaptability to different contexts. In particular, we were interested in providing a service as a component of a Digital Library System useful to build Virtual Digital Libraries.

To enlarge the clientele of our Collection Service we decided to follow the component oriented approach as well as to stress the configuration aspects of both the various modules and the service as a whole. Many characteristics of the service are easily modifiable, namely aspects of the Membership Condition Language like the set of attributes or the set of predicate supported, tuning parameters of the query based sampling process like the set of words to use in query generation or the number of information objects to examine, the process to use in building the language model, tuning parameters of the source selection algorithm like the threshold.

In this section we report some details on the implementation of the first prototype of the Collection Service performed within the context of the Cyclades project (IST-2000-25456), thus partially funded by this project. Further details are reported in [CCP03b, CS04].

It is worth noting to highlight that (*i*) the components of this prototype have been used to implement the OpenDLib [Ope, CP02, CP03] collection service and (*ii*) the on-going IST project BRICKS [BRI] is exploiting the design and implementation choices proposed here.

4.7.1 Cyclades: a Personalised and Collaborative DL

The objective of Cyclades is to provide an integrated DL environment for users and groups of users (communities) that want to use, in a highly personalised and flexible way, "open archives", i. e. electronic archives of documents compliant with the OAI [LV01]. Informally, the OAI is an initiative between several Digital Archives

in order to provide interoperability among them. In particular, the OAI defines an easy-to-implement gathering protocol over HTTP, which give *data providers* (i. e. the individual archives) the possibility to make the documents' metadata in their archives externally available. This external availability of the metadata records then makes it possible for *service providers* to build higher levels of functionality. To date, there is a wide range of archives available in terms of its content, i. e. the family of OAI compliant archives is multidisciplinary in content. Under the above definition, CYCLADES *is an OAI service provider* and provides functionality for (*i*) advanced search in *large, heterogeneous, multidisciplinary digital archives*; (*ii*) collaboration; (*iii*) filtering; (*iv*) recommendation; and (*v*) the management of records grouped into *collections*.

Worth to recall that the main principle underlying CYCLADES is the *folder paradigm*. That is, users and communities of users may organise the working space into their own folder hierarchy, as e. g. may be done with directories in operating systems, bookmark folders in Web browser and folders in e-mail programs. As a consequence the folder becomes a holder of information items, which are usually semantically related and, thus, implicitly determines what the folder's topic is about. On this principle it is based the whole recommendation mechanism constituting one of the biggest value added of CYCLADES as DL. The system automatically notifies each folder with novel information objects, collections, users, and community deemed as relevant with respect to the folder topic [RS02, ACS].

The architecture of the CYCLADES system follows a Service-oriented approach and is depicted in Figure 4.13. Its services can be easily classified in accordance to our Digital Library System Reference Architecture (Section 2.6).

The CYCLADES system is accessible through the CYCLADES *portal* that presents the system functionality via different environments accessible with a web browser provided by the single services. The *Collaborative Work Service*, the *Search & Browse Service*, the *Access Service*, and the *Collection Service* provide their own user interfaces. The CYCLADES portal (actually the user interface of the *Mediator Service*) integrates these user interfaces and ensures that those services are called only for authorised users. Moreover, it provides the registration and login interface, and a system administration interface (for assigning access rights, etc.).

The functionality allowing the CYCLADES services to co-operate are provided by the *Mediator Service*. It represents the main entry point to the CYCLADES system functionality, acts as a registry for the other services, checks if a user is entitled to use the system, and ensures that the other services are only called after proper authentication.

The *DL Management* area (Section 2.6.3) contains the Filtering & Recommendation Service and part of its functionality are covered by the *Collaborative Work Service*; the *Access* area (Section 2.6.2) in mainly composed by the Search & Browse Service and parts of the expected functionality are covered by the Collaborative Work Service; the *Information Space Management* area (Section 2.6.5) is mainly covered by the Collaborative Work Service with the support of the Access Service

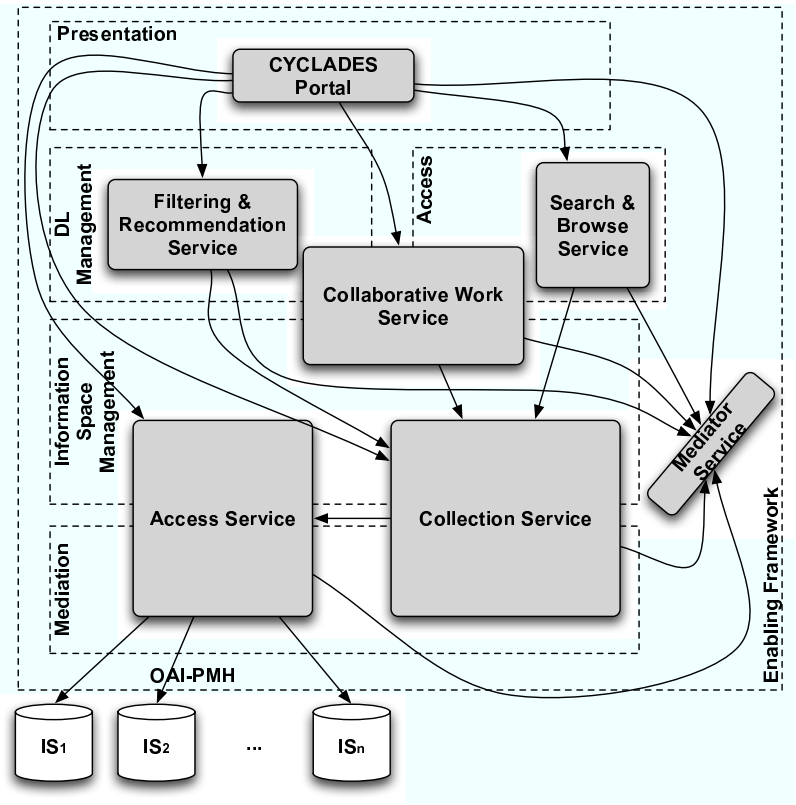

Figure 4.13: CYCLADES Architecture

and the Collection Service.

The *Collaborative Work Service* plays a central role in the system because it provides the folder-based environment for managing metadata records, queries, collections, external documents, received recommendations, ratings and annotations. Furthermore, it supports collaboration among CYCLADES users by way of folder sharing in communities, discussion forums and mutual awareness.

The *Search & Browse Service* supports the activity of searching records from the various collections, formulating and reusing queries associated to the folder by the user, and saving records to folders.

The *Filtering & Recommendation Service* provides filtered search, recommendations of records, collections, users, and communities deemed relevant to the user's interests.

With respect to the concrete access to the information objects, two services constitute the *Mediation* area, i. e. the *Access Service* and the *Collection Service*.

The *Access Service* is in charge of interfacing with the underlying metadata archives. In this prototype, only archives adhering to the OAI specification were accounted for. However, the system is extensible to other kinds of archives by just modifying the Access Service. A user may also ask CYCLADES to include newly OAI compliant archives as well. It is worth noting that this service provides an API

exposing the single archive as they are, i. e. it builds an index for each OAI archive and do not build an index of all the records constituting the information space. Thus, from the other services perspective, the information space is constituted by a set of information sources capable to reply to queries.

The *Collection Service* manages personalised collections (i. e. their definition, creation, and update) and stores them, thus allowing a dynamic partitioning of the information space according to the users' interests, and making the individual archives transparent to the user.

The major outcomes of a service like the CYCLADES Collection Service are (*i*) the users may organise the information space into more meaningful, from their perspective, and dynamic set of information objects, (*ii*) the user may focus their research by partitioning the heterogeneous information space resulting from the aggregation of information sources, and (*iii*) the system is able to recommend an entire collection of information object to the user.

4.7.2 The Cyclades Collection Service: API, GUI and other implementation details

The CYCLADES Collection Service is fully implemented in Java and in particular as a servlet. It builds the mechanisms for source selection on the facilities provided by the Jakarta Lucene[5] for storing retrieved information objects and extracting the needed statistics. The service-call protocol supported and used is the XML-RPC[6]. In order to enhance data portability we have used the XML to represent collection metadata as well as the Membership Condition. We defined the XML scheme that can be used to validate these data [CCP03b]. The service was subjected to, a series of tests to validate its performance as extensively documented in [CCP03b].

Cyclades CS API

The Collection Service provides an API allowing the other CYCLADES services to easily access its functionality. The methods constituting this API are reported in the follow.

- `cId addCollection()`
  This method creates a new collection identifier `cId` which can be assigned to a collection which will be created soon.

- `cId initializeCollection(cId, cName, cDescr, MC, userId)`
  This method creates a collection, whose parent collection is the *Cyclades* collection, if the membership condition `MC` is legal.

[5] `http://jakarta.apache.org/lucene`
[6] `http://www.xmlrpc.com/`

- `cId initializeCollection(cId, cName, cDescr, MC, userId, parentC)`
  This method creates a collection whose parent collection is `parentC`, if the membership condition `MC` is legal.

- `void deleteCollection(collectionId, userId)`
  This method removes a collection from the set of existing collections if: a) the user is authorised to do it and b) the specified collection exists.

- `(cId, cName, cDescr, parentC)* listCollections(userId)`
  This method returns the list of existing collections whose owner is `userId`.

- `(cId, cName, cDescr, parentC)* listCollections()`
  This method returns the list of existing collections.

- `void editCollection(cMetadata, userId)`
  Update collection metadata description.

- `(cId, cMetadata)* getCollectionMetadata(cIds*)`
  For each specified collection identifier, this method returns the corresponding descriptive metadata.

- `(cId, cName, cDescr, parentC)* getPersonalCollections(userId)`
  This method returns the list of personal set of collections for user `userId`.

- `void deleteUser(userId)`
  Notify the Collection Service that user `userId` was removed from the system.

- `void deleteArchive(archiveId)`
  Notify the Collection Service that archive `archiveId` was removed from the system.

Graphical User Interface

The graphical user interface of the CS is accessible via a web-browser. It has been designed keeping in mind the *easy-to-use* concept so it has been organised into two areas, the *menu area* and the *working area* as shown in figure 4.14. Menu area contains a *menu bar* (at the upper) and an *action menu*. Working area contains a *collection hierarchy area* and a *collection data area*.

The topmost part of the interface (under the Collection Management title bar) contains the menu bar with three menus and/or action shortcut.

Via the *Browse* menu the user may choose the set of collections shown in the working area among own created collections and all CYCLADES collections.

Via the *Personal Collections Set* shortcut the user can browse/edit his "personal collection set". Figure 4.15 shows the GUI that allows user to manage his personal collection set. This GUI has a working area a little bit different from the previous, there are two collections hierarchy areas, one (the left) for the "actual" personal

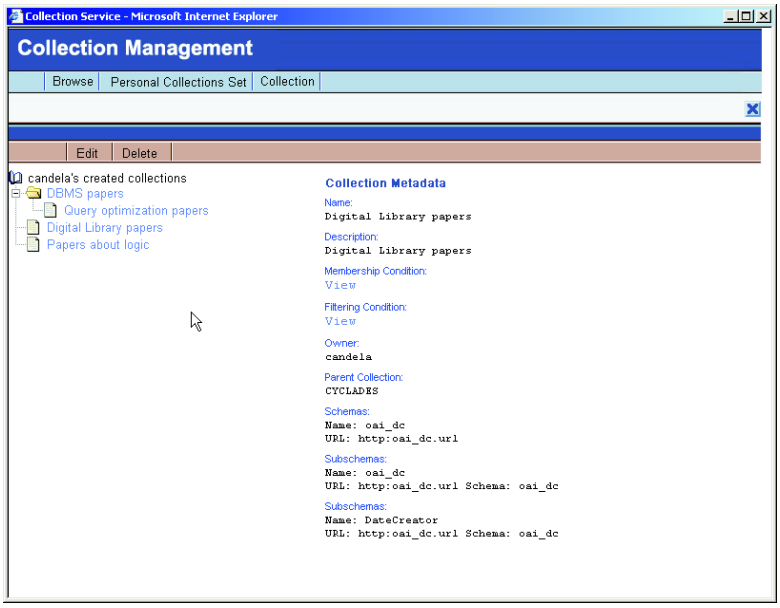

Figure 4.14: CYCLADES CS GUI – The Main View

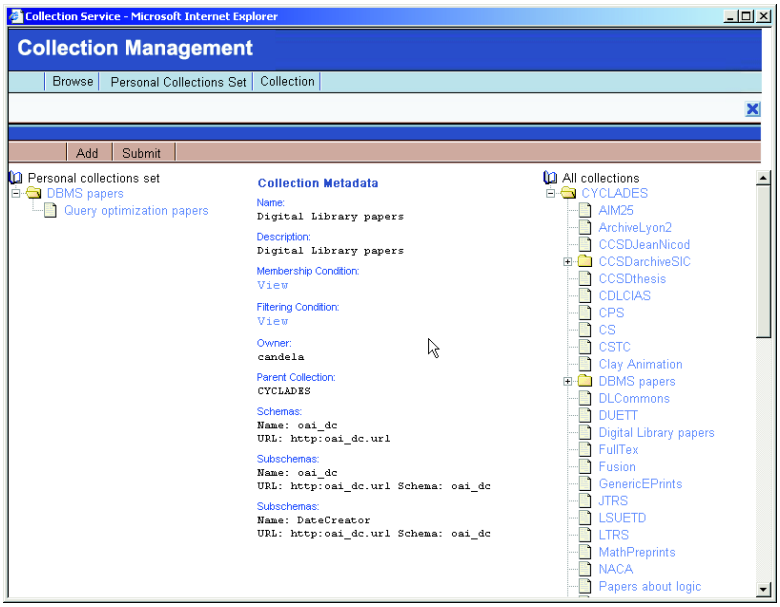

Figure 4.15: CYCLADES CS GUI – The Personal Collections Set

collections set and the other (the right) for all collections. Clicking on a collection in the left area the user can remove this from the actual personal collection set, clicking on a collection in the right area the user can add this from the actual personal collection set. Collection data area in the middle shows collection data (e. g. name, description) for the selected collection.

Via *Collection→New* the user can create a new collection. The system presents a form (Fig. 4.16) to fill in whereto enter the name and the description of the new collection (useful to identify it later), the parent collection (collections may be organised hierarchically), and the membership condition (a CYCLADES query and/or one or more archives). The membership condition is the set of conditions that characterise the set of documents belonging to the collection. Interestingly, the system automatically determines the best source from which to search for (this is the "Retrieval Condition").

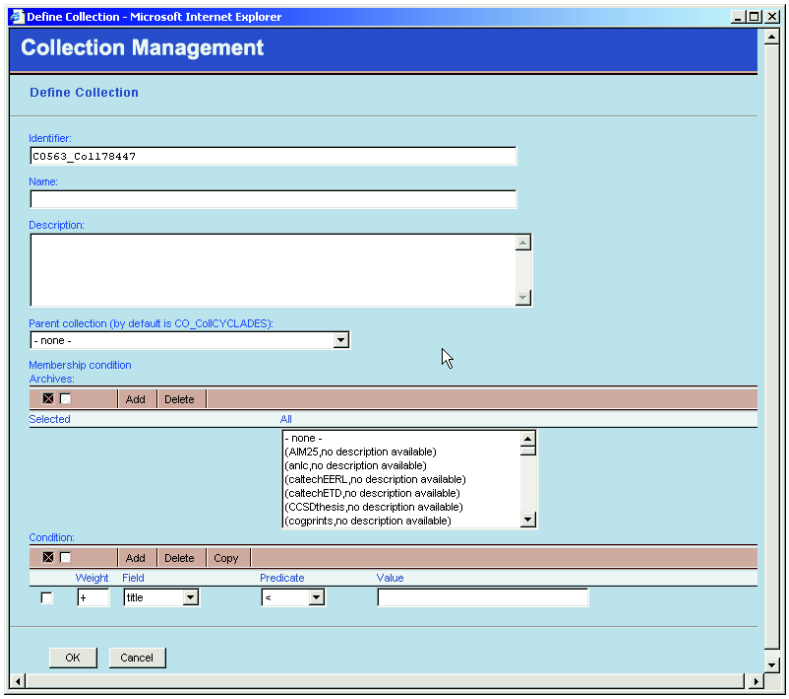

Figure 4.16: CYCLADES CS GUI – Create Collection Form

Under the menu bar the CS interface provides an action menu. The items of this menu are related to the collection shown in the collection data area.

If the collection data area shows a collection created by the user, then the action menu contains the items *Edit*, in order to edit this collection, and *Delete* in order to delete this collection.

On the left of the working area there is the collections hierarchy area. In this area there is a navigable hierarchical view of the set of collections actually in use (own created collections or all collections).

Clicking on a collection allows a user to see collection data in the collection data area and, if the user has the rights, to manage them (via the action menu).

On the right of the working area there is the collection data area. This area shows collection data (e. g. name, description) for the selected collection in the collections hierarchy area.

4.8 Related Work

In the DL field the concept of collection is broad, there is still confusion about what a collection is and what its characteristics are. In many papers, e. g. [JF02, Ber02, WBB01] the term "collection" is used as synonym of information source and the issue is how to automatically populate it. This paper has focused on collections as mechanisms for self-organising the information space that a DL manages. However, we intend a collection as a virtual information source as it does not actually store any documents.

The concept of collection service proposed by Lagoze and Fielding in [LF98] shows many similarities with our CS. The most significant of them are: (*i*) collection membership is defined through a set of criteria rather than containment and (*ii*) CS must supply an independent mechanism for introducing meaningful and dynamic structure into a distributed information space. No implementation of this concept has ever be delivered.

Greenstone [WBB01] propose a collection-centric approach where each collection has a user interface that allows users to search and browse over the collection. This kind of collection is similar to an IS, as the collection creator has to supply the documents belonging to it. This approach is quite static, the collection creator can add documents to a collection but has to do that manually.

