DL.org: Coordination Action on Digital Library Interoperability, Best Practices and Modelling Foundations

Funded under the Seventh Framework Programme, ICT Programme – "Cultural Heritage and Technology Enhanced Learning"

Project Number: 231551

Deliverable Title: D3.2b The Digital Library Reference Model

Submission Due Date: September 2010

Actual Submission Date: April 2011

Work Package: WP3

Responsible Partner: CNR

Deliverable Status: Final

European Commission
Information Society and Media

SEVENTH FRAMEWORK PROGRAMME

Document Information

Project	
Project acronym:	DL.org
Project full title:	Coordination Action on Digital Library Interoperability, Best Practices & Modelling Foundations
Project start:	1 December 2008
Project duration:	24 months
Call:	ICT CALL 3, FP7-ICT-2007-3
Grant agreement no.:	231551
Document	
Deliverable number:	D3.2b
Deliverable title:	The Digital Library Reference Model
Editor(s):	L. Candela, A. Nardi
Author(s):	L. Candela, G. Athanasopoulos, D. Castelli, K. El Raheb, P. Innocenti, Y. Ioannidis, A. Katifori, A. Nika, G. Vullo, S. Ross
Reviewer(s):	C. Thanos
Contributor(s):	(DELOS Reference Model Authors) L. Candela; D. Castelli; N. Ferro; Y. Ioannidis; G. Koutrika; C. Meghini; P. Pagano; S. Ross; D. Soergel; M. Agosti; M. Dobreva; V. Katifori; H. Schuldt
Participant(s):	CNR, NKUA, UG
Work package no.:	WP3
Work package title:	Digital Library Models and Patterns
Work package leader:	CNR
Work package participants:	CNR, NKUA, UG
Est. Person-months:	6
Distribution:	Public
Nature:	Report
Version/Revision:	1.0
Draft/Final	Final
Total number of pages:	273
(including cover)	
Keywords:	Reference Model; Content Domain Model; User Domain Model; Functionality Domain Model; Policy Domain Model; Quality Domain Model; Architecture Domain Model; Conformance Criteria; Conformance Checklist;

Disclaimer

European Commission
Information Society and Media

The European Union was established in accordance with the Treaty on European Union (Maastricht). There are currently 27 Member States of the Union. It is based on the European Communities and member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

DL.org is funded by the European Commission under the 7th Framework Programme (FP7).

Table of Contents

List of Figures

Summary

This document represents the final release of the Digital Library Reference Model produced by the DL.org project. It has been produced by using the DELOS Digital Library Reference Model released by the DELOS Network of Excellence as firm starting point. This release maintains, consolidates and enhances the previous one by applying a number of revisions and extensions.

The document maintains the structure of the previous release, i.e., it is a volume consisting of different parts. However, the current version consists of four parts each forming a self contained artefact, i.e., '*The Digital Library Manifesto'*, '*The Digital Library Reference Model in a Nutshell*', '*The Digital Library Reference Model Concepts and Relations',* and '*The Digital Library Reference Model Conformance Checklist'*. The Digital Library Manifesto declares the intentions, motives, overall plans and views of a long term initiative leading to the production of a foundational theory for Digital Libraries; it introduces the main notions characterising the whole Digital Library domain. The Digital Library Reference Model in a Nutshell briefly introduces the overall picture underlying a comprehensive model conceived to capture the essence of Digital Libraries in terms of the main domains characterising them, the principal concepts existing in each domain and the main relationships connecting such concepts. The Digital Library Reference Model Concepts and Relations present in detail the main concepts, axioms and relationships characterising the Digital Library domain independently from specific standards, technologies, implementations, or other concrete details. For each concept and relations included in the model, the document provides a detailed characterisation comprising a definition, the set of connections with other concepts, the rationale explaining its existence and a set of examples of concrete instances of the specific entity. Finally, the Digital Library Reference Model Conformance Checklist is one of the main novelties included in this release. This checklist provides a set of statements that will enable assessors to determine whether or not the 'digital library' they are analysing is compliant with the *Digital Library Reference Model*.

About this Document

The Digital Library universe is a complex framework. The growth and evolution of this framework in terms of approaches, solutions and systems has led to the need for common foundations capable of setting the basis for better understanding, communicating and stimulating further evolution in this area. The DELOS Digital Library Reference Model aims at contributing to the creation of such foundations. This document exploits the collective understanding on Digital Libraries that has been acquired by European research groups active in the Digital Library field for many years, aggregated under the DELOS Network of Excellence umbrella in the past, under the DL.org umbrella, as well as by the international scientific community operating in this domain. The resulting document identifies the set of concepts and relationships that characterise the essence of the Digital Library universe. This model should be considered as a roadmap allowing the various stakeholders involved in the Digital Library domain to follow the same route and share a common understanding when dealing with the entities of such a universe.

This document presents a revised version of the Digital Library Reference Model resulting from consolidation and enhancement activities performed in the framework of the DL.org project. It introduces the principles governing such a model 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 breadth of the Digital Library universe and its evolving nature, as well as the lack of any previous agreement on its foundations, the Reference Model is by necessity dynamic. The model is extensible and, should other concepts be needed, they could easily be added in the appropriate place. Continuous evolution of this document will lead to well-formed and robust definitions, shared by the Digital Library community.

The document is organised in four parts, each potentially constituting a document on its own. Each of the four parts describes the Digital Library universe from a different perspective between abstraction and concretisation. Thus each part is equally important in capturing the nature of this complex universe. The second part is based on the first one, and the third part is based on the second, i.e., they rely on the notions described previously when introducing additional information that characterises these notions more precisely. In particular, 'PART I The Digital Library Manifesto' sets the scene governing the whole activity and introduces the main notions characterising the whole Digital Library universe in quite abstract terms; 'PART II The Digital Library Reference Model in a Nutshell' treats these notions in more detail by introducing the main concepts and relationships related to each of the aspects captured by the previous one; 'PART III The Digital Library Reference Model Concepts and Relations' describes each of the identified concepts and relations in detail by explaining their rationale as well as presenting examples of their instantiation in concrete scenarios; finally, 'PART IV Digital Library Reference Model Conformance Checklist' identifies and documents a set of criteria that can be used to determine whether or not a 'digital library' is compliant with the Digital Library Reference Model.

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

Section I.1 introduces 'PART I The Digital Library Manifesto' by providing the driving force behind the whole activity. Section I.2 presents the relationships between the three types of relevant 'systems' in the Digital Library universe, namely Digital Library (DL), Digital Library System (DLS) and Digital Library Management System (DLMS). Section I.3 describes the main concepts characterising the above three systems and thus the whole Digital Library universe, i.e., organisation, 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, manager and software developer. Section I.5 describes the reference frameworks needed to clarify the DL universe at different levels of abstraction, i.e., the Digital Library Reference Model and the Digital Library Reference Architecture. Section I.6 records some concluding remarks on The Digital Library Manifesto.

Section II.1 introduces 'PART II The Digital Library Reference Model in a Nutshell' by summarising the content of the Manifesto and setting the basis for reading and using the rest of this part. Section II.2 presents the constituent domains by briefly describing their rationale and providing for each of them the concept map that characterise them by introducing the main related concepts and the relations connecting them. Section II.3 introduces the reader to possible exploitations of the model. In particular, it addresses Interoperability and Preservation issues. For each one, it describes the issue by pointing out the tools that the Reference Model makes available for dealing with it. Section II.4 discusses related work. In particular, it highlights the similarities and differences between this Reference Model and similar initiatives like the 5S Framework and the CIDOC Conceptual Reference Model. Section II.5 records some concluding remarks on the Digital Library Reference Model as presented in PART II.

Section III.1 introduces 'PART III The Digital Library Reference Model Concepts and Relations' by highlighting the role of this part. Section III.2 presents the hierarchy of Concepts constituting the Reference Model. Section III.3 provides a definition for each of the 200+ Concepts currently constituting the model. Each definition is complemented by the list of relations connecting the concept to the other concepts, the rationale for including this concept in the model, and examples of concrete instances of the concept in real-life scenarios. Section III.4 presents the hierarchy of the identified Relations. Section III.5 provides a definition for each of the 50+ Relations currently constituting the model. Each definition is complemented by the rationale for including it in the model and some examples of concrete instances in real-life scenarios.

Section IV.1 introduces 'PART IV Digital Library Reference Model Conformance Checklist' by setting the scene for this list of items / points to be considered to determine whether or not a 'digital library' conforms to the Reference Model principles. Sections IV.2, IV.3 and IV.4 describe (*a*) the scope of the tool; (*b*) the how the tool has been developed; and (*c*) the how the tool is expected to be used, respectively. Section IV.5 overviews the list of criteria that have been identified to achieve the checklist goal. Section IV.6 describes in detail the criteria constituting the checklist clustering them in mandatory, recommended and optional criteria.

A concluding section summarises and completes the entire document.

The document comprises also appendixes. Appendix A provides the concept maps of the Reference Model in A4 format to improve their readability. Appendix B lists others contributors to this artefact along its lifetime and acknowledges the contribution.

PART I The Digital Library Manifesto

I.1 Introduction

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

I.1.1 What is a Manifesto?

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

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

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

---

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

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

[3] http://agilemanifesto.org/

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

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

I.1.2 Motivation

Digital Libraries constitute a relatively young scientific field, whose life spans roughly the last twenty years. Instrumental to the birth and growth of the field have been the funding opportunities generated by the ‘Cultural Heritage and Technology Enhanced Learning’ (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 twenty 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.

The digital library concept can be traced back to the famous papers of foreseer scientists like Vannevar Bush (Bush, 1945) and J.C.R. Licklider (Licklider, 1965) identifying and pursuing the goal of innovative technologies and approaches toward knowledge sharing as fundamental instruments for progress. However, the evolution of “digital libraries” has not been linear and this has created several conceptions of what they are, each one influenced by the perspective of the primary discipline of the conceiver(s) or by the concrete needs it was designed to satisfy. As a natural consequence, the “history” of Digital Libraries is the history of a variety of different types of information systems that have been called “digital libraries” (Candela, Castelli, & Pagano, 2011). These systems are very heterogeneous in scope and functionality and their evolution does not follow a single path. In particular, when changes happened this has not only meant that a better quality system was been conceived superseding the “preceding” ones but also meant that a new conception of digital libraries was born corresponding to new raised needs. Nevertheless, looking back at the individual achievements of all the projects and initiatives, it can clearly be seen that there is substantial commonality among many of them; the bottom-up development of the field so far has provided enough ‘data points’ for patterns to emerge that can encapsulate all efforts.

