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 0.96

November 2007

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)
This volume is one of a series of texts written in the context of the DELOS Network of Excellence on Digital Libraries (www.delos.info).

© 2007 DELOS Network of Excellence on Digital Libraries All rights reserved.

No part of this volume may be reproduced in any form or by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission from the publisher.
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 Contents ......................................................................................... 4  
Table of Figures ............................................................................................ 6  
About this Volume ......................................................................................... 8  

## PART I The Digital Library Manifesto....................................................... 11

I. Introduction .............................................................................................. 12  
   I.1.1 What is a Manifesto ............................................................................. 12  
   I.1.2 Motivation ......................................................................................... 13  
I.2 The Digital Libraries Universe: A Three-Tier Framework ...................... 16  
I.3 The Digital Libraries Universe: Main Concepts ....................................... 18  
   I.3.1 Content ............................................................................................. 18  
   I.3.2 User .................................................................................................. 18  
   I.3.3 Functionality ..................................................................................... 19  
   I.3.4 Quality ............................................................................................. 19  
   I.3.5 Policy ................................................................................................ 19  
   I.3.6 Architecture ..................................................................................... 19  
I.4 The Digital Library Universe: The Main Roles of Actors .......................... 21  
   I.4.1 DL End-Users .................................................................................. 21  
   I.4.2 DL Designers .................................................................................. 21  
   I.4.3 DL System Administrators ............................................................... 21  
   I.4.4 DL Application Developers ............................................................... 22  
I.5 Reference Frameworks ............................................................................ 23  
I.6 Digital Library Manifesto: Conclusions ................................................. 25  

## PART II The DELOS Digital Library Reference Model in a Nutshell .......... 26

II.1 Introduction ............................................................................................ 27  
   II.1.1 Motivations ..................................................................................... 29  
   II.1.2 Guide to using the Reference Model ............................................... 30  
   II.1.2.1 Concept Maps ............................................................................. 31  
   II.1.2.2 Notational Conventions ............................................................... 31  
II.2 The Constituent Domains ........................................................................ 32  
   II.2.1 DL Resource Domain ..................................................................... 33  
   II.2.2 Content Domain ............................................................................. 34  
   II.2.3 User Domain .................................................................................. 37  
   II.2.4 Functionality Domain ..................................................................... 39  
   II.2.5 Policy Domain ............................................................................... 44  
   II.2.6 Quality Domain ............................................................................. 46  
   II.2.7 Architecture Domain ....................................................................... 48  
II.3 The Interoperability Issue ....................................................................... 52  

November 2007

4/193
II.4 The Preservation Issue .............................................................................................................. 54
II.5 Related Work.......................................................................................................................... 57
II.5.1 The CIDOC Conceptual Reference Model................................................................. 57
II.5.2 Stream, Structures, Spaces, Scenarios, and Societies: The 5S Framework......... 58
II.5.3 The DELOS Classification and Evaluation Scheme ..................................................... 60
II.5.4 DOLCE-based Ontologies for Large Software Systems ............................................. 61
II.6 Reference Model in a Nutshell: Concluding remarks ......................................................... 63

PART III The DELOS Digital Library Reference Model Concepts and Relations .......... 64
III.1 Introduction .......................................................................................................................... 65
III.2 Concepts’ Hierarchy .......................................................................................................... 66
III.3 Reference Model Concepts’ Definitions ........................................................................... 71
III.4 Relations’ Hierarchy .......................................................................................................... 150
III.5 Reference Model Relations’ Definitions ......................................................................... 152
Conclusions .............................................................................................................................. 163
Appendix A. Concept Maps in A4 format .................................................................................. 164
   A.1. DL Resource Domain Concept Map ........................................................................... 164
   A.2. Content Domain Concept Map .................................................................................... 165
   A.3. User Domain Concept Map .......................................................................................... 166
   A.4. Functionality Domain Concept Map .......................................................................... 167
   A.5. Functionality Domain Concept Map: Access Resource Functions ............................. 168
   A.6. Functionality Domain Concept Map: Specializations of the Manage Resource Function ............................................................. 169
   A.7. Functionality Domain Concept Map: General Manage Resource Functions ............ 170
   A.8. Functionality Domain Concept Map: Manage Information Object Functions .......... 171
   A.9. Functionality Domain Concept Map: Manage Actor Functions ................................. 172
   A.10. Functionality Domain Concept Map: Collaborate Functions .................................... 173
   A.11. Functionality Domain Concept Map: Manage DL Functions .................................... 174
   A.12. Functionality Domain Concept Map: Manage & Configure DLS Functions ............ 175
   A.13. Policy Domain Concept Map ....................................................................................... 176
   A.15. Quality Domain Concept Map ..................................................................................... 178
   A.16. Architecture Domain Concept Map ............................................................................. 179
Index of Concepts and Relations .............................................................................................. 180
Bibliography ............................................................................................................................... 182
Table of Figures

Figure I.2-1. DL, DLS, and DLMS: A Three-tier Framework ............................................. 16
Figure I.3-1. The Digital Library Universe: Main Concepts ............................................. 18
Figure I.4-1. The Main Roles of Actors versus the Three-tier Framework ...................... 21
Figure I.4-2. Hierarchy of Users’ Views ........................................................................... 22
Figure I.5-1. The Digital Library Development Framework ............................................. 24
Figure II.1-1. The Digital Library Universe ...................................................................... 27
Figure II.1-2. The Reference Model as the core of the Development Framework .......... 29
Figure II.1-3. A concept map showing the key feature of concept maps ......................... 31
Figure II.2-1. DL Domains Hierarchy Concept Map ....................................................... 32
Figure II.2-2. DL Resource Domain Concept Map ........................................................ 34
Figure II.2-3. Content Domain Concept Map ................................................................. 35
Figure II.2-4. User Domain Concept Map ...................................................................... 38
Figure II.2-5. Functionality Domain Concept Map ........................................................ 40
Figure II.2-6. Functionality Domain Concept Map: Access Resource Functions .......... 40
Figure II.2-7. Functionality Domain Concept Map: Specializations of the Manage Resource Functions ........................................................................................................... 41
Figure II.2-8. Functionality Domain Concept Map: General Manage Resource Functions Applied to all Resources ............................................................ 41
Figure II.2-9. Functionality Domain Concept Map: Manage Information Object Functions . 42
Figure II.2-10. Functionality Domain Concept Map: Manage Actor Functions ............. 42
Figure II.2-11. Functionality Domain Concept Map: Collaborate Functions ................ 43
Figure II.2-12. Functionality Domain Concept Map: Manage DL Functions ................. 43
Figure II.2-13. Functionality Domain Concept Map: Manage DLS Functions ............... 44
Figure II.2-14. Policy Domain Concept Map ................................................................. 45
Figure II.2-15. Policy Domain Concept Map: Policies’ Hierarchy ................................... 46
Figure II.2-16. Quality Domain Concept Map ............................................................... 47
Figure II.2-17. Architecture Domain Concept Map ....................................................... 50
Figure II.5-1. 5S - Map of formal definitions ................................................................ 59
Figure II.5-2. 5S - DL ontology ..................................................................................... 60
Figure A-1. Resource Domain Concept Map (A4 format) .............................................. 164
Figure A-2. Content Domain Concept Map (A4 format) .............................................. 165
Figure A-3. User Domain Concept Map (A4 format) .................................................... 166
Figure A-4. Functionality Domain Concept Map (A4 format) .................................... 167
Figure A-5. Functionality Domain Concept Map: Access Resource Functions (A4 format) .................................................................................................................. 168
Figure A-6. Functionality Domain Concept Map: Specializations of the Manage Resource Function (A4 format) .............................................................. 169
Figure A-7. Functionality Domain Concept Map: General Manage Resource Functions (A 4 format) .............................................................. 170
Figure A-8. Functionality Domain Concept Map: Manage Information Object Functions (A 4 format) .............................................................. 171
Figure A-9. Functionality Domain Concept Map: Manage Actor Functions (A4 format) .... 172
Figure A-10. Functionality Domain Concept Map: Collaborate Functions (A4 format) ...... 173
Figure A-11. Functionality Domain Concept Map: Manage DL Functions (A4 format) ...... 174
Figure A-12. Functionality Domain Concept Map: Manage & Configure DLS Functions (A4 format) ................................ ................................ 175
Figure A-13. Policy Domain Concept Map (A4 format) ........................................... 176
Figure A-14. Policy Domain Concept Map: Policies' Hierarchy (A4 format) .................. 177
Figure A-15. Quality Domain Concept Map (A4 format) ........................................... 178
Figure A-16. Architecture Domain Concept Map (A4 format) .................................... 179
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 of common foundations capable to set the ground for better understanding, communicating, and stimulating further evolutions in this area. The DELOS Digital Library Reference Model aims at contributing to the creation of such foundations. It exploits the collective understanding that has been acquired on Digital Libraries by European research groups active in the Digital Library field throughout many years, both within the DELOS Network of Excellence and outside, as well as by other groups around the world. It identifies the set of concepts and relationships among them characterising the essence of the digital library universe. This model has to be considered as the map allowing various Digital Library involved players to follow the same route and share a common understanding in dealing with the entities of such 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 necessarily is a living framework, so is this document to be considered. Continuous evolutions are envisaged in order to gain 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 on its own. Each of the three parts describes the Digital Library universe with a different perspective which is driven by a diverse trade-off between abstraction and concretisation. Thus each part is equally important to capture 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 one, i.e., they rely on the notions described previously for introducing additional that characterise 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”, specialises these notions by introducing the main concepts and relationships populating each of the aspects captured by the previous one; finally, “PART III The DELOS Digital Library Reference Model Concepts and Relations” precisely describes each of the identified concepts and relations by explaining their rationale as well as by presenting examples of instantiation of them in concrete scenarios.

Even if 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 motivations pervading the whole activity.

Section I.2 presents the relationships among 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 characterizing 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 that actors may play within digital libraries, i.e., end-user, designer, administrator, and application developer.

Section I.5 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.

Section I.6 reports 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 the part.

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

Section II.3 addresses the Interoperability issue. In particular, it describes this problem by pointing out the instruments that the Reference Model makes available for dealing with it.

Section II.4 introduces the Preservation issue and discusses how the current model can be used and eventually extended to capture this important aspect of the digital library universe.

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

Section II.6 reports 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 highlight the role of this part.

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

Section III.3 provides a definition for each of the 217 Concepts currently constituting the model. Each definition is complemented by the list of relations connecting the concept to the other concepts, the rationale for having introduced this concept in the model, and examples of concrete instances of the concept in real 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 having it in the model and some examples of concrete instances of them in real scenarios.

Section Conclusions sums up the volume.

Acknowledgements

Many people have contributed to this activity at different level.

First and foremost, the authors wish to thank the DELOS Network of Excellence on Digital Library 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), co-ordinator of the DELOS of Network of Excellence and to 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 to the DELOS Reference Model Workshop held in Frascati (Rome) on June 2006. The comments, visions and insights received on the initial release of the model have been
very helpful for the rest of the activity. The workshop participants besides the authors were: 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), Norberte 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 titled “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, we thanks all of them collectively and we only mention the especially 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 partaking to some of the Reference Model internal meetings and raising useful comments about the management of multimedia content, and to Prof. Hans Scheck (University of Konstanz), leader of the DELOS work package that gave rise to the Reference Model activity, for the key contribution he has given to the start-up of this activity and for his useful comments.

We are particularly grateful to Maria Bruna Baldacci (ISTI-CNR) for the notably 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 very heterogeneous in scope and yield very different functionality. These systems span 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 “overload” of the term ‘Digital Library’ is a consequence of the fact that there is not yet an agreement on what Digital Libraries are and what functionality is associated with them. This results in lack of interoperability and reuse of both contents 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 major influence on the recent history of the world. The production of manifestos in subsequent centuries in fact increased; ‘The Communist Manifesto’, issued by K. Marx and F. Engels in 1848, and the ‘The Russell-Einstein Manifesto’, issued 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 been several manifestos as well, which have tried to draw the course for the development of particular research areas. These took 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 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 up on mandatory, optional, and even open points where the designer can make several choices.

• ‘The Manifesto for Agile Software Development’, which attempts to discover better ways of developing software by putting emphasis on different items than traditionally, e.g., individuals and interactions instead of processes and tools, working software instead of comprehensive documentation, and others.

• ‘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 want to use it.