In [GGMM02] the term "virtual collection" is introduced and a set of benefits for digital libraries that contain collections are outlined. That paper focuses on how to easily generate collection-level metadata without specifying how collection's documents have been collected and selected.

Many papers have been proposed about source selection in different fields. Xu et. Al. in [XCLN98] proposes a database selection technique called TQRS for resolving the problem of query routing where the ISs are databases with multiple text attributes. That technique uses query sampling in order to acquire database's knowledge and then an extensions of the CVV ranking method [YL97] to rank each database. This is similar to the solution that we have proposed but we have used a revised version of CORI [CLC95] instead of CVV because it is one of the most stable and effective [FPC+99], and it is compatible with resource descriptions acquired by query-sampling, while CVV is not [SC02].

Chapter 5

Semantic Search Across Heterogeneous Information Sources

Search is one of the most critical functionality of a digital library. Often, the quality of the whole DL is measured with respect to the flexibility and efficacy of its search functionality. This functionality usually relies on indexing of either full text or metadata. When indexing is provided by different heterogeneous information sources, each adopting their own metadata formats and ontologies, the implementation of the search requires special approaches to offer homogeneous query languages and results set formats to the end-users.

In this chapter we present an innovative technique for the carrying out of a search functionality across heterogeneous information sources. By exploiting semantic information embedded in the metadata formats and vocabularies used by the different sources, this technique offers a new form of virtualization that supports a better user-query formulation and processing. This technique is based on the introduction of two particular services: *Index* and *Query Mediator*. While the former is mainly a Mediator service and its role is to provide an uniform view of the heterogeneous information space, the latter is a type of search service that acts as *orchestrator*, i. e. it organises the access to the various Index Services thus providing an unique point of access to a search functionality across multiple sources.

The chapter is organised as follows. Section 5.1 shows the limitations of existing techniques for performing search across shared independent information sources. Section 5.2 presents the logical architectural framework that we assumed in this work. Section 5.3 introduces the first part of a theory we have developed and shows how it is exploited by the Index Service. It also provides concrete examples for explaining the implications of the presented theory. Section 5.4 introduces the rest of the theory which is used by the Query Mediator service and shows the advantages obtained. Section 5.5 presents the exploitation of the theoretical framework introduced by previous sections by reporting the experience in building an advanced

search service for the OpenDLib Digital Library System. Finally, Section 5.6 compares our approach to related research.

5.1 Searching Across Information Sources

Institutions maintaining information sources describe their documents using specific cataloguing rules. Even when a standard metadata format is used, the semantic interpretation of the metadata fields and the cataloguing terms used are strongly influenced by the assumptions and terminology of the application context in which any institution operates. When a DL is built using shared information sources, the different cataloguing rules used at the source level become completely transparent to the DL users, who formulate queries that express their information needs in terms of the metadata formats and controlled vocabularies supported by the digital library search service.

This dichotomy between the information source cataloguing environment and the search environment complicates both the formulation and the processing of user queries. As in the DL framework the users neither know how information objects have been originally described nor have access to their original description format, they are not always able to formulate precisely the conditions required to retrieve information objects that satisfy their needs. Most DL search services attempt to minimise this problem by automatically expanding the user query with the help of stemming and query expansion algorithms.

In order to process the user queries the system must be able to map the query conditions against the descriptive metadata of the information objects provided by the different information sources. The most common solution implemented today to carry out this task is to enforce interoperability by requiring to every DL information source provider to expose the descriptions of their information objects in at least a shared common metadata format. This format is usually also the one accepted by the DL search service language. In order to fulfil this requirement, the source provider establishes a mapping between its internally used metadata format(s) and the mandatory metadata format and then it applies this mapping to all the metadata records of its resources. The DL search service thus operates in a context where the metadata descriptions and the query language are homogeneous and can process the query with traditional techniques.

Moreover, current DL systems support both query formulation and processing using techniques based on syntactic manipulations, without exploiting any semantic information about the metadata schemas and controlled vocabularies. One of the reasons of excluding such a solution is the lack of techniques for exploiting it successfully.

In this chapter we introduce a new technique that instead is able to exploit this semantic information. This technique relies on a theory that we have elaborated by modifying the work presented in [TCS01]. this work that focuses on object asso-

ciated with ontologies of terms has been extended in order to apply it to the DL framework where information objects are characterised by both metadata schemas and controlled vocabularies. The proposed technique takes advantage from the specialisation relationships among the metadata fields and among the terms of the controlled vocabularies used. This information is obtained by exploiting the translation relationships that are produced by the information source providers when they transform the local description formats into the common format. This information, usually discarded, is semantically richer than the final common format and can be used for building more powerful search services. The resulting search service is thus able to offer the choice among a range of possible different interpretations for the same query and the users can select the one that better satisfy their needs. Note that this technique does not require any explicit generation of the metadata records in a pre-defined shared format.

The illustrated technique has been experimentally integrated in the OpenDLib [CP02, CP03] search service. This particular application is described in Section 5.5. Before introducing the theory and the services in detail, in the next section we discusses the limitations of the virtualization techniques implemented by the current search services that exploit only syntactic relations.

5.1.1 Motivations

In experimenting digital libraries built by re-using content from heterogeneous sources, we have often encountered situations in which the users could not formulate queries that express their needs and the system was not able to process them properly.

Let us consider a simple DL in which the provider of the information source IS_1 publishes the following metadata records:

	Subject	***Subject.ACM***
doc1	text processing	unspecified
doc2	unspecified	I.7.1 Document and Text Editing

According to the internal rules of the DL institution, the authors can describe their documents by assigning either a code extracted from the ACM Computing Classification System to the field *Subject.ACM* or a free term to the more generic field *Subject*. The records produced are processed by the system in order to extracts the information required to process the user queries.

Imagine now that the user John Smith wants to retrieve exactly those documents that have been described with *Subject* equal to "text processing". The trivial solution is to formulate the following query: "*Subject = text processing*". The search service has only to match the query condition against the information extracted from the metadata records and it usually replies including *doc1* and excluding *doc2*.

Consider now another user of the same DL, Henry Stamp, who is interested in retrieving all the documents about the topic that his community of interest refers as "text processing". Using a traditional search service, this user cannot do anything

better than formulate the same query as that expressed by John Smith. However, the result expected in this case is different. It should include: (*i*) the documents retrieved under the previous more strict interpretation, (*ii*) the documents whose *Subject* contains values morphologically and syntactically close to the query term, e. g. "textual processing" and "documents and text processing", and (*iii*) the documents whose more specific subject, i. e. *Subject.ACM*, contains values that are semantically close to the query term. Under this interpretation the system should, therefore, return not only *doc1*, but also *doc2* since its more specific subject field, Subject.ACM, contains I.7.1 "Documents and Text Editing" which is an ACM subcategory of I.7 "Documents and Text Processing".

While the majority of DL search services that support an interpretation of the query based on automatically extracted morphological and syntactic relationships, e. g. stemming and query expansion, are able to return the documents described in (*i*) and (*ii*)) above, they are not capable to exploit the semantic relationships that exist among the different concepts represented by the metadata fields. This means that the current search services do not usually return documents, like *doc2*, which are indexed under metadata fields that are specialisation of those indicated in the query, i. e. *subject.ACM*.

Despite this example may seem very trivial, it must be remembered that in order to satisfy the requirements of the second user, the query must find *doc2* which has been classified using *a narrower subject field but a broader classification term*. When manipulating complex metadata formats and sophisticated categorisation schemas this kind of document identification is not a simple task.

The limitation described above becomes more incisive in VDLs where the information space is composed by multiple information sources, each describing its documents with different metadata formats. In order to achieve search interoperability over a set of information sources, current DLs often require them to publish their metadata in a shared format, e. g. Dublin Core (DC) [DC]. To adhere to the rules of the DL, each information source provider maps its local format into the shared format. This mapping is done locally by people that have a clear understanding of the semantics associated with the original metadata fields. This information is never transmitted to the DL system that only receives the metadata records in the shared format. The query interpretation made by the system is thus defined *without taking into account the local descriptive interpretations*. This behaviour negatively influences the quality of the DL search service.

To exemplify this point, let us add another information source, IS_2, to our example. It maintains a set of audio-video (A/V) documents of university courses described as in the following example:

	CourseArea	***CourseTopic***	***AudioVideoSubject***
doc3	Computing Methodologies	Text processing	Document Management

where *AudioVideoSubject* is the subject of the A/V document, i. e. the subject of a specific course lecture, *CourseTopic* is the topic of the course, and *CourseArea*

is the course research area. Following this semantics, the A/V document, being a course element, which belongs to a specific area, is also implicitly classified under the subject of the course and the subject of the area.

Suppose now that DC is the common metadata format. The institution that maintains IS_1 maps both *Subject* and *Subject.ACM* into *dc:subject*, whereas the institution that maintains IS_2 maps only *AudioVideoSubject* to this field. Under this hypothesis, any query interpretation provided by the search service is unable to return *doc3* as a result of the query presented at the beginning of this section even if the query term exactly matches the subject of the course which the video is a part of.

The situations exemplified above, and many others, convinced us that the search functionality implemented so far by DLs are too strict especially if it is applied to the VDL framework where the heterogeneity and the needs for interoperability are stressed. Search services that better satisfy the user needs must be provided. We propose an approach, which can be implemented with reasonable costs, able to exploit, as far as possible, the existing semantic mapping among the document description terminologies.

5.2 The Architectural Framework

Figure 5.1 shows a logical DL architectural framework for our approach. The information space of the DL is built by aggregating a number of independent heterogeneous Information Sources IS_1, IS_2, ..., IS_n that disseminate the metadata records of their information objects in one or more formats. These records are indexed by specific services, the Index Services. For simplicity, we assume that records of different Information Sources in different formats are indexed by separate Index services[1]. An Index processes queries formulated according to the same terminology, i. e. metadata format and controlled vocabularies, used for the indexed records. We assume that this terminology and the corresponding semantic descriptions are known to the Index, i. e. the Index has access to the schemas that specify the metadata format and the controlled vocabularies associated with the metadata fields. For simplicity, we also assume that all the Index services accept the same query structure and relational operators.

An Index service supports different interpretations of the same query condition. Each interpretation is characterised by a different level of precision given to the condition. For example, the different intended semantics given by John Smith and Henry Stamp in the query "*subject = text processing*" mentioned in the previous section are two different interpretations of this condition.

The DL user queries are actually not directly evaluated by the Index services but are first processed by the Query Mediator service. The role of this service

[1]This assumption is only given for simplicity of exposition, it does not compromise the generality of the solution.

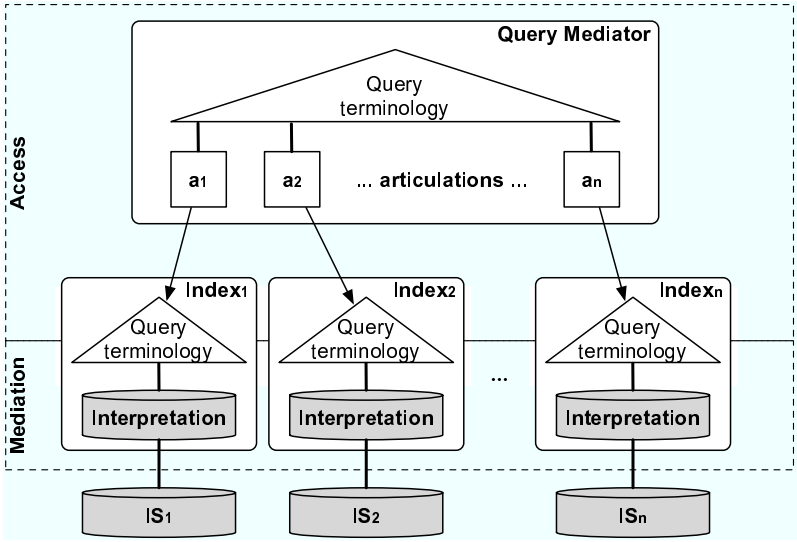

Figure 5.1: The Distributed Search Architectural Framework

is to hide the heterogeneity caused by the great number of Index Services. The Query Mediator serves search operations formulated in terms of the query terminology that is shown to the user[2]. It first maps the queries received by the user into queries formulated in the terminology of the underlying information sources, then it dispatches them to the Index services and, finally it merges the results received. The mapping is done by exploiting the knowledge of specific semantic relationships between the handled terminology and the local indexed terminologies. These relationships are defined by the Information Source providers and they are stored by the corresponding Index Services. It is worth noting the difference in terms of requirements for engagement that our approach imposes with respect to other approaches. Protocols like OAI-PMH [OAI], a simple protocol underlying the Open Archive Initiative [LV01][3], require that any information source provides at least a common DC metadata description of its items. In order to adhere to this protocol, each information source provider must first define the mapping between its local metadata format and DC, and then generate the DC records. Our approach makes less demand on the information source providers than OAI-PHM since it only requires the mapping, while it does not need the explicit generation of the records in the agreed common format. The Query Mediator, similarly to the Index, can support different mapping modalities. The choice of which mapping to apply depends on the query interpretation that is required by the user.

The next two sections introduce our approach from the theoretical point of

[2]A DL can also offer search operations defined on more than one terminology. This situation can be handled by introducing a Query Mediator for each of these terminologies

[3]The focus of this protocol is "open" the archives from the architectural perspective defining and promoting interfaces that facilitate the availability of content from a variety of providers.

view. The solution proposed is based on the theory introduced by Tzitzikas et Al. in [TCS01]. Following the terminology introduced by Tzitzikas et al., we propose a theory that applies to our framework composed by metadata schemas and controlled vocabularies. In particular, we specify the different query interpretations that can be supported by the Index and Query Mediator services and how they are obtained by the existing terminology mappings.

5.3 The Index

Each information source uses a metadata schema to describe its own documents. This metadata schema is a pair $(\mathcal{F}, \leq_{\mathcal{F}})$ where $\mathcal{F}$ is a set of schema fields and $\leq_{\mathcal{F}}$ is a *subsumption* relation over $\mathcal{F}$[4] that models the existing specialisation relationship among these fields. For example in Figure 5.2, `Subject.ACM` $\leq_{\mathcal{F}}$ `Subject` means that `Subject.ACM` is a more specialised property than `Subject`. Each field f of the schema is populated via an appropriate *terminology* defined as a pair $(\mathcal{V}_f, \leq_{\mathcal{V}_f})$ where $\mathcal{V}_f$ is a set of terms and $\leq_{\mathcal{V}_f}$ is a subsumption relation over $\mathcal{V}_f$ that models the existing specialisation relationship among these terms. For example, in Figure 5.2 `Multimedia DL` $\leq_{\mathcal{V}_f}$ `DL` means that `Multimedia DL` is a more specialised term than `DL`. In certain cases the latter assumption is too *strong*. A field is often populated via free terms or free text. In these cases, the terminology can easily and automatically be obtained considering that each term is in relation only with itself or, if we are going to use stemming, we can assume that the term t is subsumed by the stemmed term t'.

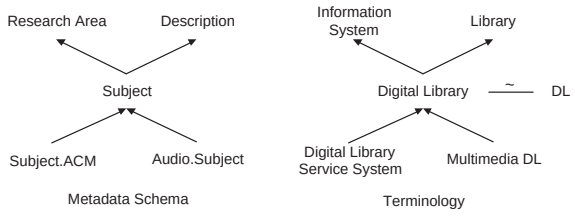

Figure 5.2: A Metadata Schema and a Terminology

Combining the metadata schema with the set of terminologies $\mathcal{V}_f$[5] that the Index uses, one for each field of the schema, we can define the *query terminology* that the Index "speaks" as a pair $(\mathcal{C}, \leq_{\mathcal{C}})$, where $\mathcal{C}$ is a set of *conditions* or pairs (f, v) such that $f \in \mathcal{F}$, $v \in \mathcal{V}_f$. The latter pair models the boolean condition "field f equals term v". For example, a valid condition for the Index in Figure 5.2 is (`Subject`, `Digital Library`) representing the information need expressed as "the documents whose *Subject* is *Digital Library*".

[4]Each subsumption relation $\leq$ is a *reflexive* and *transitive* relation over the reference universe. We write $o_1 \sim o_2$ meaning that the two objects are *equivalent* w. r. t. $\leq$ if both $o_1 \leq o_2$ and $o_2 \leq o_1$.

[5]We will use $\mathcal{V}_f$ instead of $(\mathcal{V}_f, \leq_{\mathcal{V}_f})$ were no confusion arises.

The subsumption relation over $\mathcal{C}$, $\leq_{\mathcal{C}}$, models the specialisation among these conditions and is formally defined as follows:

Definition 5.3.1 (Subsumption relation) *Let $(\mathcal{F}, \leq_{\mathcal{F}})$ be a metadata schema. Let $(\mathcal{V}_f, \leq_{\mathcal{V}_f})$ be the terminology for the field f of the schema. Given $c_1, c_2 \in \mathcal{C}$ where $c_i = (f_i, v_i)$, $f_i \in \mathcal{F}$ and $v_i \in \mathcal{V}_{f_i}$ we define $c_1 \leq_{\mathcal{C}} c_2 \iff f_1 \leq_{\mathcal{F}} f_2 \wedge v_1 = v_2$.*

Considering Figure 5.2 we are saying, for example, that (`subject.ACM`, `DLSS`) $\leq_{\mathcal{C}}$ (`Subject`, `DLSS`) or that (`Audio.Subject`, `Library`) $\leq_{\mathcal{C}}$ (`Research Area`, `Library`) meaning that the first condition is a specialisation of the second one in each of the exemplified cases.

