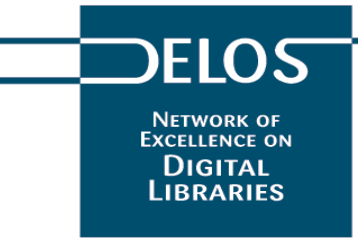
DELOS

Network of Excellence on Digital Libraries

The DELOS Digital Library Reference Model

Foundations for Digital Libraries

L. Candela, D. Castelli, N. Ferro, Y. Ioannidis, G. Koutrika, C. Meghini, P. Pagano, S. Ross, D. Soergel
M. Agosti, M. Dobreva, V. Katifori, H. Schuldt



Project no. 507618

DELOS

A Network of Excellence on Digital Libraries

Instrument: Network of Excellence

Thematic Priority: IST-2002-2.3.1.12
Technology-enhanced Learning and Access to Cultural Heritage

The DELOS Digital Library Reference Model

Foundations for Digital Libraries

Version 0.98

December 2007

Project co-funded by the European Commission within the Sixth Framework Programme
(2002-2006)

This document is one of a series of texts written in the context of the DELOS Network of Excellence on Digital Libraries (www.delos.info).

The DELOS Digital Library Reference Model

Foundations for Digital Libraries

L. Candela,[1] D. Castelli,[1] N. Ferro,[2] Y. Ioannidis,[3] G. Koutrika,[3] C. Meghini,[1] P. Pagano,[1] S. Ross,[4] D. Soergel[5]

M. Agosti,[2] M. Dobreva,[4] V. Katifori,[3] H. Schuldt[6]

[1] Institute of Information Science and Technologies (ISTI)
Italian National Research Council (CNR)
Pisa, Italy

[2] Department of Information Engineering (DEI)
University of Padova
Padova, Italy

[3] Department of Informatics and Telecommunications
University of Athens
Athens, Greece

[4] Humanities Advanced Technology and Information Institute (HATII)
University of Glasgow
Glasgow, United Kingdom

[5] College of Information Studies
University of Maryland
College Park, Maryland

[6] Database and Information Systems Group
University of Basel
Basel, Switzerland

Table of Contents

Table of Figures

About this Volume

The Digital Library universe is a complex framework. The growth and evolution of this framework in terms of approaches, solutions and systems has led to the need for common foundations capable of setting the basis for better understanding, communicating and stimulating further evolution in this area. The DELOS Digital Library Reference Model aims at contributing to the creation of such foundations. It exploits the collective understanding on Digital Libraries that has been acquired by European research groups active in the Digital Library field for many years, both within the DELOS Network of Excellence and beyond, as well as by other groups around the world. It identifies the set of concepts and relationships that characterise the essence of the Digital Library universe. This model should be considered as a roadmap allowing the various players involved in the Digital Library domain to follow the same route and share a common understanding in dealing with the entities of such a universe.

This volume presents the DELOS Reference Model by introducing the principles governing it as well as the set of concepts and relationships that collectively capture the intrinsic nature of the various entities of the Digital Library universe. Because of the broad coverage of the Digital Library universe, its evolving nature, and the lack of any previous agreement on its foundations, the Reference Model is by necessity a dynamic framework, as is also this document. Continuous evolutions of the document are envisaged in order to obtain a number of well-formed and consolidated definitions, shared by the Digital Library community.

The Structure of the Volume

The volume is organised in three parts, each potentially constituting a document in its own right. Each of the three parts describes the Digital Library universe from a different perspective that is driven by a trade-off between abstraction and concretisation. Thus each part is equally important in capturing the nature of the Digital Library universe. The second part is based on the first one, and the third part is based on the second, i.e. they rely on the notions described previously when introducing additional information that characterises these notions more precisely. In particular, 'PART I The Digital Library Manifesto', already published as a separate document in January 2006, introduces the main notions characterising the whole Digital Library universe in quite abstract terms; 'PART II The DELOS Digital Library Reference Model in a Nutshell' treats these notions in more detail by introducing the main concepts and relationships related to each of the aspects captured by the previous one; finally, 'PART III The DELOS Digital Library Reference Model Concepts and Relations' describes each of the identified concepts and relations in detail by explaining their rationale as well as presenting examples of their instantiation in concrete scenarios.

Although it is possible to choose different routes through the volume, or simply focus on a single part, the whole is structured so that it can be read from cover to cover.

Section I.1 introduces 'PART I The Digital Library Manifesto' by providing the driving force pervading the whole activity.

Section I.2 presents the relationships between three types of relevant 'systems' in the Digital Library universe, namely Digital Library (DL), Digital Library System (DLS) and Digital Library Management System (DLMS).

Section I.3 describes the main concepts characterising the above three systems and thus the whole Digital Library universe, i.e. content, user, functionality, quality, policy and architecture.

Section I.4 introduces the main roles actors may play within digital libraries, i.e. end-user, designer, administrator and application developer.

Section I.5 describes the reference frameworks needed to clarify the DL universe at different levels of abstraction, i.e. the Digital Library Reference Model and the Digital Library Reference rchitecture.

Section I.6 records concluding remarks on The Digital Library Manifesto.

Section II.1 introduces 'PART II The DELOS Digital Library Reference Model in a Nutshell' by summarising the content of the Manifesto and setting the basis for reading and using the rest of this part.

Section II.2 presents the constituent domains by briefly describing their rationale and providing each of them with a concept map that records the main related concepts and relations connecting them.

Section II.3 introduces the reader to possible exploitations of the model. In particular, it addresses Interoperability and Preservation issues. For each, it describes the problem by pointing out the instruments the Reference Model makes available for dealing with it.

Section II.4 discusses related works. In particular, this section highlights the similarities and differences between this Reference Model and similar initiatives like the 5S Framework and the CIDOC Conceptual Reference Model.

Section II.5 records concluding remarks on the DELOS Digital Library Reference Model as presented in PART II.

Section III.1 introduces 'PART III The DELOS Digital Library Reference Model Concepts and Relations' by highlighting the role of this part.

Section III.2 presents the hierarchy of Concepts constituting the Reference Model.

Section III.3 provides a definition for each of the 218 Concepts currently constituting the model. Each definition is complemented by the list of relations connecting the concept to the other concepts, the rationale for including this concept in the model, and examples of concrete instances of the concept in real-life scenarios.

Section III.4 presents the hierarchy of the identified Relations.

Section III.5 provides a definition for each of the 52 Relations currently constituting the model. Each definition is complemented by the rationale for including it in the model and some examples of concrete instances in real-life scenarios.

The Section of Conclusions sums up the volume.

Appendix A provides the concept maps of the Reference Model in A4 format to improve their readability.

Appendix B provides the concept maps of the Reference Model expressed in terms of UML Class Diagrams to both demonstrate the equivalence of the Concept Maps and UML from the perspective of this model and provide readers accustomed to using UML with a representation that is familiar to them.

Acknowledgements

Many people have contributed to this activity at different levels.

First and foremost, the authors wish to thank the DELOS Network of Excellence on Digital Libraries Consortium that gave them the chance to work on the fascinating topic of building foundations for Digital Libraries. In particular, we want to thank Costantino Thanos (CNR-ISTI), Coordinator of the DELOS Network of Excellence, and his Deputy, Vittore Casarosa

(CNR-ISTI), for their continuous encouragement and for the many useful discussions on the proposed conceptualisation.

The authors also wish to acknowledge the considerable input to the model received from the participants in the DELOS Reference Model Workshop held in Frascati (Rome) in June 2006. The comments, visions and insight received on the initial release of the model have been very helpful for the rest of the activity. The workshop participants comprised, besides the authors: José Borbinha (DEI-IST-UTL), Martin Braschler (Zurich University of Applied Sciences Winterthur), Vittore Casarosa (ISTI-CNR), Tiziana Catarci (Università degli Studi di Roma 'La Sapienza'), Stavros Christodoulakis (Technical University of Crete), Edward Fox (Virginia Tech), Norberth Fuhr (Universität Duisburg-Essen), Stefan Gradmann (Universität Hamburg), Ariane Labat (EC), Mahendra Mahey (UKOLN), Patricia Manson (EC), Andy Powell (UKOLN), Hans-Jörg Schek (ETH), MacKenzie Smith (MIT Libraries), Costantino Thanos (ISTI-CNR) and Theo van Veen (National Library of the Netherlands).

This volume has also benefited from the comments and ideas discussed during two workshops entitled 'Foundations of Digital Libraries'. The two workshops were held, respectively, in conjunction with the ACM IEEE Joint Conference on Digital Libraries (JCDL 2007) and the 11th European Conference on Research and Advanced Technologies on Digital Libraries (ECDL 2007). As it is not possible to list all the participants individually, the authors of this document wish to express their thanks to all of them collectively and only explicitly mention the particularly helpful comments received from Edward Fox (Virginia Tech), Geneva Henry (Rice University) and Marianne Backes (Centre Virtuel de la Connaissance sur l'Europe).

Thanks to Perla Innocenti (HATII) for reviewing the sections on Policy and drawing our attention to the fact that we had not represented the time dimensions.

Thanks also to Professor Stavros Christodoulakis (Technical University of Crete) for actively participating in some of the Reference Model internal meetings and raising useful comments about the management of multimedia content, and to Professor Hans-Jörg Schek (University of Konstanz), leader of the DELOS work package that gave rise to the Reference Model activity, for the key contribution he has made to the start-up of this activity and for his useful comments.

We are particularly grateful to Maria Bruna Baldacci (ISTI-CNR) for her considerable help in improving the readability of this volume and to Francesca Borri (ISTI-CNR) for her contribution to the graphical editing of this and the other volumes forming the Reference Model document suite.

PART I The Digital Library Manifesto

I.1 Introduction

The term 'Digital Library' is currently used to refer to systems that are heterogeneous in scope and yield very different functionality. These systems range from digital object and metadata repositories, reference-linking systems, archives, and content administration systems (mainly developed by industry) to complex systems that integrate advanced digital library services (mainly developed in research environments). This 'overloading' of the term 'Digital Library' is a consequence of the fact that as yet there is no agreement on what Digital Libraries are and what functionality is associated with them. This results in a lack of interoperability and reuse of both content and technologies. This document attempts to put some order in the field for the benefit of its future advancement.

I.1.1 What is a Manifesto?

According to the Merriam-Webster Dictionary, a *manifesto* is 'a written statement declaring publicly the intentions, motives, or views of its issuer'. Similarly, according to Wikipedia, a manifesto is 'a public declaration of principles and intentions'. The *Declaration of the Rights of Man and the Citizen* in France in 1789 and the *Declaration of Independence* in the US in 1776 are two well-known manifestos that have set the stage for the establishment of two major countries and have had a major influence on the recent history of the world. The production of manifestos in subsequent centuries has in fact increased: *The Communist Manifesto*, issued by K. Marx and F. Engels in 1848, and the *The Russell-Einstein Manifesto*, issued by B. Russell and A. Einstein in 1955 to confront the development of weapons of mass destruction, are some of the most famous examples.

Of smaller scope and within the realm of science, there have also been several manifestos, which have tried to provide direction for the development of particular research areas. These have taken more the form of declarations of axioms capturing the strategic ideas of a group of people with respect to a certain topic or field. Examples include the following:

- *The Third Manifesto*, from the book 'Databases, Types, and the Relational Model: The Third Manifesto' by H. Darwen and C.J. Date, Addison-Wesley, 2007, which proposes new foundations for future database systems.[1]
- *The Object-Oriented Database System Manifesto*, which describes the main features and characteristics that a system must have to qualify as an object-oriented database system, touching upon mandatory, optional and even open points where the designer can make several choices.[2]
- *The Manifesto for Agile Software Development*, which attempts to discover better ways of developing software by putting emphasis on different items from the traditional ones, e.g. individuals and interactions instead of processes and tools, working software instead of comprehensive documentation, and others.[3]
- *The GNU Manifesto*, by Richard Stallman, which uses as an axiom the idea that '… the golden rule requires that if I like a program I must share it with other people who like it' to produce a complete Unix-compatible software system freely available to everyone who wants to use it.[4]

---

[1] http://www.thethirdmanifesto.com/

[2] http://www.cs.cmu.edu/People/clamen/OODBMS/Manifesto/index.html

[3] http://agilemanifesto.org/

[4] http://www.gnu.org/gnu/manifesto.html

Relevant research and industrial efforts in the Digital Library (DL) field have now reached an advanced, though heterogeneous, stage of development, thus the time is right for this field to obtain its own Manifesto.

I.1.2 Motivation

Digital Libraries constitute a relatively young scientific field, whose life spans roughly the last fifteen years. Instrumental to the birth and growth of the field have been the funding opportunities generated by the 'Technology Enhanced Learning; Cultural Heritage' (formerly 'Cultural Heritage Applications') Unit of the Information Society Directorate-General of the European Commission and the 'Digital Library Initiatives' in the United States sponsored by the National Science Foundation and other agencies.

Digital Libraries represent the meeting point of many disciplines and fields, including data management, information retrieval, library sciences, document management, information systems, the Web, image processing, artificial intelligence, human–computer interaction and digital curation. It was only natural that these first fifteen years were mostly spent on bridging some of the gaps between the disciplines (and the scientists serving each one), improvising on what 'Digital Library functionality' is supposed to be, and integrating solutions from each separate field into systems to support such functionality, sometimes the solutions being induced by novel requirements of Digital Libraries. These have been achieved through much exploratory work, primarily in the context of focused efforts devising specialised approaches to address particular aspects of Digital Library functionality. For example, the ARTISTE project [13] from Europe's Fifth Framework Programme focused on how to develop an integrated analysis and navigation environment for art images and analogous multimedia content, the COLLATE project [60] from the same Programme focused on how to deal with old film libraries, while the Alexandria Project [9] from NSF's DLI-1 and DLI-2 Programs focused on geospatially referenced multimedia material. For the most part, every effort so far has been distinct and, in some sense, isolated from the rest. Every project has started from scratch to build a system supporting the particular needs specified in the project's description. Nevertheless, looking back at the individual achievements of all the projects, it can clearly be seen that there is substantial commonality among many of them; the bottom-up development of the field so far has provided enough 'data points' for patterns to emerge that can encapsulate all efforts.

Despite the young age of the field of Digital Libraries, it has made a long journey from its initial conception to the present state of the art and has reached a level of maturity that did not exist fifteen years ago. Substantial knowledge and experience have been accumulated. This warrants a process of self-declaration that will identify the principal ideas behind the field; it is time for a *Digital Library Manifesto* to set the ground rules for the field and lead to the development of reference documents that will capture the full spectrum of concepts that play a role in Digital Libraries.

As mentioned earlier, the nature of Digital Libraries is highly multidisciplinary. Naturally, this has created several conceptions of what a Digital Library is, each one influenced by the perspective of the primary discipline of the conceiver(s). In fact, Fox *et al*. [83] observe that the expression 'Digital Library' evokes a different impression in each person, ranging from the simple computerisation of traditional libraries to a space in which people communicate, share and produce new knowledge and knowledge products. For instance, the 1st Delos Brainstorming Workshop in San Cassiano, Italy, envisages a Digital Library as a system that enables any citizen to access all human knowledge, any time and anywhere, in a friendly, multi-modal, efficient and effective way, by overcoming barriers of distance, language and

culture and by using multiple Internet-connected devices [119]. An offspring of that activity concludes that Digital Libraries can become the universal knowledge repositories and communication conduits of the future, a common vehicle by which everyone will access, discuss, evaluate and enhance information of all forms [120][121]. Likewise, in his framework for Digital Library research, Soergel [190] starts from three very different perspectives that different people in the community have on Digital Libraries, i.e. as tools to serve research, scholarship and education, as a means for accessing information, and as providing services primarily to individual users. He then enhances each one further and fuses them all together to obtain the main guiding principles for his vision of the field. On the other hand, Belkin [23] states that a Digital Library is an institution responsible for providing at least the functionality of a traditional library in the context of distributed and networked collections of information objects. Lesk [149] analyses and discusses the importance of the terms 'Digital' and 'Library' in the expression 'Digital Library', where the former term mainly implies the existence of software for searching text, while the latter term refers to existing material that has been scanned for online access, and concludes that the research effort in the field is not usually associated with the users' needs. Borgman [36] 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 [140] discuss four myths about Digital Libraries and attempt to bring them down: (i) the Internet is 'The' Digital Library; (ii) at some point there will be a single Digital Library or a single-window view of Digital Library collections; (iii) Digital Libraries are means to provide more equitable access to content from anywhere at any time; and (iv) Digital Libraries are cheaper instruments than physical libraries. They conclude that Digital Libraries impose reinvention of the role of librarians and library models.

In addition to such a variety of perspectives that may currently exist on what a Digital Library is, the concept has evolved quite substantially since the early idea of a system providing access to digitised books and other text documents. The DELOS Network of Excellence on Digital Libraries [66] now envisages a Digital Library as a tool at the centre of intellectual activity having no logical, conceptual, physical, temporal or personal borders or barriers on information. It has moved from a content-centric system that simply organises and provides access to particular collections of data and information to a person-centric system that aims to provide interesting, novel, personalised experiences to users. Its main role has shifted from static storage and retrieval of information to facilitation of communication, collaboration and other forms of interaction among scientists, researchers or the general public on themes that are pertinent to the information stored in the Digital Library. Finally, it has moved from handling mostly centrally located text to synthesising distributed multimedia document collections, sensor data, mobile information and pervasive computing services.

This vision of Digital Libraries seems to resonate well with the concept of 'Information Space' that has arisen from the field of Computer Supported Cooperative Work (CSCW). Snowdon, Churchill and Frecon [193] have developed future visions about 'Connected Communities' and 'Inhabited Information Spaces', with the latter being closely related to the vision of Digital Libraries, in that ubiquitous information is a prerequisite for CSCW. In more detail, Inhabited Information Spaces are 'spaces and places where people and digital data can meet in fruitful exchange, i.e. they are effective social workspaces where digital information can be created, explored, manipulated and exchanged'. Thus, 'in Inhabited Information Spaces, both information and people who are using that information (viewing it, manipulating it) are represented. This supports collaborative action on objects, provides awareness of others' ongoing activities, and offers a view of information in the context of its use'. Based on

the above and according to the aforementioned DELOS vision of a Digital Library, the latter provides an Information Space that is populated by a user community and becomes an Inhabited Information Space through CSCW technology. The two fields complement each other nicely, in that one focuses on access and provision of relevant information while the other focuses on visualisation and sharing of information.

It becomes obvious that, as envisaged, 'Digital Library' is a complex notion with several diverse aspects and cannot be captured by a simple definition. A comprehensive representation encapsulating all potential perspectives is required. This has led to the drafting of *The Digital Library Manifesto*, whose aim is to set the foundations and identify the cornerstone concepts within the universe of Digital Libraries, facilitating the integration of research and proposing better ways of developing appropriate systems. Having this broad scope, the Manifesto is followed by a set of separate reference documents, which stand individually but can also be seen as parts of a whole.

The *Manifesto* exploits the collective understanding of Digital Libraries developed by European research groups, including those that are partners in DELOS, and the results of DELOS working meetings (e.g. San Cassiano in 2001, Corvara in 2004 and Frascati in 2006).

The rest of Part I of this volume presents the core parts of this Manifesto and introduces central aspects of the Digital Library framework. It first presents an examination of the three types of relevant 'systems' in this area: Digital Library, Digital Library System, and Digital Library Management System (Section I.2). It then describes the main concepts characterising the above systems, i.e. content, user, functionality, quality, policy and architecture (Section I.3), and it introduces the main roles that actors may play within digital libraries, i.e. end-user, designer, administrator and application developer (Section I.4). In Section I.5 it describes the reference frameworks that are needed to clarify the DL universe at different levels of abstraction, i.e. the Digital Library Reference Model and the Digital Library Reference Architecture. Finally, Section I.6 concludes the Manifesto part.

I.2 The Digital Library Universe: A Three-tier Framework

A Digital Library is an evolving organisation that comes into existence through a series of development steps that bring together all the necessary constituents. Figure I.2-1 presents this process and indicates three distinct notions of 'systems' developed along the way forming a three-tier framework: Digital Library, Digital Library System, and Digital Library Management System. These correspond to three different levels of conceptualisation of the universe of Digital Libraries.

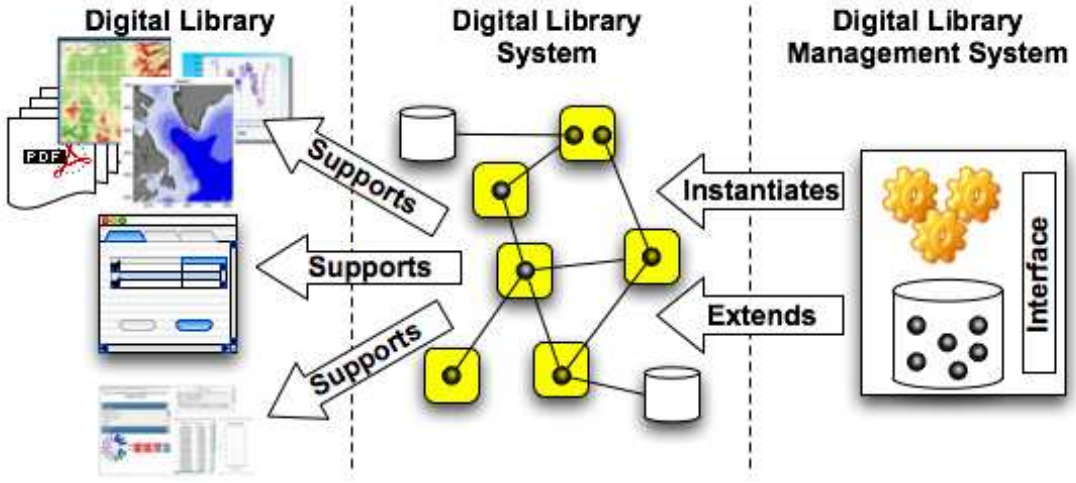


Figure I.2-1. DL, DLS and DLMS: A Three-tier Framework

These three system notions are often confused and are used interchangeably in the literature; this terminological imprecision has produced a plethora of heterogeneous entities and contributes to making the description, understanding and development of digital library systems difficult. As Figure I.2-1 indicates, all three systems play a central and distinct role in the Digital Library development process. To clarify their differences and their individual characteristics, the explicit definitions that follow may help:

Digital Library (DL)
An organisation, which might be virtual, that comprehensively collects, manages and preserves for the long term rich ***digital content,*** *and offers to its* ***user*** *communities specialised* ***functionality*** *on that content, of measurable* ***quality*** *and according to codified* ***policies****.*

Digital Library System (DLS)
A software system that is based on a defined (possibly distributed) ***architecture*** *and provides all functionality required by a particular Digital Library. Users interact with a Digital Library through the corresponding Digital Library System.*

Digital Library Management System (DLMS)
A generic software system that provides the appropriate software infrastructure both (i) to produce and administer a Digital Library System incorporating the suite of functionality considered fundamental for Digital Libraries and (ii) to integrate additional software offering more refined, specialised or advanced functionality.

A Digital Library Management System belongs to the class of 'system software'. As is the case in other related domains, such as operating systems, databases and user interfaces, DLMS software generation environments may provide mechanisms to be used as a platform

to produce Digital Library Systems. Depending on the philosophy it follows, a DLMS may belong to one of the following three types:

- **Extensible Digital Library System**

  A complete Digital Library System that is fully operational with respect to a defined core suite of functionality. DLs are constructed by instantiating the DLMS and thus obtaining the DLS. Thanks to the open software architecture, new software components providing additional capabilities can be easily integrated. The DelosDLMS [184][3] is a prototypical example of a system based on this philosophy.

- **Digital Library System Warehouse**

  A collection of software components that encapsulate the core suite of DL functionality and a set of tools that can be used to combine these components in a variety of ways (in Lego®-like fashion) to create Digital Library Systems offering a tailored integration of functionalities. New software components can be easily incorporated into the Warehouse for subsequent combination with those already there. BRICKS [41] and DILIGENT [68] are two prototypical examples of systems that are based on this philosophy.

- **Digital Library System Generator**

  A highly parameterised software system that encapsulates templates covering a broad range of functionalities, including a defined core suite of DL functionality as well as any advanced functionality that has been deemed appropriate to meet the needs of the specific application domain. Through an initialisation session, the appropriate parameters are set and configured; at the end of that session, an application is automatically generated, and this constitutes the Digital Library System ready for installation and deployment. The MARIAN framework equipped with the 5SL specification language represents an example of this process [96].

Although the concept of Digital Library is intended to capture an abstract system that consists of both physical and virtual components, the Digital Library System and the Digital Library Management System capture concrete software systems. For every Digital Library, there is a unique Digital Library System in operation (possibly consisting of many interconnected smaller Digital Library Systems), whereas all Digital Library Systems are based on a handful of Digital Library Management Systems.[5] For instance, through DILIGENT it is possible to build and run a number of DLSs, each realising a DL serving a target community. The DL is thus the abstract entity that 'lives' thanks to the software system constituting the DLS.

[5] To the extent that it is helpful, it is possible to draw an approximate analogy between the world of Digital Libraries and the world of Databases. A DBMS (e.g. the DB2, Oracle system, MySQL or PostgreSQL) corresponds to a DLMS, offering general data management services. A DBMS together with all application software running on top of it at an installation corresponds to a DLS. Finally, a DL corresponds to a so-called 'Information System', which consists of the above software, its data and its users.

I.3 The Digital Library Universe: Main Concepts

Despite the great variety and diversity of existing digital libraries,[6] in reality only a limited range of concepts are defined by all systems as core functionalities. These concepts are identifiable in nearly every Digital Library currently in use. They serve as a starting point for any researcher who wants to study and understand the field, for any system designer and developer intending to construct a Digital Library, and for any content provider seeking to expose its content via digital library technologies. In this section, we identify these concepts and briefly discuss them.

Six core concepts provide a foundation for Digital Libraries. Five of them appear in the definition of Digital Library: *Content*, *User*, *Functionality*, *Quality* and *Policy*; the sixth one emerges in the definition of Digital Library System: *Architecture*. All six concepts influence the Digital Library framework, as shown in Figure I.3-1.

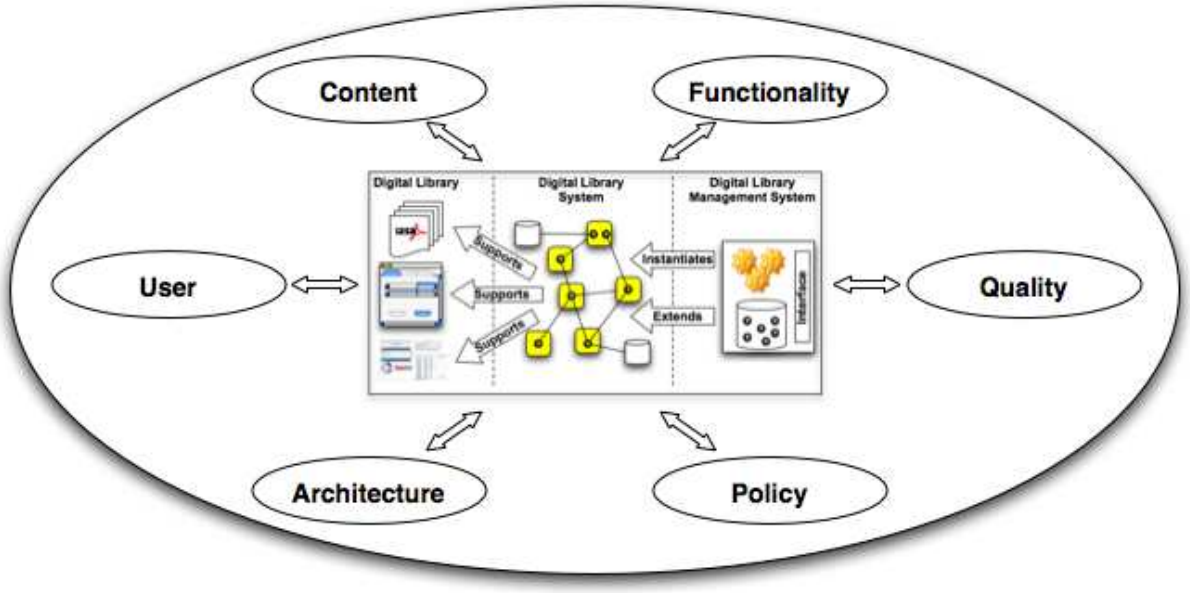


Figure I.3-1. The Digital Library Universe: Main Concepts

I.3.1 Content

The *Content* concept encompasses the data and information that the Digital Library handles and makes available to its users. It is composed of a set of information objects organised in collections. Content is an umbrella concept used to aggregate all forms of information objects that a Digital Library collects, manages and delivers. It encompasses the diverse range of information objects, including such resources as objects, annotations and metadata. For example, metadata have a central role in the handling and use of information objects, as they provide information critical to its syntactical, semantic and contextual interpretation.

I.3.2 User

The *User* concept covers the various actors (whether human or machine) entitled to interact with Digital Libraries. Digital Libraries connect actors with information and support them in

[6] From here on, we shall use the terms Digital Library (or its abbreviation DL), Digital Library System (DLS) and Digital Library Management System (DLMS) to denote the systems identified in Section I.2, while by the term 'digital libraries' we shall refer to the whole field of digital library research and applications.

their ability to consume and make creative use of it to generate new information. User is an umbrella concept including all notions related to the representation and management of actor entities within a Digital Library. It encompasses such elements as the rights that actors have within the system and the profiles of the actors with characteristics that personalise the system's behaviour or represent these actors in collaborations.

I.3.3 Functionality

The *Functionality* concept encapsulates the services that a Digital Library offers to its different users, whether classes of users or individual users. While the general expectation is that DLs will be rich in capabilities and services, the bare minimum of functions would include such aspects as new information object registration, search and browse. Beyond that, the system seeks to manage the functions of the Digital Library to ensure that the functions reflect the particular needs of the Digital Library's community of users and/or the specific requirements relating to the Content it contains.

I.3.4 Quality

The *Quality* concept represents the parameters that can be used to characterise and evaluate the content and behaviour of a Digital Library. Quality can be associated not only with each class of content or functionality but also with specific information objects or services. Some of these parameters are objective in nature and can be measured automatically, whereas others are subjective in nature and can only be measured through user evaluations (e.g. focus groups).

I.3.5 Policy

The *Policy* concept represents the set or sets of conditions, rules, terms and regulations governing interaction between the Digital Library and users, whether virtual or real. Examples of policies include acceptable user behaviour, digital rights management, privacy and confidentiality, charges to users, and collection delivery. Policies belong to different classes; for instance, not all policies are defined within the DL or the organisation managing it. The policy supports the distinction between extrinsic and intrinsic policies. The definition of new policies and re-definition of older policies will be a feature of digital libraries.

I.3.6 Architecture

The *Architecture* concept refers to the Digital Library System entity and represents a mapping of the functionality and content offered by a Digital Library on to hardware and software components.[7] There are two primary reasons for having Architecture as a core concept: (i) Digital Libraries are often assumed to be among the most complex and advanced forms of information systems [84]; and (ii) interoperability across Digital Libraries is recognised as a substantial research challenge. A clear architectural framework for the Digital Library System offers ammunition in addressing both of these issues effectively.

The concepts populating all the six areas just introduced share many similar characteristics and are all concepts referring to internal entities of a Digital Library that can be sensed by the external world. Introducing a higher-level concept referring to any of these, namely, *Resource*, enables us to reason about these characteristics in a consistent manner.

---

[7] This is an appropriate adaptation of the 'Architecture' definition from the Glossary of CMU's Software Engineering Institute. http://www.sei.cmu.edu/opensystems/glossary.html

Figure I.3-2 puts in perspective the six main concepts of the Digital Library world. Among these, three are independent, i.e. their existence does not depend on the existence of a digital library. These are *Architecture*, representing the technological design on which the Digital Library System is based, *User*, representing the external humans or hardware interacting with the Digital Library, and *Content*, representing the material handled by the Digital Library. On top of these comes *Functionality*, representing primarily the means for connecting *User* to *Content*, i.e. all procedures, transformations, actions and interactions that bring *Content* to *User* or vice versa. Finally, operation of the Digital Library and activation of its *Functionality* are based on *Policy* and aim to achieve certain *Quality*.

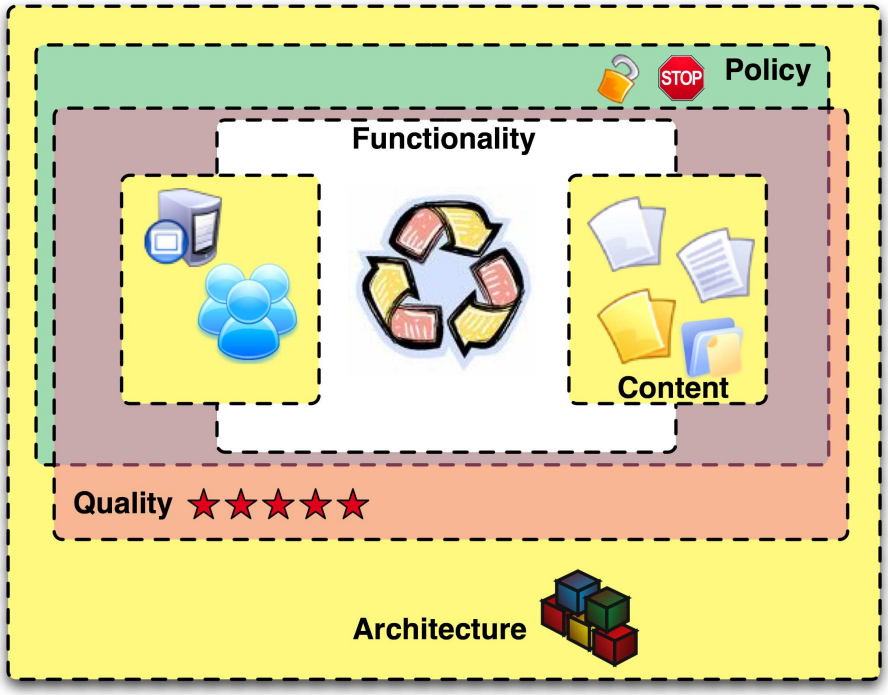


Figure I.3-2. The Digital Library Universe: The Main Concepts in Perspective

The six core concepts (Content, User, Functionality, Quality, Policy and Architecture) that lie at the heart of the Digital Library universe need to be considered in conjunction with the four main ways in which actors interact with digital library systems, as discussed in the next section.

I.4 The Digital Library Universe: The Main Roles of Actors

We envisage actors interacting with digital library systems in four different and complementary ways: *DL End-users*, *DL Designers*, *DL System Administrators* and *DL Application Developers*.

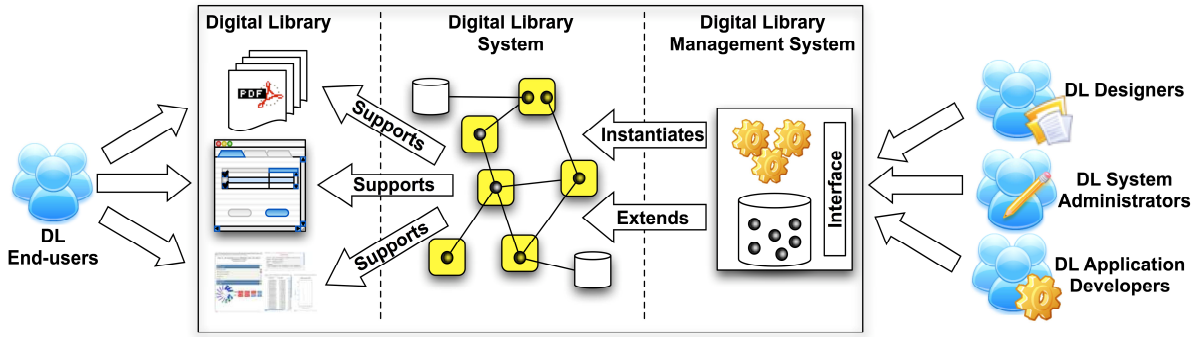


Figure I.4-1. The Main Roles of Actors versus the Three-tier Framework

As shown in Figure I.4-1, each role is primarily associated with one of the three 'systems' in the three-tier framework.

I.4.1 DL End-users

DL End-users exploit the DL functionality for the purpose of providing, consuming and managing the DL Content and some of its other constituents. They perceive the DL as a stateful entity serving their functional needs. The behaviour and output of the DL depend on the DL's state at the time a particular part of its functionality is activated. The state of the DL corresponds to the state of its *resources*, which, as we have seen above, consist of the collections of information objects managed by the DL, the set of authorised users, the DL's functionality and its set of policies. This state changes during the lifetime of the Digital Library according to the functionality activated by users and their inputs. DL End-users may be further divided into *Content Creators*, *Content Consumers* and *Librarians*.

Content Creators are the producers of the DL Content; they feed it with the resources, mainly information objects, to which other users of the DL will have access. This activity is (*i*) accomplished through the Functionality the DL provides, (*ii*) regulated by the Policies defined in the DL, and (*iii*) performed according to the Quality the DL must guarantee.

Content Consumers are the purchasers of the DL Content; in reality, these users consume all the resources a DL makes available. In fact, they access Content: (*i*) through the Functionality the DL provides, (*ii*) in accordance with the Policies defined in the DL, and (*iii*) with the guarantee of Quality the DL declares.

Librarians are End-users in charge of curating the DL Content. In fact, these actors have to curate all the resources forming the DL, e.g. establish the Policies. Section I.4.5 elaborates on this role by explaining how this model captures the various activities modern Librarians have to deal with.

I.4.2 DL Designers

DL Designers exploit their knowledge of the application semantic domain in order to define, customise and maintain the Digital Library so that it is aligned with the information and functional needs of its potential DL End-users. To perform this task, the DL Designers interact with the DLMS providing functional and content configuration parameters. Functional parameters instantiate aspects of the DL functionality that are to be perceived by

the DL End-users, including the characteristics of the result set format, query language(s), user profile formats, and document/data model employed. Content configuration parameters specify third-party resources exploited by the specific DL, e.g. repositories of content, ontologies, classification schemas, authority files, and gazetteers that will be used to form the DL Content. The values of these parameters configure the way the DL will be presented to the DL End-users, as they determine the particular Digital Library System instance serving the Digital Library. Of course, these parameters need not necessarily be fixed for the entire lifetime of the DL; they may be reconfigured to enable the DL to respond to the evolving expectations of users and changes in all aspects from policies to content.

I.4.3 DL System Administrators

DL System Administrators select the software components needed to construct the Digital Library System. Their choice of elements reflects the expectations that DL End-users and DL Designers have for the Digital Library, as well as the requirements the available resources impose on the definition of the DL. DL System Administrators interact with the DLMS by providing architectural configuration parameters, such as the chosen software components and the selected hosting nodes. Their task is to identify the architectural configuration that best fits the DLS in order to ensure the highest level of quality of service. The value of the architectural configuration parameters can be changed over the DL lifetime. Changes of parameter configuration may result in the provision of different DL functionality and/or different levels of quality of service.

I.4.4 DL Application Developers

DL Application Developers develop the software components that will be used as constituents of the DLSs, to ensure that the appropriate levels and types of functionality are available.

I.4.5 Where are the Librarians?

The reader may be surprised that this Manifesto purports to cover the Digital Library world but none of the above-envisaged classes of actors is termed ‘Librarian’. In fact, a kind of End-user was termed as *Librarian* but this captures only one particular facet of Librarians playing a fundamental role in the Digital Library universe. Today, *Librarians* are a kind of actor spanning many of the envisaged roles, as demonstrated by the description provided.

Because the DL End-user executes functionality for providing, consuming and managing the DL content, the model includes in this category the ‘End-user Librarian’, i.e. Librarians acting as cataloguers and curators in the Library world and those interfacing with and supporting the users of a Library. End-user Librarians are the front end to Library clients; as the Digital Library world has no physical place that represents the DL, these actors interact with the other users via the ‘system’.

Because the DL Designer exploits her/his knowledge of the application semantic domain to define, customise and maintain the Digital Library, it is paired with the ‘Digital Librarian’, i.e. the chief librarian who decides the policies regulating the Library.

Finally, the DL System Administrator is paired with the ‘System Librarian’, i.e. the Librarian with technical skills entitling her/him to manage the DL software system.

Thus, even if none of the actors is termed ‘Librarian’, the Manifesto is capable of representing the various ‘incarnations’ a *Librarian* can assume.

The four roles described above encompass the entire spectrum of actors interacting with digital libraries. Their models of the DL Universe are linked together in a hierarchical fashion,

as shown in Figure I.4-2. This hierarchy is a direct consequence of the above definitions, since DL End-users act on the Digital Library, whereas DL Designers, DL System Administrators and DL Application Developers operate on the DLS (through the mediation of a DLMS) and, consequently, on the DL as well. This inclusion relationship ensures that cooperating actors share a common vocabulary and knowledge. For instance, the DL End-user expresses requirements in terms of the DL model and, subsequently, the DL Designer understands these requirements and defines the DL accordingly.

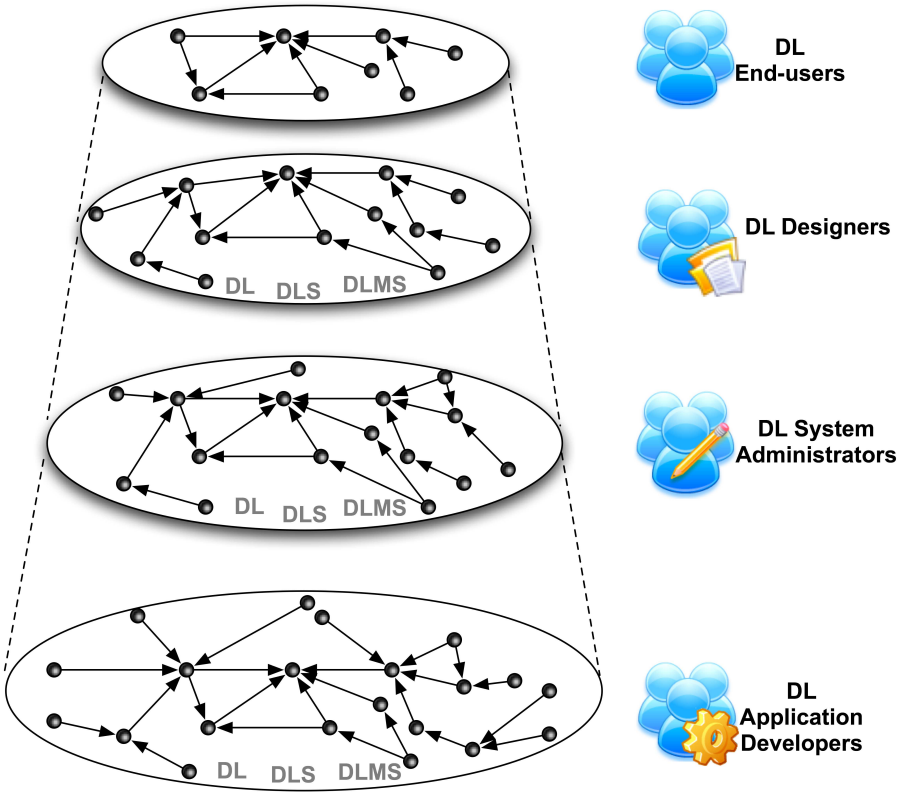


Figure I.4-2. Hierarchy of Users' Views

I.5 Digital Library Development Framework

An explained earlier, the Digital Library universe is a complex world. Consequently, it is difficult to identify a single and fully-fledged model capable of capturing all the aspects needed to represent this universe in all the necessary scenarios. One of the scenarios in which such modelling activity is particularly important is the pattern leading to the development of concrete systems. This scenario is very broad, as being capable of capturing the peculiarities of an entity at a level of detail that allow developers to implement such an entity requires the capability to capture a comprehensive set of aspects that characterise the entity and thus can be reused in a plethora of other application domains, e.g. teaching, comparing existing systems. However, such a model may be difficult to use if it is not appropriately designed, i.e. tailored to address the specific needs of the audience for which it is designed. For this reason, we structured the model needed to capture the Digital Library universe and enabled it to implement its constituents in multiple elements (see Figure I.5-1) which can be better represented in detail by introducing frameworks supporting different levels of abstraction.

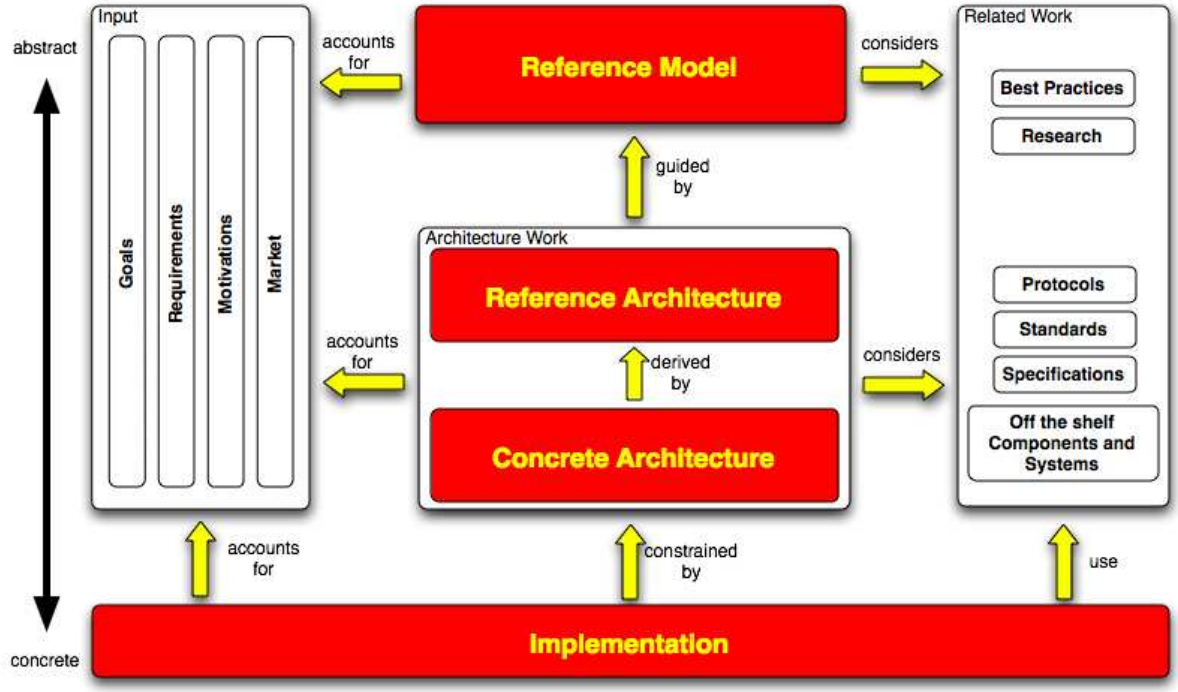


Figure I.5-1. The Digital Library Development Framework[8]

More specifically, the elements constituting the Development Framework are:

- *Reference Model* – As stated in [156], 'A Reference Model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details'. Digital libraries need a corresponding Reference Model in order to consolidate the diversity of existing approaches into a cohesive and consistent whole, to offer a mechanism for enabling the comparison of different Digital Library systems, to provide a common basis for communication within the Digital Library community, and to help focus further advancement.

[8] This diagram was inspired by the 'Reference Model for Service Oriented Architecture' document [156].

- *Reference Architecture* – The Reference Architecture is an architectural design pattern indicating an abstract solution that implements the concepts and relationships identified in the Reference Model. There may be more than one Reference Architecture that addresses how to design digital library systems built on the Reference Model. For example, we might have one Reference Architecture for DLSs supporting DLs constructed by federating local resources and multiple organisations, and another one for personal DLs or for specialised applications.
- *Concrete Architecture* – At this level, the Reference Architecture is realised by replacing the mechanisms envisaged in the Reference Architecture with concrete standards and specifications. For example, a Concrete Architecture may specify that the run-time environment deployed on the hosting nodes will be CORBA or the Web Services Application Framework, and that a number of specific communicating Web Services will implement the Search functional component.

The relationship of these three frameworks with the general digital library environment is shown in Figure I.5-1. At the top there is the most abstract Reference Model, which guides the more specific Reference Architecture and Concrete Architecture further down. In turn, these should constrain the development and implementation of any actual system. The three reference frameworks are the outcome of an abstraction process that has taken into account the goals, requirements, motivations and, in general, the digital library market, as shown on the left-hand side of Figure I.5-1, and the best practices and relevant research shown on the right-hand side of the same figure. When these frameworks are adopted and followed by the community, the resulting systems will be largely compatible with each other; the interoperability thus afforded will open up significant new horizons for the field.

The rest of this volume focuses on the Reference Model part of this framework.

I.6 Digital Library Manifesto: Concluding Remarks

The goal of *The Digital Library Manifesto* has been to set the foundations and identify the entities of discourse within the universe of digital libraries. It has introduced the relationships three kinds of relevant 'systems' in this area: Digital Library, Digital Library System, and Digital Library Management System. It has presented the main concepts characterising the above, i.e. content, user, functionality, quality, policy and architecture, and has identified the main roles that actors may play within a digital library, i.e. end-user, designer, administrator and application developer. Finally, it has described the reference frameworks that are needed to clarify the Digital Library universe at different detailed levels of abstraction, i.e. the Reference Model and the Reference and Concrete Architectures.

The Digital Library Manifesto is currently accompanied by two other documents, which provide, respectively, a high-level overview and a more detailed definition of the concepts and relationships required to capture the complex Digital Library universe. These documents are an attempt to fulfil the fundamental needs of the Digital Library field. Clearly, the diversity of needs among different digital libraries will continue to introduce new concepts that will have to be incorporated into the model. Hence, these documents should be considered as first versions of dynamic documents that will continue to evolve, having the Manifesto as a firm foundation.

The *Digital Library Manifesto* has been based on the experience and knowledge gained by many previous efforts that have taken place over the past fifteen years around Europe and the rest of the world. We hope it will serve as a basis for new advances in research and system development in the future.

PART II The DELOS Digital Library Reference Model in a Nutshell

II.1 Introduction

Despite the large number of 'systems' that are called 'digital libraries' [36][120][121][140][83][84] (where 'system' is intended as a set of interconnected things forming a whole), as yet there are no real underlying foundations for them. This limits the growth of the digital library field, as is the case for any building for which no appropriate foundation has been provided. Because of this lack, it is really difficult and indeed almost impossible to systematise activities for evaluating and comparing digital library systems, and for teaching and even performing further and focused research. The same holds true for system design and development, for promoting sustainable approaches and solutions that aim at maximising the reuse of existing knowledge and assets, and at properly addressing community needs.

In January 2005, the DELOS Network of Excellence on Digital Libraries [66] decided to initiate the definition of a *reference model* for digital libraries as a necessary step towards a more systematic approach to the research on digital libraries. In this context, by reference model we mean an abstract framework for understanding significant relationships between the entities of some universe, and for the development of consistent standards and/or specifications supporting that universe [156]. The route towards reaching this objective, summarised below, has been traced in the Manifesto (Part I of this volume).

II.1.1 The Digital Library Manifesto in Brief

It is commonly understood that the Digital Library universe is a complex and multifaceted domain that cannot be captured by a single definition. The Manifesto organises the pieces constituting the puzzle into a single framework (Figure II.1-1).

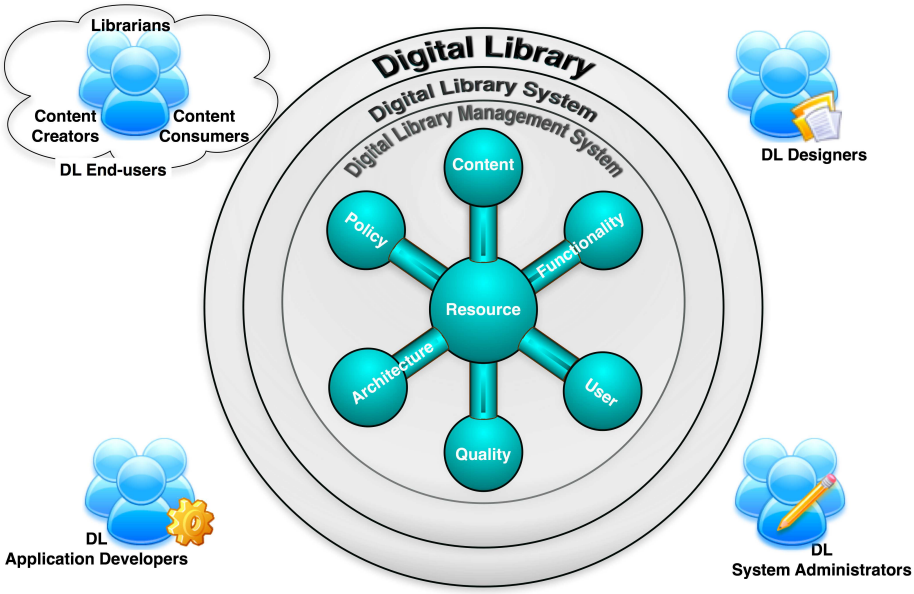


Figure II.1-1. The Digital Library Universe

In particular, it identifies the ***three different types of systems*** operating in the Digital Library universe, i.e.

(1) the *Digital Library* (*DL*) – the final 'system' actually perceived by the end-users as being the digital library;

(2) the *Digital Library System* (*DLS*) – the deployed and running software system that implements the DL facilities; and

(3) the *Digital Library Management System* (*DLMS*) – the generic software system that supports the production and administration of DLSs and the integration of additional software offering more refined, specialised or advanced facilities.

The *Manifesto* also organises the Digital Library universe into ***domains***.

(1) The *Resource Domain* captures generic characteristics that are common to the other specialised domains.

Building on this, the model introduces ***six*** orthogonal and complementary *domains* that together strongly characterise the Digital Library universe and capture its specificities with respect to generic information systems. These specialised domains are:

(2) *Content* – represents the information made available;

(3) *User* – represents the actors interacting with the system;

(4) *Functionality* – represents the facilities supported;

(5) *Policy* – represents the rules and conditions, including digital rights, governing the operation;

(6) *Quality* – represents the aspects needed to consider digital library systems from a quality point of view;

(7) *Architecture* – represents the physical software (and hardware) constituents concretely realising the whole.

Another contribution of the *Manifesto* is recognising the existence of ***various players*** acting in the DL universe and cooperating in the operation of the whole. In particular, [9]

- The *DL End-Users* are the ultimate clients the Digital Library is going to serve.
- The *DL Designers* are the organisers and orchestrators of the Digital Library from the application point of view.
- The *DL System Administrators* are the organisers and orchestrators from the physical point of view.
- The *DL Application Developers* are the implementers of the software parts needed to realise the Digital Library.

Further, it states that there is the need for modelling ***focused views***. The ultimate goal of the whole reference model activity is to clarify the Digital Library universe to the different actors by tailoring the representation to their specific needs. The three systems organise the universe in concentric layers that are revealed to interested players only. Meanwhile, the six domains constitute the complementary perspectives from which interested players are allowed to see each layer. Thus, the framework is potentially complex because it aims at accommodating all the various needs. However, it is highly modular and can therefore be easily adapted to capture the needs arising in specific application contexts.

Finally, the *Manifesto* gives reason for proceeding with ***different levels of abstraction*** while laying down the complete framework. These different levels of abstraction, which lead conceptually from the modelling to the implementation, are captured in Figure II.1-2 where the core role of the *Reference Model* is illustrated; all the other elements constituting the envisaged DL development methodology chain start from here. It drives the definition of any *Reference Architecture* that proposes an optimal architectural pattern for a specific class of

[9] It is still under discussion whether two other players should be added to this list, namely Institutions and Industries. By *Institutions* are meant organisations, either concrete or virtual, having the important role of forming the Digital Library. By *Industries* are meant the institutions performing economic activities concerned with the Digital Library, by providing either the software or the service.

digital library systems characterised by similar goals, motivations and requirements. *Concrete Architectures* are obtained by replacing the mechanisms envisaged in the Reference Architectures with concrete standards and specifications. Finally, *Implementations*, i.e. the concrete realisation of the DLS supporting a particular DL, are instances of Concrete Architectures deployed on particular machines. The definition of the DELOS Reference Model has thus also to be seen as a necessary starting point towards the introduction of all these other framework elements, which, once adopted and followed by the community, will largely enhance the DL development model and the interoperability among systems.

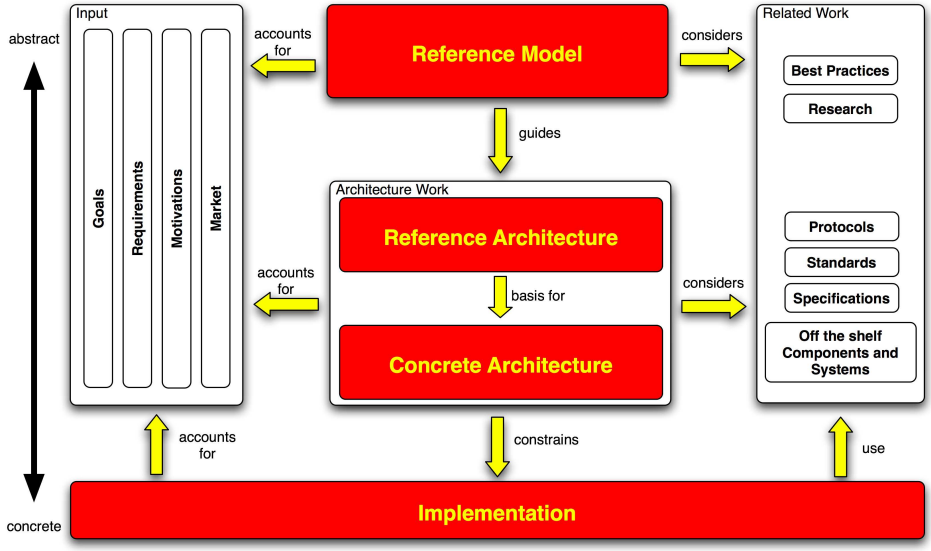


Figure II.1-2. The Reference Model as the Core of the Development Framework

The rest of Part II of this volume provides an overview of the DELOS Reference Model by illustrating the constituent concepts and relationships. It is structured as follows. The present section is completed by information that sets the stage for the rest, e.g. the background material necessary to understand the rest, graphical and notational conventions. Section II.2 introduces the constituent domains of the model, highlighting the main concepts and relationships characterising the domain model rationale. Section II.3 discusses possible exploitation of such a model; in particular it discusses some preliminary uses of the model with respect to (*i*) the interoperability issue by presenting the main concepts and relations related to it, and (*ii*) the preservation issue by presenting the concepts and relations concerning it. Section II.4 briefly investigates related work on models for digital libraries and domains. Finally, Section II.5 provides concluding remarks.

II.1.2 Guide to using the Reference Model

A Reference Model is a conceptual framework that aims at capturing significant entities and their relationships in a certain universe with the goal of developing more concrete models of it. 'Enterprise Architecture' frameworks play a similar role. The aim of the Enterprise Architecture practice is to model the relationships between the business and the technology in such a way that this information can be used to support decisions enterprise wide, e.g. revising the business processes or changing the software systems supporting certain processes. Thus, the Enterprise Architecture must be compared with the whole Reference Model activity while considering the Enterprise Architecture as a decision support process that needs a model capturing a large amount of information. Because of the breadth of information to be covered, both models recognise the need to have a means of categorising

this information. The best-known Enterprise Architecture framework was devised by Zachman [216]. This framework defines:

(1) different descriptions of the same product (similar to our domains), i.e. *Data* (the *what*), *Process* (the *how*), *Network* (the *where*), *People* (the *who*), *Time* (the *when*) and *Motivation* (the *why*), and

(2) different views in order to serve the needs of various stakeholders, i.e. scope description (planner's view), business model (owner's view), system model (designer's view), technology model (builder's view), detailed description (implementer's view), actual system (worker's view).

This Reference Model is founded on very similar principles, although tailored to address the specificities of the Digital Library universe.

Having clarified this, it is important to note that a plethora of modelling languages exists. These range from the human language to formal languages borrowed from various application domains and characterised by various types of expressing power and other interesting features, such as Entity-Relationship [55], UML [33] and Description Logic [17], to cite just a few. However, in this document concept maps have been used because of their simplicity and immediacy.

II.1.2.1 Concept Maps

Concept maps are graphical tools for organising and representing knowledge [167][168] in terms of concepts (entities) and relationships between concepts to form propositions. Concepts are used to represent regularity in events or objects, or records of events or objects. Propositions are statements about some objects or events in the universe, either naturally occurring or constructed. Propositions contain two or more concepts connected using linking words or phrases to form a meaningful statement. In the graphical representation, concepts are inscribed in circles or boxes, while propositions (proposition connectors) are represented as (directed) lines connecting concepts, labelled with words describing the linking relationship. Figure II.1-3 gives an example of a concept map, showing the structure of concept maps and illustrating their main characteristics.

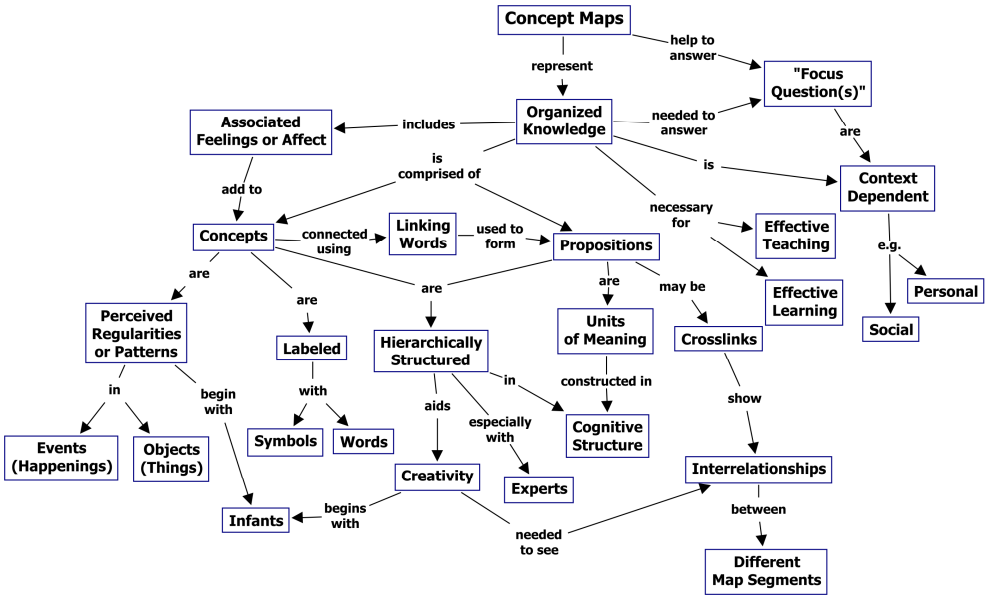


Figure II.1-3. A Concept Map showing the Key Features of Concept Maps[10]

II.1.2.2 Notational Conventions

In the following, terms expressing *concepts*, i.e. constituent entities of the Reference Model, are typed in **bold** at their first occurrence in the document and in *italic* in the rest of the document. Terms expressing *relations* in the Reference Model are typed in <*italic with angle brackets*> when they occur in the document.

[10] Figure taken from [168].

II.2 The Constituent Domains

As outlined in the previous section, the Digital Library universe is complex and multifaceted. Figure II.2-1 presents an organisation of the entities of this universe into a hierarchy of *domains*, i.e. named groups of concepts and relations, each modelling a certain aspect of the systems of the universe. In this context, domains play a role similar to that of UML packages and XML namespaces in their respective application areas. Domains may rely on each other and constitute orthogonal areas intended to capture the different aspects of the whole.

Each of the DL 'systems' is modelled by entities and relationships captured by these domains at different levels of abstraction.

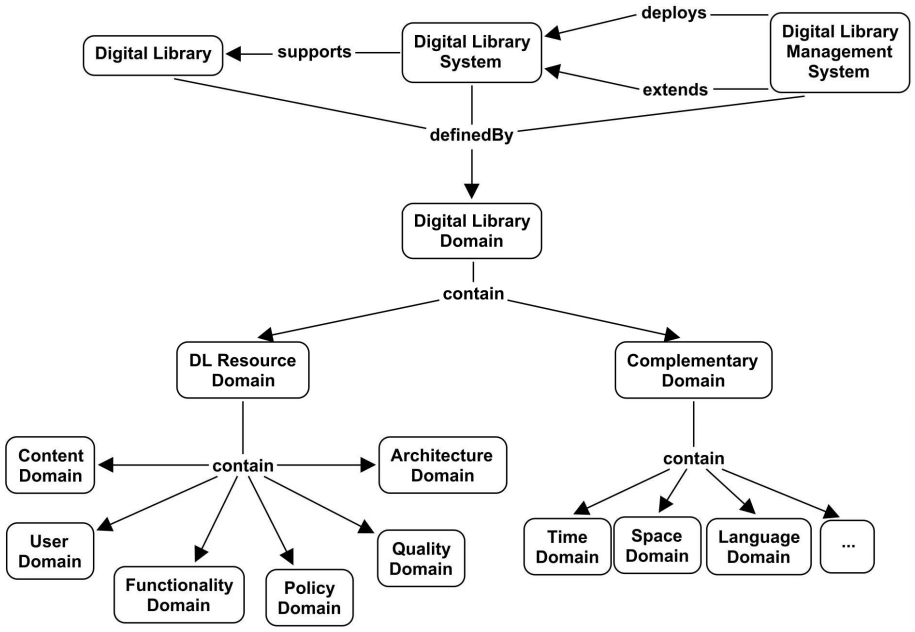


Figure II.2-1. DL Domains Hierarchy Concept Map

The ***Digital Library Domain***, which comprises all the elements needed to represent the three systems of the DL universe, is divided into two main classes: ***DL Resource Domain*** and ***Complementary Domain***.