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, although 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 in 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 specialized approaches to address particular aspects of Digital-Library functionality. For example, the ARTISTE project [11] 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 [56] from the same Programme focused on how to deal with old film libraries, while the Alexandria Project [7] 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, one may see clearly 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. There is substantial knowledge and experience that have been accumulated. This warrants a process of self-declaration that will identify the principle 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. [77] 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, envisions 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 [110]. 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 [111][112]. Likewise, in his framework for Digital Library research, Soergel [181] 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 further each one and fuses them all together to obtain the main guiding principles for his vision of the field. On the other hand, Belkin [20] states that a Digital Library is an institution in charge of providing at least the functionality of a traditional library in the context of distributed and networked collections of information objects. Lesk [140] analyses and discusses the importance of the terms “Digital” and “Library” in the expression “Digital Library”, where the former term mainly implies 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 are not usually associated with the users’ needs. Borgman [32] 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 [131] 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 digitized books and other text documents. The DELOS Network of Excellence on Digital Libraries [60] now envisions 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 organizes and provides access to particular collections of data and information, to a person-centric system that aims to provide interesting, novel, personalized experiences to users. Its main role has moved 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 synthesizing 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 [184] have developed future visions about “Connected Communities” and “Inhabited Information Spaces”, with the latter being closely related with 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 envisioned, “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). Then, it describes the main concepts characterizing 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 Libraries Universe: A Three-Tier Framework

A Digital Library is an evolving organization that comes to existence through a series of development steps that bring together all 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 conceptualization of the universe of Digital Libraries.

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 difficult the description, understanding and development of digital library systems. 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 organization, which might be virtual, that comprehensively collects, manages, and preserves for the long term rich digital content, and offers to its user communities specialized 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 foundational for Digital Libraries and (ii) to integrate additional software offering more refined, specialized, 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, 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. DLSs 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 [175] 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 easily be incorporated into the Warehouse for subsequent combination with those already there. BRICKS [37] and DILIGENT [62] 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 initialization 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 [89].

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. 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, one may 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” that consists of the above software, its data, and its users.
I.3 The Digital Libraries Universe: Main Concepts

Despite the great variety and diversity of existing digital libraries, in actuality 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 organized 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 acronym “DL”), Digital Library System (DLS) and Digital Library Management System (DLMS) to denote the systems identified in Sec.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 personalize 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 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 characterize 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 automatically measured, 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 organization 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 onto hardware and software components. 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; and (ii) interoperability across Digital Libraries is recognized as a substantial research challenge. A clear architectural framework for the Digital Library System offers ammunition in addressing both these issues effectively.

The concepts populating all the six area 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 them, 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](http://www.sei.cmu.edu/opensystems/glossary.html)
The six core concepts (Content, User, Functionality, Quality, Policy and Architecture) that lie at the heart of Digital Library universe need to be considered in conjunction with the four main ways that 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.

![Diagram](image)

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 authorized 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 partitioned into Content Creators, Content Consumers, and Librarians.

I.4.2 DL Designers

DL Designers exploit their knowledge of the application semantic domain in order to define, customize, 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, they 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, because 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 necessary 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 that the available resources impose on the definition of the DL. DL System Administrators interact with the DLMS by providing architectural configuration parameters, such 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.

These four roles described above encompass the whole spectrum of actors interacting with digital libraries. Their models of the DL Universe are linked together in a hierarchical fashion, as depicted 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 co-operating 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](image)
I.5 Reference Frameworks

The digital library universe is complex. Thus it comprises multiple elements (see Figure I.5-1) which can better be represented in detail with the introduction of frameworks supporting different levels of abstraction:

- **Reference Model** – As stated in [147], “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 DL community, and to help focus further advancement.

- **Reference Architecture** – The Reference Architecture is an architectural design pattern indicating an abstract solution implementing the concepts and relationships identified in the Reference Model. There may be more than one Reference Architecture that addresses how to design digital libraries 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 organizations, and another one for personal DLs or for specialised applications.

- **Concrete Architecture** – At this level, the Reference Architecture is actualised 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 in 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.
This picture was inspired by the “Reference Model for Service Oriented Architecture” document [147].
I.6 Digital Library Manifesto: Conclusions

The rest of this volume focuses on the Reference Model part of this framework. 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 among 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 DL Universe. These documents come as an attempt to fulfil the foundational needs of the DL field. Clearly, diversity of needs among different digital libraries will continue to introduce new concepts that will require incorporation into the model. Hence, these documents should be considered as first versions of dynamic documents that will keep evolving, 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 gone on for 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 named “digital libraries” [32][111][112][131][77][78] (where “system” has to be intended as set of interconnected things forming a whole) there are not yet real underlying foundations for them. This limits the digital library field to grow, as happens for any building to which no appropriate foundation has been provided. Because of this lack it is really difficult or almost impossible to systematize activities for evaluating and comparing digital library systems, teaching and even performing further and focused research. The same holds for system design and development, for promoting sustainable approaches and solutions that aim at maximising the re-use of existing knowledge and assets, and at properly addressing community needs.

In January 2005, the DELOS Networks of Excellence on Digital Libraries [60] 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, a reference model is meant as an abstract framework for understanding significant relationships among the entities of some universe, and for the development of consistent standards and/or specifications supporting that universe [147]. The route to reach this objective, summarised below, has been traced in the Manifesto (Part I of this volume).

It is a common understanding that the digital library universe is a complex and multifaceted domain that cannot be captured by a single definition. The Manifesto organises in a single framework the pieces constituting the puzzle (Figure II.1-1).

In particular, it identifies the three different types of systems playing 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, specialized, or advanced facilities.
The Manifesto also organises the digital library universe in domains.

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

Building on it, 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 that are needed to consider digital library systems from a quality point of view, and
(7) Architecture – represents the physical software (and hardware) constituents concretely realising the whole.

Another contribution of the Manifesto is recognizing the existence of various players acting in the DL universe and cooperating in the operation of the whole. In particular,

• 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 whole framework. These different levels of abstractions, 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 depart from it. It drives the definition of any Reference Architecture that proposes an optimal architectural pattern for a specific class of digital library systems characterized by similar goals, motivations and requirements. Concrete

---

9 It is still under discussion whether two other players should be added to this list, namely Institutions and Industries. The Institutions are meant as organisations, either concrete or virtual ones, having the important role of forming the digital library. The Industries are meant as the institutions performing economic activities concerned with the digital library, by providing either the software or the service.
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 intended as a necessary starting point towards the introduction of all these other framework elements that 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 current section is completed by information setting the stage for the rest, e.g., 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.4 discusses the Preservation issue by presenting the concepts and relations concerning it. Section II.3 presents the Interoperability issue and discusses the main concepts and relations related to it. Section II.5 briefly investigates related work on models for digital libraries and domains. Finally, Section II.6 concludes.

II.1.1 Motivations

The scope of this section is to shortly state what is meant by a reference model and why it is useful in the area of digital libraries.

In the Information Society Technologies (IST) area, a reference model can be considered an abstract framework that can be used to understand significant relationships among the entities of a specific environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and may be used as a basis for education. A reference model is not addressing implementation details, but it provides a common semantics that can be used unambiguously across and between different implementations.

A Reference Model is useful in a specific IST area, because it is abstract in nature and it can be used by system designers as a template for inventing and designing new software systems, and by final users to interact and use a system in an easier and more effective way.
One significant example of reference model can be considered the Open System Interconnection (OSI) reference model which describes how information from a software application in one computer moves through a network medium to a software application in another computer. The model was developed by the International Organization for Standardization (ISO) in 1984, and it is now considered the primary architectural model for internetworking.

In a parallel way, a reference model in the area of Digital Libraries (DL) can be useful for:

- End users of a specific DL since the reference model can help them in more easily identifying appropriate functions and search capabilities;
- Metasearch systems that need to access one or more DL and need to know in what way to interoperate with each DL and what search possibilities each DL offers;
- Administrators who want to match DL functions and reach specific level of performance to satisfy user requirements;
- Investors who want to evaluate a DL;
- Designers and application developers who want to consider design options;
- Students and educators who want to learn and teach about digital libraries and understand the full breadth of their functions.

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 practise 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 great amount of information. Because of the breadth of information to be covered, both models recognise the need to have a way to categorise this information. The best known Enterprise Architecture framework was devised by Zachman [207]. 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. They range from the human language to formal ones borrowed from various application domains and characterised by various expressing power and other interesting features, like Entity-relationship [51], UML [29], and Description Logic [15], to cite a few. However, in this document concept maps have been used because of their simplicity and immediateness.
II.1.2.1 Concept Maps

Concept maps are graphical tools for organizing and representing knowledge [158][159] 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 presents an example of a concept map that describes the structure of concept maps and illustrates their main characteristics.

![Concept Map Diagram]

Figure II.1-3. A concept map showing the key feature of concept maps

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> at their occurrence in the document.

---

10 Figure taken from [159].
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 organization 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 this universe. In this context domains play a role similar to 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 level of abstraction.

![Diagram of DL Domains Hierarchy Concept Map](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 partitioned into two main classes DL Resource Domain and Auxiliary 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 specialised in

1. Content Domain (cf. Sec. II.2.2),
2. User Domain (cf. Sec. II.2.3),
3. Functionality Domain (cf. Sec. II.2.4),
4. Policy Domain (cf. Sec. II.2.5),
5. Quality Domain (cf. Sec. II.2.6), and
6. Architecture Domain (cf. Sec. II.2.7),

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

The Auxiliary Domain contains all the other domains that although they do not constitute the focus of the Digital Libraries and can be inherited from existing models, they are 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 like...
• **Time Domain** (i.e., concepts and relations needed to capture aspects of the time sphere like time periods and intervals);

• **Space Domain** (i.e., concepts and relations needed to capture aspects of the physical sphere like regions and locations).

The rest of this 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 have not to be intended as “complete” or exhaustive, i.e., capable of representing all the possible facets of the systems in the DL Universe, rather as cores of a model of such a Universe that can be extended by specific communities to include the elements that are 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 captures 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 [114]. The Web resource notion has evolved during the Web 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 taken 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 form (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 ones (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>) 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.

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 down leads to an elegant and concise model, to efficient implementations, and to uniform user interfaces. The advantages of this modelling approach can be transformed in 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.

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.

---

11 A Concept linked to a Relation through a dotted line represents an attribute of the Relation itself.
(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 includes also 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. Borne digital information object, such as a borne digital text or a digital camera image.
   b. A digital information object produced by digitization of a non-digital information object.
   c. A non-digital information object which may be represented in the digital library by a metadata record.

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 may acquire a further specialization 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\textsuperscript{[107]} 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 in 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 specialized 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 which 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) a Metadata or an Annotation, but becomes such by virtue of playing a certain role to other information objects. The intuition is grounded 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 is 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 whether annotations are content or metadata is just an ill-posed question.

From an organisational point of view, Information Objects can be grouped in Collections (<\texttt{belongsTo}>), 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. Actually, Collections are characterised by an intension (<\texttt{hasIntension}>) and an extension (<\texttt{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 ones, e.g. [46].
Another specialisation of the Resource Set concept usually associated to 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, 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 another 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 co-operations 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 Authoritativeness (cf. Sec. 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 characterization “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 great 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 does not only serve as a representation of Actor in the system but it 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. Sec. 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).
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 every 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 1.4). 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 one of which usually has a different perspective on the Digital Library. For instance, a Content Creator may be a person that creates and inserts its 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 realized 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 generalized concepts of Actor, capturing not only human users but also inanimate entities, and of Resource, representing all entities involved into or influenced by a Digital Library, and lends a fresh perspective on 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 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.

Each Function is itself a Resource in this model and thus it 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 is 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 realize what is usually called “business process” that is in the service of meeting specific “business requirements” that satisfy a “stakeholder need”. These concepts are borrowed from [139] 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 further elaborated.
Because of the broad scope of the *Function* concept, it is infeasible to enumerate and predict all 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, its intended *Actors*. Therefore, *Function* is specialized into five other concepts that represent still quite general classes of activities, as outlined below (Figure II.2-5).

**Access Resource** captures all activities that are related to requesting, locating, retrieving, transforming, and finally persisting *Resources* (Figure II.2-6). The key characteristic of *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 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.2-5. Functionality Domain Concept Map](image)

**Figure II.2-5. Functionality Domain Concept Map**

**Access Resource** captures all activities that are related to requesting, locating, retrieving, transforming, and finally persisting *Resources* (Figure II.2-6). The key characteristic of *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 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](image)

**Figure II.2-6. Functionality Domain Concept Map: Access Resource Functions**
Manage Resource captures all activities that are 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 maybe specialized to individual Functions for each resource type (Figure II.2-7).

Figure II.2-7. Functionality Domain Concept Map: Specializations 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 specialized 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.).
**Manage Actor** contains *Functions* necessary for the management of individual *Actors* in the DL, including their registration or subscription, their login and the personalization of the *functions* they are entitled to (Figure II.2-10).
The third specialization of Function is intimately 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.