As $\leq_{\mathcal{C}}$ is a subsumption relation over $\mathcal{C}$ we can define the equivalence relation w. r. t. $\leq_{\mathcal{C}}$ as $c_1 \leq_{\mathcal{C}} c_2$ and $c_2 \leq_{\mathcal{C}} c_1$ and we will write $c_1 \sim_{\mathcal{C}} c_2$. Theorem 5.3.1 follows from Definition 5.3.1:

Theorem 5.3.1 (Equivalence among conditions) *For each $c_1, c_2 \in \mathcal{C}$ where $c_i = (f_i, v_i)$*

$$c_1 \sim_{\mathcal{C}} c_2 \iff f_1 \sim_{\mathcal{F}} f_2 \wedge v_1 = v_2$$

Proof. For each pair of objects $o_1, o_2 \in U$, where U is a generic set of object, and subsumption relation $\leq$ over U hold that $o_1 \sim o_2 \iff o_1 \leq o_2 \wedge o_2 \leq o_1$. As a consequence $c_1 \sim_{\mathcal{C}} c_2 \iff c_1 \leq_{\mathcal{C}} c_2 \wedge c_2 \leq_{\mathcal{C}} c_1$. From Definition 5.3.1 follows that $c_1 \leq_{\mathcal{C}} c_2 \iff f_1 \leq_{\mathcal{F}} f_2 \wedge v_1 = v_2$ and $c_2 \leq_{\mathcal{C}} c_1 \iff f_2 \leq_{\mathcal{F}} f_1 \wedge v_2 = v_1$. Observing that $f_2 \leq_{\mathcal{F}} f_1 \wedge f_1 \leq_{\mathcal{F}} f_2 \iff f_1 \sim_{\mathcal{F}} f_2$ we conclude $c_1 \leq_{\mathcal{C}} c_2 \wedge c_2 \leq_{\mathcal{C}} c_1 \iff f_1 \sim_{\mathcal{F}} f_2 \wedge v_1 = v_2$. ∎

In the remaining text we will write $c_i \nsim_{\mathcal{C}} c_j$ meaning that c_i is not equivalent to c_j, i. e. $\neg(c_i \sim_{\mathcal{C}} c_j)$.

A query for the Index is either a simple condition or a combination of conditions using the classical connectives $\wedge, \vee, \neg$ and is formally defined as follows:

Definition 5.3.2 (Query) *Let $\mathcal{C}$ be a query terminology, and $c \in \mathcal{C}$. A query is any expression derived by the following BNF grammar:*

$$Q ::= c \mid Q \wedge Q \mid Q \vee Q \mid \neg Q$$

For example, a simple query can be (`subject`, `Digital Library`) $\vee$ (`Description`, `Library`).

Definition 5.3.3 (Interpretation) *An interpretation I of a query terminology $\mathcal{C}$ is a function $I : \mathcal{C} \to 2^{Obj}$ that associates each condition of $\mathcal{C}$ with a set of objects of the domain.*

Each Index has an *interpretation I* that is the result of the indexing phase. Table 5.1 presents an interpretation of the Index presented in Figure 5.2[6].

[6]For simplicity, we will use the same terminology to populate all the schema fields.

Field	Value	Documents
Subject	Digital Library	$\{d_1\}$
	DL	$\{d_2\}$
	Information System	$\{d_5\}$
	Library	$\{d_6\}$
Subject.ACM	DLSS	$\{d_3\}$
Audio.Subject	Information System	$\{d_4\}$
	Library	$\{d_4\}$
Research Area	DL	$\{d_7\}$
	DLSS	$\{d_8\}$
	Information System	$\{d_9\}$
	Library	$\{d_9\}$
	Digital Library	$\{d_{10}\}$
Description	DL	$\{d_7, d_{11}\}$
	Multimedia DL	$\{d_8\}$
	Information System	$\{d_9\}$
	Library	$\{d_9\}$

Table 5.1: A Stored Interpretation

The interpretation that an Index uses for query evaluation must comply with the structure of the query terminology (i. e. $\leq_{\mathcal{C}}$). This requirement is expressed by introducing the notion of *model*.

Definition 5.3.4 (Model) *An interpretation I is a model of a query terminology if $\forall\ c_1, c_2 \in \mathcal{C}$ where $c_i = (f_i, v_i)$, $c_1 \leq_{\mathcal{C}} c_2 \Rightarrow I(c_1) \subseteq I(c_2)$ and $f_1 = f_2 \wedge v_1 \leq_{\mathcal{V}_f} v_2 \Rightarrow I(c_1) \subseteq I(c_2)$.*

For example, suppose that an Index has indexed a set of documents under the condition c_1 and another set of documents under the condition c_2 and no document under the condition c that subsumes the previous two conditions. This interpretation is acceptable as we can "respect" the structure of $\leq_{\mathcal{C}}$ by defining the interpretation of c as the union of the set of documents indexed under c_1 and the set of those indexed under c_2. Note that it is always possible to generate a model from an interpretation by extending the interpretation of the conditions that do not comply with the terminology. The smallest model generated from an interpretation is the one used to answer queries.

For technical reasons we assume that every query terminology $\mathcal{C}$ contains two special queries, the *top query* $\top_{\mathcal{C}}$ and the *bottom query* $\bot_{\mathcal{C}}$. These two queries have the following properties: the *top query* subsumes every other query, i. e. $\forall c \in \mathcal{C} : c \leq_{\mathcal{C}} \top_{\mathcal{C}}$, while the *bottom query* is strictly subsumed by every other query different from $\top_{\mathcal{C}}$ and $\bot_{\mathcal{C}}$, i. e. $\forall c \in \mathcal{C} : c \neq \top_{\mathcal{C}} \wedge c \neq \bot_{\mathcal{C}} \Rightarrow \bot_{\mathcal{C}} <_{\mathcal{C}} c$. Moreover we assume that every model I of $\mathcal{C}$ satisfies the condition $I(\bot_{\mathcal{C}}) = \emptyset$. For the same reason, we assume that (a) every metadata schema $\mathcal{F}$ contains the special fields *top field* $\top_{\mathcal{F}}$ and *bottom field* $\bot_{\mathcal{F}}$, and (b) every terminology $\mathcal{V}_f$ contains the same special fields *top term* $\top_{\mathcal{V}}$ and *bottom term* $\bot_{\mathcal{V}}$.

As there may be several models of $\mathcal{C}$, we assume that each Index is able to process queries from one or more models of its interpretation. In this paper, we will use two families of models for query processing, the *sure evaluation models* and the

possible evaluation models. In order to define these models formally we need two preliminary definitions: the first one allows us to follow the subsumption relation over the fields of the metadata schema, while the second one allows to follow the subsumption relation over the terminologies.

Definition 5.3.5 (Tail and Head) *Given a condition $c \in \mathcal{C}$, $c = (f, v)$, we define*

$$\begin{aligned} tail(c) &= \{c' \in \mathcal{C} | c' \leq_{\mathcal{C}} c\} \\ head(c) &= \{c' \in \mathcal{C} | c \leq_{\mathcal{C}} c'\} \end{aligned}$$

Intuitively, $tail(c)$ and $head(c)$ contain c and, respectively, all the conditions that are stricter than c and wider than c according to the query terminology and, in particular, to the subsumption relations over the schema fields. For example, considering Figure 5.2, $tail$(`subject`, `DL`)={(`subject`, `DL`), (`subject.ACM`, `DL`), (`Audio.subject`, `DL`)} while $head$(`subject`, `DL`)={(`subject`, `DL`), (`Research Area`, `DL`), (`Description`, `DL`)}.

These definitions can be reformulated considering the Definition 5.3.1 as follows:

$$\begin{aligned} tail(c) &= \{c' \in \mathcal{C} | f' \leq_{\mathcal{F}} f \wedge v' = v\} \\ head(c) &= \{c' \in \mathcal{C} | f \leq_{\mathcal{F}} f' \wedge v = v'\} \end{aligned}$$

Definition 5.3.6 (Value models) *Given an interpretation I of $\mathcal{C}$ and a condition $c \in \mathcal{C}$, $c = (f, v)$, we define three kinds of value models for c generated by I as follows:*

$$\begin{aligned} I^{\mathcal{V}}_{\sim}(c) &= \bigcup \{I(c') | f = f' \wedge v' \sim_{\mathcal{V}_f} v\} \\ I^{\mathcal{V}}_{\leq}(c) &= \bigcup \{I(c') | f = f' \wedge v' \leq_{\mathcal{V}_f} v\} \\ I^{\mathcal{V}}_{\geq}(c) &= \bigcap \{I^{\mathcal{V}}_{\leq}(c') | f = f' \wedge v \leq_{\mathcal{V}_f} v' \wedge v \nsim_{\mathcal{V}_f} v'\} \end{aligned}$$

The above interpretations correspond to three different ways in which the Index can evaluate a condition that involves the field f using the stored interpretations and the semantic information about the terminology. These interpretations correspond to the set of documents considered indexed under conditions involving the field f and, respectively, the value v or values equivalent to v ($I^{\mathcal{V}}_{\sim}$), the value v or values subsumed by v ($I^{\mathcal{V}}_{\leq}$), and all the values that subsume v ($I^{\mathcal{V}}_{\geq}$).

Theorem 5.3.2 (Relationship among value models) *If I is a model for a query terminology then $I^{\mathcal{V}}_{\sim}$, $I^{\mathcal{V}}_{\leq}$ and $I^{\mathcal{V}}_{\geq}$ are models and $I^{\mathcal{V}}_{\sim} \subseteq I^{\mathcal{V}}_{\leq} \subseteq I^{\mathcal{V}}_{\geq}$.*

Proof. The proof that $I^{\mathcal{V}}_{\sim}$, $I^{\mathcal{V}}_{\leq}$, $I^{\mathcal{V}}_{\geq}$ are models is trivial and follows from Definition 5.3.6.

Let $c_1 = (f_1, v_1)$ and $c_2 = (f_2, v_2)$ and $c_1 \leq_{\mathcal{C}} c_2$, i. e. $f_1 \leq_{\mathcal{F}} f_2 \wedge v_1 = v_2$:

$$\begin{aligned}
I^{\mathcal{V}}_{\sim}(c_1) &\stackrel{def}{=} \bigcup \left\{ I(c') | f_1 = f' \wedge v' \sim_{\mathcal{V}_f} v_1 \right\} \subseteq \\
&\quad \bigcup \left\{ I(c') | f_2 = f' \wedge v' \sim_{\mathcal{V}_f} v_2 \right\} \stackrel{def}{=} I^{\mathcal{V}}_{\sim}(c_2) \text{ as } f_1 \leq_{\mathcal{F}} f_2 \\
I^{\mathcal{V}}_{\leq}(c_1) &\stackrel{def}{=} \bigcup \left\{ I(c') | f_1 = f' \wedge v' \leq_{\mathcal{V}_f} v_1 \right\} \subseteq \\
&\quad \bigcup \left\{ I(c') | f_2 = f' \wedge v' \leq_{\mathcal{V}_f} v_2 \right\} \stackrel{def}{=} I^{\mathcal{V}}_{\leq}(c_2) \text{ as } f_1 \leq_{\mathcal{F}} f_2 \\
I^{\mathcal{V}}_{\geq}(c_1) &\stackrel{def}{=} \bigcap \left\{ I^{\mathcal{V}}_{\leq}(c') | f_1 = f' \wedge v_1 \leq_{\mathcal{V}_f} v' \wedge v_1 \nsim_{\mathcal{V}_f} v' \right\} \subseteq \\
&\quad \bigcap \left\{ I^{\mathcal{V}}_{\leq}(c') | f_2 = f' \wedge v_2 \leq_{\mathcal{V}_f} v' \wedge v_2 \nsim_{\mathcal{V}_f} v' \right\} \stackrel{def}{=} I^{\mathcal{V}}_{\geq}(c_2) \text{ as } f_1 \leq_{\mathcal{F}} f_2
\end{aligned}$$

Let $c_1 = (f_1, v_1)$ and $c_2 = (f_2, v_2)$ and $f_1 = f_2 \wedge v_1 \leq_{\mathcal{V}_f} v_2$:

$$\begin{aligned}
I^{\mathcal{V}}_{\sim}(c_1) &\stackrel{def}{=} \bigcup \left\{ I(c') | f_1 = f' \wedge v' \sim_{\mathcal{V}_f} v_1 \right\} \subseteq \\
&\quad \bigcup \left\{ I(c') | f_2 = f' \wedge v' \sim_{\mathcal{V}_f} v_2 \right\} \stackrel{def}{=} I^{\mathcal{V}}_{\sim}(c_2) \text{ as } v_1 \leq_{\mathcal{V}_f} v_2. \\
I^{\mathcal{V}}_{\leq}(c_1) &\stackrel{def}{=} \bigcup \left\{ I(c') | f_1 = f' \wedge v' \leq_{\mathcal{V}_f} v_1 \right\} \subseteq \\
&\quad \bigcup \left\{ I(c') | f_2 = f' \wedge v' \leq_{\mathcal{V}_f} v_2 \right\} \stackrel{def}{=} I^{\mathcal{V}}_{\leq}(c_2) \text{ as } v_1 \leq_{\mathcal{V}_f} v_2. \\
I^{\mathcal{V}}_{\geq}(c_1) &\stackrel{def}{=} \bigcap \left\{ I^{\mathcal{V}}_{\leq}(c') | f_1 = f' \wedge v_1 \leq_{\mathcal{V}_f} v' \wedge v_1 \nsim_{\mathcal{V}_f} v' \right\} \subseteq \\
&\quad \bigcap \left\{ I^{\mathcal{V}}_{\leq}(c') | f_2 = f' \wedge v_2 \leq_{\mathcal{V}_f} v' \wedge v_2 \nsim_{\mathcal{V}_f} v' \right\} \stackrel{def}{=} I^{\mathcal{V}}_{\geq}(c_2) \text{ as } v_1 \leq_{\mathcal{V}_f} v_2.
\end{aligned}$$

In order to prove that $I^{\mathcal{V}}_{\sim} \subseteq I^{\mathcal{V}}_{\leq}$ we can just observe that $\forall c = (f, v)$, $\left\{ c' | f = f' \wedge v' \sim_{\mathcal{V}_f} v \right\} \subseteq \left\{ c' | f = f' \wedge v' \leq_{\mathcal{V}_f} v \right\}$ and that I is a model.

Let us prove that $I^{\mathcal{V}}_{\leq} \subseteq I^{\mathcal{V}}_{\geq}$. $I^{\mathcal{V}}_{\geq}(c) \stackrel{def}{=} \bigcap \left\{ I^{\mathcal{V}}_{\leq}(c') | f = f' \wedge v \leq_{\mathcal{V}_f} v' \wedge v \nsim_{\mathcal{V}_f} v' \right\}$. As $I^{\mathcal{V}}_{\leq}$ is a model, it holds that $\forall c', I^{\mathcal{V}}_{\leq}(c) \subseteq I^{\mathcal{V}}_{\leq}(c')$, so we can conclude $I^{\mathcal{V}}_{\leq} \subseteq I^{\mathcal{V}}_{\geq}$. ■

By exploiting the definitions given above, we can now define the *sure evaluation model* and the *possible evaluation model* of the stored interpretation I. These are obtained taking into account the subsumption relations among the schema fields and the subsumption relations among terminologies.

Definition 5.3.7 (Sure models) *Given an interpretation I of $\mathcal{C}$ we define three kinds of sure evaluation models of $\mathcal{C}$ generated by I as follows:*

$$\begin{aligned}
I^{-}_{\sim}(c) &= \bigcup \left\{ I^{\mathcal{V}}_{\sim}(c') | c' \in tail(c) \right\} \\
I^{-}_{\leq}(c) &= \bigcup \left\{ I^{\mathcal{V}}_{\leq}(c') | c' \in tail(c) \right\} \\
I^{-}_{\geq}(c) &= \bigcup \left\{ I^{\mathcal{V}}_{\geq}(c') | c' \in tail(c) \right\}
\end{aligned}$$

Theorem 5.3.3 (Relationship among sure models) *If I is a model then $I^{-}_{\sim}$, $I^{-}_{\leq}$ and $I^{-}_{\geq}$ are models and $I^{-}_{\sim} \subseteq I^{-}_{\leq} \subseteq I^{-}_{\geq}$.*

Proof. The proof that the sure evaluation models I^{-}_{*} are models is quite trivial. Let $c_1 = (f_1, v_1)$ and $c_2 = (f_2, v_2)$ and $c_1 \leq_{\mathcal{C}} c_2$, i. e. $f_1 \leq_{\mathcal{F}} f_2 \wedge v_1 = v_2$:
$I^{-}_{*}(c_1) \stackrel{def}{=} \bigcup \left\{ I^{\mathcal{V}}_{*}(c') | c' \in tail(c_1) \right\} \subseteq \bigcup \left\{ I^{\mathcal{V}}_{*}(c') | c' \in tail(c_2) \right\} \stackrel{def}{=} I^{-}_{*}(c_2)$ as $tail(c_1) \subseteq tail(c_2)$.

Let $c_1 = (f_1, v_1)$ and $c_2 = (f_2, v_2)$ and $f_1 = f_2 \wedge v_1 \leq_{\mathcal{V}_f} v_2$:
$I^-_*(c_1) \stackrel{def}{=} \bigcup \{I^{\mathcal{V}}_*(c') | c' \in tail(c_1)\} \subseteq \bigcup \{I^{\mathcal{V}}_*(c') | c' \in tail(c_2)\} \stackrel{def}{=} I^-_*(c_2)$ as $tail(c_1) \subseteq tail(c_2)$.