The *DL Resource Domain*, described in Section II.2.1, contains elements identified as 'first class citizens' in modelling the Digital Library universe. It is further classified into:

(1) ***Content Domain*** (cf. Section II.2.2);

(2) ***User Domain*** (cf. Section II.2.3);

(3) ***Functionality Domain*** (cf. Section II.2.4);

(4) ***Policy Domain*** (cf. Section II.2.5);

(5) ***Quality Domain*** (cf. Section II.2.6); and

(6) ***Architecture Domain*** (cf. Section II.2.7),

each of which focuses on a particular aspect of the DL systems.

The *Complementary Domain* contains all the other domains, which, although they do not constitute the focus of the digital libraries and can be inherited from existing models, are nevertheless needed to represent the systems. This concept serves as a placeholder for domains different from those identified as 'first class citizens' and as a hook for future extensions of the model. It includes concepts such as:

- ***Time Domain*** (i.e. concepts and relations needed to capture aspects of the time sphere such as time periods and intervals);
- ***Space Domain*** (i.e. concepts and relations needed to capture aspects of the physical sphere such as regions and locations);
- ***Language Domain*** (i.e. concepts and relations needed to capture aspects of the method of communication, either spoken or written, consisting of the use of words in a structured and conventional way).

The rest of Part II of this volume illustrates the different domains listed above by providing an overview of their concepts and relationships. In approaching models like the one we are presenting here, it is important to keep in mind that these models are not intended to be 'complete' or exhaustive, i.e. capable of representing all the possible facets of the systems in the DL universe, but rather as cores of a model of such a Universe that can be extended by specific communities to include the elements required to capture their specific needs.

II.2.1 DL Resource Domain

The *DL Resource Domain*, being the highest-level domain in our framework, represents all entities and relationships that are managed in the Digital Library universe. The most general concept of the *DL Resource Domain* is ***Resource***, which includes any Digital Library entity (Figure II.2-2). The notion of resource as a primitive concept in a domain is not new. In the context of the Web, for example, resource is the primitive notion of the whole architecture [123]. The Web resource notion has evolved during the Web's history from the early conception of document or file to the current abstract definition that covers any entity that can be identified, named or addressed in the Web. This novel understanding fits very well with the meaning assumed by the same term in the Digital Library universe.

Instances of the concept of *Resource* in the Digital Library universe are *Information Objects* in all their forms (e.g. documents, images, videos, multimedia compound objects, annotations and metadata packets, streams, databases, collections, queries and their result sets), *Actors (*both humans and inanimate entities), *Functions*, *Policies*, *Quality Parameters* and *Architectural Components*. Each of these instances represents the main concept in their respective domain, thus every *Domain* consists of *Resources*, and *Resources* are the building blocks of all the *Digital Library Domains.*

All the different types of *Resources* share many characteristics and ways in which they can be related to other Resources (Figure II.2-2). Each *Resource* is:

(1) identified by a ***Resource Identifier*** (<*identifiedBy*>);

(2) arranged or set out according to a ***Resource Format*** (<*hasFormat*>) – such a format may be drawn from an Ontology in order to guarantee a uniform interpretation and be arbitrarily complex and structured because *Resources* may be in turn composed of smaller *Resources* (<*hasPart*>) and linked to other *Resources* (<*associatedWith*>) so as to form compound artefacts;

(3) characterised by various *Quality Parameters,* each capturing how the resource performs with respect to some attribute (<*hasQuality*>);

(4) regulated by *Policies* (<*regulatedBy*>) governing every aspect of its lifetime;

(5) expressed by (<*expressedBy*>) an *Information Object* (such as a *Policy* set down in a text or a flowchart); and

(6) described by or commented on by an *Information Object*, especially by *Metadata* (<*hasMetadata*>) and *Annotations* (<*hasAnnotation*>).

From an organisational point of view, *Resources* can be grouped in ***Resource Sets*** (<*belongTo*>), i.e. groups of *Resources* to be considered as a single entity for certain management or application purposes. Examples of a *Resource Set* in the various domains are *Collection* in the *Content Domain* or *Group* in the *User Domain*.

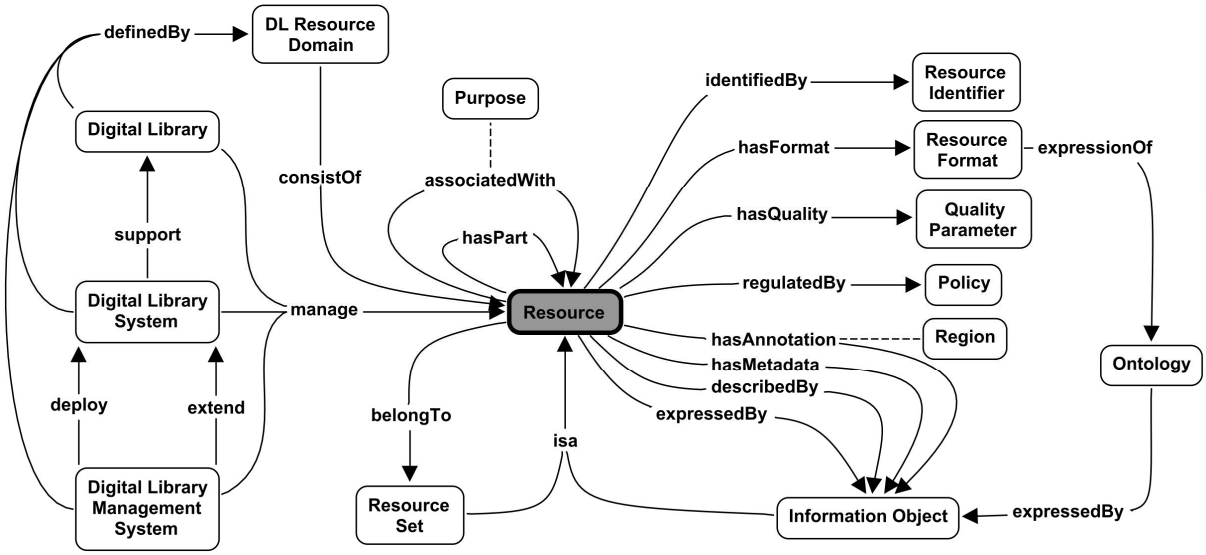


Figure II.2-2. DL Resource Domain Concept Map[11]

Modelling the characteristics shared by all the DL entities at a high level of abstraction and representing more specific entity types by inheriting the shared characteristics leads to an elegant and concise model, to efficient implementations, and to uniform user interfaces. The advantages of this modelling approach can be transformed into innovative system features and implementations. For example, unified mechanisms for handling relations and functions that apply to all resource types and unified search facilities for seamlessly discovering the various entities available in a DL can be envisaged.

II.2.2 Content Domain

The *Content Domain* represents all the entities related to the information that Digital Library 'systems' manage in order to satisfy the information needs of its users. The most general concept characterising the *Content Domain* is ***Information Object*** (Figure II.2-3), which is a *Resource.* An *Information Object* represents any unit of information managed in the Digital Library universe and includes text documents, images, sound documents, multimedia documents and 3-D objects, including games and virtual reality documents, as well as data sets and databases. *Information Object* also includes composite objects and *Collections* of *Information Objects*. Types of *Information Objects* can furthermore be distinguished by their nature along the following dimensions:

(1) By the type of data, information or knowledge contained in the *Information Object*, namely:

a. *Information Objects* containing raw data captured directly from the outside world (especially data or data streams captured by instruments). Such raw data often require metadata for proper processing and interpretation.

[11] A Concept linked to a Relation by a dotted line represents an attribute of the Relation itself.

b. *Information Objects* that contain data processed through or generated by the mind or some other system, with the result often called information (as opposed to raw data) or knowledge.

(2) By the type of information representation or encoding, namely:

a. *Information Objects* in which information/knowledge is encoded in natural language and embodied in a document. In a wider sense this also includes pictorial or sound representations.

b. *Information Objects* in which information/knowledge is encoded in a formal structure, such as database tables or formal entity-relationship statements. An ontology represented in format terms would fall here.

(3) By state of digital representation, namely:

a. Born digital information object, such as a born digital text or a digital camera image.

b. A digital information object produced by digitisation of a non-digital information object.

c. A non-digital information object that may be represented in the digital library by a metadata record.

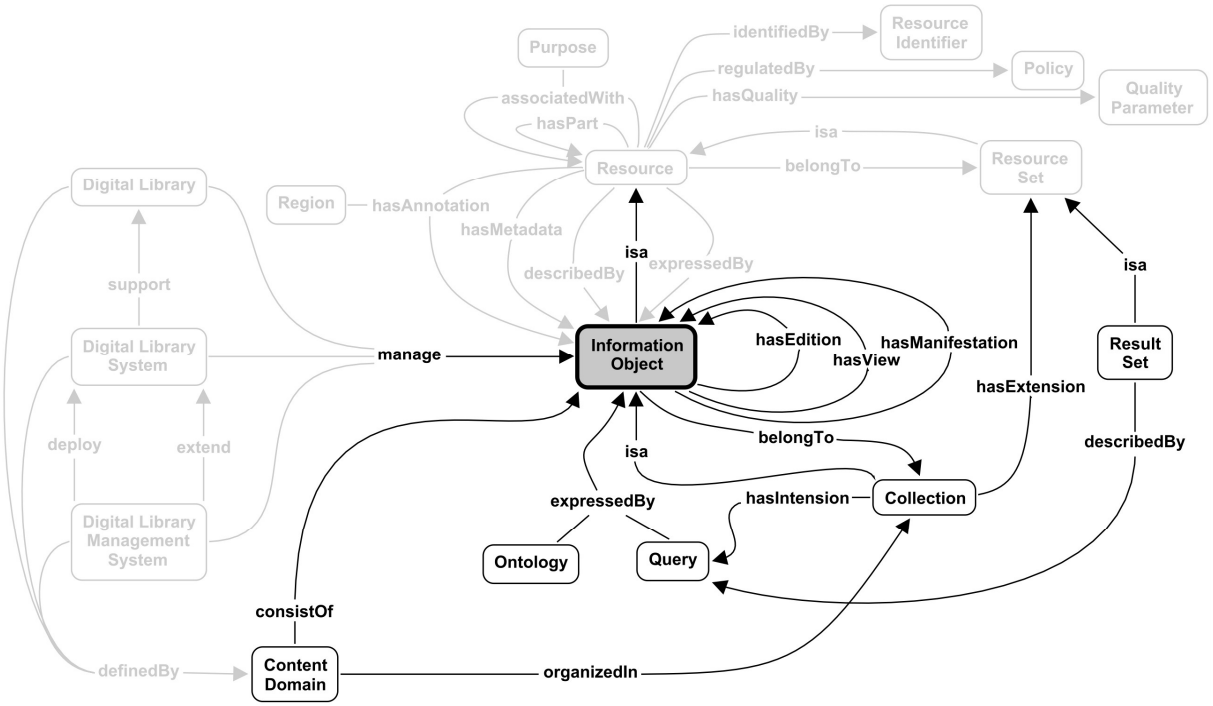


Figure II.2-3. Content Domain Concept Map

As an *Information Object* is a *Resource*, it inherits all its features; namely it

(1) has a unique identifier (*Resource Identifier*) also known as the information object identifier;

(2) is arranged according to a format (*Resource Format*) also known as the document model;

(3) can arbitrarily be composed (<*hasPart*> and <*associatedWith*>) to capture compound artefacts;

(4) is characterised by various *Quality Parameters* each capturing different object quality facets (<*hasQuality*>);

(5) is regulated by *Policies* (<*regulatedBy*>) governing every aspect of its lifetime; and

(6) can be described or augmented by ***Metadata*** (<*hasMetadata*>) and ***Annotations*** (<*hasAnnotation*>).

Information Objects can acquire a further specialisation depending on the level of abstraction at which they are specified. This leads to an abstract ***Information object by level of abstraction*** concept, which is a container or placeholder to be specialised using any of several models. For example, the IFLA FRBR model [116] distinguishes:

- *Work*, for example the general idea of a story;
- *Expression*, for example the telling of a story in a text;
- *Manifestation*, for example the graphic image showing the letters and words that make up the text that is common to all copies printed from the same typeset image;
- *Item*, for example an individual printed copy of a manifestation.

Other divisions are possible. In particular, the FRBR distinction between *Work* and *Expression* is hard to apply in the digital world and therefore problematic.

Information objects can also be specialised by the predominant role they play in their relationship to other objects; the class ***Information object by relationship*** is the abstract conceptual container for the classes these objects give rise to, namely:

- *Primary Information Object*, an *Information Object* that stands on its own, such as a book or a data set;
- ***Metadata*** object, an *Information Object* whose predominant purpose is to give information about a 'target' *Resource* (usually, but not always, a *Primary Information Object*);
- ***Annotation*** object, an *Information Object* whose predominant purpose is to annotate a 'target' *Resource* (or a ***Region*** of it). Examples of such *Annotation Objects* include notes, structured comments, and links. *Annotation Objects* assist in the interpretation of the target *Resource*, or give support or objections or more detailed explanations.

This modelling style reflects a basic intuition that distinguishes this model from most DL models or *de facto* standards, namely that an *information object* is not born as (say) *Metadata* or an *Annotation*, but becomes such by virtue of playing a certain role in relation to other information objects. The intuition is based on the simple observation that, for instance, a Dublin Core metadata record is to be primarily modelled as a relational structure (record, tuple, graph fragment) which may also be associated to the resource it describes; it is this association that gives the structure the role of metadata. A similar case arises for a piece of text; it is primarily a piece of text, and becomes an annotation only when it is linked to a certain Resource in a certain way. In other words, the long-standing issue of whether annotations are content or metadata is just an ill-posed question.

From an organisational point of view, *Information Objects* can be grouped into ***Collections*** (<*belongTo*>), i.e. groups of objects considered as a single entity for certain management or application purposes. As *Collections* are *Information Objects*, they inherit all *Information Objects*' modelling aspects and facilities, e.g. they can be annotated. Moreover, Collections are a specialisation of the *Resource Set* concept. In fact, *Collections* are characterised by an intension (<*hasIntension*>) and an extension (<*hasExtension*>). The former is the criterion underlying the grouping. The way this criterion is expressed can range from the explicit enumeration of all the objects intended to be part of the group to logical expressions capturing the characteristics of the *Resources* intended to be part of the group. The latter is the concrete set of resources (*Resource Set*) matching the intension. These characteristics are implemented

differently in diverse systems, leading to scenarios ranging from static to highly dynamic, e.g. [50].

Another specialisation of the *Resource Set* concept usually associated with the *Content Domain* is the ***Result Set***. In traditional digital libraries this is the set of documents that are retrieved by issuing a ***Query.*** In this context it represents the set of *Resources*, with no constraints on their type, resulting from a *Query*.

II.2.3 User Domain

The *User Domain* represents all the entities that are external to a Digital Library 'system' and interact with it, that is: humans, and inanimate entities such as software programs or physical instruments. The latter may, for instance, include a subscription service offered by a university to its students, which provides access to the contents of an external Digital Library, or even another Digital Library may be among the users of a different Digital Library.

Inclusion of hardware and software into the potential users of digital libraries is a major deviation from other Digital Library models and reflects the breadth of our understanding and conceptualisation as expressed here. In order to capture the extended semantics of the word 'user', we use the concept of ***Actor*** (Figure II.2-4) as the dominant concept in this domain.

As a *Resource*, the *Actor* concept inherits all key characteristics of the former, i.e. it

(1) has a unique identifier (*Resource Identifier*), a.k.a. the user identifier;

(2) is arranged according to a format (*Resource Format*), a.k.a. the user model;

(3) can be arranged into arbitrarily complex and structured groupings because of the composition (<*hasPart*>) and linking (<*associatedWith*>) resource features, e.g. user cooperations or co-authorships can be captured by instantiating the <*associatedWith*> relations with the appropriate value of the *Purpose* attribute;

(4) is characterised by various *Quality Parameters* each capturing various quality facets (<*hasQuality*>); for instance, a human may be distinguished by *Trustworthiness* (cf. Section II.2.6);

(5) is regulated by *Policies* (<*regulatedBy*>) governing the aspects of its lifetime, such as the *Functions* an *Actor* can perform and the *Information Objects* they have access to; and

(6) can be enriched with *Metadata* (<*hasMetadata*>) and *Annotation* (<*hasAnnotation*>), e.g. a particular instance of *Actor* can mark or tag another instance with the characterisation 'friend'.

An *Actor* is represented in a Digital Library (<*modelledBy*>) via an ***Actor Profile*** and interacts (<*perform*>) with the Digital Library through a set of *Functions*.

The ***Actor Profile*** is an *Information Object* that essentially models an *Actor* by potentially capturing a large variety of the *Actor*'s characteristics. This may be important for a particular Digital Library because it allows the *Actor* to use the 'system' and interact with it as well as with other *Actors* in a personalised way. It not only serves as a representation of *Actor* in the system but also essentially determines the *Policies* and *Roles* that govern which *Functions* are allowed on which *Resources* during the lifetime of the *Actor*. For example, a particular instance of *Actor* may be entitled to *Search* within particular *Collections* and *Collaborate* with particular other *Actors* (cf. Section II.2.4). The characteristics captured in an *Actor Profile* vary depending on the type of *Actor*, i.e. human or non-human, and may include: identity information (e.g. age, residence or location for humans and operating system, web server edition for software components), educational information (e.g. highest degree achieved, field

of study – only for humans), and preferences (e.g. topics of interest, pertinent for both human and software *Actors* that interact with the Digital Library).

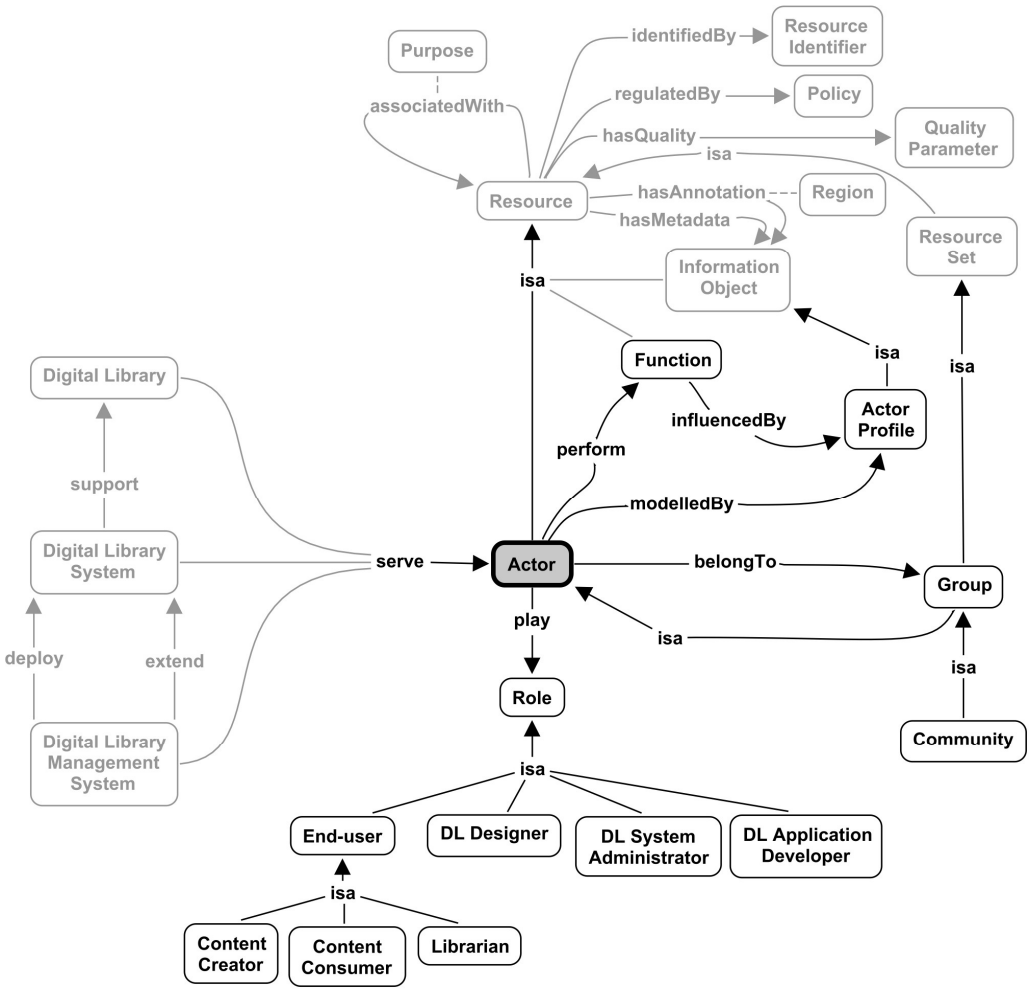


Figure II.2-4. User Domain Concept Map

An *Actor* may play a different ***Role*** at different times, something that is also a significant deviation from traditional approaches, where there are relatively impenetrable walls between *Roles* and each *Actor* can play only one of them. Among *Actor Roles*, important categories are ***End-user***, ***DL Designer***, ***DL System Administrator***, and ***DL Application Developer*** (Section I.4). Each of these *roles* plays a complementary activity along the 'system' lifetime. *End-user* exploits DL facilities for providing, consuming and managing DL content. It is further subdivided into the concepts of ***Content Creator***, ***Content Consumer*** and ***Librarian***, each of which usually has a different perspective on the Digital Library. For instance, a *Content Creator* may be a person that creates and inserts his own documents in the Digital Library or an external program that automatically converts artefacts to digital form and uploads them to the Digital Library. *Actors* in the role of *DL Designer* exploit DLMS facilities to define, customise and maintain the DL. *DL System Administrators* exploit DLMS facilities to create the DLS realising the DL. Finally, *DL Application Developers* exploit DLMS facilities to create and customise the constituents of the DLS and DLMS. Inclusion of this broad understanding of actor roles into the potential users of Digital Libraries is a major deviation from other Digital Library models that focus on the *End-user* part only.

Finally, an *Actor* may be part of (<*belongTo*>) a ***Group***. A *Group* represents an *Actor* population that exhibits cohesiveness to a large degree and can be considered as an *Actor* with its own profile and identifier. Members of a *Group* inherit (part of) the characteristics from the *Group*, such as interests, *Policies* and *Roles*, but they may have additional characteristics as described in their individual *Actor*'s profile. A particular subclass of *Group* is ***Community***, which refers to a social group of humans with shared interests. In human *Communities*, intent, belief, resources, preferences, needs, risks and a number of other conditions may be present and common, affecting the identity of the participants and their degree of cohesiveness.

II.2.4 Functionality Domain

The *Functionality Domain* represents the richest and most open-ended dimension of the world of Digital Libraries, as it captures all processing that can occur on *Resources* and activities that can be observed by *Actors* in a Digital Library. The most general functionality concept is ***Function*** (Figure II.2-5), i.e. a particular processing task that can be realised on a *Resource* or *Resource Set* as the result of an activity of a particular *Actor*. It is worth noting that this description of a *Function* is based on the generalised concepts of *Actor*, capturing not only human users but also inanimate entities, and of *Resource*, representing all entities involved in or influenced by a Digital Library, and lends a fresh perspective to the *Functionality* of a Digital Library. While functions in traditional digital library models are typically associated with content in the digital library and are performed by humans, under the new perspective a *Function* can be exercised by non-human users too on any type of *Resource*. For instance, not only can a user *Search* the contents in a digital library, i.e. *Information Objects*, but also an *Actor* can search for other *Actors*, a program can *Search* for offered *Functions*, and so forth.

Each *Function* is itself a *Resource* in this model and thus inherits all the characteristics of the former, namely:

(1) it has a unique identifier (*Resource Identifier*);

(2) it can be organised in arbitrarily complex and structured workflows because of the composition (<*hasPart*>) and linking (<*associatedWith*>) facilities, e.g. a compound function resulting by combining smaller sub-functions;

(3) it is characterised by various *Quality Parameters* covering various quality aspects (<*hasQuality*>);

(4) its lifetime and behaviour are regulated by *Policies* (<*regulatedBy*>), *e.g.* which *Actors* are allowed to perform the *Function* in a certain context; and

(5) it can be enriched with *Metadata* (<*hasMetadata*>) and *Annotation* (<*hasAnnotation*>).

Each *Function* subsumes processes on *Resources* and activities carried out by *Actors* and the supporting processes of the DLS. For example, *Browse* subsumes both the system function of generating a display suitable for browsing and the *Actor* function of browsing this display.

Besides the modelling issues, it is important to recall that the set of *Functions* each of the DL 'systems' provides is the direct consequence of its *Actor* expectations. *Functions* concur to realise what is usually called a 'business process' that is in the service of meeting specific 'business requirements' that satisfy a 'stakeholder need'. These concepts are borrowed from [148] where they are used to model the concrete business of a DL 'system'. These kinds of concepts needed to model the DL business are particularly important but they constitute a typical example of domain modelling that can be 'outsourced' to an *auxiliary domain* and thus they will not be elaborated further.

Because of the broad scope of the *Function* concept, it is not feasible to enumerate and predict all the different types and 'flavours' of *Functions* that may be included in any Digital Library.

Each Digital Library may have its own set of *Functions* depending on its objectives or its intended *Actors*. Therefore, *Function* is specialised into five other concepts that still represent quite general classes of activities, as outlined below (Figure II.2-5).

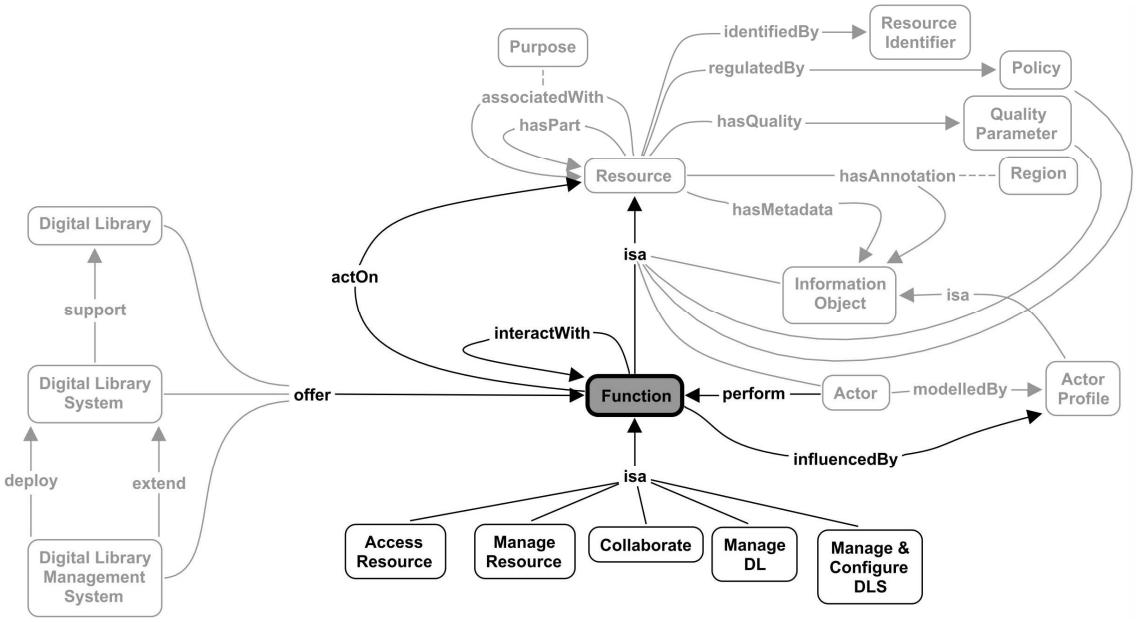


Figure II.2-5. Functionality Domain Concept Map

Access Resource encompasses all activities related to requesting, locating, retrieving, transforming and finally persisting *Resources* (Figure II.2-6). The key characteristic of the *Access Resource* concept is that it represents *Functions* that do not modify the Digital Library but help in identifying *Resources* intended to be simply examined and perceived by an *Actor* or possibly further exploited through the use of other *Functions*, such as *Manage Resource* functions. Hence, the central *Access Resource* function is *Discover*, which acts on *Resource Sets* to retrieve desired *Resources*.

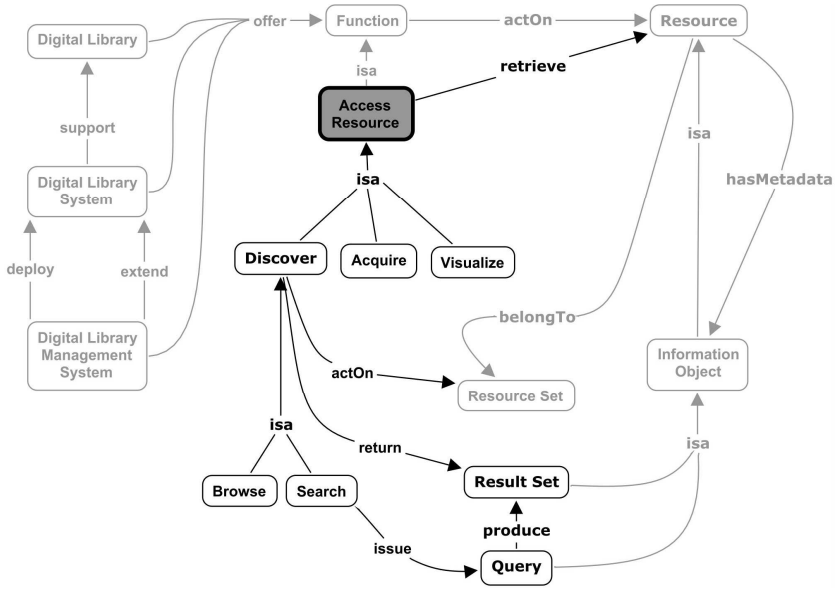


Figure II.2-6. Functionality Domain Concept Map: Access Resource Functions

Manage Resource includes all activities related to creating new *Resources*, inserting them into the DL, deleting old *Resources* from it, and updating existing ones, as well as applying

conversions and transformations on them. This transformation may lead to new *Resources* that may be submitted to the DL or be merely applied when accessing the *Resource*. These may be specialised to individual *Functions* for each resource type (Figure II.2-7).

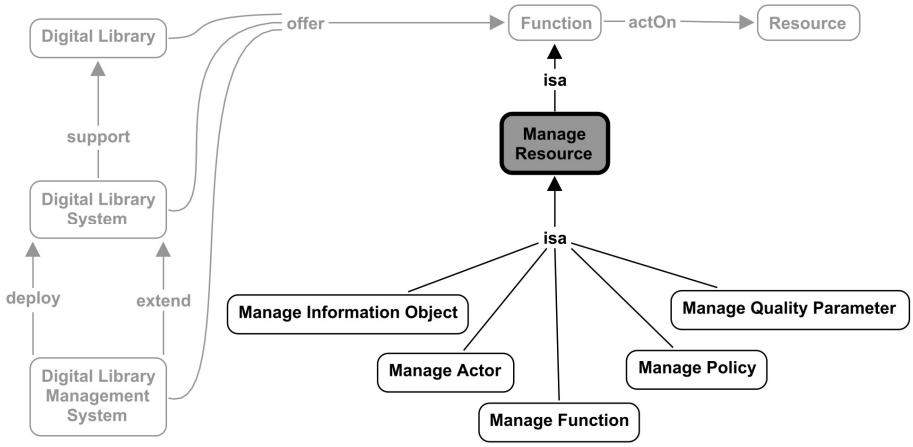


Figure II.2-7. Functionality Domain Concept Map: Specialisations of the Manage Resource Functions

Some of the *Functions* may be applied on the *Resources* and others are applied on the *metadata* describing those *Resources*. The general *Functions* that may be applied on all *Resources* are related to the creation, submission withdrawal, update, validation and annotation of *Resources*. Figure II.2-8 presents these general *Functions*. These *Functions* may be specialised for particular Resource types.

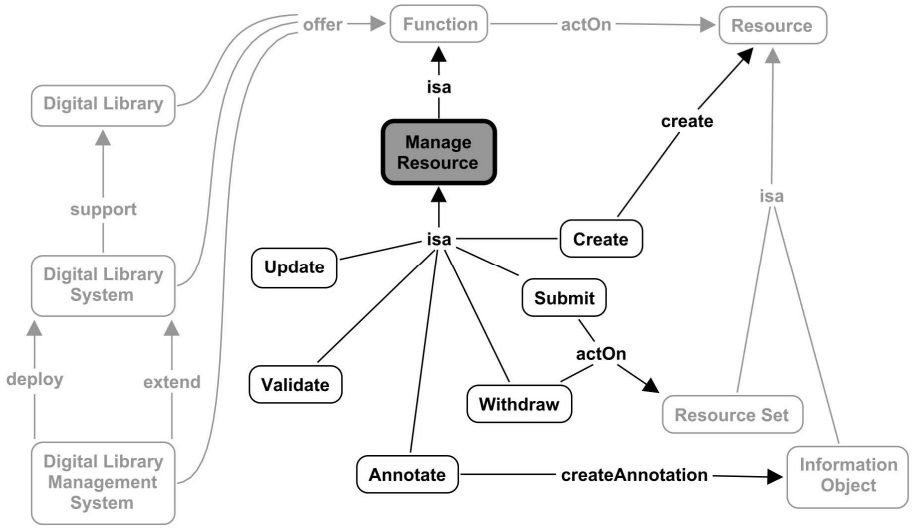


Figure II.2-8. Functionality Domain Concept Map: General Manage Resource Functions Applied to all Resources

Manage Information Object (Figure II.2-9) contains *Function* concepts that capture creation, processing and transformation for primary *Information Objects*, which are independent of any other, e.g. ***Author***, as well as other concepts that do the same for *Information Objects* that represent other *Information Objects* or *Resources* in general (such as references to others, compositions of others, etc.).

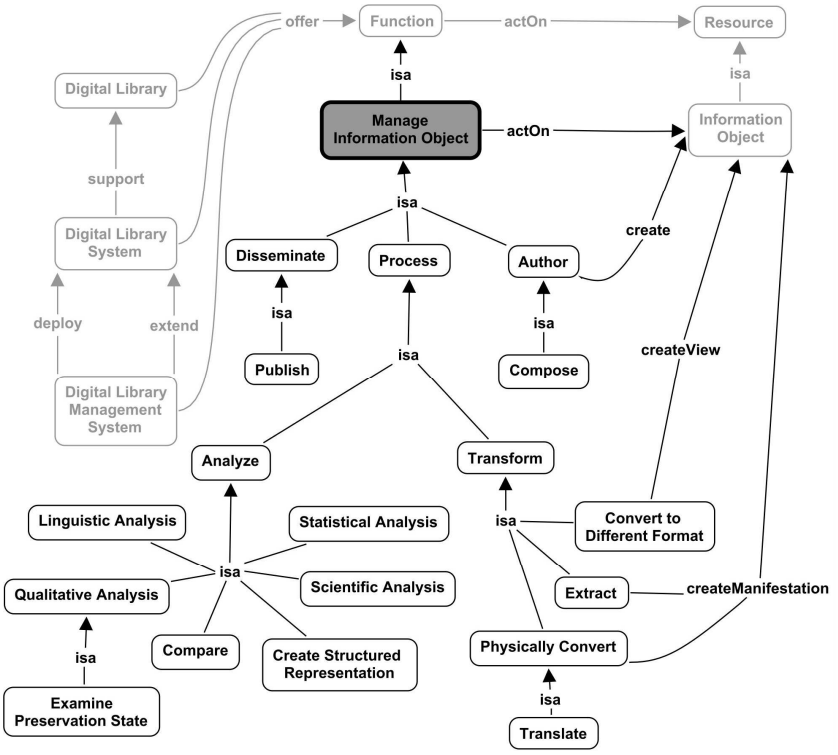


Figure II.2-9. Functionality Domain Concept Map: Manage Information Object Functions

Manage Actor contains *Functions* necessary for the management of individual *Actors* in the DL, including their registration or subscription, their login and the personalisation of the *functions* they are entitled to (Figure II.2-10).

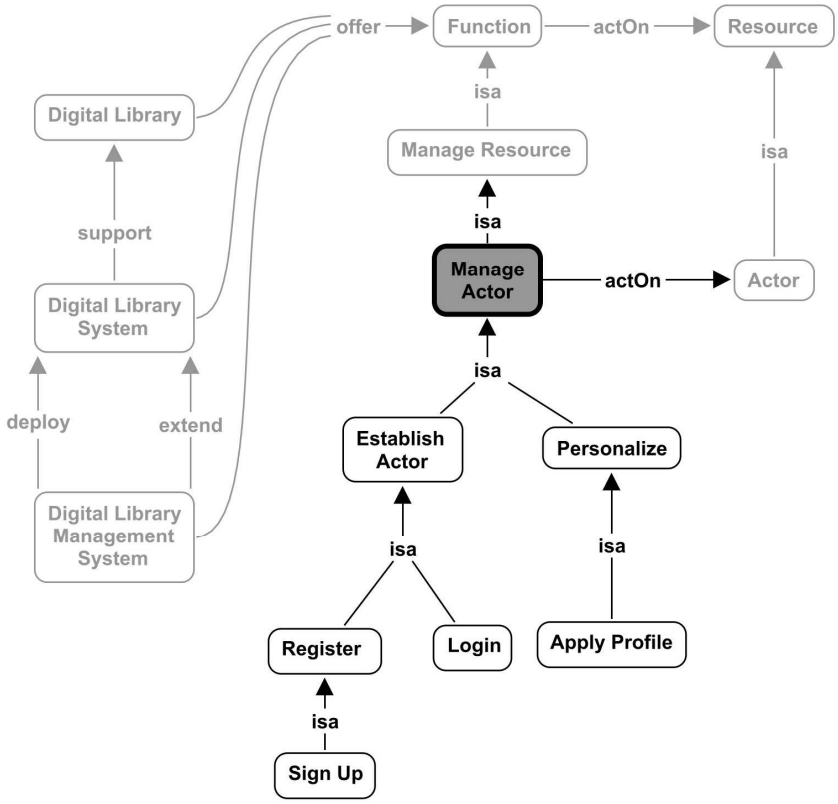


Figure II.2-10. Functionality Domain Concept Map: Manage Actor Functions

The third specialisation of *Function* is closely related to User Domain. It is the ***Collaborate*** function, which captures all activities that allow multiple *Actors* to work together through a DL to achieve a common goal.

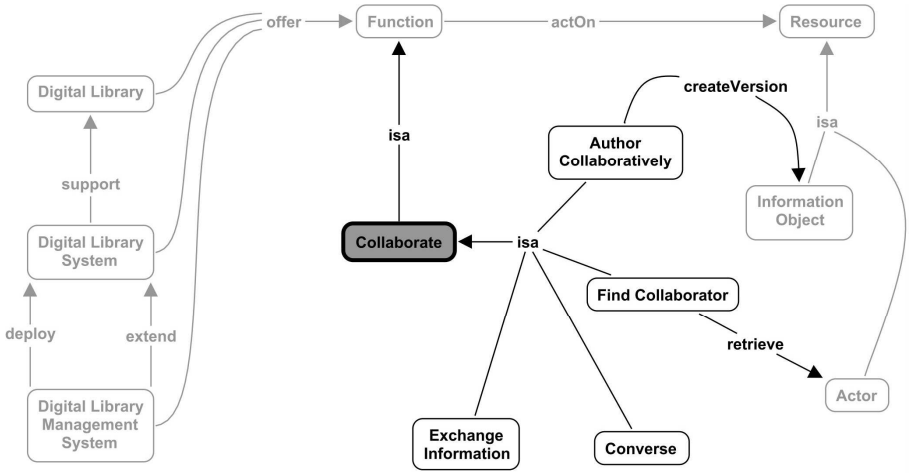


Figure II.2-11. Functionality Domain Concept Map: Collaborate Functions

The other specialisations of the *Function* concept encompass all activities related to the 'system' as a whole and its management.

Manage DL includes a wide variety of *Functions* (Figure II.2-12) that support the day-to-day management of the DL, with regard to all the DL domains. It includes the management of *collections*, user groups and membership, as well as general management of the policy, quality and functionality domain.

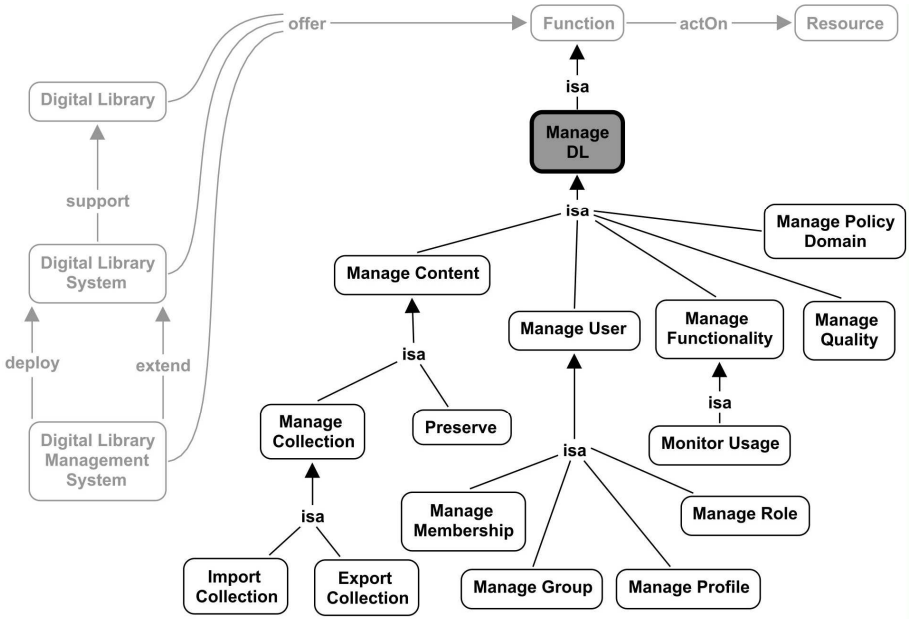


Figure II.2-12. Functionality Domain Concept Map: Manage DL Functions

Manage & Configure DLS (Figure II.2-13) contains *Functions* serving the *DL System Administrator role* with regard to setting up, configuring and monitoring the DL from a physical point of view, i.e. choosing the particular *Architectural Components* offered by the DLMS to bring the DL (actually the DLS) to an operational state, e.g. ***Deploy Architectural Component*** or ***Monitor Architectural Component***.

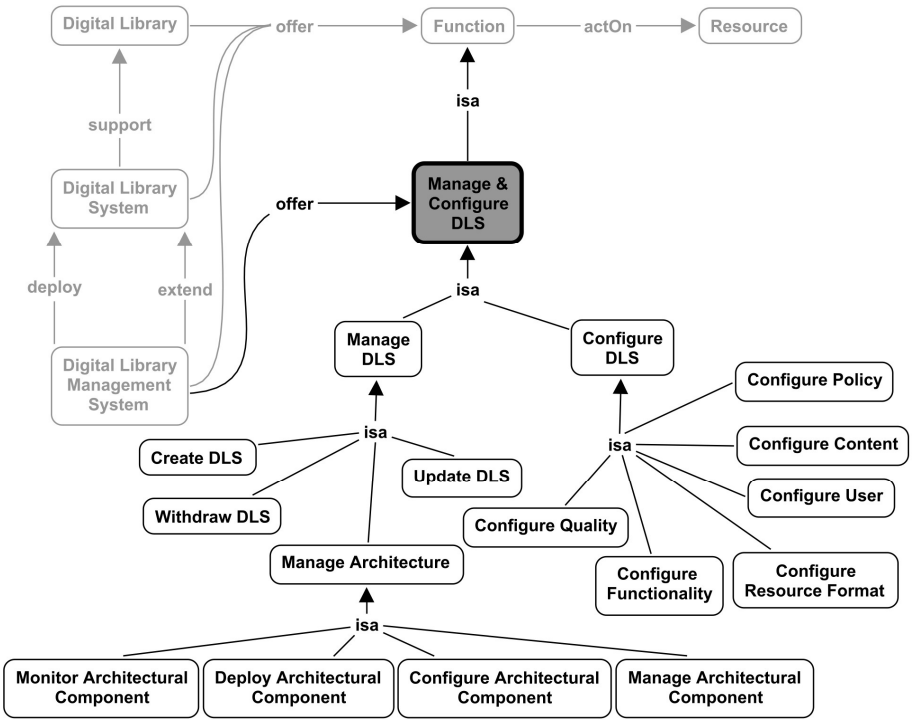


Figure II.2-13. Functionality Domain Concept Map: Manage DLS Functions

As mentioned earlier, the *Functionality Domain* is probably the most dynamic of all fundamental *DL domains*; hence, what is included in the present version of the Reference Model represents only a subset of *Functions* that one might imagine for DLs and corresponds to the concepts that are considered as most critical.

II.2.5 Policy Domain

The *Policy Domain* represents the set of conditions, rules, terms or regulations governing the operation of Digital Library systems. This domain is very broad and dynamic by nature. The representation provided by this model does not purport to be exhaustive especially with respect to the myriad of specific rules each Institution would like to model and apply. The Policy domain captures the minimal relationships connecting it to the rest and presents the subset of rules that are considered as most critical in the Digital Library universe. The model is extensible and, should other concepts be needed, they could easily be added in the appropriate place.

The most general policy concept is ***Policy*** (Figure II.2-14), the single entity governing a *Resource* with respect to a certain management point of view (<*regulatedBy*>). Each *Policy* is itself a *Resource* in this model and thus inherits all the characteristics of the former, namely:

(1) it has a unique identifier (*Resource Identifier*);

(2) it can be organised in arbitrarily complex and structured forms because of the composition (<*hasPart*>) and linking (<*associatedWith*>) facilities, e.g. a compound *Policy* can be obtained by properly combining constituent *Policies*;

(3) it is characterised by *Quality Parameters* covering various quality aspects (<*hasQuality*>), e.g. it is possible to measure the *Interoperability* or *Sustainability* (cf. Section II.2.6) of a *Policy*;

(4) it may itself be regulated by other *Policy* (<*regulatedBy*>), e.g. defining which *Actors* are subject to a certain *Prescriptive Policy* in a certain context; and

(5) it can be enriched with *Metadata* (<*hasMetadata*>) and *Annotation* (<*hasAnnotation*>).

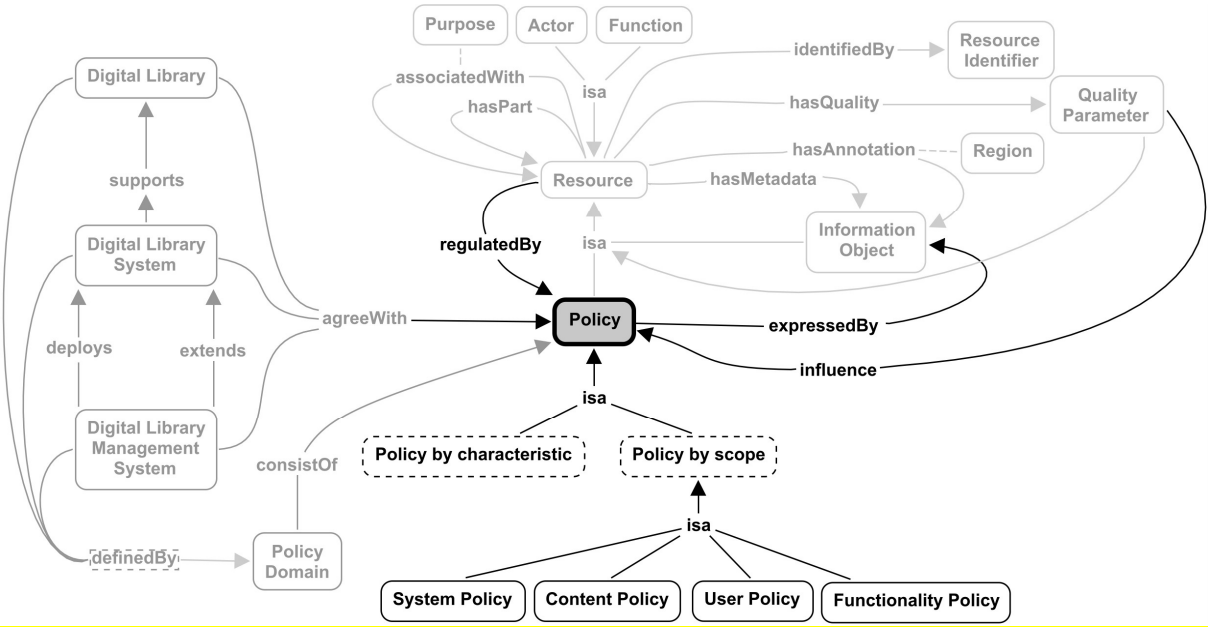


Figure II.2-14. Policy Domain Concept Map

Policy is actually a class of various types of policies (Figure II.2-15) – those currently most appropriate in digital library practice. For the purpose of this model, two abstract and orthogonal conceptual containers have been identified, i.e. ***Policy by characteristic*** and ***Policy by scope***.

Policy by characteristic is further specialised into eight subclasses grouped in four pairs, each presenting a bipolar quality a *Policy* might have: ***Extrinsic Policy*** vs. ***Intrinsic Policy***; ***Implicit Policy*** vs. ***Explicit Policy***; ***Prescriptive Policy*** vs. ***Descriptive Policy***; ***Enforced Policy*** *vs.* ***Voluntary Policy***. Understanding the characteristics of a specific *Policy* helps to express it better to the *Actors* and to clarify requirements at the functionality and implementation levels across the boundaries of the various systems.

Policy by scope is further specialised into various classes, each representing a particular *Policy* with respect to:

(1) the system as a whole, e.g. ***Resource Management Policy***,

(2) a certain domain, e.g. ***User Policy*** or ***Content Policy***; in some cases a *Policy* actually serves the needs of two domains, e.g. *Access Policy* is a *User Policy* and a *Functionality Policy* at the same time; or

(3) a specific task or entity, e.g. ***Collection Development Policy***.

It is important to recall that the model is extensible and does not intend to form an exhaustive list but rather a sample capturing some of the most important and diffused *Policies* governing the Digital Library universe. Among them, a special role is occupied by the *Digital Rights Management Policy* and *Digital Rights*. From the point of view of this model, these are instances of the *Policy* concept where *Digital Rights Management Policy* governs *Functions*, and *Digital Rights* govern *Information Objects*.

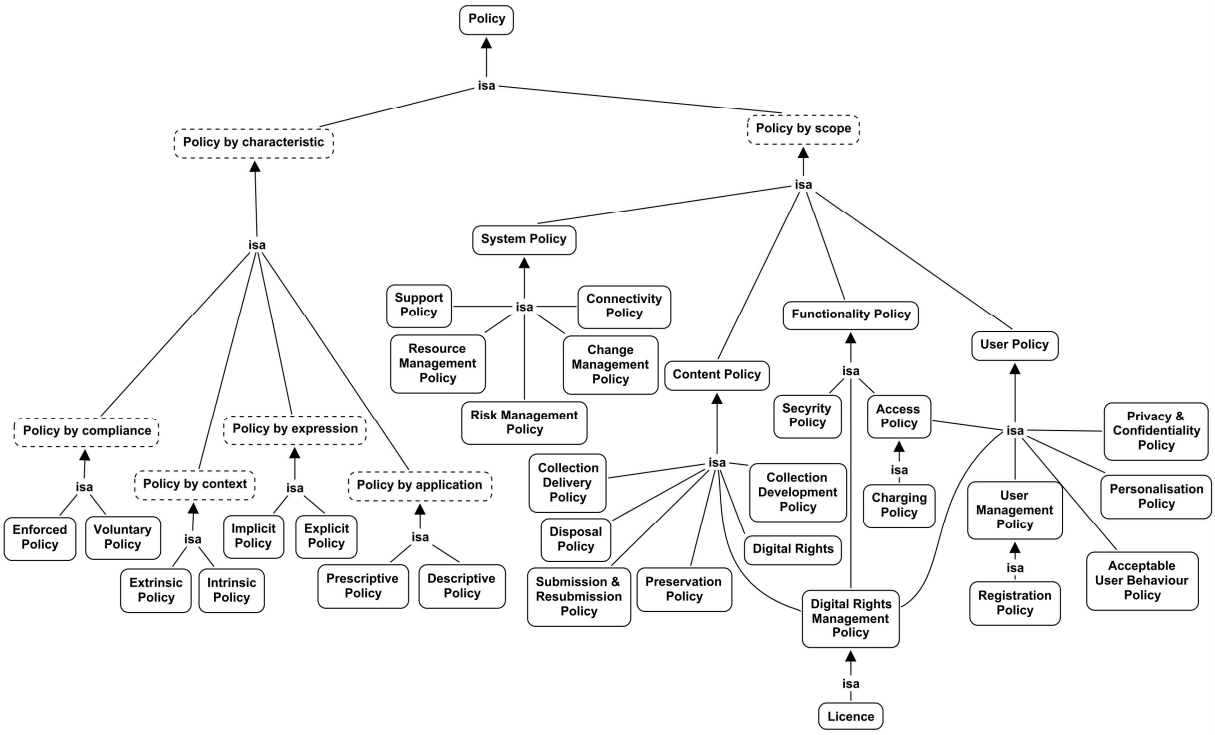


Figure II.2-15. Policy Domain Concept Map: Policies' Hierarchy

From the perspectives of *Digital Library*, *Digital Library System* and *Digital Library Management System*, there is no difference in the perception of the *Policy* concept but there are different *Resources* on which these systems apply *Policies*. Moreover, the same *Policy* has different materialisations in different systems, e.g. a private *Information Object* in a DL is managed by a DLS service instructed to deliver that object only to the *Actor* that is its owner.[12]

II.2.6 Quality Domain

The *Quality Domain* represents the aspects that permit considering digital library systems from a quality point of view, with the goal of judging and evaluating them with respect to specific facets. Any Digital Library 'system' tenders a certain level of *Quality* to its *Actors*. This level of Quality can be either implicitly agreed, meaning that *Actors* know what *Quality Parameters* guarantee, or explicitly formulated by means of a Quality of Service (QoS) agreement.

The most general quality concept is ***Quality Parameter*** (Figure II.2-16), i.e. the entity expressing the different facets of the *Quality Domain* and providing information about how and how well a *Resource* performs with respect to some viewpoint (<*hasQuality*>). Indeed, together with the concepts of *Actor*, *Resource*, ***Measure*** and ***Measurement***, the *Quality Parameter* provides the basic framework for dealing with the issues related to the broad concept of quality. *Quality Parameters* express the assessment by an *Actor*, whether human or not, of the *Resource* under consideration. The *Quality Parameters* can be evaluated according to different *Measures*, which provide alternative procedures for assessing different aspects of each *Quality Parameter* and assigning it a value. *Quality Parameters* are actually measured

[12] The DLS service is an instance of an *Architectural Component* (cf. Section II.2.7) appropriately configured by (and made available by) the DLMS.

by a *Measurement*, which represents the value assigned to a *Quality Parameter* with respect to a selected *Measure*.

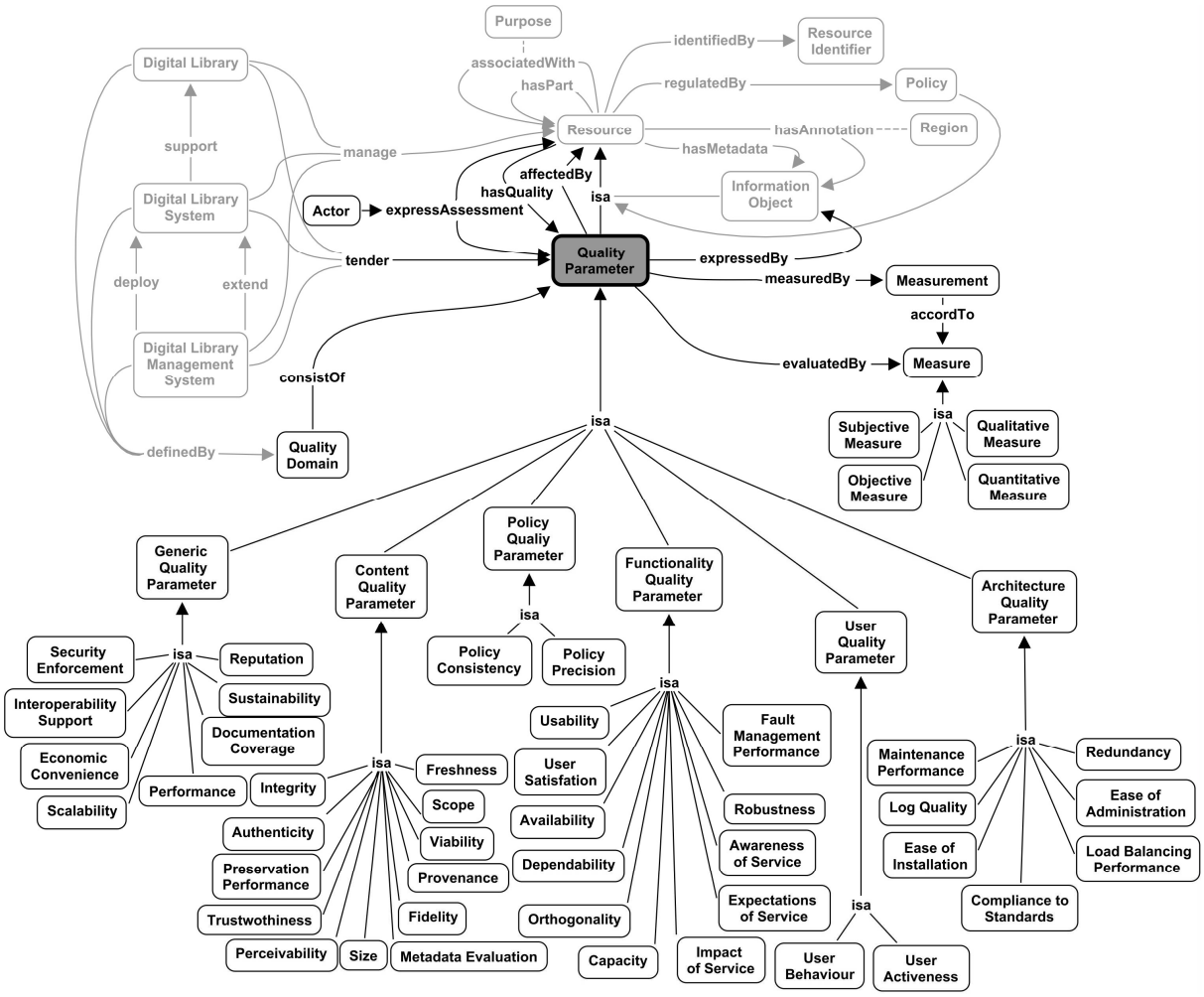


Figure II.2-16. Quality Domain Concept Map

In this model each *Quality Parameter* is itself a *Resource*, thus inheriting all its characteristics, namely:

(1) it has a unique identifier (*Resource Identifier*);

(2) it can be organised in arbitrarily complex and structured forms because of the composition (<*hasPart*>) and linking (<*associatedWith*>) facilities, e.g. a *Quality Parameter* can be the compound of smaller *Quality Parameters* each capturing a specific aspect of the whole;

(3) it is itself characterised by various *Quality Parameters* (<*hasQuality*>), e.g. it is possible to measure the *Sustainability* of the *Compliance to Standards* quality of an *Architectural Component* (cf. Section II.2.7);

(4) it may be specified by *Policies* (<*regulatedBy*>); and

(5) it can be enriched with *Metadata* (<*hasMetadata*>) and *Annotation* (<*hasAnnotation*>).

The *Quality Domain* is very broad and dynamic by nature. The representation provided by this model is therefore extensible with respect to the myriad of specific quality facets each Institution would like to model. *Quality Parameter* is actually a class of various types of quality facets, e.g. those that currently represent common practice. These parameters are grouped according to the *Resource* under examination (Figure II.2-16).

Generic Quality Parameters apply to any kind or most kinds of *Resources*.

System Quality Parameters apply to *Digital Library*, or a *Digital Library System*, or a *Digital Library Management System*.

Content Quality Parameters apply to *Resources* in the *Content Domain*, primarily *Information Objects*.

Functionality Quality Parameters apply to *Resources* in the *Functionality Domain*, primarily *Functions*.

User Quality Parameters apply to *Resources* in the *User Domain*, primarily *Actors*.

Policy Quality Parameters apply to *Resources* in the *Policy Domain*, primarily *Policies*.

Architecture Quality Parameters apply to *Architectural Components*, i.e. *Resources* belonging to the *Architecture Domain* (cf. Section II.2.7).

It is important to note that this grouping is made from the perspective of the *Resource* under examination, i.e. the object under assessment. In any case, the *Actor*, meant as the active subject who expresses the assessment, is always taken into consideration and explicitly modelled, since he is an integral part of the definition of *Quality Parameter*. Therefore, ***User Satisfaction*** has been grouped under the *Functionality Quality Parameter* because it expresses how much an *Actor* (the subject who makes the assessment) is satisfied when he/she/it uses a given *Function* (the object of the assessment). On the other hand, in the case of ***User Behaviour*** the object of the assessment is an *Actor* together with his way of behaving with respect to the *User Behaviour Policy*; for this reason, this parameter has been put under the *User Quality Parameter* group.

There is no fundamental difference in the perception of the *Quality Parameter* concept from the perspective of the *Digital Library*, that of the *Digital Library System* and that of the *Digital Library Management System*. However, each of these 'systems' applies this notion from a different perspective, e.g. the *Architecture Quality Parameters* are a peculiarity of the DLS and DLMS. Another difference consists in the fulfilment of the same *Quality Parameters* across the 'system' boundaries. For instance, if the DL specifies a certain *Quality Parameter*, it is a matter of the underlying *Digital Library System* fulfilling this claim, while it is the responsibility of the *Digital Library Management System* to provide for the assets needed to guarantee the user's expectations, e.g. by implementing the appropriate *Architecture*.

II.2.7 Architecture Domain

The *Architecture Domain* includes concepts and relationships characterising the two software systems playing an active role in the DL universe, i.e. DLSs and DLMSs. Unfortunately, the importance of this fundamental concept has been largely underestimated in the past. Having a clear architectural understanding of the software systems implementing the DL universe offers guidelines and ammunition on pragmatic realisations of a DL as a whole. In particular, it offers insights into:

- how to appropriately develop new systems, by maximising sharing and reuse of valuable assets in order to minimise the development cost and the time-to-market; and
- how to improve current systems by promoting the adoption of suitable, recognisable and accepted patterns in order to simplify interoperability issues.

The architecture of a 'software system' is a concept easily understood by most engineers, system administrators and developers, but it is not easily definable. In *An Introduction to Software Architecture* [89], Garlan and Shaw focus on design matters and suggest that software architecture is concerned with structural issues: 'Beyond the algorithms and data

structures of the computation, designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives'. The IEEE Working Group on Architecture [114], however, recognises that there is more than just structure in architecture, and defines it as 'the highest-level concept of a system in its environment'. Thus, this Group's understanding does not consider the architecture of a software system limited to an inner focus, but rather proposes to take into consideration the system as a whole in its usage and development environments.

For the purposes of this Reference Model, the architecture of a software system (at a given point) is defined as the organisation or structure of the system's significant components (***Architectural Component***) interacting with each other (<*use*>) through their interfaces (***Interface***). These components may in turn be composed of smaller and smaller components (<*composedBy*>) and interfaces (Figure II.2-17); however, different *Architectural Components* may be incompatible with each other (<*conflictWith*>), i.e. cannot coexist in the context of the same system. The software industry and the literature when using the term 'component' refer to many different concepts. Here, we use the term 'component' to mean an encapsulated part of a system, ideally a non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture. Each *Architectural Component* is a *Resource*, thus it inherits the *Resource*'s characterising aspects (cf. Section II.2.1), e.g. it is uniquely identified. As any *Resource*, components have *Metadata* (***Component Profile***) which provide fundamental information for managing them. These *Metadata* specify characteristics like the implemented or supported *Functions*, the implemented *Interfaces*, their governing *Policies*, and the *Quality Parameters* that specify the various quality facets describing how and how well the component performs with respect to some viewpoint.

Architectural Components interact through a ***Framework Specification***; they must also be conformant to it (<*conformTo*>). This framework prescribes the set of *Interfaces* to be implemented by the components and the protocols governing how components interact with each other.

Architectural Components are classified into ***Software Architecture Components*** and ***System Architecture Components***. These classes are used to describe the ***Software Architecture*** and the ***System Architecture*** of a software system respectively

Software Architecture Components are realised by ***Software Components***. In the case of each *Software Component*,

- the *Software Component* encapsulates the implementation of a portion of a software system (capturing *Content*, *User*, *Functionality*, *Policy* or *Quality Domain*s aspects of the DL universe);
- its usage is regulated by (<*regulatedBy*>) particular *Policies* (***Licenses***); and
- it is represented by an *Information Object* (<*representedBy*>).

Thus, the *Resource* representing the *Software Component* inherits the *Information Object*'s characterising aspects (Section II.2.2), e.g. it can be enriched through *Metadata* and *Annotations*.

System Architecture Components are realised by ***Hosting Nodes*** and ***Running Components***. A *Hosting Node* encapsulates the implementation of the environment needed to host and run

Software Components. A *Running Component* represents a running instance of a *Software Component* (<*realisedBy*>) active on a *Hosting Node*.

Thus, instances of *Software Architectural Components* and *System Architectural Components* capture the static (set of interacting *Software Architecture Components*) and dynamic (set of interacting *System Architecture Components*) views of the DLS and DLMS systems.