![Collaborate Function](image)

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

The other specializations of the **Function** concept capture all activities that are related to the “systems” as a whole and its management.

**Manage DL** includes a wide variety of **Functions** (Figure II.2-12) that support the day-by-day management of the DL, concerning 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.

![Manage DL Functions](image)

**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** that has to do with 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**.
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 may 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 pretend 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 to the place they belong to.

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 it 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. Sec. 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*).
Policy is actually a class of various types of policies (Figure II.2-15) – the ones which currently are most appropriate in the 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** are 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 the implementation levels across the boundaries of the various systems.

**Policy by scope** are 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 one, and does not intend to form an exhaustive list 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 this model point of view, they are instances of the Policy concept where Digital Rights Management Policy governs Functions, and Digital Rights govern Information Objects.
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 the 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 to consider 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, being it human or not, about 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

---

12 The DLS service is an instance of an Architectural Component (cf. Sec. II.2.7) appropriately configured through (and made available by) the DLMS
measured by a *Measurement*, which represents the value assigned to a *Quality Parameter* with respect to a selected *Measure.*

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. Sec. 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. the ones which currently represent common practice. These parameters are grouped according to the *Resource* under examination (Figure II.2-16).
The DELOS Digital Library Reference Model - 0.96

**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. Sec. 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 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 declares to tender a certain Quality Parameter it is a matter of the underlying Digital Library System to fulfil this claim while it is a matter of the Digital Library Management Systems to provide for the assets needed to guarantee the users expectations, e.g. by implementing the appropriate Architecture.

**II.2.7 Architecture Domain**

The Architecture Domain captures 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 foundational 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 hardly definable. In “An Introduction to Software Architecture” [83], 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 [105], 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, 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 organization or structure of the system’s significant components (Architectural Component) interacting with each other (<use> through their interfaces (Interface). These components may be in turn 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. Sec. 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 in 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. Each Software Component

• encapsulates the implementation of a portion of a software system (capturing Content, User, Functionality, Policy or Quality Domains 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.

Even though the System Architecture of DLS and the System Architecture of 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 DLS from 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, potentially by different players.
- The system may be upgraded with smaller increments, i.e. by upgrading only some of the constituent components. In particular, this aspect is one of the key points to reach interoperability, as upgrading the appropriate constituents of a system makes it able to interact with other systems.
- Components may be shared by systems; this creates opportunities for reuse that heavily contributes 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 generation of DL “systems”.
II.3 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, there arises a scenario 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 this scenario and the resulting issues. In fact, the modelling of interoperability among digital libraries is a really important aspect to be captured because 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 of particular importance for interoperability.

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

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 intended, only on metadata, protocols and few other aspects. As a matter of 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 like, for example, with defining mechanisms that assure that the measures of the content quality parameter Freshness are interoperable with the measures of the same quality parameter in the participant Resources.

By 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 one indicates a higher interoperability degree since it subsumes the former.

Alternative degree of interoperability are often put in place. For instance, in the case of search across multiple data sources provided by diverse organisations (digital libraries), usually three different approaches, characterised by different level of engagement of the sources, are realised: the federated, the harvesting, and the gathering approach. In the federated approach the partaking 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 (a notable example is represented by OAI-PMH [134]), the partaking 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 latter case, i.e., the gathering approach, is the less demanding among the three. In this case no source takes care of its potential consumers because the exposition of its content so that it can easily be used by third parties is not a requirement; i.e., in this case the resources are not required to be interoperable.

The Interoperability issue has many commonalities with preservation and multilinguality. Actually, 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 customary used to discriminate between the aspects to be bridged. In practice, the semantic interoperability is deemed to be more important and to require more sophisticated approaches than syntactic interoperability. However, semantic interoperability cannot be achieved without reaching the syntactic interoperability.

Among the various concepts reported in the Reference Model the following ones are deemed as 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 a Resource is compliant with. 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 be able to either 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 specialization 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 a Resource has originated from. 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 specialization of the Process Function of Manage Information Object. It may include format conversions, information extraction, and automatic translation and summarization techniques. Its specialization **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 specialization 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 specialization of Manage Collection, supports the export of an entire digital library or pieces 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 specialization of the Analyze Function, may be used to ascertain whether two instances of an information object are the same.
II.4 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 this work is based on 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 one would like to know and be able to reproduce the state of a digital library system at a particular point of time in the past. This includes in particular the configuration of Functions. For example, one might want to be able to reproduce the user interface working 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 personalization stored in a 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:

- **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 come 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 Information Objects as well.

**Ontology** with its specialization 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** support the capture of the context an Information Object (in general, any Resource) originated from. This information facilitates the interpretation of an object in case the context provides critical semantic value.

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.
- **Visualize** – 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 possible to drop Resources (Information Object) from a Digital Library system. From a preservation point of view, this function should provide for mechanism for deciding 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 pieces 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 (that can be considered a kind of Metadata) can be used for preservation purposes in different ways, e.g.
  - It allows for rollback operations, e.g. 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** that governs the preservation tasks including selection and appraisal of Resources.
- **Disposal policy** that 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 for what concerns the making of backup copies.

Among the quality parameters, the following ones are of particular importance for preservation:

- **Generic Quality Parameters:**
  - Security Enforcement (cf. Sec. III.3 C157)
  - Interoperability Support (cf. Sec. III.3 C155)
  - Documentation Coverage (cf. Sec. III.3 C159)

- **Content Quality Parameters:**
  - Integrity (cf. Sec. III.3 C166)
  - Authenticity (cf. Sec. III.3 C163)
  - Authoritativeness (cf. Sec. III.3 C164)
  - Performance (cf. Sec. III.3 C160)
  - Fidelity (cf. Sec. III.3 C171)
  - Dependability (cf. Sec. III.3 C183)
  - Provenance (cf. Sec. III.3 C168)

- **Functionality Quality Parameters:**
  - Fault Management Performance (cf. Sec. III.3 C180)

- **Architecture Quality Parameters:**
  - Compliance to standards (cf. Sec. III.3 C195)
II.5 Related Work

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

II.5.1 The CIDOC Conceptual Reference Model

The CIDOC Conceptual Reference Model (CRM) [190] 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 is an official ISO standard (ISO 21127:2006) [113].

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 main classes CIDOC reference model classifies the rest are 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 no further elaborated. 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.

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 represent 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 [107] 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 a 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 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, differently than the Digital Library Reference Model, the CIDOC-CRM only refer collections to physical instantiation of such aggregative mechanism.

---

13 An Information Object is also a Legal Object, i.e. a material or immaterial item to which instances of Right can be applied.
Actor, i.e., people that individually or as a group have the potential to perform actions of which they can be deemed to be responsible, is introduced as specialisation of the Persistent Item 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 model this aspect. While the Digital Library Reference Model introduces the concept of Resource Identifier without taking care to specialise it the CIDOC-CRM introduces many specialisations ranging from Object Identifier to Address, Title, and Date.

Finally, the CIDOC-CRM captures also aspect 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 Functions objects/information have been subjected to. 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.5.2 Stream, Structures, Spaces, Scenarios, and Societies: The 5S Framework

The 5S framework [88][90] is the result of an activity aiming at defining digital libraries in a rigorous manner. It is based on five fundamental abstractions, namely Streams, Structures, Spaces, Scenarios, 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 an information content represented as basic elements, e.g., a simple text is a sequence of characters while a complex object like a book may be a stream of simple text and images. Dynamic streams are used to model any information flow and thus are important for representing any communication that 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 obeying to certain constraints. This kind of construct is powerful and, as suggested by the conveiers, 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 aren’t further investigated 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. These concepts are of general purpose and represent low level constructors. Using these concepts, Gonçalves et al. introduced the whole DL ontology reported in Figure II.5-1.\(^{14}\) In accordance with this framework, they define a 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 catalogs 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 aiming at arranging the concepts and identifying the relationships among them has been proposed. It is depicted in Figure II.5-2.\(^{15}\)

\(^{14}\) Figure II.5-1 is extracted from [88].

\(^{15}\) Figure II.5-2 is extracted from [88].
Despite the commonalities in the goal of the Reference Model and the 5S the proposed approaches presents some differences. The main are:

- The 5Ss are very general purpose constructs and may result to 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 to formalize its aspects while the Reference Model focuses on identifying the main concepts and relationships characterising the whole universe considering the formalisation as a future step;
- Differently than the 5S, the DL 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 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 Software Components and Running Components, is made in terms of scenarios and thus focused on the description of their behaviour. Moreover, service to service co-operating is modelled through the structure concept but no specific instantiations are provided. The Reference Model activity plans to produce specific documents dedicated to these sensible aspects, the Reference Architecture and the Concrete Architecture (cf. Sec. II.1).

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

**II.5.3 The DELOS Classification and Evaluation Scheme**

The DELOS working group dealing with the evaluation of digital libraries problem proposed a model [80][81] broader in scope than that usually adopted in the evaluation context. The aim is to be able to satisfy the needs of all DL researchers, either coming 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 the 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 [195], 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. Sec. II.2.6). While the evaluation framework takes care of identifying the characteristics of the DL systems to be measure 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.5.4 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 commonsense. By relying on the basic constructs identified by it a framework of a set of ontologies for modelling modularization and communication in Large Software Systems has been developed [162].

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 former provides foundations for describing software, in general. In particular it introduces the notions of Software and ComputationalObject that 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 ones envisaged by the Reference Model. In addition to that the CSO ontology introduces concepts borrowed from the object-oriented paradigm like those of Class, Method and Exception and exception that 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 DOLCE ontology the concepts of PolicySubjects (that can be played by Users or UserGroups), PolicyObjects (that can be played by 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. Sec. 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 makes it able 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 to explicit introduction of the QualityOfService parameters that in the case of the Reference Model are captured through the general relationship, i.e., hasQuality, between a Resource and its Quality Parameters.
II.6 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 envision that in the future digital libraries systems will be described, classified and measured according to the key elements introduced by this model.

The presentation has been logically partitioned in 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 graphically represent the concepts and their relations. From the analysis of these maps clearly emerges that despite the complexity of some of the aspects illustrated in most of the cases few powerful aspects are sufficient to capture the essential features.

This DELOS Reference Model in a Nutshell is actually the introductory part of a larger document that presents also the definitions, motivations and exemplifications of the concepts and relationships presented so far. This complementary part is contained into the following PART III The DELOS Digital Library Reference Model Concepts and Relations.
PART III The DELOS Digital Library Reference Model
Concepts and Relations
The DELOS Digital Library Reference Model - 0.96

III.1 Introduction

As already stated, a Reference Model is a conceptual framework aiming at capturing significant entities and their relationships in a certain universe with the goal of developing more concrete models of it. Previous sections have outlined the motivation for the creation of the DL Reference Model, as well as an upper level description of its constituents. Conceptual Maps of the Reference Model Domains have been presented and described, providing a brief overview of each Domain concepts, 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 constituent parts. Concepts and relations are presented in a hierarchical fashion, providing thus an overview of the specialization 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 rational 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 DL universe. These are a kind of module which 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 marked [in squared brackets].
The DELOS Digital Library Reference Model - 0.96

[ 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 Analyze
        - 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 ]
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 Content Policy
C127 Disposal Policy
C128 Collection Development Policy
C129 Collection Delivery Policy
C130 Submission and Resubmission Policy
C132 Digital Rights
C134 Preservation Policy
C131 Digital Rights Management Policy
C133 License
C135 User Policy
C136 User Management Policy
C137 Registration Policy
C138 Personalization Policy
C139 Privacy and Confidentiality Policy
C140 Acceptable User Behaviour Policy
C141 Functionality Policy
C142 Access Policy
C143 Charging Policy
C144 Security Policy

[ Quality Resource ]
C146 Measure
C147 Objective Measure
C148 Subjective Measure
C149 Qualitative Measure
C150 Quantitative Measure
C151 Measurement
C152 Quality Parameter
C153 Generic Quality Parameter
C154 Economic Convenience
C155 Interoperability Support
C156 Reputation
C157 Security Enforcement
C158 Sustainability
C159 Documentation Coverage
C160 Performance
C161 Scalability
C162 Content Quality Parameter
C163 Authenticity
C164 Authoritativeness
C165 Freshness
C166 Integrity
C167 Preservation Performance
C168 Provenance
C169 Scope
C170 Size
C171 Fidelity
The DELOS Digital Library Reference Model - 0.96

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

[ Architectural Resource ]
. . C202 Architectural Component
. . C203 Software Architecture Component
. . . . C204 Software Component
. . . . C205 Application Framework
. . . . C206 Interface
. . . . C207 Framework Specification
. . . . C208 System Architecture Component
. . . . C209 Running Component
. . . . C210 Hosting Node
. . C13 Component Profile (also <isa> Metadata)
. . C133 License (also <isa> Policy)
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 (<identifyedBy>)
- 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 start from Web architecture and adds domain-specific aspect 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 specializations.