The proof that $I^-_\sim \subseteq I^-_\leq \subseteq I^-_\geq$ is a trivial consequence of their definitions and of Theorem 5.3.2 that states $I^{\mathcal{V}}_\sim \subseteq I^{\mathcal{V}}_\leq \subseteq I^{\mathcal{V}}_\geq$. ∎

Definition 5.3.8 (Possible models) *Given an interpretation I of $\mathcal{C}$ we define three kinds of possible evaluation models of $\mathcal{C}$ generated by I as follows:*

$$\begin{aligned} I^+_\sim(c) &= \bigcap \{I^-_\sim(c') | c' \in head(c) \wedge c' \nsim_{\mathcal{C}} c\} \\ I^+_\leq(c) &= \bigcap \{I^-_\leq(c') | c' \in head(c) \wedge c' \nsim_{\mathcal{C}} c\} \\ I^+_\geq(c) &= \bigcap \{I^-_\geq(c') | c' \in head(c) \wedge c' \nsim_{\mathcal{C}} c\} \end{aligned}$$

Theorem 5.3.4 (Relationship among possible models) *If I is a model then $I^+_\sim$, $I^+_\leq$ and $I^+_\geq$ are models and $I^+_\sim \subseteq I^+_\leq \subseteq I^+_\geq$.*

Proof. In order to prove that the possible evaluation models I^+_* are models we can just observe that $\forall c_1 = (f_1, v_1), c_2 = (f_2, v_2)$, if $c_1 \leq_{\mathcal{C}} c_2 \vee (f_1 = f_2 \wedge v_1 \leq_{\mathcal{V}_f} v_2)$ then $head(c_1) \subseteq head(c_2)$.

The proof that $I^+_\sim \subseteq I^+_\leq \subseteq I^+_\geq$ is a trivial consequence of their definitions and of Theorem 5.3.2 that states $I^{\mathcal{V}}_\sim \subseteq I^{\mathcal{V}}_\leq \subseteq I^{\mathcal{V}}_\geq$. ∎

Theorem 5.3.5 (Relationship among sure and possible models) *If I is a model then the following relationships hold between sure and possible models:*

$$I^-_\sim \subseteq I^+_\sim \qquad I^-_\leq \subseteq I^+_\leq \qquad I^-_\geq \subseteq I^+_\geq$$

Proof. All the relationships will be proved in the same way. First of all, we can observe that possible models are defined in terms of sure models, i. e. $I^+_*(c) = \bigcap \{I^-_*(c') | c' \in head(c) \wedge c' \nsim_{\mathcal{C}} c\}$. For each $c' \in \{c' | c' \in head(c) \wedge c' \nsim_{\mathcal{C}} c\}$ it holds that $I^-_*(c) \subseteq I^-_*(c')$ as $c \leq_{\mathcal{C}} c'$, so $I^-_*(c) \subseteq I^+_*(c)$. ∎

Table 5.2[7] shows the interpretation models of our Index that uses the terminology in Figure 5.2 and the stored interpretation in Table 5.1.

We have stated that an Index stores its interpretation I. Our approach allows us to observe that, even if the indexing phase is correct, certain documents may not have been indexed under all the conditions that could apply to them. So, given a simple query c, we may want the source to be able to answer including either all

[7] In this table we have used i referring to d_i of Table 5.1, e. g. 1 is d_1.

Condition	I	$I^-_\sim$	$I^-_\leq$	$I^-_\geq$	$I^+_\sim$	$I^+_\leq$
$\bot_\mathcal{C}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
(Subject,Digital Library)	{1}	{1,2}	{1,2,3}	{1,2,3,4}	{1,2,7}	{1,2,3,7,8}
(Subject,DL)	{2}	{1,2}	{1,2,3}	{1,2,3,4}	{1,2,7}	{1,2,3,7,8}
(Subject,Info. Sys.)	{5}	{4,5}	{1,2,3,4,5}	{1,2,3,4,5,6}	{4,5,9}	{1,2,3,4,5,7,8,9}
(Subject,Library)	{6}	{4,6}	{1,2,3,4,6}	{1,2,3,4,5,6}	{4,6,9}	{1,2,3,4,6,7,8,9}
(Subject.ACM,DLSS)	{3}	{3}	{3}	{3}	{3}	{3}
(Audio.Subject,Info. Sys.)	{4}	{4}	{4}	{4}	{4,5}	{1,2,3,4,5}
(Audio.Subject,Library)	{4}	{4}	{4}	{4}	{4,6}	{1,2,3,4,6}
(Research Area,DL)	{7}	{1,2,7,10}	{1,2,3,7,8,10}	{1,2,3,7,8,9,10}	{1,2,7,10,11}	{1,2,3,7,8,10,11}
(Research Area,DLSS)	{8}	{3,8}	{3,8}	{3,8}	{3,8}	{3,8}
(Research Area,Info. Sys.)	{9}	{4,5,9}	{1,2,3,4,5,7,8,9,10}	{1,2,3,4,5,6,7,8,9,10}	{4,5,9}	{1,2,3,4,5,7,8,9,10,11}
(Research Area,Library)	{9}	{4,6,9}	{1,2,3,4,6,7,8,9,10}	{1,2,3,4,5,6,7,8,9,10}	{4,6,9}	{1,2,3,4,6,7,8,9,10,11}
(Research Area,Dig. Lib.)	{10}	{1,2,7,10}	{1,2,3,7,8,10}	{1,2,3,4,7,8,9,10}	{1,2,7,10,11}	{1,2,3,7,8,10,11}
(Description,Multimedia DL)	{8}	{8}	{8}	{1,2,3,7,8,11}	{8}	{8}
(Description,DL)	{7,11}	{1,2,7,11}	{1,2,3,7,8,11}	{1,2,3,4,7,8,9,11}	{1,2,7,10,11}	{1,2,3,7,8,10,11}
(Description,Info. Sys.)	{9}	{4,5,9}	{1,2,3,4,5,7,8,9,11}	{1,2,3,4,5,6,7,8,9,11}	{4,5,9}	{1,2,3,4,5,7,8,9,10,11}
(Description,Library)	{9}	{4,6,9}	{1,2,3,4,6,7,8,9,11}	{1,2,3,4,5,6,7,8,9,11}	{4,6,9}	{1,2,3,4,6,7,8,9,10,11}

Table 5.2: Interpretations of an Information Source Index

the documents that are known to be indexed under c or all the documents that are possibly indexed under c. In the first case we are considering the sure evaluation model while in the latter case we are considering the possible evaluation model.

Referring to Definition 5.3.2, we define the query answering as follows:

Definition 5.3.9 (Sure and Possible Query answering) *Let q be a query over $\mathcal{C}$ and let I be an interpretation of $\mathcal{C}$. The sure answer $I^-_\leq(q)$ and the possible answer $I^+_\leq(q)$ are defined as follows:*

$$\begin{aligned}
I^-_\leq(c) &= \bigcup \{ I^\mathcal{V}_\leq(c') | c' \in tail(c) \} \\
I^-_\leq(q \wedge q') &= I^-_\leq(q) \cap I^-_\leq(q') \\
I^-_\leq(q \vee q') &= I^-_\leq(q) \cup I^-_\leq(q') \\
I^-_\leq(\neg q) &= \overline{I^-_\leq(q)} \\
I^+_\leq(c) &= \bigcap \{ I^-_\leq(c') | c' \in head(c) \wedge c' \nsim_\mathcal{C} c \} \\
I^+_\leq(q \wedge q') &= I^+_\leq(q) \cap I^+_\leq(q') \\
I^+_\leq(q \vee q') &= I^+_\leq(q) \cup I^+_\leq(q') \\
I^+_\leq(\neg q) &= \overline{I^+_\leq(q)}
\end{aligned}$$

where we use $\overline{I}$ to indicate the set-complement operation on the set I. All the other sure and possible answers for the other models, i. e. $I^-_\sim$, $I^-_\geq$, $I^+_\sim$ and $I^+_\geq$, are defined in a similar way.

Each of the above query answering modes represents a modality of query processing. Note that the sure answer is appropriate for users that focus on *precision* while the possible answer is for users that focus on *recall*. Moreover, in both the family of sure answers and that of possible answers, we can distinguish more precision-oriented responses, i. e. $I^-_\sim$, versus more recall-oriented responses, i. e. $I^-_\geq$. An Index that stores an interpretation, like the one given in Table 5.1, and that has access

to the semantics of the metadata schema and its controlled vocabularies, can thus potentially offer a range of additional interpretations, like the ones given in Table 5.2, to any of its clients to express their information needs more precisely.

For example, expressing the query (`Subject`, `DL`) a user could be interested in those documents that have been described using the field `Subject`, or a more specialised one, and the term `DL` or an equivalent term, so this user is asking for $I^{-}_{\sim}$. Another user expressing the same query could be interested, instead, in those documents that have been described using the field `Subject`, or a more generic field, and the term `DL` or an equivalent term, so this user is asking for $I^{+}_{\sim}$. In the case of Table 5.2, the Index will return the set of documents $\{d_1, d_2\}$ to the first user and the set of documents $\{d_1, d_2, d_7\}$ to the second user. Note that while d_1 and d_2 are indexed under the condition (`subject`, `DL`) and (`subject`, `Digital Library`), respectively, the document d_7 is indexed under a pair of conditions, (`Research Area`, `DL`) and (`description`, `DL`), more general but pertinent to the one expressed by the user.

5.4 The Query Mediator

The previous section has described which are the potential query evaluation choices of an Index service that exploits semantic information. Having clarified this point, we can now examine the more general problem of understanding which query evaluation choices can be supported by a Query Mediator service. In what follows we will assume that such kind of mediator dispatches queries to Index services that behave as described in the previous section.

Abstractly a Query Mediator service can be considered as an Index service that *virtually stores* all the objects of the underlying sources and supplies a query language that satisfies the needs of its users community.

However, there is an important difference between a Query Mediator and an Index: the Query Mediator does not store explicitly any interpretation of the information space. Such interpretations are maintained by the Index services. The Query Mediator only stores an *articulation* for each source, i. e. a set of relationships among the terminology of the Mediator and the terminology of the Index.

A Query Mediator is formally defined as follows:

Definition 5.4.1 (Query Mediator) *A Query Mediator over n Index services $I_1, \ldots, I_n$, such that $I_i = (\mathcal{C}_i, \leq_{\mathcal{C}_i})$, consists of:*

1. *a query terminology $(\mathcal{C}_M, \leq_{\mathcal{C}_M})$ and*
2. *a set of articulations a_i, one for each Index I_i; each articulation a_i is a subsumption relation over $\mathcal{C}_M \cup \mathcal{C}_i$ which contains:*
   - *a subsumption relation, $\preceq^i_{\mathcal{F}}$, over $\mathcal{F}^M \cup \mathcal{F}^i$, i. e. a set of relationships among the Mediator metadata schema and the Index metadata schema,*

- *a set of subsumption relations, $\preceq^i_{\mathcal{V}_f}$, over $\mathcal{V}^M_f \cup \mathcal{V}^i_{f'}$, i. e. a set of relationships among each field terminology of the Mediator and the corresponding ones in the Index. There exists one of such relation for each pair of (Mediator field terminology, Index field terminology).*

For simplicity, we introduce a special subsumption relation between Mediator and Index field terminologies, Π_f, that is a short-cut to indicate that every term of the first terminology is mapped into the *same* term of the second terminology. In such case we impose that $\mathcal{V}^i_{f'} = \mathcal{V}^M_f$, i. e. the terminology of the Index is the *same* as that of the Mediator, and $\preceq^i_{\mathcal{F}}$ is defined so that for each $v \in \mathcal{V}^i_{f'}$ and $v' \in \mathcal{V}^M_f$, $v \sim^i_{\mathcal{V}_f} v'$ if and only if $v = v'$, i. e. the term on the Mediator is *equivalent* to the same term of the Index w. r. t. the articulation.

The Mediator query terminology is defined similarly to the Index terminology, i. e. $\mathcal{C}_M$ is a set of pairs (f, v) such that $f \in \mathcal{F}^M$, $v \in \mathcal{V}^M_f$, and $\leq_{\mathcal{C}_M}$ is a subsumption relation over $\mathcal{C}_M$. Moreover each $\mathcal{V}^M_f$ is a terminology, i. e. a pair $(\mathcal{V}^M_f, \leq_{\mathcal{V}_f})$ where $\leq_{\mathcal{V}_f}$ is a subsumption relation over $\mathcal{V}^M_f$.

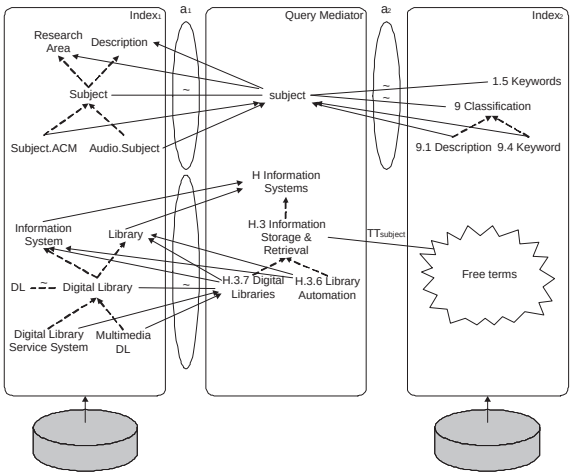

Figure 5.3: A Query Mediator over two Indexes

Figure 5.3 shows an example of a Query Mediator that operates over two Indexes. This mediator uses the DC metadata schema and the ACM Computing Classification System as controlled vocabulary for the field `subject`[8]. The Index services in Figure 5.3 are $Index_1$, that has been introduced in the previous section, and $Index_2$, an Index service that uses the LOM metadata schema [LOM] and free terms to populate the fields shown in the figure.

The query interpretations that are supported by the Query Mediator are defined in terms of both the interpretations stored by the Index services and the existing articulations. In order to identify these interpretations we proceed in the following way:

[8]For brevity the example shows only a partial view of the Query Mediator.

1. we define a query c^i for I_i as a translation of each $c \in \mathcal{C}_M$ obtained using a_i, $i = 1, \ldots, n$;

2. we evaluate c^i at I_i, $i = 1, \ldots, n$;

3. finally, we define $I(c)$ as the union of the answers to queries c^i returned by the Index services.

Several possible translations can be applied. In the following we show the possible ways to perform the translation. In order to define these translations formally we need some preliminary definitions, i. e. the concepts of *head*, *body* and *tail* w. r. t. an articulation and the concept of approximations over values.

Definition 5.4.2 (Tail, Head and Body w. r. t. an Articulation) *Given a condition $c \in \mathcal{C}_M$ where $c = (f, v)$ and an articulation a_i we define*

$$\begin{aligned} tail^i(c) &= \left\{(f', v') | f' \preceq^i_{\mathcal{F}} f \wedge v = v' \wedge f' \in \mathcal{F}^i\right\} \\ body^i(c) &= \left\{(f', v') | f' \sim^i_{\mathcal{F}} f \wedge v = v' \wedge f' \in \mathcal{F}^i\right\} \\ head^i(c) &= \left\{(f', v') | f \preceq^i_{\mathcal{F}} f' \wedge v = v' \wedge f' \in \mathcal{F}^i\right\} \end{aligned}$$

Intuitively, $tail^i(c)$, $body^i(c)$, and $head^i(c)$ contain, respectively, all the conditions in the Index query terminology that are narrower than c, equivalent to c and broader than c w. r. t. the articulation. The conditions above involve Index metadata fields that are, respectively, subsumed by, equivalent to or that subsumes the field used on the Query Mediator w. r. t. the semantic mapping among the Mediator metadata schema and Index metadata schema.

Definition 5.4.3 (Value Approximations w. r. t. an Articulation) *Given a condition $c \in \mathcal{C}_M$ where $c = (f, v)$ and an articulation a_i. Let*

$$\begin{aligned} tilde^i(c) &= \left\{c' \in \mathcal{C}_i | f' = f \wedge v \sim^i_{\mathcal{V}_f} v'\right\} \\ lower^i(c) &= \left\{c' \in \mathcal{C}_i | f' = f \wedge v \preceq^i_{\mathcal{V}_f} v'\right\} \\ upper^i(c) &= \left\{c' \in \mathcal{C}_i | f' = f \wedge v' \preceq^i_{\mathcal{V}_f} v \wedge v' \nsim_{\mathcal{V}_f} v\right\} \end{aligned}$$

we define three kinds of approximations over values:

$$\begin{aligned} c^i_{\sim} &= \bigvee tilde^i(c) \\ c^i_{\leq} &= \bigvee lower^i(c) \\ c^i_{\geq} &= \bigwedge \left\{c'^i_{\leq} | c' \in upper^i(c)\right\} \end{aligned}$$

The above approximations correspond to three different ways in which a condition on a field can be reformulated into a set of conditions that take into account the semantic relationships among the Query Mediator field terminology and the Index field terminology.