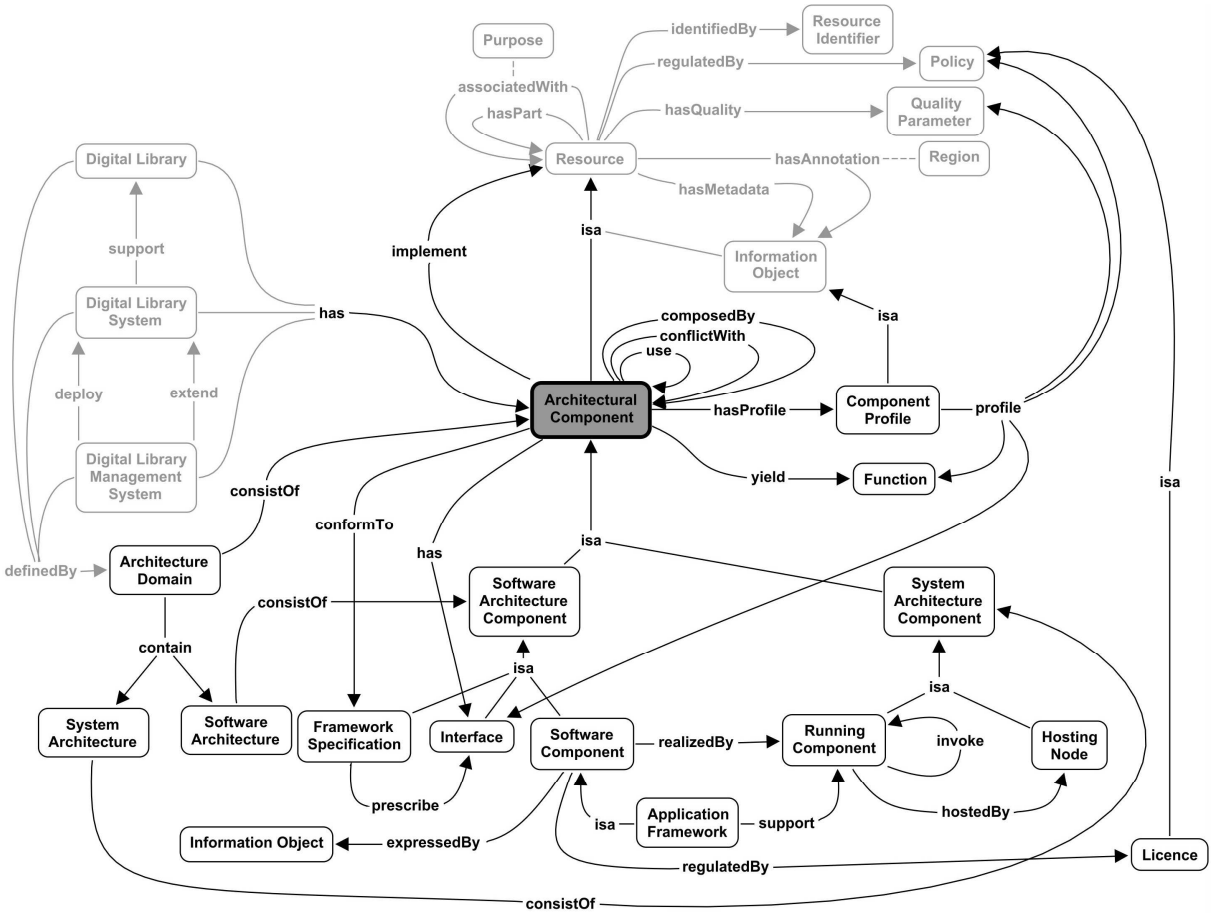


Figure II.2-17. Architecture Domain Concept Map

Even though the *System Architecture* of a DLS and the *System Architecture* of a DLMS are captured by the same set of concepts and relations, these systems are extremely different and play diverse roles in the DL universe. The aspects distinguishing a DLS from a DLMS, from the architectural point of view, reside in the concrete set of *Architectural Components* (in particular *Software Components*) constituting such systems. These differences are captured by the Reference Architecture documents, i.e. the Reference Model introduces the terminology to describe the systems, while the Reference Architecture must take care of identifying the concrete elements needed to implement an instance of either a DLS or a DLMS.

This modelling subsumes a 'component-based approach', i.e. a kind of application development in which:

- The system is assembled from discrete executable components, which are developed and deployed somewhat independently of one another, and potentially by different players.
- The system may be upgraded with smaller increments, i.e. by upgrading some of the constituent components only. In particular, this aspect is one of the key points for achieving interoperability, as upgrading the appropriate constituents of a system enables it to interact with other systems.

- Components may be shared by systems; this creates opportunities for reuse, which contributes significantly to lowering the development and maintenance costs and the time to market.
- Though not strictly related to their being component-based, component-based systems tend to be distributed.

All these characteristics represent high desiderata of current and future generations of DL 'systems'.

II.3 Reference Model in Action

The Reference Model sets out to contribute to digital library foundations, but its value is not merely theoretical. It also provides a core instrument for a large variety of different concrete usages, as demonstrated by the feedback received since the release of its first draft version. The Manifesto, for example, has been exploited several times to clarify to stakeholders the complexity of the Digital Library universe and the value of the Digital Library 'systems' in the content production and management workflow. At a very different level, the detailed specification of the concepts and relationships that characterise a Digital Library has been largely exploited in designing a concrete software service [91] that partially automates the process of creation of (virtual) digital libraries. Through this service, the effort spent by digital library designers and system administrators in performing this task is considerably reduced. The Reference Model has also been used as a basis for educational courses on digital libraries. Even if limited, the experience so far shows that the model provides a good integrated framework for introducing and explaining concepts. Starting from this framework, existing systems can easily be described and compared.

As outlined in the Manifesto, the Reference Model is also a first necessary step towards the definition of Reference Architectures. The introduction of Reference Architectures has been one of the main motivations for the definitional work carried out so far. As a matter of fact, Reference Architectures are mandatory for systematising the development of good quality digital library systems and for the integration and reuse of their components.

Among the many other usages of the Reference Model that emerged during the numerous discussions about it, two merit special attentions, namely those related to the treatment of *interoperability* and *preservation*. These are two closely related issues since preservation can be interpreted as 'interoperability over time'. They are discussed briefly in the next two sections. The considerations therein represent the result of a preliminary investigation of these issues in the light of the new framework introduced by the Reference Model. This result is very promising and we expect that a more in-depth analysis will be able to identify more systematic approaches and methodologies for handling these issues and suitable metrics to measure the degree of interoperability/preservation achieved.

II.3.1 The Interoperability Issue

Ultimately, the Digital Library Reference Model is intended to deal with the entire spectrum of Digital Library 'systems'. Whenever two or more systems decide to operate together to better serve their clientele, a scenario arises where the *interoperability* issue comes up. So far, the Reference Model focuses on describing and analysing an individual Digital Library but it is planned to extend its scope in the next phase to address the other scenario and the resulting issues. In fact, the modelling of interoperability among digital libraries is a really important aspect, as the topic of making systems able to exploit each other (either as a whole or with respect to some of their constituents, e.g. *Content*) is fundamental for the development of current and future systems. This section provides initial thoughts on this problem and lists the Reference Model concepts deemed to be of particular importance for interoperability.

In order to capture the context in which the interoperability issue arises, the notion of ***Digital Library Space*** can be introduced as a specialisation of *Resource Set* to denote a set of resources coming from several Digital Library 'systems'. Interoperability concerns providing the *Resources* constituting a *Digital Library Space* with seamless access to the rest of the *Resources* in the same space, independently of the Digital Library 'system' from which they originate.

Achieving interoperability requires a clear and detailed understanding of the participating entities. The Reference Model provides a framework for describing and understanding digital libraries in such a way that they can be easily compared, and commonalities and differences easily identified. This then leads to an assessment of interoperability problems (an interoperability audit) as the basis for a plan for achieving interoperability. By approaching the interoperability problem through the Reference Model, for example, it becomes clear that its solution does not depend, as usually thought, only on metadata, protocols and a few other aspects. In fact, interoperability is a multidimensional property that applies to the resources of all the different Digital Library universe domains, i.e. *Content*, *Functionality*, *User*, *Quality*, *Policy* and *Architecture*. This implies, for instance, that when building a digital library that integrates content from multiple different digital libraries a developer may not only be concerned with finding out cross-walks between metadata formats but also with many other aspects, such as defining mechanisms that ensure that the measures of the content quality parameter *Freshness* are interoperable with the measures of the same quality parameter in the participant *Resources*.

Through reasoning on the Reference Model, a notion of 'degree of interoperability' within a certain *Digital Library Space* can also be introduced. This degree is based on which concepts and relationships are interoperable in the Digital Library Space. A Digital Library can be classified as being interoperable with another one, for example at the level of *ontologies* and/or at the level of *Information Object*. The latter indicates a higher degree of interoperability since it subsumes the former.

Alternative degrees of interoperability are often put in place. For instance, in the case of searching across multiple data sources provided by diverse organisations (digital libraries), usually three different approaches, characterised by a different level of engagement of the sources, are realised: the federated, the harvesting, and the gathering approach. In the federated approach the participating organisations agree on a set of protocols and standards to be applied in delivering the search, e.g. each source implements the SRU/SRW protocol because the federation imposes it. In this case, the semantic interoperability is at the level of *query* and *result set*. In the case of the harvesting approach (a notable example is represented by OAI-PMH [143]), the participating organisations make their content ready to be used by third parties according to a certain standard. Thus, no imposition comes from the potential consumers. Here, semantic interoperability is usually based on the use of a common *ontology*, e.g. Dublin Core. The third case, i.e. the gathering approach, is the least demanding of the three. In this case, no source takes care of its potential consumers, as the exposition of its content so that it can easily be used by third parties is not a requirement; in other words, in this case the *resources* are not required to be interoperable.

The Interoperability issue has many commonalities with preservation and multilinguality. In fact, multilinguality can be seen as interoperability over languages while preservation can be seen as interoperability over time. Syntactic and semantic aspects pervade any form of interoperability. Both these aspects are equally important and generally used to discriminate between the aspects to be bridged. In practice, semantic interoperability is deemed to be more important and to require more sophisticated approaches than syntactic interoperability. However, semantic interoperability cannot be attained without reaching syntactic interoperability.

Among the various concepts reported in the Reference Model, the following are deemed to be particularly important to interoperability:

- ***Resource <hasMetadata> Information Object*** makes it possible to capture any Metadata for supporting interoperability.

- ***Resource <hasFormat> Resource Format*** makes it possible to capture the *Resource Format* with which a *Resource* is compliant. The notion of format is important for the correct interpretation of a *Resource*. For instance, in order for DL A to use an *Information Object* from DL B, DL A must either be able to deal with that *Information Object*'s format (read it, create displays, etc.) or be able to convert it to a format it can deal with.

  Ontology with its specialisation *Resource Format* is very important for interoperability. Format specifications need to be preserved so that *Information Objects* using an old format or a previous version of an existing format can still be interpreted. Likewise, the different versions of a subject ontology need to be preserved so that subject *metadata* prepared using a previous version of an *ontology* can be interpreted properly.
- ***Resource <associatedWith> Resource*** makes it possible to capture the context from which a *Resource* has originated. Having this knowledge is important for the correct understanding of the *Resource* meaning. Seeing an *Information Object* in its original context is important for the correct understanding of its meaning.

The following *functions* are especially important for interoperability:

- ***Transform***, a specialisation of the ***Process*** *Function* of ***Manage Information Object***. This may include format conversions, information extraction, and automatic translation and summarisation techniques. Its specialisation ***Convert*** includes conversion into a different encoding (converting a text from pdf to Word, an image to a different format or compression scheme, etc).
- ***Import Collection*****,** a specialisation of ***Manage Collection***, supports the selection of the third-party information sources whose objects will populate the DL Content or be used as *Resource Metadata*, for example, *Actor Profiles*.
- ***Export Collection*****,** a specialisation of ***Manage Collection***, supports the export of an entire Digital Library or parts of it to create a mirror site or to create a backup copy. Also making *Information Objects*, especially *metadata*, available to be imported by another system (harvesting) is a possible result of this *function*.
- ***Compare***, a specialisation of the ***Analyse*** Function, may be used to ascertain whether two instances of an *Information Object* are the same.

II.3.2 The Preservation Issue

The preservation imperative pervades all aspects of Digital Library 'systems'. This section draws together the Reference Model concepts that are deemed most important for addressing the preservation issue. Preservation applies to all types of *resources* but most importantly to *Information Objects*. We have specifically chosen to view preservation as embedded within the Digital Library System.

The working definition of preservation of *Information Objects* on which this work is based is the following:

Preservation aims to:

- *maintain a physically intact instance of a digital entity in the face of deterioration of physical storage media and signals recorded on them;*
- *ensure that the syntax of this digital entity (its encoding and format) can be interpreted and that each subsequent instantiation (e.g. access, rendering, manipulation) is identical to the initial instantiation (e.g. with regard to behaviour, including look and feel, or functionality);*

- *ensure that the semantic meaning of the digital entity is accessible across space and time in the face of technological and cultural change.*

Doing this effectively requires that the provenance and authenticity of digital entities are secured, that their 'interrelatedness' is retained, and that information about the context of their creation and use continues to be available. At the most conceptual level, full understanding of an *Information Object* requires knowledge of the cultural context and of the meaning of the representation mechanism, such as term or graphic or sound elements, used by the creator of the object at the time of creation.

Preservation might also be viewed as interoperability over time.

The preservation challenge addressed by this section applies to any form of digital information managed by the Digital Library System, thus also to information about *Actors*, *Functions*, *Policies*, and to the system as a whole. For some purposes it would be useful to know and be able to reproduce the state of a Digital Library System at a particular point in time in the past. This includes in particular the configuration of *Functions*. For example, one might want to reproduce the user interface that was in operation three years ago so that a user familiar with that particular interface can still use it. Or a scholar in the future might wish to study how individual or groups of users of the content held by a digital library were accessing that material. Further, one might want to preserve user personalisation stored in an *Actor Profile* in the face of changes in the digital library system.

This Reference Model provides the general framework for discussing preservation through the definitions of ***Resource*** and ***Information Object***. It contains the specific concepts and relations necessary to model preservation as listed below and illustrated in Figure II.3-1:

- ***Resource <hasMetadata> Information Object*** makes it possible to capture any Metadata, or representation information, necessary to support preservation. Many different kinds of metadata data are needed for preservation. Ideally, *Information Objects* (any *Resource*) would be provided with metadata sufficient to enable the automation of preservation processes. This includes, for example, the date when an *Information Object* can be destroyed.
- ***Resource <hasFormat> Resource Format*** makes it possible to capture the format (e.g. characteristics or properties) of an *Information Object* (in general, *Resource*) required if the *Information Object* is to be accessed and understood whether by person or machine. This notion of format can be used to determine when the technology needed for interpreting the object disappears, and migration to a different format is necessary. The issue of format applies both to primary *Information Objects* and to *Metadata Objects*, which are also *Information Objects*.

  Ontology with its specialisation *Resource Format* lies at the heart of preservation systems. Format specifications need to be preserved so that *Information Objects* using an old format or a previous version of an existing format can continue to be interpreted. Likewise, the different versions of a subject ontology need to be preserved so that subject metadata prepared using a previous version of an ontology can be interpreted accurately.
- ***Resource <hasQuality> Quality Parameter*** makes it possible to capture the quality parameters deemed relevant to the preservation issue.
- ***Resource <associatedWith> Resource*** supports the capture of the context from which an *Information Object* (in general, any *Resource*) originated. This information facilitates the interpretation of an object in case the context provides critical semantic value.

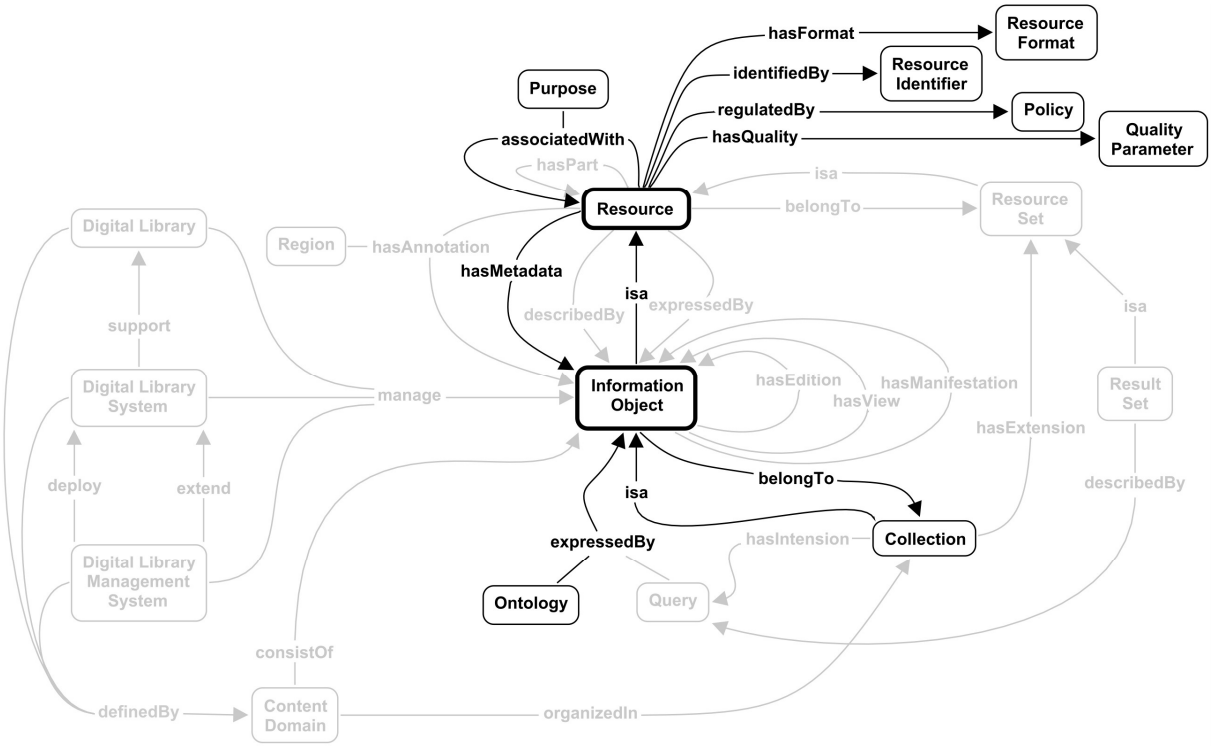


Figure II.3-1. Content Domain Concept Map with Highlighted Elements Essential for Preservation Activities

Moreover, the Reference Model introduces *Functions* that are crucial for preservation, as follows:

- ***Transform*** – the family of *Functions* through which *Resources* (*Information Objects*) represented according to a given *Resource Format* are transformed into *Resources* (*Information Objects*) expressed according to another *Resource Format*, improving the capability to transport and interpret them across representation devices and time.
- ***Visualise*** – the *Function* supporting *Resource* (*Information Object*) rendering. This should be equipped with facilities for preserving behaviour and functionality of information objects across systems and time.
- ***Withdraw*** – the *Function* making it possible to drop *Resources* (*Information Object*) from a Digital Library 'system'. From a preservation point of view, this *function* should enable mechanisms to decide whether to maintain the withdrawn object in a secondary store or to completely delete it.
- ***Export*** – the *Function* allowing exporting of an entire digital library or parts of it. This might be done to create a mirror site or a backup copy, or to move a digital library or elements of it to another technological environment. The *Resource* resulting from the execution of this *function* must have a *Resource Format* making itself interpretable and importable by another system.
- ***Compare*** – the *Function* that allows a person or a computer program to ascertain the identity or similarity between two instances of an *Information Object* (more generally, a *Resource*). By combining this *Function* with the *Quality Parameters* asserting the *Information Object* (more generally, a *Resource*) probability of being correctly interpreted across time, it will be possible to automate the application of *Preservation Policies*.
- ***Configure DL*** – For preservation, the system should save the configuration state after any changes are made to it.

- ***(actions) logging*** – the *Function* recording the actions performed on the *Information Object* (*Resource*) across time. This logging information (which can be considered a kind of *Metadata*) can be used for preservation purposes in different ways, e.g.
  - It allows for rollback operations, such as returning an *Information Object* (more generally, a *Resource*) to a state it has had at a particular time in the past;
  - It provides for usage history of *Information Objects* (more generally, a *Resource*), which is important as context for later uses.

Two ***Policies*** relate directly to preservation:

- ***Preservation policy*****,** which governs the preservation tasks including selection and appraisal of *Resources*.
- ***Disposal policy***, which governs the de-accession tasks. In the sense that *disposal policy* specifies what should not be preserved, it is subsumed under *preservation policy*.

Digital rights also play a significant role in preservation in that they govern what preservation measures can be taken, especially with regard to the making of backup copies

Among the ***Quality Parameters***, the following are of particular importance for preservation:

- Generic Quality Parameters:
  - ***Security Enforcement*** (cf. Section III.3 C158)
  - ***Interoperability Support*** (cf. Section III.3 C156)
  - ***Documentation Coverage*** (cf. Section III.3 C160)
- Content Quality Parameters:
  - ***Integrity*** (cf. Section III.3 C167)
  - ***Authenticity*** (cf. Section III.3 C164)
  - ***Trustworthiness*** (cf. Section III.3 C165)
  - ***Preservation Performance*** (cf. Section III.3 C168)
  - ***Fidelity*** (cf. Section III.3 C172)
  - ***Dependability*** (cf. Section III.3 C184)
  - ***Provenance*** (cf. Section III.3 C169)
- Functionality Quality Parameters:
  - ***Fault Management Performance*** (cf. Section III.3 C181)
- Architecture Quality Parameters:
  - ***Compliance with standards*** (cf. Section III.3 C196)

II.4 Related Work

Several initiatives related to issues discussed in this document have been performed in the past. In the remainder of this section we briefly compare this Reference Model with the most representative of these.

II.4.1 The CIDOC Conceptual Reference Model

The CIDOC Conceptual Reference Model (CRM) [199] is an initiative whose goal is to provide a model, i.e. a formal ontology, for describing implicit and explicit concepts and relationships needed to describe cultural heritage documentation. This activity started in 1996 under the auspices of the ICOM-CIDOC Documentation Standard Working Group and since December 2006 it has been an official ISO standard (ISO 21127:2006) [122].

It consists of 81 classes, i.e. categories of items sharing one or more common traits, and 132 unique properties, i.e. relationships of a specific kind linking two classes. Moreover, classes as well as properties are organised in a hierarchy through the 'is a' relationship.

The CIDOC reference model classifies the rest as the *CRM Entity*, i.e. the class comprising all things in the CIDOC universe and the *Primitive Value* class, i.e. the class representing values used as documentation elements (*Number*, *String* and *Time Primitive*). This second class is not elaborated further. The entities of the CIDOC universe are further classified in *Temporal Entity*, i.e. phenomena and cultural manifestations bounded in time and space; *Persistent Item*, i.e. items having a persistent identity; *Time-Span*, i.e. abstract temporal extents having a beginning, an end and a duration; *Place*, i.e. extents in space in the pure sense of physics; and *Dimension*, i.e. quantifiable properties that can be approximated by numerical values.

The *Persistent Item* class can be compared to our notion of *Resource* as univocal identified entity (*Resource Identifier*). It is further specialised to form a hierarchy. *Thing* is the direct subclass and represents usable discrete, identifiable instances of persistent items documented as single units. At this point a complex hierarchy of *things* classes is introduced. In this hierarchy three classes need to be further explained, namely *Conceptual Object*, *Information Object* and *Collection*. A *Conceptual Object* is defined as 'non-material product of our minds, in order to allow for reasoning about their identity, circumstances of creation and historical implications'. It shares many commonalities with the IFLA-FRBR concept of Work [116], while its counterpart in the Digital Library Reference Model is the *Information Object*. The CIDOC-CMR *Information Objects* are defined as 'identifiable immaterial items, such as poems, jokes, data sets, images, texts, multimedia objects, procedural prescriptions, computer program code, algorithm or mathematical formulae, that have an objectively recognisable structure and are documented as single units'. The CIDOC *Information Object* concept falls within the concept of *Information Object* of the Digital Library Reference Model. The CIDOC model takes care of complex *Information Objects* through the '*is composed of*' property as well as of rights ownership through the linking between *Legal Object*[13] and *Right*. *Collection* is defined as 'aggregation of physical items that are assembled and maintained by one or more instances of Actor over time for a specific purpose and audience, and accounting to a particular collection development plan'. Thus, differing from the Digital Library Reference Model, the CIDOC-CRM only refers collections to physical instantiation of such aggregative mechanism.

Actor, i.e. people who individually or as a group have the potential to perform actions of which they can be deemed responsible, is introduced as a specialisation of the *Persistent Item*

[13] An *Information Object* is also a *Legal Object*, i.e. a material or immaterial item to which instances of *Right* can be applied.

class. This concept presents many commonalities with the one introduced in the Digital Library Reference Model and presented in Section II.2.2.

Another specialisation of the *Persistent Item* class is *Appellation*, i.e. any sort of identifier that can be used to identify specific instances of all the classes. The two models dedicate a different effort to modelling this aspect. While the Digital Library Reference Model introduces the concept of *Resource Identifier* without specialising it, the CIDOC-CRM introduces many specialisations ranging from *Object Identifier* to *Address*, *Title* and *Date*.

Finally, the CIDOC-CRM captures also aspects related to the notion of *Functionality*. In fact, even if its goal is to provide an ontology for modelling cultural heritage information, some of its classes aim at capturing the history and evolution of such information and thus can be considered as a sort of *Function* to which objects/information have been subjected. In particular, the role of the *Activity* class is to comprise 'actions intentionally carried out by instances of Actor that result in changes of state in the cultural, social, or physical systems documented'.

II.4.2 Stream, Structures, Spaces, Scenarios and Societies: The 5S Framework

The 5S framework [95][97] is the result of an activity aimed at defining digital libraries in a rigorous manner. It is based on five fundamental abstractions, namely *Streams*, *Structures*, *Spaces*, *Scenario*s and *Societies*.

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 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 takes 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 conforming to certain constraints. This type of construct is powerful and, as suggested by the conceivers, 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 proposed, e.g. measurable spaces, measure spaces, probability spaces, vector spaces and topological spaces.
- *Scenarios* are sequences of events that may have parameters, and events represent state transitions. The state is determined by the content in a specific location but the value and the location are not investigated further because these aspects are system dependent. Thus a scenario tells what happens to the *streams* in *spaces* and through the *structures*. When considered together, the *scenarios* describe the services, the activities and the tasks representing digital library *functions*. DL workflows and dataflows are examples of scenarios.
- *Societies* are sets of entities and relationships. The entities may be humans or software and hardware components, which either use or support digital library services. Thus, society

represents 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.

We can relate the 5S to some of the aims of a Digital Library:

- Societies define how a Digital Library helps in satisfying the information needs of its users.
- Scenarios provide support for the definition and design of different kinds of services.
- Structures support the organisation of the information in usable and meaningful ways.
- Spaces deal with the presentation and access to information in usable and effective ways.
- Streams concern the communication and consumption of information by users.

These concepts are of general purpose and represents low-level constructors. Using these concepts, Gonçalves *et al*. introduced the whole DL ontology reported in Figure II.4-1.

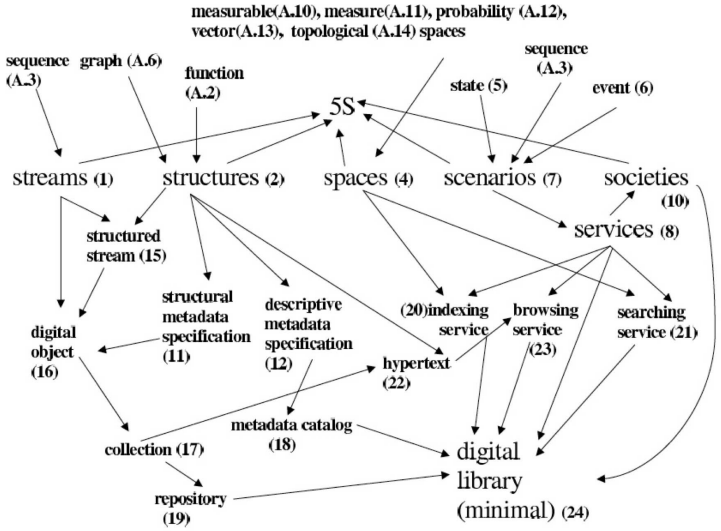


Figure II.4-1. 5S: Map of Formal Definitions[14]

As shown in the figure above, where the arrows signify that a concept depends on another concept for its definition, the different Ss are defined starting from basic mathematical concepts, such as graph or function, and are then combined and used to introduce the specific concepts that characterise the Digital Library universe. For example, the concept of digital object is defined in terms of the streams and structures that constitute it and, in turn, is used for introducing the concept of collection.

In accordance with this framework, Gonçalves *et al*. define a minimal Digital Library as a quadruple (R,Cat,Serv,Soc) where:

(1) R is a repository, a service encapsulating a family of collections and specific services (get, store and del) to manipulate the collections;

(2) Cat is a set of metadata catalogues for all collections in the repository;

(3) Serv is a set of services containing at least services for indexing, searching and browsing; and

(4) Soc is a society.

On top of this, a framework aimed at arranging the concepts and identifying the relationships between them has been proposed. It is depicted in Figure II.4-2.

[14] Figure II.4-1 is extracted from [95].

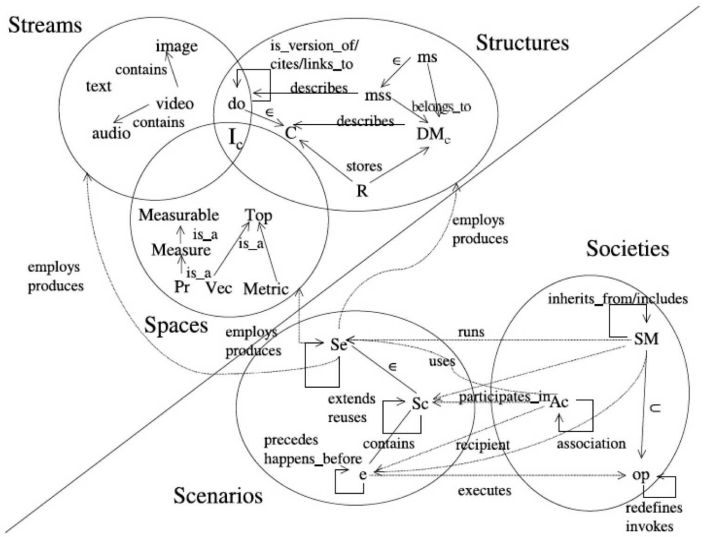


Figure II.4-2. 5S: DL ontology[15]

Figure II.4-3 shows a correspondence between the area covered by the 5S framework and the Reference Model: 5S basically covers what in the Reference Model have been called *Content*, *Functionality* and *User* main concepts; the *Quality* main concept has been addressed separately in the 5S Quality model [98], while the *Policy* main concept has scarcely been dealt with in the 5S framework. Moreover, the degree of detail in the different areas can vary, since in some areas the 5S framework introduces very fine-grained concepts while in other areas it adopts a more high-level approach; similar considerations also hold for the Reference Model.

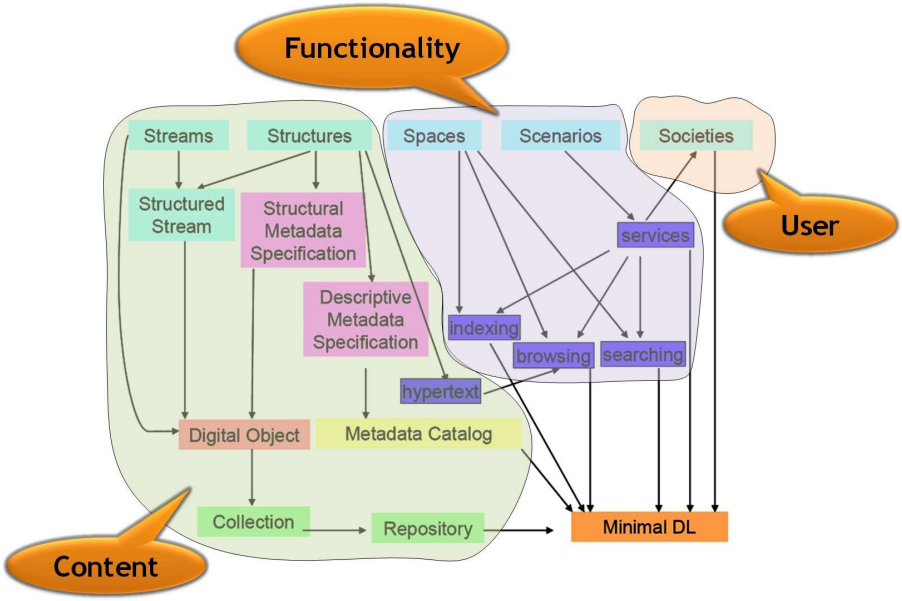


Figure II.4-3. 5S: Areas Covered by the Reference Model

Despite the commonalities in the goal of the Reference Model and the 5S, the proposed approaches contain some differences. The main differences are:

[15] Figure II.4-2 is extracted from [95].

- The Ss are very general-purpose constructs and may therefore be less immediate than the pragmatic approach proposed in the Digital Library Reference Model. Moreover, Gonçalves *et al*. have focused on identifying the 'minimal digital library' with the aim of formalising its aspects, while the Reference Model focuses on identifying the main concepts and relationships characterising the whole universe, considering formalisation as a future step;
- Differing from the 5S, the DELOS Reference Model explicitly accommodates the need to provide different perspectives of the same entity, i.e. the Digital Library, because different users have diverse perceptions of this complex universe, as stressed in Section II.1.
- By relying on the concept of *space*, Gonçalves *et al*. introduced probability spaces, vector spaces, topological spaces, etc. as first-class citizens. The Reference Model deems such concepts to be too fine grained with respect to the goal of the whole model and decides to leave them out.
- The 5S modelling of services, the counterpart of the Reference Model's *Software Components* and *Running Components*, is realised in terms of *scenarios* and thus focused on the description of their behaviour. Moreover, service-to-service cooperation is modelled through the *structure* concept but no specific instantiations are provided. The Reference Model activity plans to produce specific documents dedicated to these fundamental aspects, the *Reference Architecture* and the *Concrete Architecture* (cf. Section II.1)

Besides these differences, it is also important to note the similarity arising around the notion of *Information Object*, termed *digital object* in the 5S framework. This probably indicates that the information object concept has been investigated more and is probably better understood than other elements constituting the Digital Library universe.

II.4.3 The DELOS Classification and Evaluation Scheme

The DELOS Working Group dealing with the *evaluation of digital libraries* problem proposed a model [86][87] that is broader in scope than the one usually adopted in the evaluation context. The aim is to be able to satisfy the needs of all DL researchers, either from the research community or from the library community.

This group started from a general-purpose definition of Digital Library and identified three non-orthogonal components within this digital library domain: the *users*, the *data/collection* and the chosen *system/technology*. These entities are related and constrained by means of a series of relationships, namely:

(1) the definition of the set of users predefines the range and content of the collection relevant and appropriate for them;

(2) the nature of the collection predefines the range of technologies that can be used; and

(3) the attractiveness of the collection content with respect to the user needs and the ease of use of the technologies by these users determine 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 [204], 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 three aspects, techniques and principles for producing quantitative data and implementing their evaluation have been introduced.

The Reference Model addresses similar issues through the *Quality* domain (cf. Section II.2.6). While the evaluation framework takes care of identifying the characteristics of the DL systems to be measured and evaluated, the Digital Library Reference Model introduces this notion at the general level of *Resource*, i.e. each *Resource* is potentially subject to various judgement processes capturing different perspectives.

II.4.4 DOLCE-based Ontologies for Large Software Systems

DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) is a foundational ontology developed to capture the ontological categories underlying natural language and human common sense. By relying on the basic constructs it identifies, a framework of a set of ontologies for modelling modularisation and communication in Large Software Systems has been developed [171].

This framework consist of three ontologies:

(1) the Core Software Ontology (CSO);

(2) the Core Ontology of Software Components (COSC); and

(3) the Core Ontology of Web Services (COWS).

The first of these provides foundations for describing software in general. In particular, it introduces the notions of 'Software' and 'ComputationalObject', which represent respectively the encoding of an algorithm and the realisation of a code in a concrete hardware. These notions are similar to the *Software Component* and *Running Component* notions envisaged by the Reference Model. In addition, the CSO ontology introduces concepts borrowed from the object-oriented paradigm such as 'Class', 'Method' and 'Exception', which from the Reference Model point of view are considered fine-grained and relegated to *Concrete Architecture* models. This ontology contains also the concepts for dealing with access rights and policies. In particular, by relying on the 'Descriptions & Situations' constructs of the DOLCE ontology, the concepts of 'PolicySubjects' (which can be 'Users' or 'UserGroups'), 'PolicyObjects' (which can be 'Data') and 'TaskCollections' (set of 'ComputationalTasks') are introduced. The former two aspects are captured in a general manner by the Reference Model through the relationship between the *Resource* and the *Policy* concepts, i.e. <*regulatedBy*>, and through the concept of *Role* (and *Resource Set*) with respect to the intuition behind 'TaskCollections'.

The Core Ontology of Software Components provides concepts needed to capture software components related aspects like libraries and licenses, component profiles and component taxonomies. The notion of 'SoftwareComponent' (having a 'Profile' aggregating knowledge about it) is the main entity in this ontology and it is formalised as a 'Class' that conforms to a 'FrameworkSpecification' (a set of 'Interfaces'). Moreover, the notion of 'SoftwareLibrary' and 'License' completes the scenario by introducing notions for supporting the automatic check of conflicting libraries and incompatible licenses. The similarities with the set of concepts captured by the Reference Model Architecture Domain (cf. Section II.2.7) are evident. However, it is important to notice that the way the dependencies between the various components are captured by the Reference Model enables it to be more flexible with respect to this point.

The *Core Ontology of Web Services* reuses all the other ones to establish a well-founded ontology for Web Services. This is a very specific ontology that captures the component-oriented approach in terms of standards for protocols (SOAP) and descriptions (WSDL). The other interesting feature is the explicit introduction of the 'QualityOfService' parameters, which in the case of the Reference Model are captured through the general relationship, i.e. <*hasQuality*>, between a *Resource* and its *Quality Parameters*.

II.5 Reference Model in a Nutshell: Concluding Remarks

This part of the volume has provided an overview of the DELOS DL Reference Model by presenting the principles governing the identification and organisation of its constituent elements. It has also described the core concepts and relationships that collectively capture the intrinsic nature of the Digital Library universe. This conceptual framework can be exploited for coordinating approaches, solutions and systems development in the digital library area. In particular, we envisage that in the future Digital Library 'systems' will be described, classified and measured according to the key elements introduced by this model.

The presentation has been logically divided into seven sections, each of which illustrates the concepts and relationships pertaining to one of the core aspects that characterise the digital library systems. Concept maps have been used to represent the concepts and their relations graphically. From the analysis of these maps it clearly emerges that, despite the complexity of some of the aspects illustrated, in most cases a few powerful concepts and relations are sufficient to capture the essential features.

This DELOS Reference Model in a Nutshell can be seen as the introductory part of the larger document implementing the Reference Model, which also presents the definitions, motivations and examples of the concepts and relationships presented so far. This complementary part is contained in PART III The DELOS Digital Library Reference Model Concepts and Relations.

PART III The DELOS Digital Library Reference Model Concepts and Relations

III.1 Introduction

As already stated, a Reference Model is a conceptual framework aimed at capturing significant entities and their relationships in a certain universe with the goal of developing more concrete models of it. The previous sections have outlined the motivation for the creation of the DL Reference Model, as well as providing an upper-level description of its constituents. Conceptual Maps of the Reference Model Domains have been presented and described, providing a brief overview of the concepts of each Domain, the relations that bind them as well as the interaction between concepts of different domains.

This part of the volume delves more deeply into the Reference Model's constituent parts. Concepts and relations are presented in a hierarchical fashion, thus providing an overview of the specialisation relations between them. Concept and relation definitions are provided for each of the concepts and relations of the concept maps.

Each concept definition contains a brief definition of the concept, its relations to other concepts, the rationale behind the addition of the concept and an example. Each relation, accordingly, is described by a definition, a rationale and an example.

III.2 Concepts' Hierarchy

This section presents a more formal description of the model in terms of a hierarchy of classes corresponding to the high-level concepts of the current model. This hierarchy does not include the *Domain* concepts that characterise the Digital Library universe. These are kinds of modules that have been introduced as a way of structuring the model into easily understandable units.

C1 Resource
. C2 Resource Identifier
. C3 Resource Set
. . C4 Result Set (also <isa> Information Object)
. . C15 Collection
. . C20 Group (also <isa> Actor)
. C5 Resource Format
. C16 Query
. C17 Ontology

. [Content Resource][16]
. . C7 Information Object
. . . [Information Object by level]
. . . . C8 Edition (see <hasEdition> relation)
. . . . C9 View (see <hasView> relation)
. . . . C10 Manifestation (see <hasManifestation> relation)
. . . [Information Object by relationship]
. . . . C11 Metadata (see <hasMetadata> relation)
. . . . . C12 Actor Profile
. . . . . C13 Component Profile
. . . . C14 Annotation
. . . C15 Collection
. . . C4 Result Set (also <isa> Resource Set)

. [User Resource]
. . C19 Actor
. . . C20 Group (also <isa> Resource Set)
. . . . C21 Community
. . C22 Role
. . . C23 End-user
. . . . C24 Content Consumer
. . . . C25 Content Creator
. . . . C26 Librarian
. . . C27 DL Designer
. . . C28 DL System Administrator
. . . C29 DL Application Developer
. . C12 Actor Profile (also <isa> Metadata)

[16] 'Classifiers', i.e. items added to the hierarchy for organisational purposes, are indicated [in square brackets].

. [Functionality Resource]
. . C31 Function
. . . C32 Access Resource
. . . . C33 Discover
. . . . . C34 Browse
. . . . . C35 Search
. . . . C36 Acquire
. . . . C37 Visualise
. . . C38 Manage Resource
. . . . C39 Create
. . . . C40 Submit
. . . . C41 Withdraw
. . . . C42 Update
. . . . C43 Validate
. . . . C44 Annotate
. . . . C45 Manage Information Object
. . . . . C46 Disseminate
. . . . . C47 Publish
. . . . . C48 Author
. . . . . C49 Compose
. . . . . C50 Process
. . . . . C51 Analyse
. . . . . C52 Linguistic Analysis
. . . . . C53 Qualitative Analysis
. . . . . C54 Examine Preservation State
. . . . . C55 Statistical Analysis
. . . . . C56 Scientific Analysis
. . . . . C57 Create Structured Representation
. . . . . C58 Compare
. . . . . C59 Transform
. . . . . C60 Physically Convert
. . . . . C61 Translate
. . . . . C62 Convert to a Different Format
. . . . . C63 Extract
. . . . C64 Manage Actor
. . . . . C65 Establish Actor
. . . . . C66 Register
. . . . . C67 Sign Up
. . . . . C68 Login
. . . . . C69 Personalise
. . . . . C70 Apply Profile
. . . . C71 Manage Function
. . . . C72 Manage Policy
. . . . C73 Manage Quality Parameter
. . . C74 Collaborate
. . . . C75 Exchange Information
. . . . C76 Converse
. . . . C77 Find Collaborator
. . . . C78 Author Collaboratively
. . . C79 Manage DL

. . . . C80 Manage Content
. . . . . C81 Manage Collection
. . . . . . C82 Import Collection
. . . . . . C83 Export Collection
. . . . . C84 Preserve
. . . . C85 Manage User
. . . . . C86 Manage Membership
. . . . . C87 Manage Group
. . . . . C88 Manage Role
. . . . . C89 Manage Actor Profile
. . . . C90 Manage Functionality
. . . . . C91 Monitor Usage
. . . . C92 Manage Quality
. . . . C93 Manage Policy Domain
. . . C94 Manage & Configure DLS
. . . . C95 Manage DLS
. . . . . C96 Create DLS
. . . . . C97 Withdraw DLS
. . . . . C98 Update DLS
. . . . . C99 Manage Architecture
. . . . . C100 Manage Architectural Component
. . . . . C101 Configure Architectural Component
. . . . . C102 Deploy Architectural Component
. . . . . C103 Monitor Architectural Component
. . . . C104 Configure DLS
. . . . . C105 Configure Resource Format
. . . . . C106 Configure Content
. . . . . C107 Configure User
. . . . . C108 Configure Functionality
. . . . . C109 Configure Policy
. . . . . C110 Configure Quality

. [Policy Resource]
. . C112 Policy
. . . [Policy by characteristic]
. . . . [Policy by context]
. . . . . C113 Extrinsic Policy
. . . . . C114 Intrinsic Policy
. . . . [Policy by expression]
. . . . . C115 Explicit Policy
. . . . . C116 Implicit Policy
. . . . [Policy by application]
. . . . . C117 Prescriptive Policy
. . . . . C118 Descriptive Policy
. . . . [Policy by compliance]
. . . . . C119 Enforced Policy
. . . . . C120 Voluntary Policy
. . . [Policy by scope]
. . . . C121 System Policy
. . . . . C122 Change Management Policy

. . . . . C123 Resource Management Policy
. . . . . C124 Support Policy
. . . . . C125 Connectivity Policy
. . . . . C126 Risk Management Policy
. . . . C127 Content Policy
. . . . . C128 Disposal Policy
. . . . . C129 Collection Development Policy
. . . . . C130 Collection Delivery Policy
. . . . . C131 Submission and Resubmission Policy
. . . . . C133 Digital Rights
. . . . . C135 Preservation Policy
. . . . . C132 Digital Rights Management Policy
. . . . . C134 License
. . . . C136 User Policy
. . . . . C137 User Management Policy
. . . . . C138 Registration Policy
. . . . . C139 Personalisation Policy
. . . . . C140 Privacy and Confidentiality Policy
. . . . . C141 Acceptable User Behaviour Policy
. . . . C142 Functionality Policy
. . . . . C143 Access Policy
. . . . . C144 Charging Policy
. . . . . C145 Security Policy

. [Quality Resource]
. . C147 Measure
. . . C148 Objective Measure
. . . C149 Subjective Measure
. . . C150 Qualitative Measure
. . . C151 Quantitative Measure
. . C152 Measurement
. . C153 Quality Parameter
. . . C154 Generic Quality Parameter
. . . . C155 Economic Convenience
. . . . C156 Interoperability Support
. . . . C157 Reputation
. . . . C158 Security Enforcement
. . . . C159 Sustainability
. . . . C160 Documentation Coverage
. . . . C161 Performance
. . . . C162 Scalability
. . . C163 Content Quality Parameter
. . . . C164 Authenticity
. . . . C165 Trustworthiness
. . . . C166 Freshness
. . . . C167 Integrity
. . . . C168 Preservation Performance
. . . . C169 Provenance
. . . . C170 Scope
. . . . C171 Size

. . . . C172 Fidelity
. . . . C173 Perceivability
. . . . C174 Viability
. . . . C175 Metadata Evaluation
. . . C176 Functionality Quality Parameter
. . . . C177 Availability
. . . . C178 Awareness of Service
. . . . C179 Capacity
. . . . C180 Expectations of Service
. . . . C181 Fault Management Performance
. . . . C182 Impact of Service
. . . . C183 Orthogonality
. . . . C184 Dependability
. . . . C185 Robustness
. . . . C186 Usability
. . . . C187 User Satisfaction
. . . C188 User Quality Parameter
. . . . C189 User Activeness
. . . . C190 User Behaviour
. . . C191 Policy Quality Parameter
. . . . C192 Policy Consistency
. . . . C193 Policy Precision
. . . C194 Architecture Quality Parameter
. . . . C195 Ease of Administration
. . . . C196 Compliance with Standards
. . . . C197 Ease of Installation
. . . . C198 Load Balancing Performance
. . . . C199 Log Quality
. . . . C200 Maintenance Performance
. . . . C201 Redundancy

. [Architectural Resource]
. . C203 Architectural Component
. . . C204 Software Architecture Component
. . . . C205 Software Component
. . . . . C206 Application Framework
. . . . C207 Interface
. . . . C208 Framework Specification
. . . C209 System Architecture Component
. . . . C210 Running Component
. . . . C211 Hosting Node
. . C13 Component Profile (also <isa> Metadata)
. . C134 License (also <isa> Policy)
. . C212 Software Architecture
. . C213 System Architecture

III.3 Reference Model Concepts' Definitions

C1 Resource

Definition: An identifiable entity in the *Digital Library* universe.

Relationships:

- *Resource* must have at least one unique *Resource Identifier* (<*identifiedBy*>)
- *Resource* <*hasPart*> *Resource*
- *Resource* is <*associatedTo*> *Resource* for a certain *Purpose*
- *Resource* <*hasFormat*> *Resource Format*
- *Resource* <*hasMetadata*> *Information Object*
- *Resource* <*hasAnnotation*> *Information Object* to a certain *Region*
- *Resource* may be regulated by (<*regulatedBy*>) *Policy*
- *Resource* may have (<*hasQuality*>) *Quality Parameter*

Rationale: In the Digital Library universe there are entities belonging to diverse and heterogeneous areas and systems that share common modelling attributes and principles supporting their management. These heterogeneous entities are grouped under the concept of *Resource*, as it is defined in the context of Web architecture. The Web is intended as an information space in which the items, referred to as resources, are identified by a unique and global identifier called Uniform Resource Identifier (URI). The Resource Model presented here starts from Web architecture and adds domain-specific aspects needed to accommodate digital library requirements. Thus the model allows for the use of Web standards, technologies and implementations.

The *Resource* concept is abstract, in the sense that it cannot be instantiated directly but only through the instantiation of one of its specialisations.

Examples:

- *Information Object;*
- *Actor;*
- *Function;*
- *Policy;*
- *Ontology.*

C2 Resource Identifier

Definition: A token bound to a *Resource* that distinguishes it from all other *Resources* within a certain scope, which includes the *Digital Library*.

Relationships:

- *Resource* is <*identifiedBy*> *Resource Identifier*

Rationale: Various types of resource identifiers have been proposed, from simple sequential numbers to tokens drawn from more sophisticated schemes, designed to function across *DLs* and time (time is particularly important for preservation purposes). Such persistent identification schemes include URIs, IRIs, ARKs, Digital Object Identifiers (DOIs) and persistent handles. Clearly, each of these has a different discriminating power when considered in the context of digital libraries.

Selecting a Resource Identifying scheme implies a trade-off. Usually, the wider the scope of the scheme, the more costly it is to set up and maintain the scheme. Ideally, the scheme having the widest scope within the acceptable cost range should be selected.

Examples:

- Uniform Resource Identifiers (URIs)[17];
- Internationalized Resource Identifiers (IRIs)[18];
- Archival Resource Keys (ARKs)[19];
- Digital Object Identifier (DOI)[20];
- Persistent handles.

C3 Resource Set

Definition: A set of *Resources*, which is in turn a *Resource*, often defined for some management or application purpose.

Relationships:

- *Resource Set <isa> Resource*
- *Resource <belongTo> Resource Set*

Rationale: The grouping of *Resources* is required in many operations of a Digital Library. For instance, in the *Content Domain, Collections* are *Resource Sets*, as are search results (*Result Set*) or a subset of the search results marked by an *Actor*. In the *User Domain*, *Groups* are *Resource Sets.*

Examples:

- The set of *Collections*, *Functions* and *Actors* forming a 'virtual research environment', i.e. the set of *Resources* grouped to serve a research need.

C4 Result Set

Definition: A *Resource Set* whose constituent *Resources* are the result of a *Query* execution.

Relationships:

- *Result Set <isa> Resource Set*

Rationale: A set of *Resources* returned by the system as a consequence of an *Actor* that issues a *Query*. *Result Set* is a group of *Resources* that are highly dynamic and time dependent, i.e. different *Result Sets* can be obtained by issuing the same *Query* in different time periods. This is a consequence of the changes in the *Information Objects* and *Collections* made available in the system.

Examples:

- The set of *Information Objects* representing Picasso outcomes retrieved by a *Query.*

[17] http://tools.ietf.org/html/rfc3986

[18] http://www.w3.org/International/O-URL-and-ident.html

[19] http://www.cdlib.org/inside/diglib/ark/

[20] http://www.doi.org/

C5 Resource Format

Definition: A description of the structure of a *Resource*. May build explicitly on an *Ontology* or imply an *Ontology*.

Relationships:

- *Resource <hasFormat> Resource Format*
- *Resource Format* is *<expressionOf> Ontology*

Rationale: The schema defines the properties and attributes of a resource and assigns a name to this kind of structure. The resource schema of information objects (a kind of resource) gives the structural composition of the object; for instance, the objects stored in a collection of PhD theses might share a common format called 'thesis', defined as an aggregation of multiple parts: the cover page, the preface, a sequence of chapters, images, audio files and supporting evidence in the form of data stored in a database. For other types of resources, such as users or policies, the schema describes the set of properties or attributes by which the resources are modelled.

We do not make any recommendation as to what form a schema should take, or which schema works best as 'the' schema for a specific kind of *Resource*. From a practical point of view, this leaves room for one of two options: (1) either the developers of a digital library choose some schemas and make them part of the digital library conceptual model; or (2) they leave open the possibility of 'plugging in' any schema, in which case a suitable meta-model must be selected for each resource type in order to express the various resource schemas handled by the system; for instance, JCR is a suitable meta-model for information objects.

Examples:

- OOXML is a *Resource Format* for electronic document *Resources*;
- MPEG-21 is a *Resource Format* for multimedia *Resources*.

C6 Content Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents the various aspects related to the modelling of information managed in the Digital Library universe to serve the information needs of the *Actors*.

Relationships:

- *Digital Library <definedBy> Content Domain*
- *Digital Library System <definedBy> Content Domain*
- *Digital Library Management System <definedBy> Content Domain*
- *Content Domain <consistOf> Information Object*
- *Content Domain <organisedIn> Collection*

Rationale: The Content concept represents the information that *Digital Libraries* handle and make available to their *Actors*. It is composed of a set of *Information Objects* organised in *Collections*. *Content Domain* is an umbrella concept that is used to aggregate all forms of information that a *Digital Library* may require in order to offer its services. *Metadata* play an important role in the *Content Domain* because they describe a clearly defined category of *Information Objects* in the domain of discourse.

Examples: --

C7 Information Object

Definition: The main *Resource* of the *Content Domain*. An *Information Object* is a *Resource* identified by a *Resource Identifier*. It must belong to at least one *Collection*. It may have *Metadata*, *Annotations* and multiple *Editions*, *Views*, *Manifestations*. In addition, it may have *Quality Parameters* and *Policies*.

Relationships:

- *Information Object <isa> Resource*
- *Information Object <hasFormat> Resource Format* (inherited from *Resource*)
- *Information Object* is *<identifiedBy> Resource Identifier* (inherited from *Resource*)
- *Information Object <belongTo> Collection*
- *Information Object <hasMetadata> Information Object* (*Metadata*)
- *Information Object <hasAnnotation> Information Object* (*Annotation*)
- *Information Object <hasEdition> Information Object*
- *Information Object <hasView> Information Object*
- *Information Object <hasManifestation> Information Object*
- *Information Object <hasQuality> Quality Parameter*
- *Information Object* is *<regulatedBy> Policy*

Rationale: The notion of *Information Object* represents the main entity populating the *Content Domain*.

An *Information Object* can be a complex, multimedia and multi-type object with parts, such as a sound recording associated with a set of slides, a music score, political and economic data associated with interactive simulations, a PhD thesis which includes a representation of a performance, or a simulation experiment and the experimental data set adopted, or a data stream representing the pool of data continuously measured by a sensor. This information is given in the *Resource Format* linked to the *Information Object* via a *<hasFormat>* relationship. Thanks to this relationship the mechanism identifying the boundaries and the structure of each *Information Object* is particularly flexible and powerful. For instance, it is possible to have a huge *Information Object* representing a soccer game, composed of 27 parts each representing the soccer game gathered by a different camera. Another way to organise the same soccer game *Information Object* is to have a *Collection* of *Information Objects*, one for each of the highlights of the match; each of these *Information Objects* can be further broken down into parts, each representing the highlight as captured by a different camera, etc. Moreover, the notions of *Edition*, *View* and *Manifestation* represent yet another way of modelling *Information Objects* according to the semantics fixed by the IFLA-FRBR model [116]. This model is particularly useful in dealing with 'document' *Information Objects* but can be extended and applied to advantage to any kind of *Information Object*, e.g. the various *Editions* (usually termed versions) of a software product or a data set.

The *Information Object* concept is also part of the CIDOC-CRM [199], where it is used to refer to 'identifiable immaterial items, such as poems, jokes, data sets, images, texts, multimedia objects, procedural prescriptions, computer program code, algorithm or mathematical formulae, that have an objectively recognisable structure and are documented as single units'.

The notion of *Information Object* is a complex one, and can be used to capture different concepts. It certainly complies with the notion of 'work' in the IFLA-FRBR model, but also

with the more concrete notions of *Edition*, *View* and *Manifestation*, also part of the IFLA-FRBR model.

Examples:

- The electronic version of this volume along with its *Metadata*.

C8 Edition

Definition: The *Information Object* representing the realisation along the time dimension of another *Information Object* to which it is related via a <*hasEdition*> relationship.

Relationships:

- *Edition <isa> Information Object*
- *Information Object <hasEdition> Information Object*

Rationale: *Editions* represent the different states of an *Information Object* during its lifetime. From a technical point of view, they are defined similarly to *Metadata* or *Annotations*, i.e. as derived concepts from a relation, in this case <*hasView*>.

An *Edition* is an *Information Object* and thus a *Resource*, therefore it is independent of the *Information Object* of which it is an edition.

Examples:

- An *Information Object* representing a study may be linked to the following *Information Objects* via <*hasEdition*> relationships:
  - its draft version is an *Edition;*
  - the version submitted is an *Edition;*
  - the version published in the conference proceedings with colour images is an *Edition.*

C9 View

Definition: An *Information Object* representing a different expression of another *Information Object*, to which it is related via a <*hasView*> relation.

Relationships:

- *View <isa> Information Object*
- *Information Object <hasView> Information Object*

Rationale: This entity represents a view of an *Information Object*. The concept responds to the diversity of expressions of the same object that are instantiated using different digital technologies. *Views* do not represent different physical aspects; rather they are mechanisms to differentiate types of representations or visualisations that can be given to the *Information Objects*. The concept of *View* fits very well with those used in the DBMS; in this context a view is a virtual or logical table (i.e. the organisational unit of data) composed as the result of a query over the actual data stored in potentially different tables and different ways in order to provide a new organisational unit presenting data in a more useful way.

Edition and *View* together capture the expression concept of the IFLA-FRBR model [116].

Examples:

- An example of View that can be envisaged over the same *Information Object* representing a data stream of an environmental sensor consists of its raw form as a series of numerical values or as a graph representing the evolution of the values measured by the sensor over time.

- Another example might consider an *Information Object* representing the outcomes of a workshop; three different views of this object can be envisaged:
  - the 'full view' containing a preface prepared by the conference chair and the whole set of papers accepted and organised thematically;
  - the 'handbook view' containing the conference programme and the slides of each lecturer accompanied by the abstract of the papers organised per session; and
  - the 'informative view' reporting the goal of the workshop and the title list of the accepted papers together with the associated abstract.

C10 Manifestation

Definition: An *Information Object* representing the physical embodiment of another *Information Object*, to which it is related via a <*hasManifestation*> relationship.

Relationships:

- *Manifestation <isa> Information Object*
- *Information Object <hasManifestation> Information Object*

Rationale: Like *Editions* and *Views*, *Manifestations* are derived from a relation (<*hasManifestation*>). However, while the *Editions* and *Views* deal with the intellectual and logical organisation of *Information Objects*, *Manifestations* deal with their physical presentation. Another important difference is that *Manifestations* may, transparently to the *Actor*, be dynamically generated through a possibly complex process, taking into account *Actor* preferences, templates, size restrictions and other factors.

Examples:

- Examples of manifestations are the pdf file or the Microsoft Word file of the same paper, the MPEG file containing the video recording of a lecture, a file containing the raw data observed by a sensor, an XML file reporting the results of a certain elaboration, or the audio representation of a text that can be used for providing access to the object for visually impaired users.

C11 Metadata

Definition: Any *Information Object* that is connected to one or more *Resources* through a <*hasMetadata*> relationship.

Relationships:

- *Metadata <isa> Information Object*
- *Resource <hasMetadata> Information Object* (*Metadata*)
- *Information Object <hasMetadata> Information Object* (*Metadata*)
- *Metadata <hasFormat> Resource Format* that is an *<expressionOf> Ontology* (inherited by *Resource*)
- *Actor Profile <isa> Metadata*
- *Policy Metadata <isa> Metadata*
- *Component Profile <isa> Metadata*

Rationale: The 'classic' definition of metadata is 'data about data'. However, it depends from the context whether an object is or is not metadata. This is the main motivation leading to their modelling as a derived notion from the instances of the <*hasMetadata*> relation.

Metadata are used for describing different aspects of data, such as the semantics, provenance, constraints, parameters, content, quality, condition and other characteristics. These data can be used in different contexts and for a diversity of purposes; usually, they are associated with an *Information Object* (more generally to a *Resource* through the <*hasMetadata*>) as a means of facilitating the effective discovery, retrieval, use and management of the object.

There are a number of schemes for classifying metadata.

One of them consists in classifying metadata according to the specific role they play:

- Descriptive metadata, i.e. metadata that provide a mechanism for representing attributes describing and identifying the *Resource*. Examples include bibliographical attributes (e.g. creator, title, publisher, date), format, list of keywords characterising the contents. The term 'descriptive' is used here in a consistent but broader sense than in 'descriptive cataloguing'.
- Administrative metadata, i.e. metadata for managing a *Resource*. This category of metadata may include metadata detailing: (*i*) technical characteristics of the Resource; (*ii*) the history of the operations performed on the Resource since its creation/ingest; (*iii*) means of access; and (*iv*) how the authenticity and integrity of the Resource can be verified.
- Preservation metadata, i.e. metadata designed to support the long-term accessibility of a Resource by providing information about its content, technical attributes, dependencies, management, designated community(ies) and change history. Preservation metadata have been identified as essential for the long-term management of digital objects. The Reference Model for an Open Archival Information System (OAIS) [53] provides an excellent overview of the role of preservation metadata in the management over time of digital resources. PREservation Metadata: Implementation Strategies Working Group, commonly referred to as PREMIS, has defined a core set of preservation metadata elements that would provide support for the management of digital objects across systems and time. They acknowledged that, while they had identified the key aspects of the necessary preservation metadata, there was room for more work in the area of technical metadata and this might be necessary at the level of each DL *Resource* or *Collection*. 'Preservation metadata' encompasses technical elements necessary to enable access to, manipulation and/or rendering of a DL *Resource*, data about the structure and syntax of an *Information Object*, information to support semantic understanding of *Resources* and details of the responsibilities and rights governing the application of preservation actions to *Resources*.

Another scheme classifies metadata according to what kind of *Resource* feature they present:

- Syntactic metadata present data about the syntax or structure of the resource. They provide data such as time or space of *Resource* creation or inclusion into the *Digital Library*, or size of *Resource*. Inherent metadata can be obtained from the *Resource* itself. They are dependent on the *Manifestation* used.
- Semantic metadata provide data about the *Resource* itself (the semantics of the *Resource*). These metadata could be explicitly or implicitly derived from the *Resource*, or generated by humans.

- Contextual metadata provide data related neither to the structure nor to the semantics of a *Resource* but to other issues within the context of the DL. They might be needed to understand the *Resource* or the ways of its possible use.

Other examples of classification criteria are: by purpose (for search or wider use), by fluidity (static or dynamic) or by mode of generation (human or automatic).

All the above-mentioned classifications are orthogonal, i.e. they are not mutually exclusive, in the sense that metadata may fall into more than one of the identified categories. For instance, the metadata describing the creator of a DL resource can be used for discovering the resource, for managing its digital rights or for authentication purposes.

Examples:

- Keywords are Metadata because they represent the content of a *Resource*.

C12 Actor Profile

Definition: An *Information Object* that models any external entity (*Actor*) that interacts with the Digital Library. It is identified by a *Resource Identifier*. An *Actor Profile* may belong to a distinct *Actor* or it may model more than one *Actor*, i.e. a *Group* or a *Community*.

Relationships:

- *Actor Profile* <isa> *Information Object*
- *Actor Profile* is <*identifiedBy*> *Resource Identifier* (inherited from *Resource*)
- *Actor* is <*modelledBy*> *Actor Profile*
- *Function* is <*influencedBy*> *Actor Profile*

Rationale: An *Actor Profile* is an *Information Object* that models an *Actor* by potentially capturing a large variety of the *Actor*'s characteristics, which may be important for a particular Digital Library for allowing the *Actor* to use the 'system' and interact with it as well as with other *Actors*. It not only serves as a representation of *Actor* in the system but essentially determines *Policies* and *Roles* that govern which *Functions* an *Actor* is entitled to perform through the *Actor*'s lifetime and how these functions should behave when exploited by him (<*influencedBy*>). For example, a particular instance of *Actor* may be entitled to *Search* within particular *Collections* and *Collaborate* with certain other *Actors*. The characteristics captured in an *Actor Profile* vary depending on the type of *Actor*, i.e. human or non-human, and may include: identity information (e.g. age, residence or location for humans and operating system, web server edition for software components), educational information (e.g. highest degree achieved, field of study – only for humans) and preferences (e.g. topics of interest, pertinent for both human and software *Actors* that interact with the *Digital Library*).

Examples:

- Group Profile, i.e. the *Actor profile* capturing the characteristics of a *Group* as a single entity.
- Community Profile, i.e. the *Actor profile* capturing the characteristics of a *Community* as a single entity.