Despite the young age of the field of Digital Libraries, it has made a long journey from its initial conception to the present state of the art and has reached a level of maturity that did not exist twenty years ago. Substantial knowledge and experience have been accumulated. This warrants a process of self-declaration that identifies the principal ideas behind the field. The *Digital Library Manifesto* sets the ground rules for the field and lead to the development of reference documents that 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*. in (Fox, Akscyn, Furuta, & Leggett, 1995) 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, Belkin states that a Digital Library is an institution responsible for providing at least the functionality of a traditional library in the context of distributed and networked collections of information objects (Belkin, 1999). Lesk analyses and discusses the importance of the terms ‘Digital’ and ‘Library’ in the expression ‘Digital Library’, where the former term mainly implies the existence of software for searching text, while the latter term refers to existing material that has been scanned for online access, and concludes that the research effort in the field is not usually associated with the users’ needs (Lesk, 1999). Borgman 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 (Borgman, 1999). More recently the participants to the 1st Delos Brainstorming Workshop in San Cassiano, Italy, envisage 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 (Bertino, et al., 2001). An offspring of that 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 (Ioannidis Y. , 2005; Ioannidis, et al., 2005). Likewise, in his framework for Digital Library research (Soergel, 2002), Soergel starts from three very different perspectives that different people in the community have on Digital Libraries, i.e., (i) tools to serve research, scholarship and education, (ii) a means for accessing information, and (iii) providing services primarily to individual users. He then enhances each one further and fuses them all together to obtain the main guiding principles for his vision of the field. On the other hand, Kuny and Cleveland discuss four myths about Digital Libraries (Kuny & Cleveland, 1996) and attempt to bring them down: (i) the Internet is ‘The’ Digital Library; (ii) at some point there will be a single Digital Library or a single-window view of Digital Library collections; (iii) Digital Libraries are means to provide more equitable access to content from anywhere at any time; and (iv) Digital Libraries are cheaper instruments than physical libraries. They conclude that Digital Libraries impose reinvention of the role of librarians and library models.

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

This vision of Digital Libraries seems to resonate well with the concept of ‘Information Space’ that has arisen from the field of Computer Supported Cooperative Work (CSCW). Snowdon, Churchill and Frecon have developed future visions about ‘Connected Communities’ and ‘Inhabited Information Spaces’ (Snowdon, Churchill, & Frecon, 2004), with the latter being closely related to the vision of Digital Libraries, in that ubiquitous information is a prerequisite for CSCW. In more detail, Inhabited Information Spaces are ‘spaces and places where people and digital data can meet in fruitful exchange, i.e., they are effective social workspaces where digital information can be created, explored, manipulated and exchanged’. Thus, ‘in Inhabited Information Spaces, both information and people who are using that information (viewing it, manipulating it) are represented. This supports collaborative action on objects, provides awareness of others’ ongoing activities, and offers a view of information in the context of its use'. Based on the above and according to the aforementioned DELOS vision, a Digital Library provides an Information Space that is populated by a user community and becomes an Inhabited Information Space through CSCW technology. The two fields complement each other nicely, in that one focuses on access and provision of relevant information while the other focuses on visualisation and sharing of information.

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

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

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

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

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

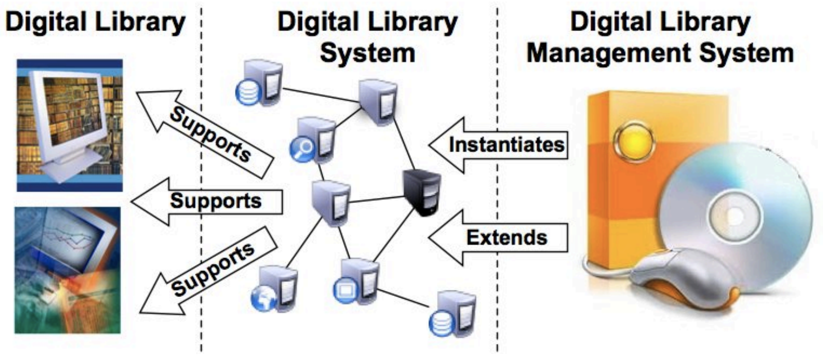

Figure I.2-1. DL, DLS and DLMS: A Three-tier Framework

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

Digital Library (DL)

A potentially virtual **organisation**, that comprehensively collects, manages and preserves for the long depth of time rich **digital content,** and offers to its targeted **user** communities specialised **functionality** on that content, of defined **quality** and according to comprehensive codified **policies**.

Digital Library System (DLS)

A deployed software system that is based on a possibly distributed **architecture** and provides all facilities 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 facilities considered fundamental for Digital Libraries and (ii) to integrate additional software offering more refined, specialised or advanced facilities.

Although the concept of Digital Library is intended to capture an abstract system that consists of both physical and virtual components, the Digital Library System and the Digital Library Management System capture concrete software systems. For every Digital Library, there is a unique Digital Library System in operation (possibly consisting of many interconnected smaller Digital Library Systems), whereas all

Digital Library Systems are based on a handful of Digital Library Management Systems.[5] The DL is thus the abstract entity that 'lives' thanks to the software system constituting the DLS and the DLMS is the software system that is conceived to support the lifecycle of one or more DLSs.

It is important to note that all these concepts underlie all types of information environment and systems, e.g., databases, hospital info systems, banking systems, the web, Wikipedia, etc. However, it is the particular characterizations given in the definitions of the previous section that distinguishes "digital libraries" from the others: the content should be rich, annotated and preserved for depth of time, the user communities should be targeted, the functionality should be specialized, the quality should be measurable and according to comprehensive policies. All of these characterizations, of course, are abstract and subject to interpretation, so they cannot lead to a precise, formal definition. Nevertheless, they offer conceptual yardsticks by which systems can be measured and mutually compared and psychological lower bounds can be established regarding the nature of digital libraries.

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

I.3 The Digital Library Universe: Main Concepts

Despite the great variety and diversity of existing digital libraries,[6] there is a small number of fundamental concepts that underlie all systems. 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.

Seven core concepts provide a foundation for digital libraries. One of them appears in the definition of Digital Library to capture the commonalities between this universe and other social arrangements: *Organisation*. Five of them appear in the definition of Digital Library to capture the features characterising this kind of Organisation and the expected service: *Content*, *User*, *Functionality*, *Quality* and *Policy*. The seventh one emerges in the definition of Digital Library System to capture the systemic features underlying the expected service: *Architecture*. All seven concepts influence the Digital Library three-tier framework, as shown in Figure I.3-1.

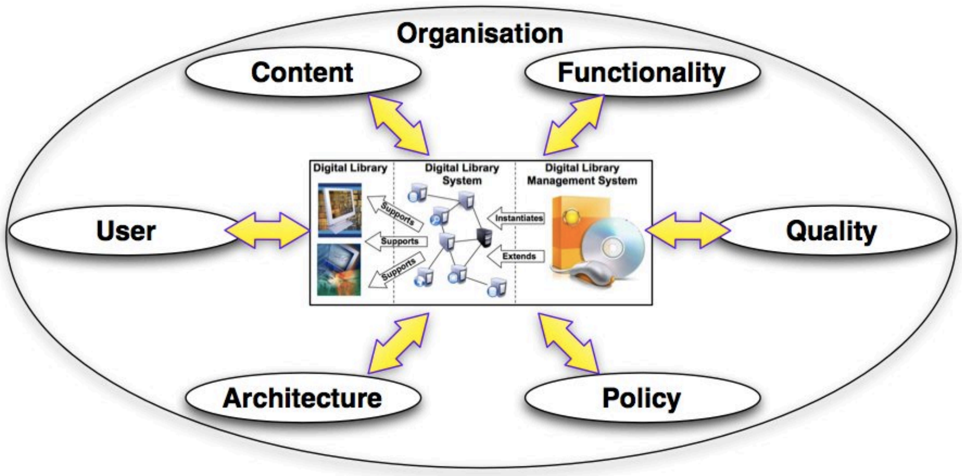

Figure I.3-1. The Digital Library Universe: Main Concepts

I.3.1 Organisation

The *Organisation* concept is surrounding the entire Digital Library universe. A Digital Library is a kind of Organisation by its own, it is a social arrangement pursuing a well defined goal (the digital library service). This concept subsumes the mission the Digital Library has been conceived for and every other aspect that is needed to define this mission and the operation of the resulting service. However, this should not be confused with “the” Organisation/Institution that decided to set up the Digital Library and drive its development although there are overlaps and dependencies between the two. It is quite easy to recognise the dependency relationship between the two, to some extent the Institution sets the scene for the Digital Library Organization, the Institution is the establisher of the Digital Library

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

Organisation and has the power to define the overall service this organisation is requested to realise. However, the Digital Library, being an Organisation by its own, has the power to control its own behaviour and evolution in the frame defined by the Institution.

This concept is fundamental to characterise the Digital Library universe because it highlights the commonalities between this universe and another one dedicated to capture organised body of people having a particular purpose.

I.3.2 Content

The *Content* concept encompasses the data and information that the Digital Library handles and makes available to its users. It is composed of a set of information objects organised in collections. Content is an umbrella concept used to aggregate all forms of information objects that a Digital Library collects, manages and delivers. It encompasses a diverse range of information objects, including primary objects, annotations and metadata.

This concept is fundamental to characterise the Digital Library universe because it captures one of the major resource these Organisations are called to manage, i.e. the data and information that is made available through it.

I.3.3 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 their ability to consume and make creative use of it to generate new information. User is an umbrella concept including all notions related to the representation and management of actor entities within a Digital Library. It encompasses such elements as the rights that actors have within the system and the profiles of the actors with characteristics that personalise the system's behaviour or represent these actors in collaborations.

This concept is fundamental to characterise the Digital Library universe because it captures the actors of the overall Organisation.

I.3.4 Functionality

The *Functionality* concept encapsulates the services that a Digital Library offers to its different users, whether individual users or user groups. While the general expectation is that Digital Libraries will be rich in functionality, the bare minimum of functions includes new information object registration, search and browse. Beyond that, the system seeks to manage the functions of the Digital Library to ensure that the overall service reflects the particular needs of the Digital Library's community of users and/or the specific requirements related to its Content.

This concept is fundamental to characterise the Digital Library universe because it captures the facilities offered by the overall Organisation.

I.3.5 Policy

The *Policy* concept represents the set or sets of conditions, rules, terms and regulations governing every single aspect of the Digital Library service including acceptable user behaviour, digital rights management, privacy and confidentiality, charges to users, and collection formation. Policies may be defined within the Digital Library or be superimposed by the Institution establishing the Digital Library, or outside of that (e.g., Policy governing our Society). The policies can be extrinsic or intrinsic policies.

Definition of new policies and re-definition of older policies, is part of the policy-related functionality that must be supported by a Digital Library.

This concept is fundamental to characterise the Digital Library universe because it captures the rules and conditions regulating the overall Organisation.

I.3.6 Quality

The *Quality* concept represents the parameters that can be used to characterise and evaluate the overall service of a Digital Library including every aspect of it, i.e. Content, User, Functionality, Policy, Quality, and Architecture. 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 quantitative and objective in nature and can be measured automatically, whereas others are qualitative and subjective in nature and can only be measured through user evaluations (e.g., focus groups).

This concept is fundamental to characterise the Digital Library universe because it captures qualitative aspects characterising the Organisation.