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 them 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:
- URI
- IRI
- ARKs
- Digital Object Identifier (DOI)
- 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, so 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 are group of Resources highly dynamic and time dependant, 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

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 into a Digital Library of Ph.D. thesis 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 on how a schema should be, or which schema best works as “the” schema for a specific kind of Resource. From a practical point of view, this leaves space 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: --

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 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 Ph.D. thesis which includes a representation of a performance, or a simulation experiment and the experimental data set adopted, 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 is to organize 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 decomposed in parts, each representing the highlight as captured by a different camera, etc. Moreover, the notions of Edition, View and Manifestation represent a further way of modelling Information Objects according to the semantics fixed by the IFLA-FRBR model [107]. This model is particularly useful in dealing with “document” Information Objects but can be extended and applied with profit 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 [190], where it is used to refer “identifiable immaterial items, such as a 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 analogously 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 from the Information Object it is an edition of.

Examples:
• An Information Object representing a study may be linked to the following Information Objects via <hasEdition> relationships:
  o its draft version is an Edition
  o the version submitted is an Edition
  o 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. This concept responds to the diversity of expressions of the same object that 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 table 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 [107].

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 along the time.

• Another example might consider an Information Object representing the outcomes of a workshop, three different views of this object can be envisaged:
  o the “full view” containing a preface prepared by the conference chair and the whole set of papers accepted and organised thematically,
  o the “handbook view” containing the conference program and the slides of each lecturer accompanied with the abstract of the papers organised per session, and
  o 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
The DELOS Digital Library Reference Model

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.

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 the modeling of them 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 characteristic. These data can be used in different contexts and for a diversity of purposes; usually, they are associated with an Information Object (more in general to a Resource through the <hasMetadata>) as a means for 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, (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) [49] provides an
excellent overview of the role of preservation metadata in the management of digital resources. PREservation Metadata: Implementation Strategies Working group, commonly referred to as PREMIS, defined a core set of preservation metadata elements that would provide support for the management of digital objects across systems, and time. They acknowledged 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 that 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, 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 human.
- **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), 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 a 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 great 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 does not only serve as a representation of Actor in the system but it 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 (<i>influencedBy</i>). For example, a particular instance of <i>Actor</i> may be entitled to <i>Search</i> within particular <i>Collections</i> and <i>Collaborate</i> with particular other <i>Actors</i>. The characteristics captured in an <i>Actor Profile</i> vary depending on the type of <i>Actor</i>, 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 <i>Actors</i> that interact with the <i>Digital Library</i>).

Examples:
- Group Profile, i.e., the <i>actor profile</i> capturing the characteristics of a <i>Group</i> as a single entity.
- Community Profile, i.e., the <i>actor profile</i> capturing the characteristics of a <i>Community</i> as a single entity.

**C13 Component Profile**

**Definition:** The <i>Metadata</i> attached to an <i>Architectural Component</i>.

**Relationship**
- Component Profile <i>isa</i> Metadata
- Component Profile <i>isa</i> Information Object (inherited from Metadata)
- Architectural Component <i>hasProfile</i> Component Profile
- Component Profile <i>profile</i> Policy
- Component Profile <i>profile</i> Quality Parameter
- Component Profile <i>profile</i> Function
- Component Profile <i>profile</i> Interface

**Rationale** The Component Profile is a specialization of the Metadata objects and plays exactly the same role, i.e. provide 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:** --

**C14 Annotation**

**Definition:** An Annotation is any kind of super-structural <i>Information Object</i> including notes, structured comments, or links, that an <i>Actor</i> may associate with a Region of a Resource via the <i>hasAnnotation</i> relation, 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 <i>regulatedBy</i> Policy). An Annotation may relate a Resource to one or more other Resources via the appropriate <i>hasAnnotation</i> relationship.

**Relationships:**
- Annotation <i>isa</i> Information Object
- Annotation is <i>identifiedBy</i> Resource Identifier
- Resource <i>hasAnnotation</i> Information Object about a Region
**Rationale:** Annotations can support co-operative 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 usage.

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 it 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 be themselves annotated. Actually, in literature there is an ongoing discussion whether annotations have to be considered either metadata or information objects. For the time being an Annotation is modelled as an Information Object because (i) it has been considered an additional information that increase the existing content by providing an additional layer of elucidation and explanation of it, and (ii) because of this, the Annotation itself takes the shape of an additional Information Object which can help people in understanding the annotated Resource. Actually, the status of Annotation is derived from the <hasAnnotation> relation linking Resources; this choice settles the long-standing issue whether Annotations are to be considered as Information Objects or as Metadata.

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

**Examples:**

---

**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 to thematic parts of the whole; they can be created by the Librarians in order to keep the set of Information Objects organised and to improve its access and usage; further, they can be created by authorised Content Consumers in order to implement their own personal views of the Digital Library Information 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 are the properties that information objects must satisfy in order to be collection members (truth conditions).

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

Examples: --

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 issues 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 foundational for the Search Function. However, it can be used for other purposes. This reference model uses them to capture the intension (<hasIntension>) definition of a Collection.

Examples:
* “Digital Library” is the representation of a query constituted by two tokens issued by an 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 contains the specified values in the identified fields.

C17 Ontology
Definition: An ontology is a formal conceptualization that defines the terms about a domain. Ontologies formalize a shared vocabulary about a domain [93]

Relationships:
* Ontology <isa> Information Object
* Resource Format is <expressionOf> Ontology

Rationale: The notion of ontology generalizes that of schema or format, as well as related notions, such as that of thesaurus. Ontologies may refer to different aspects of Information Objects, such as their structure, their content, their preservation and 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) [190], 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

November 2007
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 to its 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: --

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> Actor Domain
• Digital Library System <definedBy> Actor Domain
• Digital Library Management System <definedBy> Actor Domain
• Actor 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, their profiles (Actor Profile) exploited to personalize 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 it 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 characterized 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
The DELOS Digital Library Reference Model - 0.96

- **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 from or push content to the particular Digital Library. Although each distinct entity may be recognized 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 a different profile 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 Authoritativeness and a software agent may be characterized 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 co-operations 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.

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 may 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 enumerated its members or by capturing the similar traits of the Actors in a Query. In this second case the membership to 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 <belongsTo> 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 a 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.

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 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: --

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 [74] and works in accordance with the policy mechanism pervading the Policy Domain (Section II.2.5). A role is a kind of a pre-packaged generic profile and may be viewed as a packet of statements identifying the kind of Functions a 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 Role are distinguished within a DL context and are subsequently defined: Content Consumer, Content Creator, and Librarian, which are sub-roles of the End-User role. Apart from these roles and sub-roles that 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 only annotate existing Information Objects.

Examples: --

C23 End-User
Definition: The Role of the Actors that access the Digital Library for exploiting its Resources and possibly producing new ones.

Relationships
- End-User <isa> Role

Rationale: End-Users exploit DL facilities for providing, consuming, and managing DL content (usually Information Objects, in general Resources). It is actually a class of Actors further subdivided into the concepts of Content Creator, Content Consumer, and Librarian, each one 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: --

C24 Content Consumer
Definition: The Role of the Actors that access the digital library for 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 for exploiting (part of) its Resources. A person that searches (Search function) the contents of a digital collection or an external subscription service are instances of Content Consumers.

Examples: --

C25 Content Creator
Definition: The Role of the Actors that provide new Information Objects to be stored into the digital library or update already 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 that 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: --

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: See Section II.2.3.

Examples: --

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: See Section II.2.3.

Examples: --

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 take care to monitor its status as to guarantee the operation of the Digital Library.

Relationships
•  DL System Administrator <isa> Role

Rationale: See Section II.2.3.

Examples: --

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: See Section II.2.3.

Examples: --

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 a 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 visualization. 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 realized 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 be potentially 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 specialized into a set of specific but still quite generic subclasses, such as Access Resource. In practice, a Digital Library can use different specializations and combinations of these Functions intended to different Actors and Resources.

Examples:
- Access Resource
- Manage Resource

C32 Access Resource

Definition: The class of Functions which 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 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:** It is an Access Resource function which 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 Digital Library’s *Resources* and it 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 for searching 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\(^{17}\), i.e., a visual depiction of descriptors, namely tags, that are 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:** --

C35  Search

**Definition:** It is 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 a person 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. 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 them. 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 desired 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 resides 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 example, 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 judgment on the retrieved Resources at each iteration step. It improves effectively the discovery mechanism and the user satisfaction because it enhances the expressive power of the query language supported by the digital library.

C36 Acquire
Definition: It is an Access Resource function supporting an Actor in retaining Resources in existence past the lifetime of the Actor 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: It is an Access Resource function enabling an Actor to graphically perceive a Resource, such as an Information Object or an Actor Profile.

Relationships:
• Visualise <isa> Access Resource
• Visualise <actOn> Resource

Rationale: Resources may be complex and they may be comprised of several parts. For instance, an Information Object may combine information manifested in different media. The Visualise function must thus be tailored according to the End-User characteristics, like the device it uses or its personal setting, as well as to the characteristics of the object to be
Visualization is any technique for creating images, diagrams, animations, and so forth to communicate a message. Examples: Animation or drawing of diagrams are examples of Visualise function.

C38 Manage Resource

Definition: The class of Functions which 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. In specific, Manage Resource contains general categories of functions applied to each specific domain, as well as general Functions the specializations of which may appear in each individual domain.

Examples: --

C39 Create

Definition: It is 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: It is 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 may add the newly created Resource to either an incoming Resource Set, i.e. a temporary area that contains...
all the objects waiting for being published by the librarians, or directly to the DL Resource Set, i.e., the set of resource seen by DL Actors.

Examples: --

**C41 Withdraw**

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

**Relationships:**

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

**Rationale:** --

**Examples:** --

**C42 Update**

**Definition:** It is a Manage Resource function allowing an Actor to modify an already existing Resource.

**Relationships:**

- Update <isa> Manage Resource

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

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

**C43 Validate**

**Definition:** It is 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:** It is 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, which 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:** It is 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*, actually of their *metadata* or description, to the public through the DL in accordance with the *policies* assigned to them. In particular, the DL system may alert *Groups* or the wider public on the import of new *Information Objects* or *Collections* to the DL. Thanks to this characteristic, digital libraries become proactive systems instead of being just passive systems in charge of replying to user queries.  
**Examples:** --  

C47  Publish  
**Definition:** It is a *Manage Information Object* supporting an *Actor* in making *Information Objects* available into 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:** It is 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 accepted *information object’ Resource Format*.  

November 2007  
91/193
Examples: --

C49 Compose

**Definition:** It is 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 re-using 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:** It is a *Manage Information Object* function supporting the *Actor* in all activities related with 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 analyze 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 Analyze

**Definition:** It is a *Process* function supporting the *Actor* in all activities related with the analysis of an *Information Object*.

**Relationships**
- Analyze <isa> Process
- Linguistic Analysis <isa> Analyze
- Qualitative Analysis <isa> Analyze
- Statistical Analysis <isa> Analyze
- Scientific Analysis <isa> Analyze
- Create Structured Representation <isa> Analyze
- Compare <isa> Analyze

**Rationale:** This *function* encapsulates the capabilities to analyze 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:** It is an Analyze function supporting the Actor in all activities related with the analysis of the textual content of an Information Object.

**Relationships**
- Linguistic Analysis <isa> Analyze

**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 very crucial one especially in case of textual content of particular importance in that respect, i.e. old manuscripts, literature, etc. A very important specialization 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:** It is an Analyze function supporting the Actor in all activities related with the analysis of the quality of an Information Object or its metadata. It computes an appropriate Content Quality Parameter.

**Relationships**
- Qualitative Analysis <isa> Analyze
- 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 may 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:** It is 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 very crucial as it may provide the incentive for restorative or preventive measures for the ensuring 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:** It is an Analyze function supporting the Actor in all activities related with the statistical analysis of an Information Object.

**Relationships**
• **Statistical Analysis** <isa> Analyze

**Rationale:** This function represents the group of functions relevant to the computation of statistics related to an information object.

**Examples:** --

### C56 Scientific Analysis

**Definition:** It is an Analyze function supporting the Actor in all activities related with the scientific analysis of data represented as an Information Object.