C13 Component Profile

Definition: The *Metadata* attached to an *Architectural Component*.

Relationships:

- *Component Profile* <*isa*> *Metadata*
- *Component Profile* <*isa*> *Information Object* (inherited from *Metadata*)

- *Architectural Component <hasProfile> Component Profile*
- *Component Profile <profile> Policy*
- *Component Profile <profile> Quality Parameter*
- *Component Profile <profile> Function*
- *Component Profile <profile> Interface*

Rationale: The *Component Profile* is a specialisation of the *Metadata* objects and plays exactly the same role, i.e. provides additional information for management purposes. Neither statements nor constraints are imposed on the *Component Profile* associated with each *Architectural Component*. However, it is envisaged that this additional information should deal with the *Interfaces* the component has, the *Quality Parameters* it has, the *Policies* regulating it, and the *Functions* it yields.

Examples:

- The WSDL document (http://www.w3.org/TR/wsdl) describing a Web Service.

C14 Annotation

Definition: An *Annotation* is any kind of super-structural *Information Object* including notes, structured comments, or links, that an *Actor* may associate with a *Region* of a *Resource* via the *<hasAnnotation>* relation, in order to add an interpretative value. An annotation must be identified by a *Resource Identifier*, be authored by an *Actor*, and may be shared with *Groups* according to *Policies* regulating it (*Resource* is *<regulatedBy> Policy*). An *Annotation* may relate a *Resource* to one or more other *Resources* via the appropriate *<hasAnnotation>* relationship.

Relationships:

- *Annotation <isa> Information Object*
- *Annotation* is *<identifiedBy> Resource Identifier*
- *Resource <hasAnnotation> Information Object* about a *Region*

Rationale: *Annotations* can support cooperative work by allowing *Actors* to merge their intellectual work with the DL *Resources* provided by the DL to constitute a single working context. *Annotations* 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 Object*;
- to add personal notes about a retrieved *Information Object* for future use.

Annotations are not only a way of explaining and enriching a DL *Resource* with personal observations but also a means of transmitting and sharing ideas in order to improve collaborative work practices. Thus, *Annotations* can be geared not only to the way of working of the individual and to a method of study but also to a way of doing research, as happens in the Humanities.

As *Annotations* are *Information Objects*, they may be in different formats, be expressed in different media, be associated with *metadata*, and can themselves be annotated. At present, in the literature there is an ongoing discussion as to whether *Annotations* should be considered as *Metadata* or as *Information Objects*. For the time being, an *Annotation* is modelled as an *Information Object* because (*i*) it has been considered as additional information that increases the existing content by providing an additional layer of elucidation and explanation, and (*ii*) because of this the *Annotation* itself takes the shape of an additional *Information Object* that

can help people understand the annotated *Resource*. In fact, the status of *Annotation* is derived from the <*hasAnnotation*> relation linking *Resources*; this choice settles the long-standing issue of whether *Annotations* should be considered as *Information Objects* or as *Metadata*.

A final observation relates to the evolving nature of the *Information Objects* and of the *Resources* in general, which may result in invalidating a previously expressed *Annotation*. Usually each update results in creating a new *Edition*; thus, it is sufficient to link the *Annotation* to the appropriate version to which it refers.

Examples:

- The commentary texts accompanying each Reference in an Annotated Bibliography.

C15 Collection

Definition: A content *Resource Set*. The 'extension' of a collection consists of the *Information Objects* it contains. A *Collection* may be defined by a membership criterion, which is the 'intension' of the collection.

Relationships:

- *Collection <isa> Resource Set*
- *Collection <isa> Resource*
- *Information Object <belongTo> Collection*
- *Collection <hasIntension> Query*
- *Collection <hasExtension> Resource Set* (set of *Information Object*)

Rationale: *Collections* represent the classic mechanism to organise *Information Objects* and to provide focused views of the *Digital Library Information Object Resource Set*. These focused views enable *Actors* to access thematic parts of the whole; they can be created by *Librarians* to keep the set of *Information Objects* organised and to improve its access and usage; further, they can be created by authorised *Content Consumers* to implement their own personal views of the *Digital Library Information Object Resource Set*.

The definition and identification of the *Information Objects* constituting a *Collection* (the collection extension) is based on a characterisation criterion (the collection intension). These criteria can range from an enumeration of the extension to conditions that specify which properties information objects must satisfy in order to be collection members (truth conditions).

Typically, *Collections* are hierarchically structured in sub-collections, but for general purposes we do not include this structuring in the present model.

Examples:

- The set of items exported through the set mechanism of the OAI-PMH protocol;
- The set of *Information Objects* characterised by having 'Leonardo da Vinci' as author and contained in the user preferred Digital Library at the time he/she access to that collection.

C16 Query

Definition: A characterisation criterion capturing the common traits of the *Resources* forming a *Resource Set*.

Relationships:

- *Query <isa> Information Object*

Rationale: The notion of query is well known in the DB area where it indicates an expression issued according to a query language, e.g. SQL, to obtain the data stored in the DB. Digital Libraries, as well as other Information Retrieval systems, borrowed this term to represent the information need of their users. In the case of Digital Libraries, queries can be expressed according to various query languages ranging from keyword-based to fielded forms.

The notion of *Query* is fundamental to the *Search* Function. However, it can be used for other purposes. This reference model uses it to capture the intension (<*hasIntension*>) definition of a *Collection*.

Examples:

- 'Digital Library' is the representation of a query constituted by two tokens issued by a user interested in retrieving *Resources* dealing with Digital Libraries;
- 'subject=H3.7 Digital Library AND author=Arms' is the representation of a complex and fielded query issued by an *Actor* interested in finding the *Resources* having *metadata* that contain the specified values in the identified fields.

C17 Ontology

Definition: An *ontology* is a formal conceptualisation that defines the terms about a domain. *Ontologies* formalise a shared vocabulary about a domain [101].

Relationships:

- *Ontology* <*isa*> *Information Object*
- *Resource Format* is <*expressionOf*> *Ontology*

Rationale: The notion of *ontology* generalises that of schema or format, as well as related notions, such as thesaurus. *Ontologies* can refer to different aspects of *Information Objects*, such as their structure, their content, their preservation among others. Although a Digital Library might define and adopt its own proprietary formats, it is widely acknowledged that standard representation models (e.g. Dublin Core for descriptive metadata, MPEG for the structure of audio-visual objects, OAIS for preservation) enhance the interoperability and reuse of *Resources*. The emergence of rich schemas, such as CIDOC Conceptual Reference Model (CRM) [199], which enable content owners or holders to define articulated descriptions of their digital assets, and to exploit such descriptions in accessing the information or in managing complex applications around them demands greater flexibility at the level of generalisation. Semantic Web technologies, notably the Web Ontology Language (OWL), which builds upon Description Logics and the associated inferential capabilities, is another driver.

The Reference Model does not make any commitment to a specific Ontology; rather it assumes that the various 'systems', *DL*, *DLS* and *DLMS*, will be able to offer their users the ability to handle multiple ontologies either sequentially or independently. A mechanism to support this could offer:

- an ontology language able to represent any ontology the DL users may want to work with (e.g. OWL);
- an ontology mapping framework, consisting of a language for expressing relations between elements from different ontologies and an associated engine to exploit such mappings in query evaluation.

Examples:

- DOLCE (Descriptive Ontology for Linguistic nd Cognitive Engineering) is a foundational ontology developed to capture the ontological categories underlying natural language and human commonsense.

C18 User Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents the various aspects related to the modelling of external entities, either human or machines, interacting with the Digital Library.

Relationships:

- *Digital Library <definedBy> User Domain*
- *Digital Library System <definedBy> User Domain*
- *Digital Library Management System <definedBy> User Domain*
- *User Domain <consistOf> Actor*

Rationale: The *User Domain* concept represents the *Actors* (whether human or not) entitled to interact with *Digital Libraries*. The aim of *Digital Libraries* is to connect such *Actors* with information (the *Information Objects*) and to support them in consuming already available information and produce new information (through the *Functions*). *User Domain* is an umbrella concept that covers all notions related to the representation and management of *Actor* entities within a *Digital Library*, e.g. the digital entities representing the *Actors*, their rights within the system, and their profiles (*Actor Profile*) exploited to personalise the system's behaviour or to represent these actors in collaborations.

Examples: --

C19 Actor

Definition: A *Resource* that represents an external entity that interacts with the Digital Library and is identified by a *Resource Identifier*. Furthermore, it may have at least one *Actor Profile* and it may belong to at least one *Group* and be regulated by a set of *Policies*. An *Actor* may be characterised by *Quality Parameters* and may be linked to other *Actors*.

Relationships:

- *Actor <isa> Resource*
- *Actor* is *<identifiedBy> Resource Identifier* (inherited from *Resource*)
- *Actor* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Actor <belongTo> Group*
- *Actor* is *<modelledBy> Actor Profile*
- *Actor* is *<associatedWith> Actor*
- *Actor <hasQuality> Quality Parameter*
- *Actor <play> Role*
- *Actor <perform> Function*
- *Group <isa> Actor*

Rationale: An *Actor* captures any entity external to a Digital Library that interacts with it or with other similar entities through the *Functions* offered by the Digital Library and includes humans, and inanimate entities such as software programs or physical instruments. The latter may range from subscription services offered by external systems to portals and other digital libraries that pull content from, or push content to, the particular Digital Library. Although

each distinct entity may be recognised in the system by a single *Resource Identifier*, it may play a different *Role* at different times, belong to more than one *Group* and be associated with more than one *Actor Profile*. For instance, an *Actor* may have one profile when assuming the *Content Creator* role and a different profile under the *Content Consumer* role. *Policies* that are associated with an *Actor*, through an individual or group *Actor Profile*, govern the *Actor*'s interactions with the system and with other *Actors* through their lifetime, e.g. the set of permissible *Functions* for an *Actor*. An *Actor* may be characterised by various *Quality Parameters*. For instance, a human may be distinguished on the basis of *Trustworthiness* and a software agent may be characterised by its *Robustness*. Such quality parameters may be used to guide or value an *Actor*'s interactions. For instance, in actor groupings, such as human cooperations or co-authorships or software component integrations, captured by instantiating the <*associatedWith*> relations, a more authoritative *Actor* can be trusted for sharing content from an *Actor* of disputable quality.

Examples:

- A *Group* is an *Actor*.
- A *Community* is an *Actor*.
- John is an *Actor*.
- A Web Service harvesting the set of *Information Objects* forming a *Collection* in a *Digital Library System* is an *Actor*.

C20 Group

Definition: A *Resource Set* that models a set of external entities with common characteristics and following specific interaction rules and patterns within the Digital Library. It is identified by a *Resource Identifier*. A *Group* can be modelled by an *Actor Profile* that specifies the characteristics of the members of the group. The membership to the *Group* (<*belongTo*>), i.e. the set of *Actors* belonging to it, can be determined by enumerating its members or by capturing the similar traits of the *Actors* in a *Query*. In this second case, membership of the *Group* will be dynamically determined by evaluating the *Query*.

Relationships:

- *Group* <*isa*> *Resource Set*
- *Group* is <*identifiedBy*> *Resource Identifier* (inherited from *Resource*)
- *Group* <*isa*> *Actor*
- *Group* is <*modelledBy*> *Actor Profile* (inherited from *Actor*)
- *Actor* <*belongTo*> *Group*
- *Community* <*isa*> *Group*

Rationale: A *Group* represents an *Actor* population that exhibits cohesiveness to a large degree and can be considered as an *Actor* with its own profile and identifier. A *Group* is described by an *Actor Profile* that essentially specifies explicitly (through enumeration) or implicitly (through a set of desired characteristics) the members of the group, and specifies the *Roles* an *Actor* of the *Group* can take and the *Policies* that govern the *Actor* interactions in the system, such as permissible *Functions* and accessible *Resources*. Members of a *Group* inherit (part of) the characteristics from the *Group* but they may have additional characteristics as described in their individual *Actor*'s profile.

Examples:

- *Community* is an example of *Group*.

- John, Mary and Paul are the *Actors* constituting the *Group* entitled to curate Leonardo da Vinci *Collection* disseminated through their University Digital Library.

C21 Community

Definition: A social *Group*.

Relationships:

- *Community <isa> Group*

Rationale: A *Community* is a particular subclass of *Group*, which refers to a social group of humans with shared interests. In human *Communities*, intent, belief, resources, preferences, needs, risks and a number of other conditions may be present and common, affecting the identity of the participants and their degree of cohesiveness. *Community* is a pre-existing group of people with shared interests, which is online in the Digital Library, or a group that is formed as *Actors* in the Digital Library interact with the Library's contents or with other *Actors*. For instance, in a Digital Library with publications, there may be the *Community* of people interested in Artificial Intelligence and the *Community* of people providing test collections for Information Retrieval algorithms. On the other hand, as a *Group* it is a well-defined user community identified by a specific *Actor Profile* and *Resource Identifier*. The *Profile* records permissible *Roles*, *Functions* and *Resources* according to specific *Policies*.

Examples:

- Scientists joining the Human Genome Organisation (HUGO) are a *Community* involved in human genetics. This community is interested in accessing a Digital Library providing them with the information objects and functions they need to accomplish their mission. Such a Digital Library may also serve other communities by providing them with (part of) the data and functions related to human genetics to promote cross discipline synergies.

C22 Role

Definition: A set of functions within the context of an organisation with some associated semantics regarding the authority and responsibility conferred on the user assigned role.

Relationships:

- *Actor <play> Role*
- *End-User <isa> Role*
- *DL Designer <isa> Role*
- *DL System Administrator <isa> Role*
- *DL Application Developer <isa> Role*

Rationale: The above definition comes from [80] and works in accordance with the policy mechanism pervading the *Policy Domain* (Section II.2.5). A role is a kind of pre-packaged generic profile and may be seen as a packet of statements identifying the kind of *Functions* an *Actor* is eligible to perform within the system. Thus, a role may be stored as a profile that represents an individual or (most likely) a population of users. *Roles* are also called stereotypes in user modelling. An *Actor* can be assigned to a *Role*; this means, the *Actor* inherits all the *Role* statements. Clearly, an *Actor* can play different *Roles* at different times or more than one *Role* at the same time. Apart from the four main *Actor Roles* defined (*End User*, *DL Designer*, *DL System Administrator* and *DL Application Developer*), the following generic *Roles* are distinguished within a DL context and subsequently defined: *Content Consumer*, *Content Creator* and *Librarian*, which are sub-roles of the *End-user* role. Apart

from these roles and sub-roles, which are prototypically defined in the Reference Model, any digital library could, and should, define additional roles. A sub-role may be defined, providing it with some of the *Functions* of a generic *Role*. For example, a content annotator *Role* might be a sub-role of information creator that entitles *Actors* to annotate only existing *Information Objects*.

Examples:

- Student is a typical *Role* in a University Digital Library granting access to specific *Collections* and *Functions*.

C23 End-user

Definition: The *Role* of the *Actors* that access the *Digital Library* to exploit its *Resources* and possibly produce new ones.

Relationships:

- *End-user <isa> Role*

Rationale: *End-users* exploit DL facilities for providing, consuming and managing DL content (usually *Information Objects*, generally *Resources*). This is actually a class of *Actors* further subdivided into the concepts of ***Content Creator***, ***Content Consumer*** and ***Librarian***, each of which usually has a different perspective on the Digital Library. For instance, a *Content Creator* may be a person that creates and inserts their own objects in the Digital Library or an external program that automatically converts artefacts to digital form and uploads them to the Digital Library.

Examples:

- John is an *End-user* in a University Digital Library accessing its *Collections* and *Functions* to prepare its examination. Mary in another *End-user* accessing the same Digital Library to complete its doctoral thesis and once this thesis is discussed publishes it for future uses.

C24 Content Consumer

Definition: The *Role* of the *Actors* that access the Digital Library for the purpose of consuming its *Resources,* usually *Information Objects*, through the available *Functions*.

Relationships:

- *Content Consumer <isa> End-user*

Rationale: A *Content Consumer* is any entity that accesses the Digital Library to exploit (part of) its *Resources*. A person who searches (*Search* function) the contents of a digital *Collection* or an external subscription service are instances of *Content Consumers*.

Examples:

- John, the *End-user* of a scientific Digital Library, downloads a set of *Collections* in order to process this data with its novel procedure.

C25 Content Creator

Definition: The *Role* of the *Actors* that provide new *Information Objects* to be stored in the Digital Library or update existing *Information Objects*.

Relationships:

- *Content Creator <isa> End-User*

Rationale: A *Content Creator* may be a human or a program or another system. For instance, it may be a person who creates and inserts their own documents in the Digital Library or an external program that automatically converts artefacts to digital form and uploads them to the Digital Library.

Examples:

- John, the *End-user* of a scientific Digital Library, uploads a new version of a working paper reporting on the latest results of its experimentation in a *Collection* shared with other colleagues working on a similar topic to prompt and informed feedback.

C26 Librarian

Definition: The *Role* of the *Actors* that manage digital library's *Resources,* namely *Information Objects* and *End-users*.

Relationships:

- *Librarian <isa> End-user*

Rationale: Librarians are *End-users* in charge of curating the DL Content. In fact, one of the aspects distinguishing Digital Libraries from the Web is the effort spent in the Digital Libraries to guarantee a quality of the service, i.e. the effort spent by these actors have to curate all the resources forming the DL.

Examples:

- Frank, the *Librarian* of a University Digital Library, is in charge to appropriately revise and classify scholarly works as to simplify the discovery by Digital Library *End-users*. He is also in charge to implement the *content policies* by appropriately configuring the Digital Library *Functions*.

C27 DL Designer

Definition: The *Role* of the *Actors* that, by interacting with the *Digital Library Management System*, define the characteristics of the *Digital Library*.

Relationships:

- *DL Designer <isa> Role*

Rationale: *DL Designers* are *Actors* that by exploiting their knowledge of the application semantic domain define, customise and maintain the Digital Library so that it is aligned with the information and functional needs of its potential DL End-users. These actors are expected to interact with the DLMS, i.e. the 'system' providing them with functional and content configuration facilities. Functional configuration instantiate aspects of the DL functions perceived by the DL End-users, including the characteristics of the result set format, query language(s), user profile formats, and Information Object model employed. Content configuration specify aspects of the DL Content domain, e.g. repositories of content, ontologies, classification schemas, authority files, and gazetteers that will be made available through the DL.

Examples:

- Frank, the *DL Designer* of a scientific Digital Library, is in charge to identify the set of *Collections* and *Functions* constituting the Digital Library and define the characteristics of the *User Domain* (e.g. *Roles* and *Groups*), *Policy Domain* (e.g. establish the *Content Policy*) and *Quality Domain* (e.g. establish the *Generic Quality Parameters*) instances.

C28 DL System Administrator

Definition: The *Role* of the *Actors* that, by interacting with the *Digital Library Management System*, define the characteristics of the *Digital Library System*, put this in action and monitor its status so as to guarantee the operation of the *Digital Library*.

Relationships:

- *DL System Administrator <isa> Role*

Rationale: DL System Administrators are in charge to select and deploy the *Architectural Components* (C203) needed to operate the *Digital Library System* implementing the expected *Digital Library*. Their choice of elements reflects the expectations that DL End-users and DL Designers have for the *Digital Library*, as well as the requirements the available resources impose on the definition of the DL. DL System Administrators interact with the DLMS by providing architectural configuration parameters, such as the chosen software components and the selected hosting nodes. Their task is to identify the architectural configuration that best fits the DLS in order to ensure the highest level of quality of service. The value of the architectural configuration parameters can be changed over the DL lifetime. Changes of parameter configuration may result in the provision of different DL functionality and/or different levels of quality of service.

Examples:

- John, the *DL System Administrator* of a scientific Digital Library, by interacting with the DLMS decides to deploy three replicas of the *Software Component* implementing Repository related *Functions* on three different servers (*Hosting Nodes*) in order to address the needs on its community.

C29 DL Application Developer

Definition: The *Role* of the *Actors* that, by interacting with the *Digital Library Management System*, enrich or customise the set of *Software Components* that will be used by the *DL System Administrator* to implement the *Digital Library System* serving the *Digital Library*.

Relationships:

- *DL Designer <isa> Role*

Rationale: DL Application Developers develop the *Software Components* (C205) that will be used as constituents of the DLSs, to ensure that the appropriate levels and types of functionality are available.

Examples:

- Mark, one of the *DL Application Developers* of a scientific Digital Library, is in charge to develop a new *Software Component* for managing *Annotations* of specific *Information Objects*.

C30 Functionality Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents the various aspects related to the modelling of facilities/services provided in the Digital Library universe to serve *Actor* needs.

Relationships:

- *Digital Library <definedBy> Functionality Domain*
- *Digital Library System <definedBy> Functionality Domain*
- *Digital Library Management System <definedBy> Functionality Domain*

- *Functionality Domain <consistOf> Functions*

Rationale: The *Functionality Domain* concept represents the services that *Digital Libraries* offer to their *Actors*. The set of facilities expected from *Digital Libraries* is extremely broad and varies according to the application context. There are a number of *Functions* that *Actors* expect from each *Digital Library*, e.g. search, browse, information objects visualisation. Beyond that, any *Digital Library* offers additional *Functions* to serve the specific needs of its community of users.

Examples: --

C31 Function

Definition: A particular operation that can be realised on a *Resource* or *Resource Set* as the result of an activity of a particular *Actor*. It is identified by a *Resource Identifier*. It may be performed by an *Actor* or it may refer to the respective supporting process of the DLS.

Relationships:

- *Function <isa> Resource*
- *Function* is *<identifiedBy> Resource Identifier* (inherited from *Resource*)
- *Function* is *<influencedBy> Actor Profile*
- *Function* is *<influencedBy> Policy*
- *Function <actOn> Resource*
- *Function* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Function <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Actor <perform> Function*

Rationale: A *Function* captures any processing that can occur on *Resources* and is typically perceived as a result of an activity of an *Actor* in a Digital Library. It can possibly involve any type of *Resource* and can potentially be performed by any kind of *Actor*. For instance, not only a user can *Search* the contents in a digital library, i.e. *Information Objects*, but also an *Actor* can search for other *Actors*, a program can *Search* for offered *Functions*, and so forth. Due to its broad scope, *Function* is specialised into a set of specific but still quite generic subclasses, such as *Access Resource*. In practice, a *Digital Library* can use different specialisations and combinations of these *Functions* intended for different *Actors* and *Resources*.

Examples:

- *Access Resource*
- *Manage Resource*

C32 Access Resource

Definition: The class of *Functions* that provide *Actors* with mechanisms for discovering and accessing *Resources*.

Relationships:

- *Access Resource <isa> Function*
- *Access Resource <retrieve> Resource*
- *Discover <isa> Access Resource*
- *Acquire <isa> Access Resource*
- *Visualise <isa> Access Resource*

Rationale: This is a family of *Functions* that do not modify the Digital Library or its *Resources* but help in identifying *Resources* intended to be simply examined and perceived by an *Actor* or possibly further exploited through the use of other functions, such as *Manage Resource* functions.

Examples:

- *Discover*, *Acquire* and *Visualise* are three classic Access Resource functions.

C33 Discover

Definition: The family of *Functions* to find a *Resource,* which may be an individual one or a *Resource Set* compliant with the specification of the *Actor* request, as expressed by a *Query* or by browsing.

Relationships:

- *Discover <isa> Access Resource*
- *Discover <actOn> Resource Set*
- *Discover <return> Result Set*
- *Search <isa> Discover*
- *Browse <isa> Discover*

Rationale: *Discover* is the central *Access Resource* function, which acts on *Resource Sets* and aims at retrieving desired *Resources*.

Examples: --

C34 Browse

Definition: An *Access Resource* function that lists *Resources* in a *Resource Set* ordered or organised according to a given characteristic or scheme.

Relationships:

- *Browse <isa> Discover*

Rationale: The *Browse* function allows an *Actor* to explore a digital library's *Resources* and may be used alternately with *Search* for this purpose. A digital library can be equipped with different *Browse* capabilities. For instance, it may provide a different ordering or grouping of *Resources*, such as browse per-author, when a *Collection* of publications is explored to search for the correct form of the name of an author, or through an ontology representing the underlying *Collection* of *Information Objects* or the set of permissible *Functions*. Alternatively, graphical representations of a *Resource Set* may be used for browsing DL *Resources*. For instance, it may be possible to have a digital library *Collection* 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 example is that of a tag cloud,[21] i.e. a visual depiction of descriptors, namely tags, used to annotate Resources. Tags are typically listed alphabetically, and tag frequency is shown with font size or colour. The tags are usually hyperlinks that lead to a collection of items that are associated with that tag.

Examples: --

[21] http://en.wikipedia.org/wiki/Tag_cloud

C35 Search

Definition: An *Access Resource* function that allows an *Actor* to discover the *Resources* matching a *Query*, which are returned as a *Result Set*. *Search* must be triggered by a *Query*.

Relationships:

- *Search <isa> Access Resource*
- *Search <issue> Query*
- *Search <return> Result Set*

Rationale: There are several types of *Search* that can be performed by different types of *Actors* and for accessing different types of *Resources*. For instance, not only can a person *Search* the contents in a digital library, i.e. *Information Objects*, but also an *Actor* can *Search* for other *Actors*, a program can *Search* for offered *Functions*, and so forth. Furthermore, the *Query* describing the desired objects may be based on the content of a *Resource*, its *Actor Profile*, its *metadata*, its *annotations* and so forth, and any combination of these. The form of the *Query* does not constrain the type of *Resource* retrieved, e.g. a textual query can be used to retrieve *Information Objects* whose *manifestations* are videos or audio files. An important characteristic of the *Search* function is the search paradigm adopted. For example, the *Information Objects* sought may be described through a query specification or condition. This may consist of an unstructured query condition, i.e. sequence of search terms, combined with operators, such as 'and', 'or' and 'not', or it can be a structured or fielded search, where query conditions are expressed in terms of the metadata fields, e.g. 'all the information objects on a given research topic created by a certain author and published in a specific period of time'. Moreover, an important characteristic of the *Search* functionality lies in which model is adopted in identifying the pertinence of the objects with respect to a query, e.g. the Boolean model or the vector-space model.

Examples:

- 'Query-By-Example', which is based on an example *Resource* provided by the *Actor*. This allows end-users, for instance, to *Search* for *Resources* similar to a provided sample image as well as to *Search* for those deemed similar to an excerpt of an audio.
- 'Relevance feedback'. This supports the iterative improvement of the search *Result Set* by allowing the *Actor* to express a relevance judgement on the retrieved *Resources* at each iteration step. It improves the discovery mechanism and the user satisfaction effectively as it enhances the expressive power of the query language supported by the digital library.

C36 Acquire

Definition: An *Access Resource* function supporting an *Actor* in retaining *Resources* in existence past the lifetime of the *Actor*'s interaction with the system.

Relationships:

- *Acquire <isa> Access Resource*
- *Acquire <actOn> Resource*

Rationale: This *Function* provides mechanisms such as locally saving and printing the content or *metadata* related to *Information Objects*.

Examples: --

C37 Visualise

Definition: An *Access Resource* function enabling an *Actor* to view a *Resource* graphically, such as an *Information Object* or an *Actor Profile*.

Relationships:

- *Visualise <isa> Access Resource*
- *Visualise <actOn> Resource*

Rationale: *Resources* may be complex and may be comprised of several parts. For instance, an *Information Object* may combine information manifested through different media. The *Visualise* function must therefore be tailored according to the *End-user* characteristics, like the device it uses or its personal setting, as well as the characteristics of the object to be rendered. Visualisation is any technique for creating images, diagrams, animations and so forth to communicate a message.

Examples:

- Animation and the drawing of diagrams are examples of the *Visualise* function.

C38 Manage Resource

Definition: The class of *Functions* that supports the production, withdrawal or update of *Resources*.

Relationships:

- *Manage Resource <isa> Function*
- *Manage Information Object <isa> Manage Resource*
- *Manage Actor <isa> Manage Resource*
- *Manage Function <isa> Manage Resource*
- *Manage Policy <isa> Manage Resource*
- *Manage Quality Parameter <isa> Manage Resource*
- *Create <isa> Manage Resource*
- *Update <isa> Manage Resource*
- *Validate <isa> Manage Resource*
- *Submit <isa> Manage Resource*
- *Withdraw <isa> Manage Resource*
- *Annotate <isa> Manage Resource*

Rationale: This is a family of *Functions*, since the tasks to be performed in order to manage a set of objects are numerous. Specifically, *Manage Resource* contains general categories of *functions* applied to each specific domain, as well as general *Functions*, the specialisations of which may appear in each individual domain.

Examples: --

C39 Create

Definition: A *Manage Resource* function supporting an *Actor* in creating a new *Resource*.

Relationships:

- *Create <isa> Manage Resource*
- *Create <actOn> Information Object*

Rationale: This *function* encapsulates the capabilities to create new *Resources*.

Examples: --

C40 Submit

Definition: A *Manage Resource* function supporting an *Actor* in providing the digital library with a new *Resource*.

Relationships:

- *Submit <isa> Manage Resource*
- *Submit <actOn> Resource Set*

Rationale: This *function* supports the *Actor* in submitting a new *Resource* to the DL. According to the policies established by the DL designer, the submit function can add the newly created *Resource* to either an incoming *Resource Set*, i.e. a temporary area that contains all the objects waiting to be published by the librarians, or directly to the DL *Resource Set*, i.e. the set of *Resources* seen by DL *Actors*.

Examples: --

C41 Withdraw

Definition: A *Manage Resource* function supporting an *Actor* in withdrawing *Resources* from the DL.

Relationships:

- *Withdraw <isa> Manage Resource*
- *Withdraw <actOn> Resource Set*

Rationale: This *function* support the *Actor* in revising the set of *Resources* the Digital Library provides its End-users with. In fact, in addition to the adjunction to new *Resources* to the Digital Library (*Submit*) it should be possible to remove (Withdraw) or make up to date (*Update*) existing *Resources*. Because of the

Examples: --

C42 Update

Definition: A *Manage Resource* function allowing an *Actor* to modify an existing *Resource*.

Relationships:

- *Update <isa> Manage Resource*

Rationale: This *Function* implies capabilities to modify the *Resource*.

Examples:

- In the case of *Information Objects*, it may add a new *Edition* or a new *View* to an existing *Information Object*.

C43 Validate

Definition: A *Manage Resource* function supporting the *Actor* in validating the quality status of a DL *Resource*.

Relationships:

- *Validate <isa> Manage Resource*

Rationale: This function supports the *Actor* in validating the quality status of a *Resource* of the DL. The *Function* makes use of relevant *quality parameters*.

Examples: --

C44 Annotate

Definition: A *Manage Resource* function allowing an *Actor* to create an *Annotation* about a *Resource*.

Relationships:

- *Annotate <isa> Manage Resource*
- *Annotate <createAnnotation> Annotation*

Rationale: This *Function* allows an *Actor* to add *Annotations*. *Annotations* are *Information Objects*. Management of existing *Annotations* may be performed using *Manage Resource* and *Manage DL* functions. Moreover, since there are different types of *Annotations*, such as notes and bookmarks, the *Annotate* function may allow for the definition of one or more types that comply with the different meanings of *Annotation* in use.

Examples: --

C45 Manage Information Object

Definition: The class of *Functions* that support the production, withdrawal, update, publishing and processing of *Information Objects*.

Relationships:

- *Manage Information Object <isa> Manage Resource*
- *Manage Information Object <actOn> Information Object*
- *Disseminate <isa> Manage Information Object*
- *Process <isa> Manage Information Object*
- *Author <isa> Manage Information Object*

Rationale: This category of *Functions* contains a broad set of *Functions* related to all the aspects of the creation, dissemination and processing of *Information Objects*.

Examples: --

C46 Disseminate

Definition: A *Manage Information Object* function supporting an *Actor* in making *Information Objects* known to the *End-users* according to certain *Policies*.

Relationships:

- *Disseminate <isa> Manage Information Object*
- *Publish <isa> Disseminate*

Rationale: This *Function* performs the dissemination of the *Information Object*, more precisely of its *metadata* or description, to the public through the DL in accordance with the *Policies* assigned to it. In particular, the DL system may alert *Groups* or the wider public to the import of new *Information Objects* or *Collections* to the DL. Thanks to this characteristic, digital libraries have become proactive systems instead of being just passive systems responding to user queries.

Examples: --

C47 Publish

Definition: A *Manage Information Object* supporting an *Actor* in making *Information Objects* available to the DL according to certain *Policies*.

Relationships:

- *Publish <isa> Disseminate*

Rationale: The *Information Objects* become available within the DL in accordance with the *Policies* assigned to them.

Examples: --

C48 Author

Definition: A *Manage Information Object* function supporting an *Actor* in creating *Information Objects*.

Relationships:

- *Author <isa> Manage Information Object*
- *Author <creates> Information Object*

Rationale: This *Function* enables the *Actor* to create *Information Objects* according to one of the DL's accepted *Information Objects' Resource Format*.

Examples: --

C49 Compose

Definition: A *Manage Information Object* function supporting the *Actor* in using (parts of) existing *Information Objects* in order to build compound objects.

Relationships:

- *Compose <isa> Author*

Rationale: This *Function* encapsulates the capabilities to create new *Information Objects* by reusing existing objects, either in part or as a whole. For example, the user may compose a multimedia album by putting together audio files, song lists and singer biographies.

Examples: --

C50 Process

Definition: A *Manage Information Object* function supporting the *Actor* in all activities related to the transformation and analysis of an *Information Object*.

Relationships:

- *Process <isa> Manage Information Object*
- *Analyze <isa> Process*
- *Transform <isa> Process*

Rationale: This *Function* encapsulates the capabilities to analyse and transform *Information Objects* in order to view, disseminate or extract information from them. This represents a very important category of *Functions* as it contains fundamental activities for taking advantage of the DL *Content* for scientific, educational and recreational purposes.

Examples: --

C51 Analyse

Definition: A *Process* function supporting the *Actor* in all activities related to the analysis of an *Information Object*.

Relationships:

- *Analyse <isa> Process*

- *Linguistic Analysis <isa> Analyse*
- *Qualitative Analysis <isa> Analyse*
- *Statistical Analysis <isa> Analyse*
- *Scientific Analysis <isa> Analyse*
- *Create Structured Representation <isa> Analyse*
- *Compare <isa> Analyse*

Rationale: This *Function* encapsulates the capability to analyse *Information Objects* in order to extract information from them. It includes *Functions* related to the analysis of the *Information Object* content or *metadata*, for statistical, scientific, linguistic, preservation, etc. purposes.

Examples: --

C52 Linguistic Analysis

Definition: An *Analyse* function supporting the *Actor* in all activities related to the analysis of the textual content of an *Information Object*.

Relationships:

- *Linguistic Analysis <isa> Analyse*

Rationale: This *Function* represents the group of *Functions* relevant to the linguistic analysis of the textual parts of an *Information Object*, related to both its structure (grammar) and meaning (semantics). It is a crucial one, especially in the case of textual content of particular importance in that respect, i.e. old manuscripts, literature, etc. A very important specialisation of this function could be the detection of named entities in the text or the identification of conceptual relationships in order, for example, to automatically extract concepts and relations for the creation of *Ontologies* related to the *Content*.

Examples: --

C53 Qualitative Analysis

Definition: An *Analyse* function supporting the *Actor* in all activities related to the analysis of the quality of an *Information Object* or its *metadata*. It computes an appropriate *Content Quality Parameter*.

Relationships:

- *Qualitative Analysis <isa> Analyse*
- *Qualitative Analysis <measure> Content Quality Parameter*
- *Examine Preservation State <isa> Qualitative Analysis*

Rationale: This *Function* represents the group of *Functions* relevant to the qualitative analysis of an *Information Object* or its *metadata*. This can include authenticity, preservation state, integrity, provenance etc. The result of this analysis is the measurement of one or more *Content Quality Parameters*.

Examples: --

C54 Examine Preservation State

Definition: A *Qualitative Analysis* function supporting the *Actor* to examine the preservation state of an *Information Object* or its *metadata*. It computes an appropriate *Preservation Quality Parameter*.

Relationships:

- *Examine Preservation State <isa> Qualitative Analysis*

Rationale: This *Function* plays the very important role of examining the preservation state of *Information Objects* in the DL. It is crucial, as it may provide the incentive for restorative or preventive measures to ensure high standards of content quality. The result of this analysis is the measurement of one or more *Preservation Quality Parameters*.

Examples: --

C55 Statistical Analysis

Definition: An *Analyse* function supporting the *Actor* in all activities related to the statistical analysis of an *Information Object*.

Relationships:

- *Statistical Analysis <isa> Analyse*

Rationale: This *Function* represents the group of *Functions* relevant to the computation of statistics related to an *Information Object*.

Examples: --

C56 Scientific Analysis

Definition: An *Analyse* function supporting the *Actor* in all activities related to the scientific analysis of data represented as an *Information Object*.

Relationships:

- *Scientific Analysis <isa> Analyse*

Rationale: This *Function* represents the group of *Functions* relevant to the scientific analysis of an *Information Object*. It includes actions and tools such as running a model on a set of data, making scientific computations, offering handbooks with 'live' formulae, etc. As DLs with scientific data are of specific importance to the scientific community, their incorporation of a wide range of tools for the analysis of these data will be crucial in promoting research and knowledge creation, as well as education, in the fields of natural sciences, medicine, etc.

Examples: --

C57 Create Structured Representation

Definition: An *Analyse* function supporting the *Actor* in all activities related to the analysis of the structure of an *Information Object* and the creation of a representation of this structure.

Relationships:

- *Create Structured Representation <isa> Analyse*

Rationale: This *Function* represents the group of *Functions* relevant to the identification of the structure of an *Information Object*, which may refer to an ontology or a table of contents extracted from a text, a grouping of specific scientific data, etc. It is closely related to the *Visualise* function, as the created structure may have different ways of being presented to the *Actor*.

Examples: --

C58 Compare

Definition: An *Analyse* function supporting the *Actor* in comparing two or more *Information Objects*, either primary ones or their *metadata*.

Relationships:

- *Compare <isa> Analyse*

Rationale: This *Function* represents the group of *Functions* relevant to the comparison of *Information Objects*. This may be performed for many reasons, preservation being a very important one among them.

Examples: --

C59 Transform

Definition: A *Process* function enabling an *Actor* to create different views or manifestations of an *Information Object* (or a set of *Information Objects*).

Relationships:

- *Transform <isa> Process*
- *Physically Convert <isa> Transform*
- *Extract <isa> Transform*
- *Convert to Different Format <isa> Transform*

Rationale: Different representations of an *Information Object* (or a set of *Information Objects*) enable the *Actor* to perceive information at different levels of abstraction, as desired. Such possible conversions may be achieved with the help of approaches such as format conversions, information extraction, automatic translation and summarisation techniques.

Examples: --

C60 Physically Convert

Definition: A *Transform* function supporting the *Actor* in creating new manifestations of an *Information Object*.

Relationships:

- *Physically Convert <isa> Transform*
- *Physically Convert <createManifestation> Information Object*
- *Translate <isa> Physically Convert*

Rationale: This *Function* represents a wide range of *Functions* related to the transformation of the content of the *Information Object*. The transformation may include translation, text-to-speech and speech-to-text conversions, tables in texts into spreadsheet or database format, data into graphs, from 3D to 2D, different media (including from paper to digital form), images into colour histograms etc.

Examples: --

C61 Translate

Definition: A *Physically Convert* function enabling *Actors* to perceive an *Information Object* in a language that is different from the object's or the user's native language. In this context, languages can range from country languages, e.g. Italian, English, to community and cultural languages, e.g. Muslim culture.

Relationships:

- *Translate <isa> Physically Convert*

Rationale: Digital libraries must support access to the *Information Objects* in as many different languages as possible to enhance the usage of their content. This function enables multilingual information access. Multilingual information access approaches include query translation, information object translation and combinations of these.

Examples: --

C62 Convert to a Different Format

Definition: A *Transform* function enabling an *Actor* to obtain a different *View* of an *Information Object* (or a set of *Information Objects*).

Relationships:

- *Convert to a Different Format <isa> Transform*
- *Convert to a Different Format <createView> Information Object*

Rationale: This *Function* enables the user to create a new *Version* (e.g. convert the object into another encoding). Depending on the type of object, different types of conversions may be possible, such as conversion into different encoding (converting a text from pdf to Word, an image to a different format or compression scheme, etc). This *Function* is particularly useful for interoperability purposes.

Examples: --

C63 Extract

Definition: A *Transform* function enabling an *Actor* to obtain a different manifestation of an *Information Object* (or a set of *Information Objects*).

Relationships:

- *Extract <isa> Transform*
- *Extract <createManifestation> Information Object*

Rationale: This *Function* enables the user to create a new manifestation of an object that may contain several parts of it. An example of such a *Function* may be the extraction of citations or text summaries.

Examples: --

C64 Manage Actor

Definition: A *Manage Resource* function supporting the administration of the set of *Actors* that access the digital library.

Relationships:

- *Manage Actor <isa> Manage Resource*
- *Manage Actor <actOn> Actor*
- *Establish Actor <isa> Manage Actor*
- *Personalise <isa> Manage Actor*

Rationale: This is a family of *Function*s supporting the DL administrators in dealing with the DL user management. In particular, they cover the creation of new *Actors*, remove existing ones, and regulate their rights, i.e. establish the tasks they are entitled to perform and the

Information Objects they are entitled to use as well their profile and associated personalisation issues.

Examples: --

C65 Establish Actor

Definition: A *Manage Actor* function dealing with the specific issues of the creation of the *Actors* and their recognition by the DL.

Relationships:

- *Establish Actor <isa> Manage Actor*
- *Register <isa> Establish Actor*
- *Login <isa> Establish Actor*

Rationale: An important aspect of the management of the DL *Actors* is user creation, registration, login and application of their profile to their actions.

Examples: --

C66 Register

Definition: An *Establish Actor* function supporting the adding of a new *Actor* to the set of those managed and recognised by the digital library.

Relationships:

- *Register <isa> Establish Actor*
- *Sign Up <isa> Register*

Rationale: This function is responsible for populating the digital library user community. Usually, the fewer the requirements imposed on the registration of new users, the harder it is for the system to safeguard the identity of a user. The constraints imposed at the time of registration are a direct consequence of the audience for which the digital library is designed. All these aspects are decided by the *DL Designer* at the DL design stage and are related to *Policies* and requirements that define the available *Actor Profiles*.

Examples: --

C67 Sign Up

Definition: A *Register* function supporting *Actors* in actively requesting their registration in the DL and possibly expressing an interest in particular aspects of the DL.

Relationships:

- *Sign Up <isa> Register*

Rationale: This function encapsulates actions relevant to the active request of the *Actor* to be registered in the DL and have access to its content. It is closely related to personalisation, as during this process the *Actor* may fine-tune certain aspects of their *Actor Profile*.

Examples: --

C68 Login

Definition: An *Establish Actor* function that enables an *Actor* to establish their identity in the DL.

Relationships:

- *Login <isa> Establish Actor*

Rationale: *Login* is performed by matching a set of qualities or characteristics that uniquely identify an *Actor*. Assurance of identification can be increased by a number of practices

appropriate to the need. These practices range from passwords to tokens, smart cards, and public keys with Certificates. The system then performs authentication, and may also perform authorisation of the user. The execution of this *Function* should be regulated by *Policies*.

Examples: --

C69 Personalise

Definition: The class of *Manage Actor* that supports *Actors* in having personalised access to the *Content* and *Functionality* of the DL.

Relationships:

- *Personalise <isa> Manage Actor*
- *Apply Profile <isa> Personalise*

Rationale: This is a family of *Functions* designed to adapt aspects of a digital library to the DL user's needs. These aspects may range from the DL 'look and feel' to the organisation of the digital library *Content* so that it satisfies the personal interest of its users. A main group of personalisation functions includes customisation and application of the *Actor Profile* to all DL *Resources*, whereas other *Functions* may be related to user feedback to the DL in order to improve the *Functionality* and *Content* provided.

Examples: --

C70 Apply Profile

Definition: A *Personalise* function enabling the applications of the *Actors* to the various types of *Function* offered by a digital library.

Relationships:

- *Apply Profile <isa> Personalise*

Rationale: This *Function* assumes that the system (semi-)automatically constructs a profile per user. Then, profile information is used to personalise the DL *Functions*, e.g. personalised search, recommendations, and so forth.

Examples: --

C71 Manage Function

Definition: A *Manage Resource* supporting the administration of the features of the *Functions* provided by the DL.

Relationships:

- *Manage Function <isa> Manage Resource*

Rationale: This is a family of *Functions* supporting the administration of the DL functionality. In particular, they cover the addition, withdrawal and updating of new *Functions*.

Examples: --

C72 Manage Policy

Definition: A *Manage Resource* supporting the administration of the set of *Policies* governing the DL and its *Resources*.

Relationships:

- *Manage Policy <isa> Manage Resource*

Rationale: This is a family of *Functions* supporting the administration of the DL *Policies*, which are related to all the DL *Resources*. In particular, they cover the creation of new *Policies* and remove or update existing ones.

Examples: --

C73 Manage Quality Parameter

Definition: A *Manage Resource* supporting the administration of the individual *Quality Parameters*, which refer to all aspects of the DL.

Relationships:

- *Manage Quality Parameter <isa> Manage Resource*

Rationale: This is a family of *Functions* supporting the administration of *quality parameters*, e.g. their definition.

Examples: --

C74 Collaborate

Definition: The class of functions that supports *Actors* in sharing information, working and communicating effectively and efficiently with peers.

Relationships:

- *Collaborate <isa> Function*
- *Exchange Information <isa> Collaborate*
- *Converse <isa> Collaborate*
- *Find Collaborator <isa> Collaborate*
- *Author Collaboratively <isa> Collaborate*

Rationale: This is a family of *Functions* that consists of a set of capabilities designed to support *Actors* in using the DL as a common workspace. Some of the *Functions* may be specialisations of other *Functions* also, related to information access.

Examples: --

C75 Exchange Information

Definition: A *Collaborate* function that supports *Actors* in sharing and exchanging information with peers.

Relationships:

- *Exchange Information <isa> Collaborate*

Rationale: This is a group of *Functions* that allows *Actors* to exchange *Information Objects*, which may be *Annotations* or *Metadata*, or even e-mails and personal messages with attached documents exchanged through the DL system.

Examples: --

C76 Converse

Definition: A *Collaborate* function that supports an *Actor* in conversing through the DL system.

Relationships:

- *Converse <isa> Collaborate*

Rationale: This is a group of *Functions* that allows *Actors* to talk to peers and exchange views and opinions through DL chat services, online fora or list servers.

Examples: --

C77 Find Collaborator

Definition: A *Collaborate* function that supports an *Actor* in conversing through the DL system.

Relationships:

- *Find Collaborator <isa> Collaborate*
- *Find Collaborator <retrieve> Actor*

Rationale: This *Function* allows an *Actor* to locate other *Actors* of the DL system that will be eligible for collaboration.

Examples: --

C78 Author Collaboratively

Definition: A *Collaborate* function that supports an *Actor* in authoring *Information Objects* collaboratively.

Relationships:

- *Author Collaboratively <isa> Collaborate*
- *Author Collaboratively <createVersion> Information Object*

Rationale: This *Function* allows the *Actors* to collaborate in authoring an *Information Object* in order to create a new version (<*hasView*> or <*hasEdition*>) of it.

Examples: --

C79 Manage DL

Definition: The class of *Functions* managing the *Content*, *Actors* and other *Resources* of the DL in order to achieve the desired *Quality Parameters* in agreement with the established *Policies*.

Relationships:

- *Manage DL <isa> Function*
- *Manage Content <isa> Manage DL*
- *Manage User <isa> Manage DL*
- *Manage Functionality <isa> Manage DL*
- *Manage Quality <isa> Manage DL*
- *Manage Policy Domain <isa> Manage DL*

Rationale: This class involves *Functions* dealing with managing issues of the DL domains, involving the import and export of *Collections*, the definition of *Actor Roles*, the management of general *Policy* issues, etc.

Examples: --

C80 Manage Content

Definition: The class of *Functions* managing the *Content* of the DL in order to achieve the desired *Quality Parameters* in line with the established *Policies*.

Relationships:

- *Manage Content <isa> Manage DL*
- *Manage Collection <isa> Manage Content*
- *Preserve <isa> Manage Content*

Rationale: This family of *Functions* is related to the management of general *Content* issues such as the import and export of *Collections* from and to other DLs to support interoperability as well as preservation-related functions, such as monitoring the overall preservation state of *Collections*.

Examples: --

C81 Manage Collection

Definition: A *Manage Content* function supporting *Actors* in creating, updating, exporting, importing and removing *Collections*.

Relationships:

- *Manage Collection <isa> Manage Content*
- *Import Collection <isa> Manage Collection*
- *Export Collection <isa> Manage Collection*

Rationale: The importance of *Collections* as a mechanism for organising digital library *Content* was introduced in Section II.2.2. The *Manage Collection* function models the class of *Functions* for dealing with *Collections*. For example, by exploiting this class of functions, *Librarians* can build new *Collections* or modify existing ones, which are accessed by many users. On the other hand, *Content consumers* are enabled to construct their personal virtual organisation of the digital library *Content*. This organisation might resemble the file system folder paradigm, with the main difference that it is virtual 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*.

It should be noted here that the *Functions* for collection management can also be grouped under the *Manage Resource* function. However, the management of *Collections* should be differentiated as it contains two very important *Functions* that are related to issues of interoperability and preservation – the import and export of collections from and to other DLs.

Examples: --

C82 Import Collection

Definition: The *Manage Collection* function that supports the selection of the third-party information sources whose objects will populate the DL *Content*.

Relationships:

- *Import Collection <isa> Manage Collection*

Rationale: This *Function* can be realised in different ways according to which typology of the DLMS ingestion mechanism is supported.

Examples:

- Harvesting the content of an OAI [143] compliant Repository through OAI-PMH [144] is a kind of *Import Collection*.

C83 Export Collection

Definition: The *Manage Collection* function that supports the export of the DL *Content*.

Relationships:

- *Export Collection <isa> Manage Collection*

Rationale: This functionality can be realised in different ways in order to make its collection content available to other DLs.

Examples:

- Having the DL *Content* comply with the OAI [143] and thus available through OAI-PMH [144] is a possible implementation of the *Export Collection* function.

C84 Preserve

Definition: A *Manage Content* function supporting an *Actor* in all actions that involve the preservation of the DL *Content*.

Relationships:

- *Preserve <isa> Manage Content*

Rationale: This group of *Functions* supports the definition of general preservation programs for specific *Collections*, the monitoring of their preservation state and the organisation of preservation works for the DL *Content*. *Preservation Policies* are very important for the preservation-related *Functions*.

For a comprehensive description of the *Preservation* issue, please refer to Section II.3.2.

Examples: --

C85 Manage User

Definition: A *Manage DL* function supporting an *Actor* in defining and managing *Roles*, *Groups* and, in general, concepts related to the *User Domain*.

Relationships:

- *Manage User <isa> Manage DL*
- *Manage Membership <isa> Manage User*
- *Manage Group <isa> Manage User*
- *Manage Profile <isa> Manage User*
- *Manage Role <isa> Manage User*

Rationale: This group of *Functions* supports the definition of *Actor groups*, *Profiles* and *Roles* as well as any *Function* that is related to the management of general issues in the *User Domain*, such as organising campaigns for new membership in the DL.

Examples: --

C86 Manage Membership

Definition: A *Manage User* function supporting an *Actor* in organising campaigns for new DL subscribers or maintaining the current ones.

Relationships:

- *Manage Membership <isa> Manage User*

Rationale: The DL as an organisation in some cases aims at acquiring new subscribers (*Content Consumers*), either for profit or to become known and support its status, and also at maintaining its current subscribers. This is a function group containing *Function*s such as sending e-mails to the current users or the wider public informing them about new *Content* in

the DL, making suggestions to users about *Content* that may interest them, informing them about the expiry of their subscription and suggesting renewal, etc.

Examples: --

C87 Manage Group

Definition: A *Manage User* function that supports the management of *Groups* and the fine-tuning of their characteristics.

Relationships:

- *Manage Group <isa> Manage User*

Rationale: This *Function* supports an *Actor* in managing the *Groups* that are supported by the DL, in terms of characteristics, rights and permissions, etc.

Examples: --

C88 Manage Role

Definition: A *Manage User* function that supports the management of *Roles* and the fine-tuning of their characteristics.

Relationships:

- *Manage Role <isa> Manage User*

Rationale: This *Function* supports an *Actor* in defining the roles supported by the created DL, giving each of them rights and permissions, creating new ones and so forth.

Examples: --

C89 Manage Actor Profile

Definition: A *Manage User function* that supports the management of the *Actor Profile* characteristics.

Relationships:

- *Manage Role <isa> Manage User*

Rationale: This *Function* supports the *Actors* in updating the structure and the information types that may be stored in the *Actor profiles* supported by the created DL.

Examples: --

C90 Manage Functionality

Definition: A *Manage DL* function supporting *Actors* in defining and managing functionality-related issues.

Relationships:

- *Manage Functionality <isa> Manage DL*

Rationale: This *Function* supports the *Actors* in handling functionality-related issues such as defining general issues of how the *Functions* will be presented and provided to the *End-users*.

Examples: --

C91 Monitor Usage

Definition: A *Manage Functionality* function supporting an *Actor* in monitoring the use of the DL *Functions*.

Relationships:

- *Monitor Usage <isa> Manage Functionality*

Rationale: This *Function* supports the *Actors* in monitoring the use of the provided *Functions* in order to gain insight on *End-user* problems and issues relevant to specific *Functions*.

Examples: --

C92 Manage Quality

Definition: A *Manage DL* function supporting an *Actor* in the management of general *Quality Domain* issues.

Relationships:

- *Manage Quality <isa> Manage DL*

Rationale: This *Functio*n supports the management of quality domain issues.

Examples: --

C93 Manage Policy Domain

Definition: A *Manage DL* function supporting an *Actor* in defining and managing general *Policy Domain* aspects in order to regulate the usage of the digital library.

Relationships:

- *Manage Policy Domain <isa> Manage DL*

Rationale: This *Function* supports the management of general policy-related issues.

Examples: --

C94 Manage & Configure DLS

Definition: The class of *Functions* that supports the management and configuration of the DLS that implements the DL.

Relationships:

- *Manage & Configure DLS <isa> Function*
- *Manage DLS <isa> Manage & Configure DLS*
- *Configure DLS <isa> Manage & Configure DLS*

Rationale: This class allows the *Actors* to create and manage the *Digital Library System*. In particular, its *Functions* are related to 'Configure', 'Deploy' and 'Monitor', corresponding, respectively, to the configuration, deployment and monitoring phases of a digital library development process.

Examples: --

C95 Manage DLS

Definition: The class of *Functions* supporting the management of the DLS that implements the DL.

Relationships:

- *Manage DLS <isa> Manage & Configure DLS*
- *Create DLS <isa> Manage DLS*
- *Withdraw DLS <isa> Manage DLS*
- *Update DLS <isa> Manage DLS*
- *Manage Architecture <isa> Manage DLS*

Rationale: This class allows the *Actors* to create, update and withdraw the DLS as well as manage its *Architecture* so that they provide the DL required.

Examples: --

C96 Create DLS

Definition: A *Function* that supports the creation of the DLS that implements the DL**.**

Relationships:

- *Create DLS <isa> Manage DLS*

Rationale: This *Function* supports the creation of a new DLS through a DLMS.

Examples: --

C97 Withdraw DLS

Definition: A *Function* that supports the withdrawal of the DLS that implements the DL**.**

Relationships:

- *Withdraw DLS <isa> Manage DLS*

Rationale: This *Function* supports the removal of the DLS (and thus of the DL it is realising).

Examples: --

C98 Update DLS

Definition: A *Function* that supports the update of the DLS that implements the DL**.**

Relationships:

- *Update DLS <isa> Manage DLS*

Rationale: This *Function* supports the update of the DLS (and thus of the DL realised by it).

Examples: --

C99 Manage Architecture

Definition: This *Function* supports the overall management and configuration of *Architectural Components*.

Relationships:

- *Manage Architecture <isa> Manage DLS*
- *Manage Architectural Component <isa> Manage Architecture*
- *Configure Architectural Component <isa> Manage Architecture*
- *Deploy Architectural Component <isa> Manage Architecture*
- *Monitor Architectural Component <isa> Manage Architecture*

Rationale: This family of *Functions* supports the creation, configuration, update, deletion and monitoring of *Architectural Components*.

Examples: --

C100 Manage Architectural Component

Definition: A *Function* that supports the management of a *DLS Architectural Component*.

Relationships:

- *Manage Architectural Component <isa> Manage Architecture*

Rationale: This *Function* supports the creation, update and deletion of A*rchitectural Components* for the DLS.

Examples: --

C101 Configure Architectural Component

Definition: A *Function* that supports the configuration of a DLS *Architectural Component*.

Relationships:

- *Configure Architectural Component <isa> Manage Architecture*

Rationale: The model does not establish the way in which this *Function* is supported. For instance, the configuration of an *Architectural Component* can be performed by manually editing the configuration files as well as through a graphical configuration environment capable of guiding the *DL System Administrator* during this complex task and of verifying and maintaining the consistency of the configured aspects.

Examples: --

C102 Deploy Architectural Component

Definition: A *Function* that supports the deployment of a DLS *Architectural Component* on one or more *Hosting Nodes* and their start-up.

Relationships:

- *Deploy Architectural Component <isa> Manage Architecture*

Rationale: The deployment phase consists of assigning components to *Hosting Nodes* in order to ensure the quality values required by the *DL Designer*. As for the configuration functionality, the model does not restrict how this functionality is provided. Some DLMSs may offer sophisticated mechanisms for supporting a dynamic deployment while others may rely on manual deployment performed by the *DL System Administrator*.

Examples: --

C103 Monitor Architectural Component

Definition: A *Function* that keeps the *DL System Administrator* informed of the current status of a deployed DLS *Architectural Component*.

Relationships:

- *Monitor Architectural Component <isa> Manage Architecture*

Rationale: This *Function* relies on information about the status of the allocation of DLS *Architectural Components*. The behaviour of this *Function* can vary according to the information available and the level of automatic monitoring supported. For instance, it can provide a mechanism that allows the *DL System Administrator* to manually access component status. Alternatively, it can offer both a user interface graphically reporting the status of certain component characteristics and an automatic warning mechanism alerting the *DL System Administrator* when a certain characteristic of the deployed components exceeds an established threshold.

Examples: --

C104 Configure DLS

Definition: The class of *Functions* that enable selecting and configuring the entities that constitute a specific digital library, in the respective domain: i.e. *Content*, *User*, *Functionality*, *Quality* and *Policy* aspects.

Relationships:

- *Configure DLS <isa> Manage & Configure DLS*
- *Configure Resource Format <isa> Configure DLS*
- *Configure Content <isa> Configure DLS*
- *Configure User <isa> Configure DLS*
- *Configure Functionality <isa> Configure DLS*
- *Configure Policy <isa> Configure DLS*
- *Configure Quality <isa> Configure DLS*

Rationale: This class of *Functions* supports the DLS configuration.

Examples: --

C105 Configure Resource Format

Definition: The *Configure DLS* function that supports the definition of *Resource Format* with which the DL *Resources* must comply.

Relationships:

- *Configure Resource Format <isa> Configure DLS*

Rationale: This *Function* supports the *DL Designer* in defining the *Resource Formats* in terms of the general resource model that is desirable for the DL.

Examples: --

C106 Configure Content

Definition: The *Configure DLS function* supporting the *DL Designer* in configuring the Digital Library *Content Domain*.

Relationships:

- *Configure Content <isa> Configure DLS*

Rationale: This *Function* supports the personalisation of the contant domain aspects. In particular, by interacting with this *Function*, the *DL Designer* may configure the *Resource Format* of the class of *Information Objects* supported by the DL, the set of *Collections* forming the initial DL information space.

Examples: --

C107 Configure User

Definition: The *Configure DLS* function supporting the *DL Designer* in configuring the digital library *Actors* both in quantitative and qualitative terms.

Relationships:

- *Configure User <isa> Configure DLS*

Rationale: This *Function* supports the personalisation of the user domain related aspects. In particular, by interacting with this *Function*, an *Actor* may configure the *Actor Profile* formats, initialise the DL with *Actor* specialisations, initialise *Groups*, etc.

Examples: --

C108 Configure Functionality

Definition: The *Configure DLS* function supporting the *DL Designer* in configuring the digital library *Functionality Domain* both in quantitative and qualitative terms.

Relationships:

- *Configure Functionality <isa> Configure DLS*

Rationale: This *Function* takes as input a DL customisable functionality and assigns values to its parameters, thus selecting a specific configuration for the DL. It is obvious that the broader the range of customisations supported by a *Digital Library Management System*, the greater its capability to adapt to different scenarios.

Examples: --

C109 Configure Policy

Definition: The *Configure DLS* function supporting the *DL Designer* in setting up the DL *Policy Domain*.

Relationships:

- *Configure Policy <isa> Configure DLS*

Rationale: This *Function* is the highest-level *Function* with respect to the management of *Policies*, i.e. all the other *Functions* dealing with *Policies* are constrained by its choices and outcome. For instance, the *Manage Policy* domain is constrained by the values specified when invoking the *Establish Policies* function at DL design time.

Examples: --

C110 Configure Quality

Definition: The *Configure DLS* function supporting the *DL Designer* in describing the expected *Quality Parameters* of the digital library service.

Relationships:

- *Configure Quality <isa> Configure DLS*

Rationale: It is a key *Function* enabling the *DL Designer* to define the *Quality Parameters* of the system. In particular, it supports the initialisations of *Quality Parameters* and the selection of measurement units and processes for these parameters.

Examples: --

C111 Policy Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents a set of guiding principles designed to organise actions in a coherent way and to help in decision making.

Relationships:

- *Digital Library <definedBy> Policy Domain*
- *Digital Library System <definedBy> Policy Domain*

- *Digital Library Management System <definedBy> Policy Domain*
- *Policy Domain <consistOf> Policy*

Rationale: The term *Policy* usually describes a set of principles that describe the acceptable processes and/or procedures within an organisation. *Policy Domain* affects how the complete system is designed and how it functions. This means that *Policies* should be incorporated in the *Architecture Domain*, implemented in the *Functionality Domain* and should be clear to *Actors* as it affects their work with the *Content Domain* and influences their perception of the *Quality Domain*.

Within the three systems in the Digital Library universe, *Policy Domain* plays different roles.

From the *Digital Library* perspective, *Policies* mean conditions, rules, terms and regulations governing the interaction between *Actors* and the *Digital Library*. They provide mechanisms to constrain operations that *Actors* may/may not perform at the *DL* on individual *Resources* or at the level of *Digital Library* at a given time.

Policies within this system reflect the management goals of the institution providing the *Digital Library* and should influence, rather than be influenced by, technical architecture, functionality, quality or information content.

From the *Digital Library System* perspective, *Policy* is the provision of the capability to define *Policies* and enforce them. The *Digital Library System* provides formal mechanisms for defining *Policies* and ensuring that they are effectively enforced.

From the *Digital Library Management System* perspective, the emphasis is on the capabilities to implement the elements of the *Policy Domain* that underpin a digital library model.

Building Digital Library *Policies* is a complicated task, as they must serve the needs of institutions of various types and sizes that work together in a continuously evolving distributed environment.

Policies exist at different levels: some ensure the effective functioning of the organisation that manages the DL and others relate more directly to *Actor* services and how they are provided and accessed. They make manifest operational expectations in such areas as: collection development and management guidelines; human resource policies; space use policies; confidentiality practices; user registration and enrolment, library card and borrowing policies; and service use policies, e.g. acceptable user behaviour.

While *Policy Domain* is a general term, specific aspects within a single area are covered by *Policy* and are manifested through a document which usually consists of policy statement, rationale, enforcement, responsible office (*Policy <expressedBy> Information Object*).

Examples: --

C112 Policy

Definition: A condition, rule, term or regulation governing the operation of a *Digital Library*.

Relationships:

- *Policy <isa> Resource*
- *Resource <regulatedBy> Policy*
- *Actor <regulatedBy> Policy*
- *Function <regulatedBy> Policy*
- *Policy <expressedBy> Information Object*
- *System Policy <isa> Policy*
- *Content Policy <isa> Policy*

- *User Policy <isa> Policy*
- *Functionality Policy <isa> Policy*
- *Enforced Policy <isa> Policy*
- *Voluntary Policy <isa> Policy*
- *Explicit Policy <isa> Policy*
- *Implicit Policy <isa> Policy*
- *Extrinsic Policy <isa> Policy*
- *Intrinsic Policy <isa> Policy*
- *Descriptive Policy <isa> Policy*
- *Prescriptive Policy <isa> Policy*

Rationale: A *Policy* regulates *Actors* consuming *Resources* through *Functions* with respect to a validity interval (*Time Domain* can be used here). *Policy* has a specific area coverage, for example *Registration Policy* or *Preservation Policy*.

Policy may be descriptive (e.g. Collection Development Policy, which explains what the content of the collection is and how it will be developed in future) or prescriptive (there are strict procedures to follow, e.g. *Registration Policy*).

The currently identified *Policy* entities should be considered as examples; they are at present the most important in digital libraries.

Examples:

- *Privacy and Confidentiality Policy* is a *Policy* that describes what rules are followed to assure the privacy and confidentiality of the *Actors*. This is seen as a part of the DL system.
- The same *Policy* within the *Digital Library System* is seen as the specification of what *Functions* should be present, and in the *Digital Library Management System* refers to the practical implementation of the *Functions*.

C113 Extrinsic Policy

Definition: A *Policy* defined outside, and applied within, the DL.