Now we are able to define formally the *precise approximations*, the *lower approximations* and the *upper approximations* of a condition $c_i \in \mathcal{C}_M$. Roughly speaking the *precise approximation* of c_i w. r. t. a_j is the disjunction of all the conditions in $\mathcal{C}_j$ that are *equivalent* to c_i in a_j, $c^i_{\sim}$; the second one, $c^i_{\leq}$, is the disjunction of all the conditions in $\mathcal{C}_j$ that c_i subsume in a_j; while the last one, $c^i_{\geq}$, is the conjunction of all the conditions that subsume c_i in a_j. These families of approximations are formally defined as follows:

Definition 5.4.4 (Precise Approximations w. r. t. an Articulation) *Let* $M = (\mathcal{C}_M, \leq_{\mathcal{C}_M}, a_1, \ldots, a_n)$ *be a mediator over sources* $S_1, \ldots, S_n$. *Given a condition* $c = (f, v) \in \mathcal{C}_M$ *we define three kinds of precise approximations of* c *w. r. t.* a_i *as:*

$$\begin{aligned} c^i_{p_\sim} &= \bigvee \{c^i_\sim | c' \in body^i(c)\} \\ c^i_{p_\leq} &= \bigvee \{c^i_\leq | c' \in body^i(c)\} \\ c^i_{p_\geq} &= \bigvee \{c^i_\geq | c' \in body^i(c)\} \end{aligned}$$

Definition 5.4.5 (Lower Approximations w. r. t. an Articulation) *Let* $M = (\mathcal{C}_M, \leq_{\mathcal{C}_M}, a_1, \ldots, a_n)$ *be a mediator over sources* $S_1, \ldots, S_n$. *Given a condition* $c = (f, v) \in \mathcal{C}_M$ *we define three kinds of lower approximations of* c *w. r. t.* a_i *as:*

$$\begin{aligned} c^i_{l_\sim} &= \bigvee \{c^i_\sim | c' \in tail^i(c)\} \\ c^i_{l_\leq} &= \bigvee \{c^i_\leq | c' \in tail^i(c)\} \\ c^i_{l_\geq} &= \bigvee \{c^i_\geq | c' \in tail^i(c)\} \end{aligned}$$

Definition 5.4.6 (Upper Approximations w. r. t. an Articulation) *Let* $M = (\mathcal{C}_M, \leq_{\mathcal{C}_M}, a_1, \ldots, a_n)$ *be a mediator over sources* $S_1, \ldots, S_n$. *Given a condition* $c = (f, v) \in \mathcal{C}_M$ *we define three kinds of upper approximations of* c *w. r. t.* a_i *as:*

$$c^i_{u_\sim} = \begin{cases} \bigwedge \{c'^i_{l_\sim} | c' \in head^i(c) \wedge c' \nsim c\} & \text{if } head^i_{\leq}(c) \setminus c \neq \emptyset \\ c^i_{l_\sim} & \text{otherwise} \end{cases}$$

The other upper approximations $c^i_{u_\leq}$ *and* $c^i_{u_\geq}$ *are defined in a similar way changing accordingly the kind of lower approximations to use.*

Here are reported some examples of approximations for the mediator shown in Figure 5.3[9]:

$$\begin{aligned}(\texttt{DC.subject},\texttt{H.3.7})^1_{p_\sim} = \; & (\texttt{subject},\texttt{Digital Library}) \vee \\ & (\texttt{subject},\texttt{DL}) \\ (\texttt{DC.subject},\texttt{H.3.7})^2_{l_\sim} = \; & (\texttt{1.5},\texttt{H.3.7}) \vee (\texttt{9},\texttt{H.3.7}) \vee \\ & (\texttt{9.1},\texttt{H.3.7}) \vee (\texttt{9.2},\texttt{H.3.7})\end{aligned}$$

The approximations are just queries to the information source S_i and can have sure (three kinds) or possible (three kinds) answer as shown in Section 5.3. For this reason we can define at least 54 possible interpretations I for the mediator[10]. We denote these interpretations using this formalism $I_{a,b}$ where a is the kind of approximation that the mediator uses and b is the kind of answer from the source, e. g. $I_{u_\leq,+_\leq}$ means that the mediator uses the upper approximation with $\leq$, while the sources reply following the possible model $I^+_\leq$. Note that these approximations are defined as the set union over the source interpretations w. r. t. the mediator approximation, e. g. $I_{u_\leq,+_\leq}(c) = \bigcup_{i=1}^n I^+_\leq(c^i_{u_\leq})$.

As the mediator can be considered an information source it can give either one of the three sure answers or one of the three possible answer for each of the above interpretations, i. e. we can have 324 possible modes under which the mediator can operate. We denote these operations modes using this formalism $I^c_{a,b}$ where a is the kind of approximation that mediator uses, b is the kind of answer from the source and c is the answer that the mediator produces, e. g. $I^{+_\leq}_{u_\leq,+_\leq}$ means that the mediator uses the upper approximation with $\leq$ and replies following the possible model with $\leq$ while the sources reply following the possible model $I^+_\leq$.

5.5 Implementation: the Enhanced OpenDLib Search Service

The approach we have described so from a theoretical point of view has been exploited for building an advanced search service for the OpenDLib Service System [CP02, CP03].

The OpenDLib architecture of the search service is very similar to the logical one described in Section 5.2. However, this service does not support any subsumption between attributes of the metadata format and assumes the standard subsumption relation between terms and their stems. Moreover, the search functionality over heterogenous metadata formats is supported thanks to a common metadata format.

[9]For brevity we have used the code of the fields or the code of a terminology term instead of the whole value as no confusion arise. Clearly the abbreviated terms must be replaced by the whole term.

[10]For simplicity we assume that all the Indexes respond using the same kind of answer.

One of the on-line DLs powered by the OpenDLib software is called *tLibrary*. It manages documents harvested from different ISs. Some of these sources represent their content using DC, others use the qualified version of this format, others apply proprietary metadata descriptions. The different semantic interpretations of the same metadata fields and the presence of a variety of field qualifiers reduce the quality of the search functionality when heterogeneous information sources are selected by the user, even when all the different metadata descriptions of the content are indexed.

To overcome this problem we decided to design an experimental search service fully based on the illustrated techniques. We needed to (*i*) easily drive users in querying both homogeneous and heterogenous information sources, (*ii*) simply present how to ask for a more precision-oriented, or recall-oriented, query evaluation, and (*iii*) hide the complexity of the proposed approach.

Taking into account that our harvested information sources have not used controlled vocabularies, and therefore was not possible to identify subsumption relations between values, we decided to maintain the support of the standard subsumption relation between terms and their stems. Moreover, we decided to only support the Π_f approximation, i. e. we chose to simplify the approach of the users with the system loosing the exploitation of the relation among different controlled vocabularies, e. g. the Dewey Decimal Classification (DDC), the Library of Congress Classification, etc.

The resulting search service is based on two relation operators[11], *literal* and *contain*, and two search functionalities, *simple* and *cross-schema*.

The *simple* search functionality supports query requests on homogeneous information sources. It allows to choose between two possible query interpretation models, *sure* and *possible*. This means that, for each query, users can now specify the personalized recall that they think is needed to satisfy their needs. For example, the user John Smith, who is confident that he is interested only in documents that are classified exactly with token "text processing", can specify the query as "subject *literal* text processing". The second user, Henry Stamp, who searches for documents about the same token but does not know how they have been classified, can ask for an interpretation of the query that also takes into account documents that are classified under the semantically specialized "subject" field, i. e. he can select the *sure* interpretation that will return also *doc2*. Finally, we can consider a third user, who want to retrieve documents about "digital libraries", clearly focusing his interest on recall, can specify the query as "subject *contain* digital libraries" and select the *possible* interpretation, implicitly asking for a $I^+_{\geq}$ query answering. In order to implement this functionality the Index service has been enhanced to support the *sure*, $I^-_{\sim}(c)$ and $I^-_{\geq}(c)$, and *possible*, $I^+_{\sim}(c)$ and $I^+_{\geq}(c)$, evaluation models described in Section 5.3. Preliminary tests demonstrate that we can best manage field qualifiers using the *sure evaluation model* if the query has been expressed on

[11] Using relation operators, the user specifies how the system must interpret the query tokens.

a field that supports qualifiers, and the *possible evaluation model* if the query has been expressed on a qualifier of a field.

The *cross-schema* search functionality supports requests on heterogenous information sources. It allows to choose between three possible query interpretation models, *precise*, *lower*, and *upper*, that indicate the type of approximation the system applies to navigate heterogenous metadata schemas. This means that, for each search request, users can now specify how the system, using the relation among the different metadata schemas, must reformulate the user query. Clearly, the *lower* is more precision-oriented while the *upper* is more recall-oriented. In order to implement this functionality we enhanced the QM service to ingest the mapping schema, which contains the definition of the non-trivial articulations between metadata schemas, and to support the *precise*, *lower*, and *upper* approximations as defined in Section 5.4. In particular, we verified the benefits in query processing where the QM applies *lower approximations* asking the Indexes to use the *sure evaluation models*, and where the the QM applies *upper approximations* and Indexes use *possible evaluation models*.

These restrictions on the set of possible combinations mean that a user of tLibrary can only ask for six possible interpretations of the query on heterogenous information sources and four possible interpretations of the query on information sources that use the same metadata schema. Nevertheless, from the user's point of view, the appropriate use of these personalized search evaluations makes it possible to improve recall without losing search precision.

We are now working to identify other combinations between approximations and query evaluation models that could help users to satisfy their needs without increasing too much the complexity of the interaction between users and system. We also plan to support the articulation between terminologies to offer a second generation search service over metadata schema and ontologies.

5.6 Related Work

Prior works, mainly in the area of Distributed Information Retrieval (DIR), address problems similar to those introduced into this chapter. Actually, in the DIR area there are three approaches for interoperability and distributed discovery which differ in the amount of standardization or effort required, i. e. federated approach, harvesting, and gathering.

In the federated approach a number of organisations agree on a number of specifications, usually selected from formal standards, in building the services that provide the data. The problem of establishing a federation is the effort required by each organization to implement and keep them current with all the agreements. For instance, many libraries have standardized on the Z39.50 protocol to meet the needs for record sharing and distributed search [Z3903]. This protocol specifies rules allowing a client to connect, search, request information (about available collections, formats, and syntaxes), and browse indexes available on the server. Most implementations

emphasize searches on bibliographic attributes of MARC catalog records [MAR05] however, since the protocol is flexible, large, and complex, there exist (*i*) different implementations that have different features and (*ii*) catalogs that can be internally organized and presented in different ways. All these factors limit the interoperability. Dienst is another example of protocol built on top of web technologies to operate the Networked Computer Science Technical Reference Library (NCSTRL) [DL96]. In this environment, the services forming the library are divided in four categories: repositories, indexes, collections, and user interface. When a distributed query is issued, it is broadcasted to all the servers and the results are locally rearranged by the user interface.

The best current example of harvesting approach is illustrated by the Open Archive Initiative (OAI) [LV01]. Here, at the root of the technical agreement lies a distinction between two classes of participants: (*i*) data providers, i. e. participants that adopt the OAI technical framework as a means of exposing metadata about their content, and (*ii*) service providers, i. e. participants that harvest metadata from data providers using the OAI protocol and use the metadata as the basis for value-added services. The initiative promotes the use of the Dublin Core [DC] as standard metadata format, but community-oriented metadata formats can be used. In this case the problem of interoperability in providing a search functionality is shifted on the service providers that still have to deal with different data quality, semantic interoperability, and duplicate detection.

Gathering is the approach less demanding among those presented. It represents the mechanism commonly used by web search engines that gather the information objects via appropriate crawlers and then provide search facilities over them [WSY+02]. However, this approach suffers the *hidden web* problem, i. e. valuable information objects are only accessible through search interface of web-accessible databases and the traditional crawlers fails in accessing them.

All these approaches suffer, in a different measure, the *semantic interoperability* problem. This problem can easily be tackled within a federated approach by adopting a common standard, even though this results in greater costs in terms of effort in participating to the federation. In the case of the others two types of methods a solution for transforming the acquired information objects and the related metadata into a common format have to be envisaged. The problem, i. e. finding correspondences between information represented in different schemas or format, is known in literature as *schema matching* or *information integration*. It arises in different contexts, namely each time different information sources are grouped together to form an homogeneous source to search in as in the case of federated databases, search engines or virtual digital libraries. The common approach is to transform queries from the global schema to the local schema while information objects are transformed from local schema to the global schema. Many efforts exist in automatizing this process [RB01, Len02]. In particular, a hot-topic is the automatic discovery and learning of the mapping rules, e. g. [NS05, NF01, DMD+03]. The major improvement brought by our approach and its novelty are represented by that

users can influence the way through which the system uses these rules to reply to queries. Our framework can easily manage mapping rules both manually provided or dynamically generated.

Finally, a detailed and general purpose comparisons of approaches to interoperability for digital libraries can be found in [PCWGM98].

Chapter 6

Virtual Digital Libraries Generator

In the previous chapters we focused on dealing with various heterogeneity problems in reusing the elements of the information space by providing virtual views of them in order to support their reuse in building virtual digital libraries. As formalised in our reference model (*i*) a digital library is made by other resources than information objects and (*ii*) behind a digital library there is a digital library system in charge to implement the perceived digital library. Thus, in order to provide virtual digital libraries we must be able to aggregate all the components constituting a digital library system and thus a digital library. The goal of this chapter is to present the *Virtual Digital Libraries Generator*, i. e. a service allowing DL Designers to define the characteristics of the digital libraries they are interested in and partially replacing the task of the DL System Administrators in implementing such digital libraries[1]. We demonstrate the feasibility of the proposed service by providing examples illustrating the facilities provided and reporting on its exploitation in the context of the ongoing IST project DILIGENT [DIL].

The chapter is organised as follows. Section 6.1 gives an overview of the proposed approach. Section 6.2 presents the design of the Virtual Digital Libraries (VDL) Generator introducing its main components. Section 6.3 reports on the logical model governing the behaviour of the service. Section 6.4 introduces the on going DILIGENT project and provides example of exploitation of the VDL Generator in such environment. Finally, Section 6.5 concludes by presenting and discussing related work.

[1]This work is partially funded by the European Commission in the Context of the DILIGENT project, under the 2nd call of the FP6 IST priority.

6.1 The Approach

In the reference model introduced in Chapter 2 we have identified two actors that are in charge of creating a digital library fulfilling the requirements of the DL End-users, i. e. the *DL Designer* and the *DL System Administrator*. The former actor is responsible for gathering the requirements of the digital library the user community is interested in and to express them in terms of information space entities and functionality the digital library must be equipped with. This actor is usually an expert librarian who has no perception of which digital library system is needed to fulfil such requirements. Instead, the latter actor, i. e. the DL System Administrator, must be able to understand the DL Designer requirements and map them into the components of the digital library system to be configured and deployed in order to match the expectation of the designer. Usually the envisaged approach is conducted without any form of automatic support and the creation of a digital library is up to the capabilities of the human actors.

This manual approach is not feasible in the case of the virtual digital library scenario, where the number of digital library resources potentially available becomes huge. Some of the reasons why such an approach is unfeasible are (*i*) the presence of thousands of possible choices and (*ii*) the presence of thousands of constraints that can result to be difficult to deal with even for experts. Automatic or semi-automatic processes aiding the activity of both the DL Designers and the DL System Administrators reduce the risks of inconsistency and increase productivity.

From an abstract point of view, the digital library management system acts as a *broker*, while the clients of the broker are DL resource producers and consumers. The producers are the individuals and the organisations that decide to share, under the supervision of the broker, their resources according to certain access and use policies, while the consumers are the user communities that want to build their own VDLs. The resources managed by this broker can be of different types ranging from collections of information objects to software components and hosting nodes. Moreover, it is important to recall the different views the DLMS must provide over such resources according to our reference model. The DL Designer perceive the information space as composed by collections of information objects and a digital library as an entity providing functionality to act over such objects. The DL System Administrator perceives such abstract resources in terms of concrete resources to be produced, i. e. software components and hosting nodes where such components can be deployed in order to produce the expected functionality and information space.

The digital library management system manages the registered resources by supporting their discovering, monitoring, reservation, and by implementing a number of functionality that aim at supporting the required controlled sharing and level of quality of service.

A user community, more precisely a DL Designer, can create one or more DLs by specifying a set of requirements with the support of the system. These requirements specify conditions on the information space and on the functionality the digital

library must provide. The DLMS, acting as a DL System Administrator, satisfies the given requirements by selecting, and in many cases also deploying, a number of resources among those accessible to the community, gluing them appropriately and, finally, making the new DL accessible to DL users.

In the next section the design of the virtual digital libraries generator is presented and the assumptions about its operating environment are reported.

6.2 The Virtual Digital Libraries Generator Design

The virtual digital library generator envisaged in this dissertation is a component designed to rely on an *infrastructure* represented by the digital library management system that is in charge of actually realizing the virtual views defined and identified by it. However, the dependencies with respect to such enabling infrastructure are well known and can be limited to the gathering of the information about the available resources and to the provision of the DL specification the infrastructure must implement in order to provide the virtual digital library.

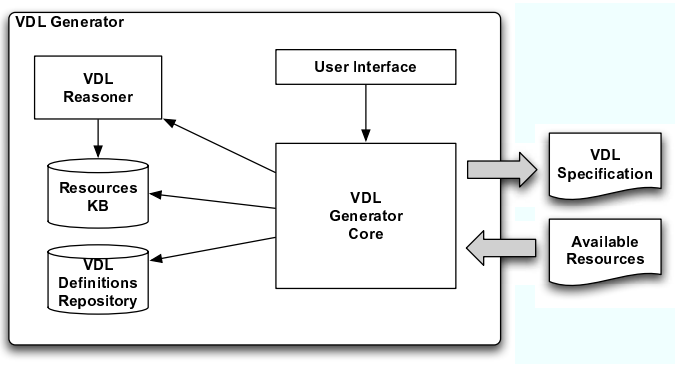

Figure 6.1: The VDL Generator Logical Architecture

Having clarified this point, the logical architecture of the VDL Generator is presented in Figure 6.1.

The *User Interface* provides the graphical environment enabling the DL Designer to perform its tasks. In particular, the expected functionality must offer the possibility to define a VDL by selecting the resources among those available, modify a previously defined VDL starting from the previous definition, and removing a previously defined virtual digital library while allowing the designer to establish the actions to be executed in order to preserve the information objects produced in the context of such a digital library and make them available for further digital libraries. It is worth noting that the design of this component relies on the outcome of the reference model, i. e. we decided to organise the environment on the DL main concepts in a hierarchical way, e. g. under the category Information Space the DL

Designer is entitled to define the characteristics of the related concepts while under the category Functionality it is entitled to select the functionality the digital library must be equipped with. Moreover, it is important to notice that, by interacting with the VDL Generator Core component, the user interface is in charge to prevent the designer from performing inconsistent choices.