I.3.7 Architecture

The *Architecture* concept refers to a Digital Library System and represents a mapping of the overall service offered by a Digital Library (and characterised by Content, User, Functionality, Policy and Quality) on to hardware and software components.[7] There are two primary reasons for having Architecture as a core concept: (i) Digital Libraries are often assumed to be among the most complex and advanced forms of information systems (Fox & Marchionini, 1998); and (ii) interoperability across Digital Libraries is recognised as a major challenge. A clear architectural framework for Digital Library Systems offers ammunition in addressing both of these issues effectively.

This concept is fundamental to characterise the Digital Library universe because it captures the systemic part of the service offered by the Organisation.

The concepts populating the areas just introduced (Organisation is a special case since it subsumes all the rest) share many similar characteristics and all refer to internal entities of a Digital Library that can be sensed by the external world. Therefore, there has also been introduced a higher-level concept referring to all of these, i.e., *Resource*, which enables us to reason about the common characteristics in a consistent manner.

Figure I.3-2 puts in perspective the main concepts of the Digital Library universe. The Organisation concept surrounds and subsumes all the other concepts. Among the remaining six, two of them are independent each other, i.e., they exist independently of a specific Digital Library. These are *User*, representing the external humans or hardware interacting with the Digital Library, and *Content*, representing the material handled by the Digital Library. *Architecture*, representing the technological design on which the Digital Library System is based, represents the underlying technology that is called to implement all the rest. On top of these concepts there comes *Functionality*, primarily representing the means for connecting *User* to *Content*, i.e., all procedures, transformations, actions and interactions

[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

that bring *Content* to *User* or vice versa. Finally, operation of the Digital Library and activation of its *Functionality* are based on *Policy* and aim to achieve certain *Quality*.

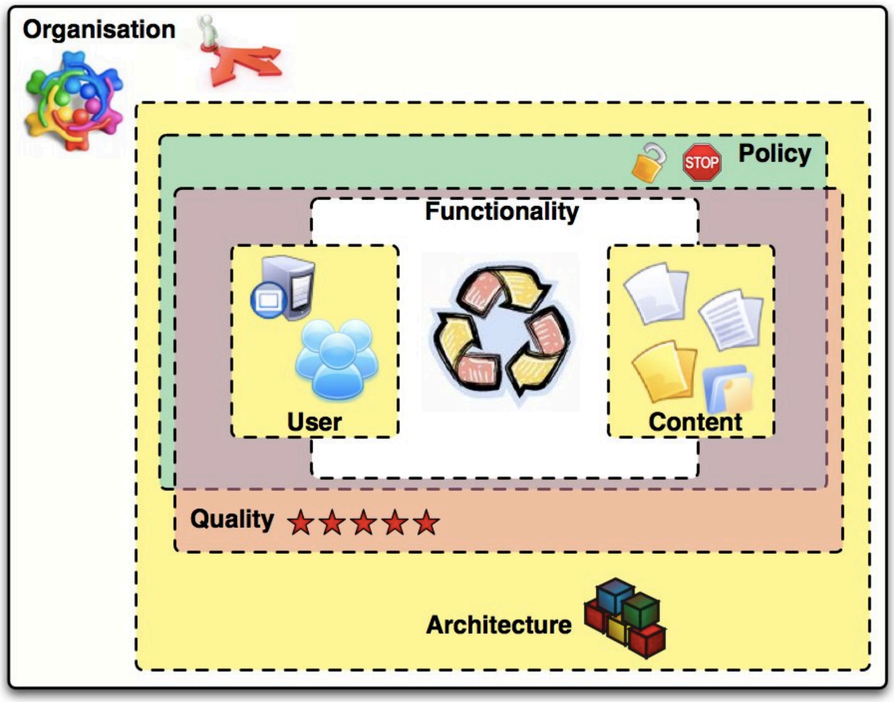

Figure I.3-2. The Digital Library Universe: The Main Concepts in Perspective

In order to describe how a Digital Library Organisation is expected to work, it is fundamental to identify which are the main roles that actors can play while interacting with the digital library systems previously identified and which are their relations with the six core concepts (Content, User, Functionality, Quality, Policy and Architecture) characterising such a kind of Organisation. These roles are discussed in the next section.

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

In order to describe the overall operation of the Digital Library Organisation and the way it is expected to deliver the service it has been established for, we envisage actors interacting with digital library playing roles in three different and complementary categories: *DL End-users*, *DL Managers* and *DL Software Developers*.

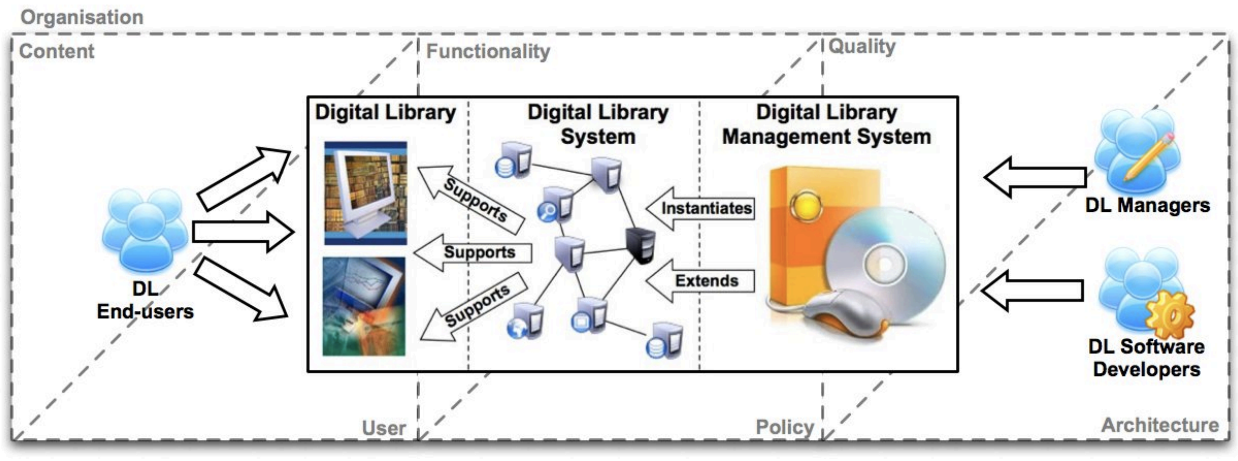

Figure I.4-1. The Main Roles of Actors versus the Three-tier Framework

As shown in Figure I.4-1, each role is primarily associated with one of the three 'systems' in the three-tier framework. The 'system' a role is associated with represents the entity that is expected to provide the actor playing such a role with the facilities needed to accomplish the mandate assigned to the role. Moreover, every actor, independently from the role he/she is playing, is expected to deal with all the foundational concepts characterising the Digital Library universe.

I.4.1 DL End-users

DL End-users exploit the overall Digital Library service for the purpose of providing, consuming, and managing the DL. They are the target clients of the service defined by the DL Organisation in terms of the Content to be managed, the User(s) to be served, the Functionality to be supported, the Policy(ies) to be put in place and the Quality to be exposed. They perceive the DL as a stateful entity serving their needs. This state of the Digital Library is a complex condition resulting from and influencing Content, User, Functionality, Policy and Quality aspects of the DL Organisation. Moreover, the state is expected to evolve during the lifetime of the Digital Library as a consequence of a series of actions and activities performed in the context of the DL Organisation as well as of external factors influencing the DL Organisation.

DL End-users may be further divided into *Content Creators*, *Content Consumers* and *Digital Librarians*.

Content Creators are the "producers" of the Digital Library Content, i.e., they take care of producing new items contributing to the Digital Library Content. Their activity is performed (*i*) through the Functionality the DL is provided with, (*ii*) in accordance with the Policies defined in the DL, and (*iii*) with the guarantee of Quality the DL declares.

Content Consumers are the "clients" of the Digital Library Content, i.e., they access and use the items in the Digital Library Content. Their activity is performed (*i*) through the Functionality the DL is provided with, (*ii*) in accordance with the Policies defined in the DL, and (*iii*) with the guarantee of Quality the DL declares.

Digital Librarians are the "curators" of the Digital Library Content, i.e. they select, organise and look after the items in the Digital Library Content. Their activity is performed (*i*) through the Functionality the DL is provided with, (*ii*) in accordance with the Policies defined in the DL, and (*iii*) with the guarantee of Quality the DL declares. Moreover, they might influence the behaviour of the overall Digital Library service by acting as mediators between the final clients of it – i.e., Content Creators and Content Consumers – and those defining and operating this service – i.e., DL Managers (cf. Section I.4.2) – by distilling and elaborating feedbacks on the DL.

I.4.2 DL Managers

DL Managers are the actors driving the overall Digital Library service. They are expected to rely on the facilities offered by the DLMS to define and operate the Digital Library and the DLS implementing it.

DL Managers may be further divided into *DL Designers* and *DL System Administrators*. The former are called to devise the overall service while the latter are called to deploy and operate the DLS implementing the planned service.

DL Designers exploit their knowledge of the application environment that a DL is called to serve in order to define, customise, and maintain the Digital Library so that it is aligned with the needs of its target DL End-users. To perform this task, the DL Designers interact with the DLMS to decide upon the characteristics the Digital Library should have in terms of (i) Content, e.g., the set of repositories, ontologies, classification schemas, information object types, metadata formats, authority files, and gazetteers that form the DL Content; (ii) User, e.g., the allowed actors, the allowed roles, the information characterising the actors; (iii) Functionality, e.g., the functional facilities to be offered, the behaviour these facilities should implement; (iv) Policy e.g., the rules and principles governing the evolution of the DL Content, the allowed actions per actor or family of actors, the exploitation of a resource; and, (v) Quality, e.g. the minimal availability of a DL Functionality, the minimal response time of a DL Functionality, the completeness and authoritativeness of the DL Content, the confidentiality of the User actions. These aspects characterise the overall Digital Library service, actually the way it is perceived by the DL End-users. 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 target users and changes in all aspects.

DL System Administrators work in tandem with DL Designers to put in place the Digital Library System implementing the planned Digital Library service. They select, deploy and manage a set of networked computers and software modules needed to fulfil the expectations that DL End-users and DL Designers have for the Digital Library. DL System Administrators perform their tasks by interacting with the DLMS and relying on the facilities these systems offer for DLS constituents identification, linking, allocation, deployment, configuration, tuning, monitoring, alerting, and any other management facility requested to manage potentially distributed software systems as DLSs are expected to be. Different DLMSs are expected to offer diverse management facilities ranging from manual installation and configuration of the computers and the software modules on the target computers to fully autonomic solutions aiming at reducing human intervention to a few corner cases.

I.4.3 DL Software Developers

DL Software Developers develop and/or customise the software components that will be used as constituents of the DLSs. They are requested to produce the software implementing every aspect of the Digital Library service ranging from the DL Content and User to Functionality, Policy and Quality. However, DL Software Developers should not start from scratch and their activity is expected to be performed by relying on the offering of a DLMS. In fact, a DLMS is a software system that is equipped with a bunch of off-the-shelf software modules implementing – to some extent – some Digital Library facilities, e.g., content repositories, users management systems, cooperative working environments, information retrieval engines, policies enforcement modules. DL Software Developers include Software Engineers and Programmers that are requested to customise and complement the set of software modules provided by the exploited DLMS as to obtain the set of software constituents needed to implement the planned Digital Library.