Relationships:

- *Extrinsic Policy <isa> Policy*
- *Extrinsic Policy* is *<antonymOf> Intrinsic Policy*
- *Extrinsic Policy <isa> Policy by context*

Rationale: *Extrinsic Policy* is a *Policy* imposed by a body outside the Digital Library (e.g. legal and regulatory frame works). According to the type of DL, the regulatory framework might differ – a DL in the pharmaceutical arena will operate in a very different regulatory framework from one in the area of tourism.

Examples:

- Legal and regulatory frameworks of a specific country applied to a Digital Library developed by a local body.

C114 Intrinsic Policy

Definition: A *Policy* defined inside, and applied within, the DL.

Relationships:

- *Intrinsic Policy <isa> Policy*

- *Intrinsic Policy* is *<antonymOf> Extrinsic Policy*
- *Intrinsic Policy <isa> Policy by context*

Rationale: *Intrinsic Policy* manifests the *Policy* principles implemented in the DL.

A *Policy* that is defined by the DL or its organisational context that reflects the organisation's mission and objectives, the intended expectations as to how *Actors* will interact with the DL, and the expectations of *Content Creators* as to how their content will be used.

Examples:

- A *Policy* within the *Policy* of the respective Digital Library is an *Intrinsic Policy*.

C115 Explicit Policy

Definition: A *Policy* that has been stated and approved.

Relationships:

- *Explicit Policy <isa> Policy*
- *Explicit Policy* is *<antonymOf> Implicit Policy*
- *Explicit Policy <isa> Policy by expression*

Rationale: *Explicit Policy* is a *Policy* defined by the DL managing organisation and reflecting the objectives of the DL and how it wishes its users to interact with the DL. The implementation of an *Explicit Policy* at the Digital Library Management System level corresponds to the definition and *Actor* expectations.

Examples:

- Limitation for upload of files over a specified size, e.g. over 1 MB, which is clearly stated at the user interface in addition to the explanation within the text of the *Submission and Resubmission Policy*.

C116 Implicit Policy

Definition: A *Policy* that is inherent in the DL either through accident of design or undocumented development decisions, but was not explicitly planned or stated.

Relationships:

- *Implicit Policy <isa> Policy*
- *Implicit Policy* is *<antonymOf> Explicit Policy*
- *Implicit Policy <isa> Policy by expression*

Rationale: *Implicit Policies* usually arise as a result of ad-hoc decisions taken at system development level or as a consequence of the inadequate testing of a DLS that results in an interaction of *Policies* leading to unintended policy deployment.

This is an illustration of how improper actions at *Digital Library System* level or *Digital Library Management System* level can have consequences for the DL.

Implicit Policies should be avoided as they tend to be opaque, have unintended and unexpected consequences which impact on the interaction of all *Actor* communities with the DL.

Examples:

- An implemented – but not communicated to the *Actors* – limitation in the file size while uploading or downloading resources from the *Digital Library* is an example of *Implicit Policy*.

C117 Prescriptive Policy

Definition: A *Policy* that constrains or manages interactions between DL *Actors* (virtual or real) and the DL.

Relationships:

- *Prescriptive Policy <isa> Policy*
- *Prescriptive Policy <isa> Policy by application*

Rationale: *Prescriptive Policies* can cover a broad range of *Policies* from the kind of *Function* to which specific types of *Actors* can have access, to those that govern *Collection* development.

Examples:

- Termination of file upload, if the file is of a format that is not permitted, is an example of action taken as a result of a *Prescriptive Policy*.

C118 Descriptive Policy

Definition: A *Policy* that provides explanation on a certain *Policy*.

Relationships:

- *Descriptive Policy <isa> Policy*
- *Descriptive Policy <isa> Policy by application*

Rationale: *Descriptive Policies* are used to present the aspects of a particular *Policy* in the form of explanation. A *Descriptive Policy* is a *Policy* that describe modes of behaviour, expectations of *Actor* interaction, collecting and use guidelines, but which do not manifest themselves through the automated application of rules, as a *Prescriptive Policy* does.

Examples:

- The *Collection Development Policy* describes the scope and coverage of the DL.

C119 Enforced Policy

Definition: A *Policy* that is deployed and strictly applied within the DL.

Relationships:

- *Enforced Policy <isa> Policy*
- *Enforced Policy <isa> Policy by compliance*

Rationale: An *Enforced Policy* is a *Policy* developed, deployed and strictly used in the DL. Monitoring and reporting tools are necessary to follow up how the *Policy* is being applied.

Examples:

- A *Charging Policy*, which had been introduced into the DL, is an *Enforced Policy*.

C120 Voluntary Policy

Definition: A *Policy* that is either not deployed within the DL, or which might be followed by the *Actor* through his own choice.

Relationships:

- *Voluntary Policy <isa> Policy*
- *Voluntary Policy <isa> Policy by compliance*

Rationale: *Voluntary Policy* basically means a *Policy* that is followed according to the decision of the *Actor*. This is valid for all *Policies* for which application is a matter of choice. In some cases, users may comply with *Policies* that are not officially communicated within

the particular digital library – perhaps based on their previous experience with other digital libraries.

Examples:

- The *Collection Development Policy* might be outlined in broad terms, but not enforced in practice.

C121 System Policy

Definition: A *Policy* that concerns an aspect of a system as a whole, be it a *Digital Library*, a *Digital Library System* or a *Digital Library Management System*

Relationships:

- *System Policy <isa> Policy*
- *Change Management Policy <isa> System Policy*
- *Connectivity Policy <isa> System Policy*
- *Support Policy <isa> System Policy*
- *Resource Management Policy <isa> System Policy*

Rationale: This is a class of *Policies* governing generic processes within the digital library system in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- *System Policies* cover most general processes in the digital library, such as regulation of changes or management of resources.

C122 Change Management Policy

Definition: The purpose of the *Change Management Policy* is to regulate how changes are being carried out on the three levels and within the six domains of a digital library in a rational and consistent manner that would be effectively communicated to the *Actors* and would not harm their routine work.

Relationships:

- *Change Management Policy <isa> Policy*
- *Resource <regulatedBy> Change Management Policy*
- *Change Management Policy <govern> Manage DL Function*
- *Change Management Policy <govern> Manage DLS Function*
- *Change Management Policy <govern> Manage Resource*

Rationale: The aim of *Change Management Policy* in the DL is to ensure stability in the process of restructuring and assure coherence of actions on the three levels (DL, DLS and DLMS). The complexity of the DL could be approached when, in the process of change management, the issues relevant to the six basic areas – *Information Objects*, *Actors*, *Policies*, *Quality Parameters*, *Architectural Components*, *Functions* – and the three levels (DL, DLS and DLMS) are addressed in a rational and consistent manner.

It is of the utmost importance to define roles and responsibilities in change management, and to consider in detail the change management process and the support the DLS and DLMS should provide for its smooth execution.

Examples:

- *Quality Parameter Measures* that demonstrate the change management progress may be part of the *Change Management Policy*.

C123 Resource Management Policy

Definition: *Policies* defining how *Resources* in the DL are allocated.

Relationships:

- *Resource Management Policy <isa> Policy*
- *Resource Management Policy <isa> System Policy*
- *Resource Management Policy <govern> Resource*

Rationale: Resource management is a key area within the organisation and use of *Resources* in the DL. *Resource Management Policy* is the *Policy* that describes the principles and procedures related to this field.

Since *Resources* may be of a different nature, this *Policy* would usually be a combination of different actions and procedures.

Examples:

- Checking the consistency of *Resource Identifiers* may be a task from the *Resource Management Policy*.

C124 Support Policy

Definition: *Policies* describing the kinds of support *Actors* can expect when using the DL system and the *Resources* it contains.

Relationships:

- *Support Policy <isa> Policy*
- *Support Policy <isa> System Policy*
- *Support Policy <isa>* (should be) *Explicit Policy*
- *Support Policy <isa>* (should be) *Descriptive Policy*
- *Support Policy <isa>* (should be) *Intrinsic Policy*
- *Support Policy <govern> Actor*
- *Resource <regulatedBy>Support Policy*

Rationale: *Support Policy* refers to the technical and educational support on issues arising from the exploitation of a *Digital Library Management System*. In this case, the *Support Policy* should clearly describe what services are offered. Sometimes it is also helpful to include a list of excluded services.

Support Policy should be explicit (*Explicit Policy*), descriptive (*Descriptive Policy*) and intrinsic (*Intrinsic Policy*).

In some circumstances, policies related to support might be prescriptively enforced (*Prescriptive Policy* and *Enforced Policy*).

The procedure to be followed in order to request a service, and the conditions for its provision (charges, prioritising in the case of several simultaneous requests, and timing) should be clearly expressed in the *Support Policy*.

Examples:

- Priorities (critical requests receive higher priority than standard requests) may be a component of *Support Policy*.

C125 Connectivity Policy

Definition: *Policy* assuring maximum access to DL *Resources*.

Relationships:

- *Connectivity Policy <isa> Policy*
- *Connectivity Policy <isa> System Policy*
- *Connectivity Policy <govern> Actor*

Rationale: Connectivity can be defined as an organisation's capacity for communicating with itself and with its global environment through the use of ICT.

Connectivity Policy should promote all means that enable *Actors* from various environments to access *Resources*. The DL should be accessible via various communication channels, including mobile devices.

Examples:

- The digital divide is one of the threats that can be addressed within the *Connectivity Policy.*

C126 Risk Management Policy

Definition: A *Policy* that explains the approach within the DL towards various identified risks, the likelihood of their occurrence and the strategy for risk management.

Relationships:

- *Risk Management Policy <isa> System Policy*
- *Risk Management Policy <isa> Intrinsic Policy*

Rationale: This *Policy* should identify and provide an evaluation of, and correcting actions for, the risks within the six DL domains.

Examples:

- Good risk management would contribute to increasing *Quality Parameters* and in particular the *Trustworthiness* of the DL.

C127 Content Policy

Definition: *Policy* regulating the *Content* domain.

Relationships:

- *Content Policy <isa> Policy*
- *Disposal Policy <isa> Content Policy*
- *Collection Delivery Policy <isa> Content Policy*
- *Collection Development Policy <isa> Content Policy*
- *Digital Rights Management Policy <isa> Content Policy*
- *Preservation Policy <isa> Content Policy*
- *Submission and Resubmission Policy <isa> Content Policy*

Rationale: This is a class of *Policies* that govern processes related to the *Content* domain within the Digital Library 'system' in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- The issues of strategic planning and development of the *Content* of a Digital Library are addressed in the *Collection Development Policy*.

C128 Disposal Policy

Definition: *Policy* concerning de-accession of DL material.

Relationships:

- *Disposal Policy <isa> Policy*

- *Disposal Policy <isa> Content Policy*
- *Disposal Policy <isa>* (may be a) *Prescriptive Policy*
- *Disposal Policy <isa>* (may be a) *Descriptive Policy*
- *Disposal Policy <govern> Actor*

Rationale: *Policies* defining de-accession of DL material from their collections. They are both prescriptive (*Prescriptive Policy*) and descriptive (*Descriptive Policy*).

Examples:

- De-accession of a *Resource* that has not been requested for a certain period of time is part of a *Disposal Policy*.

C129 Collection Development Policy

Definition: *Policy* presenting the current *Content* and the intentions for further development of the DL.

Relationships:

- *Collection Development Policy <isa> Policy*
- *Collection Development Policy <isa> Content Policy*
- *Resource <regulatedBy> Collection Development Policy*
- *Information Object <regulatedBy> Collection Development Policy*
- *Collection <regulatedBy> Collection Development Policy*

Rationale: The institution(s) taking care of the DL development make their vision on the further development of the DL publicly available through their *Collection Development Policy*.

These intentions may reflect different aspects – for example, number of *Resources*, *Resource sets*, *Collections*. *Collection Development Policy* can also affect issues of subject areas, genres, data formats, or services related to better use of the *Collection* and adding value to its *Content*.

Basically, they should describe:

- access to what *Resources* are provided and how *Resources* will be enriched over time – in the short-, mid- and long-term future.
- information on formats, encodings, and recommendations for use of software tools for consulting *Resources* (*Resource Formats*).
- guidance on handling or tracking new *Editions*.

Collection Development Policies are of help to *Actors* in comparing different DLs in terms of what they offer and how relevant they are to their purposes.

Collection Development Policies can assist institutions developing DLs as they help to find and demonstrate the unique standing of particular DLs.

The *Collection Development Policy* text (*Policy <expressedBy> Information Object*) usually describes categories of *Resources*, selection criteria, goals, priorities, services and accessibility.

Examples:

- Estimation of current coverage of a DL is part of the *Collection Development Policy*.

C130 Collection Delivery Policy

Definition: *Policy* encompassing the constraints affecting how *Collections* will be delivered, under what conditions and for what purposes.

Relationships:

- *Collection Delivery Policy <isa> Policy*
- *Collection Delivery Policy <isa> Content Policy*
- *Collection Delivery Policy <govern> Actor*
- *Resource <regulatedBy> Collection Delivery Policy*
- *Collection Delivery Policy <govern> Browse Function*
- *Collection Delivery Policy <govern> Visualise Function*

Rationale: *Collection Delivery Policy* covers methods of providing access to the DL – through Internet services, removable memory, stand-alone computers, mobile devices, print on demand services.

The *Collection Delivery Policy* should also specify the conditions for the delivery (free of charge or paid; conditions for purchase of single items or through use licenses).

The *Collection Delivery Policy* may also specify the acceptable uses of *Resources*.

Examples:

- Purchasing a DVD with selected *Resources*.
- Offering a print-on-demand service for selected *Resources*.
- Defining the conditions for commercial use of images from illuminated manuscripts.
- Announcing free use of material for education and research purposes.

C131 Submission and Resubmission Policy

Definition: *Policies* regulating submission and resubmission of *Resources* to the DL.

Relationships:

- *Submission and Resubmission Policy <isa> Policy*
- *Submission and Resubmission Policy <isa> Content Policy*
- *Submission and Resubmission Policy <isa> Explicit Policy*
- *Submission and Resubmission Policy <isa> Prescriptive Policy*
- *Submission and Resubmission Policy <isa> Intrinsic Policy*

Rationale: *Submission and Resubmission Policies* govern which *Actors* can submit and resubmit *Information Objects* to the DL.

They should be explicit (*Explicit Policy*), prescriptive (*Prescriptive Policy*) and intrinsic (*Intrinsic Policy*).

Time constraints may be part of a *Submission and Resubmission Policy*.

Examples:

- *Actors* may have the right to submit but not to edit *Resources*.

C132 Digital Rights Management Policy

Definition: *Policy* that explains what technologies control how *Conten*t is used within the DL.

Relationships:

- *Digital Rights Management Policy <isa> Policy*
- *Digital Rights Management Policy <isa> Content Policy*
- *Digital Rights Management Policy <isa> User Policy*
- *Digital Rights Management Policy <isa> Functionality Policy*
- *Digital Rights Management Policy <govern> Function*
- *Digital Rights Management Policy <govern> Configure User Function*
- *Digital Rights Management Policy <govern> Digital Rights*

Rationale: Digital rights management (DRM) is the technological framework which should guarantee persistent access and use restrictions to *Resources*. *Digital Rights Management Policy* is the *Policy* explaining how the DL manages digital rights from the perspective of both the content creator/originator/owner and the *Actor*, and which technologies control the use of *Content* within the DL. While DRM regulates the types of actions that can be performed with information (for example, view, save, print, modify certain *Manifestation*), *Digital Rights Management Policy* explains DRM.

Digital rights management has been developed so that copyright holders on digital content have exclusive rights of copyright (e.g. the right to make a copy or the right to distribute a work to the public). Copyright holders, however, cannot control how digital content is used (e.g. the right to view, save, print, read or modify a work). Traditional library materials are better protected from unauthorised use because of their 'physical' nature. The development of digital content along with electronic publishing and the Internet, which gives access to *Manifestation* of the *Resources* in the digital environment, is creating new issues in the area of copyright regulations.

Restrictions within *Digital Rights Management Policy* may depend on the *Actor*. Some restrictions may be time-dependent.

From the *Digital Library* perspective, *Digital Rights Management Policy* means conditions, rules, terms and regulations governing the interaction between *Actors* (virtual or real) and the DL in all cases where copyright or other rights on *Resources* apply.

From the *Digital Library System* perspective, *Digital Rights Management Policy* is the provision of the capability to define copyright-related policies.

From the *Digital Library Management System* perspective, the emphasis is on the capabilities to implement the elements of the *Digital Rights Management Policy*. This means that the system should be capable of tracing certain actions undertaken by the user and of reacting correctly.

Examples:

- The *Actors* belonging to a specific *Group* can view *Resources* but not save local copies.
- Viewing *Resources* expires within a specific period of time.

C133 Digital Rights

Definition: *Policy* defining the rights of use of *Information Objects*.

Relationships:

- *Digital Rights <isa> Policy*
- *Digital Rights <isa> Content Policy*
- *Digital Rights <isa> Descriptive Policy*
- *Digital Rights <govern> Information Object*

Rationale: *Digital Rights* define the specific rights of use of digital objects. In this sense, they are a *Descriptive Policy* regulating the possible uses of *Information Objects*. The practical implementation of the *Digital Rights* falls within *Digital Rights Management Policy*.

A broader understanding of *Digital Rights* defines them as all human rights that are affected by technology, including the rights to use computers, communication networks and resources.

Examples:

- The right to access knowledge is affected by digital technology and not all people have equal opportunities in this respect.
- The right to use without a license is another example.

C134 License

Definition: A *Policy* regulating the exploitation of a *Resource*.

Relationships:

- *License <isa> Policy*
- *License <isa> Digital Rights Management Policy*
- *Resource <regulatedBy> License*
- *License <grantedTo> Actor*

Rationale: A *License* is the agreement by which the owner of intellectual property permits its use. In digital libraries, a *License* may be issued for specific uses of *Resources*, or for designated functionality features that should be downloaded and installed by the users.

Examples:

- GPL (GNU General Public License), a popular license for free software; GNU LGPL (Lesser General Public License); and BSD (Berkeley Software Distribution or Berkeley System Distribution) are examples of software licenses.

C135 Preservation Policy

Definition: *Policy* defining the approach to preservation taken by the DL.

Relationships:

- *Preservation Policy <isa> Policy*
- *Preservation Policy <isa> Content Policy*
- *Resource <regulatedBy> Preservation Policy*

Rationale: *Preservation Policy* prescribes how to implement actions assuring long-term preservation of *Resources*, such as decision making on archival needs, archiving practices, timing issues, access to archived materials, subsequent preservation measures for already archived materials, maintaining preservation metadata, issues of interoperability of preserved materials.

Examples:

- Reuse of preserved materials is part of the *Preservation Policy*.

C136 User Policy

Definition: *Policy* regulating the *User domain*.

Relationships:

- *User Policy <isa> Policy*
- *Digital Rights Management Policy <isa> User Policy*

- *User Management Policy <isa> User Policy*
- *Acceptable User Behaviour Policy <isa> User Policy*
- *Personalisation Policy <isa> User Policy*
- *Privacy and Confidentiality Policy <isa> User Policy*
- *Access Policy <isa> User Policy*

Rationale: This is a class of *Policies* governing processes related to the *User domain* within the Digital Library 'system' in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- All *Policies* that regulate issues regarding digital rights and user behaviour.

C137 User Management Policy

Definition: *Policy* defining how user management is handled.

Relationships:

- *User Management Policy <isa> Policy*
- *User Management Policy <isa> User Policy*
- *User Management Policy <govern> Actor*

Rationale: The *User Management Policy* makes it possible to execute *Functions* such as issuing, managing, changing, sharing accounts; administration rights; sharing resources between multiple users.

Examples:

- Account management is part of the *User Management Policy*.

C138 Registration Policy

Definition: *Policy* describing the information that is required for *Actors*, human and machine, to register with the DL and how this information is validated, managed and maintained.

Relationships:

- *Registration Policy <isa> Policy*
- *Registration Policy <isa> User Management Policy*
- *Registration Policy <govern> Actor*
- *Registration Policy <govern> Login Function*
- *Registration Policy <govern> Subscribe Function*

Rationale: This *Policy* explains how virtual and human users should register in order to use the DL.

The *DLMS* should perform functions on user log-in, validation, management and maintenance.

Examples:

- Storage of sessions and IP addresses is an element from the *Registration Policy*.

C139 Personalisation Policy

Definition: *Policy* enabling the DL to define what kinds of personalisation will be allowable and under what circumstances.

Relationships:

- *Personalisation Policy <isa> Policy*

- *Personalisation Policy <isa> User Policy*
- *Personalisation Policy <govern> Actor*
- *Personalisation Policy <govern> Personalise Function*

Rationale: The *Personalisation Policy* has two roles; on the one hand it makes it possible to recognise the user and his/her access rights, and on the other hand it enables the DL to serve its *Actors*, guaranteeing better *Quality Parameters* by offering *Information Objects* (generally *Resources*) that are in line with user preferences. In the DLS the *Functions* used to assure personalisation are *Apply Profile*, *Customise*, *Login* and *Subscribe*.

Examples:

- The choice of representation layout based on statistics of user behaviour is an example of *Personalisation Policy*.

C140 Privacy and Confidentiality Policy

Definition: A *Policy* outlining the terms by which the organisation that manages the DL will handle personal information on its *Actors*.

Relationships:

- *Privacy and Confidentiality Policy <isa> Policy*
- *Privacy and Confidentiality Policy <isa> User Policy*

Rationale: *Policies* prescribing *Actor* details from application and enrolment information through to actor interaction data will be handled by the DL and the organisation that manages the DL.

Typically, the DL should only maintain personal information on *Actors* that is relevant to its better functioning and services.

Data about the *Actors* could be entered directly by them (e.g. user names, passwords) or obtained automatically (e.g. IP address).

The personal data collected should be protected against unauthorised access, destruction, misuse, modification, improper disclosure and loss.

Different rules may be applied to the use of various types of personal information, e.g. e-mail addresses, postal address, log-in names and passwords, users' opinions entered via web pages.

Privacy and Confidentiality Policy principles should be embedded in the DL *Functions* that require collection of data about the *Actors* (supplied or automatically collected).

Examples:

- The use of the e-mail addresses of the *Actors* to announce new DL collections may be justified as a part of the *Privacy and Confidentiality Policy*.
- Selling or sharing with other organisations lists of e-mail addresses of the *Actors* is typically not in line with the *Privacy and Confidentiality Policy*, unless the users have agreed to this.

C141 Acceptable User Behaviour Policy

Definition: *Policy* covering how the *Actors* may or may not interact with the DL.

Relationships:

- *Acceptable User Behaviour Policy <isa> Policy*
- *Acceptable User Behaviour Policy <isa> User Policy*

Rationale: *Acceptable User Behaviour Policy* presents rules and regulations for appropriate use of the DL content and services, prescribing what a user can do and what he/she should refrain from doing.

Examples:

- Regulations on copying material from a DL are part of the *Acceptable User Behaviour Policy*.
- Rules for citation of the source of material from a DL are part of the *Acceptable User Behaviour Policy*.
- Rules on downloading images of workstations for within-institutional use of a DL are part of the *Acceptable User Behaviour Policy*.

C142 Functionality Policy

Definition: *Policy* regulating the *Functionality* domain.

Relationships:

- *Functionality Policy <isa> Policy*
- *Access Policy <isa> Functionality Policy*
- *Security Policy <isa> Functionality Policy*

Rationale: This is a class of *Policies* governing processes related to the *Functionalit*y domain within the Digital Library 'system' in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- Taking care of the security of the Digital Library is a serious concern, for which the practical implementation would be a *Security Policy*.

C143 Access Policy

Definition: *Policy* regulating permission or denial of use of *Resources* by *Actors* in the *Digital Library* 'system'.

Relationships:

- *Access Policy <isa> Policy*
- *Access Policy <isa> User Policy*
- *Access Policy <isa> Functionality Policy*
- *Charging Policy <isa> Access Policy*

Rationale: *Access Policy* regulates the use of digital library *Resources* (permission or denial of use) by *Actors*. It should guarantee that *Resources* are accessed by their intended *Actors* and not by others who might harm them unintentionally or deliberately. *Access Policy* belongs to both *Functionality* and *User* domains, as on the one hand it prescribes what *Functions* are possible, and on the other hand regulates the work of the *Actors*.

Examples:

- Access to *Resources* provided on the basis of IP address identification is an example of *Access Policy*.

C144 Charging Policy

Definition: *Policy* defining how charging schemes will be implemented and handled by the DL.

Relationships:

- *Charging Policy <isa> Policy*

- *Charging Policy <isa> Access Policy*
- *Charging Policy <govern> Actor*

Rationale: The *Charging Policy* explains what mechanisms are applied for collecting payments.

There are various models that could be applied: services provided on the basis of a longer time period; micro-payments; exchange of use of content for uploading user's own content into the DL.

Examples:

- Institution has unlimited access to all high-quality images stored in a DL based on an annual fee. *Actors* not coming from such an institution only have access to low-quality images.

C145 Security Policy

Definition: *Policy* regulating how a system provides security and protects *Resources* within the DL.

Relationships:

- *Security Policy <isa> Policy*
- *Security Policy <isa> Functionality Policy*
- *Resource <regulatedBy> Security Policy*

Rationale: *Security Policies* address the protection of the Digital Library. They implement the rules and tools that assure the security of services and integrity of the Digital Library.

Examples:

- Ingest of *Resources* into the library on the basis of virus checking is an example of *Security Policy*.

C146 Quality Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents the various aspects related to features and attributes of *Resources* with respect to their degree of excellence.

Relationships:

- *Digital Library <definedBy> Quality Domain*
- *Digital Library System <definedBy> Quality Domain*
- *Digital Library Management System <definedBy> Quality Domain*
- *Quality Domain <consistOf> Quality Parameters*

Rationale: The *Quality Domain* concept represents the various facets used to characterise, evaluate and measure *Digital Libraries*, *Digital Library Systems*, *Digital Library Management Systems* and their *Resources*. *Digital Library, Digital Library System* and *Digital Library Management System <tenders>* a certain level of *Quality Parameters* to its *Actors*, which can be either implicitly agreed or explicitly formulated by means of a Quality of Service (QoS) agreement.

Examples: --

C147 Measure

Definition: A process for computing and assigning a value to a *Quality Parameter* according to a unit of measurement.

Relationships:

- *Quality Parameter <evaluatedBy> Measure*
- *Subjective Measure <isa> Measure*
- *Objective Measure <isa> Measure*
- *Qualitative Measure <isa> Measure*
- *Quantitative Measure <isa> Measure*

Rationale: See *Quality Parameter*.

Examples:

- See Quality Parameter.

C148 Objective Measure

Definition: A *Measure* obtained via a well-defined process that does not depend on individual perception.

Relationships:

- *Objective Measure <isa> Measure*

Rationale: *Objective Measures* could be obtained by taking measurements and using an analytical method to estimate the quality achieved. They could also be based on processing and comparing measurements between a reference sample and the actual sample obtained by the system.

The distinction between *Objective Measure* and *Subjective Measure* is due to the fact that *Quality Parameters* can involve measurement methods that can either be independent of the subject who is conducting them or, on the other hand, express the viewpoint and perception of the subject.

Examples:

- Examples of objective factors related to the perception of audio recordings in a *Digital Library* are: noise, delay and jitter.

C149 Subjective Measure

Definition: A *Measure* based on, or influenced by, personal feelings, tastes or opinions.

Relationships:

- *Subjective Measurement <isa> Measure*

Rationale: *Subjective Measures* involve performing opinion tests, user surveys and user interviews which take into account the inherent subjectivity of the perceived quality and the variations between individuals. The perceived quality is usually rated by means of appropriate scales, where the assessment is often expressed in a qualitative way using terms such as bad, poor, fair, good, excellent to which numerical values can be associated to facilitate further analyses.

The distinction between *Objective Measure* and *Subjective Measure* is due to the fact that *Quality Parameters* can involve measurement methods that can either be independent of the subject who is conducting them or, on the other hand, express the viewpoint and perception of the subject.

Examples:

- Examples of factors related to the subjective perception of audio recordings in a *Digital Library* are: listening quality, loudness, listening effort.

C150 Qualitative Measure

Definition: A *Measure* based on a unit of measurement that is not expressed via numerical values.

Relationships:

- *Qualitative Measure <isa> Measure*

Rationale: *Qualitative Measures* are applied when the collected data are not numerical in nature. Although qualitative data can be encoded numerically and then studied by quantitative analysis methods, qualitative measures are exploratory while quantitative measures usually play a confirmatory role. Methods of *Qualitative Measure* that could be applied to a DL are direct observation; participant observation; interviews; auditing; case study; collecting written feedback.

Examples:

- The opinions of the users expressed in a DL forum or blog can be used as a source for *Qualitative Measure* of important issues for the users (content analysis is one of the popular techniques for analysing texts).

C151 Quantitative Measure

Definition: A *Measure* based on a unit of measurement that is expressed via numerical values.

Relationships:

- *Quantitative Measure <isa> Measure*

Rationale: *Quantitative Measures* are based on collecting and interpreting numerical data. There is a wide range of statistical methods for their analysis.

Examples:

- *Quantitative Measure* is applied when collecting data and calculating the mean time spent by users in locating content.

C152 Measurement

Definition: The action of, and the value obtained by, measuring a *Quality Parameter* in accordance with a selected *Measure*.

Relationships:

- *Quality Parameter <measuredBy> Measurement*
- *Measurement* is assigned according to (*<accordTo>*) a *Measure*

Rationale: See *Quality Parameter*.

Examples:

- See Quality Parameter.

C153 Quality Parameter

Definition: A *Resource* that indicates, or is linked to, performance or fulfilment of requirements by another *Resource*. A *Quality Parameter* is evaluated by (*<evaluatedBy>*) a *Measure*, is *<measuredBy>* a *Measurement*, and expresses the assessment (*<expressAssessment>*) of an *Actor*.

Relationships:

- *Quality Parameter <isa> Resource*
- *Quality <expressedBy> Quality Parameters*

- *Resource <hasQuality>* with respect to *Quality Parameter*
- *Actor <expressAssessment>* about *Resources* according to *Quality Parameters*
- *Quality Parameter <evaluatedBy> Measure*
- *Quality Parameter <measuredBy> Measurement*
- *Quality Parameter <affectedBy> Resource*
- *Generic Quality Parameter <isa> Quality Parameter*
- *Content Quality Parameter <isa> Quality Parameter*
- *Functionality Quality Parameter <isa> Quality Parameter*
- *User Quality Parameter <isa> Quality Parameter*
- *Policy Quality Parameter <isa> Quality Parameter*
- *Architecture Quality Parameter <isa> Quality Parameter*

Rationale: *Quality Parameters* serve the purpose of expressing the different facets of *Quality* and provide information about how and how well a *Resource* performs with respect to a particular viewpoint. They express the assessment of an *Actor*, be it human or not, about the *Resource* under examination. They can be evaluated according to different *Measures*, which provide alternative procedures for assessing different aspects of a *Quality Parameter* and assigning it a value. *Quality Parameters* are actually measured by a *Measurement*, which represents the value assigned to a *Quality Parameter* with respect to a selected *Measure*.

Note that the *Resource* under examination in a *Quality Parameter* can be either a singleton *Resource*, as in the case of the *Integrity* of an *Information Object*, or a *Resource Set*, as in the case of the *Orthogonality* of a set of *Functions*.

Finally, a *Quality Parameter* may be affected by other *Resources*, such as other *Quality Parameters*, *Policies* or *Functions*; this allows us to create a 'chain' of *Resources* which leads to the determination of the *Quality Parameter* in question. For example, *Availability* is affected by *Robustness* and *Fault Management*: in fact, when a *Function* is both robust and able to recover from error conditions, it is probable that its *Availability* is also increased. As a further example, *Economic Convenience* may be affected by *Charging Policy*, since the latter is responsible for the definition of the charging strategies.

Note that, being a *Resource*, a *Quality Parameter* may have *Metadata* and *Annotations* linked to it; the former can provide useful information about the provenance of a *Quality Parameter*, while the latter can offer the possibility to add comments about a *Quality Parameter*, interpreting the obtained values, and proposing actions to improve it.

Please note that the groupings of *Quality Parameters* in broad categories, such as *Content Quality Parameter*, are made from the perspective of the *Resources* under assessment, in the case of the example mainly *Information Objects*. This means that *User Content Parameter* does not concern issues such as *User Satisfaction* or *Usability*, where the *Actor* is the subject who makes the assessment, but in this group the *Actor* is the object of the assessment from different points of view, such as *User Behaviour*. Nevertheless, the active role of an *Actor* in expressing an assessment is always preserved in the *Quality Parameter* by the fact the *Actor <expressAssessment>* about a *Resource* in each *Quality Parameter*.

The definition of *Quality Parameter* complies with the notion of quality dimension used in [20] and [98].

Examples:

- In order to clarify the relationship between *Quality Parameter*, *Measure* and *Measurement*, we can take an example from the information retrieval field. One of the

main *Quality Parameters* in relation to an information retrieval system is its effectiveness, meaning its capability to answer user information needs with relevant items. This *Quality Parameter* can be evaluated according to many different *Measures*, such as precision and recall [180]: precision evaluates effectiveness in the sense of the ability of the system to reject useless items, while recall evaluates effectiveness in the sense of the ability of the system to retrieve useful items. The actual values for precision and recall are *Measurements* and are usually computed using standard tools, such as trec_eval,[22] which are *Actors*, but in this case not human.

C154 Generic Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of a 'system' as a whole, be it a *Digital Library*, a *Digital Library System* or a *Digital Library Management System*.

Relationships:

- *Generic Quality Parameter <isa> Quality Parameter*
- *Reputation <isa> Generic Quality Parameter*
- *Economic Convenience <isa> Generic Quality Parameter*
- *Sustainability <isa> Generic Quality Parameter*
- *Security Enforcement <isa> Generic Quality Parameter*
- *Interoperability Support <isa> Generic Quality Parameter*
- *Documentation Coverage <isa> Generic Quality Parameter*
- *Performance <isa> Generic Quality Parameter*
- *Scalability <isa> Generic Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the 'system' in its entirety, in particular the *Digital Library*, the *Digital Library System* and the *Digital Library Management System*.

Examples: --

C155 Economic Convenience

Definition: A *General Quality Parameter* reflecting how favourable the economic efficiency is when using a *Digital Library*.

Relationships:

- Economic Convenience *<isa> Generic Quality Parameter*
- Economic Convenience *<affectedBy> Charging Policy*

Rationale: This parameter evaluates the economic conditions for using the *Digital Library* in order to determine if they are sufficiently advantageous.

There are various appraisal methods that can be applied: for example, comparing the economic conditions offered with market rates for similar services, evaluating the possibility of obtaining value-added services in the case of longer subscriptions, or assessing the flexibility of the offering with respect to their own usage needs.

Note that the *Charging Policy* implemented may influence judgement about the *Economic Convenience* parameter.

Examples:

[22] http://trec.nist.gov/trec_eval/

- An institution may find it advantageous to pay a moderate subscription for offering access to standard functionalities to all of its users and then pay an extra amount of money for access to more advanced functionalities for a restricted set of users who actually need them.
- As another example, consider the possibility of paying a basic fee for subscription to a set of standard *Collections* of a *Digital Library* and pay on a per-*Information Object* basis when you access *Information Objects* belonging to a *Collection* you are not subscribed to.

C156 Interoperability Support

Definition: A *Generic Quality Parameter* reflecting the capability of a *Digital Library* to inter-operate with other *Digital Libraries*.

Relationships:

- *Interoperability Support <isa> Generic Quality Parameter*
- *Interoperability Support <affectedBy> Connectivity Policy*
- *Interoperability Support <affectedBy> Compliance to Standards*

Rationale: This parameter concerns the capability of interoperating with other *Digital Libraries* as well as the ability to integrate with legacy systems and solutions. As discussed in Section II.3, this is a very prominent issue in the Digital Library universe and this parameter can help in expressing the 'degree of interoperability' among *Digital Libraries* and/or *Resources*.

Connectivity Policy may affect *Interoperability Support* since it defines and controls how, and to what extent, a *Digital Library* should be accessible.

Compliance To Standards may affect *Interoperability Support* since their use makes it easier to interact with other systems.

The cost estimation of interoperability may be a component of the *Economic Convenience* measure.

Interoperability Support problems can cause delays or impossibility to fulfil user requests; thus they are also related to user satisfaction.

Examples:

- A relevant example of effort to offer interoperability at the data level is the OAI-PMH protocol [143] and [144] and the OAI-ORE initiative;[23] examples of interoperability efforts at the service level are the SRU/SRW[24] protocol and the Web Services.[25]

C157 Reputation

Definition: A *Generic Quality Parameter* reflecting the trustworthiness of a *Digital Library*.

Relationships:

- *Reputation <isa> Generic Quality Parameter*

Rationale: *Reputation* concerns the 'good name' of a *Digital Library*, the credit it has gained from the user community, and its ability as a point of reference.

Other *Quality Parameters* may greatly affect the *Reputation* and we may consider it as a sort of overall indicator of the appreciation of a *Digital Library*.

---

[23] http://www.openarchives.org/ore/

[24] http://www.loc.gov/standards/sru/

[25] http://www.w3.org/2002/ws/

Examples:

- Examples of aspects that influence the *Reputation* of a *Digital Library* are whether a *Digital Library* provides *Resources* that can be regarded as true, real, impartial, credible and conveying the right information.
- Examples of Quality Parameters that influence Reputation are: Economic Convenience, Usability, Dependability, and so on.

C158 Security Enforcement

Definition: A *Generic Quality Parameter* reflecting the level and kind of security features offered by a *Digital Library*.

Relationships:

- *Security Enforcement <isa> Generic Quality Parameter*
- *Security Enforcement <affectedBy> Digital Rights Management Policy*
- *Security Enforcement <affectedBy> Access Resource*
- *Security Enforcement <affectedBy> Configure DL*
- *Security Enforcement <affectedBy> User Behaviour*

Rationale: This parameter reflects the capability of the *Digital Library* to support the management of different levels of security as expected by users, content depositors, rights owners and librarians themselves.

Security Enforcement can be affected by both *Policies* and *Functions*. In particular, the *Digital Rights Management Policy* affects the level of *Security Enforcement* of a *Digital Library*, since it defines how the content has to be controlled. The *Access Resources* functions and their implementation influence *Security Enforcement*, since they provide *Actors* with mechanisms for consuming *Information Objects*; the *Configure DL* functions impact *Security*, since the possibility of correct and careful configuration of the *Digital Library* is a prerequisite for security; finally, *User Behaviour* can affect the *Security Enforcement*, since an *Actor* may compromise security, for example by careless use of username and password.

Examples:

- An example of a factor that influences Security Enforcement is the capability to prevent unauthorised access to content or the saving of local copies of copyrighted material. Within the Policy domain the regulations should be clearly stated in the Digital Rights Management Policy.

C159 Sustainability

Definition: A *Generic Quality Parameter* reflecting the prospects of durability and future development of a *Digital Library*.

Relationships:

- *Sustainability <isa> Generic Quality Parameter*
- *Sustainability <affectedBy> Change Management Policy*
- *Sustainability <affectedBy> Collection Development*
- *Sustainability <affectedBy> Compliance with Standards*
- *Sustainability <affectedBy> Maintenance*

Rationale: *Sustainability* should take into consideration various factors, such as the organisational and economic aspects of a *Digital Library*, as well as its capability of ensuring the preservation of its *Content* and of keeping pace with future innovations.

Sustainability may be affected by the *Policies* adopted by the *Digital Library*, such as the *Change Management Policy* or the *Collection Development Policy*.

Furthermore, *Compliance with Standards* may affect *Sustainability*, since they support the future development of a *Digital Library*. Also, *Maintenance* may affect *Sustainability*, as it controls how the *Digital Library System* evolves over time.

Examples:

- Examples of factors that influence *Sustainability* are: the funding scheme that ensures the economic conditions for carrying on the *Digital Library*; the skills and willingness within the organisation that provides for the *Digital Library*; the presence of accurate development plans for the collections held by the *Digital Library*, as well as for the software and hardware resources needed for the *Digital Library System* and the *Digital Library Management System*.

C160 Documentation Coverage

Definition: A *Generic Quality Parameter* measuring the accuracy and clarity of the documentation describing a given *Resource*.

Relationships:

- *Documentation Coverage <isa> Generic Quality*

Rationale: This *Quality Parameter* addresses the quality of the written documentation of a *Resource*. The importance of documentation associated to *Resources* of any form is usually underestimated. On the contrary, having a valuable documentation reflects in an optimal usage of the available Resources.

Examples:

- Manuals explaining the use of *Function*s are typical examples of *Documentation Coverage*.
- Other examples are the accuracy of online help, better if contextual, or the selection provided by the Frequently Asked Question sections.

C161 Performance

Definition: A *Generic Quality Parameter* measuring the achievements of a *Resource*.

Relationships:

- *Performance <isa> Generic Quality Parameter*
- *Performance <affectedBy> Capacity*

Rationale: This *Generic Quality Parameter* provides an overall assessment of how well a *Resource* performs from different points of view, e.g. efficiency, effectiveness, efficacy and so on.

Examples:

- The response time upon invocation of a *Function* is an example of a generic *Performance* indicator.
- The presence of delays and/or jitter is an example of *Performance* indicators more tailored to the multimedia and streaming contexts.
- Precision and recall are widely used *Performance* indicators in the information retrieval field.

C162 Scalability

Definition: A *Generic Quality Parameter* measuring the capability of increasing *Capacity* as much as needed.

Relationships:

- *Scalability <isa> Generic Quality Parameter*

Rationale: *Scalability* denotes the ability of a system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and/or to be susceptible to enlargement; it is a desirable attribute of a network, system or process [32]. This is a very wide concept that affects many entities in the Digital Library universe and it is often difficult to define precisely and formally [111].

Examples:

- The ability of a DLS to support a growing number of users and/or provide access to (massively) growing collections without deterioration in performance.
- Another example is the ability to increase the number of requests served by a *Function* while keeping response time reasonable.

C163 Content Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *Content* main concept.

Relationships:

- *Content Quality Parameter <isa> Quality Parameter*
- *Authenticity <isa> Content Quality Parameter*
- *Integrity <isa> Content Quality Parameter*
- *Provenance <isa> Content Quality Parameter*
- *Freshness <isa> Content Quality Parameter*
- *Preservation Performance <isa> Content Quality Parameter*
- *Size <isa> Content Quality Parameter*
- *Scope <isa> Content Quality Parameter*
- *Trustworthiness <isa> Content Quality Parameter*
- *Fidelity <isa> Content Quality Parameter*
- *Perceivability <isa> Content Quality Parameter*
- *Viability <isa> Content Quality Parameter*
- *Metadata Evaluation <isa> Content Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *Content*, in particular *Information Objects*, in a *Digital Library*.

Examples:

- Content quality is in a sense a moving target, but the requirements on the level of quality of various materials in the *Digital Library* and its scope have to be presented in the *Collection Development Policy*.

C164 Authenticity

Definition: A *Content Quality Parameter* reflecting whether an *Information Object* retains the property of being what it purports to be.

Relationships:

- *Authenticity <isa> Content Quality Parameter*

Rationale: The definition takes into account the results and experience of the InterPARES I project[26] [72][73].

Examples:

- The methods for data protection are key to assuring authenticity of *Resources*. Document sealing engines which timestamp and sign digitally every item in the *Digital Library* are an example of a solution that creates the proof that the documents have not been modified from the original.

C165 Trustworthiness

Definition: A *Content Quality Parameter* measuring the trustfulness and credibility of a *Resource* based on the reliability of the creator of the *Resource*.

Relationships:

- *Trustworthiness <isa> Content Quality Parameter*
- *Trustworthiness <affectedBy> Provenance*

Rationale: *Trustworthiness* concerns the reliability and believability of a given *Resource*, meaning the possibility of both placing the *Actor*'s trust in it and resting assured that the trust will not be betrayed. It may be helpful to compare digital libraries that have a similar or identical scope where one might be more trustworthy than the other.

Provenance may affect *Trustworthiness*, since knowing the lineage and history of a *Resource* may improve its reliability and credibility.

Examples:

- NISO Z39.7 Library Statistics and ISO 11620 Library Performance Indicators suggest measures of usage especially for libraries; in this context, *Trustworthiness* could be measured by estimating the number of visitors (general number or different users). Another possibility is to gather transaction information (number of downloads and printouts).

C166 Freshness

Definition: A *Content Quality Parameter* measuring the *Information Object* quality of being current and promptly updated.

Relationships:

- *Freshness <isa> Content Quality Parameter*

Rationale: This parameter evaluates whether an *Information Object* and the information it carries are fresh and updated with respect to the task in hand.

Examples:

- A stream of data coming from a sensor that monitors the temperature and blood pressure of a patient should be updated at regular intervals in order to provide meaningful information for a physician.
- Another relevant example is a *Digital Library* keeping weather forecast information, where it is important to know if this information is updated and reflects the current weather conditions. *Information Objects* might be replicated in order to increase their availability. When a replicated *Information Object* is updated, these changes have to be

[26] http://www.interpares.org/

propagated to all replicas. The *Freshness* value of a replica denotes how up-to-date it is, i.e. how many update operations on this *Information Object* are still outstanding.

C167 Integrity

Definition: A *Content Quality Parameter* measuring the *Information Object* quality of being complete and integral.

Relationships:

- *Integrity <isa> Content Quality Parameter*

Rationale: This parameter encompasses the extent to which an *Information Object* is of sufficient breadth, depth and scope for the task in hand, as pointed out in [20].

Examples:

- From the point of view of data protection, integrity should guarantee that there are no losses in the stored resources. This is an important parameter connected with the preservation of the content.

C168 Preservation Performance

Definition: The *Content Quality Parameter* is used to evaluate the need to undertake actions that would ensure that the digital resources will be accessible over the long term.

Relationships:

- *Preservation Performance <isa> Content Quality Parameter*

Rationale: The *Preservation Performance* parameter helps to monitor the need to apply digital curation actions to the separate resources, collections and *Digital Library* as a whole.

Examples:

- If the policy of the *Digital Library* is to make copies of content stored on DVDs every five years, a *Preservation Performance* parameter would help to comply with this requirement.

C169 Provenance

Definition: A *Content Quality Parameter* recording how well the origins and history of an *Information Object* are known and traced.

Relationships:

- *Provenance <isa> Content Quality Parameter*
- *Provenance <affectedBy> Metadata*
- *Provenance <affectedBy> Annotation*
- *Provenance <affectedBy> Preservation Policy*
- *Provenance <affectedBy> Information Object*

Rationale: This *Quality* parameter is aimed at determining how far it is possible to reconstruct the history and evolution of an *Information Object* in order to know if it fits the purpose. An *Information Object* may be derived from other *Information Objects* (e.g. due to a merger and/or transformations); tracing its provenance is not always a trivial task.

In particular, when we are dealing with scientific data, *Provenance* of data must be traced since a scientist needs to know where the data came from and what cleaning, rescaling or modelling was done to arrive at the data to be interpreted [1].

Note that this parameter resembles what [20] calls ‘interpretability’.

Provenance <affectedBy> Metadata, since the *Metadata* hold the additional information needed to trace the history of an *Information Object*.

Provenance <affectedBy> Annotation, since *Annotations* allow us to trace the provenance and flow of data, report errors or remarks about a piece of data, and describe the quality or the security level of a piece of data [26]. In addition, [100] uses annotations in interactive visualisation systems as a means of both capturing the history of user interaction with the visualisation system and keeping track of the observations that a user may make while exploring the visualisation.

Provenance <affectedBy> Preservation Policy, since it may influence the kind of *Metadata* that are kept about an *Information Object*.

Examples:

- Consider a bioinformatics DL, which supports the analysis of gene expressions. This usually requires a set of tools that need to be applied to raw data in a certain order by a dedicated workflow. Since reproduction of results is a very important requirement in this domain, not only the result of a workflow but also all intermediate steps of this workflow, including the configuration of tools and algorithms, need to be kept as part of the preservation metadata of the *Information Object* that represents the final result.

C170 Scope

Definition: A *Content Quality Parameter* measuring the areas of coverage of the *Content* and/or *Resources* of the *Digital Library*.

Relationships:

- *Scope <isa> Content Quality Parameter*

Rationale: The *Scope* parameter helps to understand the coverage of a *Digital Library* both in the sense of *Content* and in the sense of *Functionality*. While the *Size* provides quantitative insight, *Scope* is more qualitatively oriented.

Examples:

- A *Digital Library* could contain the complete collection of works of a certain author, time period or genre. This is a content-related example.

C171 Size

Definition: A *Content Quality Parameter* measuring the magnitude of *Resource*, *Collection* or a *Digital Library* as a whole.

Relationships:

- *Size <isa> Content Quality Parameter*
- *Size <isa> Quantitative Measure*

Rationale: Sizes can be provided according to different measures: for example, numbers of items, pages, bytes, articles, words, images, multimedia files. The evaluation of the size of a *Digital Library* helps the user to get an idea about the resources. *Size* is also an important parameter for the architecture and functionality of the DL.

Examples:

- The physical size of a collection calculated in bytes is important for estimating the migration effort.

C172 Fidelity

Definition: A *Content Quality Parameter* measuring the accuracy with which an electronic system reproduces a given *Resource*.

Relationships:

- *Fidelity <isa> Content Quality Parameter*

Rationale: The *Fidelity* parameter is used to evaluate to what degree a particular representation of a given *Resource* is different from its original representation.

Examples:

- The rendition of a text document may be identical to its original appearance in the word processing software used at the time of creating the document, but may significantly differ from its original appearance especially in layout – this difference is expressed through *Fidelity*.

C173 Perceivability

Definition: A *Content Quality Parameter* measuring the effort an *Actor* needs to invest in order to understand and absorb a *Resource*.

Relationships:

- *Perceivability <isa> Content Quality Parameter*

Rationale: The *Perceivability* parameter is used to evaluate how easily an *Actor* would understand and retain the information/knowledge within a *Resource* from the *Content* domain. This quality parameter is essential for evaluating which *Resources* are most likely to be well understood within a specific target group of users.

Examples:

- When numerous *Resources* in the *Digital Library* represent the same topic, perceivability may help to choose those that are most likely to be quickly understood. Quite often, images might be found to have higher perceivability than texts. Perceivability can also be used to answer the needs of special groups of users, for example providing audio content to visually impaired users.

C174 Viability

Definition: A *Content Quality Parameter* measuring whether the *Resource*'s bit stream is intact and readable with the existing technology.

Relationships:

- *Viability <isa> Content Quality Parameter*

Rationale: *Viability* is essential for preservation activities within a *Digital Library*. It would estimate whether a digital object could be read and manipulated with the existing hardware and software.

Examples:

- The minimum time specified by the supplier for the media's viability under prevailing environmental conditions.

C175 Metadata Evaluation

Definition: A *Content Quality Parameter* measuring characteristics of *Metadata*.

Relationships:

- *Metadata Evaluation <isa> Content Quality Parameter*

Rationale: *Metadata Evaluation* is essential for various processes in the *Digital Library*, and most specifically in tasks related to access, preservation and operability. According to a functionality-oriented definition of Guy, Powell and Day, 'high quality metadata supports the functional requirements of the system it is designed to support'. Metadata evaluation could be

as simple as checking whether metadata (or specific metadata elements) are available, or it could be a more sophisticated evaluation of incomplete, inaccurate or inconsistent metadata elements. In the most detailed case, *Metadata Evaluation* would be a compound parameter consisting of several others – for example, Completeness, Accuracy, Provenance, Conformance to Expectations, Timeliness, User Satisfaction, Perceivability. This combination would depend on the purpose of the *Metadata Evaluation*.

Examples:

- Completeness in the context of *Metadata evaluation* could be used to measure whether a minimal required set of elements is available in the metadata records.

C176 Functionality Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *Functionality* main concept.

Relationships:

- *Functionality Quality Parameter <isa> Quality Parameter*
- *Usability <isa> Functionality Quality Parameter*
- *User Satisfaction <isa> Functionality Quality Parameter*
- *Availability <isa> Functionality Quality Parameter*
- *Dependability <isa> Functionality Quality Parameter*
- *Robustness <isa> Functionality Quality Parameter*
- *Fault Management Performance <isa> Functionality Quality Parameter*
- *Capacity <isa> Functionality Quality Parameter*
- *Orthogonality <isa> Functionality Quality Parameter*
- *Awareness of Service <isa> Functionality Quality Parameter*
- *Expectations of Service <isa> Functionality Quality Parameter*
- *Impact of Service <isa> Functionality Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *Functionality*, in particular *Functions*, of a *Digital Library*.

Examples: --

C177 Availability

Definition: A *Functionality Quality Parameter* indicating the ratio of the time a *Function* is ready for use to the total lifetime of the system.

Relationships:

- *Availability <isa> Functionality Quality Parameter*
- *Availability <affectedBy> Robustness*
- *Availability <affectedBy> Fault Management*
- *Availability <affectedBy> Capacity*

Rationale: *Availability* is a fundamental parameter for assessing the quality of a *Function*, as *Actors* may be very disappointed when they try to use a *Function* and it is not available.

Availability may be affected by other parameters, such as *Robustness* and *Fault Management*: the former guarantees that a *Function* will continue to work and be available even in the case of bad input; the latter guarantees that a *Function* will be able to recover from an error condition and thus continue to be available. Finally, *Capacity* may also affect *Availability*, as, in the case of starvation of resources, a *Function* may stop being available.

Availability typically parallels *Dependability*.

Examples:

- In the telephone services, high levels of availability are demanded – the well-known 'five-nines', the 99.999% of uptime of the system – since nobody expects to pick up the receiver and not hear the signal.

C178 Awareness of Service

Definition: A *Functionality Quality Parameter* measuring how well the *Actors* of a *Digital Library* are aware of its existence and *Functions*.

Relationships:

- *Awareness of Service <isa> Functionality Quality Parameter*

Rationale: To measure *Awareness of Service*, surveys are most frequently used. To increase *Awareness of Service*, an awareness system could be established as a DL functionality component.

Examples:

- *Awareness of Service* for target user groups is an important component of the current information literacy.

C179 Capacity

Definition: A *Functionality Quality Parameter* representing the limit to the number of requests a *Function* can serve in a given interval of time.

Relationships:

- *Capacity <isa> Functionality Quality Parameter*

Rationale: *Capacity* determines how many concurrent requests can be served successfully.

It may affect *Availability*, *Dependability* and *Performance*. Indeed, when a *Function* operates beyond its *Capacity*, *Availability* may be compromised as the *Function* may stop working, for example in the case of denial of service attacks; similarly, *Dependability* and *Performance* may be negatively affected if the *Function* does not complete its tasks or takes too much time to complete.

Examples:

- The number of *Information Objects* that an information access component can index in a certain unit of time is an example of *Capacity*, as is the maximum number of users that can connect to the portal of a *Digital Library* at the same time.

C180 Expectations of Service

Definition: A *Functionality Quality Parameter* measuring what *Actors* believe a *Function* should offer.

Relationships:

- *Expectations of Service <isa> Functionality Quality Parameter*

Rationale: The *Expectations of Service* from the point of view of the digital library service can be clarified through user agreements on the Quality of Service (QoS), which outline the actual service and the existing framework to the user. However, users might have different expectations based on their experience with other DLs or other digital services. User expectations could be studied through surveys.

Examples:

- Users expect that clicking on an image thumbnail will open up a larger size and higher quality image file.

C181 Fault Management Performance

Definition: A *Functionality Quality Parameter* measuring the ability of a *Function* to react to and recover from failures in a transparent way.

Relationships:

- *Fault Management Performance <isa> Functionality Quality Parameter*
- *Fault Management Performance <affectedBy> Robustness*

Rationale: *Fault Management Performance* reflects the capacity of a *Function* to recover from error conditions, thus avoiding the interruption of the service provided.

It may be affected by *Robustness*, meaning the capacity to recover from faulty inputs.

Examples:

- Consider the case of a *Function* that crashes due to some problem but is able, during its functioning, to save its state and seamlessly restart from the last valid state.
- As a further example, consider the capability of switching to another *Architectural Component* with similar capabilities if the one being used stops working.

C182 Impact of Service

Definition: A *Functionality Quality Parameter* measuring the influence that the service offered by a *Function* has on the *Actor*'s knowledge and behaviour.

Relationships:

- *Impact of service <isa> Functionality Quality Parameter*

Rationale: The user of *Digital Libraries* does not have static skills; in the ideal case, his or her knowledge is increased and the practical skills of exploring digital collections are improved over time. This parameter has special importance if we consider the applications of digital libraries in the educational area, in particular e-Learning applications using *Digital Libraries*.

Examples:

- The user who has experience with a specific visual interface will generally be able to use another similar interface. Since the user has mastered how to use a specific set of functionalities organised in a particular interface, his expectation of service is also different.

C183 Orthogonality

Definition: A *Functionality Quality Parameter* indicating to what extent different *Functions* are independent of each other, i.e. do not affect each other.

Relationships:

- *Orthogonality <isa> Functionality Quality Parameter*

Rationale: *Orthogonality* measures whether sets of *Functions* are independent of each other. DLs with full functional orthogonality, or at least pronounced orthogonality, will usually be much more intuitive for their users than DLs with a high degree of functional overlap.

Orthogonality may affect *Usability* and may also affect *User Satisfaction*, when the usage of the DL might become too complicated.

Examples:

- The idea of *Orthogonality* is that the same function is invoked by the same commands, from the same menu entries.

C184 Dependability

Definition: A *Functionality Quality Parameter* measuring the ability of a DL to perform a *Function* under stated conditions for a specified period of time.

Relationships:

- *Dependability <isa> Functionality Quality Parameter*
- *Dependability <affectedBy> Capability*

Rationale: *Dependability* reflects whether a given *Function* works correctly without producing errors.

Capacity may affect *Dependability*, since in the case of starvation of resources a *Function* may not work properly.

Examples:

- When an *Actor* types the URL of a portal that gives access to a *Digital Library*, he/she expects the address to be correctly resolved and to be redirected to the correct site and not to an incorrect one.

C185 Robustness

Definition: A *Functionality Quality Parameter* measuring the resilience to ill-formed input or incorrect invocation sequences of a *Function*.

Relationships:

- *Robustness <isa> Functionality Quality Parameter*

Rationale: *Robustness* is a key parameter that may affect other *Quality Parameters*, such as *Security Enforcement* or *Availability*. Indeed, many kinds of attack that compromise the functioning of a service or gain unauthorised access to services are based on ill-formed input, such as buffer overflows.

Examples:

- Consider the capacity of preventing buffer overflows, which are often exploited to gain unauthorised access to a system.

C186 Usability

Definition: A *Functionality Quality Parameter* that indicates the ease of use of a given *Function*.

Relationships:

- *Usability <isa> Functionality Quality Parameter*
- *Usability <affectedBy> Orthogonality*

Rationale: *Usability* records to what extent a given *Function* makes it easy for an *Actor* to achieve its goals.

It can be evaluated by using different *Measures*: for example, the *Actor* can indicate on a subjective scale the degree of *Usability* of a *Function*; alternatively, the time needed to complete a task can be measured.

Examples:

- *Usability* concerns many different aspects of a *Digital Library*, ranging from the user interface, the facility in finding and accessing relevant information, the presentation of

search results, to support for facilitating complex or difficult tasks, such as the provision of query-by-example functionalities or browsing and navigation facilities for complex metadata schemas or ontologies.

C187 User Satisfaction

Definition: A *Functionality Quality Parameter* indicating to what extent an *Actor* is satisfied with a given *Function*.

Relationships:

- *User Satisfaction <isa> Functionality Quality Parameter*
- *User Satisfaction <affectedBy> Usability*
- *User Satisfaction <affectedBy> Expectations of Service*
- *User Satisfaction <affectedBy> Documentation Coverage*
- *User Satisfaction <affectedBy> Performance*
- *User Satisfaction <affectedBy> Availability*
- *User Satisfaction <affectedBy> Dependability*
- *User Satisfaction <affectedBy> Orthogonality*

Rationale: The *User Satisfaction* parameter reflects to what extent an *Actor* is satisfied by the capabilities offered by a given *Function*. Many factors can influence *User Satisfaction*, such as *Usability*, *Expectations of Service*, *Documentation Coverage*, *Performance*, *Availability*, *Dependability* and so on.

Examples:

- *User Satisfaction* can be explicitly assessed by making use of surveys and questionnaires where the user's opinion is explicitly requested, or it may be implicitly deduced by observing how much a given *Function* is used and preferred over other similar ones.

C188 User Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *User Domain* main concept.

Relationships:

- *User Quality Parameter <isa> Quality Parameter*
- *User Behaviour <isa> User Quality Parameter*
- *User Activeness <isa> User Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *User Domain*, in particular *Actors*, of a *Digital Library*.

Examples: --

C189 User Activeness

Definition: A *User Quality Parameter* that reflects to what extent an *Actor* is active and interacts with a *Digital Library*.

Relationships:

- *User Activeness <isa> User Quality Parameter*

Rationale: This parameter concerns whether and how much an *Actor* is active with respect to the *Content* and *Functionality* offered by a *Digital Library*.

Examples:

- Factors that influence this parameter are, for example, whether an *Actor* frequently contributes his own *Content* to the *Digital Library* or whether an *Actor* often participates in discussions with other *Actors*, perhaps by using *Annotations*.

C190 User Behaviour

Definition: A *User Quality Parameter* that reflects how an *Actor* behaves and interacts with a *Digital Library*.

Relationships:

- *User Behaviour <isa> User Quality Parameter*

Rationale: This parameter concerns whether and how much an *Actor* abides by the *Policies* and regulations of a *Digital Library*.

Examples:

Factors that influence this parameter are, for example, whether an *Actor* respects the copyright on the *Resources* of a *Digital Library* or if he/she makes unauthorised copies of such material.

C191 Policy Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the top-level *Policy* concept.

Relationships:

- *Policy Quality Parameter <isa> Quality Parameter*
- *Policy Consistency <isa> Policy Quality Parameter*
- *Policy Precision <isa> Policy Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of a set of *Policies*.

Examples: --

C192 Policy Consistency

Definition: A *Policy Quality Parameter* that characterises the extent to which a set of *Policies* are free of contradictions.

Relationships:

- *Policy Consistency <isa> Policy Quality Parameter*

Rationale: This parameter concerns whether or not a set of *Policies* (each of them well defined) are free of contradictions.

Examples:

- *Digital Rights* is a policy regulating rights of use of digital objects. *Digital Rights Management Policy* governs the *Functions* that implement rights issues in the use of *Resources*. These two policies have to be consistent in their approach to rights issues.

C193 Policy Precision

Definition: A *Policy Quality Parameter* that represents the extent to which a set of *Policies* have defined impacts and do not have unintended consequences.

Relationships:

- *Policy Precision <isa> Policy Quality Parameter*

Rationale: *Architecture*, *Functionality* and the underlying technologies need to be well understood when designing DL *Policies*. A lack of knowledge of the technology used may lead to undesired DLS behaviour. Since *Digital Libraries* are such a complex field, we would like to stress the importance of understanding the reasons that cause unexpected behaviour. It

might be the fault of the *Policy*, if aspects it should govern have not been envisaged in the necessary detail (in this case, precision of policy is not sufficient). Other causes of deviant behaviour might be found in insufficient knowledge of technology, or inadequate reflection of architecture or software in the policy design.

Examples:

- A policy limiting the rate of sending data over a network cannot be enforced in a DL if the underlying DLS does not provide some means for adjusting the data transmission rate; this could be of special importance in very large digital libraries or for institutions that have limited resources and need to keep the bandwidth consumption low.

C194 Architecture Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *Architecture Domain* main concept.

Relationships:

- *Architecture Quality Parameter <isa> Quality Parameter*
- *Redundancy* <isa> *Architecture Quality Parameter*
- *Ease of Administration <isa> Architecture Quality Parameter*
- *Load Balancing Performance <isa> Architecture Quality Parameter*
- *Ease of Installation <isa> Architecture Quality Parameter*
- *Log Quality <isa> Architecture Quality Parameter*
- *Maintenance Performance <isa> Architecture Quality Parameter*
- *Compliance to Standards <isa> Architecture Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *Architecture Domain*, in particular *Architectural Components*, of a *Digital Library System*.

Examples: --

C195 Ease of Administration

Definition: An *Architecture Quality Parameter* measuring the presence and ease of use of tools for configuring, administering and monitoring *System Architecture Components*.

Relationships:

- *Ease of Administration <isa> Architecture Quality Parameter*

Rationale: The presence of good administration tools is crucial for configuring and monitoring the functioning of complex and distributed systems, which *Digital Library Systems* potentially are.

Examples:

- A DLS which supports dynamic (re-)configuration by adding or removing *Software Components* without the need to recompile the system after each change.
- The presence of automatic procedures for installing software and patches in a networked and distributed context, or of tools for informing and alerting administrators in the case of malfunctioning are another example of factors that influence the *Ease of Administration*.

C196 Compliance with Standards

Definition: An *Architecture Quality Parameter* measuring the degree to which standards have been adopted in developing an *Architectural Component*.

Relationships:

- *Compliance with Standards <isa> Architecture Quality Parameter*

Rationale: This parameter influences both *Interoperability Support*, since the adoption of standards increases the ease of interoperation with other entities, and the *Sustainability* of a *Digital Library*, since open standards support keeping an *Architectural Component* up-to-date with future technological developments.

Examples:

- Open Source standards are a relevant example of standards that may help in keeping an *Architectural Component* updated and interoperable.

C197 Ease of Installation

Definition: An *Architecture Quality Parameter* measuring the ease of installation and configuration of *Software Components*.

Relationships:

- *Ease of Installation <isa> Architecture Quality Parameter*

Rationale: The *Ease of Installation* parameter concerns the presence of tools and procedures for seamlessly installing and deploying *Software Components*, as well as adding new *System Architecture Components* to an operating *Digital Library System*.