The *VDL Generator Core* component represents the most important element of the proposed architecture. It is responsible for implementing the application logic needed to (*i*) guide the User Interface in showing, step-by-step, which components can be used and how they can be customized, thus preventing the designer from performing inconsistent choices, (*ii*) produce the VDL specification that represents the mapping of the DL Designer requirements into the components needed to implement the expected digital library, (*iii*) notify the infrastructure when a digital library previously created must be removed, and (*iv*) maintain updated its knowledge of the available resources.

In order to support the VDL Generator Core component in providing its functionality we equipped the service with the *VDL Reasoner*. This component is in charge of implement a logic-based approach to the identification of the components needed to fulfil the specified requirements. Further details on this process are presented in Section 6.3. In brief, the idea is as follows. Both the VDL definition criteria and the available components together with the related applicability/usability constraints are expressed by a knowledge representation mechanism. By adopting the inference mechanism supported by such a mechanism the system is able to identify the set of available components needed to satisfy the definition criteria.

The reasoner and in general the whole service thus needs to be aware of the resources that the infrastructure makes available and maintains an in dept knowledge about them into the *Resource KB*.

The latter component is represented by the *VDL Definitions Repository*. This component is in charge of maintaining the original definition of the virtual digital library as well as the computed virtual digital library definition.

6.3 The Components Selection Model

As discussed by presenting the architecture of the VDL Generator service carefully attention must be posed on and effort must be spent in realising a powerful user interface making easy the work of the DL Designer and on the process in charge to match the set of components needed to fulfil the expressed requirements. In this section we describe the second point by focusing on the logic based approach we envisaged.

The first problem we face is related to the identification of an appropriate knowledge representation mechanism/formalism. It must be powerful enough to allow to expressing the knowledge we need to represent as well as it must have an inference mechanism that is computable in an human acceptable time. Moreover, it must exist

a system that provides support for the implementation of the identified mechanism because our goal is not to implement a knowledge representation system. In particular, we analysed the approach proposed for the semantic web [BLHL01]. In this research area two very different modelling paradigms have been proposed [PSH], i. e. the *Classical paradigm* and the *Datalog paradigm*. The classical paradigm is based on the notions from standard logics, such as propositional logic, first-order logic, and Description Logic [BCM+03]. This approach is embodied in the W3C recommendation for Semantic Web languages, RDF [MM04] and OWL [DSB+04]. The Datalog is based on notions coming from Object-oriented Databases [AG89] and rule languages [Ull88]. This approach enbodied in previous version of RDF and proposal for Semantic Web languages, e. g. OWL-Flight [dBPLF04]. There are significant diffrences between these two paradigms. These differences range from computational aspects, to expressive power and naturalness of modelling [PSH].

Basing on these characteristics, we decided that a suitable solution for our problem is represented by the Datalog approach. Datalog is a rule-based declarative language. This means that a user has not to write a program that solves some problem but instead specifies what the solution should look like, and a Datalog inference engine tries to find the way to solve the problem and the solution itself with the available registered components. This is done with rules and facts. Facts are the input data, and rules can be used to derive more facts, and the solution of a given problem. In particular we concentrate on Disjunctive Datalog [EGM97], i. e. an extension of Datalog in which the logical OR expression (the disjunction) is allowed to appear in the head of a rule.

The VDL Generator Service, and in particular the functionalities related to the definition of a VDL and the automatic identification of the pool of resources needed to fulfil the VDL requirements, can be modelled as a *configuration problem* [BCM+03]. The following two aspects characterize these kinds of problems: (*i*) the artifact to build is composed by instances of components and (*ii*) components interact in predefined ways, i. e. their dependencies from other components are well-known. Moreover, all the systems for product configuration converge on describing this problem representing two kinds of knowledge: (*i*) the domain description, i. e. a description of all the types of components available, and (*ii*) the specification of the desired product. This is also the approach we propose where the domain description is characterised by the description of all the available components available via the Resources KB while the specification of the desired product is provided by the DL Designer via its interaction with the VDL Generator User Interface.

In particular, for domain modeling we adopt the component-port approach, i. e. components are characterized by three elements: the *type*, the *attributes*, and the *ports*:

- Types allow organizing components into a hierarchy that can be used during configuration.
- Attributes specify descriptive features, such as functional or technical charac-

teristics, configuration parameters, etc. Each attribute has a single value or can take values from a predefined range.

- Ports are used to establish connections between components. Usually when defining ports restrictions may be imposed on the type and number of components that can be connected to it. Constraints placed on ports are the natural way to express compatibility between components. They allow expressing conditions on attributes and ports that must hold in the model built to satisfy the DL requirements.

It is worth noting that in order to make such approach working appropriately, an ontology on these aspects must be identified and known by services providers that are in charge to describe their resources accordingly. In our case, this ontology will be guided by the reference model. It is also important to notice that the logic of the application is guided by a Datalog set of rules and that these rules can be easily modified making the behaviour of the whole service easily adaptable to different contexts.

6.3.1 A Trivial Example

In this section we report a trivial example of a Datalog set of facts and rules in order to show the power of the adopted formalism.

In order to represent the domain description we use two predicates: `component` and `yields`. The former represents an available component while the latter take care to declare the functionality provided by the component and the level of quality of service of such functionality. The following set of facts represent a scenario in which there are five available components and the first of them support the search functionality with two type of level of quality of service, depending by a configuration parameter.

```
component(1,searchType1).    % component(id,name)
component(2,repository).
component(3,browse).
component(4,searchType2).
component(5,indexDistr).
yields(1,search,qos1).      % yields(componentID,functionality,qos)
yields(1,search,qos2).
yields(2,repository,qos1).
yields(3,browse,qos1).
yields(4,search,qos3).
yields(5,index,qos3).
```

By adding a simple rule like the following it is possible to derive all the allowed

functionality[2].

```
allowedDLFun(FID,QOS) :- yields(_,FID,QOS).
```

In order to identify the components needed to fulfil a certain user requirement it is sufficient to add the rule identifying the appropriate component and the fact identifying requirement as follows:

```
dlComp(DL,CID,CN) :- reqFun(DL,FID,QOS),
                     component(CID,CN),
                     yields(CID,FID,QOS).
reqFun(myDL,search,qos3).
```

In this case the system identify the component 4 named `searchType2` as those capable to provide a search functionality with a quality of service `qos3`.

If we enrich the knowledge base by adding the facts modelling the dependencies between components we are able to identify all the components needed to operate the digital library correctly as follows. We add a fact predicate `requires` indicating the component that in providing a certain functionality needs another component (in the example, 4 needs 5 to provide the search) and overload the `dlComp` rule in order to take care of such additional requirements.

```
requires(4,search,5).
dlComp(DL,CID,CN) :- dlComp(DL,CID2,CN2),
                     requires(CID2,FID,CID),
                     component(CID,CN),
                     reqFun(DL,FID,QOS).
```

In this case, if the DL Designer asks for a digital library with a search functionality having a quality of service `qos3` and the system identifies as components of such digital library both component 4 and component 5.

This trivial example illustrates some of the capabilities that can be easily modelled via the Datalog paradigm. We concentrate on the quality of service as discriminant parameter guiding system choices, other aspects can be modelled similarly. In a concrete example the predicates and the rules can assume more complex forms however the feasibility of the approach remain unchanged.

6.4 The DILIGENT Experience

DILIGENT [DIL] is an ongoing IST project that aims to combine Grid [FK04] and digital libraries technologies in order to provide a test-bed digital library infrastructure. The main goal of the project is to create an advanced test-bed that will

[2]This functionality is particularly useful in deriving the knowledge needed to populate the user interface with the allowed functionality.

allow members of dynamic virtual e-Science organizations to access shared knowledge and to collaborate in a secure, coordinated, dynamic and cost-effective way. Actually this mean to provide the capability to implement virtual digital libraries as presented in Section 6.1 where the whole infrastructure represents the resources broker.

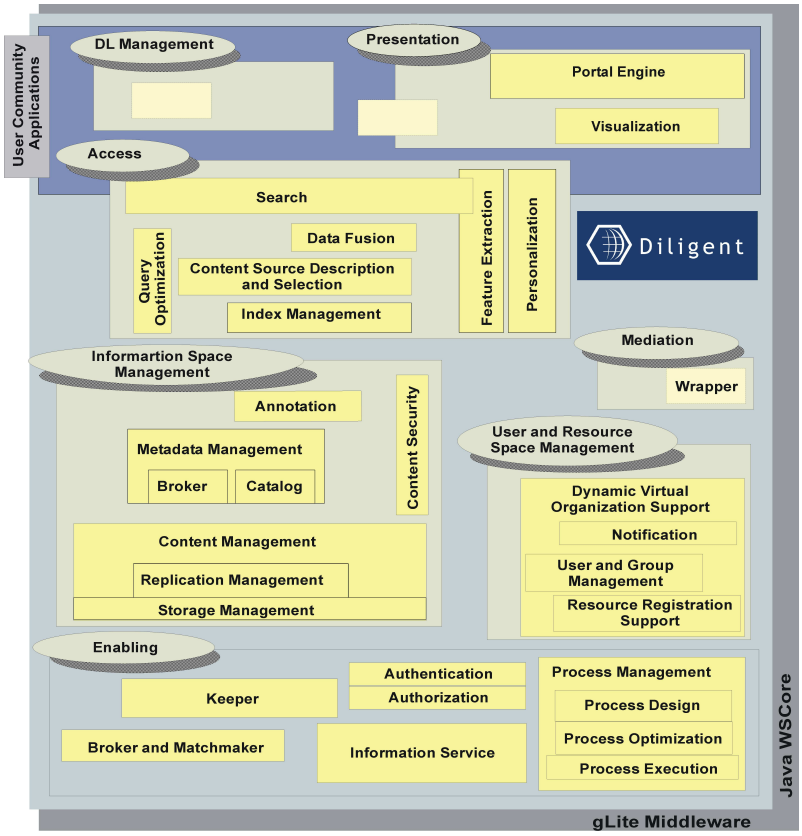

Figure 6.2: The DILIGENT Logical Architecture

The architecture of the DILIGENT system is depicted in Figure 6.2[3] accordingly to the reference architecture presented in Chapter 2.

The DILIGENT infrastructure is being constructed by implementing a service-oriented architecture in a Grid framework. In particular, DILIGENT exploits the *gLite* Grid middleware [EGEb] and the Grid production infrastructure provided by the EGEE project [EGEa] as well as the WSRF specification [Ban05] implementation released by the Globus® project [Glo]. By relying on such software framework the DILIGENT services are entitled to act as Grid Services and have access to shared resources via the grid based mechanisms. These resources are represented by both the computing elements and storage elements provided by the EGEE project as well as the Grid Services provided by the project itself.

[3]Provided by Pasquale Pagano.

A brief overview of the services constituting the functional area of the DILIGENT system is presented in the remaining of this section.

The *Mediation* area includes a number of wrapper components all together managed by the Wrapper service. They are in charge to access external information sources in order to transform the external objects into DILIGENT information objects.

The *Information Space Management* area contains the Content Management service that relies on the Replication Management and on the Storage Management to maintain its data and thus represents the DILIGENT information objects repository; the Metadata Management service that exploits the capabilities provided by the Content Management for the storage and management of the metadata manifestations; the Annotation service and the Content Security service.

The *Access* area includes the Search service that exploits the capabilities provided by the Index Management, the Content Source Selection and Description, and the Personalization services by means of the Query Optimization service. This area contains also the Feature Extraction service that collects a number of features extraction components specialized for different kinds of media.

The *User and Resource Space Management* area includes the Dynamic Virtual Organization Support service. It is an aggregator that exploits the capabilities provided by the Notification, User and Group Management, and Resource Registration Support services. Key functionality is the creation of the trusted environment needed for ensuring a controlled sharing of the DILIGENT resources.

The *Presentation* area is strongly user-oriented. It supports the automatic generation of user-community specific portals, providing personalised access to the DLs. It has been designed to support the plug and play of user communities specific visualization tools. For this reason the DILIGENT project does not provide any specific service of the *DL Management* area.

Finally, the services of the *Enabling* framework are in charge to provide the following functionality:

- the monitoring and discovering of all the available DILIGENT resources (*Information Service*);
- the implementation of a global strategy offering the optimal use of the hosting node resources supplied by the DILIGENT infrastructure (*Broker & Matchmaker Service*);
- the orchestration needed to maintain up and running the pool of resources that populate the various virtual digital libraries and to ensure certain levels of fault tolerance and QoS (*Keeper Service*);
- the support to design and verify the specification of workflows, as well as services ensuring their reliable execution and optimization (*Process Management*).

6.5 Related Work

Virtual Digital Libraries represent the mechanism we envisaged in order to overcome the drawbacks of the actual digital library development processes. The VDL Generator, being a component of the digital library management system, has not been created nor proposed elsewhere simply because none of the actual digital library systems provides the DLMS expected functionality.

The closest work to our approach is represented by a family of tools built on top of the 5S framework and discussed in detail in the following section.

The problem of service composition has been studied in the web services community [MBH+04, AVMM04, PBB+04, NM02]. However, the proposed techniques seem to suffer from the general purpose approach. The digital library area is restricted to certain type of components and well known constraints, therefore the problem is more manageable and can be tackled with different and domain specific technique.

6.5.1 The 5S Products: 5SL, 5SGraph, and 5SGen

By relying on the 5S framework [GFWK04, Gon04], Gonçalves presented in his dissertation a series of tools and applications for modelling and semi-automatically customising digital library services named *5SL*, *5SGraph*, and *5SGen*.

5SL [GF02] is a declarative domain specific language for digital library specification. With this language the specification of a digital library consists of five models related to the dimensions of the underlying formal framework. The stream model is devoted to specify the format of media objects supported by the digital library by relying on the web standard MIME types. The structural model defines via an XML Schema the structure of the information objects as well as the properties of collections and metadata the digital library deals with. The spatial model gives details about the digital library retrieval model, the characteristics on indexes, and the user interface appearance. The societal model makes it possible to model the characteristics of actors and services by identifying the five core services each digital library must provide, i. e. user interface, index, search, repository, and browse. For both actors and services, the set of attributes and the set of interactions with the services are modelled. In the case of services the description of the operations is provided as well. Finally, the scenario models the behaviour of a service via a sequence of events. All these constructs are provided in XML.

As any domain specific language 5SL has its own problems, namely (*i*) different semantics (at least one for each model) must be understood to define a digital library, (*ii*) the definition of a complex digital library is difficult even for experts since there is a great amount of XML to be manually produced and a number of semantic constraint and dependencies to be verified in order to ensure consistency, and (*iii*) it is difficult to obtain the big picture of the defined digital library. To overcome these problems, the 5SGraph is proposed.

5SGraph [ZGS+03] is a domain specific visual digital library modelling tool whose output is a specification of a digital library in terms of the 5SL language. This tool can be configured with a set of characteristics on the allowed digital libraries (expressed in terms of a 5S metamodel) and thus is able to enforce these constraints and ensure the semantic consistency and correctness of the digital library specifications produced.

5SGen [KGF03] is the last link in the digital library development chain proposed. In particular, this software system is dedicated to the semi-automatic production of digital library components fulfilling the model of societies and scenarios expressed in terms of the 5SL language. The approach they adopt is based on a component oriented view of the digital library systems and thanks to this the DL designers are entitled to model (via sequence diagrams and state chart diagrams) the behaviour and co-operation of such basic components in delivering the digital library expected functionality. Then, they are able to obtain a set of service manager modules that produce the designed digital library functionality by relying on freeware tools capable to dynamically generate Java code (*i*) from the XMI[4] representation of the societies models (XMI2Java) and (*ii*) from the finite state machine representation of the society models.

In comparison, our proposed approach is less software engineering and code generation oriented as it aims at creating digital libraries without the production of code. We consider coding as an activity conducted by the DL Application Developers (Section 2.5 on page 46) when a functionality that cannot be provided by any existing digital library system component is needed. As a consequence, we do not take care to define any specification language. Instead, by relying on the formal model we support the DL Designers in expressing their information space and functional requirements via an intuitive graphical user interface. Then, it is up to the VDL Generator the identification and configuration of the pool of services needed to fulfil the expressed requirements. The "glue" allowing the various components to co-operate is represented by the *enabling framework*. It is important to note that the digital library system components have been designed and implemented to rely on this framework in accordance with the Digital Library System Reference Architecture (Section 2.6 on page 50).

[4]An XML serialisation of the UML diagrams.

Chapter 7

Conclusion and Future Work

7.1 Summary

Current models for developing digital library are proving to be inadequate to fulfil the large emerging demand for such systems. This fact has motivated the work presented in this thesis in which a new development model is proposed and discussed, and its implementation presented. This model is based on (*i*) *controlled sharing* of resources among different digital libraries and (*ii*) *virtualisation* of such resources in order to provide personalised and focused views matching the specific needs of diverse user communities, i. e. we have proposed a model for *virtual digital libraries*.

To support such a paradigm a common understanding of what a digital library is and what its characteristics are must be shared by all the actors interacting with the library. Our model covers the different perspectives of such actors and highlights and formalises the existence of both the *digital library system* and the *digital library management system* as the entities creating the digital library.

Motivated by the need to reuse third-party pre-existing resources in order to profitably build virtual digital libraries, we have introduced a series of approaches to virtualisation making such a reuse easy and effective. In particular, we have discussed the requirement for *information object virtualisation* and proposed a powerful and flexible document model as an approach to satisfy this requirement. The feasibility of such an approach has been proved; in fact a service to support the proposed document model has been developed and exploited in three concrete and complementary application scenarios.

The second requirement we have identified is related to *collection virtualisation*. We have discussed the issues involved and proposed an approach for supporting virtual views over a heterogeneous information space. The viability of this approach has been demonstrated through the implementation of a service supporting virtual collections and applying it in the context of the IST project Cyclades [CYC].