The three roles described above encompass the entire spectrum of actors working in the digital libraries universe. Their conceptual models of such a universe are linked together in a hierarchical way, as shown in Figure I.4-2. This hierarchy is a direct consequence of the above definitions, since DL End-users act on the Digital Library, whereas DL Managers and DL Application Developers operate on the DLS (through the mediation of a DLMS) and, consequently, on the DL as well. This inclusion relationship ensures that cooperating actors share a common vocabulary and knowledge. For instance, the DL End-user expresses requirements in terms of the DL model and, subsequently, the DL Designer understands these requirements and defines the DL accordingly.

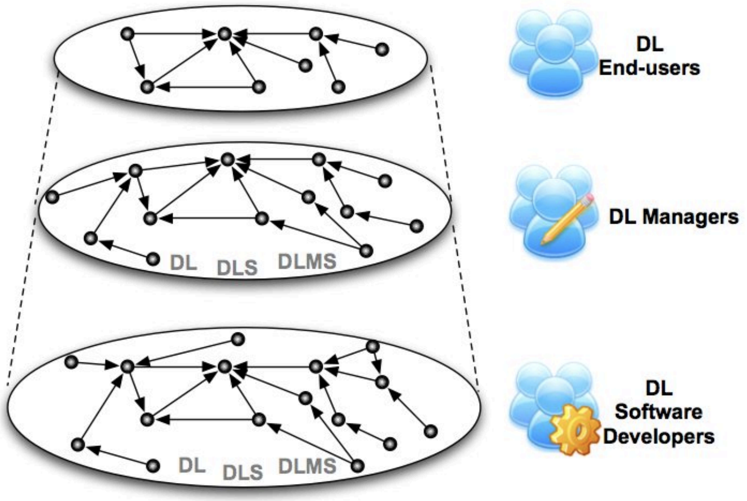

Figure I.4-2. Hierarchy of Users' Conceptual Models

Note about Librarians

The reader may be surprised that a Manifesto purporting to cover the Digital Library universe has no actor role termed **'Librarian'**. Librarians are not expected to cover a unique one of the envisaged roles, rather their activities span many of them. Among the functions envisaged for DL End-users there are provision, consumption and management of the DL content, thus Librarians acting as cataloguers and curators in the Library world and those interfacing with and supporting the users of a Library perform these activities in the DL domain. Because the DL Designer exploits her/his knowledge of the application semantic domain to define, customise and maintain the Digital Library, this role is usually covered by the chief Librarian (a.k.a. DL Manager or Director) who decides the overall service offered. The DL System Administrator role is played by the Librarian with technical skills entitling her/him to manage the DLS realising the DL service. Moreover, some Librarians might be engaged in the customisation of the software system realising the service, thus they act as DL Software Developers. Thus, even if none of the main actors is termed “Librarian”, the Reference Model captures the various activities modern Librarians are requested to perform in the digital library realm.

I.5 Digital Library Development Framework

The digital library universe is a complex world. Consequently, it is difficult to identify a single and fully-fledged model capable of capturing all the aspects needed to represent this universe independently of the scenario this model is expected to serve. One of the scenarios in which the existence of a proper model is fundamental is that leading to the development of concrete systems. This scenario is very broad and requires a comprehensive and detailed model capable to capture the peculiarities of every entity of the universe at a level of detail that allows developers to implement it in a precise manner. As a consequence, the resulting model should be reach enough to be reused in a plethora of other scenarios including teaching and systems assessment. However, such a model may be difficult to use if it is not appropriately designed, i.e., tailored to address the specific needs of the audience for which it is conceived for. For this reason, we envisage "the" model needed to capture the digital library universe and promote its implementation as a framework supporting modelling at different levels of abstraction. Figure I.5-1 depicts such a framework whose elements are: the *Reference Model*, the *Reference Architecture*, and the *Concrete Architecture*.

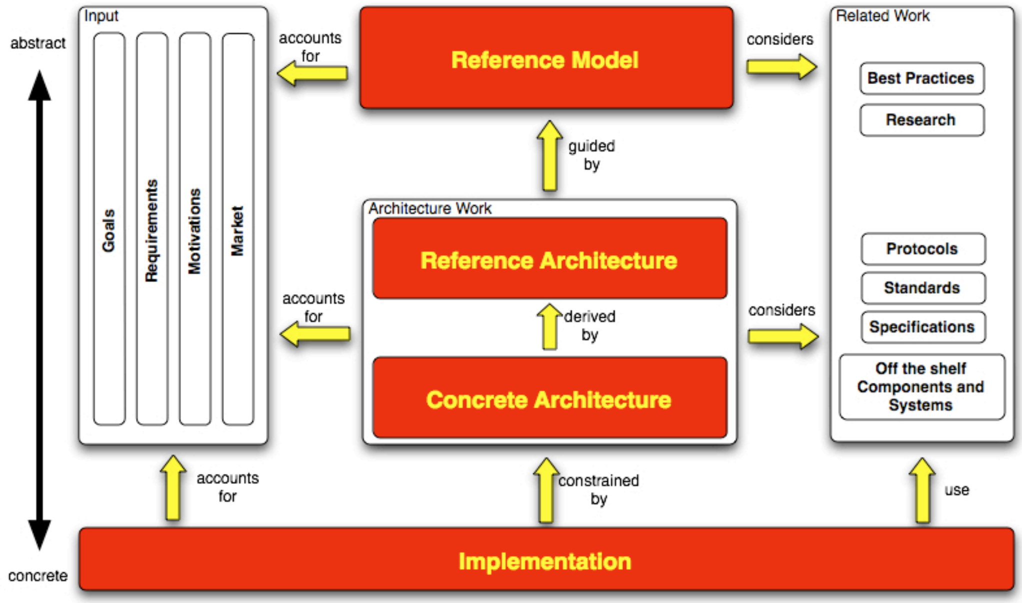

Figure I.5-1. The Digital Library Development Framework[8]

Reference Model – As stated elsewhere (MacKenzie, Laskey, McCabe, Brown, & Metz, 2006), '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 to consolidate the diversity of existing approaches into a cohesive and consistent whole, to offer a mechanism for enabling the comparison of different digital library systems, to provide a common basis for communication within the digital library community, and to help focus further advancement.

[8] This diagram was inspired by MacKenzie, M., Laskey, K., McCabe, F., Brown, P., & Metz, R. (2006). *Reference Model for Service Oriented Architecture.* OASIS.

Reference Architecture – The Reference Architecture is an architectural design pattern indicating an abstract solution that implements the concepts and relationships identified in the Reference Model. There may be more than one Reference Architecture that addresses how to design digital library systems built on the Reference Model. For example, we might have one Reference Architecture for DLSs supporting DLs constructed by federating local resources and multiple organisations, and another one for personal DLs or for specialised applications.

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

I.6 Digital Library Manifesto: Concluding Remarks

The goal of *The Digital Library Manifesto* has been to set the foundations and identify the entities of discourse within the universe of digital libraries. It has introduced the relationships 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., organisation, 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, manager and software developer. Finally, it has described the DL Development Frameworks that capture the above systems at different levels of abstraction, i.e., the Reference Model and the Reference and Concrete Architectures.

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

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

PART II The Digital Library Reference Model in a Nutshell

II.1 Introduction

Despite the large number of 'systems' that are called 'digital libraries' (Fox, Akscyn, Furuta, & Leggett, 1995; Kuny & Cleveland, 1996; Fox & Marchionini, 1998; Borgman, 1999; Ioannidis Y. , 2005; Ioannidis, et al., 2005) – where 'system' is intended as a set of interconnected things forming a whole – there are no real foundations for them. This limits the growth of the digital library field, as it is really difficult to systematise activities for evaluating and comparing digital library systems, for teaching about 'digital libraries' and for performing further and focused research. The same holds for system design and development, and for promoting sustainable approaches and solutions that aim at maximising the reuse of existing knowledge and assets, and at properly addressing community needs.

In January 2005, the DELOS Network of Excellence on Digital Libraries[9] initiated a process to draft a *reference model* for digital libraries as a necessary step towards a more systematic approach on digital libraries research. In this context, a reference model is an abstract framework that captures the significant entities of some universe and relationships between them and aids in the development of consistent standards and/or specifications supporting that universe (MacKenzie, Laskey, McCabe, Brown, & Metz, 2006). The route towards reaching this objective, summarised below, has been traced in the Digital Library Manifesto (Part I of this volume).

II.1.1 The Digital Library Manifesto in Brief

It is commonly understood that the Digital Library universe is a complex and multifaceted domain that cannot be captured by a single definition. The Digital Library Manifesto organises the pieces constituting the puzzle into a single framework (Figure II.1.1).

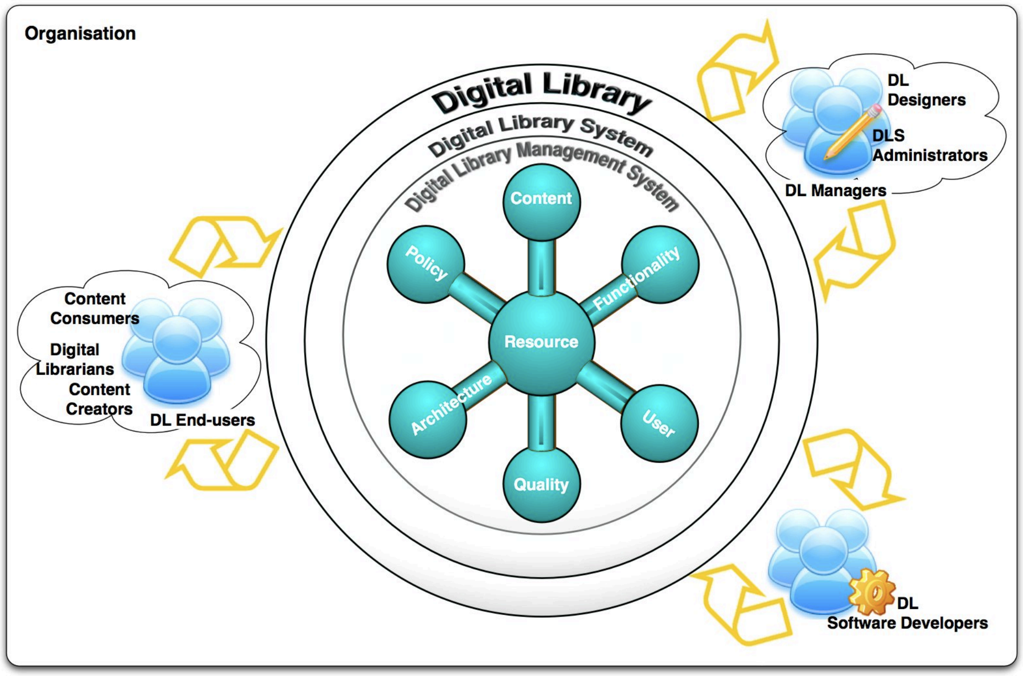

Figure II.1.1. The Digital Library Universe

[9] www.delos.info

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

1. *Digital Library* (*DL*) – the final 'system' actually perceived by the end-users as being the digital library;
2. *Digital Library System* (*DLS*) – the deployed and running software system that implements the DL facilities;
3. *Digital Library Management System* (*DLMS*) – the generic software system that supports the production and administration of DLSs and the integration of additional software offering more refined, specialised or advanced facilities.