Examples:

- The presence of intuitive wizards for installing new components or the possibility of adding components without restarting the whole system are examples of factors that influence *Ease of Installation*.

C198 Load Balancing Performance

Definition: An *Architecture Quality Parameter* measuring the capacity to spread and distribute work evenly across *System Architecture Components*.

Relationships:

- *Load Balancing Performance <isa> Architecture Quality Parameter*

Rationale: *Load Balancing Performance*, together with *Redundancy*, may help in improving the overall performance and responsiveness of a *Digital Library System*.

Examples:

- For a DLS on top of a Grid environment, which takes into account several instances of *Architectural Components*, *Load Balancing Performance* includes the ability of the system to distribute requests equally among different components of the same type within the system. In particular, this capability consists in selecting *Hosting Nodes* according to their workload or moving a job from one *Hosting Node* to another in order to achieve optimal *Resource* utilisation so that no *Resource* is over/under-utilised.

C199 Log Quality

Definition: An *Architecture Quality Parameter* measuring the presence and accuracy of logs which monitor the activity and functioning of *System Architecture Components*.

Relationships:

- *Log Quality <isa> Architecture Quality Parameter*

Rationale: The presence of accurate logs is crucial for understanding, analysing, debugging and improving the functioning of a *Digital Library System*.

Furthermore, log analysis can be an effective means of understanding *Actor* behaviour and personalising the *Digital Library System* accordingly; therefore, logs can provide useful input for the *Personalise* functions and for creating *Actor Profiles*.

Examples:

- There are various standards for creating logs. For example, in the case of the Web, there is W3C Extended Log Format [107].

C200 Maintenance Performance

Definition: An *Architecture Quality Parameter* addressing the design and implementation of software and hardware maintenance plans for *Architectural Components*.

Relationships:

- *Maintenance Performance <isa> Architecture Quality Parameter*
- *Maintenance Performance <affectedBy> Change Management Policy*

Rationale: *Maintenance Performance* concerns the design of plans for keeping *Architectural Components* updated with research and technological advances.

Change Management Policy may affect *Maintenance Performance*, since it regulates the change process in a *Digital Library*.

It may influence *Sustainability*, as it involves keeping the current system functioning properly and evolving it to face future technological developments.

Examples:

- A maintenance plan may concern programmed hardware updates, controlled migration towards new software and hardware environments, and so on.

C201 Redundancy

Definition: An *Architecture Quality Parameter* measuring the degree of (partial) duplication of *System Architecture Components* to decrease the probability of a system failure.

Relationships:

- *Redundancy <isa> Architecture Quality Parameter*

Rationale: A redundant architecture helps in improving the overall performance of a system and may improve the *Availability*, *Dependability* and *Robustness* of a *Digital Library System*.

Examples:

- Availability of a system can be increased by *Redundancy* of *Architectural Components*. In the event that one component fails, another component of the same type is able to take over.

C202 Architecture Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents the various aspects related to the software systems that concretely realise the Digital Library universe.

Relationships:

- *Digital Library <definedBy> Architecture Domain*
- *Digital Library System <definedBy> Architecture Domain*
- *Digital Library Management System <definedBy> Architecture Domain*
- *Architecture Domain <consistOf> Architectural Component*

Rationale: The *Architecture Domain* encompasses concepts and relationships characterising the two software systems that play an active role in the DL universe, i.e. DLSs and DLMSs. Unfortunately, the importance of this fundamental concept has been largely underestimated in the past. The importance of the domain and its modelling is described in Section II.2.7.

Examples: --

C203 Architectural Component

Definition: A constituent part or an element of a software system implementing one or more *Functions* that can be managed autonomously and that contributes to implement the *Architecture* of a *Digital Library System*.

Relationships:

- *Architectural Component <isa> Resource*
- *Architectural Component <yield> Function*
- *Architectural Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Architectural Component* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Architectural Component <hasProfile> Component Profile*
- *Architectural Component <conformTo> Framework Specification*
- *Architectural Component <use> Architectural Components*
- *Architectural Component <composedBy> Architectural Components*
- *Architectural Component <conflictWith> Architectural Components*
- *Architectural Component <has> Interface*
- *Software Architecture Component <isa> Architectural Component*
- *System Architecture Component <isa> Architectural Component*

Rationale: The notion of *Component* has been introduced in modern software systems to represent 'elements that can be reused or replaced'. By exploiting such an approach, systems gain the potential to be:

- flexible – users' needs change over time, even while the system is being developed. It is important to be able to apply changes to the system at a later stage. Moreover, it should be possible/easy to fix the bugs;
- affordable – both to buy and to maintain. Reuse and replacement features of the component-oriented approach contribute to reducing 'costs'.

An *Architectural Component* is a *Resource* in the Digital Library universe. In particular, this kind of *Resource* becomes relevant in the context of *Digital Library Systems* and *Digital Library Management Systems*, which are responsible for concretely realising the *Digital Library*. As a *Resource* to be managed, such components should have a description, i.e. *Component Profile*, characterising them and promoting their 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. WSDL, as in the case of Web Services. Neither statements nor constraints are imposed on the *Component Profile* associated with each *Architectural Component*.

Examples:

- *Architectural Components* are classified in two main categories: *Software Architecture Components* and *System Architecture Components*. These components are the constituents of a *Software Architecture* and *System Architecture* respectively. Examples of *Software*

Architecture Components and *System Architecture Component* are presented in the respective sections.

C204 Software Architecture Component

Definition: An *Architectural Component* contributing to implementing the *Software Architecture* of a system.

Relationships:

- *Software Architecture Component <isa> Architectural Component*
- *Software Architecture Component <isa> Resource* (inherited from *Architectural Component*)
- *Software Architecture Component <yield> Function* (inherited from *Architectural Component*)
- *Software Architecture Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Software Architecture Component* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Software Architecture Component <hasProfile> Component Profile* (inherited from *Architectural Component*)
- *Software Architecture Component <conformTo> Framework Specification* (inherited from *Architectural Component*)
- *Software Architecture Component <use> Software Architecture Components* (inherited from *Architectural Component*)
- *Software Architecture Component* is *<composedBy> Software Architecture Components* (inherited from *Architectural Component*)
- *Software Architecture Component <conflictWith> Software Architecture Components* (inherited from *Architectural Component*)
- *Software Architecture Component <has> Interface* (inherited from *Architectural Component*)
- *Software Component <isa> Software Architecture Component*
- *Interface <isa> Software Architecture Component*

Rationale: The notion of *Component* has been introduced in modern software systems to represent 'elements that can be reused or replaced'. The advantages of such an approach in implementing software systems are introduced in Section II.2.7 Architecture Domain.

This notion may have different manifestations in present-day systems. In particular, due to the fact that *Software Architecture Components* (being *Architectural Components*) may in turn be composed of smaller and smaller parts (*<composedBy>*), it is possible to model *Software Architecture Components* at different levels of abstraction. For instance, a *Software Architecture Component* implementing a Web Service responsible for providing a range of *Functions* may consist of smaller *Software Architecture Components* (usually logical components in which the whole service is organised), each implementing specific sub-tasks needed to carry out the expected component functions. Each of such smaller *Software Architecture Components* is in turn organised in packages and classes (smaller *Software Architecture Components*), effectively containing the code (program instructions, data structures, etc.) that implements a constituent piece of the main *Software Architecture Component*.

Examples:

- A service in a system following the Service Oriented Architecture.
- A software library, i.e. one or several files that either are necessary for the execution/running of the *Software Architecture Component* or add features to it once co-deployed on the same *Hosting Node*.
- A software package in object-oriented programming. It is a named group of related classes (another example of *Software Architecture Component*). Classes are groups of methods (set of instructions) and variables.

C205 Software Component

Definition: A *Software Architecture Component* representing a program coded to provide a set of *Functions*.

Relationships:

- *Software Component <isa> Software Architecture Component*
- *Software Component <isa> Architectural Component* (inherited from *Software Architecture Component*)
- *Software Component <isa> Resource* (inherited from *Architectural Component*)
- *Software Component <yield> Function* (inherited from *Architectural Component*)
- *Software Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Software Component <regulatedBy> Policy* (inherited from *Resource*)
- *Software Component <regulatedBy> License*
- *Software Component <hasProfile> Component Profile* (inherited from *Architectural Component*)
- *Software Component <conformTo> Framework Specification* (inherited from *Architectural Component*)
- *Software Component <use> Software Components* (inherited from *Architectural Component*)
- *Software Component <composedBy> Software Components* (inherited from *Architectural Component*)
- *Software Component <conflictWith> Software Components* (inherited from *Architectural Component*)
- *Software Component <has> Interface* (inherited from *Architectural Component*)
- *Software Component <representedBy> Information Object*
- *Software Component <realisedBy> Computational Component*

Rationale: The *Software Component* is the core of the component-oriented approach when applied to software systems. This approach promotes software reuse and replacement, and thus makes system development potentially inexpensive.

Examples:

- A Java class implementing a specific Function.

C206 Application Framework

Definition: A *Software Architecture Component* representing middleware, i.e. software that connects and supports the operation of other *Software Architecture Components* available at the *Hosting Nodes*. It provides the runtime environment for the *Running Component*.

Relationships:

- *Application Framework <isa> Software Component*
- *Application Framework <support> Running Component*

Rationale: The middleware guarantees proper operation of *Architectural Components*. The application framework influences the way in which components are implemented. It must be provided before the deployment and configuration of the components. For instance, in the case of components relying on an application framework that offers a SOAP library, the components are implemented expecting that such a library is available on the *Hosting Node*.

Examples:

- Apache Tomcat (http://tomcat.apache.org/)

C207 Interface

Definition: A *Software Architecture Component* representing a set of methods and parameters implemented by an *Architectural Component.* The client of such an *Architectural Component* may rely on them while interacting with it.

Relationships:

- *Interface <isa> Software Architecture Component*
- *Architectural Component <has> Interface*
- *Framework Specification <prescribe> Interface*
- *Component Profile <profile> Interface*

Rationale: The *Interface* encapsulates knowledge about the component, i.e. the rest of the system can use the component according to the patterns enabled by the *Interface*(s).

Examples:

- OAI-PMH [144] prescribed the *Interface* an *Architectural Component* acting as an OAI compliant data provider [143] must implement in order to serve an *Architectural Component* willing to act as an OAI application provider.

C208 Framework Specification

Definition: The *Software Architecture Component* prescribing (*<prescribe>*) the set of *Interfaces* and protocols to which an *Architectural Component* should conform (*<conformTo>*) in order to interact with the other *Architectural Components* of the same system by design.

Relationships:

- *Framework Specification <isa> Software Architecture Component*
- *Architectural Component <conformTo> Framework Specification*
- *Framework Specification <prescribe> Interface*

Rationale: The notion of *Framework Specification* is needed to capture the operational context in which an *Architectural Component* has been designed to operate.

Examples:

- Enterprise JavaBeans
- Component Object Model

C209 System Architecture Component

Definition: An *Architectural Component* contributing to implementing the *System Architecture* of a system.

Relationships:

- *System Architecture Component <isa> Architectural Component*
- *System Architecture Component <isa> Resource* (inherited from *Architectural Component*)
- *System Architecture Component <yield> Function* (inherited from *Architectural Component*)
- *System Architecture Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *System Architecture Component <regulatedBy> Policy* (inherited from *Resource*)
- *System Architecture Component <hasProfile> Component Profile* (inherited from *Architectural Component*)
- *System Architecture Component <conformTo> Framework Specification* (inherited from *Architectural Component*)
- *System Architecture Component <use> Architectural Components* (inherited from *Architectural Component*)
- *System Architecture Component <composedBy> System Architecture Components* (inherited from *Architectural Component*)
- *System Architecture Component <conflictWith> System Architecture Components* (inherited from *Architectural Component*)
- *System Architecture Component <has> Interface* (inherited from *Architectural Component*)
- *Running Component <isa> System Architecture Component*
- *Hosting Node <isa> System Architecture Component*

Rationale: The notion of *Component* has been introduced in modern software systems to represent 'elements that can be reused or replaced'. The advantages of such an approach in implementing software systems are given in Section II.2.7 Architecture Domain as well as discussed in the *Architectural Component* definition.

Examples:

- A server ready to host and run (*Hosting Node*) the software (*Software Component*) implementing a certain *Function,* e.g. the *Search*.

C210 Running Component

Definition: An *Architectural Component* realising a *Software Component*.

Relationships:

- *Running Component <isa> Architectural Component*
- *Running Component <invoke> Running Components*
- *Running Component <hostedBy> Hosting Node*

Rationale: The concrete realisation of the code captured by the notion of *Software Component* in a concrete hardware, i.e. it corresponds to the standard notion of 'software process'.

Examples:

- The operational web server implementing the user interface of the DELOS Digital Library. (http://www.delos.info)

C211 Hosting Node

Definition: A hardware device providing computational and storage capabilities such that (*i*) it is networked, (*ii*) it is capable of hosting components, and (*iii*) its usage is regulated by *Policies*.

Relationships:

- *Hosting Node <isa> System Architecture Component*
- *Running Component <hostedBy> Hosting Node*

Rationale: *Hosting Nodes*, being *System Architecture Components* (and thus *Architectural Components*), should be equipped with *Component Profiles* that represent their description. An example of the usage of such information is the automatic matchmaking process used to assign a *Software Component* to the most appropriate *Hosting Node* for its deployment (i.e. the creation of the *Running Component*) by relying on its descriptive characteristics.

Examples:

- The server equipped with the bundle of software needed to host and run the *Software Component* implementing the user interface of the DELOS Digital Library. (http://www.delos.info)

C212 Software Architecture

Definition: The set of *Software Architecture Components* organised to form a system.

Relationships:

- *Software Architecture <consistOf> Software Architecture Component*

Rationale: Each software system is characterised by a set of software pieces organised in a structure that enables them to work together. This organised set of software is the *Software Architecture*. To help software engineers design their systems, a set of well-proven generic schemes for the solution of recurring design problems have been identified, i.e. *Software Architecture* patterns [43]. Patterns capture existing, well-proven experience in software development and help to promote good design practice. The *Reference Architecture* envisaged in Section I.5 and constituting an important part of the Digital Library development framework is a pattern for *Digital Library Systems*. Similarly to patterns, it is important to recall that many *Reference Architectures* can be designed, each dealing with a specific and recurring problem in designing or implementing DLSs. Moreover, different *Reference Architectures* can be used to construct DLSs with specific properties.

Examples:

- Client-Server Architecture
- Service-oriented Architecture

C213 System Architecture

Definition: The set of *System Architecture Components* organised to form a system.

Relationships:

- *System Architecture <consistOf> System Architecture Component*

Rationale: Each software system is characterised by the set of its constituents. This Reference Model classifies the constituents of a software system along two dimensions, that of the *Software Architecture* and that of the *System Architecture*. The *System Architecture*, as an architecture, is an organised set of constituents. In this case, constituents are *System Architecture Components*, namely *Running Instances* and *Hosting Nodes*. Because of (*i*) the strong relations between *Running Instances* and *Software Components*, i.e. a *Running*

Component is the result of the deployment of a *Software Component*, and (*ii*) the fact that *Software Components* are the main constituents of the *Software Architecture* of the system, there is a strong relation between *Software Architecture* and *System Architecture*. A *System Architecture* is one of the possible instances that are obtainable according to the *Software Architecture* of the system in use. It is well known that, by exploiting a software system developed according to a monolithic application pattern, it is not possible to realise a system with a distributed *System Architecture*. The more flexible the *Software Architecture* a system adopts, the larger will be the potential range of application scenarios that can be successfully exploited.

Examples:

- The set of servers and services realising the DELOS Digital Library. (http://www.delos.info)

C214 Purpose

Definition: The motivation characterising the <*associatedWith*> relationship.

Relationships:

- *Resource* <*associatedWith*> *Resource* to a certain *Purpose*

Rationale: The <*associatedWith*> relation is one of the powerful ones enabling the building of compound *Resources*, i.e. *Resources* obtained by combining existing constituent *Resources* so as to form a new knowledge bundle that has a value added with respect to the single *Resources* when considered as a single island of information. Various kinds of associations are possible, and this diversity is captured by the *Purpose* concept attached to each instance of the <*associatedWith*> relation.

Examples:

- An *Information Object* representing an experiment (itself composed of various *Information Objects* representing, for example, the dataset the experiment is carried on, the dataset representing the outcomes, the description of the procedure adopted) is <*associatedWith*> the *Information Object* representing the scientific publication in an outstanding Journal in the field with the <*Purpose*> of scholarly dissemination.

C215 Region

Definition: A contiguous portion of a given *Resource* with the desired degree of granularity identified in order to anchor a given *Annotation* to it.

Relationships:

- *Resource* <*hasAnnotation*> *Annotation* about a *Region*

Rationale: The idea of 'contiguous portion' of a *Resource* resembles and complies with the concept of segment introduced in Navarro *et al*. [165]. The granularity of such a kind of 'identifier' can vary according to the meaningful ways of locating a part of a *Resource*, which depend on the actual specialisation of the *Resource* we are dealing with. As a consequence, we can have *Regions* that anchor an *Annotation* to the whole *Resource*, as well as *Regions* that can anchor an *Annotation* to a specific part of a *Resource*.

Examples:

- A piece of text, e.g. a paragraph, of an *Information Object* representing this volume is a *Region* to which an *Annotation* can be attached.

C216 Digital Library

Definition: An organisation, which might be virtual, that comprehensively collects, manages and preserves for the long term rich *Information Objects*, and offers to its *Actors* specialised *Functions* on those *Information Objects*, of measurable quality, expressed by *Quality Parameters*, and according to codified *Policies*.

Relationships:

- *Digital Library <manage> Resource*
- *Digital Library <manage> Information Object*
- *Digital Library <serve> Actor*
- *Digital Library <offer> Function*
- *Digital Library <agreeWith> Policy*
- *Digital Library <tender> Quality Parameter*
- *Digital Library System <support> Digital Library*
- *Digital Library* is *<definedBy> Resource Domain*
- *Digital Library* is *<definedBy> Content Domain*
- *Digital Library* is *<definedBy> User Domain*
- *Digital Library* is *<definedBy> Functionality Domain*
- *Digital Library* is *<definedBy> Policy Domain*
- *Digital Library* is *<definedBy> Quality Domain*

Rationale: Digital Library is a complex universe and usually the term 'digital library' is used with many different semantics. This Reference Model introduces three notions of systems (cf. Section I.2) active in this universe, i.e. *Digital Library*, *Digital Library System* and *Digital Library Management System*. The first is the most abstract of the three and represents the set of *Information Objects*, *Actors*, *Functions*, *Policy* and *Quality Parameters* forming the *Digital Library* and perceived by *End-users* as the service they can exploit. This service is supported by a running system, i.e. the *Digital Library System*.

Examples:

- The DELOS Digital Library (http://www.delos.info)
- The European Library (http://www.theeuropeanlibrary.org)
- The National Science Digital Library (http://nsdl.org)

C217 Digital Library System

Definition: A software system based on a given (possibly distributed) *Architecture* and providing all the *Functions* required by a particular *Digital Library*. *Actors* interact with a *Digital Library* through the corresponding *Digital Library System*.

Relationships:

- *Digital Library System <support> Digital Library*
- *Digital Library System* is *<definedBy> Resource Domain*
- *Digital Library System* is *<definedBy> Content Domain*
- *Digital Library System* is *<definedBy> User Domain*
- *Digital Library System* is *<definedBy> Functionality Domain*
- *Digital Library System* is *<definedBy> Policy Domain*

- *Digital Library System* is *<definedBy> Quality Domain*
- *Digital Library System* is *<definedBy> Architecture Domain*
- *Digital Library System <has> Software Architecture*
- *Digital Library System <has> System Architecture*

Rationale: This Reference Model introduces three notions of systems (cf. Section I.2) active in the universe, i.e. the *Digital Library*, the *Digital Library System* and the *Digital Library Management System*. The *Digital Library System* is the running software system serving the *Digital Library*. Like any running software system, it is characterised by two facets, its *Software Architecture* and its *System Architecture*. The former consists of a set of *Software Architecture Components*, i.e. *Software Components* and *Interfaces* that compose the software implementing the system. The *System Architecture* is the set of *System Architecture Components* that form the running system, namely the servers, *Hosting Nodes* and the processes, *Running Components*, resulting from the deployment of the *Software Components*.

Examples:

- The set of servers, services and software realising the DELOS Digital Library (http://www.delos.info)
- The set of servers, services and software realising The European Library (http://www.theeuropeanlibrary.org)
- The set of servers, services and software realising the National Science Digital Library (http://nsdl.org)

C218 Digital Library Management System

Definition: A generic software system that provides the appropriate software infrastructure both *(i)* to produce and administer a *Digital Library System* incorporating the suite of *Functions* considered fundamental for *Digital Libraries*, and *(ii)* to integrate additional *Software Components* offering more refined, specialised or advanced functionality.

Relationships:

- *Digital Library Management System <deploy> Digital Library System*
- *Digital Library Management System <extend> Digital Library System*
- *Digital Library Management System* is *<definedBy> Resource Domain*
- *Digital Library Management System* is *<definedBy> Content Domain*
- *Digital Library Management System* is *<definedBy> User Domain*
- *Digital Library Management System* is *<definedBy> Functionality Domain*
- *Digital Library Management System* is *<definedBy> Policy Domain*
- *Digital Library Management System* is *<definedBy> Quality Domain*
- *Digital Library Management System* is *<definedBy> Architecture Domain*

Rationale: This Reference Model introduces three notions of systems (cf. Section I.2) active in the universe, i.e. the *Digital Library*, the *Digital Library System* and the *Digital Library Management System*. The *Digital Library Management System* (*DLMS*) is the system that provides *DL Designers*, *DL System Administrators* and *DL Application Developers* with *Functions* supporting their tasks (cf. Section I.4). Depending on the set of *Functions* with which the *DLMS* provides *Actors*, different types of such systems can be implemented (cf. Section I.2).

Examples:

- OpenDLib (http://www.opendlib.com): the *DLMS* used to create and maintain the DELOS Digital Library (http://www.delos.info).
- DILIGENT [68]: a prototypical *DLMS* capable of deploying *Digital Library Systems* by relying on a set of *Resources* ranging from *Software Components* to *Hosting Nodes* dynamically gathered through Grid technologies.
- The DelosDLMS [184]: a *DLMS* built by integrating software and services developed by DELOS partners.

III.4 Relations' Hierarchy

[Generic Relations][27]
. isa
. [Resource Relations]
. . R1 identifiedBy
. . R2 hasFormat
. . . R3 expressionOf
. . . R4 conformTo
. . R5 hasQuality
. . R6 regulatedBy
. . R7 hasMetadata
. . . R8 describedBy
. . . R9 modelledBy (isa User Relation)
. . . R10 hasProfile (isa Architecture Relation)
. . R11 hasAnnotation
. . R12 expressedBy
. . R13 hasPart
. . . R14 composedBy (isa Architecture Relation)
. . R15 associatedWith
. . . R16 use (isa Architecture Relation)
. . . R17 conflictWith (isa Architecture Relation)
. . . R18 invoke
. . R19 belongTo
. . . R25 profile
. [Content Relations]
. . R20 hasEdition
. . R21 hasView
. . R22 hasManifestation
. . R23 hasIntension
. . R24 hasExtension
. [User Relations]
. . R26 perform
. . R9 modelledBy (isa hasMetadata)
. . R27 play
. . R19 belongTo (isa Resource Relation)
. [Functionality Relations]
. . R28 interactWith
. . R29 influencedBy
. . R30 actOn
. . R31 create
. . . R32 createAnnotation
. . . R33 createVersion
. . . R34 createView
. . . R35 createManifestation
. . R36 retrieve

[27] 'Classifiers', i.e. items added to the hierarchy for organisational purposes are indicated [in square brackets].

. . . R37 return
. . R38 produce
. . R39 issue
. [Policy Relations]
. . R6 regulatedBy (isa Resource Relation)
. . R40 govern
. . . R41 prescribe
. . R42 antonymOf
. . R43 influence
. [Quality Relations]
. . R44 expressAssessment
. . R45 evaluatedBy
. . R46 measuredBy
. . R47 affectedBy
. . R48 accordTo
. [Architecture Relations]
. . R49 implement
. . . R50 realisedBy
. . . R51 support
. . . R52 hostedBy
. . R14 composedBy (isa hasPart)
. . R16 use (isa associatedWith)
. . R17 conflictWith (isa associatedWith)
. . R10 hasProfile (isa hasMetadata)
. . R4 conformTo (isa hasFormat)
. . . R41 prescribe (isa govern)
. . R25 profile (isa belongTo)

III.5 Reference Model Relations' Definitions

R1 identifiedBy

Definition: The relation connecting a *Resource* to its *Resource Identifier*.

Rationale: The issue of univocally identifying the constituents of a system is a fundamental task for their management. This relation captures the *Resource Identifier* attached to each *Resource* for this identification purpose. Various types of *Resource Identifiers* have been proposed; for a discussion of these, please refer to the *Resource Identifier* concept.

Each *Resource* must have at least one *Resource Identifier*. Each *Resource Identifier* can be assigned to one *Resource* only.

Examples:

- The paper 'Setting the Foundations of Digital Libraries: The DELOS Manifesto' is identified by the DOI 10.1045/march2007-castelli.

R2 hasFormat

Definition: The relation connecting a *Resource* to its *Resource Format*, which establishes the attributes or properties of the *Resource*, their types, cardinalities and so on.

Rationale: A *Resource* must have one format only, whereas the same format can (obviously) be used by many *Resources*.

This relation is commonly called 'instance of' in object models. However, to avoid confusion with the instance of relation of the present model, it is given a different name.

Examples:

- The paper 'Setting the Foundations of Digital Libraries: The DELOS Manifesto' has a *Resource Format* 'text/html'.
- The electronic version of this volume has a *Resource Format* that is a pdf file;
- The OAI-ORE[28] Resource Map is the Resource Format of an OAI-ORE Aggregation.

R3 expressionOf

Definition: The relation connecting a *Resource Format* to the *Ontology* that defines the terms of the schema and states the main constraints on them.

Rationale: A schema gives a concrete status to the terms abstractly defined in an ontology, by establishing implementation details such as the data type of the primitive concepts of the ontology. The schema must retain the constraints expressed in the ontology and may consistently add other constraints, reflecting the implementation decisions taken in the schema.

The same ontology can be expressed in many schemas, while a schema is preferably the expression of a single ontology, even though for practical reasons it is possible for a schema to borrow from a number of ontologies. In this latter case, the schema performs a sort of integration of several ontologies.

Examples:

- The Multipurpose Internet Mail Extensions (MIME) is an Ontology of possible types for Information Objects.

[28] http://www.openarchives.org/ore/

R4 conformTo

Definition: The relation connecting *Architectural Components* to the *Framework Specifications* they comply with.

Rationale: The *Framework Specification* is the *Software Architecture Component* that describes the design of the set of *Architectural Components* planned to form the *Software Architecture* of a system. A *Framework Specification* <*prescribe*> the *Interfaces* that *Architectural Components* should implement. Compliance with the *Framework Specification* guarantees that the *Architectural Components* will by design be interoperable with the other *Architectural Components* of the same system.

Examples:

- A *Framework Specification* may <*prescribe*> the publish/subscribe *Interface* that each *Software Component* of a system must implement in order to conform to the publish/subscribe mechanism planned for such a system.

R5 hasQuality

Definition: The relation connecting a *Resource* to its *Quality Parameters*.

Rationale: A *Resource* will have as many *Quality Parameters* as the number of quality features with which it is associated. The same *Quality Parameter* can be associated with many *Resources*.

Examples:

- A *Function* may be assessed with regard to *User Satisfaction* to understand to what extent the needs of the *Actors* using it are fulfilled.

R6 regulatedBy

Definition: The relation connecting *Resources* to the *Policies* regulating them.

Rationale: This relation is used to show what *Resources* are being regulated by a specific *Policy*.

Each *Resource* may be regulated by more *Policies*. The same *Policy* may regulate more *Resources*.

Examples:

- Saving a local copy of an *Information Object* by an *Actor* is regulated by *Digital Rights*.

R7 hasMetadata

Definition: The relation connecting *Resources* to *Information Objects* for management purposes.

Rationale: In classic *Digital Library* models, *Metadata* is a concept that is a primary notion modelling a clearly defined category of objects in the domain of discourse. However, it depends from the context whether an object is or is not *Metadata*. For instance, a relational tuple describing an event (such as an artistic performance) can be a primary *Information Object* in some contexts (e.g. in a database storing the programme of the theatre season) and *Metadata* in a different context (e.g. in a repository storing a digital representation of the performance). For this reason, the notion of *Metadata* is more clearly seen as a role that an *Information Object* plays to another *Information Object* (more precisely, to a resource), hence it is defined as a relation.

From this relation, the notion of *Metadata* is then derived as meaning any *Information Object* that is *Metadata* of a *Resource*. In so doing, we are following the same linguistic convention that, in everyday speech, leads to the usage of the word 'father' as a noun.

Each *Resource* can be associated with zero or more *Information Objects* implementing *Resource's Metadata*. An *Information Object* can be the *Metadata* on zero or one *Resource*.

Examples:

- This volume is an *Information Object* associated with another *Information Object* representing its Dublin Core metadata record via the <*hasMetadata*>.

R8 describedBy

Definition: The relation connecting *Resources* to *Information Objects* describing them.

Rationale: This is a specialisation of the <*hasMetadata*>. A *Resource* can be associated with many descriptive *Information Objects*. A descriptive *Information Object* is associated with one *Resource*.

Examples:

- This volume is associated with an *Information Object* implementing a summary of the volume content for advertisement actions.

R9 modelledBy

Definition: The relation connecting *Actors* to *Actor Profiles* representing them.

Rationale: This is a specialisation of the <*hasMetadata*>. An *Actor* may have many *Actor Profiles*. An *Actor Profile* must be associated with one *Actor*.

Examples:

- John is associated with an *Information Object* containing John's full name, date of birth, address and reading preferences.

R10 hasProfile

Definition: The relation connecting *Architectural Components* to *Component Profiles* representing them.

Rationale: This is a specialisation of the <*hasMetadata*>. An *Architectural Component* can be associated with *Component Profiles*. A *Component Profile* must be associated with one *Architectural Component*.

Examples:

- A Web service is associated with an *Information Object* implementing its WSDL document, i.e. a description of the web service in terms of the operations performed and the messages, the data types and the communication protocols used.

R11 hasAnnotation

Definition: The relation connecting *Resources* to *Information Objects* to add an interpretative value to a certain *Region*.

Rationale: This relation is analogous to <*hasMetadata*>. *Annotations* are sometimes modelled as concepts; however, they are more clearly seen as roles that *Information Objects* play to *Resources* in specific contexts. Hence, *Annotation* (cf. Section III.3 C14) is defined as the range of the <*hasAnnotation*> relation. Fortunately, this definition settles the long-standing issue as to whether *Annotations* are to be considered as *Objects* or as *Metadata*.

Each *Resource* can be associated with zero or more *Information Objects* implementing *Resource's Annotations*. An *Information Object* can be the *Annotation* on zero or one *Resource*.

Examples:

- Each note John, a reader of this volume, will produce can be linked to the version he holds and the published as a the 'The DELOS Digital Library Reference Model annotated by John'.

R12 expressedBy

Definition: The relation connecting *Resources* to *Information Objects* materialising them.

Rationale: This relation has been introduced to capture the materialisation of otherwise abstract *Resources*. It is intended mainly for the materialisation of *Resources* such as *Policy* and *Quality Parameter* but can be applied to any type of *Resource*.

A *Resource* can be associated with many *Information Objects* materialising it in different ways. A materialising *Information Object* must be associated with one *Resource*.

Examples:

- The *Information Object* recording the 'Mean Average Precision' *Measure* of a *Performance Quality Parameter*.

R13 hasPart

Definition: The relation connecting *Resources* to their constituent *Resources*.

Rationale: The relation where a *Resource* 'child' is a subset or part of the 'parent' *Resource*. This 'part of' association may have two different natures: the aggregative and the compositional one. In the aggregative nature, the single parts stand by themselves and may be constituents of any number of *Resources*. In the compositional nature, the whole strongly owns its parts, i.e. if the whole *Resource* is copied or deleted, its parts are copied or deleted along with it.

Examples:

- A book has parts: the preface, the chapters, the bibliography

R14 composedBy

Definition: The relation connecting *Architectural Components* to constituent *Architectural Components*.

Rationale: This is the specialisation of the <*hasPart*> relation in the case of *Architectural Components*. Also, in this case the relation can implement the aggregative and the compositional nature of the 'part of'.

An *Architectural Component* can be comprised of many *Architectural Components*. The same *Architectural Component* can be a component of many *Architectural Components*.

Examples:

- A Fedora[29] Repository is comprised of a federation of services among wich the Fedora Search Service, the Preservation Integrity Service and the Fedora Repository Service.

[29] http://www.fedora.info

R15 associatedWith

Definition: The relation connecting a *Resource* to the *Resources* that are linked to the former according to a certain *Purpose*.

Rationale: In addition to the explicitly identified pool of relations connecting *Resources*, this relation makes it possible to specify cross-resource links with respect to a well-known *Purpose*.

No constraints regarding the cardinality of this relation are established, i.e. a *Resource* may be connected to zero or more *Resources* through the <*associatedWith*> with a certain *Purpose*; a *Resource* may be, or may appear as, the second term of an <*associatedWith*> relationship with a certain *Purpose*.

Examples: --

R16 use

Definition: The relation connecting *Architectural Components* to *Architectural Components* they use.

Rationale: *Architectural Components* are the constituents of the architectures of *Digital Library System*. Thus, despite this model permit to represent monolithic systems, i.e. system composed by a single Architectural Component, it is recommended that systems exploit the component-oriented approach because of its benefits. The <*use*> relations capture the usage relationships between the constituents of compound systems.

An *Architectural Component* can rely on the functions of zero or more *Architectural Components* and can be used by zero or more other *Architectural Components*.

Examples: --

R17 conflictWith

Definition: The relation connecting *Architectural Components* to *Architectural Components* they conflict with.

Rationale: In software systems exploiting the component oriented approach each architectural component must fit with the characteristics of the environment in which it have to operate. The <*conflictWith*> relation captures incompatibilities between *Architectural Components* preventing the coexistence of such *Architectural Components* in the same system. In particular, this relation is particularly useful in the case of Digital Library Systems dynamically deployed, i.e. the set of constituent *Architectural Components* is automatically aggregated by the *DLMS* in order to implement a *Digital Library* (and thus deploying the *DLS* realising the *DL*) matching the *DL Designer* criteria.

An *Architectural Component* can conflict with zero or more *Architectural Components*.

Examples:

- The *Software Component* A conflicts with the *Software Component* B; thus A and B cannot coexist in the same system.
- The *Software Component* A conflicts with the *Hosting Node* B; thus A cannot be deployed on B.

R18 invoke

Definition: The relation connecting *Running Components* to *Running Components* they use to accomplish their task.

Rationale: In software systems exploiting the component oriented approach a lot of relations hold between the constituents. Some of these relations are static, i.e. established at design time and valid in each environment the connected components appear, other are dynamic, i.e. they can evolve along the time and exist in the specific environment in which they have been defined. Usually, dynamic relations are a consequence (more precisely an instantiation) of static relations in an operational context. The <*invoke*> relation represents a dynamic dependency between *Running Components*. In particular, this relation is used to capture the run-time dependency between a *Running Component* and the set of other *Running Components* it uses to deliver its expected functions.

Examples:

- The *Running Component* A invokes the *Running Component* B to obtain the set of *Information Object* it must process to serve a specific request.

R19 belongTo

Definition: The relation connecting *Resources* to the *Resource Sets* in which they belong. A specialisation of this is the relation connecting *Information Objects* to the *Collections* that defines which *Collections* an *Information Object* belongs to. Another specialisation of this is the relation connecting an *Actor* to a *Group* that defines which user group an actor belongs to.

Rationale: A *Resource* may be a member of any number of *Resource Sets* and, conversely, a *Resource Set* may include any (finite) number of *Resources*.

Examples:

- An *Information Object* belongs to one or more *Collections*;
- An *Actor* belongs to one or more *Groups*.

R20 hasEdition

Definition: The relation connecting *Information Objects* to the *Information Objects* that realise them along the time dimension.

Rationale: In classic *Digital Library* models, *Editions* represent the different states of an *Information Object* during its lifetime, i.e. they play the role usually assigned to versions.

Versioning usually creates a tree, because an object may be the version of at most one other object. However, in the Digital Library world a more liberal approach may be appropriate, allowing an *Information Object* to be the *Edition* of possibly many different *Information Objects*. The resulting structure will be a directed graph, which must be acyclic to avoid unintuitive situations.

Examples: An *Information Object* representing a study:

- <*hasEdition*> another *Information Object* representing the draft version of such a study;
- <*hasEdition*> another *Information Object* representing the 'submitted version' of such a study;
- <*hasEdition*> another *Information Object* representing the 'version published in the conference proceedings' with colour images.

R21 hasView

Definition: The relation connecting *Information Objects* to the *Information Objects* that are *Views* of them.

Rationale: The concept of *View* captured by this relation fits very well with those used in the database world. In this context, a view is a virtual or logical table expressed as a query

providing a new organisational unit to support some application. Similarly, *Information Object Views* are introduced to provide multiple presentations of the information represented/captured by the *Information Object*, which may prove useful in specific application contexts.

The same *Information Object* may have different *Information Objects* linked through the <*hasView*> relation. Conversely, an *Information Object* may or may not be a *View*, that is the second term of a <*hasView*> relationship.

Examples:

- An *Information Object* representing a data stream of an environmental sensor
  - <*hasView*> the *Information Object* consisting of the raw data, i.e. a series of numerical values measured by the sensor
  - <*hasView*> the *Information Object* consisting of a picture representing the graph of the evolution of the values measured by the sensor over time.
- An *Information Object* representing the outcomes of a workshop
  - <*hasView*> the *Information Object* representing the 'full view' and containing a preface prepared by the conference chair and the whole set of papers accepted and organised thematically
  - <*hasView*> the *Information Object* representing the 'handbook view' and containing the conference programme and the slides of each lecturer accompanied by the abstract of the papers organised per session, and
  - <*hasView*> the *Information Object* representing the 'informative view' and reporting the goal of the workshop and the title list of the accepted papers together with the associated abstract.

R22 hasManifestation

Definition: The relation associating *Information Objects* to the *Information Objects* representing their physical embodiment.

Rationale: While *Edition* and *View* concepts deal with the intellectual and logical organisation of *Information Objects*, the *Manifestation* concept captured by the <*hasManifestation*> relation deals with the physical presentation of objects.

Examples:

- The *Information Object* representing a conference paper <*hasManifestation*> the pdf file or the Microsoft Word file embodying it.
- A lecture *Information Object* <*hasManifestation*> the MPEG file containing the video recording of the event.
- A sensor *Information Object* <*hasManifestation*> the file containing the raw data captured by the sensor.

R23 hasIntension

Definition: The relation connecting *Collection* to the *Query* describing the criterion underlying the *Collection*.

Rationale: In logic, the intension of an expression is its sense, as distinguished by the reference (or denotation) of the expression, called the extension of the expression. This distinction was first made by G. Frege, for whom the sense of an expression corresponded to what we intuitively think of as the meaning of the expression. R. Carnap later suggested that

the sense of an expression is a function that gives, for each state of affairs, the extension of the expression. S. Kripke, upon defining a semantics for modal logic, finally established the notion of 'possible world as state of affairs' [69]. Davidson argued that giving the meaning of a sentence is equivalent to stating its 'truth conditions'.

The intension of a *Collection* can thus be understood as a statement of what must be true of an object for it to be a member of the *Collection*.

Examples:

- The *Collection* of 'Leonardo Da Vinci' works <*hasIntension*> the *Query* 'author=Leonardo Da Vinci'.

R24 hasExtension

Definition: The relation connecting *Collections* to the *Resource Sets* representing the *Information Objects* belonging to them.

Rationale: In logic, the extension of an expression is its denotation. For a proposition, this is the truth value in the considered interpretation; the extension of a predicate is the set of objects that are denoted by the predicate in the considered interpretation.

Examples:

- The *Collection* of 'Leonardo Da Vinci' works <*hasExtension*> the set of *Information Objects* authored by Leonardo Da Vinci.

R25 profile

Definition: The relation connecting *Component Profiles* to the *Architectural Components* they describe. *Component Profiles* should describe (at least) *Functions*, *Policies*, *Quality Parameters* and *Interfaces* inherent in *Architectural Components* with which they are associated via the <*hasProfile*> relation.

Rationale: *Component Profile* is the *Metadata* associated with each *Architectural Component* for its management. The <*profile*> relation captures the aspects that are expected to be captured by this kind of *Metadata*.

Examples:

- The *Functions* yielded by the *Architectural Component* are typical information expected to be included in the *Component Profile.*
- The *Quality Parameters* guaranteed by the *Architectural Component* are typical information expected to be included in the *Component Profile.*

R26 perform

Definition: The relation connecting *Actors* to *Functions* they use to accomplish their Digital Library activities.

Rationale: *Functions* have no meaning by themselves if no *Actor* is executing them. This relation is fundamental to a DL, as it expresses the interaction of the DL with the *Actors* through specific *Functions* in order to achieve their goals.

Examples: --

R27 play

Definition: The relation connecting an *Actor* to a *Role* that defines the *Role*(s) of the *Actor*.

Rationale: DL *Actors* can play different *Roles* in the DL; for example, they can at the same time be *Content Creator* and *Content Consumer*.

Examples: --

R28 interactWith

Definition: The relation connecting *Functions* to *Functions* that expresses the interaction between them.

Rationale: This facility is fundamental to the modelling of the workflow of execution for the *Functions*. It defines an order between them so as to clarify which *Function* follows the current one.

Examples: --

R29 influencedBy

Definition: The relation connecting *Functions* to *Actor Profiles* that expresses the fact that *Functions* are influenced by specific user characteristics.

Rationale: This relation is very important for personalisation, as it expresses the fact that functionality is related to and influenced by the *Actor Profile* of the *Actor* executing it, thus adapting to the *Actor*'s specific needs.

Examples:

- The *Search Function* is influenced by the *Actor Profile* of the *Actor* that perform it by personalising the returned *Result Set*.

R30 actOn

Definition: The relation connecting *Functions* to *Resources* on which they operate.

Rationale: This relation expresses the connection between specific *Functions* and the *Resources* they interact with, either to manage or access them. A *Function* in most cases produces a result to be presented to the *Actor*; it represents an action performed on one of the DL constituents, which are not only primary *Information Objects* but also *Actor* profiles, *Policies*, etc.

Examples:

- The *Publish Function* acts on the *Collections* the *Actor* performing it requests to submit the *Information Object*.

R31 create

Definition: The relation connecting the *Create Functions* to *Resources* they create. A specialisation of this is the relation connecting the *Author Functions* to the *Information Objects* created.

Rationale: This connection expresses the relation of the creation of *Resources* to the *Resource* created by the *Function*. Note that in this case the new *Resource* is not actually inserted in the library until a *Submit Function* has been performed.

Examples: --

R32 createAnnotation

Definition: The relation connecting *Annotate Functions* to the *Information Objects* they create.

Rationale: This relation expresses the relation of the creation process of an *Annotation* with its end-result.

Examples: --

R33 createVersion

Definition: The relation connecting *Author Collaboratively Functions* to *Information Objects* they create. These *Information Objects* are linked to the originating *Information Objects* via the <*hasEdition*> relation.

Rationale: This relation expresses the fact that a by-product of collaborative authoring is the creation of different versions of the authored *Information Object*.

Examples: --

R34 createView

Definition: The relation connecting *Convert to Different Format Functions* to *Information Objects* they create. These *Information Objects* are linked to the originating *Information Objects* via the <*hasView*> relation.

Rationale: This relation records the fact that the conversion of an *Information Object* to a different format creates another view of it. An example of this is the conversion of a Word document to pdf.

Examples: --

R35 createManifestation

Definition: The relation connecting *Extract* and *Physically Convert Functions* to *Information Objects* they create. These *Information Objects* are linked to the originating *Information Objects* via the <*hasManifestation*> relation.

Rationale: In this case, the primary *Information Object* itself is transformed and a new *Manifestation* is created.

Examples: --

R36 retrieve

Definition: The relation connecting *Access Resource Functions* to *Resources* they find. A specialisation of this relation connects *Find Collaborator Functions* to *Actors* they find. Another specialisation is the relation <*return*> which connects the *Function Discover* and *Result Set.*

Rationale: This *Function* connects a retrieval function to the retrieved result.

Examples: --

R37 return

Definition: The relation connecting *Discover Functions* to *Result Sets* they find. It is a specialisation of the <*retrieve*> relation connecting *Access Resource Functions* to *Resources*.

Rationale: This *Function* connects a *Discover Function* to the *Result Set* it returns.

Examples: --

R38 produce

Definition: The relation connecting *Queries* to *Result Sets* they characterise.

Rationale: When a *Query* is issued as the input to a *Search Function*, it produces a *Result Set*.

Examples: --

R39 issue

Definition: The relation connecting *Search Functions* to the *Queries* they use to retrieve results.

Rationale: In order for the *Function* to retrieve the results the *Actor* has requested, it has to issue a *Query* to a *Collection* and retrieve a *Result Set*.

Examples: --

R40 govern

Definition: The relation connecting *Policies* to the *Resources* they control/govern. It is the inverse relation of <*regulatedBy*>.

Rationale: Each *Policy* to be effectively implemented must be applied to *Resources*. This relation captures those *Resources* for which each *Policy* is designed to influence the actions and conduct.

Examples:

- Digital Rights Management Policy governs Functions, while Digital Rights govern Information Objects.

R41 prescribe

Definition: The relation connecting *Framework Specifications* to the *Interfaces* they state as a rule that should be carried out by *Architectural Components* that <*conformTo*> it.

Rationale: *Framework Specification* is the *Software Architecture Component* that describes the design of the set of *Software Architecture Components* designed to form the *Software Architecture* of a system. By establishing the *Interfaces* each *Software Architecture Component* (actually a *Software Component*) is expected to implement, it is possible to guarantee by design that the set of *Software Architecture Components* forming a *Software Architecture* will work successfully collaboratively so as to form a whole.

Examples:

- A *Framework Specification* may <*prescribe*> the publish/subscribe *Interface* each *Software Component* of a system must implement in order to conform to the publish/subscribe mechanism planned for such a system.

R42 antonymOf

Definition: The relation connecting *Policy* to *Policy* with opposite meaning.

Rationale: This relation is used when we have a set of two *Policies* (generally *Resources*) with opposite meaning. It is introduced in order to facilitate the understanding of bipolar sets of concepts.

Examples:

- *Extrinsic Policy* and *Intrinsic Policy* form a pair where each concept is the <*antonymOf*> the other concept.

R43 influence

Definition: The relation connecting *Quality Parameters* to *Policy* they affect.

Rationale: This Reference Model does not present the digital library as a static entity but also highlights the processes within the functioning of a digital library. An important aspect is how decisions for applying specific *Policies* could be taken within the DL. This relation captures those cases in which the decision is based on *Quality Parameters*.

Examples:

- The value of the *Security Enforcement Quality Parameter* supported by a *Digital Library System* will influence the *Digital Right Management Policy*.

R44 expressAssessment

Definition: The relation connecting *Quality Parameters* to the *Actors* who are expressing an assessment of a *Resource*.

Rationale: The *expressAssessment* relation models the fact that a *Quality Parameter* serves the purpose of recording the judgement of an *Actor*, who is the active subject expressing an opinion about some feature of a *Resource*, which is the object under examination.

Examples:

- See *Quality Parameter*, *Actor* and *Resource*.

R45 evaluatedBy

Definition: The relation connecting *Quality Parameters* to the *Measures* according to which they are evaluated.

Rationale: The <*evaluatedBy*> relation defines the process followed to determine the assessment of a *Resource* with respect to the specific feature taken into consideration by a *Quality Parameter*. This relation takes into account that different *Measures* can be used for assessing the same *Quality Parameter*.

Examples:

- For example, an *Objective Measure* of the *Performance* of a given *Function* is its response time; an *Objective Measure* of the *Usability* of a given *Function* is the time needed to complete a task, while a *Qualitative Measure* and *Subjective Measure* of it is a score expressed by a user on a like-dislike scale.

R46 measuredBy

Definition: The relation connecting *Quality Parameters* to the *Measurements* that assign them a value.

Rationale: See *Quality Parameter* and *Measurement*.

Examples:

- See *Quality Parameter* and *Measurement*.

R47 affectedBy

Definition: The relation connecting *Quality Parameters* to other *Resources* that influence their determination.

Rationale: *Quality Parameters* and *Resources* are interrelated concepts not only in the sense that each *Resource* can be associated with one or more *Quality Parameters* that characterise their features, as expressed by the <*hasQuality*> relation, but also that *Resources* may affect and impact the assessment expressed by a *Quality Parameter*.

Examples:

- For example, *Security Enforcement* is affected by other *Quality Parameters*, such as *User Behaviour*, by *Policies*, such as *Digital Rights Managements Policy*, and by *Functions*, such as *Access Information*.

R48 accordTo

Definition: The relation connecting *Measurements* to the *Measures* that define how they should be obtained.

Rationale: The <*accordTo*> relation associates an actual value computed for assessing a *Quality Parameter*, i.e. a *Measurement*, with the procedure adopted for estimating it in order to make clear and traceable how each value has been produced.

Examples:

- For example, when evaluating information access components, you may need to specify that the value 0.3341 (the *Measurement*) is the Mean Average Precision computed by the trec_eval[30] tool, which truncates the computation after 1,000 retrieved documents (the *Measure*).

R49 implement

Definition: The relation connecting *Architectural Components* to the *Resources* they realise.

Rationale: The <*implement*> relation associates notion of *Resource* with the *Architectural Component* (another *Resource*) that makes it real, put it into effect. This notion of 'implementation' covers the whole spectrum of possible interpretations ranging from the implementation of a *Policy* or a *Function* (meaning that the *Architectural Component* contains the logic to put into effect the *Policy* or the *Function*) to the implementation of an *Information Object* (meaning that through the *Architectural Component* it is possible to have access to the *Information Object*).

The same *Resource* can be implemented by many *Architectural Components* as well as an *Architectural Component* can implement many *Resources*.

Examples: --

R50 realisedBy

Definition: The relation connecting *Software Components* to the *Running Components* realising them.

Rationale: This relation is a specialisation of the <*implement*> relation in the context of *Software Components* and *Running Components*. In particular, it is used to capture the fact that a *Running Component* implements one or more *Software Components* thus it is the process putting into effect what is coded in the software artifact.

A *Software Components* can be put in action by zero or more *Running Components*; a *Running Component* can put in action one or more *Software Components*.

Examples: --

R51 support

Definition: The relation connecting *Application Frameworks* to the *Running Components* that support the operation.

[30] http://trec.nist.gov/

Rationale: This relation is a specialisation of the <*implement*> relation in the context of *Application Frameworks* and *Running Components*. In particular, it is used to capture the fact that a *Running Component* implements zero or more *Application Frameworks,* thus it is the process putting into effect the software connecting and supporting the operation of other *Software Components* forming the system.

An *Application Framework* can be put in action by zero or more *Running Components*; a *Running Component* can put in action zero or more *Application Frameworks.*

Examples: --

R52 hostedBy

Definition: The relation connecting *Running Components* to the *Hosting Nodes* physically hosting them.

Rationale: This relation is a specialisation of the <*implement*> relation in the context of *Running Components* and *Hosting Nodes*. In particular, it is used to capture the fact that a *Hosting Node* hosts zero or more *Running Components* and thus by providing them with the environment supporting their operation it puts them into action.

A *Running Component* is hosted on one *Hosting Node* a time; a *Hosting Node* can put in action zero or more *Running Components.*

Examples: --

Conclusions

This volume has presented the DELOS Reference Model for Digital Libraries. It consists of separate parts that illustrate the DELOS model from different perspectives and at different levels of abstraction. This structure has been introduced to accommodate the needs of the many different players of the Digital Library universe who are interested in understanding Digital Library 'systems' at different levels of detail.

The model presented is the result of a three-year effort aimed at contributing to the ambitious process of laying foundations for Digital Libraries as a whole. Research on 'digital libraries' addresses many different areas. The lack of any agreement on the foundations for this broad research field has led to a plethora of systems, methodologies and results that are difficult to combine and reuse to produce enhanced outcomes.

The model illustrated draws on the understanding of Digital Library Systems acquired by several European research groups active in the Digital Library field for many years, both within the DELOS Network of Excellence and beyond, as well as by other groups around the world. In such a context, it is intended as a collective effort by the Digital Library community to agree on common ground. This is meant to be useful not only for current activities but also as a springboard for future work.

The model presented is an abstract framework for understanding significant relationships among the entities of the Digital Library Universe, and for the development of consistent standards or specifications supporting the different elements of this universe. It aims at providing a common semantics that can be used unambiguously across and between different application areas both to explain and organise existing Digital Library 'systems' and to support the evolution of research and development in this area.

Because of the broad coverage of the Digital Library field, the heterogeneity of the actors involved, and the lack of any previous agreement on the foundations of the field, the DELOS Reference Model should be considered as a living document shared by the Digital Library community. For this reason, this document is to be made available in its subsequent versions, each taking advantage of the previous one and of the public comments received.

The framework introduced so far does not aim to cover every specific aspect of the Digital Library universe. Rather, its objective is to provide a core context representing the main aspects of these systems. Other specific aspects can easily be modelled by building on this core part and by introducing more detailed concepts and relationships. We expect that in the future many more focused, fine-grained models, developed by other communities, will be progressively integrated into the present model, thus creating an increasingly richer framework capable of capturing more and more aspects of the Digital Library universe.

Appendix A. Concept Maps in A4 format

A.1. DL Resource Domain Concept Map

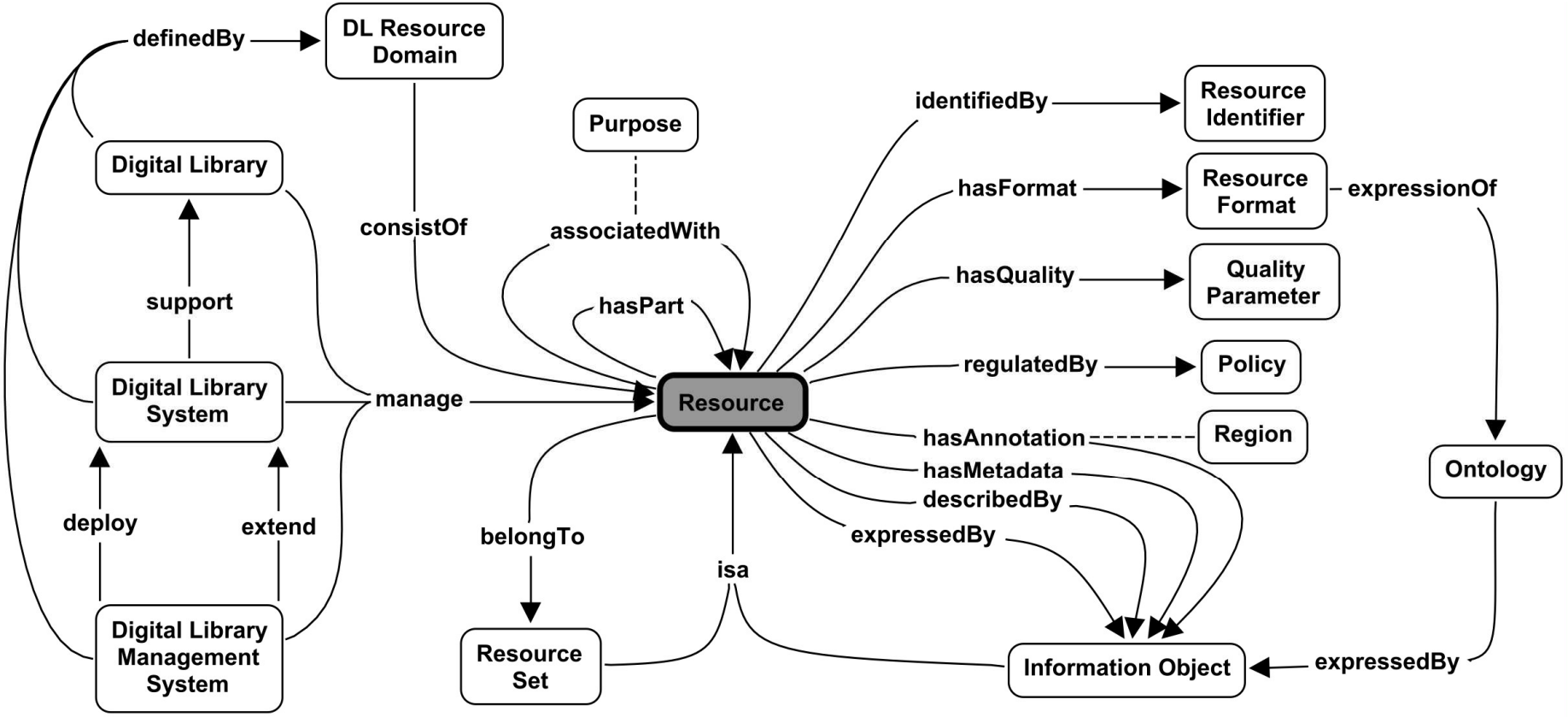


Figure A-1. Resource Domain Concept Map (A4 format)

A.2. Content Domain Concept Map

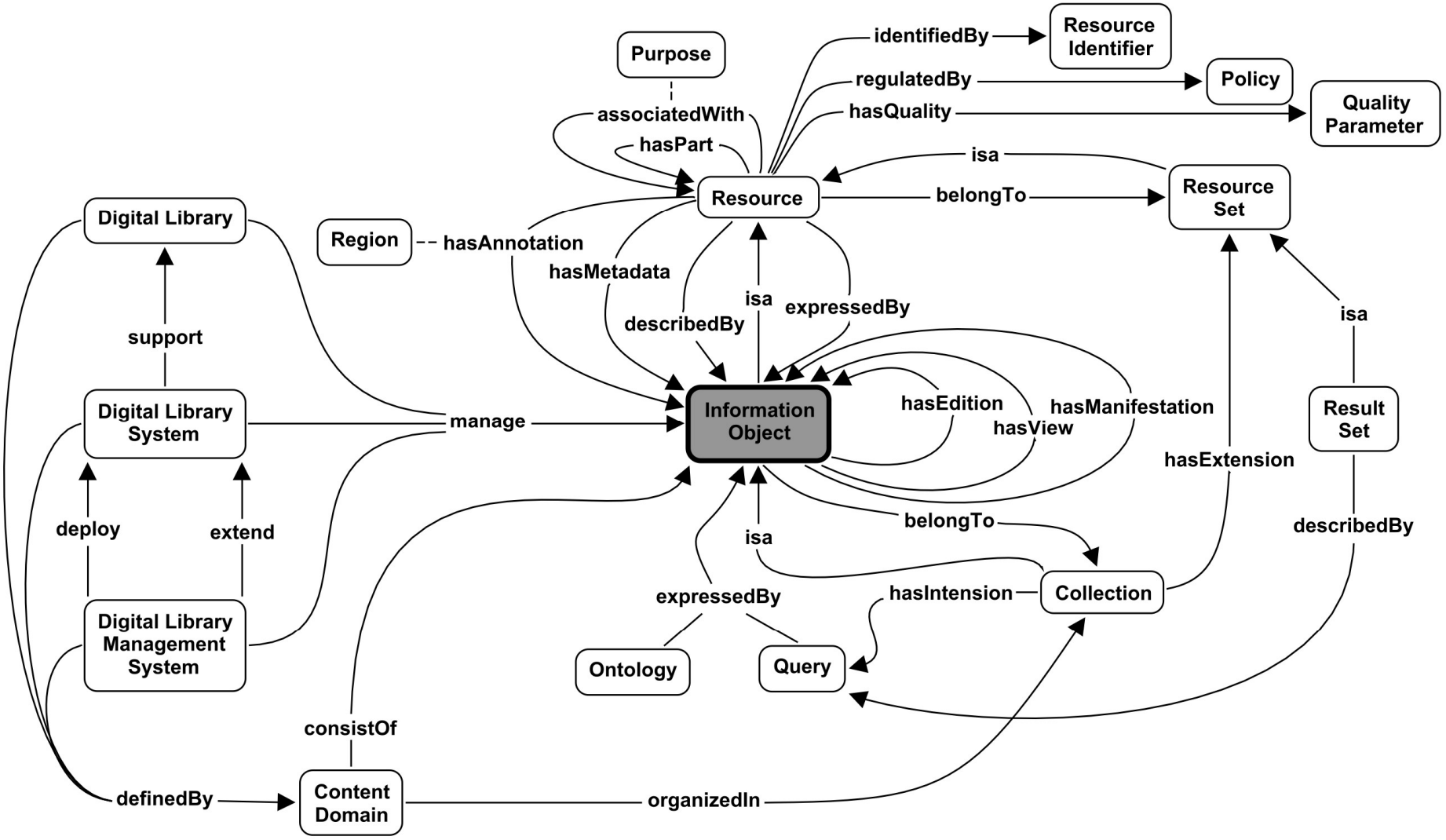


Figure A-2. Content Domain Concept Map (A4 format)

A.3. User Domain Concept Map

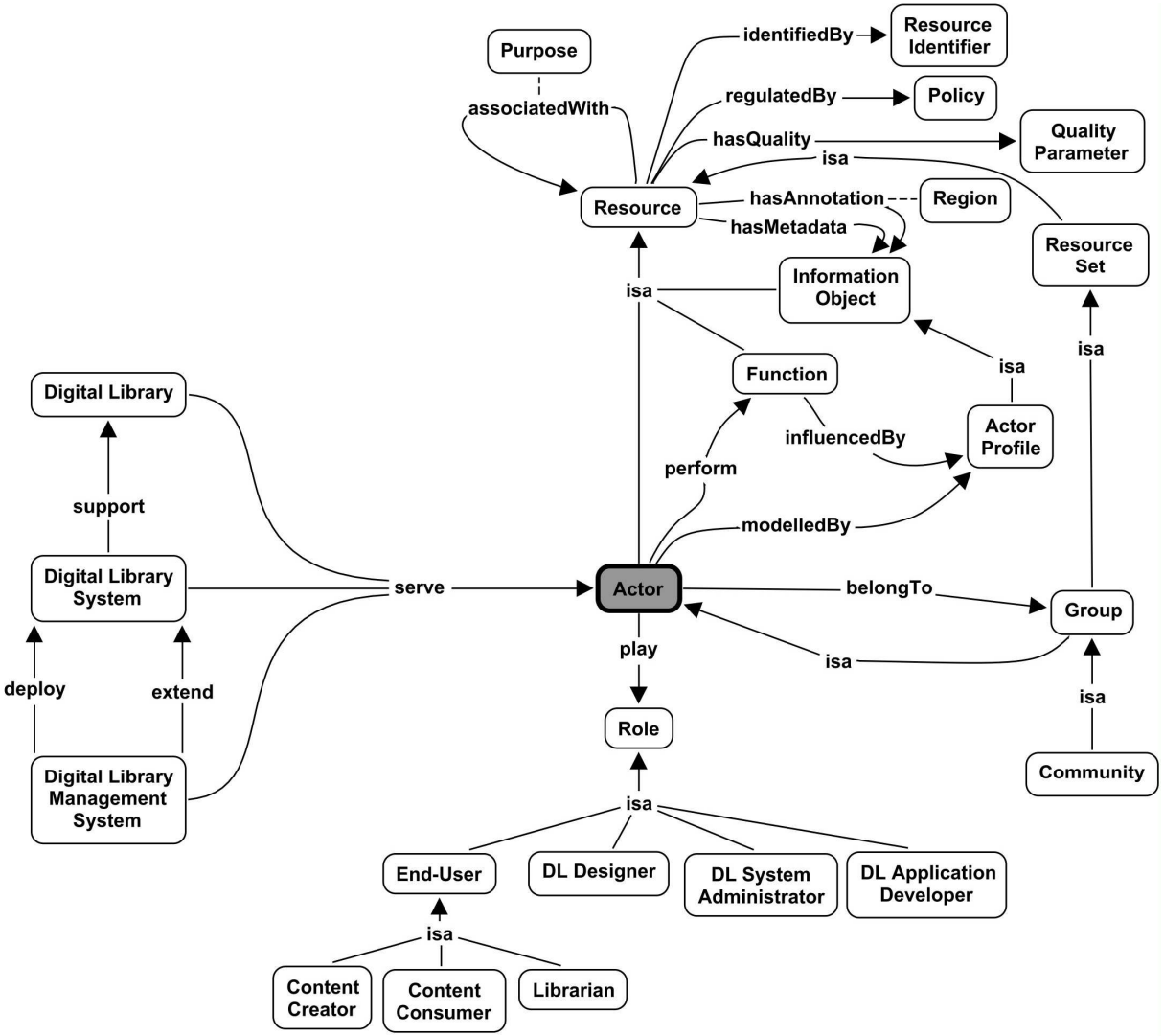


Figure A-3. User Domain Concept Map (A4 format)