**Relationships**

- **Scientific Analysis** <isa> Analyze

**Rationale:** This function represents the group of functions relevant to the scientific analysis of an Information Object. It includes actions and tools like running a model on a set of data, make scientific computations, offering handbooks with “live” formulas, etc. As DLs with scientific data are of specific importance to the scientific community, their incorporating a wide range of tools for the analysis of this data will be crucial in promoting research and knowledge creation, as well as education, in fields of natural sciences, medicine, etc.

**Examples:** --

### C57 Create Structured Representation

**Definition:** It is an Analyze function supporting the Actor in all activities related with the analysis of the structure of an Information Object and the creation of a representation of this structure.

**Relationships**

- **Create Structured Representation** <isa> Analyze

**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:** It is an Analyze function supporting the Actor in comparing two or more Information Objects, either primary ones or their metadata.

**Relationships**

- **Compare** <isa> Analyze

**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:** It is 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 summarization techniques.

Examples: --

C60 Physically Convert

Definition: It is 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. This 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 medium (including from paper to digital form), images into colour histograms etc.

Examples: --

C61 Translate

Definition: It is a Physically Convert function enabling Actors to perceive an Information Object in a language 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 the 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 both.

Examples: --

C62 Convert to a Different Format

Definition: It is 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 the 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 is a Function particularly useful for interoperability purposes.

Examples: --

C63 Extract

Definition: It is 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 which 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: It is 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 functions supporting the DL administrators in dealing with the DL user management. In particular, they cover the creation of new Actors, remove already existing ones, and regulating their rights, i.e. establishing the tasks they are entitled to perform and the information objects they are entitled to use as well their profile and associated personalization issues.

Examples: --

C65 Establish Actor

Definition: It is 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 the user creation, registration, login and application of their profile on their actions.

Examples: --
C66 Register

**Definition:** It is the 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 are the requirements imposed on the registration of novel users, the harder is for the system to ensure the identity of a user. The constraints imposed at registration time are a direct consequence of the audience the digital library is designed for. All these aspects are decided at the DL design time by the DL Designer and are related to policies and requirements that define the available Actor Profiles.

**Examples:** --

C67 Sign Up

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

**Relationships:**
- Sign Up <isa> Register

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

**Examples:** --

C68 Login

**Definition:** It is an Establish Actor function that enables an Actor to establish its identity in the DL.

**Relationships:**
- Login <isa> Establish Actor

**Rationale:** Login is performed by matching a set of qualities or characteristics that uniquely identifies 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 it may further perform authorization 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 a personalized 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 dedicated to adapt aspects of a digital library to the DL users 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 personalization functions contains customization and application of the Actor Profile to all DL Resources, whereas other functions may be related with the user feedback to the DL in order to improve provided Functionality and Content.

**Examples:** --

**C70 Apply Profile**

**Definition:** It is 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 personalize the DL functions, e.g. personalized search, recommendations, and so forth.

**Examples:** --

**C71 Manage Function**

**Definition:** It is 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 an update of new Functions.

**Examples:** --

**C72 Manage Policy**

**Definition:** It is 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 with all the DL Resources. In particular, they cover the creation of new Policies and remove or update already existing ones.

**Examples:** --

**C73 Manage Quality Parameter**

**Definition:** It is 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 dedicated to support *Actors* in using the DL as a common workspace. Some of the contained *Functions* may be specializations of other *Functions*, as well, related to information access.

**Examples:** --

C75  Exchange Information

**Definition:** It is 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:** It is 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, on-line forum or list servers.

**Examples:** --

C77  Find Collaborator

**Definition:** It is 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:** It is a *Collaborate* function that supports an *Actor* in authoring collaboratively *Information Objects.*

**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 agreement 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 with the management of general Content issues like 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:** It is a *Manage Content* function supporting *Actors* in creating, updating, exporting, importing, and removing Collections.

**Relationships:**
- Manage Collection <isa> Manage Content
The importance of collections as a mechanism for organising the digital library Content has been introduced in Section II.2.2. The \textit{collection management} function models the class of \textit{functions} for dealing with \textit{collections}. For example, by exploiting this class of functions, librarians can build new \textit{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 \textit{functions} for collection management may also be grouped under the \textit{Manage Resource} function. However, the management of collections should be differentiated as it contains two very important \textit{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 \textit{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 \textit{function} can be realised in different ways according to which typology of the DLMS ingestion mechanism is supported.

**Examples:** Harvesting of the content of an OAI [134] compliant Repository through OAI-PMH [135] is a kind of \textit{Import Collection}.

**C83 Export Collection**

**Definition:** The \textit{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 compliant with the OAI [134] and thus available through OAI-PMH [135] is a possible implementation of the \textit{Export Collection} function.

**C84 Preserve**

**Definition:** It is a \textit{Manage Content function} supporting an \textit{Actor} in all actions that involve the preservation of the DL Content.

**Relationships**

- Preserve <isa> Manage Content

**Rationale:** This group of \textit{functions} supports the definition of general preservation programs for specific \textit{collections}, the monitoring of their preservation state and the organization 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.4.

Examples: --

C85 Manage User

Definition: It is 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, like organizing campaigns for new membership in the DL.

Examples: --

C86 Manage Membership

Definition: It is a Manage User function supporting an Actor in organizing campaigns for new DL subscribers or maintaining the current ones.

Relationships
- Manage Membership <isa> Manage User

Rationale: The DL as an organization 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 ones. This is a function group containing functions like sending e-mails to the current users or the wider public informing of new content in the DL, making suggestions to users on content that may interest them, informing them on the ending of their subscription and suggesting a renewal, etc.

Examples: --

C87 Manage Group

Definition: The 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: The 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: The 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: It is 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 like defining general issues of how the functions will be presented and provided to the End Users.

Examples: --

C91 Monitor Usage

Definition: It is 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 have an insight on End User problems and issues relevant to specific functions.

Examples: --

C92 Manage Quality

Definition: It is a Manage DL function supporting an Actor in the management of general Quality Domain issues.

Relationships:
• Manage Quality <isa> Manage DL

Rationale: This function supports the management of quality domain issues.

Examples: --

C93 Manage Policy Domain

Definition: It is 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 with the “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* that supports the management of the DLSs 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:** It is a *function* that supports the creation of the DLSs 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:** It is 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:** It is 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:** It is 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 architectural components for the DLS.

**Examples:** --

**C101 Configure Architectural Component**

**Definition:** It is 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 a *architectural component* can be done by manually editing the configuration files as well as through a graphical configuration environment capable of driving 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:** It is 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 ensuring the quality values required by the *DL Designer*. As for the configuration functionality, the model does not constrain how this functionality is provided. Some DLMSs may offer sophisticated mechanisms for supporting a dynamic deployment while others may rely on a manual deployment to be done by the *DL System Administrator*.

**Examples:** --

C103 Monitor Architectural Component

**Definition:** It is 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 the information about the status of the allocation of DLS *system components*. The behaviour of this function can vary according to the information available and to the level of automatic monitoring supported. For instance, it can provide a mechanism allowing 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 advertising mechanism alerting the *DL System Administrator* when a certain characteristic of the deployed components exceed an established threshold.

**Examples:** --

C104 Configure DLS

**Definition:** The class of *Functions* which enable selecting and configuring the entities constituting 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 the DL Resources must comply with.

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 that supports the configuration of general Content issues.

Relationships:

• Configure Content <isa> Configure DLS

Rationale: --

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, initialize the DL with Actor specializations, initialize 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 it. It is obvious that the broader the range of customisations supported by a digital library management system is, the greater its capability to adapt to different scenarios is.

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 initializations 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 decisions 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 which describe the acceptable processes and/or procedures within an organization. Policy Domain impact is on how the complete system is designed and how it functions. This means that Policies should be incorporated in the Architecture Domain, implemented in Functionality Domain and should be clear to Actors because it effects their work with the Content Domain and influences their perception of 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 regulation governing interaction between Actors and the Digital Library. They provide mechanisms to constrain operations that Actors may/may not perform at 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 they should influence rather than be influenced by technical architecture, functionality, quality, or information content.

From 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 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 since they must serve the needs of institutions of various types and sizes which 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 aspect 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 is the content of the collection and how it will be developed in future) or prescriptive (there are strict procedures to follow, e.g. Registration Policy).

The currently identifies policy entities should be considered as examples; they are currently the most important in the digital libraries.

Examples:
Privacy and Confidentiality Policy is a Policy which describes what rules are followed to assure 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 what Functions should be present, and on the Digital Library Management System refers to the practical implementation of the Functions.

C113 Extrinsic Policy

Definition: A Policy defined outside of 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 that is imposed from a body outside the Digital Library (e.g. legal and regulatory frameworks). According to the type of the 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 of 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 reflect organisation’s mission and objectives, intended expectations as to how Actors will interact with the DL, 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 Explicit Policy on 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 in 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 $<$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 DLS that result in an interaction of policies resulting in unintended policy deployment.

This is an illustration how improper actions on Digital Library System level or Digital Library Management System level can cause 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 kinds of function specific types of Actors can have access to those that govern collection development.

**Examples:**

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

**C118 Descriptive Policy**

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

**Relationships:**
- Descriptive Policy $<$isa$>$ Policy
- Descriptive Policy $<$isa$>$ Policy by application

**Rationale:** Descriptive Policies are used to present in the form of explanation the aspects of a particular Policy. A Descriptive Policy is a Policy which 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 what is the scope and coverage of the DL.

C119 Enforced Policy
Definition: A Policy which 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:
Charging Policy which had been introduced into the DL is an Enforced Policy.

C120 Voluntary Policy
Definition: A Policy which is either not deployed within the DL, or which might be followed by the Actor according to his own decision.
Relationships:
- Voluntary Policy <isa> Policy
- Voluntary Policy <isa> Policy by compliance
Rationale: Voluntary Policy basically means a Policy which is followed according to the decision of the Actor. This is valid for all Policies which application is a matter of choice. In some cases users may comply to policies which are not officially communicated within the particular digital library – based on their previous experience with other digital libraries.
Examples:
The Collection Development Policy might be outlined in broad terms, but not enforced into practice.

C121 System Policy
Definition: A Policy that concerns an aspect of a system as a whole, being 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 which govern 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 done on the three levels and within the six domains of a digital library in a rational and consistent manner which would be effectively communicated to the Actors and would not harm their routine work.
The aim of Change Management Policy in 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 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 highest importance to define roles and responsibilities in change management, and to consider in detail the change management process and the support which the DLS and DLMS should provide for its smooth execution.

Examples:

Quality Parameter Measures which 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 which describes the principles and procedures related to this field.

Since Resources may be of 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 that Actors can expect in 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 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).

Under some circumstances polices 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 to provide it (charges, prioritizing in the case of several simultaneous requests, and timing) should be clearly expressed in the Support Policy.

**Examples**

Priorities (critical requests receive higher priority compared to standard ones) may be a component of Support Policy.

**C125 Connectivity Policy**

**Definition:** Policy taking care for maximum access to resources of the DL.

**Relationships:**

• Connectivity Policy <isa> Policy
• Connectivity Policy <isa> System Policy
• Connectivity Policy <govern> Actor

**Rationale:** Connectivity can be defined as the society’s capacity for communication with itself and with its global environment through the use of ICT.

Connectivity Policy should promote all ways which enable Actors from various environments to access resources. The DL should be accessible through various communication channels, incl. mobile devices.

**Examples:**

Digital divide is one of the threats which may be addressed within the Connectivity Policy.

**C126 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 which govern processes related to Content domain within the digital library system in its entirety on the three levels (DI, DLS and DLMS).

**Examples:** --
C127 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 to DL material from their collections. They are both prescriptive (Prescriptive Policy) and descriptive (Descriptive Policy).

**Examples:**
De-accession of a Resource which has not been requested for a certain time is part of a Disposal Policy.

C128 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) which take care of the DL development make publicly available their vision on the further development of DL through Collection Development Policy.

These intentions may reflect different scales – for example number of Resources, Resource sets, Collections. Collection Development Policy also can touch 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 is provided and how resources will be enriched through 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 are they to their purposes.

Collection Development Policies are of help to the institutions which develop DLs because they help to find and demonstrate the unique standing of a particular DLs.

The Collection Development Policy text (Policy <expressedBy> Information Object) usually describes categories of Resources, selection criteria, goals, priorities, services, accessibility.

**Examples:**
Estimation of current coverage of a DL is part of the Collection Development Policy.
C129 Collection Delivery Policy

**Definition:** Policy encompassing the constraints effecting how will Collections 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 what are 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.
- Offering 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.