The *Manifesto* also identifies the **seven core concepts** characterising the digital library universe. These corresponds to 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:

1. Organization – represents the social arrangement characterising the expected DL service. It is a super domain that comprises the remaining six domains that actually characterise the service;
2. Content – represents the information managed;
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 of the whole;
6. *Quality* – represents the aspects needed to consider digital library systems from a quality point of view;
7. *Architecture* – represents the software (and hardware) constituents concretely realising the whole.

Unifying the above is an 8th domain

8. *Resource* – captures generic characteristics that are common to the other specialised domains.

Another contribution of the *Manifesto* is recognising the existence of ***three categories of actors*** fundamental to characterise the operation of the Digital Library service. In particular:

- The *DL End-Users* are the ultimate clients the Digital Library is going to serve. They are further divided in *(i) Content Creators* – the "producers" of the DL Content; *(ii) Content Consumers* – the "clients" of the DL Content; and *(iii) Digital Librarians* – the "curators" of the DL Content.
- The *DL Managers* are the "drivers" of the Digital Library service, i.e. the actors requested to put in place the expected service. They are further divided in *(i) DL Designers* – the actors requested to characterise the DL service before its being deployed; and *(ii) DL System Administrators* – the actors requested to deploy the DLS needed to implement the DL Designers plan.
- The *DL Software Developers* are the implementers of the software parts needed to realise the Digital Library service.

Furthermore, 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 domains constitute the complementary perspectives from which interested players are allowed to see each layer. Thus, the settings established by the Manifesto are rich because they aim at accommodating all the various needs. However, they are highly modular and can therefore be easily adapted to capture the needs arising in specific application contexts.

Finally, the *Manifesto* gives reason for proceeding with ***different levels of abstraction*** while laying down the complete framework. These different levels of abstraction, which lead conceptually from the modelling to the implementation, are captured in Figure II.1.2 where the core role of the *Reference Model* is illustrated; all the other elements constituting the envisaged DL service supporting technology development methodology chain start from here. It drives the definition of any *Reference Architecture* that proposes an optimal architectural pattern for a specific class of DLMS supporting a class of DLS characterised by similar goals, motivations and requirements. *Concrete Architectures* are obtained by replacing the mechanisms envisaged in the Reference Architectures with concrete standards and specifications. Finally, *Implementations*, i.e., the concrete realisation of the DLMS supporting a particular class of DLS, are instances of Concrete Architectures deployed on particular machines. The definition of the Reference Model has thus also to be seen as a necessary starting point towards the introduction of all these other framework elements, which, once adopted and followed by the community, will largely enhance the digital library development model and the interoperability among systems.

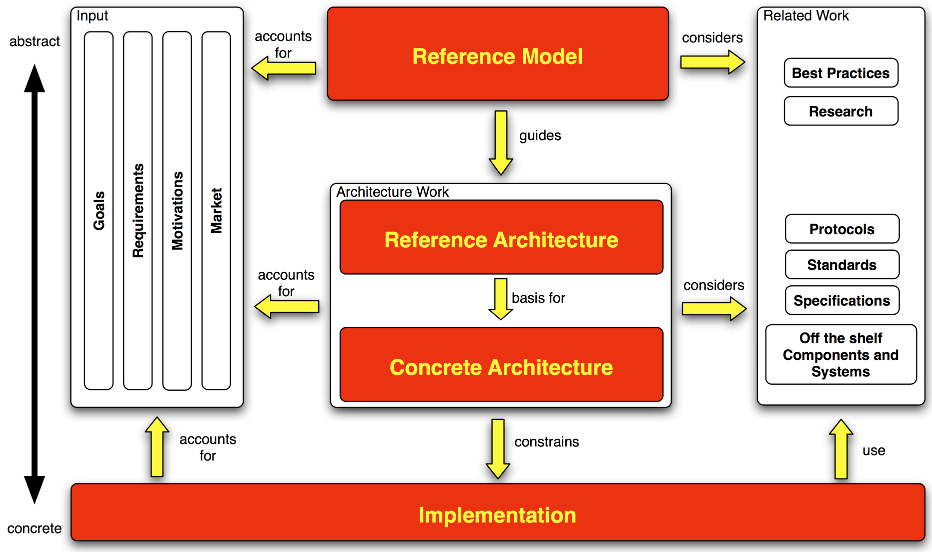

Figure II.1.2. The Reference Model as the Core of the Development Framework

The rest of Part II of this volume provides an overview of the Digital Library Reference Model by illustrating the constituent concepts and relationships. It is structured as follows. The present section is completed by information that sets the stage for the rest, e.g., the background material necessary to understand the graphical and notational conventions. Section II.2 introduces the constituent domains of the model, highlighting the main concepts and relationships characterising the domain model rationale. Section II.3 discusses possible exploitations of such a model with respect to interoperability and preservation. Section II.4 briefly investigates related work on models for digital libraries and domains. Finally, Section II.5 provides concluding remarks.

II.1.2 Guide to Using the Reference Model

A Reference Model is a conceptual framework that aims at capturing significant entities and their relationships in a certain universe with the goal of developing more concrete models of it. In order to express such a model a modelling language is needed. There exist a plethora of modelling languages. These range from the human language to formal languages borrowed from various application domains and characterised by various types of expressing power and other interesting features, such as Entity-Relationship (Chen, 1976), UML (Booch, Rumbaugh, & Jacobson, 2005) and Description Logic (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2002), to cite just a few. However, in this document **Concept Maps** have been used because of their simplicity and immediacy.

II.1.2.1 Concept Maps

Concept maps are graphs that organise and represent knowledge (Novak & Gowin, 1984; Novak & Cañas, 2008) in terms of concepts (entities) and relationships between concepts to form propositions. Concepts are used to represent independent objects capturing regularity in events or objects, or records of events or objects. Propositions are statements about objects or events in the universe and involve two or more concepts connected using linking terms that form a meaningful statement. In the graphical representation, concepts are represented in circles or boxes, propositions (proposition connectors) are represented as (directed) lines connecting concepts, and linking relationship are represented as labels. Figure II.1.3 gives a self-serving example of a concept map, showing the structure of concept maps and illustrating their main characteristics.

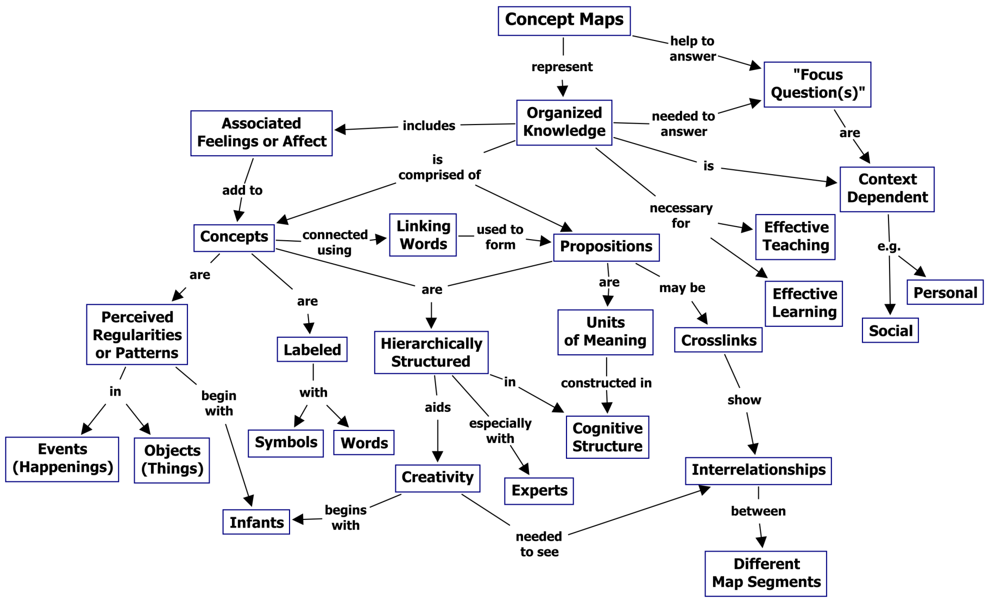

Figure II.1.3. A Concept Map showing the Key Features of Concept Maps[10]

II.1.2.2 Notational Conventions

In the following, terms expressing *concepts* are in **bold** at their first occurrence in the document and in *italic* in the rest of the document. Terms expressing *relationships* are in <*italic with angle brackets*> whenever they occur in the document.

Note about Enterprise Architecture

This Reference Model and the 'Enterprise Architecture' frameworks play a similar role. The aim of the Enterprise Architecture practice is to model the relationships between the business and the technology

[10] Figure taken from Novak, J. D., & Cañas, A. J. (2008). *The Theory Underlying Concept Maps and How to Construct and Use Them.* Institute for Human and Machine Cognition. Florida Institute for Human and Machine Cognition.

in such a way that this information can be used to support decisions enterprise wide, e.g., revising the business processes or changing the software systems supporting certain processes. Thus, the Enterprise Architecture must be compared with the whole Reference Model activity while considering the Enterprise Architecture as a decision support process that needs a model capturing a large amount of information. Because of the breadth of information to be covered, both models recognise the need to have a means of categorising this information. The best-known Enterprise Architecture framework was devised by Zachman (Zachman, 1987). This framework defines:

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

(b) different views that serve the needs of different 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.

II.2 The Constituent Domains

As outlined in the previous section, the digital library universe is complex and multifaceted. Figure II.2.1 presents an organisation of the concepts of this universe into a hierarchy of *domains*, i.e., named groups of concepts and relations, each modelling a certain aspect of the systems of the universe[11]. Domains may rely on each other and constitute orthogonal areas intended to capture the different aspects of the whole.