A.4. Functionality Domain Concept Map

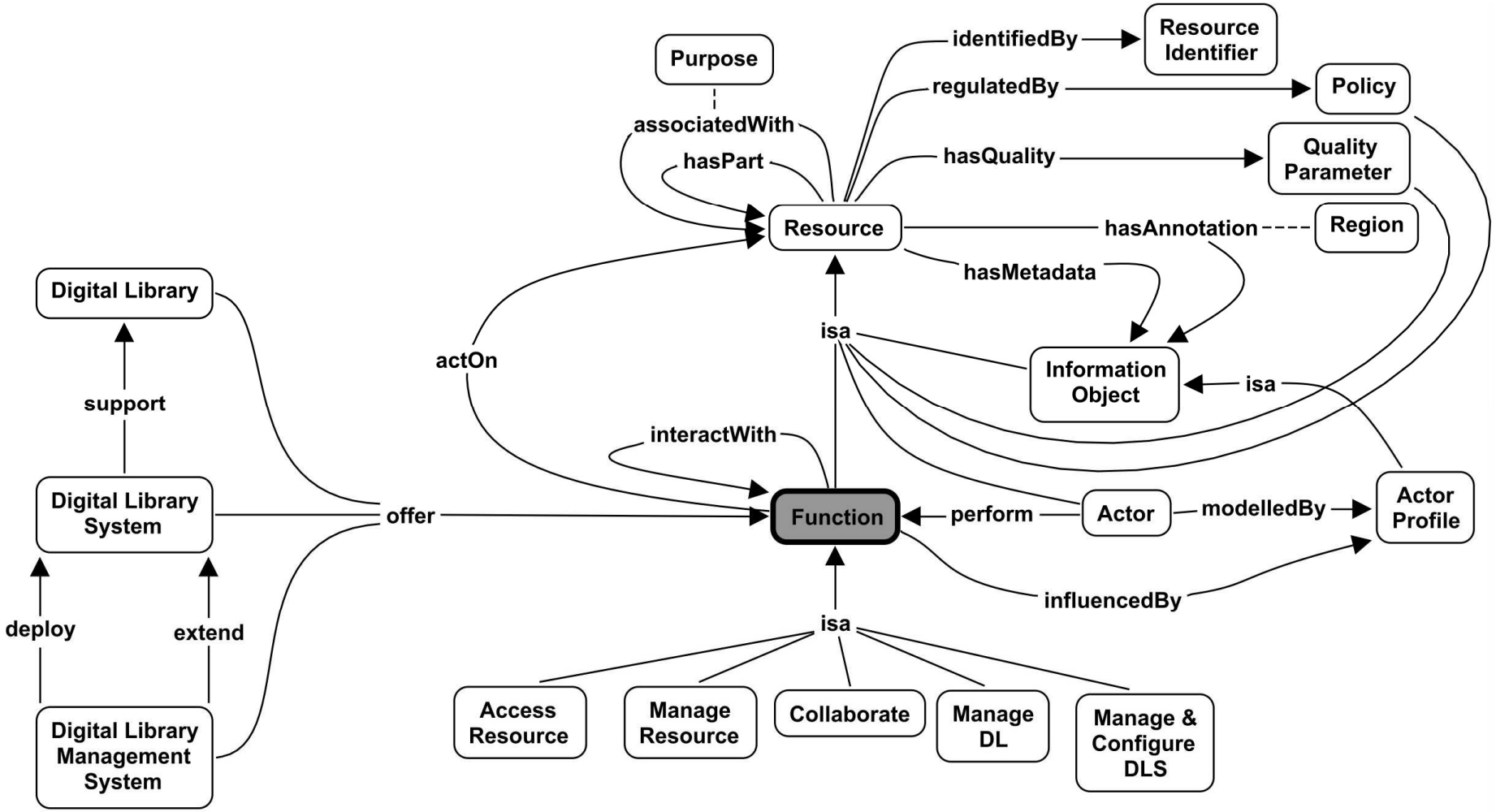


Figure A-4. Functionality Domain Concept Map (A4 format)

A.5. Functionality Domain Concept Map: Access Resource Functions

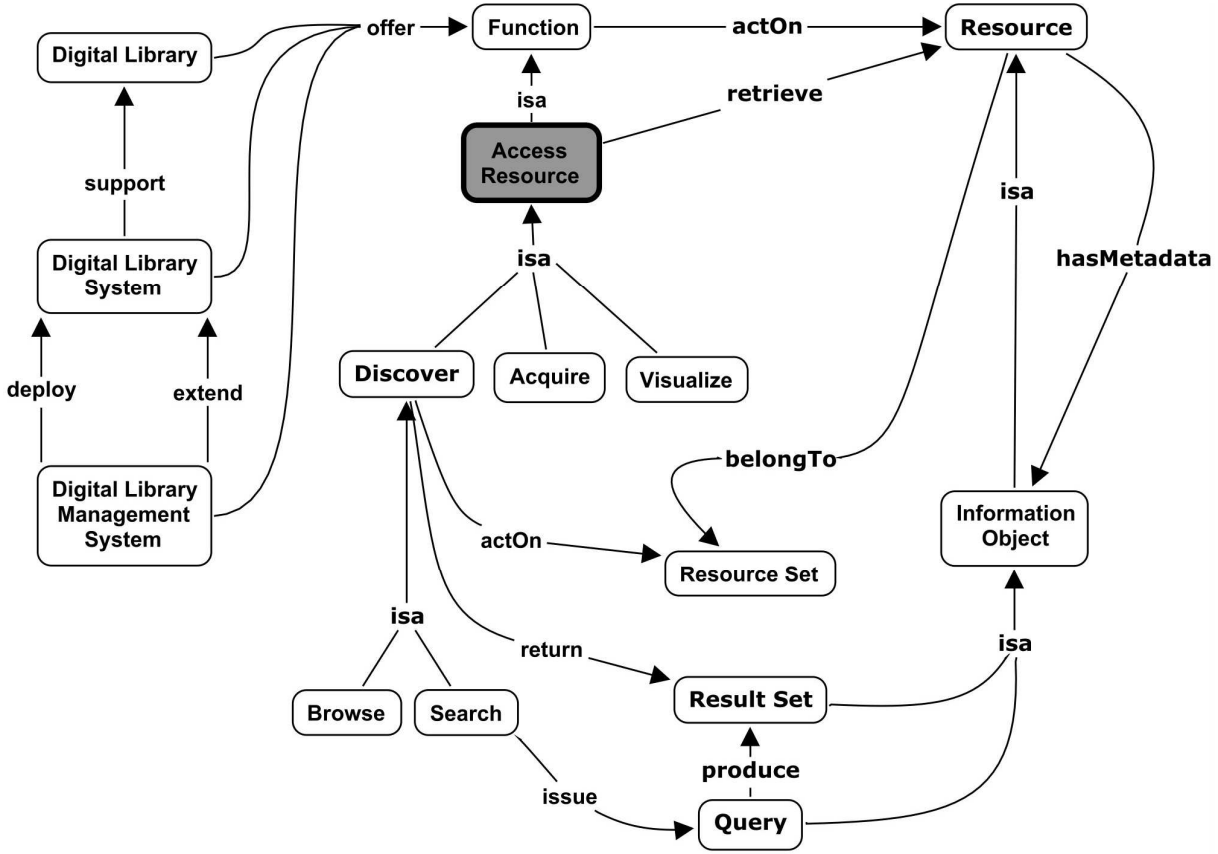


Figure A-5. Functionality Domain Concept Map: Access Resource Functions (A4 format)

A.6. Functionality Domain Concept Map: Specialisations of the Manage Resource Function

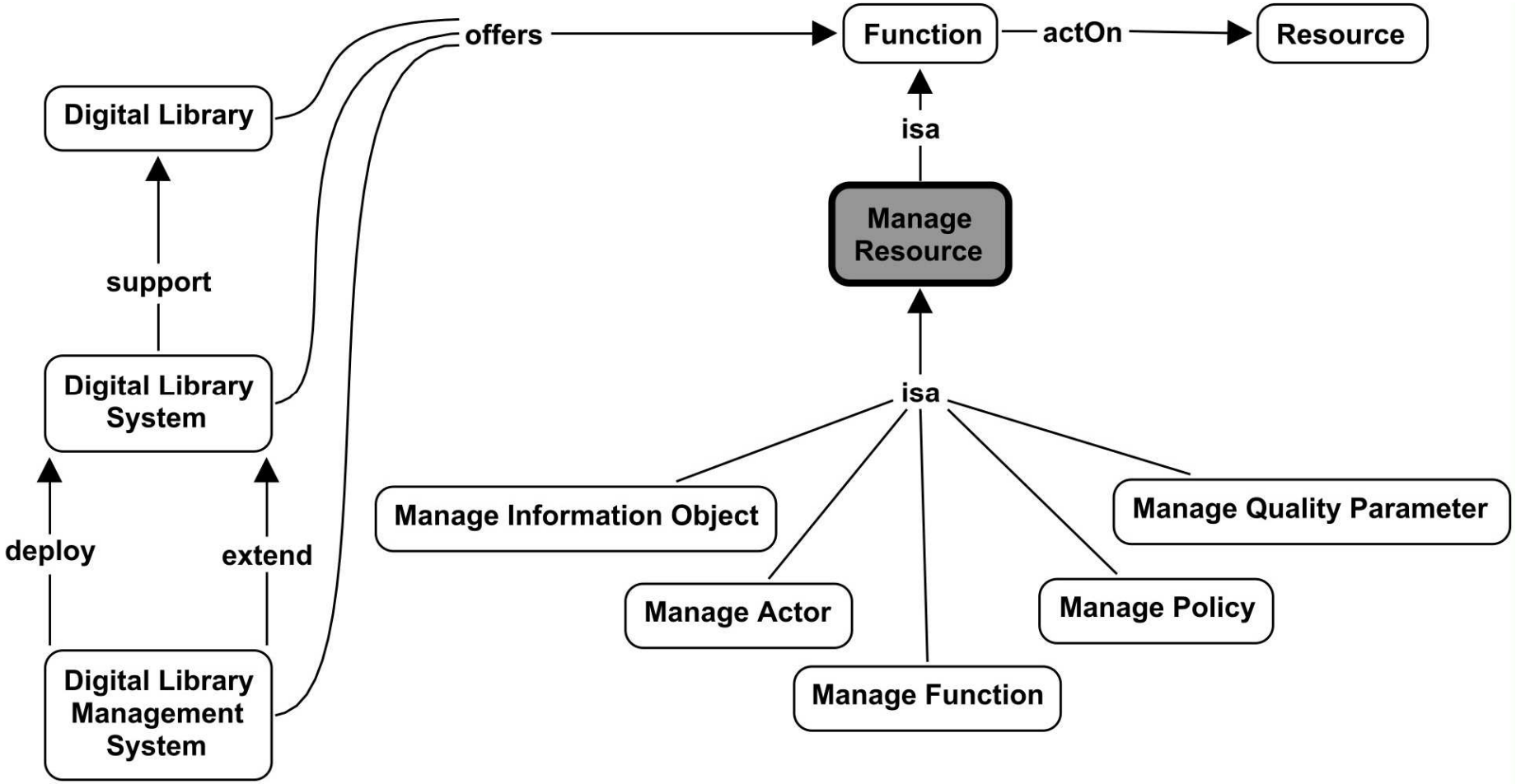


Figure A-6. Functionality Domain Concept Map: Specialisations of the Manage Resource Function (A4 format)

A.7. Functionality Domain Concept Map: General Manage Resource Functions

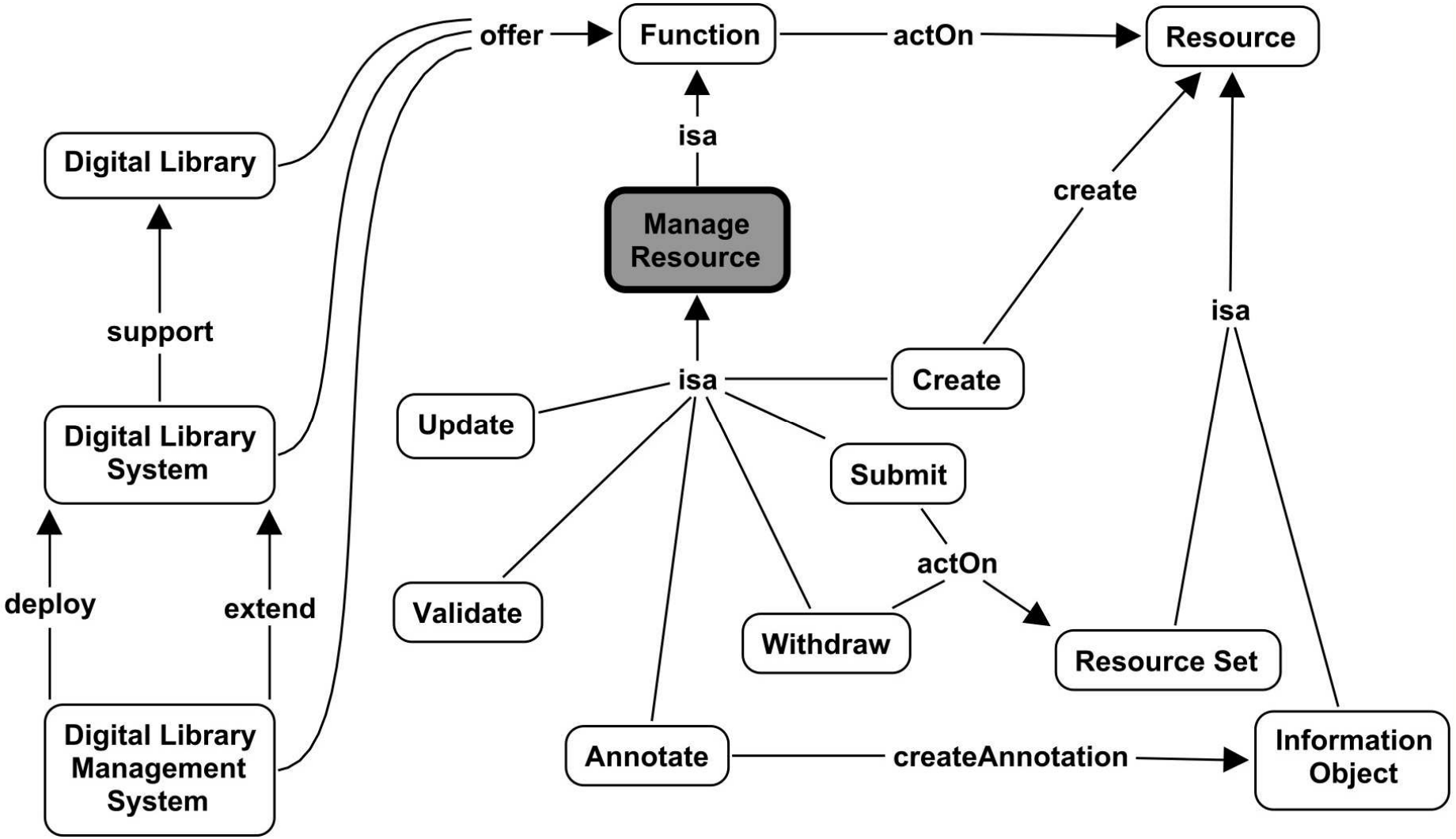


Figure A-7. Functionality Domain Concept Map: General Manage Resource Functions (A4 format)

A.8. Functionality Domain Concept Map: Manage Information Object Functions

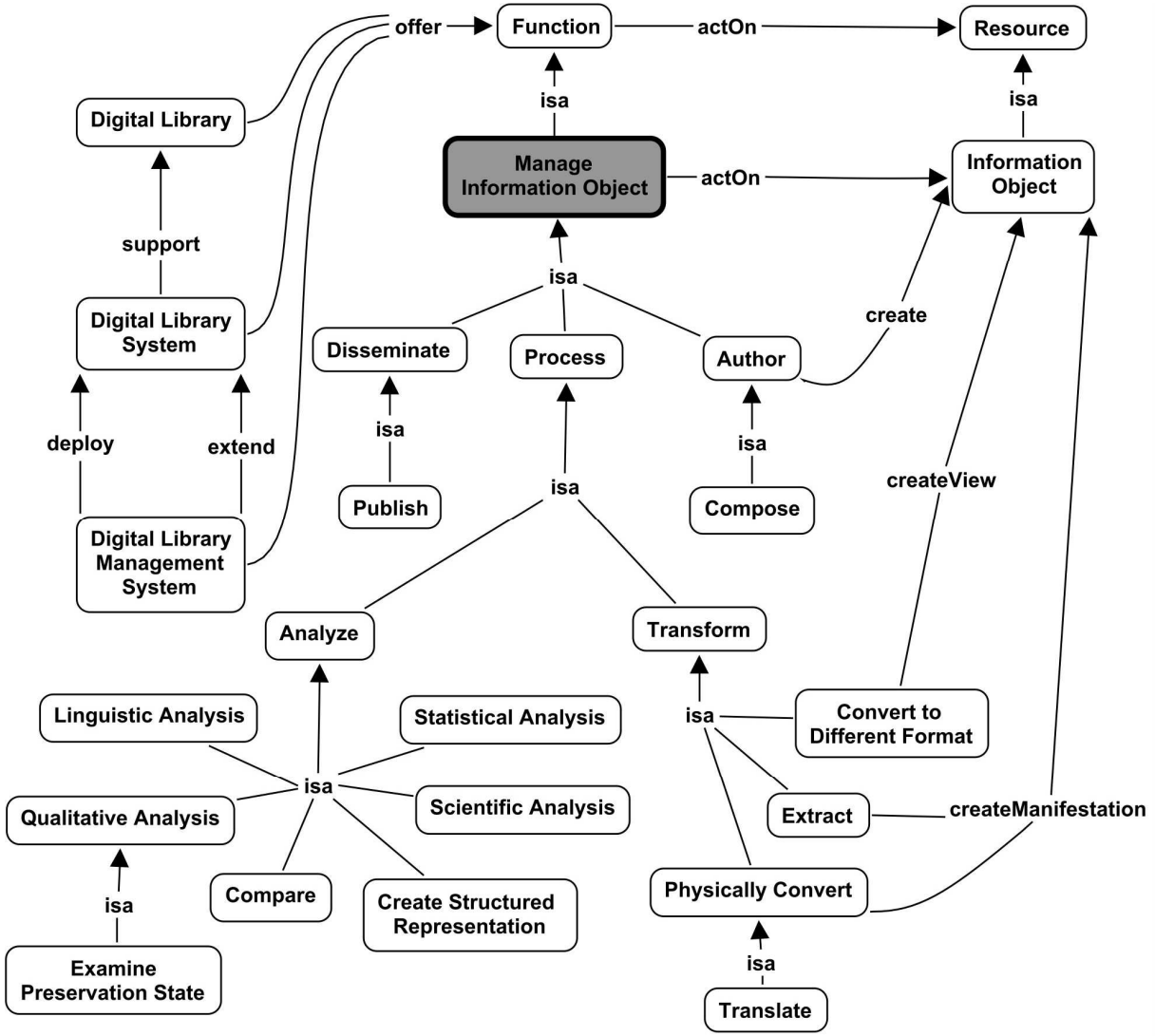


Figure A-8. Functionality Domain Concept Map: Manage Information Object Functions (A4 format)

A.9. Functionality Domain Concept Map: Manage Actor Functions

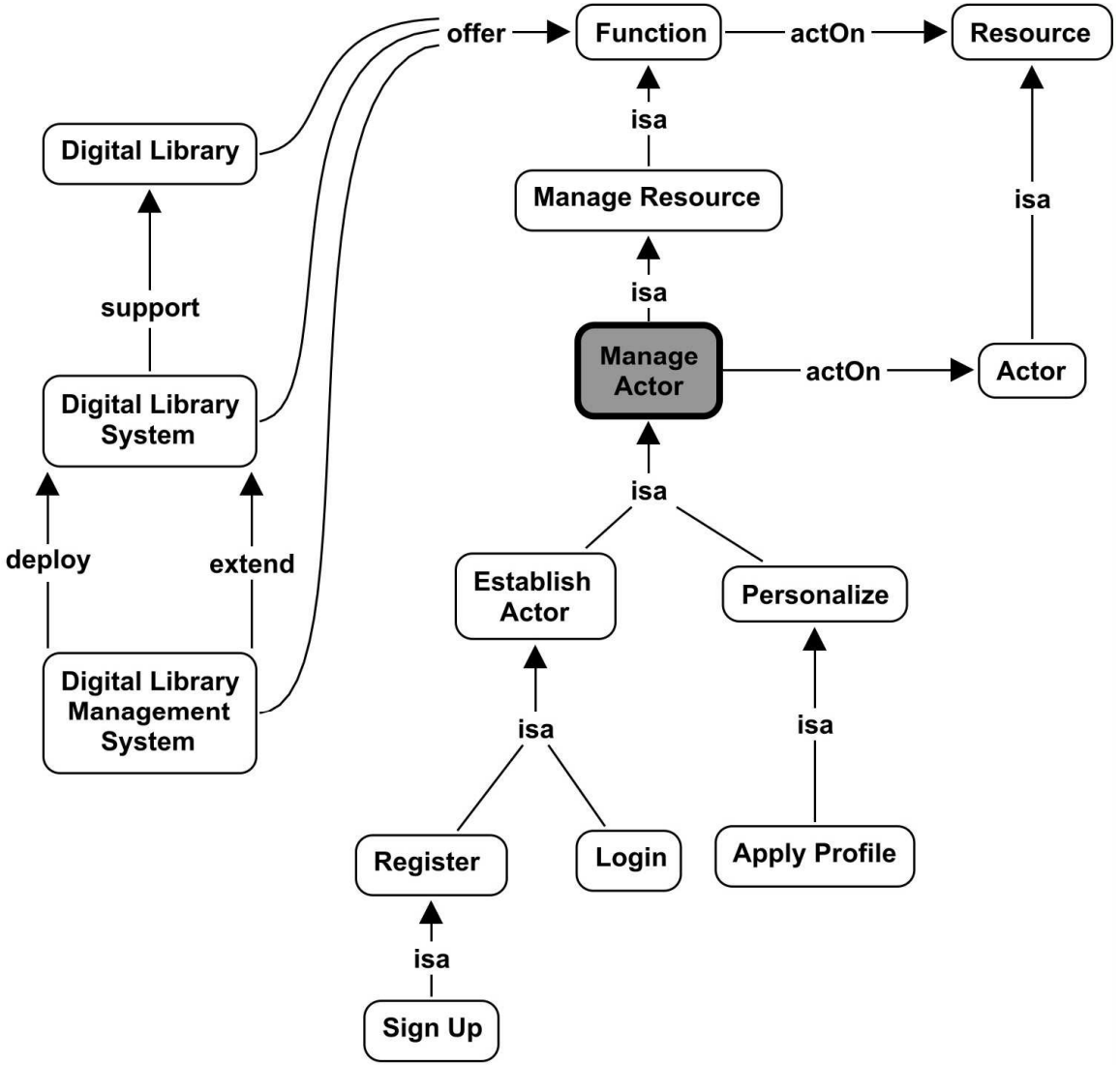


Figure A-9. Functionality Domain Concept Map: Manage Actor Functions (A4 format)

A.10. Functionality Domain Concept Map: Collaborate Functions

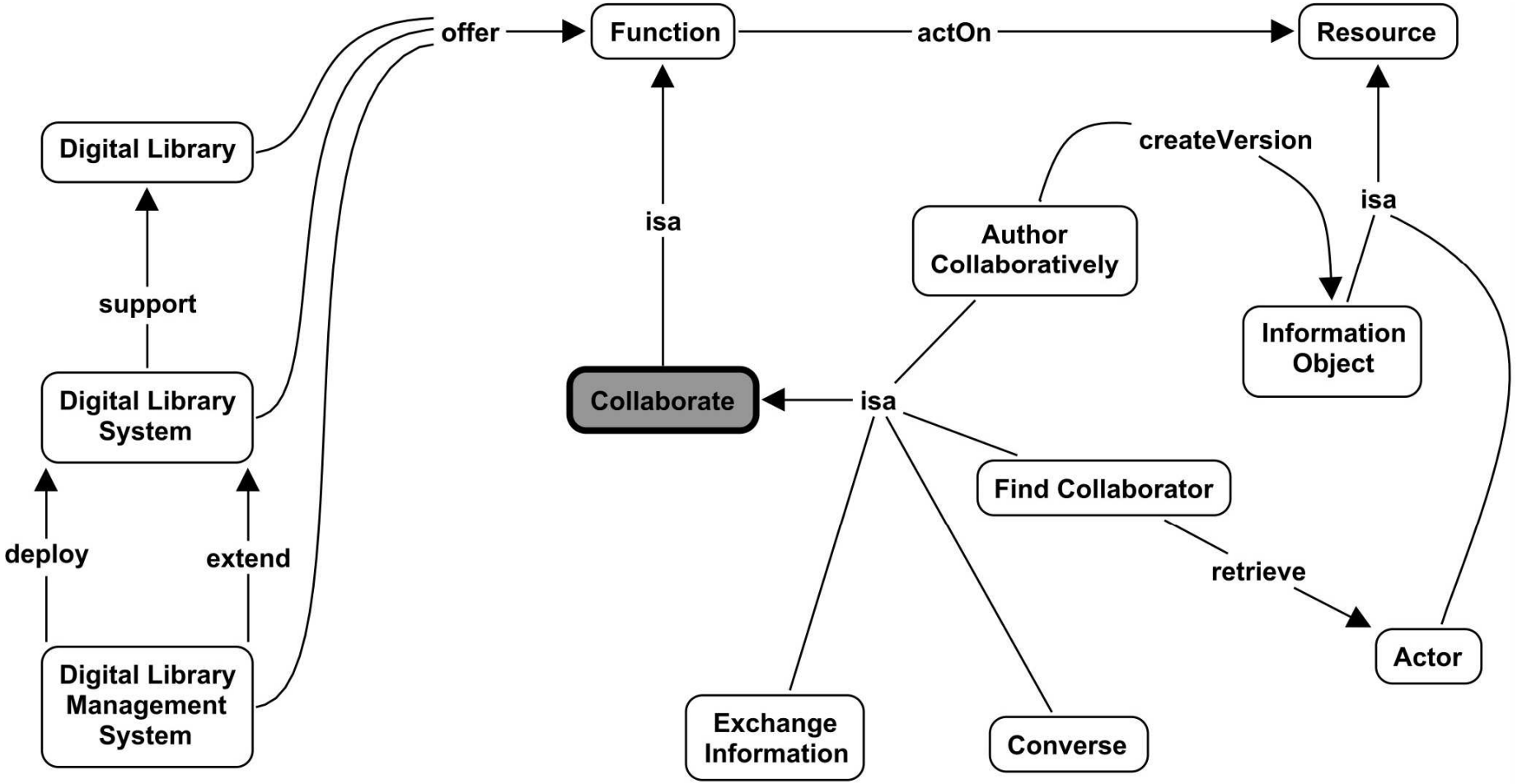


Figure A-10. Functionality Domain Concept Map: Collaborate Functions (A4 format)

A.11. Functionality Domain Concept Map: Manage DL Functions

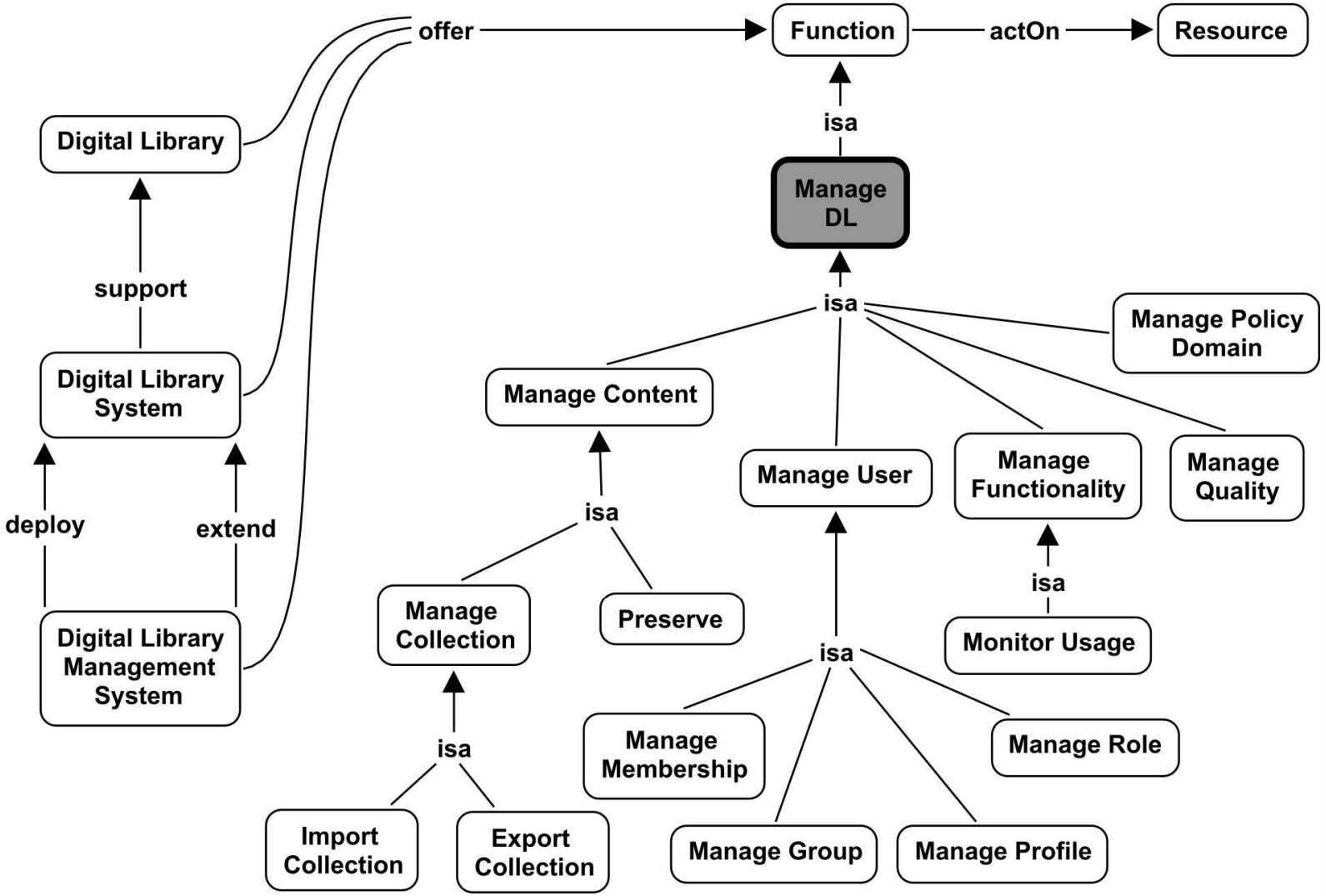


Figure A-11. Functionality Domain Concept Map: Manage DL Functions (A4 format)

A.12. Functionality Domain Concept Map: Manage & Configure DLS Functions

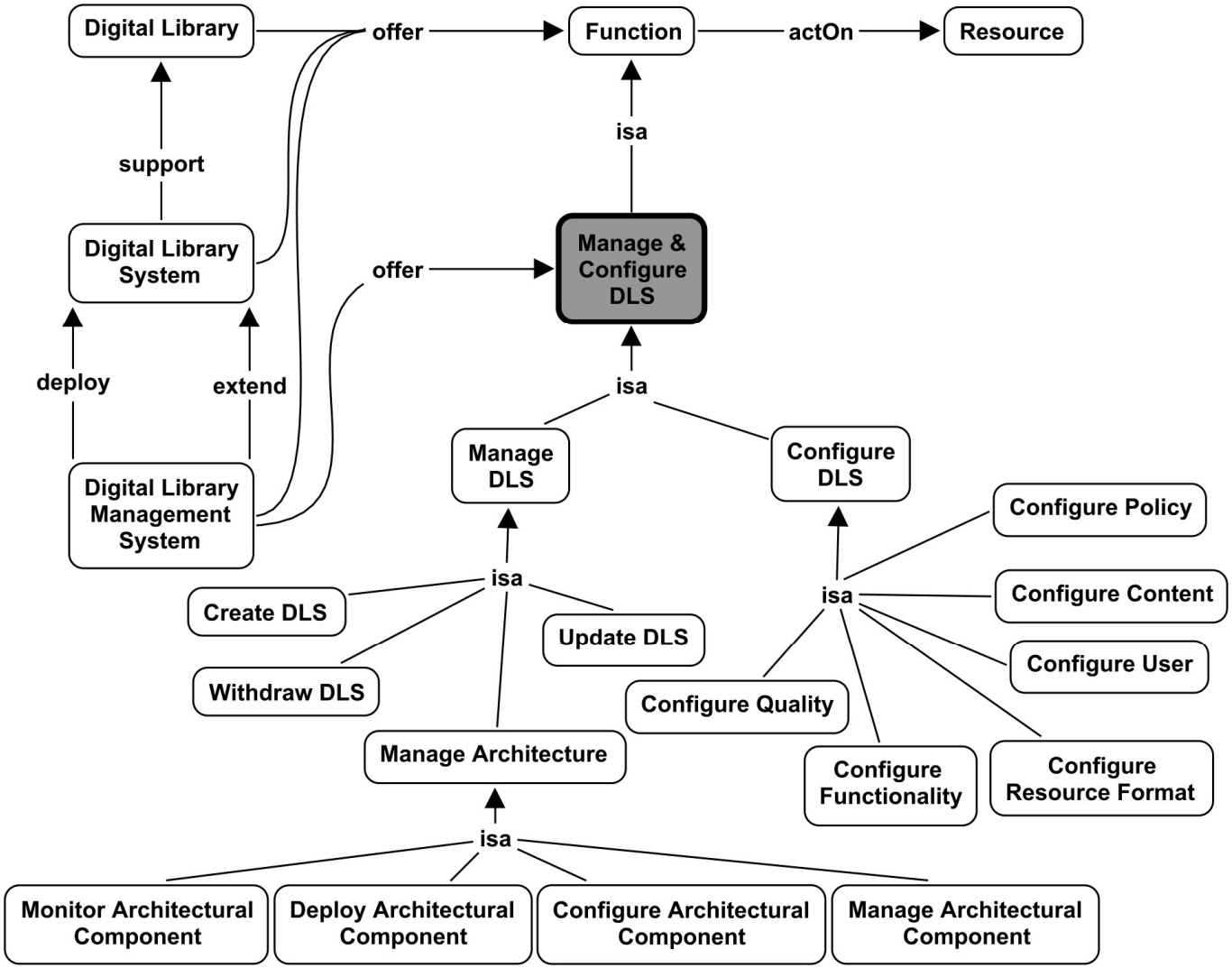


Figure A-12. Functionality Domain Concept Map: Manage & Configure DLS Functions (A4 format)

A.13. Policy Domain Concept Map

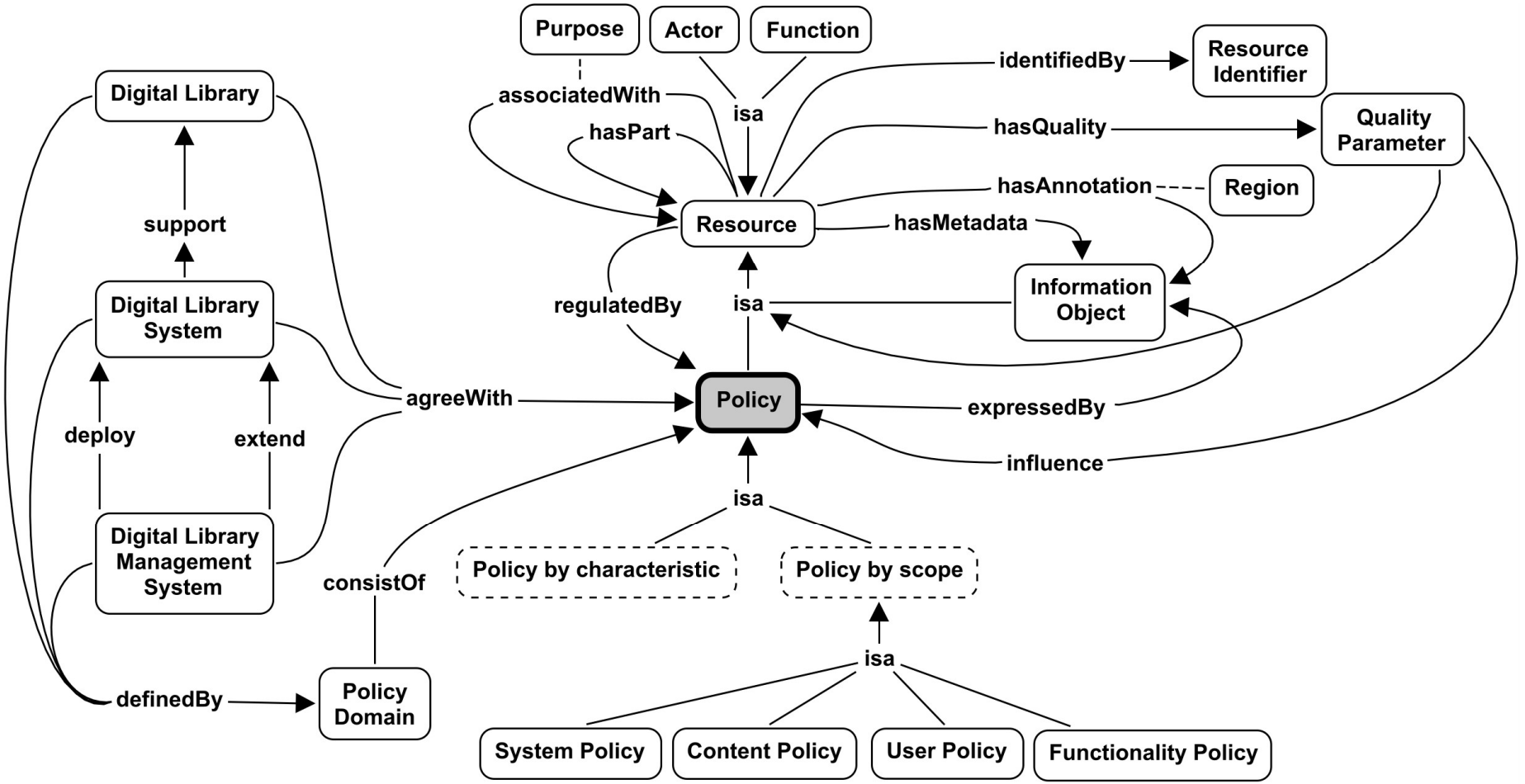


Figure A-13. Policy Domain Concept Map (A4 format)

A.14. Policy Domain Concept Map: Policies' Hierarchy

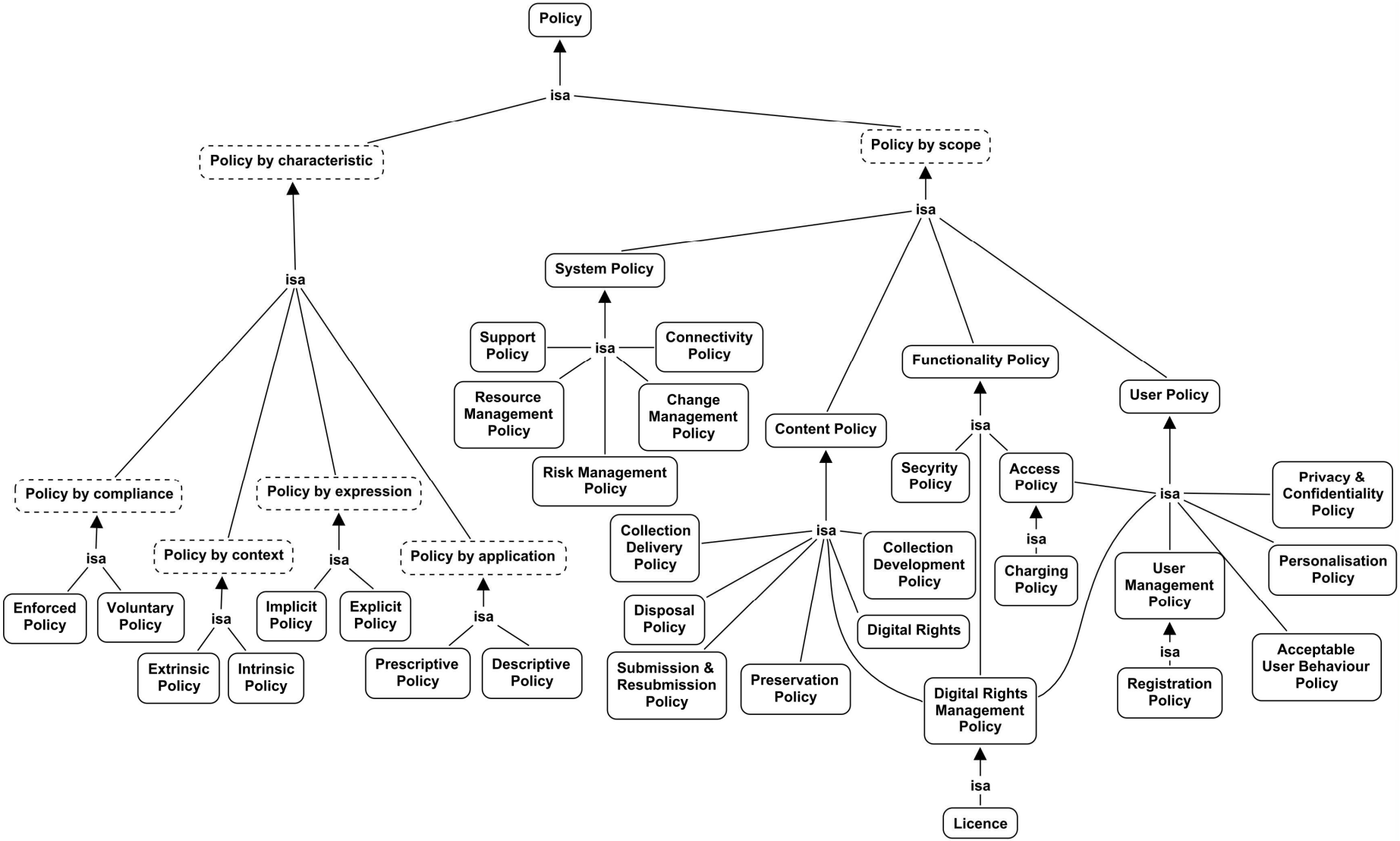


Figure A-14. Policy Domain Concept Map: Policies' Hierarchy (A4 format)

A.15. Quality Domain Concept Map

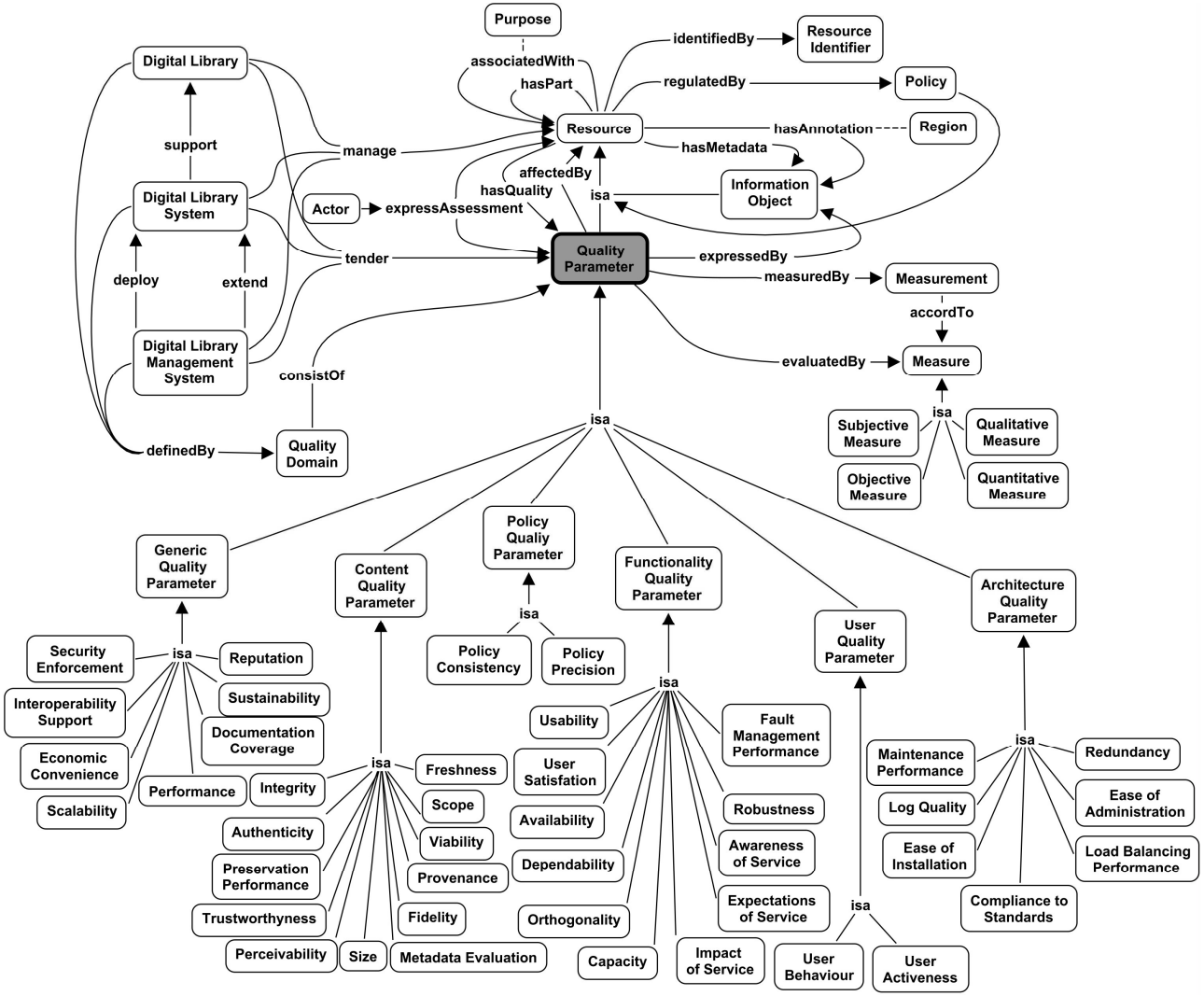


Figure A-15. Quality Domain Concept Map (A4 format)

A.16. Architecture Domain Concept Map

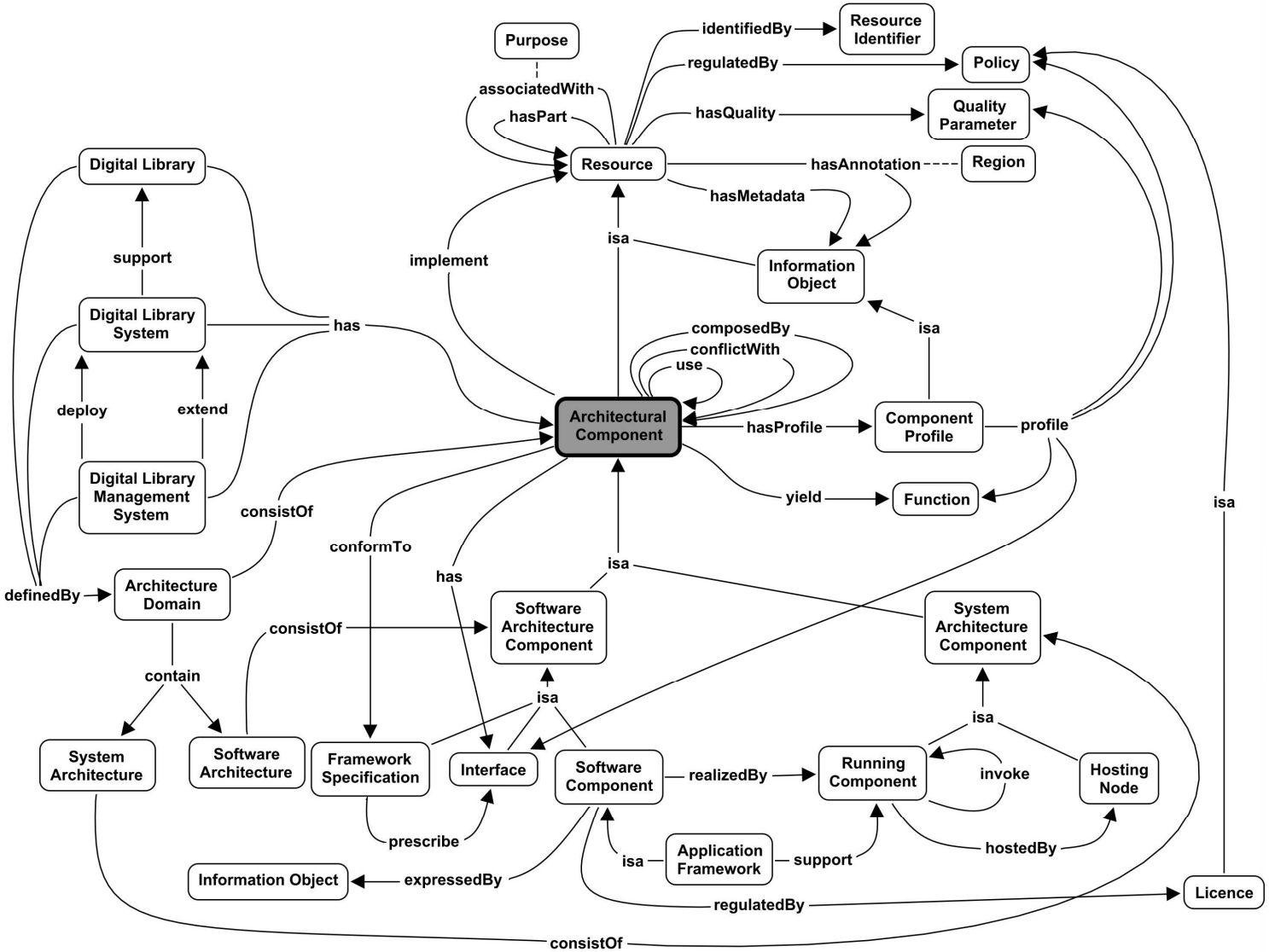


Figure A-16. Architecture Domain Concept Map (A4 format)