C130 Submission and Resubmission Policy

**Definition:** Policies regulating editing 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 are governing which Actors can submit and resubmit Information Objects to the DL. They should be explicit (Explicit Policy), prescriptive (Prescriptive Policy), and intrinsic (Intrinsic Policy).

**Examples:**
Actors may have the right to submit, but not to edit Resources.

C131 Digital Rights Management Policy

**Definition:** Policy which explains what technologies control how content 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 which explains how the DL manages digital rights both from the perspective of the content creator/originator/owner and that of the Actor, and what technologies control the use of content within the DL. While DRM regulates the types of actions that can be done with information (for example, view, save, print, modify certain Manifestation), Digital Rights Management Policy explains DRM.

Digital rights management is developed as the 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, can not control how digital content is used (e.g. the right to view, save, print, read, or modify a work). Traditional library materials are more protected from unauthorised use because of their ‘physical’ nature. The development of digital content along with the electronic publishing and Internet which places in the digital environment the access to Manifestation of the resources imposes new issues within the field of copyright regulations.

Restrictions within Digital Rights Management Policy may depend on the Actor. Some restrictions may be time-dependant.

From Digital Library perspective, Digital Rights Management Policy means conditions, rules, terms and regulations governing interaction between Actors (virtual or real) and the DL in all cases where copyright or other rights on resources apply.

From Digital Library System perspective, Digital Rights Management Policy is the provision of the capability to define copyright-related policies.

From 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 to trace certain actions undertaken by the user and react properly.

**Examples:**
The actors belonging to a specific group can view resources, but not save local copies.

Viewing resources expires within a particular time.

**C132 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 particular rights of use of digital objects. In this sense they are a Descriptive Policy which regulates the possible uses of Information Object. The practical implementation of the Digital Rights is within Digital Rights Management Policy.
A broader understanding of digital rights defines them as all human rights which 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 any license is another example.

**C133 License**  
**Definition:** A *Policy* regulating the exploitation of a *Resource*.

**Relationship:**
- License <isa> Policy
- License <isa> Digital Rights Management Policy
- Resource <regulatedBy> License
- License <grantedTo> Actor

**Rationale:** License is the agreement by which the owner of intellectual property permits its use. In digital libraries license may be issued for specific uses of resources, or for designated functionality features which 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), BSD (Berkeley Software Distribution or Berkeley System Distribution) are examples of software licenses.

**C134 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 - decision making on archival needs, archiving practices, timing issues, access to archived materials, subsequent preservation measures for already archived materials, subsequent preservation measures for already archived material, maintaining preservation metadata, issues of interoperability of preserved materials.

**Examples:**
Reuse of preserved materials is part of the *Preservation Policy*.

**C135 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 which govern processes related to User domain within the digital library system in its entirety on the three levels (DL, DLS and DLMS).

Examples:
All policies which regulate issues regarding digital rights and user behaviour.

C136 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 enables to execute functions like issuing, managing, changing, sharing accounts; administration rights; sharing resources between multiple users.

Examples:
Account management is part of the User Management Policy.

C137 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.

C138 Personalization Policy
Definition: Policy enabling the DL to define what kinds of personalisation will be allowable and under what circumstances.

Relationships:
- Personalization Policy <isa> Policy
- Personalization Policy <isa> User Policy
- Personalization Policy <govern> Actor
- Personalization Policy <govern> Personalize Function

Rationale: The Personalization Policy has two roles, on the one hand it enables to recognise the user and his/her access rights, and on the other hand it enables DL to serve its Actors guaranteeing better Quality Parameters by offering Information Objects (in general Resources) which agree with user preferences. In the DLS the Functions which are used to assure personalization are Apply Profile, Customize, Login and Subscribe.
The DELOS Digital Library Reference Model - 0.96

Examples
The choice of representation layout on the basis of statistics of user behaviour is an example of Personalization Policy.

C139 Privacy and Confidentiality Policy
Definition: A policy which outlines 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 unauthorized access, destruction, misuse, modification, improper disclosure and loss.

Different rules may be applied to the use of various types of personal information - e.g., email addresses, postal address, log in names and passwords, users’ opinions entered through webpages.

Privacy and Confidentiality Policy principles should be embedded in the DL Functions which requires collection of data about the Actors (supplied or automatically collected).

Examples:
The use of e-mail addresses of the Actors for announcing new DL collection 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 agreed to this.

C140 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.

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 DL are part of the Acceptable User Behaviour Policy.
C141 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 which govern processes related to Functionality 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 one serious concern which practical implementation would be a security policy.

C142 Access Policy
Definition: Policy regulating permission or deny 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 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 intentionally or deliberately. Access Policy belongs to functionality and user domains, because on the one hand it prescribes what Functions are possible, and on the other hand regulate the work of the Actors.

Examples:
Use to Resources provided on the basis of IP address identification is an example of Access Policy.

C143 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 which could be applied: services provided on a longer period basis; 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 on an annual fee basis. The users which do not come from such an institution have only access to low quality images.
C144 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 are targeted to the protection of the digital library. They implement the rules and tools which assure security of services and integrity of the digital library.

**Examples:**
Ingest of resources into the library bound by a virus checking is an example of security policy.

C145 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 characterize, 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 Quality of Service (QoS) agreement.

**Examples:** --

C146 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.
C147 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 utilizing an analytical way to estimate the achieved quality. They also could be based on processing and comparing measurements between a reference sample and the actual sample as obtained by the system.

The distinction between Objective Measure and Subjective Measure is due to the fact that Quality Parameters can involve measurement methods which can either be independent from the subject who is conducting them or, on the contrary, express the viewpoint and the 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.

C148 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 which can either be independent from the subject who is conducting them or, on the contrary, express the viewpoint and the 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.

C149 Qualitative Measure

Definition: A Measure based on unit of measurement which 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 which could be applied to 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 to analyse texts).

**C150 Quantitative Measure**

**Definition:** A Measure based on unit of measurement which 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 the users in locating content.

**C151 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.

**C152 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
- Generalic 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 the Quality and provide information about how and how well a Resource performs with respect to some viewpoint. They express the assessment of an Actor, being it human or not, about the
Resource under examination. They can be evaluated according to different Measures, which provides 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 for example in the case of the Integrity of an Information Object, or a Resource Set, as for example 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 at hand. For example, the Availability is affected by Robustness and Fault Management: indeed, when Function is both robust and able to recover from error conditions, it is probable that also its Availability is 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 for adding 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 the 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 [17].

**Examples:** In order to make clear the relationship between Quality Parameter, Measure, and Measurement, we can take an example from the information retrieval field. One of the main Quality Parameters about an information retrieval system is the effectiveness, meant as its capability of answering to the user information needs with relevant items. This quality parameter can be evaluated according to many different Measures, such as precision and recall [171]; precision evaluates effectiveness meant as the ability of the system to reject useless items, while recall evaluates effectiveness meant as the ability of the system to retrieve useful items. The actual values for the precision and recall are Measurements and are usually computed by using standard tools, such as trec_eval18, which are Actors, in this case not human.

**C153 Generic Quality Parameter**

**Definition:** A Quality Parameter that concerns an aspect of a “system” as a whole, being 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

---

The DELOS Digital Library Reference Model - 0.96

• 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 characterize the quality of the a “system” in its entirety, in particular the Digital Library, the Digital Library System, and the Digital Library Management System.

Examples: --

C154 Economic Convenience

Definition: The General Quality Parameter which reflects how much favourable is the economic efficiency when using of 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 advantageous enough.

There are various appraisal methods that can be applied: for example, comparing the offered economic conditions with market rates for similar services, evaluating the possibility of obtaining value-added services in case of longer subscriptions, or assessing the flexibility of the offering with respect to their own usage needs.

Note that the implemented Charging Policy may influence the judgement about the Economic Convenience parameter.

Examples: An institution may find advantageous to pay a moderate subscription for offering access to standard functionalities to all of its users and pay an extra amount of money for having 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 the subscription to a set of standard Collections of a Digital Library and pay on a per-Information Object basis when you access Information Objects which belong to a Collection you are not subscribed to.

C155 Interoperability Support

Definition: The 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 interoperation with other Digital Libraries as well as the ability of integrating with legacy systems and solutions.

Connectivity Policy may affect Interoperability Support since it defines and controls how and how much a Digital Library should be accessible.

November 2007
Compliance To Standards may affect Interoperability Support since their use makes easier to interact with other systems.

The cost estimation of interoperability may be a component of the Economic Convenience measure.

Examples: Interoperability Support problems can cause delays or impossibility to fulfil user requests; thus they are also related to user satisfaction.

C156 Reputation

Definition: The Generic Quality Parameter which reflects the trustworthiness of a Digital Library.

Relationships:

- Reputation <isa> Generic Quality Parameter

Rationale: The Reputation concerns the “good name” of a Digital Library, the credit it owns by the user community, and its ability of being 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.

Examples: Examples of aspects which 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 which influence Reputation are: Economic Convenience, Usability, Dependability, and so on.

C157 Security Enforcement

Definition: The Generic Quality Parameter which reflects 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 Information
- Security Enforcement <affectedBy> Configure DL
- Security Enforcement <affectedBy> Enabling Function
- 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 may be affected by both Policies and Functions. In particular the Digital Rights Management Policy impacts the level of Security Enforcement of a Digital Library, since it defines how the content has to be controlled. The Access Information 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; the Enabling Functions, such as Authentication, Authorization, and Encryption, contribute to enforce the security of a Digital Library. Finally, also the User Behaviour can affect the Security Enforcement, since an Actor may compromise security for example by a careless use of its username and password.
Examples: An example of a factor which influences Security Enforcement is the capability of preventing un-authorized access to content or the saving of local copies of copy-righted material. Within the Policy domain the regulations should be clearly stated in the Digital Rights Management Policy.

C158 Sustainability
Definition: The Generic Quality Parameter which reflects the prospects of lastingness 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 to Standards
• Sustainability <affectedBy> Maintenance

Rationale: The Sustainability should take into consideration various factors, such as organizational and economical 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 for example the Change Management Policy or the Collection Development Policy.

Furthermore, Compliance To Standards may affect Sustainability, since they support future development of a Digital Library. Also Maintenance may affect Sustainability, since it controls how the Digital Library System evolves over the time.

Examples: Examples of factors which influence Sustainability are: the funding scheme which ensures the economic conditions for carrying on the Digital Library, the skills and willingness within the organization which 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.

C159 Documentation Coverage
Definition: The Generic Quality Parameter measuring the accuracy and clarity of the documentation describing a given Resource.

Relationships:
• Documentation Coverage <isa> Generic Quality Parameter

Rationale: The Documentation Coverage parameter concern the quality of the written documentation of a Resource.

Examples: Manuals which explain the use of Functions are typical Documentation Coverage examples. Other examples are the accuracy of on-line helps, better if contextual, or the selection provided by the Frequently Asked Question sections.

C160 Performance
Definition: The Generic Quality Parameter measuring the accomplishments of a Resource.

Relationships:
• Performance <isa> Generic Quality Parameter
• Performance <affectedBy> Capacity
**Rationale:** This *Generic Quality Parameter* provides an overall assessment about 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.

### C161 Scalability

**Definition:** The *Generic Quality Parameter* measuring the capability to increase *Capacity* as much as needed.

**Relationships:**
- Scalability <isa> Generic Quality Parameter

**Rationale:**

**Examples:**

### C162 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
- Authoritativeness <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 characterize the quality of the *Content*, in particular *Information Objects*, in a Digital Library.

**Examples:** Content quality is a moving target in a sense, but the requirements to the level of quality of various materials in the digital library and its scope have to be presented in the Collection Development Policy.

### C163 Authenticity

**Definition:** The *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\(^\text{19}\)\(^\text{[66]}\)[67].

**Examples:** The methods for data protection are key to assure authenticity of resources. Document sealing engines which timestamp and sign digitally every item in the digital library are an example of a solution which creates the proof that the documents they have not been modified from the original.

**C164 Authoritativeness**

**Definition:** The *Content Quality Parameter* measuring the reliability of *Resource* origin based on the trustfulness to the creator of the *Resource* or the frequency of accessing/referring to the resource by *Actors*.

**Relationships:**
- *Authoritativeness* \(<isa\>\) *Content Quality Parameter*

**Rationale:** The authoritativeness parameter measures the quality of being widely accepted or frequently accessed. It is helpful to compare digital libraries with similar or identical scope. NISO Z39.7 Library Statistics and ISO 11620 Library Performance Indicators suggest measures of usage especially for the libraries.

**Examples:** The authoritativeness 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 print-outs).

**C165 Freshness**

**Definition:** The *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 at hand.

**Examples:** For example, a stream of data coming from a sensor which monitors the temperature and the pressure of a patient should be updated a 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.