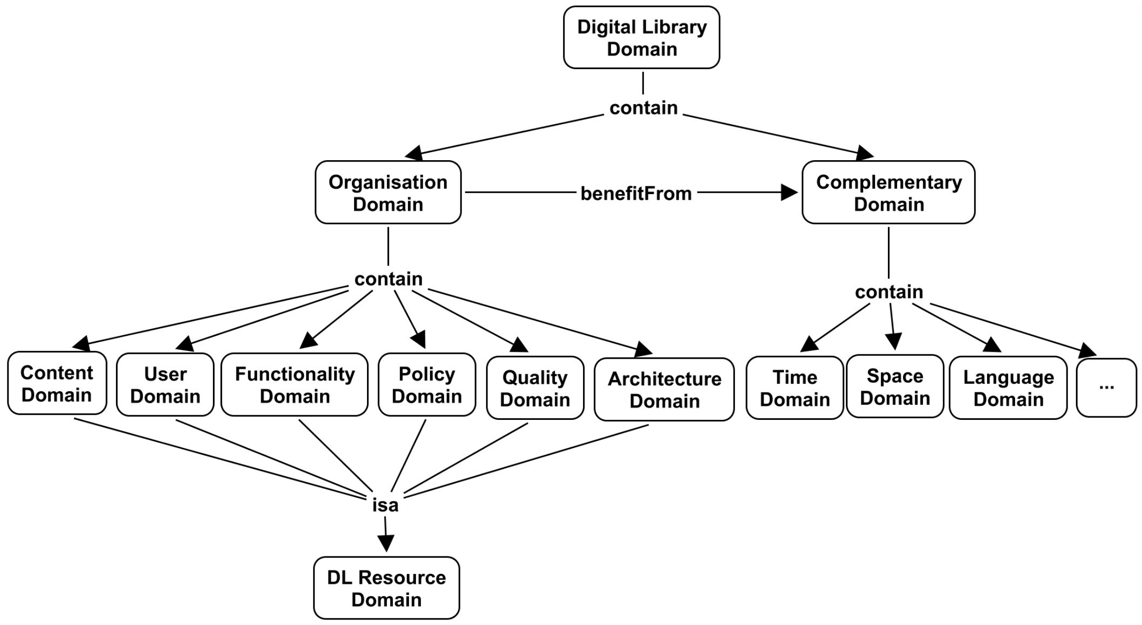

Figure II.2.1. DL Domains Hierarchy Concept Map

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

The *Organisation Domain* stems from the Organisation core concept and it is conceived to represent the main settings for characterising the DL service, the aspects that are specific to the digital library universe. It contains the following sub-domains, in full correspondence with the remaining core concepts identified in the Digital Library Manifesto:

- **Content Domain** (cf. Section II.2.2);
- ***User Domain*** (cf. Section II.2.3);
- ***Functionality Domain*** (cf. Section II.2.4);
- ***Policy Domain*** (cf. Section II.2.5);
- ***Quality Domain*** (cf. Section II.2.6);
- ***Architecture Domain*** (cf. Section II.2.7).

[11] In this context, domains play a role similar to that of UML packages and XML namespaces in their respective application areas.

Each of such domains focuses on a particular aspect characterising the digital library universe. However, independently from the specific aspect each domain is dedicated to, there are some commonalities that these aspects share and these have been captured by the *DL Resource Domain*, described in Section II.2.1.

The *Complementary Domain* contains all the other domains, which, although they do not constitute the focus of the digital libraries and can be inherited from existing models, are nevertheless needed to represent the DL service. This domain 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 domains such as:

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

Each of the 'systems' envisaged in the digital library universe – DL, DLS, and DLMS – is modelled by entities and relationships captured by these domains at different levels of abstraction (cf. Figure II.2.2).

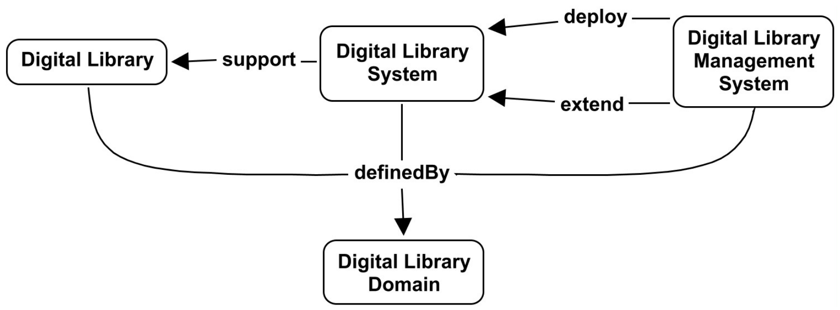

Figure II.2.2. Digital Library Domain and Systems Concept Map

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

II.2.1 DL Resource Domain

Being the domain dedicated to capture the commonalities that all the 'first class elements' of the digital library universe have, the *DL Resource Domain* captures all entities and relationships that are managed in every "digital library". The most general concept of the *DL Resource Domain* is ***Resource***, which captures the characteristics of any Digital Library entity.[12] Instances of the concept of *Resource* in the

[12] 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. The Web resource notion has evolved during the Web's history from the early conception of document or file to the current abstract definition that covers any entity that

Digital Library universe are *Information Objects* in all their forms, *Actors*, *Functions*, *Policies*, *Quality Parameters* and *Architectural Components*. These instantiates the main concepts in their respective domain, thus every *Domain* consists of *Resources*, and *Resources* are the building blocks of all the *Digital Library Domains* (Figure II.2.3).

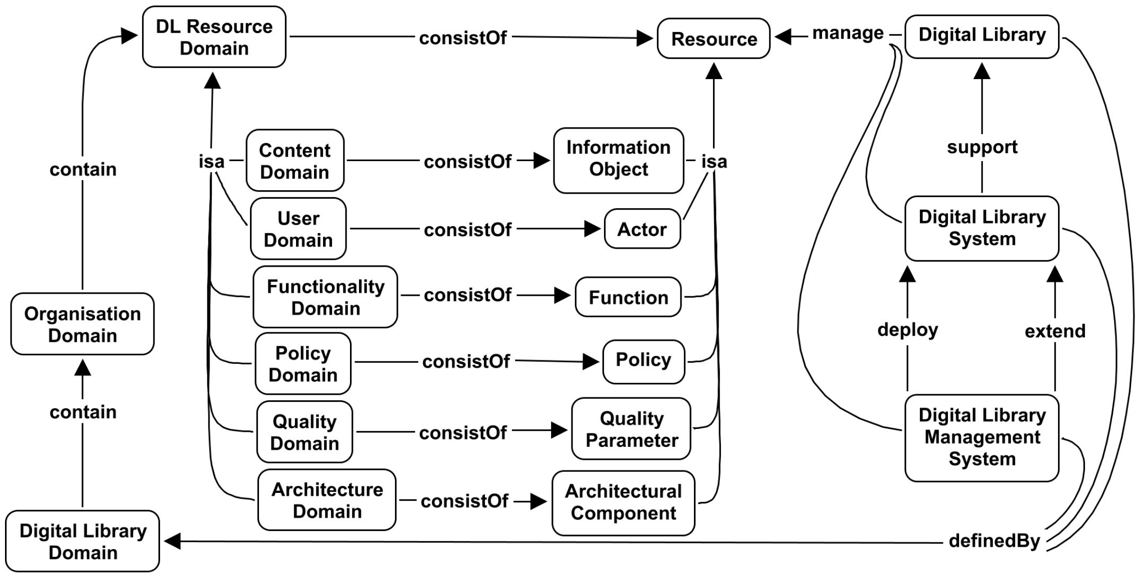

Figure II.2.3. DL Resource Domain Concept Map: Resource Instances

All the different types of *Resources* share many characteristics and ways in which they can be related to other Resources (Figure II.2.4). Each *Resource* is:

- identified by a **Resource Identifier** (<identifiedBy>);
- arranged or laid out according to a ***Resource Format*** (<*hasFormat*>) – such a format may be drawn from an Ontology to guarantee a uniform interpretation; it can be arbitrarily complex and structured, because *Resources* may be composed of smaller *Resources* (<*hasPart*>) and linked to other *Resources* (<*associatedWith*>);
- characterised by various *Quality Parameters,* each capturing how the resource performs with respect to some attribute (<*hasQuality*>);
- regulated by *Policies* (<*regulatedBy*>) governing every aspect of its lifetime;
- expressed by (<*expressedBy*>) an *Information Object* (such as a *Policy* set down in a text or a flowchart); and
- described by or commented on by an *Information Object*, especially by those dedicated to record *Metadata* (<*hasMetadata*>), *Annotations* (<*hasAnnotation*>), *Context* (<hasContext>) or *Provenance* (<hasProvenance>).

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*. Every Resource Set is characterised by an intension (<*hasIntension*>) and an extension (<*hasExtension*>). The former is a criterion underlying the grouping and corresponds to a *Query*, i.e.,

can be identified, named or addressed in the Web. This novel understanding fits very well with the meaning assumed by the same term in the Digital Library universe.

every *Query* identifies a set of *Resources*. 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* matching the intension, i.e., the set of Resources belonging to (<*belongTo*>) the *Resource Set*. These characteristics are implemented differently in diverse systems, leading to scenarios ranging from static to highly dynamic, e.g., (Candela, Castelli, & Pagano, 2003).

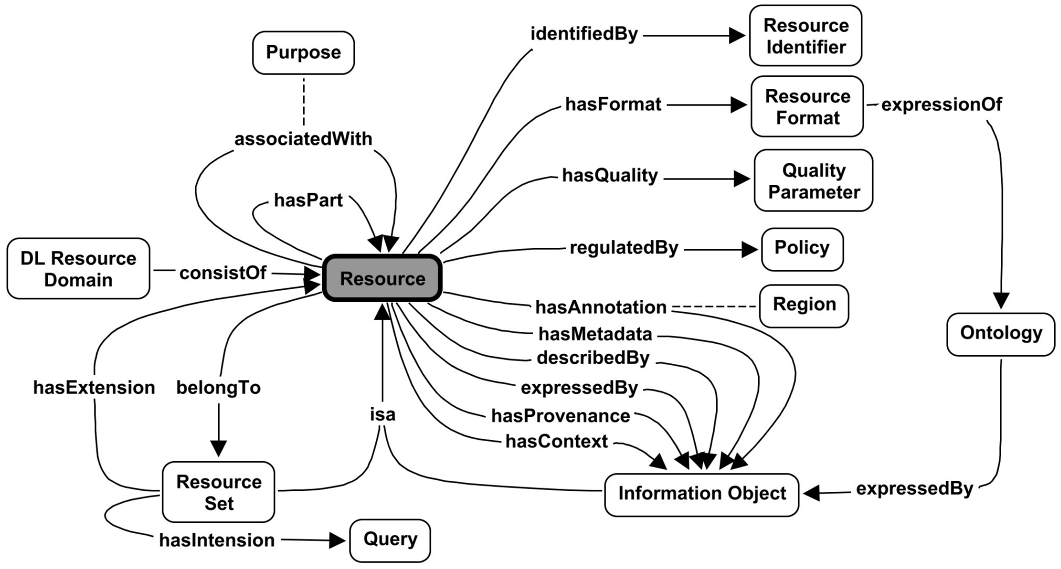

Figure II.2.4. DL Resource Domain Concept Map[13]

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

II.2.2 Content Domain

The *Content Domain* represents all the entities managed by the Digital Library 'systems' to satisfy the information needs of their users. The most general concept in the *Content Domain* is ***Information Object*** (Figure II.2.5), which is a *Resource.* An *Information Object* represents any unit of information such as 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*.

As an *Information Object* is a *Resource*, it inherits all its features (Figure II.2.5).

Information Objects can be grouped into ***Collections*** (<*belongTo*>), i.e., special type of *Resources* which are themselves *Information Objects* and inherit all *Information Objects*' features, e.g., they can be annotated. Collections are a specialisation of the *Resource Set* concept. They are characterised by an

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

intension (<*hasIntension*>) – the *Query* capturing the criterion underlying the group – and an extension (<*hasExtension*>) – the set of *Information Objects* matching the intension.