We have also envisaged the need for seamless search across multiple and heterogeneous information sources. In this respect, we have proposed, developed and

discussed an approach that allows users to express their information needs with a powerful language capable of capturing diverse semantic interpretations of the same textual query and thus obtaining information objects that are closer to their information needs than in the "classic" approach.

Finally, we have proposed the *virtual digital libraries generator*, i. e. a service supporting the novel digital library development model by assisting users in expressing their needs and automatically identifying the appropriate pool of resources needed to fulfil them. We discussed the exploitation of this approach in the context of the on-going IST project DILIGENT [DIL] and proved the feasibility by reporting concrete examples demonstrating the features of the proposed service.

7.2 Future Work

Each of the facets investigated in this dissertation could be further studied and developed.

The activity related with the reference model is now a task of the DELOS Network of Excellence on Digital Libraries. The importance of such a reference model is also highlighted in the recent DELOS Brainstorming Report [CDISS05] prepared to reply to the European Commission's call for online consultation.

This activity, supported by the international research institutions participating in the network is aimed at validating the main organisation of the proposed model and refining and enhancing the set of concepts and relationships introduced to cover the characteristics of digital libraries, digital library systems, and digital library management systems. Particular attention is posed on the reference architecture. According to the model, standards, protocols, and best practices will be studied in order to ease the co-operation and sharing of information objects among different digital libraries.

As for virtual digital libraries, DILIGENT is an ongoing project and the implementation of the virtual digital library generator service is still in progress. This component probably represents the most important issue to be solved for making the process of digital libraries definition an easy task.

Further the in-depth investigation of the technicalities needed for describing components and user requirements will be needed, in particular, mechanisms for dealing with the description of the service provided in heterogeneous languages and schemas must be studied in order to enlarge the pool of the components now available.

The evolution and the exploitation of mediation services also is an important issue for future work. In particular, all the aspects of heterogeneity left out from this dissertation could be object of future investigations, e. g. policies regulating access to information objects are usually expressed in different languages, ontology used to annotate information objects can be used for different purposes than the search.

In the context of a new EU funded project named DRIVER (Digital Repository

Infrastructure Vision for European Research) the needs for mediator services are pressing. However the mediator services and approaches introduced in this dissertation fulfil the most important requirements of this project.

Appendix A

OpenDLib: A Digital Library Service System

OpenDLib [Ope, CP02, CP03] is a software toolkit that can be used to easily set up a digital library, according to the requirements of a given user community, by instantiating the software appropriately and then either loading or harvesting the content to be managed. It consists of a federation of services that implement the digital library functionality making few assumptions about the nature of the information objects to be stored and disseminated. From a deployment viewpoint, the entire set of services can be managed and hosted either by a single or by a multitude of organisations that collaborate on the maintenance of the shared digital library, each according to its own computational and human resources.

A.1 Services Model

Figure A.1 depicts a conceptual model that specifies the notion of OpenDLib service and its characteristics.

This model is central to the digital library system design since it highlights the properties (descriptive metadata) that define each service. Each service instance is known by the other instances through the values of these properties, which are disseminated on demand. These properties are modelled using structured attributes that are associated with each entity of the model. As a natural consequence of the relation between entities, a child entity inherits all attributes of its parent entity. The whole set of attributes characterise an instance of a service in the OpenDLib architecture.

A generic service of the OpenDLib architecture is modelled by the entity *Service*. A service can be distributed over different servers, replicated, or if necessary centralised. The model has an entity for each service type.

The *distributed services* are those that implement the same service through multiple instances, each of which manages data stored on a different server. The data

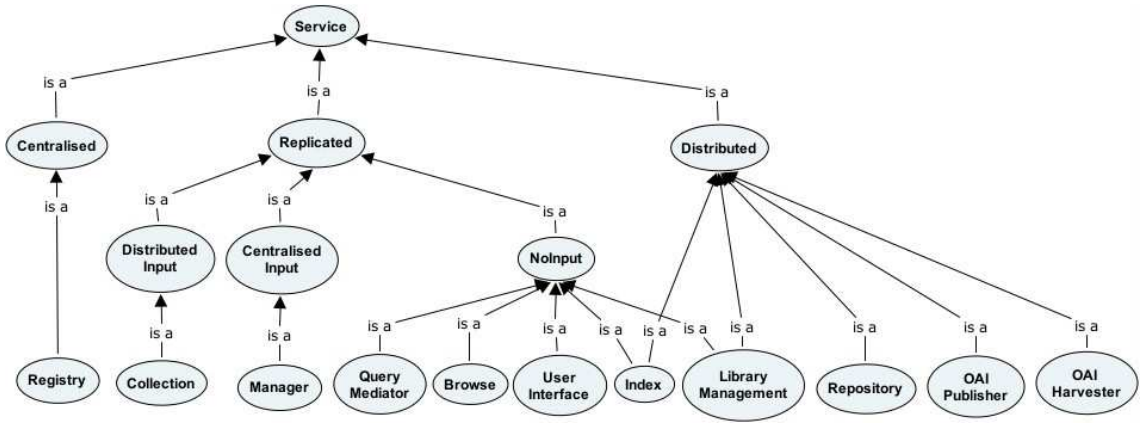

Figure A.1: The OpenDLib Services Model

are distributed according to a set of criteria that may differ from service to service. Note that each instance of a distributed service does not need to know anything about the other instances of the same service. In the current version of OpenDLib that we are illustrating, the services that are distributed are those that maintain a huge amount of data and/or those that are strictly related to the information objects publishing institutions. These institutions usually prefer to maintain their own object on their own server to have a physical control over them. Moreover, each institution usually has its own rules for information object submission/withdrawal, or information space management, and therefore prefers to maintain these procedures also in a common shared environment.

The *replicated services* are those implemented by a set of service instances, possibly located on different servers, where each instance is able to cover completely the service functionality over the entire set of data. OpenDLib distinguishes three kinds of replicated services: *NoInput*, *CentralisedInput*, or *DistributedInput*. A service is of NoInput type if it is instantiated by pure replications, i. e. the different instances are never distinguishable since they handle the same data and behave in the same way. A service of CentralisedInput or DistributedInput type has one replication, which acts as a master, and a set of replicates which act as slaves. In the case of the CentralisedInput, the master is a special instance of the service whose only purpose is to maintain and distribute on demand an updated version of the information handled by the service. The slave instances update their content by periodically invoking the master. Both the master and slave replicas of a DistributedInput service can accept new information and serve information requests. The master maintains the global state of the service information: each time a slave updates its local information it communicates the change to the master, which merges the new information with its own information. Periodically, each slave updates its state by invoking the master. The role of the instances of a replicated service, i. e. master and slave, is not statically assigned but can be changed in order to achieve a better connectivity or to

overcome temporary crash. In the current version of OpenDLib we have chosen to replicate those services that are either not constrained by any proprietary (see distributed services), security or privacy constraint (see centralised). This replication makes it possible to improve service efficiency and to increase its robustness. This is, for example, the case of the replication of indexes to improve content access, or the replication of meta information, i. e. the set of service descriptive parameters and other configuration parameters, to improve digital library service access, and so on.

A *centralised service* has always a single instance in the DL. Each time the security and the privacy of the content of a service is an issue, the centralised solution is preferred to the distributed and replicated ones.

Although the presence of multiple instances of a service increases fault tolerance, reduces the overload of each instance, and makes it possible to dynamically reorganise the environment when a server hosting a service instance is not reachable, the replication and distribution of the services is not mandatory and therefore each service can be instantiated as a single instance. This means that the level of distribution and replication, and the physical location of the service instances may be freely chosen to better satisfy the needs of the specific digital library context.

A.2 Design Considerations

OpenDLib is a general purpose framework to build DLs in multiple domains ranging from traditional libraries to multimedia and/or virtual courses, support for conference reviewing process, electronic publishing, etc. The reuse of OpenDLib in different contexts imposes a set of constraints to the overall design of the services. Most of them are made possible by the adoption of the Service Model presented in the previous section.

Configuration – As stated in the previous section, services are designed to be distributed and replicated to better fulfil the needs to face out during a DL lifetime by providing a great flexibility in the digital library design. This means that each time a new DL has to be created, the appropriate set of services is instantiated and this set can be expanded at any time by adding replicas or additional distributed instances. For example, a replication of an Index instance can be created to reduce workloads when the number of search requests exceeds an established threshold, whereas an Index instance, able to serve queries in a language not previously supported, can be added to satisfy the needs a new community of users. All these expansions can be done on the fly, i. e. without switching off the digital library.

Deployment – The physical distribution of the services can be customised to satisfy specific requirements. For example, an institution can decide to maintain an instance of the Repository Service in order to have local control over its own documents, but to share all the other services with other institutions. The architectural configuration is chosen when the digital library is set up, but it can also be changed

later to satisfy new emerging needs.

Interoperability – From this viewpoint, the OpenDLib architecture has been designed to be highly interoperable with other digital libraries that rely on other systems. This ability of the system is mainly concentrated on the export of its content upon request, and the import of information objects from existing content sources. For the former, besides using its own protocol, OpenDLib can act as an OAI [LV01] data provider by appropriately implementing the OAI-PMH [OAI] protocol. This implies that the metadata records about information objects maintained by an OpenDLib digital library can be open to other digital libraries. The latter ability, i. e. the import of information objects from other systems, is a crucial feature that can determine the success of any digital library system, is obtained in two ways: (*i*) OpenDLib can access the metadata published by any other OAI-PHM compliant digital library since it is also a service provider (in the OAI meaning), and (*ii*) it also takes care of the compatibility with non-OAI-PHM compliant systems by providing basic tools to easily interact with existing information sources and development kits to facilitate the development of customised importing tools.

Openness – The set of services constituting the OpenDLib system can be expanded at any time by adding new services to satisfy community-specific needs. Also for this aspect, appropriate development kits have been developed to facilitate the implementation of new OpenDLib compliant services.

A.3 Services and Functionality

The overall functionality of OpenDLib is partitioned into a set of well-defined interacting services that we have classified accordingly to our reference architecture (Section 2.6) and described in the follow.

A.3.1 Enabling Framework

Manager Maintains and continually updates a picture of the status of the DL service federation and disseminates it on request to all the other services.

A.3.2 User Space Management

Registry Maintains information about the users, groups, and communities.

A.3.3 Information Space Management and Mediation

Repository Stores and disseminates documents that conform to the DoMDL document model which can represent structured, multilingual and multimedia documents. A detailed description is provided in Chapter 3.

Collection Service Mediates between the virtual dynamic organisation of the content space, built according to the requirements of the DL community of users, and the concrete organization into basic collections of documents hold by publishing institutions. A detailed description is provided in Chapter 4.

OAI Harvester Harvests content published by OAI-PMH compliant archives.

A.3.4 DL Management

Library Management Supports the submission, withdrawal, and replacement of documents through a complete review workflow.

A.3.5 Presentation

OAI Publisher Publishes the content of the OpenDLib DL through the OAI-PMH protocol.

User Interface Mediates between human actions and all the OpenDLib services. It is mainly used to interact with access services. As result of their search or browse operations, users obtain a set of results pages with the list of information objects that satisfy their requests. The User Interface provides multiple and customisable ways to visualise these objects as reported in Section 3.4.

A.3.6 Access

Query Mediator Dispatches queries to Index service instances, according to availability and replica priorities.

Browse Supports the construction of indexes to browse the entire library content. The Browse function is parametric with respect to the metadata formats, to the set of fields to be browsed, and to the set of formats for result sets.

Index Accepts queries and returns information objects matching those queries. The Index is parametric with respect to the metadata formats, to the set of indexed fields, to the set of result sets formats and the language of the terms. It offers different search options: free text or advanced (with fields selected from a variety of configurable metadata formats); single or cross-language; with or without relevance feedback.

Finally, it is worth noting that the services interaction is more complex than a simple client-server communication. A service can act both as a provider and as a consumer, and sharing relationships exist a priori among a subset of the services. Communication with and among individual services takes place via the proprietary OpenDLib Protocol (OLP). The OpenDLib Protocol is an evolution of the Dienst

protocol [LD95]. It inherits from Dienst the basic rules and conventions, and many protocol requests.

Bibliography

[ABB+99] Antonella Andreoni, Maria Bruna Baldacci, Stefania Biagioni, Carlo Carlesi, Donatella Castelli, Pasquale Pagano, and Carol Peters. Developing a European Technical Reference Digital Library. In *Proceedings of the third European Conference on Digital Libraries (ECDL'99)*, pages 343–362. Springer-Verlag Berlin Heidelberg New York, 1999.

[ABO97] William Y. Arms, Christophe Blanchi, and Edward A. Overly. An Architecture for Information in Digital Libraries. *D-Lib Magazine*, February 1997.

[ACS] Henri Avancini, Leonardo Candela, and Umberto Straccia. Recommenders in a Personalized, Collaborative Digital Library Environment. *Journal of Intelligent Information Systems*, To Appear.

[AG89] R. Agrawal and N. H. Gehami. Ode (object database and environment): The language and the data model. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 36–45, June 1989.

[Arm95] William Y. Arms. Key Concepts in the Architecture of the Digital Library. *D-Lib Magazine*, July 1995.

[Arm01] William Y. Arms. *Digital Libraries*. The MIT Press, September 2001.

[AVMM04] Rohit Aggarwal, Kunal Verma, John Miller, and William Milnor. Constraint Driven Web Service Composition in METEOR-S. In *Proceeding of the IEEE SCC 2004*, 2004.

[Ban05] Tim Banks. Web Services Resource Framework (WSRF) - Primer. Committee draft 01, OASIS, December 2005. `http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf`.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, implementation, and applications*. Cambridge University Press, 2003.

[BdWH+03] I. Burnett, R. Van de Walle, K. Hill, J. Bormans, and F. Pereira. MPEG-21: Goals and Achievements. *IEEE Multimedia*, Oct-Dec:60–70, 2003.

[Bel99] Nicholas J. Belkin. Understanding and Supporting Multiple Information Seeking Behaviors in a Single Interface Framework. In *Proceeding of the Eighth DELOS Workshop: User Interfaces in Digital Libraries*, pages 11–18. European Research Consortium for Informatics and Mathematics, 1999.

[Ber02] Donna Bergmark. Collection Synthesis. In *Proceeding of the second ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 253–262. ACM Press, 2002.

[BFH03] F. Berman, G. Fox, and A. Hey. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, April 2003.

[BHdS03] Jeroen Bekaert, Patrick Hochstenbach, and Herbert Van de Sompel. Using MPEG-21 DIDL to Represent Complex Digital Objects in the Los Alamos National Laboratory Digital Library. *D-Lib Magazine*, 9(11), November 2003.

[BHM+04] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web Services Architecture. Technical report, W3C, February 2004. W3C Working Group Note.

[Bla02] David C. Blair. The challenge of commercial document retrieval, Part II: a strategy for document searching based on identifiable document partitions. *Information Processing and Management*, 38:293–304, 2002.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic Web. *Scientific America*, 284(5):34–43, 2001.

[BMR+96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture*. John Wiley & Sons, 1996.

[Bor99] Christine L. Borgman. What are digital libraries? Competing visions. *Information Processing and Management*, 35(3):227–243, 1999.

[BRI] BRICKS. Building resources for Integrated Cultural Knowledge Services. `http://www.brickscommunity.org`. IST-2003-507457.

[Bus45] Vannevar Bush. As We May Think. *The Atlantic Monthly*, 176(1):101–108, July 1945.

[BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[Cal00] Jamie Callan. *Advances in Information Retrieval*, chapter 5 Distributed Information Retrieval, pages 127–150. Kluwer Academic Publishers, Hingham, MA, USA, 2000.

[CC01] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems (TOIS)*, 19(2):97–130, 2001.

[CCP03a] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. A Service for Supporting Virtual Views of Large Heterogeneous Digital Libraries. In Traugott Koch and Ingeborg Sølvberg, editors, *7th European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2003*, Lecture Notes in Computer Science, pages 362–373, Trondheim, Norway, August 2003. Springer-Verlag.

[CCP03b] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. The Cyclades Collection Service. Technical Report 2003-TR-68, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, 2003.

[CCP04] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. Enhancing the OpenDLib Search Service. In Rachel Heery and Liz Lyon, editors, *8th European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2004*, Lecture Notes in Computer Science, pages 353–365, Bath, UK, September 2004. Springer-Verlag.

[CCP06a] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. Digital Library Reference Model. Project Report 2006-PR-02, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, January 2006.

[CCP06b] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. Digital Library System Reference Architecture. Project Report 2006-PR-01, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, January 2006.

[CCPS05a] Leonardo Candela, Donatella Castelli, Pasquale Pagano, and Manuele Simi. From Heterogeneous Information Spaces to Virtual Documents. In Edward A. Fox, Erich J. Neuhold, Pimrumpai Premsmit, and Vilas Wuwongse, editors, *Digital Libraries: Implementing Strategies and Sharing Experiences, 8th International Conference on Asian Digital Libraries, ICADL 2005*, Lecture Notes in Computer Science, pages 11–22, Bangkok, Thailand, December 2005. Springer.

[CCPS05b] Leonardo Candela, Donatella Castelli, Pasquale Pagano, and Manuele Simi. Moving Digital Library Service Systems to the Grid. In Can Türker, Maristella Agosti, and Hans-Jörg Schek, editors, *Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures*, number 3664 in Lecture Notes in Computer Science, pages 236–259. Springer Verlag, 2005.

[CCPS05c] Donatella Castelli, Leonardo Candela, Pasquale Pagano, and Manuele Simi. DILIGENT: A DL Infrastructure for Supporting Joint Research. In IEEE Computer Society, editor, *2nd IEEE-CS International Symposium Global Data Interoperability - Challenges and Technologies*, pages 56–69, 2005.