Appendix B. Reference Model Maps in UML

B.1. DL Resource Domain

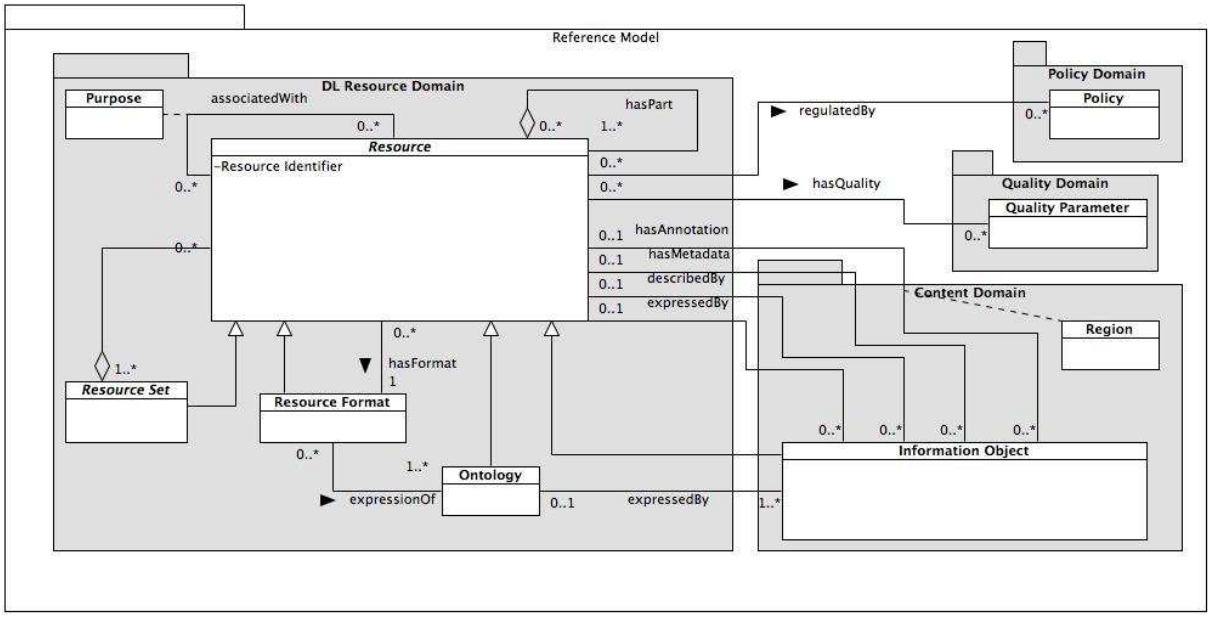


Figure B-1. DL Resource Domain UML Class Diagram

B.2. Content Domain

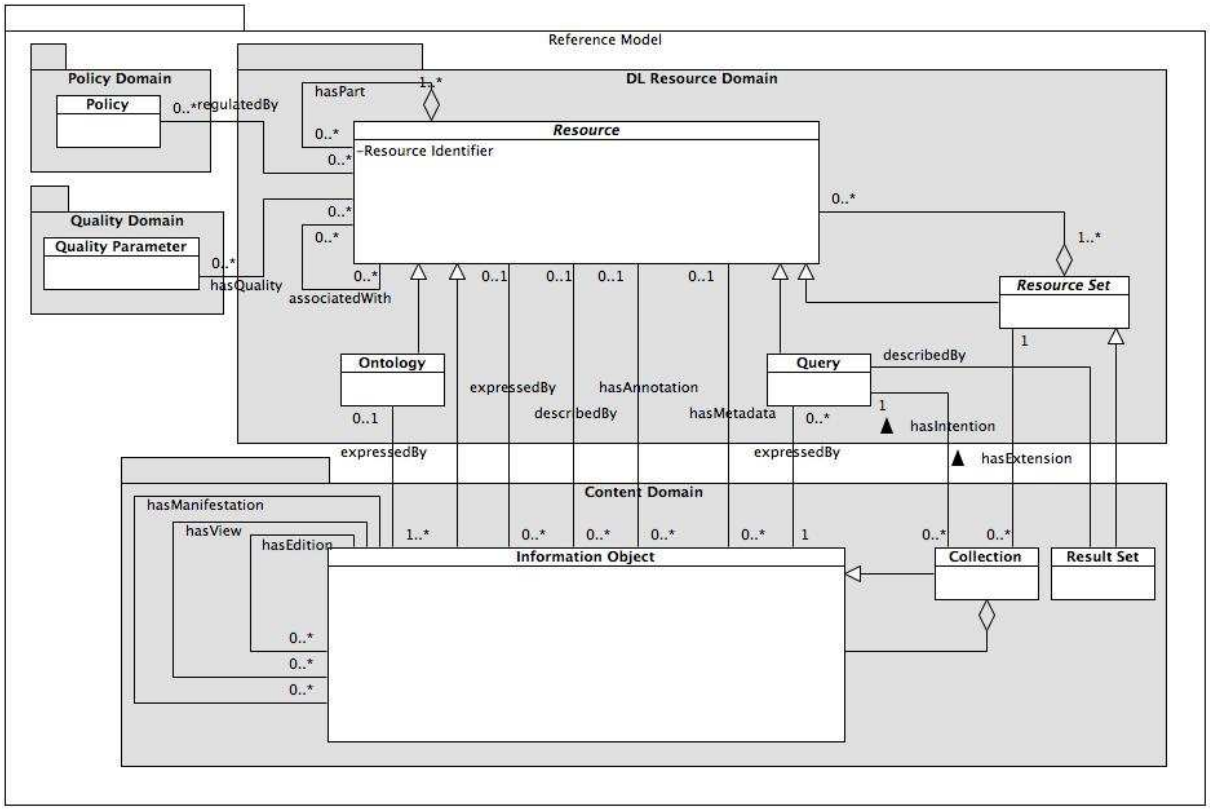


Figure B-2. Content Domain UML Class Diagram

B.3. User Domain

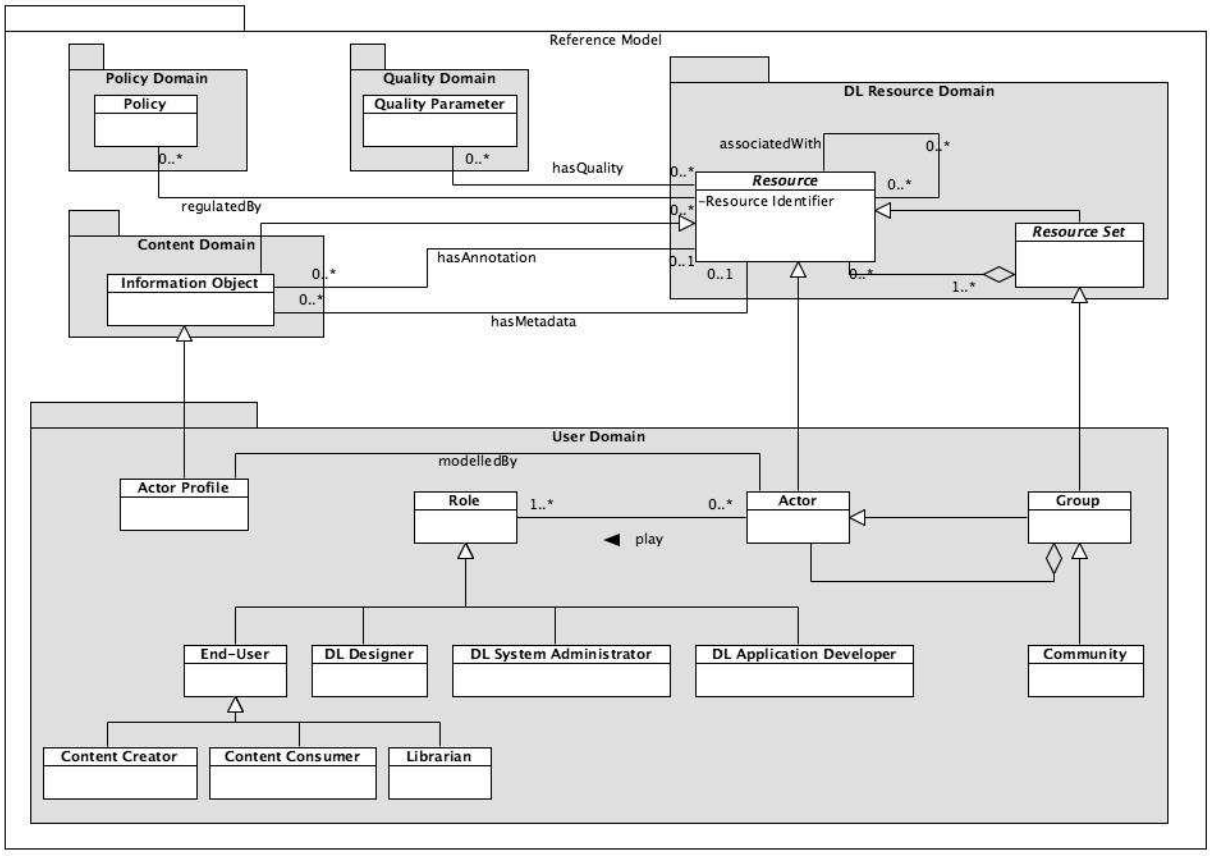


Figure B-3. User Domain UML Class Diagram

B.4. Functionality Domain

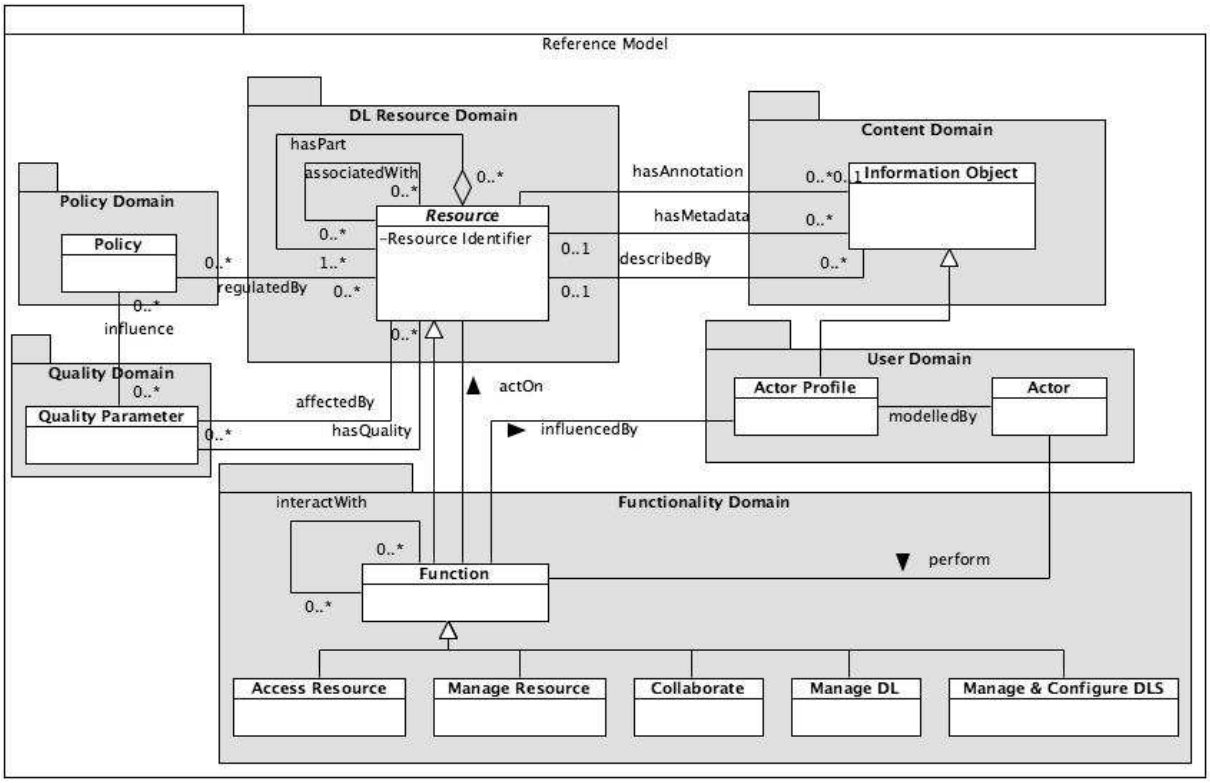


Figure B-4. Functionality Domain UML Class Diagram

B.5. Functions Hierarchy

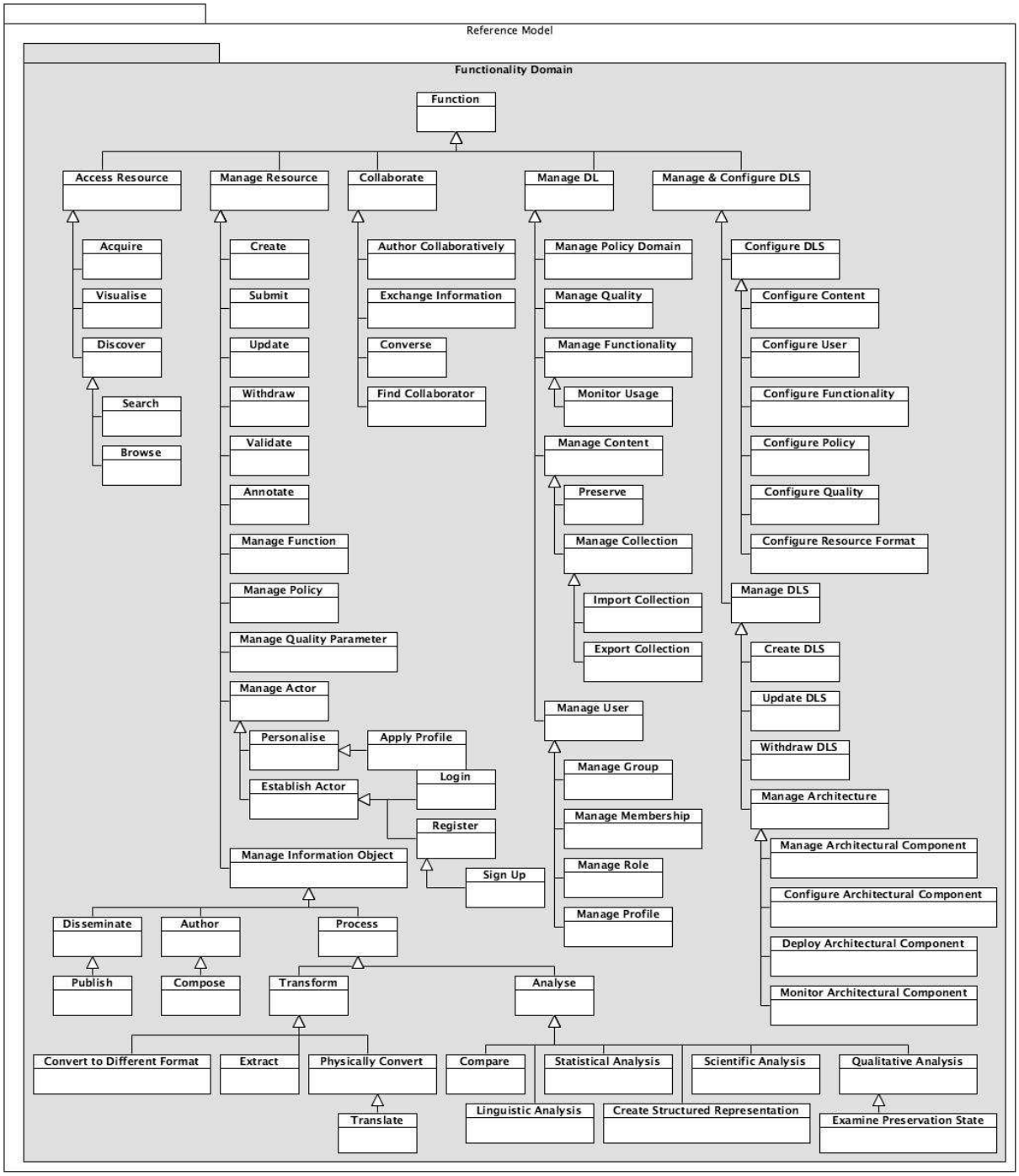


Figure B-5. Functions Hierarchy UML Class Diagram

B.6. Policy Domain

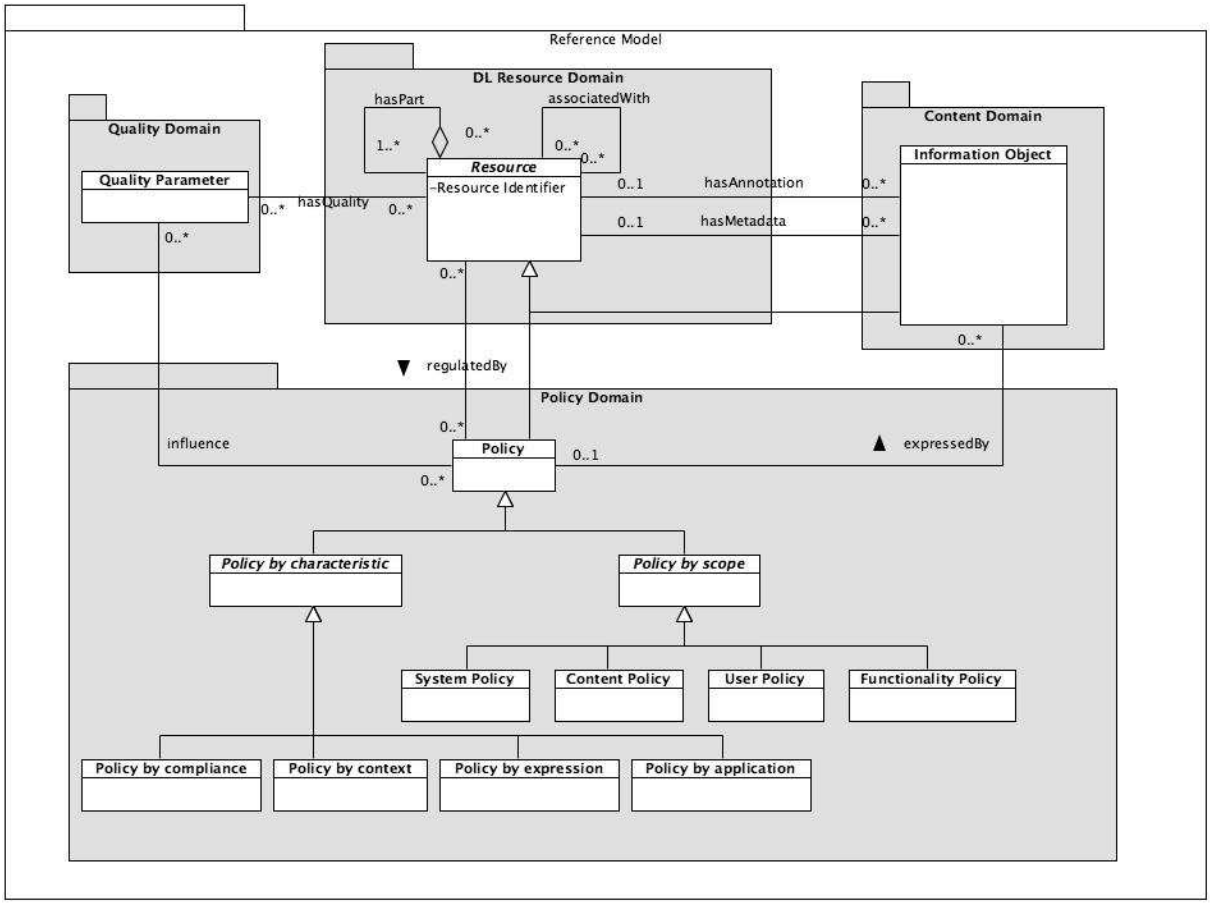


Figure B-6. Policy Domain UML Class Diagram

B.7. Policy By Characteristic Hierarchy

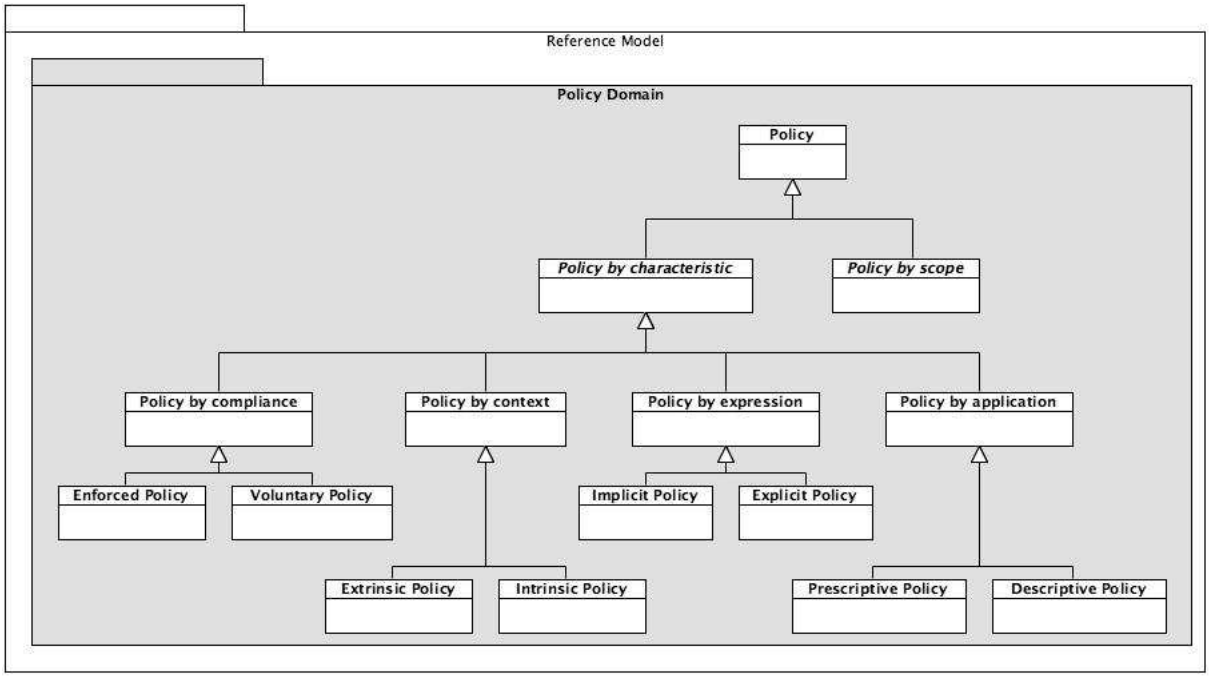


Figure B-7. Policy by Characteristic Hierarchy UML Class Diagram

B.8. Policy By Scope Hierarchy

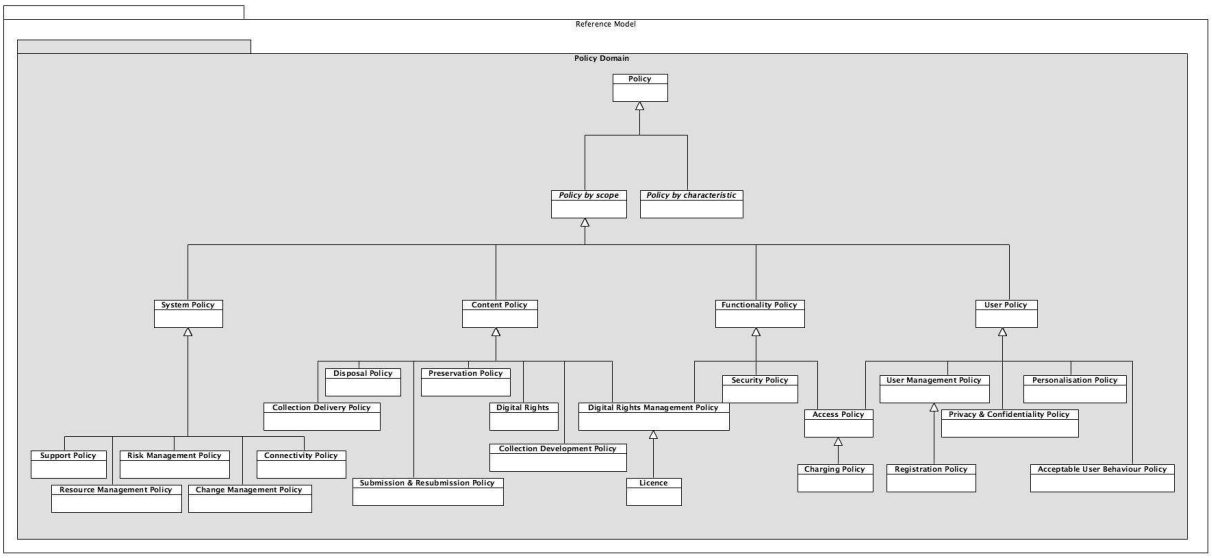


Figure B-8. Policy By Scope Hierarchy UML Class Diagram

B.9. Quality Domain

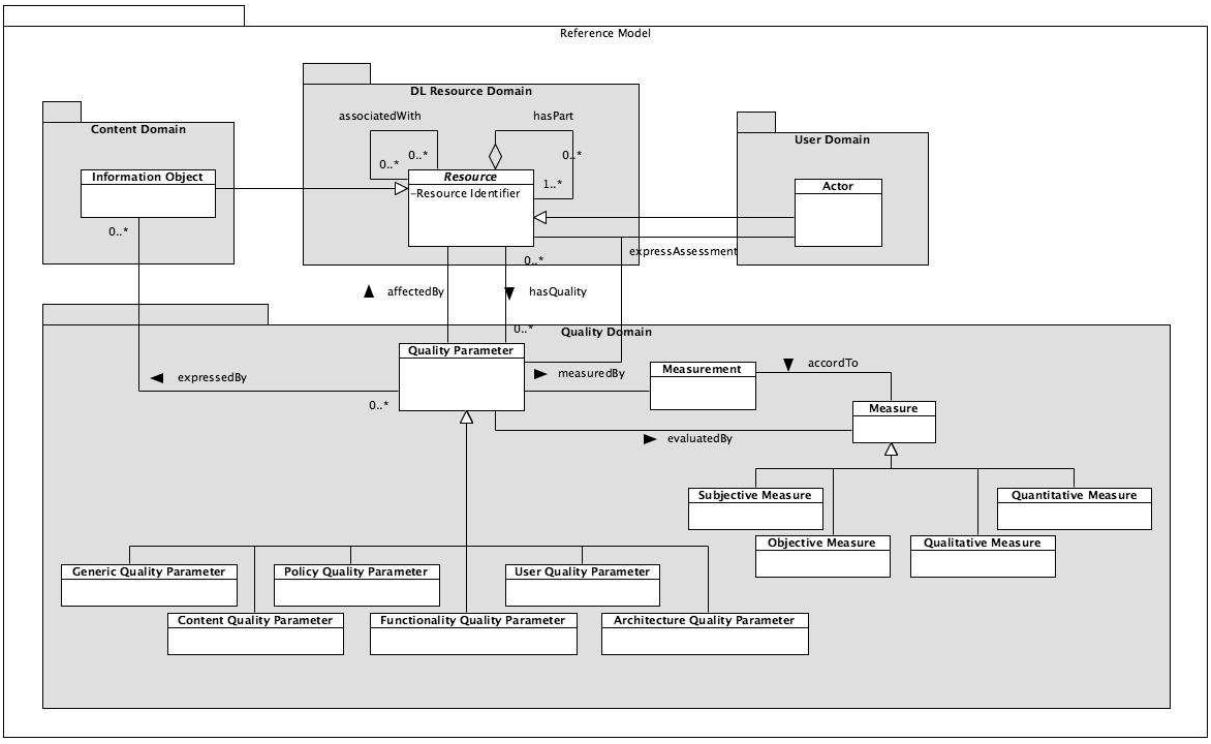


Figure B-9. Quality Domain UML Class Diagram

B.10. Quality Parameter Hierarchy

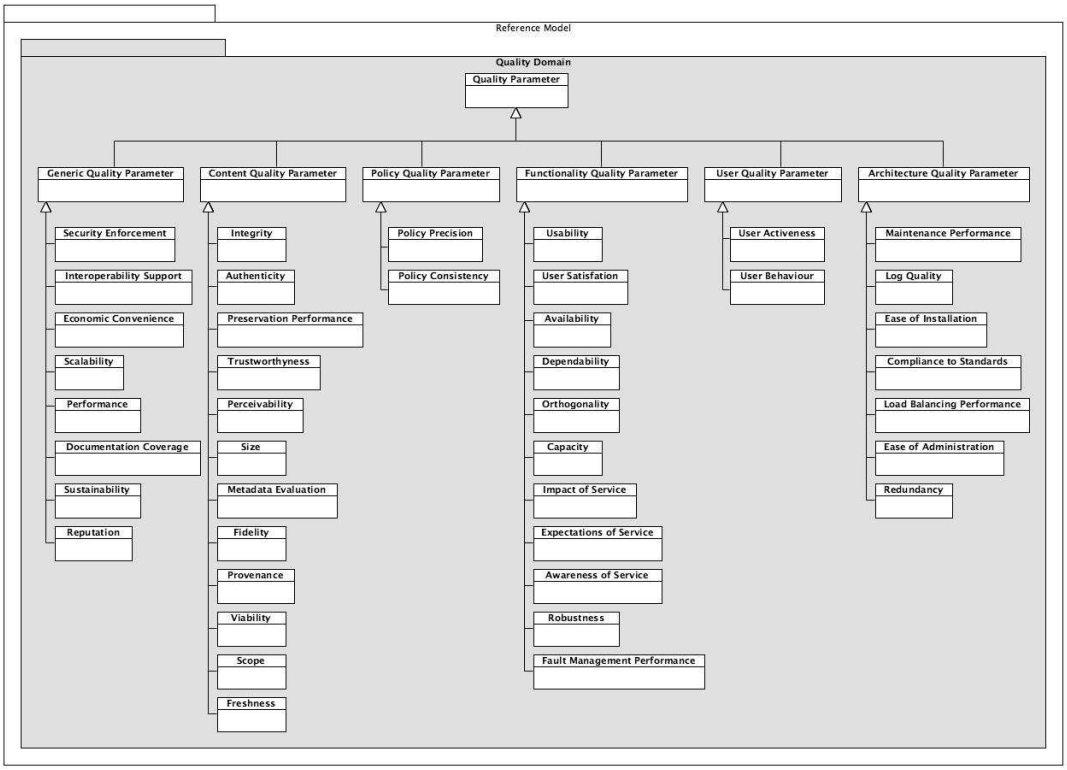


Figure B-10. Quality Parameter Hierarchy UML Class Diagram

B.11. Architecture Domain

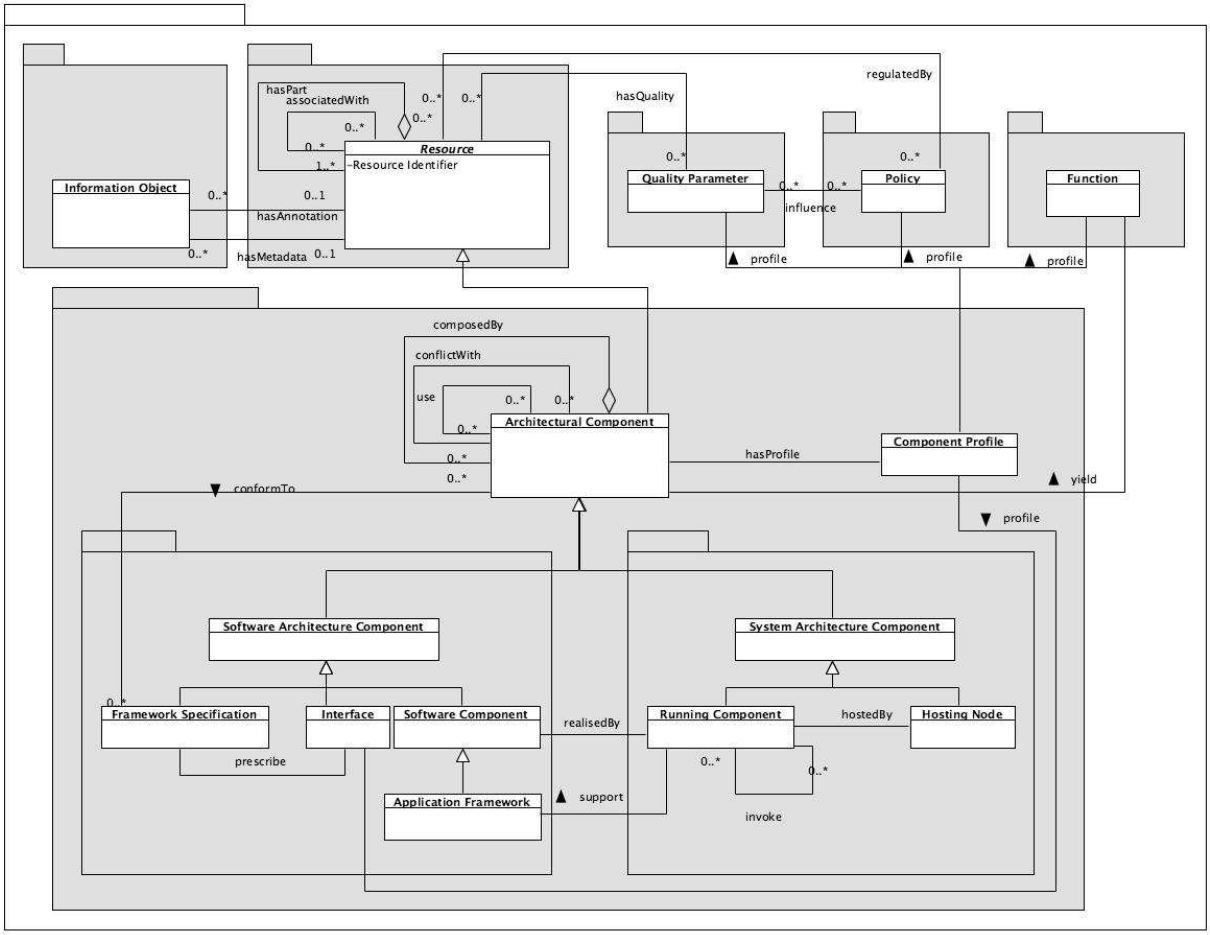


Figure B-11. Architecture Domain UML Class Diagram

Index of Concepts and Relations

Bibliography

[1] Abiteboul, S.; Agrawal, R.; Bernestein, P.; Carey, M.; Ceri, S.; Croft, B.; DeWitt, D.; Franklin, M.; Garcia-Molina, H.; Gawlick, D.; Gray, J.; Haas, L.; Halevy, A.; Hellerstein, J.; Ioannidis, Y.; Kersten, M.; Pazzani, M.; Lesk, M., Maier, D.; Naughton, J.; Schek, H.-J.; Sellis, T.; Silberschatz, A.; Stonebraker, M.; Snodgrass, R.; Ullman, J.D.; Weikum, G.; Widom, J.; Zdonik, S. *The Lowell Database Research Self-Assessment*. Communications of the ACM (CACM), 48(5), 111–118, 2005.

[2] Agosti, M.; Albrechtsen, H.; Ferro, N.; Frommholz, I.; Hansen, P.; Orio, N.; Panizzi, E.; Pejtersen, A.M.; Thiel, U. 'DiLAS: a Digital Library Annotation Service'. In: *Proceedings of International Workshop on Annotation for Collaboration – Methods, Tools, and Practices* (IWAC 2005), 91–101, 2005.

[3] Agosti, M.; Berretti, S.; Brettlecker, G.; Del Bimbo, A.; Ferro, N.; Fuhr, N.; Keim, D.A.; Klas, K.-P.; Lidy, T.; Milano, D.; Norrie, M.C.; Ranaldi, P.; Rauber, A.; Schek, H.-J.; Schreck, T.; Schuldt, H.; Signer, B.; Springmann, M. 'DelosDLMS – The Integrated DELOS Digital Library Management System'. In: C. Thanos, F. Borri, L. Candela (eds): Digital Libraries: Research and Development, First International DELOS Conference, Pisa, Italy, 13–14 February 2007, Revised Selected Papers. Volume 4877 of *Lecture Notes in Computer Science*, Springer 2007, ISBN 978-3-540-77087-9.

[4] Agosti, M.; Ferro, N. 'A Formal Model of Annotations of Digital Content'. *ACM Transactions on Information Systems* (TOIS), 26(1), 3:1–3:57, 2007.

[5] Agosti, M.; Ferro, N. 'A System Architecture as a Support to a Flexible Annotation Service', Volume 3664 of *Lecture Notes In Computer Science*, 147–166. Springer-Verlag, Berlin; Heidelberg. Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures: 6th Thematic Workshop of the EU Network of Excellence DELOS. Revised Selected Papers, 2005.

[6] Agosti, M.; Ferro, N. 'Annotations as Context for Searching Documents'. Volume 3507 of *Lecture Notes In Computer Science*, 155–170. Springer-Verlag, Berlin; Heidelberg. Proceedings 5th International Conference on Conceptions of Library and Information Science – Context: nature, impact and role (Colis 5), 2005.

[7] Agosti, M.; Ferro, N.; Frommholz, I.; Thiel, U. 'Annotations in Digital Libraries and Collaboratories: Facets, Models and Usage', Volume 3232 of *Lecture Notes In Computer Science*, 244–255. Springer-Verlag, Berlin; Heidelberg. Research and Advanced Technology for Digital Libraries: *Proceedings 8th European Conference, ECDL 2004*.

[8] Agosti, M.; Ferro, N.; Orio, N. 'Annotating Illuminated Manuscripts: an Effective Tool for Research and Education'. In: *Proceedings of the 5th ACM/IEEE 2005 Joint Conference on Digital Libraries* (JCDL 2005), 121-130, 2005.

[9] Alexandria Digital Library http://www.alexandria.ucsb.edu/

[10] Alves, M.; Viegas Damásio, C.; Olmedilla, D.; Nejdl, W.A. 'Distributed tabling algorithm for rule based policy systems'. In: 7th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2006), London, Ontario, Canada, June 2006. IEEE Computer Society.

[11] Andrews, J.; Law, D., *Digital Libraries: Policy, Planning and Practice*, Ashgate, 2004, ISBN 0 7546 3448 5, 283 pages.

[12] Antoniou, G.; Baldoni, M.; Bonatti, P.; Nejdl, W.; Olmedilla, D. 'Rule-based policy specification'. In: Ting Yu and Sushil Jajodia (eds), *Secure Data Management in Decentralized Systems*. Springer, 2007.

[13] ARTISTE: An Integrated Art Analysis and Navigation Environment http://www.ecs.soton.ac.uk/~km/projs/artiste/

[14] Avancini, H.; Candela, L.; Straccia, U. 'Recommenders in a Personalized, Collaborative Digital Library Environment'. *Journal of Intelligent Information Systems*, 28(3), 253–283, 2007.

[15] Ackerman, M.S. 'Providing Social Interaction in the Digital Library'. In: *Proceedings of the First Annual Conference on the Theory and Practice of Digital Libraries*, 1994.

[16] Arkin, A.; Askary, S.; Bloch, B.; Curbera, F.; Goland, Y.; Kartha, N.; Liu, C.K.; Mehta, V.; Thatte, S.; Yendluri, P.; Yiu, A.; Alves, A. *Web Services Business Process Execution Language Version 2.0*. OASIS, 2005.

[17] Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P.F. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002. ISBN 0521781760.

[18] Banwell, L.; Ray, K.; Coulson, G.; Urquhart, C.; Lonsdale, R.; Armstrong, C.; Thomas, R.; Spink, S.; Yeoman, A.; Fenton, R.; Rowley, J. (2004). 'The JISC User Behaviour Monitoring and Evaluation Framework'. *Journal of Documentation*, 60(3), 302–320.

[19] Barlas, C.; Cunard, J.; Hill, K. *Current Developments in the Field of Digital Rights Management.* World Intellectual Property Rights Organization, Standing Committee on Copyright and Related Rights, 2003. http://www.wipo.int/documents/en/meetings/2003/sccr/pdf/sccr_10_2.pdf

[20] Batini, C.; Scannapieco, M. *Data Quality*. Springer-Verlag, Berlin; Heidelberg, Germany, 2006.

[21] Beall, J. 'Metadata and Data Quality Problems in the Digital Library'. *Journal of Digital Information*, 6(3), Article No. 355, 12 June 2005.

[22] Becker, M.; Sewell, P. 'Cassandra: Distributed access control policies with tunable expressiveness'. In: *Proceedings of 5th IEEE International Workshop on Policies for Distributed Systems and Networks* (POLICY 2004), 159–168, Yorktown Heights, NY, USA, June 2004. IEEE Computer Society.

[23] Belkin, N.J. 'Understanding and Supporting Multiple Information Seeking Behaviors in a Single Interface Framework'. In: *Proceedings of the Eighth DELOS Workshop: User Interfaces in Digital Libraries*, 11–18. European Research Consortium for Informatics and Mathematics, 1999.

[24] Belkin, N.J.; Oddy, R.N.; Brooks, H.M. 'ASK for Information Retrieval: part 1. Background and Theory'. *Journal of Documentation*, 38(2), 61–71, 1982.

[25] Besek, J.M. *Copyright issues relevant to the creation of a digital archive: A preliminary assessment*. Council on Library and Information Resources, Library of Congress. (January 2003) http://www.clir.org/pubs/reports/pub112/pub112.pdf

[26] Bhagwat, D.; Chiticariu, L.; Tan, W.-C.; Vijayvargiya, G. 'An Annotation Management System for Relational Databases'. In M.A. Nascimento, M.T. Özsu, D. Kossmann, R.J. Miller, J.A. Blakeley and K.B. Schiefer (eds), *Proceedings 30th International Conference on Very Large Data Bases* (VLDB 2004), 900–911. Morgan Kaufmann, 2004.

[27] Bonatti P.A.; Shahmehri, N.; Duma, C.; Olmedilla, D.; Nejdl, W.; Baldoni, M.; Baroglio, C.; Martelli, A.; Patti, V.; Coraggio, P.; Antoniou, G.; Peer, J.; Fuchs, N. *Rule-based policy specification: State of the art and future work.* Technical report, Working Group I2, EU NoE REWERSE, August 2004. http://rewerse.net/deliverables/i2-d1.pdf

[28] Bonatti, P.; Olmedilla, D.; Peer, J. 'Advanced policy explanations'. In: *Proceedings of 17th European Conference on Artificial Intelligence (ECAI 2006)*, Riva del Garda, Italy, August 2006. IOS Press.

[29] Bonatti, P.A., De Capitani di Vimercati, S.; Samarati, P. 'An algebra for composing access control policies'. *ACM Transactions on Information System Security*, 5(1), 1-35, 2002.

[30] Bonatti, P.A.; Duma, C.; Fuchs, N.; Nejdl, W.; Olmedilla, D.; Peer, J.; Shahmehri, N. 'Semantic web policies – a discussion of requirements and research issues'. In: 3rd European Semantic Web Conference (ESWC), Volume 4011 of *Lecture Notes in Computer Science*, Budva, Montenegro, June 2006. Springer.

[31] Bonatti, P.A.; Olmedilla, D. *Policy language specification*. Technical report, Working Group I2, EU NoE REWERSE, February 2005. http://rewerse.net/deliverables/m12/i2-d2.pdf

[32] Bondi, A.B. 'Characteristics of Scalability and Their Impact on Performance'. In: *Proceedings of the 2nd International Workshop on Software and Performance* (WOSP '00), 195–2003, ACM Press, New York, USA, 2000.

[33] Booch, G.; Rumbaugh, J.; Jacobson, I. *The Unified Modeling Language User Guide*. Addison Wesley Longman. ISBN 0321267974, 2005.

[34] Booth, D.; Haas, H.; McCabe, F.; Newcomer, E.; Champion, M.; Ferris, C.; Orchard, D. *Web Service Architecture*. W3C Working Group Note, February 2004. http://www.w3.org/TR/ws-arch/ (last visited 21 February 2007)

[35] Borbinha, J.; Kunze, J.; Spinazzé, A.; Mutschke, P.; Lieder, H.-J.; Mabe, M.; Dixson, L.; Besser, H.; Dean, B.; Cathro, W. 'Reference models for digital libraries: actors and roles'. *International Journal of Digital Libraries*, 5(4), 325-330, August 2005.

[36] Borgman, C.L. 'What are digital libraries? Competing visions'. *Information Processing and Management*, 35(3), 227-243, 1999.

[37] Bottoni, P.; Civica, R.; Levialdi, S.; Orso, L.; Panizzi, E.; Trinchese, R. 'MADCOW: a Multimedia Digital Annotation System'. In: *Proceedings of Working Conference on Advanced Visual Interfaces (AVI 2004)*, 55–62, 2004.

[38] Börner, K. 'Extracting and Visualizing Semantic Structures in Retrieval Results for Browsing'. In: *Proceedings of the Fifth ACM Conference on Digital Libraries*, San Antonio, Texas, United States, 2000.

[39] Börner, K.; Chen, C. 'Visual Interfaces to Digital Libraries'. *Lecture Notes in Computer Science*, Springer, 2003.

[40] Bradner, S. *Key words for use in RFCs to Indicate Requirements Levels*. RFC 2119, IETF, 1997.

[41] BRICKS Integrated Project: Building Resources for Integrated Cultural Knowledge Services. http://www.brickscommunity.org (last visited 21 February 2007)

[42] Bruce, T.R.; Hillman, D.I. 'The Continuum of Metadata Quality', In: D. Hillman and E.L. Westbrooks (eds), *Metadata in Practice*. Chicago: American Library Association (2004).

[43] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerla, P.; Stal, M. *Pattern-Oriented Software Architecture.* John Wiley & Sons. ISBN 0-471-95869-7, 1996.

[44] Callan, J.P.; Lu, Z.; Croft, W.B. 'Searching Distributed Collections with Inference Networks'. In: E.A. Fox, P. Ingwersen and R. Fidel (eds), *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 21–28, Seattle, Washington, 1995. ACM Press.

[45] Candela, L.; Castelli, D.; Fuhr, N.; Ioannidis, Y.; Klas, C.-P.; Pagano, P.; Ross, S.; Saidis, C.; Schek, H.-J.; Schuldt, H.; Springmann, M. *Current Digital Library Systems: User Requirements vs Provided Functionality*. DELOS Deliverable D1.4.1, March 2006.

[46] Candela, L.; Castelli, D.; Ioannidis, Y.; Koutrika, G.; Pagano, P.; Ross, S.; Schek, H.-J.; Schuldt, H. *The Digital Library Manifesto*. DELOS, 2006. ISBN 2-912335-24-8.

[47] Candela, L.; Castelli, D.; Pagano, P.; Simi, M. 'OpenDLibG: Extending OpenDLib by exploiting a gLite Grid Infrastructure'. In: Research and Advanced Technology for Digital Libraries, 10th European Conference on Digital Libraries (ECDL 2006), September 2006.

[48] Candela, L.; Castelli, D.; Pagano, P. *Digital Library Reference Model*. Project Report 2006-PR-02, Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR, January 2006.

[49] Candela, L.; Castelli, D.; Pagano, P.; Simi, M. 'From Heterogeneous Information Spaces to Virtual Documents'. In: Digital Libraries: Implementing Strategies and Sharing Experiences, 8th International Conference on Asian Digital Libraries (ICADL 2005), December 2005.

[50] Candela, L.; Castelli, D.; Pagano, P. 'A Service for Supporting Virtual Views of Large Heterogeneous Digital Libraries'. In: Research and Advanced Technology for Digital Libraries, 7th European Conference on Digital Libraries (ECDL 2003), August 2003.

[51] Caplan, P. 'PREMIS – Preservation Metadata Implementation Strategies Update 1: Implementing preservation repositories for digital materials: current practice and emerging trends in the cultural heritage community'. *RLG DigiNews,* 8(5), October 2004. http://www.rlg.org/en/page.php?Page_ID=20462#article2

[52] Castelli, D.; Pagano, P. 'A System for Building Expandable Digital Libraries'. In: *Proceedings of ACM/IEEE 2003 Joint Conference on Digital Libraries* (JCDL 2003), 335–345, 2003

[53] CCSDS 650.0-B-1: Reference Model for an Open Archival Information System (OAIS). Blue Book. Issue 1. January 2002. This Recommendation has been adopted as ISO 14721:2003 http://ssdoo.gsfc.nasa.gov/nost/isoas/ref_model.html (last visited 21 February 2007)

[54] Chen, C.; Paul, R.J. 'Visualizing a Knowledge Domain's Intellectual Structure'. *IEEE Computer*, 34(3), 65–71, 2001.

[55] Chen, Peter P. 'The Entity-Relationship Model – Toward a Unified View of Data'. *ACM Transactions on Database Systems*, 1(1), 9–3, 1976.

[56] Chinnici, R.; Gudgin, M.; Moreau, J-J.; Schlimmer, J.; Weerawarana, S. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C Working Draft http://www.w3.org/TR/wsdl20 (last visited 21 February 2007)

[57] Clark, J.A. 'A Usability Study of the Belgian American Research Collection: Measuring the Functionality of a Digital Library'. *OCLC Systems and Services: International Digital Library Perspectives*, 20(3), 115–127, 2004.

[58] Clement, L.; Hately, A.; von Riegen, C.; Rogers, T. *UDDI Version 3.0.2*. UDDI Spec Technical Committee Draft, OASIS. http://uddi.org/pubs/uddi_v3.htm (last visited 21 February 2007)

[59] Cohen, J.E. 'The Law and Technology of Digital Rights Management: DRM and Privacy'. *Berkeley Technology Law Journal*, 18 (Spring 2003), 575–617.

[60] COLLATE Collaboratory for Annotation, Indexing and Retrieval of Digitized Historical Archive Material. http://www.collate.de

[61] Constantopoulos, P.; Doerr, M.; Theodoridou, M.; Tzobanakis, M. 'On Information Organization in Annotation Systems', Volume 3359 of *Lecture Notes In Computer Science*, 189–200. Springer-Verlag, Berlin; Heidelberg. International Workshop on Intuitive Human Interfaces for Organizing and Accessing Intellectual Assets, 2004.

[62] Coyle, K. *Rights Expression Languages: A Report for the Library of Congress.* February 2004. http://www.loc.gov/standards/Coylereport_final1single.pdf

[63] Creative Commons, 2004. http://www.creativecommons.org/

[64] D'Agostino, G. Copyright Treatment of Freelance Work in the Digital Era. *Santa Clara Computer and High Technology Law Journal*, 19 (December 2002), 37–110.
[65] Del Bimbo, A.; Gradmann, S.; Ioannidis, Y. (eds), *Future Research Directions*. 3rd DELOS Brainstorming Workshop Report, Corvara, Italy, July 2004.
[66] DELOS Network of Excellence on Digital Libraries. http://www.delos.info/ (last visited 21 February 2007)
[67] Deutsch, L.P. 'Design Reuse and Frameworks in the Smalltalk-80 System'. In: *Software Reusability*, 57–71, ACM Press, 1989.
[68] DILIGENT Integrated Project: Digital Library Infrastructure on Grid Enabled Technology. http://www.diligentproject.org (last visited 21 February 2007)
[69] Dowty, D.R.; Wall, R.; Peters, S. Introduction to Montague Semantics Series: Studies in Linguistics and Philosophy, Vol. 11, Springer-Verlag, 1980.
[70] Duane, N. *Service Oriented Architecture*. Adobe Systems, Inc. White Paper, 2005.
[71] Dublin Core Metadata Element Set, Version 1.1: Reference Description. http://dublincore.org/documents/dces/ (last visited 21 February 2007)
[72] Duranti, L. 'The long-term preservation of accurate and authentic digital data: the INTERPARES project'. *Data Science Journal*, 4, 106–118, 2005.
[73] Duranti, L. 'The Long-Term Preservation of Authentic Electronic Records'. In: *Proceedings of 27th International Conference on Very Large Data Bases* (VLDB 2001), 625–628, 2001.
[74] Ellis, D.; Haugan, M. 'Modeling the Information Seeking Patterns of Engineers and Research Scientists in an Industrial Environment'. *Journal of Documentation*, 53(4), 384–403, 1997.
[75] Endrei, M.; Ang, J.; Arsanjani, A.; Chua, S.; Comte, P.; Pal, K.; Min, L.; Newling, T. *Patterns: Service-Oriented Architecture and Web Services*. IBM Redbook, April 2004.
[76] ERPANET, 2005. *European Investment Bank Case Study Report* 2004. http://www.erpanet.org/studies/index.php
[77] Experts' Workgroup on the Preservation of Digital Memory. Firenze Agenda, 2005. Available from http://www.erpanet.org/events/workgroup/documents/firenze%20agenda.pdf
[78] Faensen, D.; Faultstich, L.; Schweppe, H.; Hinze, A.; Steidinger, A. 'Hermes: a notification service for digital libraries'. In: *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries* (JCDL '01), 2001.
[79] Farrell, S; Housley, R. *An Internet Attribute Certificate Profile for Authorization*. RFC 3281. http://www.faqs.org/rfcs/rfc3281.html (last visited 21 February 2007)
[80] Ferraiolo, D.; Kuhn, D.R.; Chandramouli, R. *Role-based Access Control*. Artech House, 2003.
[81] Ferrara, R.; Gentili, D.; Ponzani, V. (2006) 'How "free" we are: survey of the Italian policy in the document supply service'. In: *Proceedings EAHIL X European Conference*, Cluj (Romania), 11–16 September 2006. http://www.eahilconfcluj.ro/docs/4b/FERRARA_L.doc
[82] Foster, A.; Ford, N. Serendipity and Information Seeking: an Empirical Study. *Journal of Documentation*, 59(3), 321–340, 2003.
[83] Fox, E.A.; Akscyn, R.M.; Furuta, R.; Leggett, J.J. *Digital Libraries*. Communications of the ACM, 38(4), 23–28, April 1995.
[84] Fox, E.A.; Marchionini, G. *Toward a Worldwide Digital Library*. Communications of the ACM, 41(4), 29–32, April 1998.

[85] Frommholz, I.; Brocks, H.; Thiel, U.; Neuhold, E.; Iannone, L.; Semeraro, G.; Berardi, M.; Ceci, M. 'Document-Centered Collaboration for Scholars in the Humanities – The COLLATE System', Volume 2769 of *Lecture Notes in Computer Science*, 434–445. Springer-Verlag, Berlin; Heidelberg. Research and Advanced Technology for Digital Libraries, *Proceedings 7th European Conference, ECDL 2003*.

[86] Fuhr, N.; Hansen, P.; Mabe, M.; Micsik, A.; Solvberg, I. (2001). '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*, 187-199, London, UK.

[87] Fuhr, N.; Tsakonas, G.; Aalberg, T.; Agosti, M.; Hansen, P.; Kapidakis, S.; Klas, C.-P.; Kovács, L.; Landoni, M.; Micsik, A.; Papatheodorou, C.; Peters, C.; Solvberg, I. (2006). 'Evaluation of Digital Libraries'. *International Journal of Digital Libraries*, 2007 (online first).

[88] Gachet, A. '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.

[89] Garlan, D; Shaw, M. *An Introduction to Software Architecture*. SEI Technical Report CMU/SEI-94-TR-21.

[90] Gavriloaie, R.; Nejdl, W.; Olmedilla, D.; Seamons, K.; Winslett, M. 'No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web'. In: 1st European Semantic Web Symposium (ESWS 2004), Volume 3053 of *Lecture Notes in Computer Science*, 342-356, Heraklion, Crete, Greece, May 2004. Springer.

[91] gCube: grid for computing, grid for content, grid for services. http://www.gcube-system.org/

[92] Ghezzi, C.; Jazayeri, M.; Mandrioli, D. *Fundamentals of Software Engineering* (2nd edn). Upper Saddle River, NJ, USA, Prentice Hall, 2003.

[93] Ginsburg, J.C. 'Copyright and Control Over New Technologies of Dissemination'. *Columbia Law Review*, 101 (November 2001), 1613 –1647.

[94] Gladney, H.M. *Principles for Digital Preservation*. Communication of the ACM 49, (2), 111 –116, February 2006.

[95] Gonçalves, M.A. *Streams, Structures, Spaces, Scenarios, and Societies (5S): A Formal Model for Digital Library Framework and Its Applications*. PhD thesis, Virginia Polytechnic Institute and State University, November 2004.

[96] Gonçalves, M.A.; Fox, E.A. '5SL – A Language for Declarative Specification and Generation of Digital Libraries'. In: *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries* (JCDL 02), Portland, Oregon, USA, 2002.

[97] Gonçalves, M.A.; Fox, E.A.; Watson, L.T.; Kipp, N.A. 'Streams, Structures, Spaces, Scenarios, Societies (5S): A Formal Model for Digital Libraries'. *ACM Transactions on Information Systems* (TOIS), 22(2), 270 –312, 2004.

[98] Gonçalves, M.A.; Moreira, B.L.; Fox, E.A.; Watson, L.T. 'What is a good digital library? – A quality model for digital libraries'. *Information Processing & Management*, 43(5), 1416–1437, 2007.

[99] Griffin, S.; Peters, C.; Thanos, C. 'Towards the new-generation digital libraries: recommendations of the NSF/EU-DELOS working groups'. Guest editor introduction. *International Journal of Digital Libraries*, 5(4), 253 –254, August 2005.

[100] Groth, D.P.; Streefkerk, K. 'Provenance and Annotation for Visual Exploration Systems'. *IEEE Transactions on Visualization and Computer Graphics*, 12(6), 1500–1510, November/December 2006.

[101] Guarino, N. 'Formal Ontology and Information Systems', *Proceedings of the 1st International Conference on Formal Ontology in Information Systems* (FOIS'98), 3–15, Trento, June 1998.
[102] Gudgin, M.; Hadley, M.; Mendelsohn, N.; Moreau, J-J.; Nielsen, H. *SOAP Version 1.2 Part 0: Primer*. W3C Recommendation. http://www.w3.org/TR/soap12-part0/ (last visited 21 February 2007)
[103] Gudgin, M.; Hadley, M.; Mendelsohn, N.; Moreau, J-J.; Nielsen, H. *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation. http://www.w3.org/TR/soap12-part1/ (last visited 21 February 2007)
[104] Guenther, R. 'PREMIS – Preservation Metadata Implementation Strategies Update 2: Core Elements for Metadata to Support Digital Preservation'. *RLG DigiNews,* 8(6), 15 December 2004. Retrieved 22 December 2004. http://www.rlg.org/en/page.php?Page_ID=20492#article2
[105] Guercio, M.; Lograno L.; Battistelli, A. *Legislation, Rules and Policies for the Preservation of Digital Resources, A Survey*, 45 pages, 2003. http://www.erpanet.org/events/workgroup/documents/Regulations_Policy%20_Dossier_English%20version.pdf
[106] Guy, M.; Powell, A.; Day, M., 'Improving the quality of metadata in Eprint archives', *Ariadne*, Issue 38, 2004. http://www.ariadne.ac.uk/issue38/guy/
[107] Hallam-Baker, P.M.; Behlendorf, B. *Extended Log File Format* – W3CWorking Draft WD-logfile-960323, March 1996. http://www.w3.org/TR/WD-logfile.html
[108] Hartson, R.H.; Shivakumar, P.; Perez-Quinones, M.A. 'Usability Inspection of Digital libraries: a Case Study'. *International Journal on Digital Libraries*, 4(2), 108–123, 2004.
[109] Harvey, R. *Preserving Digital Materials.* Munich: K.G. Saur, 246 pages (2005).
[110] Heery, R.; Patel, M. 'Application profiles: mixing and matching metadata schemas'. *Ariadne*, Issue 25, September 2000.
[111] Hill, M.D. 'What is scalability?' ACM SIGARCH *Computer Architecture News*, 18(4), 18–21, ACM Press, New York, USA, 1990.
[112] Hodge, G.; Frangakis, E. *Digital Preservation and Permanent Access to Scientific Information; The State of the Practice*: A report sponsored by The International Council for Scientific and Technical Information and CENDI, 2004. http://www.cendi.gov/publications/04-3dig_preserv.pdf
[113] Huang, Z.; Chung, W.; Ong, T.-H.; Chen, H. 'A graph-based recommender system for digital library'. In: *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries* (JCDL '02), 2002.
[114] *IEEE Recommended Practice for Architectural Description*, IEEE Std P1471, 2000.
[115] *IEEE recommended practice for software requirements specifications*, IEEE Std 830-1998, 20 October 1998.
[116] IFLA Study Group on the Functional Requirements for Bibliographic Records. *Functional Requirements for Bibliographic Records: Final Report*. September 1997. http://www.ifla.org/VII/s13/frbr/frbr.htm (last visited 21 February 2007)
[117] Ingersoll, P.; Culshaw, J. *Managing Information Technology*. Libraries Unlimited, 2004.
[118] *InterPARES Strategy Task Force Report: An Intellectual Framework for Policies, Strategies, and Standards*. In: The Long-Term Preservation of Authentic Electronic Records: Findings of the InterPARES Project, 2005. http://www.interpares.org/book/index.htm
[119] Ioannidis, Y. (ed.) *Digital Libraries: Future Directions for a European Research Programme*. DELOS Brainstorming Report, San Cassiano, Italy, June 2001.

[120] Ioannidis, Y. 'Digital libraries at a crossroads'. *International Journal of Digital Libraries*, 5(4), 255 –265, August 2005.
[121] Ioannidis, Y.; Maier, D.; Abiteboul, S.; Buneman, P.; Davidson, S.; Fox, E.; Halevy, A.; Knoblock, C.; Rabitti, F.; Schek, H.; Weikum, G. 'Digital library information-technology infrastructures'. *International Journal of Digital Libraries*, 5(4), 266 –274, August 2005.
[122] ISO 21127:2006 Information and documentation – A reference ontology for the interchange of cultural heritage information, December 2006.
[123] Jacobs, I.; Walsh, N. (eds), *Architecture of the World Wide Web*, Volume One. W3C Recommendation. http://www.w3.org/TR/webarch/
[124] Johnson, R.E.; Foote, B. Designing reusable classes. *Journal of object-oriented programming*, 1(2), 22-35, 1988.
[125] Jones, S. 'Graphical Query Specification and Dynamic Result Previews for a Digital Library'. In: *Proceedings of 11th Annual ACM Symposium on User Interface Software and Technology*, San Francisco, California, United States, 1998.
[126] Jones, S.; Cunningham, S.J.; McNab, R.; Boddie, S. 'A Transaction Log Analysis of a Digital Library'. *International Journal on Digital Libraries*, 3(2), 152–169.
[127] Kagal, L.; Finin, T.; Joshi, A. 'A policy based approach to security for the semantic web'. In: 2nd International Semantic Web Conference (ISWC), Vol. 2870 of *Lecture Notes in Computer Science*, 402–418, Sanibel Island, FL, USA, October 2003. Springer.
[128] Kagal, L.; Finin, T.; Joshi, A. 'A policy language for a pervasive computing environment'. In: *Proceedings of 4th IEEE International Workshop on Policies for Distributed Systems and Networks* (POLICY), 63, Lake Como, Italy, June 2003. IEEE Computer Society.
[129] Kagal, L.; Paolucci, M.; Srinivasan, N.; Denker, G.; Finin, T.W.; Sycara, K.P. 'Authorization and privacy for semantic web services'. *IEEE Intelligent Systems*, 19(4), 50–56, 2004.
[130] Kahan, J.; Koivunen, M.-R. 'Annotea: an open RDF infrastructure for shared Web annotations'. In: *Proceedings of the 10th International Conference on World Wide Web* (WWW 2001), 623–632, 2001.
[131] Kahle, B.; Jackson, M.; Prelinger, R. 'Public access to digital materials'. *D-Lib Magazine*, 7(10), 2001.
[132] Kaushik, S.; Ammann, P.; Wijesekera, D.; Winsborough, W.; Ritchey, R. 'A policy driven approach to email services'. In: *Proceedings of 5th IEEE International Workshop on Policies for Distributed Systems and Networks* (POLICY), 169, Yorktown Heights, NY, USA, June 2004. IEEE Computer Society.
[133] Kaushik, S.; Winsborough, W.; Wijesekera, D.; Ammann, P. 'Email feedback: a policy-based approach to overcoming false positives'. In: *Proceedings of the 2005 ACM Workshop on Formal Methods in Security Engineering*, Fairfax, VA, USA, 11 November 2005. FMSE '05. ACM, New York, NY, 73–82, 2005.
[134] Ke, H.-R.; Kwakkelaar, R.; Tai, Y.-M.; Chen, L.-C. 'Exploring Behavior of E-journal Users in Science and Technology: Transaction Log Analysis of Elsevier's ScienceDirect OnSite in Taiwan'. *Library and Information Science Research*, 24, 265–291, 2002.
[135] Kelly, B.; Guy, M.; James, H. 'Developing A Quality Culture For Digital Library Programmes'. *Proceedings of the EUNIS 2003 Conference*, Amsterdam, Holland, 2003.
[136] Kengeri, R.; Fox, E.A.; Reddy, H.P.; Harley, H.D.; Seals, C.D. 'Usability Study of Digital Libraries', ACM, IEEE-CS, NCSTRL, NDLTD. *International Journal on Digital Libraries*, 2(2), 157–169, 1999.

[137] Kerry, A. *Digital Preservation: The Research and Development Agenda in Australia.* 21 pages, 2001. http://www.swinburne.edu.au/lib/digcontforum/DigitalContinuity_AKerry.pdf

[138] Kolari, P.; Ding, L.; Ganjugunte, S.; Joshi, A.; Finin, T.; Kagal, L. 'Enhancing web privacy protection through declarative policies'. In: *Proceedings of 6th IEEE International Workshop on Policies for Distributed Systems and Networks* (POLICY), 57–66, Stockholm, Sweden, June 2005. IEEE Computer Society.

[139] Kuhlthau, C.C. (1991). 'Inside the Search Process: Information Seeking from the User's Perspective'. *Journal of the American Society for Information Science*, 42(5), 361–371.

[140] Kuny, T.; Cleveland, G. 'The Digital Library: Myths and Challenges'. In: *Proceedings 62nd IFLA General Conference*, August 1996.

[141] Kyrillidou, M.; Giersch, S. 'Developing the DigiQUAL Protocol for Digital Library Evaluation'. *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, Denver, CO, USA, 172–173, 2005.

[142] Lagoze, C.; Payette, S.; Shin, E.; Wilper, C. 'Fedora: an architecture for complex objects and their relationships'. *International Journal of Digital Libraries*, 6, 2006.

[143] Lagoze, C.; Van de Sompel, H. 'The open archives initiative: building a low-barrier interoperability framework'. In: *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, Roanoke, Virginia, United States, JCDL '01. ACM, New York, NY, 54–62, 2001. http://doi.acm.org/10.1145/379437.379449

[144] Lagoze, C.; Van de Sompel, H.; Nelson, M.; Warner, S. *The Open Archives Initiative Protocol for Metadata Harvesting*. http://www.openarchives.org/OAI/openarchivesprotocol.html (last visited 21 February 2007)

[145] Lannom, L. 'Handle System Overview'. In: *Proceedings of the 66th IFLA Council and General Conference*, Jerusalem, Israel, August 2000.

[146] Lavoie, B. *The Incentives to Preserve Digital Materials: roles, scenarios and economic decision-making*, 2003. http://www.oclc.org/research/projects/digipres/incentives-dp.pdf

[147] Lavoie, B.; Dempsey, L. 'Thirteen ways of looking at digital preservation'. *DLib Magazine*, 10(7/8), 2004. http://www.dlib.org/dlib/july04/lavoie/07lavoie.html

[148] Lavoie, B.; Henry, G.; Dempsey, L. 'A Service Framework for Libraries'. *D-Lib Magazine*, 11(7/8), July/August 2006.

[149] Lesk, M. 'Expanding Digital Library Research: Media, Genre, Place and Subjects'. In: *Proceedings of the International Symposium on Digital Libraries 1999*, ISDL'99, 51–57, Tsukuba, Ibaraki, Japan, September 1999.

[150] Library of Congress. *MARC Standards*. http://www.loc.gov/marc/ (last visited 21 February 2007)

[151] Library of Congress. *METS Metadata Encoding & Transmission Standard*. http://www.loc.gov/standards/mets/ (last visited 21 February 2007)

[152] Lomow, G.; Newcomer, E. *Understanding SOA with Web Services*. Addison Wesley Professional, 2005.

[153] Lord, P.; Macdonald, A. *e-Science Curation Report. Data curation for e-science in the UK: an audit to establish requirements for future curation and provision.* The JISC Committee for the Support of Research (JCSR), 2003. http://www.jisc.ac.uk/uploaded_documents/e-ScienceReportFinal.pdf

[154] Lord, P.; Macdonald, A.; Lyon, L.; Giaretta, D. *From Data Deluge to Data Curation*, 2005. http://www.dcc.ac.uk/docs/AHM-150-rev-ll-dg.doc

[155] Lynch, C. Canonicalization: 'A Fundamental Tool to Facilitate Preservation and Management of Digital Information'. *D-Lib Magazine*, 5(9), September 1999.

[156] MacKenzie, M.; Laskey, K.; McCabe, F.; Brown, P.; Metz, R. *Reference Model for Service Oriented Architecture*. OASIS Committee Draft 1.0, February 2006.
[157] Marsh, J. *XML Base*. W3C Recommendation. http://www.w3.org/TR/xmlbase/ (last visited 21 February 2007)
[158] Maximilien, E.M.; Singh, M.P. 'A Framework and Ontology for Dynamic Web Services Selection'. *IEEE Internet Computing*, 8(5), 84–93, September-October 2004.
[159] McClure, C.R.; Bertot, J.C. 'Evaluating Networked Information Services: Techniques, Policy, and Issues'. Medford, NJ. *Information Today*, 2001.
[160] McMillan, G. *The NDLTD, and Issues of Long Term Preservation and Archiving. IT'S ABOUT TIME!* Paper read at Next Steps: Electronic Theses and Dissertations Worldwide ETD 2003, at Humboldt University, Berlin, Germany, 2003. http://edoc.hu-berlin.de/etd2003/mcmillan-gail/PDF/index.pdf
[161] Melnik, S.; Garcia-Molina, H.; Paepcke, A. 'A mediation infrastructure for digital library services'. *Proceedings of the Fifth ACM Conference on Digital Libraries*, 123–132, 2–7 June 2000, San Antonio, Texas, United States.
[162] Menell, P.S. 'Envisioning Copyright Law's Digital Future'. *New York Law School Law Review*, 46, (2002/2003), 63–199.
[163] Metrics for Metadata website. http://ariadne.cti.espol.edu.ec/M4M/
[164] Mullen, S.; Crawford, M.; Lorch, M.; Skow, D. *Site Requirements for Grid Authentication, Authorization and Accounting*. GGF Document Series, Document Number GFD-I.032, October 2004.
[165] Navarro, G.; Baeza-Yates, R. 'Proximal Nodes: A Model to Query Document Databases by Content and Structure'. *ACM Transactions on Information Systems* (TOIS), 15(4), 400–435, October 1997.
[166] Nejdl, W.; Olmedilla, D.; Winslett, M.; Zhang, C. 'Ontology-based policy specification and management'. In: *Proceedings of the 2nd European Semantic Web Conference (ESWC)*, Volume 3532 of *Lecture Notes in Computer Science*, 290–302, Heraklion, Crete, Greece, May 2005. Springer.
[167] Novak, J.D.; Gowin, D.B. *Learning How to Learn*. Cambridge University Press, 1984.
[168] Novak, J.D.; Cañas, A.J. *The Theory Underlying Concept Maps and How to Construct Them*, Technical Report IHMC CmapTools 2006-01, Florida Institute for Human and Machine Cognition, 2006.
[169] NSF/DELOS Working Group. *Invest to Save. Report and Recommendations of the NSF/DELOS Working Group on Digital Archiving and Preservation*, 2003. http://eprints.erpanet.org/archive/00000048/01/Digitalarchiving.pdf
[170] Nuseibeh, B.; Easterbrook, S. 'Requirements Engineering: A Roadmap'. In: A. Finkelstein (ed.), *The Future of Software Engineering*, 22nd International Conference on Software Engineering ICSE 2000, Limerick, Ireland, June 2000.
[171] Oberle, D.; Lamparter, S.; Grimm, S.; Vrandecic, D.; Staab, S.; Gangemi, A. 'Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems'. *Journal of Applied Ontology*, 2006.
[172] OCLC/RLG PREMIS Working Group *Implementing Preservation Repositories for Digital Materials: Current Practice and Emerging Trends in the Cultural Heritage Community*. Report by the Joint OCLC/RLG Working Group Preservation Metadata: Implementation Strategies (PREMIS). Dublin, OH.: OCLC Online Computer Library Center, Inc., 2004. http://www.oclc.org/research/projects/pmwg/surveyreport.pdf
[173] Paepcke, A.; Chang, C.K.; Winograd, T.; García-Molina, H. 1998. *Interoperability for digital libraries worldwide*. Communications of the ACM 41, 4 (April 1998), 33–42. http://doi.acm.org/10.1145/273035.273044

[174] Payette, S.D.; Rieger, O.Y. 'Supporting Scholarly Inquiry: Incorporating Users in the Design of the Digital Library'. *Journal of Academic Librarianship* 24(2), 121–129, 1998.
[175] PREMIS Working Group. *Data Dictionary for Preservation metadata*: Final Report of the PREMIS Working Group. May 2005.
[176] Rieger, O. *Preservation in the Age of Large-Scale Digitization*. Washington, DC: Council on Library and Information Resources, 2007. http://www.clir.org/activities/details/lsdi.pdf
[177] Ross, S. 'Digital Preservation, Archival Science and Methodological Foundations for Digital Libraries'. In: *Proceedings ECDL 2007*, Budapest, 2007. http://eprints.erpanet.org/131/01/Keynote_ECDL2007_SROSS.pdf
[178] Ross, S.; Hedstrom, M. 'Preservation research and sustainable digital libraries'. *International Journal of Digital Libraries*, 5(4), 317–324, August 2005.
[179] Rousseau, G.K.; Rogers, W.; Mead, S.E.; Sit, R.A.; Rousseau-Jamieson, B.A. 'Assessing the Usability of On-line Library Systems'. *Behaviour and Information Technology*, 17(5), 274–281, 1998.
[180] Salton, G.; McGill, M.J. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.
[181] Saracevic, T. 'Evaluation of Digital Libraries: an Overview'. In: M. Agosti and N. Fuhr (eds), Notes of the DELOS WP7 Workshop on the Evaluation of Digital Libraries. Padua, Italy, 2004. http://dlib.ionio.gr/wp7/workshop2004_program.html (last visited 21 February 2007)
[182] Sfakakis, M.; Kapidakis, S. 'User Behavior Tendencies on Data Collections in a Digital Library', Volume 2458 of *Lecture Notes In Computer Science*, 550–559. Springer-Verlag, Berlin; Heidelberg. *Proceedings of Research and Advanced Technology for Digital Libraries: 6th European Conference, ECDL 2002*, Rome, Italy, 16–18 September 2002.
[183] Sharp, H.; Finkelstein, A.; Galal, G. 'Stakeholder Identification in the Requirements Engineering Process'. *Proceedings of Workshop on Requirements Engineering Processes* (REP'99) – DEXA'99, Florence, Italy, 1-3 September 1999, 387-391, 1999.
[184] Schek, H.-J.; Schuldt, H. 'DelosDLMS – Infrastructure for the Next Generation of Digital Library Management Systems'. *ERCIM News*, No. 66, July 2006.
[185] Shen, R. *Applying the 5S Framework To Integrating Digital Libraries*. Virginia Polytechnic Institute and State University, PhD Thesis in Computer Science and Applications, Blacksburg, Virginia, USA, 2006.
[186] Sheneiderman, B.; Feldman, D. 'Visualizing Digital Library Search Results with Categorical and Hierarchical Axes'. Fifth ACM International Conference on Digital Libraries. San Antonio, Texas, United States, 2000.
[187] Shoshani, A. 'Storage Resource Managers: Middleware Components for Grid Storage'. *Proceedings of the Nineteenth IEEE Symposium on Mass Storage Systems* (MSS '02), 2002.
[188] Shoshani, A.; Sim, A.; Gu, J. 'Storage Resource Managers'. In: *Grid Resource Management: State of the Art and Future Trends*, Chapter 20, 2003.
[189] Sobel, L.S. 'DRM as an Enabler of Business Models: ISPs as Digital Retailers'. *Berkeley Technology Law Journal*, 18 (Spring 2003), 667-695.
[190] Soergel, D. 'A Framework for Digital Library Research'. *DLib Magazine*, 8(12), December 2002.

[191] Soergel, D.; Christensen-Dalsgaard, B.; Ioannidis, Y.; Schuldt, H. (eds), *Recommendations and Observations for a European Digital Library (EDL)*. 4th DELOS Brainstorming Workshop Report, Juan-les-Pins, France, 5–6 December 2005. ISBN 2-912335-19-1.

[192] Smeaton, A.F.; Callan, J. 'Personalisation and recommender systems in digital libraries'. *International Journal of Digital Libraries*, 5(4), 299–308, August 2005.

[193] Snowdon, D.N.; Churchill, E.F.; Frecon, E. (eds), *Inhabited Information Spaces – Living with your Data*. Springer, London, 2004.

[194] Strodl, S.; Becker, C.; Neumayer, R.; Rauber, A. 'How to Choose a Digital Preservation Strategy: Evaluating a Preservation Planning Procedure'. *Proceedings of the 2007 Conference on Digital Libraries,* Vancouver, BC, Canada, JCDL '07. ACM Press, New York, NY, 29–38 (2007).

[195] Sutcliffe, A.G.; Ennis, M.; Hu, J. 'Evaluating the Effectiveness of Visual User Interfaces for Information Retrieval'. *International Journal of Human-Computer Studies*, 53, 741–763, 2000.

[196] Suzan, K.D. 'Tapping to the Beat of a Digital Drummer: Fine Tuning U.S. Copyright Law for Music Distribution on the Internet'. *Albany Law Review*, 59, 789–829, 1995.

[197] Tansley, R.; Bass, M.; Smith, M. 'DSpace as an Open Archival Information System: Current Status and Future Directions'. In: Research and Advanced Technology for Digital Libraries, 7th European Conference on Digital Libraries (ECDL 2003), August 2003.

[198] Tedd, L.A., Large, A. *Digital libraries: principles and practice in a global environment*. Germany: K.G. Saur, 2005.

[199] The CIDOC Conceptual Reference Model. http://cidoc.ics.forth.gr (last visited 21 February 2007)

[200] The International DOI Foundation. *The DOI Handbook*. Edition 4.2.0, doi:10.1000/186, February 2005.

[201] Thong, J.Y.L.; Hong, W.; Tam, K. 'Understanding User Acceptance of Digital Libraries: What Are the Roles of Interface Characteristics, Organizational Context, and Individual Differences?' *International Journal of Human-Computer Studies*, 57(3), 215–242, 2002.

[202] Tonti, G.; Bradshaw, J.; Jeffers, R.; Montanar, R.; Suri, N.; Uszok A. 'Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder'. In: 2nd International Semantic Web Conference (ISWC), Volume 2870 of *Lecture Notes in Computer Science*, 419–437, Sanibel Island, FL, USA, October 2003. Springer.

[203] Traw, J. *Library Web Site Policies (Clip Notes)*, American Library Association (May 2000). 98 pages, ISBN-10: 0838980880, ISBN-13: 978-0838980880.

[204] Tsakonas, G.; Kapidakis, S.; Papatheodorou, C. 'Evaluation of User Interaction in Digital Libraries'. In: Agosti, M.; Fuhr, N. (eds): *Notes of the DELOS WP7 Workshop on the Evaluation of Digital Libraries*, Padua, Italy, 2004. http://dlib.ionio.gr/wp7/workshop2004_program.html (last visited 21 February 2007)

[205] Uszok, A.; Bradshaw, J.; Jeffers, R. Kaos: 'A policy and domain services framework for grid computing and semantic web services'. In *iTrust*, Vol. 2995, *Lecture Notes in Computer Science*, 16–26, Oxford, UK, April 2004. Springer.

[206] Van de Sompel, H.; Lagoze, C.; Bekaert, J.; Liu, X.; Payette, S.; Warner, S. 'An Interoperable Fabric for Scholarly Value Chains'. *D-Lib Magazine*, 12(10), October 2006. doi:10.1045/october2006-vandesompel

[207] Waller R. (ed.) 'DELOS Clusters to Contribute to a Joint Prototype'. *DELOS Newsletter #5*. http://www.delos.info/index.php?option=com_content&task=view&id=356&Itemid=133 (last visited 21 February 2007)
[208] Walter, V.A. 'Becoming digital: Policy implications for library and youth services'. *Library Trends,* 45(4), 585–601, 1997.
[209] Weiser, P.J. 'The Internet, Innovation, and Intellectual Property Policy'. *Columbia Law Review*, 103, 534–613, 2003.
[210] Wiederhold, G. 'Mediators in the Architecture of Future Information Systems'. *Computer*, 25(3), 38–49, 1992.
[211] Wiederhold, G.; Wegner, P.; Ceri, S. *Toward Megaprogramming*. Communication of the ACM, 38(11), 89–99, November 1992.
[212] Wilson, T.D. Information Behaviour: An Interdisciplinary Perspective. *Information Processing and Management*, 33(4), 551–572, 1997.
[213] Wilson, T.D. 'Exploring Models of Information Behaviour: the Uncertainty Project'. Information Processing and Management, (35), 839–849, 1999.
[214] Witten, I.H.; Bainbridge, D.; Boddie, S.J. *Power to the People: End-user Building of Digital Library Collections*. ACM Press, 94–103, 2001.
[215] Wormell, I. (ed.). *Information quality: Definitions and dimensions.* Los Angeles, CA, USA, Taylor Graham, 1990.
[216] Zachman, A.J. 'A framework for information systems architecture'. *IBM Systems Journal*, 26(3), 1987.