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

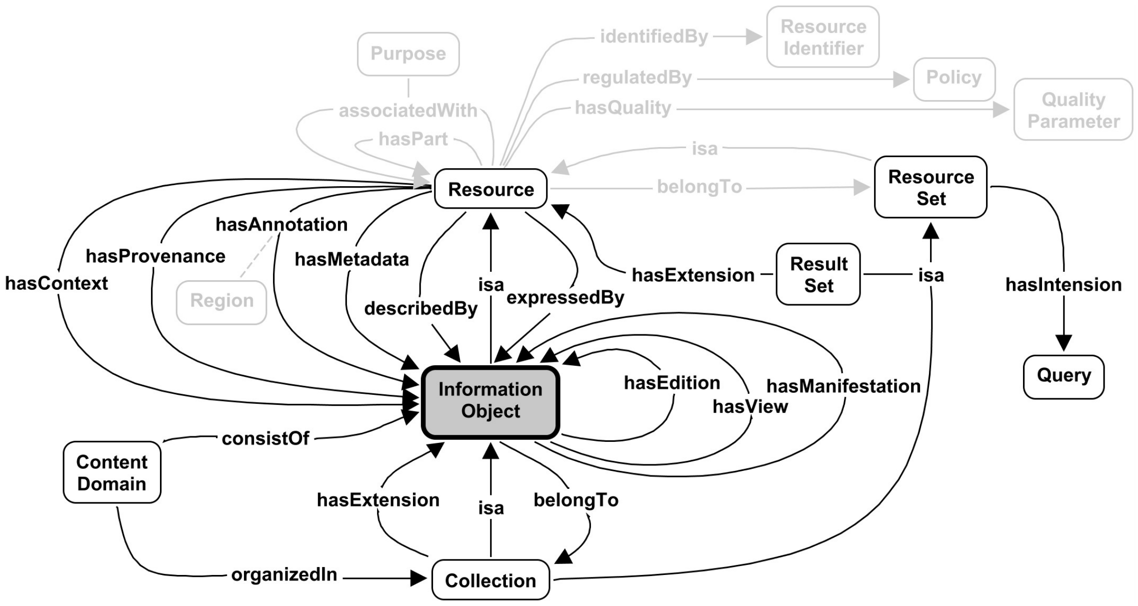

Figure II.2.5. Content Domain Concept Map

Information Objects can acquire specialisations depending on various aspects. All of them are expected to be captured by relying on relations between Information Objects.

One of these aspects is the level of abstraction at which they are specified. This leads to an abstract *Information object by level of abstraction* concept[14]*,* which is a container or placeholder to be specialised using any of several models. For example, the IFLA FRBR model (Madison, et al., 2009) distinguishes:

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

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

[14] This abstract concept is not depicted in the Content Domain Concept Map as well as it is not further defined in the rest of this document. It has been introduced here for the sake of clarifying the relationship between the single notion of *Information Object* captured by this model and the multiple notions captured by other models. In this Reference Model the multiple notions are captured by relations, this modelling style reflects a basic intuition that an Information Object is not born as a specific type but becomes such by virtue of playing a certain role in relation to other information objects.

Information objects can also be specialised by the predominant role they play in their relationship to other objects; the class *Information object by relationship* is the abstract conceptual container[15] for the classes these objects give rise to, namely:

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

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

Finally, various types of *Information Objects* can be distinguished although all of them are instances of the same concept. Possible dimensions are:

- By the type of representation or encoding:
  - *Information Objects* encoded in some natural form directly interpretable by human, text in natural language, images, sounds, etc.
  - *Information Objects* encoded in a formal structure, such as database tables, formal entity-relationship statements, ontologies in formal terms.
- By the relationship to real world objects:
  - Born digital, e.g., information object such as a born digital text or digital camera images, which are the real world objects themselves and do not correspond to any other real world objects.
  - Information objects produced by digitisation of non-digital information objects, such as digitised versions of ancient manuscripts;
  - Information object representing metadata, such as the descriptive information of the Mona Lisa, describing real-world object, whether the latter one digital or not, or represented in the DL or not.

[15] This abstract concept is not depicted in the Content Domain Concept Map as well as it is not further defined in the rest of this document. It has been introduced here for the sake of clarifying the relationship between the single notion of *Information Object* captured by this model and the multiple notions captured by other models. In this Reference Model the multiple notions are captured by relations, this modelling style reflects a basic intuition that an Information Object is not born as a specific type but becomes such by virtue of playing a certain role in relation to other information objects.

II.2.3 User Domain

The *User Domain* represents all the entities that interact with any Digital Library 'system', i.e., humans and inanimate entities such as software programs or physical instruments. Exemplars of inanimate entities include a subscription service offered by a university to its students, which provides access to the contents of an external Digital Library, or another Digital Library.

Inclusion of hardware and software into the potential users of digital libraries is a major deviation from other Digital Library models (Borbinha, et al., 2005) and reflects a broader concept of 'digital library'. To capture these extended semantics, we use the concept of ***Actor*** (Figure II.2.6) as the dominant concept in this domain.

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

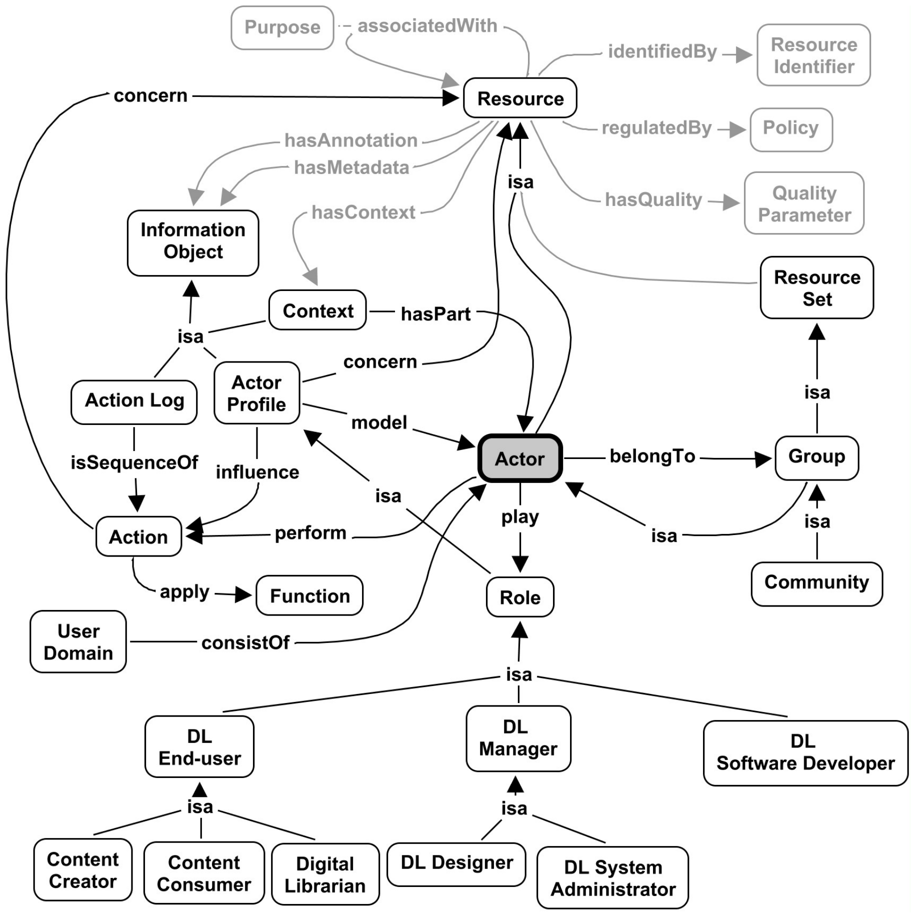

Figure II.2.6. User Domain Concept Map

An ***Actor Profile*** is used to model (<*model*>) an *Actor*. Every *Actor* interacts (<*perform*>) with the Digital Library, Digital Library System or Digital Library Management System by performing certain ***Action(s)***.

The ***Actor Profile*** is an *Information Object* that concern (<*concern*>) *Resources* and essentially models an *Actor* by capturing a large variety of the *Actor*'s potential characteristics. It allows the *Actor* to interact with the 'system' as well as with other *Actors* in a personalised, customised way. Not only does it serve as a representation of *Actor* in the system but also essentially captures the *Policies* and *Roles* that govern which *Functions* are allowed on which *Resources* by the *Actor*. For example, a particular instance of *Actor* may be entitled to *Search* within particular *Collections* and to *Collaborate* with particular other *Actors* (cf. Section II.2.4). The characteristics captured in an *Actor Profile* vary depending on the type of

Actor, i.e., human or non-human, and may includes: demographic information (e.g., age, residence or location for humans and operating system, web server edition for software components), educational information (e.g., for humans highest degree achieved, field of study), 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, a conception that is also a significant deviation from traditional approaches, where there are typically strong dependencies between *Roles* and *Actors*, an *Actor* can typically play one *Role*. Among *Actor Roles*, important categories are ***End-user***, ***DL Manager***, and ***DL Software Developer*** (Section I.4). Each of these *roles* plays a complementary activity along the 'system' lifetime. *End-user* exploits DL facilities for providing, consuming and managing DL content. It is further subdivided into the concepts of ***Content Creator***, ***Content Consumer*** and ***Digital Librarian***, each of which usually has a different perspective on the Digital Library. For instance, a *Content Creator* may be a person that creates and inserts his own documents in the Digital Library or an external program that automatically converts documents to digital form and uploads them to the Digital Library. *Actors* in the role of *DL Manager* exploit DLMS facilities to define, customise and maintain the DL service. It is further subdivided in ***DL Designers*** – they define, customise and maintain the DL service – and ***DL System Administrators*** – they exploit DLMS facilities to create and operate the DLS realising the envisaged DL service. Finally, *DL Software 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 (Borbinha, et al., 2005).

Finally, an *Actor* may be part of (<*belongTo*>) a ***Group***. A *Group* represents a set of *Actors* 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 (some of) the characteristics from the *Group*, such as interests and *Policies*, 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 several 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 one of the richest and most open-ended dimensions of the world of digital libraries, as it captures all processing that can occur on *Resources* and actions that can be observed by *Actors* in a Digital Library, Digital Library System or Digital Library Management System. The most general functionality concept is ***Function*** (Figure II.2.7), i.e., a particular processing task that can be realised on a *Resource* or *Resource Set* as the result of an activity of a particular *Actor*. It is worth noting that this description of a *Function* is based on the generalised concepts of *Actor*, capturing not only human users but also inanimate entities, and of *Resource*, representing all entities involved in or influenced by a Digital Library, Digital Library System or Digital Library Management System. Hence, this description lends a fresh perspective to the *Functionality* of this domain. For instance, not only can a human *Actor Search* the contents in a digital library, i.e., *Information Objects*, but also for other *Actors*; a program can *Search* for offered *Functions*, and so forth.