**C166 Integrity**

**Definition:** The *Content Quality Parameter* measuring the *Information Object* quality of being complete and integer.

**Relationships:**
- *Integrity* \(<isa\>\) *Content Quality Parameter*

**Rationale:** This parameter encompasses both the extent to which an *Information Object* is of sufficient breadth, depth, and scope for the task at hand, as pointed out in [17].

**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.

\(^{19}\) [http://www.interpares.org/](http://www.interpares.org/)
C167 Preservation Performance
Definition: The Content Quality Parameter is used to evaluate the need of undertaking actions which would assure 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 term.

C168 Provenance
Definition: The Content Quality Parameter reporting how well 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 much is possible to reconstruct the history and the 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 some join and/or transformations), tracing the provenance is not always a trivial task.

In particular, when we deal 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 [17] calls interpretability.

Provenance <affectedBy> Metadata, since they keep the additional information needed to trace the history of an Information Object.

Provenance <affectedBy> Annotation, since they allow us to trace the provenance and flow of data, report errors or remarks about a piece of data, and describing the quality or the security level of a piece of data [23]. In addition, [92] uses annotations in interactive visualization systems as a means for both capturing the history of user interaction with the visualization system and keeping track of the observations that a user may make while exploring the visualization.

Provenance <affectedBy> Preservation Policy, since it may influence the kind of Metadata which are kept about an Information Object.

Examples: --

C169 Scope
Definition: The Content Quality Parameter which measures 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 qualitative-oriented.

Examples: A Digital Library could contain the complete collection of works of a certain author, time period, genre. This is a content-related example.

C170 Size
Definition: The 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 is helping the user to get an idea about the resources. Size is also important parameter for the architecture and functionality of the DL.

Examples: The physical size of a collection calculated in bytes is important for estimation of the migration effort.

C171 Fidelity
Definition: The Content Quality Parameter measuring the accuracy with which an electronic system reproduces given a 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 very first 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 also significantly differ from its original appearance especially in layout – this difference is expressed through Fidelity.

C172 Perceivability
Definition: The Content Quality Parameter measuring what effort a 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 easy a 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 of them which are most likely to be quickly understood. Quite often images might be found as having higher perceivability than texts. Perceivability may also be used for answering the needs of special groups of users, for example providing audio content to visually impaired users.
C173 Viability

**Definition:** The 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.

C174 Metadata Evaluation

**Definition:** The 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.

C175 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 characterize the quality of the Functionality, in particular Functions, of a Digital Library.
Examples: --

C176 Availability

**Definition:** The Functionality Quality Parameter which indicates the ratio of the time a Functions 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, since Actors may be very disappointed when they try to use a Function and this 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, also Capacity may affect Availability, since in case of starvation of resources a Function may stop to be available.

Availability typically parallels with Dependability.

**Examples:** In the telephone services, high levels of availability are requested, the well-known “five-nines”, the 99.999% of up-time of the system, since nobody expects to pick up the received and to not hear the signal.

C177 Awareness of Service

**Definition:** The Functionality Quality Parameter which measures how well the perspectives Actor of a Digital Library are aware of its existence and Functions.

**Relationships:**
- Awareness of Service \(<isa\> Functionality Quality Parameter

**Rationale:** To measure the Awareness of Service, most often surveys are used. To increase the Awareness of Service, awareness system could be established as a DL functionality component.

**Examples:** Awareness of Service for target user groups is important component of the current information literacy.

C178 Capacity

**Definition:** The Functionality Quality Parameter measuring the limit on the number of requests that 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 successfully served.

It may affect Availability, Dependability and Performance. Indeed, when a Function operates beyond its Capacity, Availability MAY be compromised since the Function may stop working, as in the case of denial of service attacks; similarly, Dependability and Performance may be negatively affected if the Function does not complete its works or takes too much to complete.

**Examples:** --
C179  Expectations of Service
**Definition:** The *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 outlines the actual service and outlines the existing framework to the user. However, users might have different expectations from their experience with other DLs or other digital services. User expectations could be studied through surveys.

**Examples:** Users expect that clicking on image thumbnail will lead to opening of a larger size and higher quality image file.

C180  Fault Management Performance
**Definition:** The *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 offered service. It may be affected by *Robustness*, meant as the capacity of recovering from wrong inputs.

**Examples:**

C181  Impact of Service
**Definition:** The *Functionality Quality Parameter* measures the influence which the service offered by a *Function* have on the *Actor* 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 being improved with time. This parameter has a 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 be able to usually use similar interface. Since the user mastered how to use a specific set of functionalities organised in a particular interface, his expectation of service are also changed.

C182  Orthogonality
**Definition:** The *Functionality Quality Parameter* which indicates to what extent different *Functions* are independent from each other, i.e. do not affect each other.

**Relationships:**
- *Orthogonality* <isa> *Functionality Quality Parameter*

**Rationale:** The *Orthogonality* measures whether sets of Functions are independent from each other. DLs with full functional orthogonality or at least pronounced orthogonality will usually be much more intuitive to their users that 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 get too complicated.

Examples: The idea of orthogonality is that the same function is invoked by the same commands, from the same menu entries.

**C183 Dependability**

**Definition:** The 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 case of starvation of resources a Function may not work properly.

Examples: When a Actor types the URL address of a portal which gives access to a Digital Library, he/she expects that the address is correctly resolved and to be redirected to the correct site and not to a wrong one.

**C184 Robustness**

**Definition:** The 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 which may affect other Quality Parameters, such as Security Enforcement or Availability. Indeed, many kinds of attack which compromise the functioning of a service or gain unauthorized access to services, are based on ill-formed input, such as buffer overflows.

Examples: --

**C185 Usability**

**Definition:** The Functionality Quality Parameter which indicates the ease of use of a given Function.

**Relationships:**
- Usability <isa> Functionality Quality Parameter
- Usability <affectedBy> Orthogonality

**Rationale:** The Usability reports how much a given Function makes it easy for a Actor to achieve its goals. It can be evaluated by using different Measures: for examples, the Actor can indicate on a subjective scale the degree of Usability of a Function, or the time need to complete a task can be measured.

Examples: --

**C186 User Satisfaction**

**Definition:** The Functionality Quality Parameter which indicates how much an Actor is satisfied by a given Function.
Relationships:
- User Satisfaction <isa> Functionality Quality Parameter
- User Satisfaction <affectedBy> Usability
- 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 how much an Actor is satisfied by the capabilities offered by a given Function. Many factors can influence the User Satisfaction, such as the Usability, the Documentation Coverage, the Performance, the Availability, the Dependability and so on.

Examples: --

C187 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 characterize the quality of the User Domain, in particular Actors, of a Digital Library.

Examples: --

C188 User Activeness

Definition: A User Quality Parameter that reflects how much 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.

C189 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, if an Actor respects the copyright on the Resources of a Digital Library or if he makes unAUTHORIZED copies of such material.
C190 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 characterize the quality of a set of Policies.

**Examples:** --

C191 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 Policy (each of them being well defined) are free of contradictions.

**Examples:** --

C192 Policy Precision

**Definition:** A Policy Quality Parameter that represents the extent to which a set of policies have impacts that are defined and do not have unintended consequences.

**Relationships:**
- Policy Precision <isa> Policy Quality Parameter

**Rationale:**

**Examples:** --

C193 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 characterize the quality of the Architecture Domain, in particular Architectural Components, of a Digital Library System.

**Examples:** --
C194 Ease of Administration
Definition: The Architecture Quality Parameter measuring the presence and easiness 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, as Digital Library Systems potentially are.
Examples: --

C195 Compliance to Standards
Definition: The Architecture Quality Parameter measuring the degree to which standards have been adopted in developing an Architectural Component.
Relationships:
• Compliance to Standards <isa> Architecture Quality Parameter
Rationale: This parameter influences both the Interoperability Support, because the adoption of standards increase the easiness of interoperation with other entities, and the Sustainability of a Digital Library, since open standards support keeping up a Architectural Component 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.

C196 Ease of Installation
Definition: The Architecture Quality Parameter measuring the easiness 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 a running Digital Library System.
Examples: --

C197 Load Balancing Performance
Definition: The Architecture Quality Parameter measuring the capacity to evenly spread and distributed work 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 performances and responsiveness of a Digital Library System.
Examples: --

C198 Log Quality
Definition: The 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 way of understanding the Actor behaviour and personalize the Digital Library System accordingly; therefore logs can be a useful input for the Personalize functions and for creating Actor Profiles.

Examples: There are various standards for creating logs. For example, in the case of Web, there is W3C Extended Log Format [99].

C199 Maintenance Performance
Definition: The Architecture Quality Parameter concerning 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, since it involves keeping the current system properly functioning and evolving it for facing future technological developments.

Examples: A maintenance plan may concern programmed hardware updates, controlled migration towards new software and hardware environments, and so on.

C200 Redundancy
Definition: The 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 performances of a system and may improve the Availability, Dependability and Robustness of a Digital Library System.

Examples: --

C201 Architecture Domain
Definition: One of the six main concepts characterising the digital library universe. It represents the various aspects related to the software systems concretely realising the digital library universe.

Relationship
• 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 captures 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 foundational concept has been largely underestimated in the past. The importance of its domain and its modeling is described in Section II.2.7.

Examples: --
C202 Architectural Component

**Definition:** A constituent part or an element of a software system implementing one or more Functions that can be autonomously managed and that contributes to realize the Architecture of a Digital Library System.

**Relationship**
- 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 potentiality 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 later. Moreover, it should be possible/easy to fix the bugs;
- **Affordable** - both to buy and to maintain. Reuse and replacement feature of the component oriented approach contribute to reduce the "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 that are in charge to concretely realize the Digital Library. Being a Resource to be managed such components should have a description, i.e. Component Profile, characterising it and promoting its correct usage. This description may assume diverse forms ranging from human oriented description, e.g. a textual description in natural language, to a machine understandable one, e.g. the WSDL as in the case of web services. 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.

C203 Software Architecture Component

**Definition** An Architectural Component contributing to implement the Software Architecture of a system.

**Relationship**
• 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 represents “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 current systems. In particular, thanks 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 in charge to provide a bunch of Functions may consist of smaller Software Architecture Components (usually logical components the whole service is organised in) each implementing specific sub-tasks needed to accomplish 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.) implementing 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 feature 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.
C204 Software Component

**Definition** The *Software Architecture Component* representing a program coded to provide a set of Functions.

**Relationship**
- 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 it makes system development potentially “cheap”.

**Examples:** --

C205 Application Framework

**Definition:** A *Software Architecture Component* representing a middleware, i.e., a software that connect and support the operation of other *Software Architecture Components*, available at the *Hosting Nodes*. It provides the run-time environment for the *Running Component*.

**Relationship**
- 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:**
C206 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.

**Relationship**
- 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 [135] prescribed the Interface an Architectural Component acting as OAI compliant data provider [134] must implement in order to serve an Architectural Component willing to act as OAI application providers

C207 Framework Specification

**Definition:** The Software Architecture Component prescribing (<prescribe>) the set of Interfaces and protocols an Architectural Component should conform to (<conformTo>) in order to interact with the other Architectural Components of the same system by design.

**Relationship**
- 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 to operational context an Architectural Component has been designed to operate in.

**Examples:**
- Enterprise JavaBeans
- Component Object Model

C208 System Architecture Component

**Definition** An Architectural Component contributing to implement the System Architecture of a system.

**Relationship**
- 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)
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 as well as discussed in the Architectural Component definition.

Examples:

- An server ready to host and run (Hosting Node) the software (Software Component) implementing a certain Function, e.g. the Search

C209 Running Component

Definition: The Architectural Component realizing a Software Component.

Relationship

- Running Component <isa> Architectural Component
- Running Component <invoke> Running Components
- Running Component <hostedBy> Hosting Node

Rationale: The concrete realization 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 running web server implementing the user interface of the DELOS Digital Library (http://www.delos.info)

C210 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.

Relationship

- 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 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 bunch of software needed to host and run the Software Component implementing the user interface of the DELOS Digital Library (http://www.delos.info)

C211 Software Architecture

Definition: The set of Software Architecture Components organised to form a system.

Relationship

• Software Architecture <consistOf> Software Architecture Component

Rationale: Each software system is characterised by a set of software pieces organised in a structure making them capable to work together. This organised set of software is the Software Architecture. To help software engineers in designing their systems a set of well-proven generic scheme for solution of recurring design problems have been identified, i.e. software architecture patterns [39]. Patterns capture existing, well-proven experience in software development and help to promote good design practise. 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

C212 System Architecture

Definition: The set of System Architecture Components organised to form a system.