[CDISS05] Birte Christensen-Dalsgaard, Yannis Ioannidis, Heiko Schuldt, and Dagobert Soergel. Recommendations and Observations for a European Digital Library (EDL). Technical report, DELOS Network of Excellence on Digital Libraries, December 2005.

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks . In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995. ACM Press.

[CLE] CLEF. Cross Language Evaluation Forum. `http://www.clef-campaign.org/`.

[CMP02] Donatella Castelli, Carlo Meghini, and Pasquale Pagano. Foundations of a Multidimensional Query Language for Digital Libraries. In Maristella Agosti and Costantino Thanos, editors, *6th European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2002*, Lecture Notes in Computer Science, pages 251–265, Rome, Italy, September 2002. Springer-Verlag.

[Con02] Consultative Committee for Space Data Systems. Reference Model for an Open Archival Information System. Technical Report CCSDS 650.0-B-1, National Aeronautics and Space Administration, January 2002. Blue Book.

[CP02] Donatella Castelli and Pasquale Pagano. OpenDLib: A Digital Library Service System. In Maristella Agosti and Costantino Thanos, editors, *6th European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2002*, Lecture Notes in Computer Science, pages 292–308, Rome, Italy, September 2002. Springer-Verlag.

[CP03] Donatella Castelli and Pasquale Pagano. A System for Building Expandable Digital Libraries. In *ACM/IEEE 2003 Joint Conference on Digital Libraries (JCDL 2003)*, pages 335–345. Springer-Verlag, 2003.

[CS04] Leonardo Candela and Umberto Straccia. The Personalized, Collaborative Digital Library Environment CYCLADES and its Collections Management. In Jamie Callan, Fabio Crestani, and Mark Sanderson, editors, *Multimedia Distributed Information Retrieval*, number 2924 in Lecture Notes in Computer Science, pages 156–172. Springer Verlag, 2004.

[CYC] CYCLADES. An Open Collaborative Virtual Archive Environment. `http://www.ercim.org/cyclades`. IST-2000-25456.

[dBGI04] Alberto del Bimbo, Stefan Gradmann, and Yannis Ioannidis. Toward a Long Term Agenda for Digital Library Research. Brainstorming Workshop Report - Corvara - Italy, DELOS Network of Excellence on Digital Library, July 2004.

[dBPLF04] Jos de Bruijn, Axel Polleres, Ruben Lara, and Dieter Fensel. OWL Flight. Technical report, Working Draft D20.3 v0.1, 2004.

[DC] Dublin Core Metadata Initiative. `http://dublincore.org`.

[DDC] Dewey Decimal Classification. `http://www.oclc.org/dewey`.

[DEL] DELOS. Network of Excellence on Digital Libraries. `http://www.delos.info`. G038-507618.

[Deu89] L. P. Deutsch. Design Reuse and Frameworks in the Smalltalk-80 System. In T. J. Biggerstaff and A. J. Perlis, editors, *Software Reusability*, pages 57–71. ACM press, New York, 1989.

[DIL] DILIGENT. A DIgital Library Infrastructure on Grid ENabled Technology. `http://www.diligentproject.org/`. IST-2003-004260.

[DL96] J. Davis and Carl Lagoze. The Networked Computer Science Technical Report Library. Technical Report TR94-1595, Department of Computer Science, Cornell University, July 1996.

[DMD+03] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the Semantic Web. *The VLDB Journal*, 12(4):303–319, 2003.

[DSB+04] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language:

Reference. Technical report, W3C Recommendation, 2004. http://www.w3.org/TR/owl-ref/.

[Dub] Dublin Core Metadata Initiative. Dublin Core Metadata element set, version 1.1: Reference description. http://dublincore.org/documents/dces/.

[EGEa] EGEE. Enabling Grids for E-sciencE. http://public.eu-egee.org/. INFSO 508833.

[EGEb] EGEE. gLite: Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite/.

[EGM97] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive Datalog. *ACM Transaction on Database Systems*, 22(3):364–418, September 1997.

[FAFL95] Edward A. Fox, Robert M. Akscyn, Richard Furuta, and John J. Leggett. Digital Libraries. *Communications of the ACM*, 38(4):23–28, April 1995.

[FFS+01] D. Faensen, L. Faultstich, H. Schweppe, A. Hinze, and A. Steidinger. Hermes: a notification service for digital libraries. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 373–380, New York, NY, USA, 2001. ACM Press.

[FHM+01] Norbert Fuhr, Preben Hansen, Michael Mabe, Andras Micsik, and Ingeborg Sølvberg. Digital Libraries: A Generic Classification and Evaluation Scheme. In *ECDL '01: Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, pages 187–199, London, UK, 2001. Springer-Verlag.

[FK04] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan-Kaufmann, 2004.

[FKC03] D. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-based Access Control*. Computer Security Series. Artech House, 2003.

[FKNT02] Ian Foster, Carl Kesselman, Jeffrey Nick, and Steve Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

[FKT01] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[FM98] Edward A. Fox and Gary Marchionini. Toward a Worldwide Digital Library. *Communications of the ACM*, 41(4):29–32, April 1998.

[FPC+99] James C. French, Allison L. Powell, Jamie Callan, Charles L. Viles, Travis Emmitt, Kevin J. Prey, and Yun Mou. Comparing the performance of database selection algorithms. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245. ACM Press, 1999.

[Fri03] Amy Friedlander. Knowledge Lost in Information. Technical report, University of Pittsburg, June 2003. Report of the NSF Workshop on Research Directions for Digital Libraries.

[Gac03] A. Gachet. Software Frameworks for Developing Decision Support Systems - A New Component in the Classification of DSS Development Tools. *Journal of Decision Systems*, 12(3/4):271–281, 2003.

[GF02] M. A. Gonçalves and E. A. Fox. 5SL - A Language for Declaratively Specification and Generation of Digital Libraries. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02)*, pages 263–272, Portland, Oregon, July 2002.

[GFW04] M. A. Gonçalves, E. A. Fox, and L. T. Watson. Toward a digital library theory: A formal digital library ontology. In *Proceedings of the ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, Sheffield, England, 2004.

[GFWK04] M. A. Gonçalves, E. A. Fox, L. T. Watson, and N. A. Kipp. Streams, Structures, Spaces, Scenarios, Societies (5S): A Formal Model for Digital Libraries. *ACM Transactions on Information Systems (TOIS)*, 22(2):270–312, 2004.

[GGMM02] Gary Geisler, Sarah Giersch, David McArthur, and Marty McClelland. Creating Virtual Collections in Digital Libraries: Benefits and Implementation Issues. In *Proceedings of the second ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 210–218. ACM Press, 2002.

[Gla06] H. M. Gladney. Principles for Digital Preservation. *Communications of the ACM*, 49(2):111–116, February 2006.

[Glo] Globus Alliance. The Globus Alliance Website. `http://www.globus.org/`.

[Gon04] Marcos André Gonçalves. *Streams, Structures, Spaces, Scenarios, and Societies (5S): A Formal Digital Library Framework and Its Applications*. PhD thesis, Virginia Polytechnic Institute and State University, November 2004.

[HCOC02] Zan Huang, Wingyan Chung, Thian-Huat Ong, and Hsinchun Chen. A graph-based recommender system for digital library. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73, New York, NY, USA, 2002. ACM Press.

[IFL] IFLA Study Group on the Functional Requirements for Bibliographic Records. Functional Requirements for Bibliographic Records: Final Report. `http://www.ifla.org/VII/s13/frbr/frbr.htm`.

[IMA+05] Yannis Ioannidis, David Maier, Serge Abiteboul, Peter Buneman, Susan Davidson, Edward Fox, Alon Halevy, Craig Knoblock, Fausto Rabitti, Hans Schek, and Gerhard Weikum. Digital library information-technology infrastructures. *International Journal on Digital Libraries*, 5(4):266 – 274, August 2005.

[Ioa05] Yannis Ioannidis. Digital libraries at a crossroads. *International Journal on Digital Libraries*, 5(4):255–265, August 2005.

[JF88] R. E. Johnson and B. Foote. Designing reusable classes. *Journal of object-oriented programming*, 1(2):22–35, 1988.

[JF02] Greg Janée and James Frew. The ADEPT digital library architecture. In *Proceeding of the second ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 342–350. ACM Press, 2002.

[JW04] Ian Jacobs and Norman Walsh. Architecture of the World Wide Web, Volume One. Technical report, W3C, December 2004.

[KC96] Terry Kuny and Gary Cleveland. The Digital Library: Myths and Challenges. In *62nd IFLA General Conference*, August 25-31 1996.

[KGF03] R. Kelapure, M. A. Gonçalves, and E. A. Fox. Scenario-based Generation of Digital Library Services. In Traugott Koch and Ingeborg Sølvberg, editors, *7th European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2003*, Lecture Notes in Computer Science, Trondheim, Norway, August 2003. Springer-Verlag.

[LD95] C. Lagoze and J. R. Davis. Dienst - An Architecture for Distributed Document Libraries. *Communication of the ACM*, 38(4):47, 1995.

[Len02] Maurizio Lenzerini. Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA, 2002. ACM Press.

[Les99] Michael Lesk. Expanding Digital Library Research: Media, Genre, Place and Subjects. In *Proceedings of the International Symposium on Digital Library 1999: ISDL'99*, pages 51–57, Tsukuba, Ibaraki, Japan, September 28-29 1999.

[LF98] Carl Lagoze and David Fielding. Defining Collections in Distributed Digital Libraries. *D-Lib Magazine*, November 1998.

[Lic65] J. C. R. Licklider. *Libraries of the Future*. MIT Press, 1965.

[LKPJ05] C. Lagoze, D. B. Krafft, S. Payette, and S. Jesuroga. What Is a Digital Library Anyway? *D-Lib Magazine*, 11(11), November 2005.

[LN05] Greg Lomow and Eric Newcomer. *Understanding SOA with Web Services*. Independent Technology Guides. Addison Wesley Professional, 2005.

[LOM] IEEE Standard for Learning Object Metadata. `http://ltsc.ieee.org/wg12/par1484-12-1.html`.

[LPSW05] C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: An Architecture for Complex Objects and their Relationships. *Journal of Digital Libraries, Special Issue on Complex Objects*, 2005.

[LV01] Carl Lagoze and Herbert Van de Sompel. The open archives initiative: building a low-barrier interoperability framework. In *Proceedings of the first ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 54–62. ACM Press, 2001.

[MAR05] MARC Standards Web Page. `http://www.loc.gov/marc/`, September 2005.

[MBH+04] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIIraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic Markup for Web Services. `http://www.daml.org/services/owl-s`, 2004.

[MLM+06] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter Brown, and Rebekah Metz. Reference Model for Service Oriented Architecture 1.0. Technical report, OASIS, February 2006. Public Review Draft 1.0.

[MM04] Frank Manola and Eric Miller. RDF primer. Technical report, W3C Recommendation, 2004. http://www.w3.org/TR/rdf-primer/.

[MS04] E. Michael Maximilien and Munindar P. Singh. A Framework and Ontology for Dynamic Web Services Selection. *IEEE Internet Computing*, 8(5):84–93, September-October 2004.

[MT00] Nemad Medvidovic and Richard N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transaction on Software Engineering*, 28(1):70–93, January 2000.

[NF01] Henrik Nottelmann and Norbert Fuhr. Learning probabilistic datalog rules for information classification and transformation. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 387–394, New York, NY, USA, 2001. ACM Press.

[NG84] Joseph D. Novak and D. Bob Gowin. *Learning How to Learn*. Cambridge University Press, 1984.

[NM02] Srini Narayanan and Sheila A. McIlraith. Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, New York, NY, USA, 2002. ACM Press.

[NS05] Henrik Nottelmann and Umberto Straccia. sPLMap: A probabilistic approach to schema matching. In David E. Losada and Juan M. Fernndez Luna, editors, *27th European Conference on Information Retrieval Research (ECIR 2005)*, 2005.

[OAI] The OAI Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/openarchivesprotocol.html.

[Ope] OpenDLib. A Digital Library Service System. http://www.opendlib.com/.

[PBB+04] M. Pistore, F. Bardon, P. Bertoli, D. Shaparau, and P. Traverso. Planning and Monitoring Web Service Composition. In *Workshop on Planning and Scheduling for Web and Grid Services in conjunction with ICAPS 2004*, 2004.

[PCWGM98] Andreas Paepcke, Chen-Chuan K. Chang, Terry Winograd, and Héctor García-Molina. Interoperability for digital libraries worldwide. *Communications of the ACM*, 41(4):33–42, 1998.

[PRE05] PREMIS. Data Dictionary for Preservation Metadata. Final Report of the PREMIS Working Group, OCLC and RLG, May 2005.

[PSH] Peter F. Patel-Schneider and Ian Horrocks. A Comparison of Two Modelling Paradigms. `http://www.cs.man.ac.uk/~horrocks/Publications/download/2005/HoPa05a.pdf`.

[PT02] Sandra Payette and Staples Thornton. The Mellon Fedora Project: Digital Library Architecture Meets XML and Web Services. In Maristella Agosti and Costantino Thanos, editors, *Research and Advanced Technology for Digital Libraries, 6th European Conference, ECDL 2002, Rome, Italy, September 2002, Proceedings*, Lecture Notes in Computer Science, pages 406–421. Springer-Verlag, 2002.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[RS02] Maria Elena Renda and Umberto Straccia. A personalized collaborative digital library environment. In *5th International Conference on Asian Digital Libraries (ICADL2002)*, number 2555 in Lecture Notes in Computer Science, pages 262–274. Springer-Verlag, 2002.

[SAG+01] Hussein Suleman, Anthony Atkins, Marcos André Gonçalvess, Robert K. France, Edward A. Fox, Vinod Chachra, Murray Crowder, and Jeff Young. Networked Digital Library of Theses and Dissertations: Bridging the Gaps for Global Access. *D-Lib Magazine*, 7(9), September 2001.

[SC02] Luo Si and Jamie Callan. Using Sampled Data and Regression to Merge Search Engine Results. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM Press, 2002.

[TBS03a] Robert Tansley, Mick Bass, and MacKenzie Smith. DSpace as an Open Archival Information System: Current Status and Future Directions. In Traugott Koch and Ingeborg Sølvberg, editors, *Research and Advanced Technology for Digital Libraries, 7th European Conference, ECDL 2003, Trondheim, Norway, August 17-22, 2003, Proceedings*, Lecture Notes in Computer Science, pages 446–460. Springer-Verlag, 2003.

[TBS+03b] Robert Tansley, Mick Bass, David Stuve, Margret Branschofsky, Daniel Chudnov, Greg McClellan, and MacKenzie Smith. The DSpace Institutional Digital Repository System: current functionality. In *Proceedings of the third ACM/IEEE-CS joint conference on Digital libraries*, pages 87–97. IEEE Computer Society, 2003.

[TCS01] Yannis Tzitzikas, Panos Constantopoulos, and Nicolas Spyratos. Mediators over Ontology-Based Information Sources. In *WISE (1)*, pages 31–40, 2001.

[The02] The Library of Congress. Metadata Encoding and Transmission Standard. `http://www.loc.gov/standards/mets/`, February 2002.

[TKP04] Giannis Tsakonas, Sarantos Kapidakis, and Christos Papatheodorou. Evaluation of User Interaction in Digital Libraries. In Maristella Agosti and Norbert Fuhr, editors, *Revised Notes of the DELOS WP7 Workshop on the Evaluation of Digital Libraries*, Padua, Italy, November 2004.

[Tzi02] Yannis Tzitzikas. *Collaborative Ontology-based Information Indexing and Retrieval*. PhD thesis, Department of Computer Science, University of Crete, September 2002.

[Ull88] Jeffrey D. Ullmann. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1988.

[WB02] Ian H. Witten and David Bainbridge. *How to Build a Digital Library*. Elsevier, 2002.

[WBB01] Ian H. Witten, David Bainbridge, and Stefan J. Boddie. Power to the People: End-user Building of Digital Library Collections. In *Proceedings of the first ACM/IEEE-CS joint conference on Digital libraries*, pages 94–103. ACM Press, 2001.

[Wie95] G. Wiederhold. Digital libraries, value, and productivity. *Communication of the ACM*, 38(4):85–96, 1995.

[WSY+02] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 136–147, New York, NY, USA, 2002. ACM Press.

[WWC92] G. Wiederhold, P. Wegner, and S. Ceri. Toward Megaprogramming. *Communication of the ACM*, 38(11):89–99, November 1992.

[XCLN98] Jian Xu, Yinyan Cao, Ee-Peng Lim, and Wee-Keong Ng. Database selection techniques for routing bibliographic queries. In *Proceedings of the third ACM conference on Digital Libraries*, pages 264–274. ACM Press, 1998.

[YL97] Budi Yuwono and Dik Lun Lee. Server Ranking for Distributed Text Retrieval Systems on the Internet. In *Database Systems for Advanced Applications*, pages 41–50, 1997.

[Z3903] ANSI/NISO Z39.50 Information retrieval: Application Service Definition & Protocol Specification. NISO Press, National Information Standards Organization, 2003.

[ZCF+97] Carlo Zaniolo, Stefano Ceri, Christos Faloutsos, Richard T. Snodgrass, V. S. Subrahmanian, and Roberto Zicari. *Advanced Database Systems*. Data Management Systems Series. Morgan Kaufmann, 1997.

[ZGS+03] Q. Zhu, M. A. Gonçalves, R. Shen, L. Cassell, and E. A. Fox. Visual Semantic Modeling of Digital Libraries. In Traugott Koch and Ingeborg Sølvberg, editors, *7th European Conference on Research and AdvancedTechnology for Digital Libraries, ECDL 2003*, Lecture Notes in Computer Sciences, pages 325–337, Trondheim, Norway, August 2003. Springer-Verlag.