Relationship

• 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, the one of the Software Architecture and the one of the System Architecture. The System Architecture, being 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 having a distributed System Architecture. The more flexible is the Software Architecture a system adopt, the larger will be the potential range of application scenarios it can be successfully exploited.

Examples:

• The set of servers and services realising the DELOS Digital Library (http://www.delos.info)
C213 Purpose
Definition: The motivation characterizing the \texttt{associatedWith} relationship.

Relationships:
- \texttt{Resource <associatedWith> Resource} to a certain \texttt{Purpose}

Rationale: The \texttt{associatedWith} relation is one of the powerful ones allowing to build compound Resources, i.e., Resources obtained by combining existing constituent Resources as to form a new knowledge bundle having a value added with respect to the single Resources when considered as single island of information. Various kind of associations are possible, and this diversity is captured by the \texttt{Purpose} concept attached to each instance of the \texttt{associatedWith} relation.

Examples:
An Information Object representing an experiment (itself composed of various Information Objects representing, e.g., the dataset the experiment is carried on, the dataset representing the outcomes, the description of the procedure adopted) is \texttt{associatedWith} the Information Object representing the scientific publication in an outstanding Journal of the filed with the \texttt{Purpose} of scholarly dissemination.

C214 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:
- \texttt{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. [156]. 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 which anchor an Annotation to the whole Resource, as well as, Regions able to anchor an Annotation to a specific part of a Resource.

Examples: A piece of text, e.g. a paragraph, of a Information Object representing this volume is a Region an Annotation can be attached to.

C215 Digital Library
Definition: An organization, which might be virtual, that comprehensively collects, manages, and preserves for the long term rich Information Objects, and offers to its Actors specialized Functions on those Information Objects, of measurable quality, expressed by Quality Parameters, and according to codified Policies.

Relationship
- \texttt{Digital Library <manage> Resource}
- \texttt{Digital Library <manage> Information Object}
- \texttt{Digital Library <serve> Actor}
- \texttt{Digital Library <offer> Function}
- \texttt{Digital Library <agreeWith> Policy}
- \texttt{Digital Library <tender> Quality Parameter}
- \texttt{Digital Library System <support> Digital Library}
- \texttt{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. Sec. I.2) active in this universe, i.e. Digital Library, Digital Library System, and Digital Library Management System. The former is the more abstract among 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](http://www.delos.info))
- The European Library ([http://www.theeuropeanlibrary.org](http://www.theeuropeanlibrary.org))
- The National Science Digital Library ([http://nsdl.org](http://nsdl.org))

### C216 Digital Library System

**Definition:** A software system that is based on a given (possibly distributed) Architecture and provides all Functions required by a particular Digital Library. Actors interact with a Digital Library through the corresponding Digital Library System.

**Relationship**
- 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. Sec. 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. As any running software systems 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](http://www.delos.info))
The set of servers, services, and software realising The European Library ([http://www.theeuropeanlibrary.org](http://www.theeuropeanlibrary.org))

The set of servers, services, and software realising the National Science Digital Library ([http://nsdl.org](http://nsdl.org))

**C217 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 foundational for Digital Libraries, and (ii) to integrate additional software components offering more refined, specialized, or advanced functionality.

**Relationship:**
- 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. Sec. 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. Sec. I.4). Depending on the set of Functions DLMS provide Actors with, different types of such systems can be implemented (cf. Sec. I.2).

**Examples:**
- OpenDLib ([http://www.opendlib.com](http://www.opendlib.com)): the DLMS used to create and maintain the DELOS Digital Library ([http://www.delos.info](http://www.delos.info)).
- DILIGENT [62]: a prototypical DLMS capable to deploy Digital Library Systems by relying on a set of Resources ranging from Software Components to Hosting Nodes dynamically gathered through Grid technologies.
- The DelosDLMS [175]: a DLMS built by integrating software and services developed by DELOS partners.
III.4 Relations’ Hierarchy

[ Generic Relations ]
.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 invokes
.R19 belongTo
.R20 hasEdition
.R21 hasView
.R22 hasManifestation
.R23 hasIntension
.R24 hasExtension

[ Content Relations ]
.R25 profile

[ 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

---

“Classifiers”, i.e. items added to the hierarchy for organisational purposes are marked [in squared brackets].
The DELOS Digital Library Reference Model - 0.96

. . R36 retrieve
. . . . 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 foundational 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 about them 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:**

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:** One *resource* must have exactly one format, whereas the same format can (obviously) be used by many resources.

This relation is commonly called “instance of” in object models. However, in order to avoid confusion with the instance of relation of the present model, it is given a different name.

**Examples:** --

R3 expressionOf

**Definition:** The relation connecting a *Resource Format* to the *Ontology* which 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 that a schema borrows from many ontologies. In this latter case, the schema operates a sort of integration of several ontologies.

**Examples:** --

R4 conformTo

**Definition:** The relation connecting *Architectural Components* to *Framework Specifications* they comply with.

**Rationale:** *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. *Framework Specification* <prescribe> *Interfaces Architectural Components* should implement. The compliance with the *Framework Specification* guarantee the *Architectural Components* of being interoperable with the others *Architectural Components* of the same system by design.

**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.

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 it is associated to. The same Quality Parameter can be associated with many Resources.

Examples: --

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 context (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 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.

Examples: This volume is an Information Object associated to another Information Object representing the its Dublin Core metadata record via the <hasMetadata>.

R8 describedBy
Definition: The relation connecting Resources to Information Objects to 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 to one Resource.

Examples:
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 to one *Actor*.

**Examples:**

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:**

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. Sec. III.3 C14) is defined as the range of the `<hasAnnotation>` relation. Happily, this choice settles the long-standing issue whether annotations are to be considered as objects or as metadata.

**Examples:** --

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 mainly intended for the materialisations of *Resources* like *Policy* and *Quality Parameter* but can be applied to any type of *Resource*.

A *Resource* can be associated to many *Information Objects* materialising it in different ways. A materialising *Information Object* must be associated with one *Resource*.

**Examples:** --

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 their selves and may be constituents of any number of *Resources*. In the case of compositional nature the whole strongly owns its parts, i.e. if the whole *resource* is copied or deleted, its parts are copied or deleted with it.

**Examples:**

- A book has part 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 composed by many Architectural Components. The same Architectural Component can be a component of many Architectural Components.

Examples: --

R15 associatedWith
Definition: The relation connecting a Resource to the Resources which are linked to the former according to a certain Purpose.
Rationale: In addition to the explicitly identified pool of relations connecting Resources, this relation allows to specify cross-resource links with respect to a well-known Purpose.
No constraints about 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 or may appear as the second term of a <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 all the digital library system. Thus it is supposed that the system follows the component-oriented approach. The <use> relations capture the

Examples: --

R17 conflictWith
Definition: The relation connecting Architectural Components to Resources they realises.
Rationale: --
Examples: --

R18 invokes
Definition: The relation connecting Running Components to Running Components they use to accomplish their task.
Rationale: --
Examples: --

R19 belongTo
Definition: The relation connecting Resources to the Resource Sets in whose extension they belong. A specialisation of it is the relation connecting an Actor to a Group which defines which user group an actor belongs to.
Rationale: A Resource may be a member of any number of Resource Set extensions and, vice versa, the extension of a Resource Set may include any (finite) number of resources.

Examples:
- An Information Object belongs to a Collection
- An Actor belongs to a Group

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 makes up 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 which 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 that may result useful in specific application contexts.

The same Information Object may have different Information Objects linked through the <hasView> relation. Vice versa, 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 reporting the graph of the evolution of the values measured by the sensor along the 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;
o  <hasView> the Information Object representing the “handbook view” and containing the conference program and the slides of each lecturer accompanied with the abstract of the papers organised per session, and

o  <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 firstly made by G. Frege, for whom the sense of an expression corresponded to what we intuitively think as the meaning of the expression. R. Carnap later suggested that the sense of an expression is a function which 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 [63]. Davidson argued that giving the meaning of a sentence is equivalent to stating its truth conditions.

The intention 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: --

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: --

R25  profile

Definition: The relation connecting Component Profiles to the Resources they describe. Component Profiles should describe (at least) Functions, Policies, Quality Parameters, and Interfaces inherent to Architectural Components they are associated with via the <hasProfile> relation.
Rationale: Component Profile is the Metadata associated with each Architectural Component for its management. The <profile> relation captures the aspects expected to be captured by this kind of profile.

Examples:
- The Functions yielded by the Architectural Component is a typical information expected to be included in the Component Profile.
- The Quality Parameters guaranteed by the Architectural Component is a typical information expected to be included in the Component Profile.

R26 perform
Definition: The relation connecting Actors to Functions they use to do accomplish their Digital Library activities.

Rationale: Functions has 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 accomplish their goals.

Examples:

R27 play
Definition: The relation connecting an Actor to a Role which defines the Role(s) of the Actor.

Rationale: DL Actors may play different Roles in the DL, for example be at the same time Content Creator and Content Consumer.

Examples: --

R28 interactWith
Definition: The relation connecting Functions to Functions that expresses the interaction between them.

Rationale: This Function is fundamental for the modelling of the workflow of execution for the Functions. It defines an order between them in order to clarify which Function follows the current one.

Examples:

R29 influencedBy
Definition: The relation connecting Functions to Actor Profiles which expresses that functions are influenced by specific user characteristics.

Rationale: This relation is very important for personalization, as it expresses that functionality is related to and influenced by the Actor Profile of the Actor executing it, adapting thus to the Actor specific needs.

Examples:

R30 actOn
Definition: The relation connecting Functions to Resources they operate on.

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:

R31 create
Definition: The relation connecting the Create Functions to Resources they create.
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 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 Function 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 collaborating authoring is 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 connect Find Collaborator Functions to Actors they find. Another specialization 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 specialization 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 Query to Result Sets they return.

**Rationale:** When a Query is performed as a result of 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.

**Rationale:** Each Policy to be effectively implemented must be applied to Resources. This relation captures those Resources 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 planned 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 properly work collaboratively 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 Policy (in general Resources) with opposite meaning. It is introduced in order to facilitate the understanding of bi-polar sets of concepts.

**Examples**: Extrinsic policy and Intrinsic policy form a pair where each concept is `<antonymOf>` the other concept.

**R43 influence**

**Definition**: The relation connecting Quality Parameter 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. One important aspect is how decisions for applying specific Policies could be taken within the DL. This relation captures the cases in which the decision is based on Quality Parameters.

**Examples**: The value of 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 about a Resource.

**Rationale**: See Quality Parameter, Actor, and Resource.

**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**: See Quality Parameter and Measure.

**Examples**: See Quality Parameter and Measure.

**R46 measuredBy**

**Definition**: The relation connecting Quality Parameters to the Measurements which 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 which influence their determination.

**Rationale**: See Quality Parameter and Resource.

**Examples**: See Quality Parameter and Resource.

**R48 accordTo**

**Definition**: The relation connecting Measurements to the Measures which define how they have to be obtained.

**Rationale**: See Quality Parameter, Measure, and Measurement.

**Examples**: See Quality Parameter, Measure, and Measurement.
R49  implement
Definition: The relation connecting Architectural Components to the Resources they realises.
Rationale: --
Examples: --

R50  realisedBy
Definition: The relation connecting Software Components to the Running Components realising them.
Rationale: --
Examples: --

R51  support
Definition: The relation connecting Application Frameworks to the Running Components they support the operation.
Rationale: --
Examples: --

R52  hostedBy
Definition: The relation connecting Running Components to the Hosting Nodes physically hosting them.
Rationale: --
Examples: --
Conclusions

This document 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 level of abstractions. This structure has been introduced to accommodate the needs of the many different players of the digital library universe that are interested in understanding the digital library systems at different level of details.

The model presented is the result of a three years 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 upon 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 outside, as well as by other groups around the world. In such an aspect, it has to be intended as a collective effort made by the digital library community to agree on a common ground. This is meant to be useful not only for current activities but also as a springboard for future work.

The presented model 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 developments in this area.

Because of the broad coverage of the digital library field, the heterogeneity of the involved actors, and the lack of any previous agreement on the foundations of the field, the DELOS Reference Model has to be considered as a living document shared by the digital library community. For this reason, this document is made available in its subsequent versions, each taking advantage from the previous one and from the public comments received.

The framework introduced so far does not aim at covering every specific aspects 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 relying on this core part and by introducing more detailed concepts and relationships. We expect that in the future many more focussed, fine-grained models, developed by other communities will be progressively integrated in the current model thus creating an increasingly richer framework capable of capturing more and more aspects of the digital library universe.