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

Because of the broad scope of the *Function* concept, it is not feasible to enumerate and predict all the different types and 'flavours' of *Functions* that may be included in a Digital Library, Digital Library System or Digital Library Management System. Each one may have its own set of *Functions* depending on its objectives or its intended *Actors*. Therefore, the *Function* concept is specialised into five sub-concepts that still represent quite general classes of activities (Figure II.2.7).

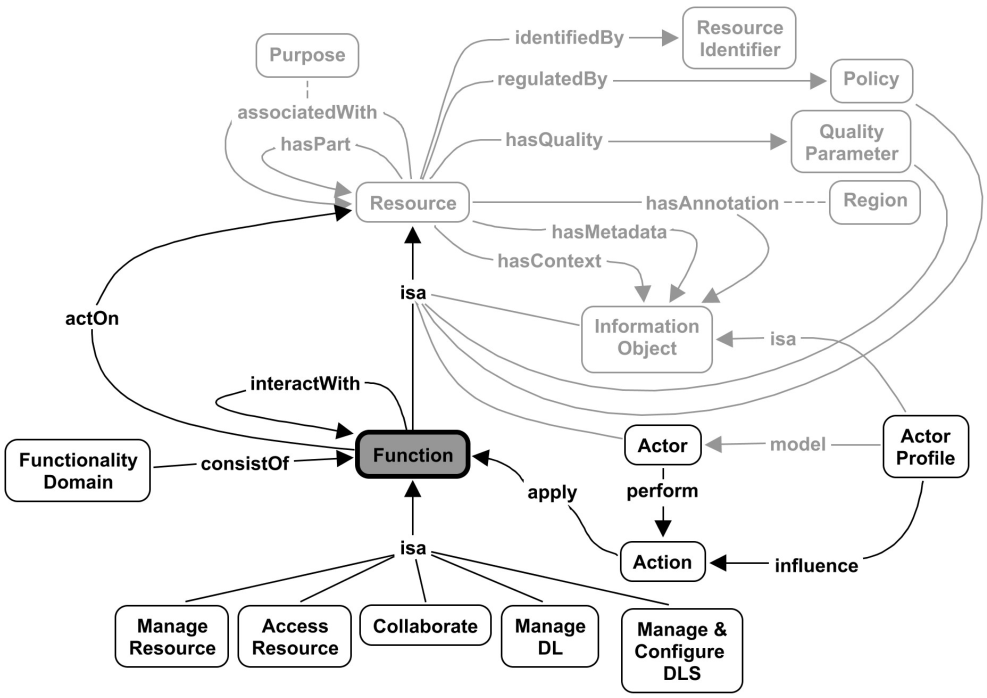

Figure II.2.7. Functionality Domain Concept Map

The first three types of *Function*s (*Manage Resource*, *Access Resource*, *Collaborate*) accommodate activities related to the prime actions, which are performed by the digital library *Actors* – namely *End-user*.

Manage Resource includes all activities related to creating new *Resources* and making them available through the DL, deleting old *Resources* from it, and updating existing ones. General management *Functions* that are applicable on all *Resources* include the creation, submission, withdrawal, update, preservation, validation and annotation (Figure II.2.8). In addition to these general functions, other Functions result when dealing with specific kind of Resources, e.g. Information Objects, Actors, Policy.

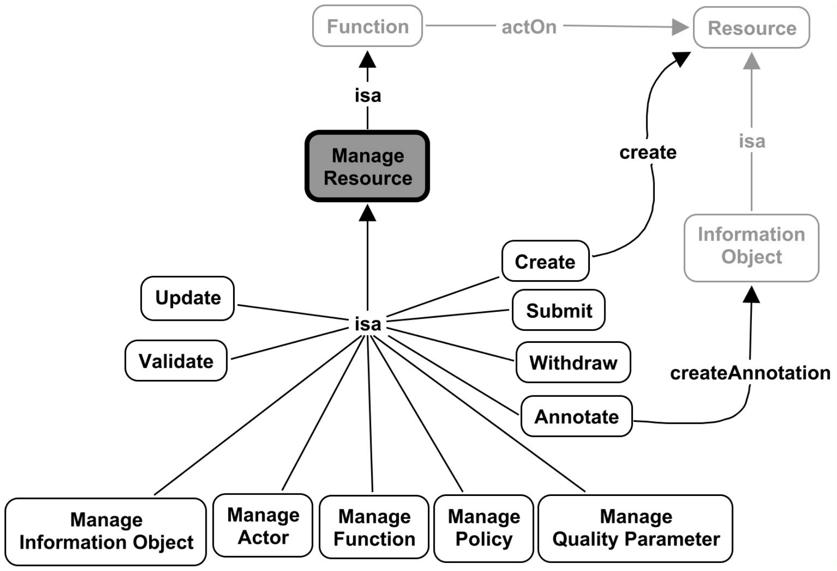

Figure II.2.8. Functionality Domain Concept Map: Manage Resource Functions

Because of their basic role, two of the *Manage Resource Functions* are worthy to be detailed. The are the *Manage Information Object* and *Manage Actor*.

Manage Information Object (Figure II.2.9) is the family of *Manage Resource Functions* conceived to capture those dedicated to *Information Objects*. This family contains Functions supporting authoring and dissemination as well as a rich array of actions dedicated to Information Object processing.

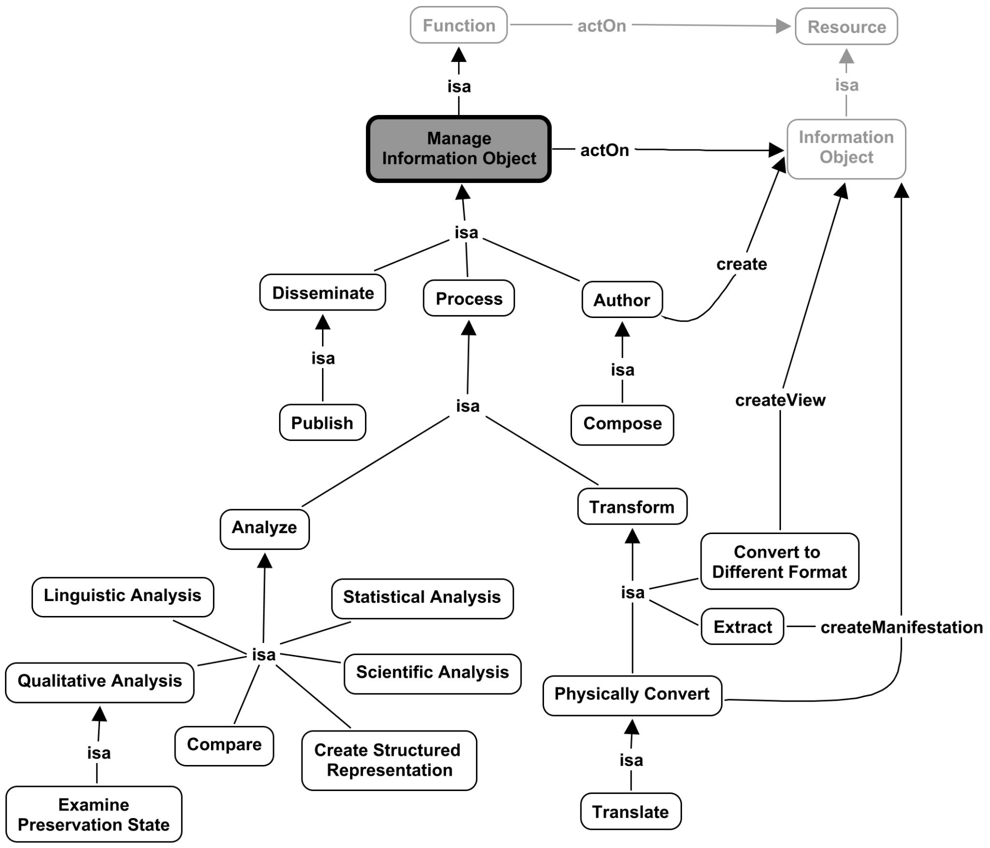

Figure II.2.9. Functionality Domain Concept Map: Manage Information Object Functions

Manage Actor is the family of *Manage Resource Functions* conceived to capture those *Functions* necessary for the management of individual *Actors*, including their registration or subscription, their login and profiling.

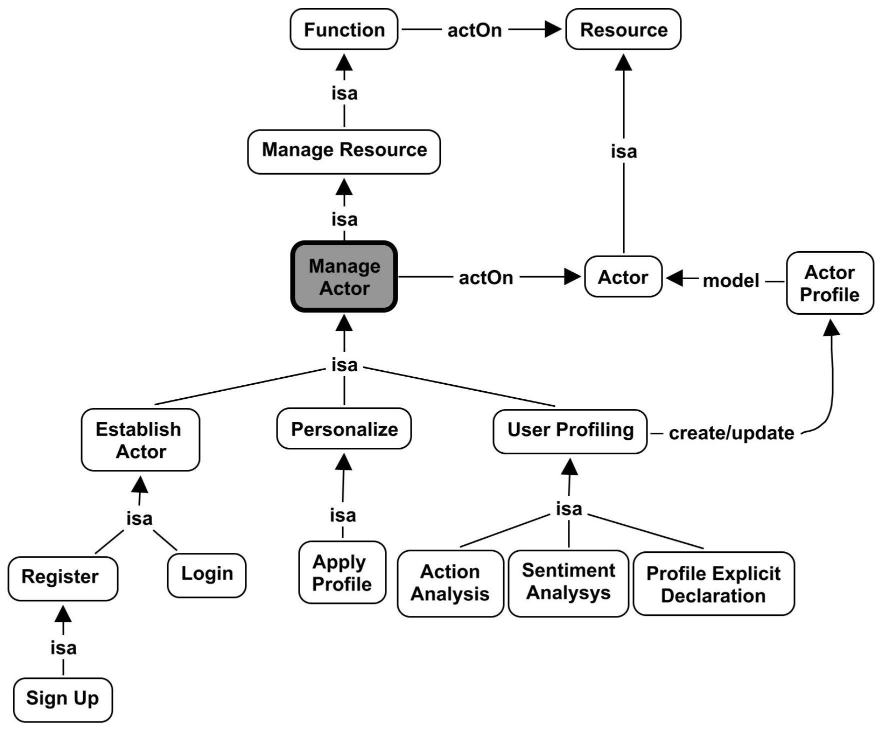

Figure II.2.10. Functionality Domain Concept Map: Manage Actor Functions

The second type of prime action expected to be performed by *End-user* deals with accessing the digital library offering. ***Access Resource*** encompasses all activities related to requesting, locating, retrieving, browsing, and representing *Resources* (Figure II.2.11). The key characteristic of the *Access Resource* concept is that it represents *Functions* that do not modify the Digital Library (DLS and DLMS as well) but identify *Resources* to be sensed by *Actors* or possibly further exploited by other *Functions*. Hence, the central *Access Resource* function is *Discover*, which acts on *Resource Sets* to retrieve desired *Resources*.

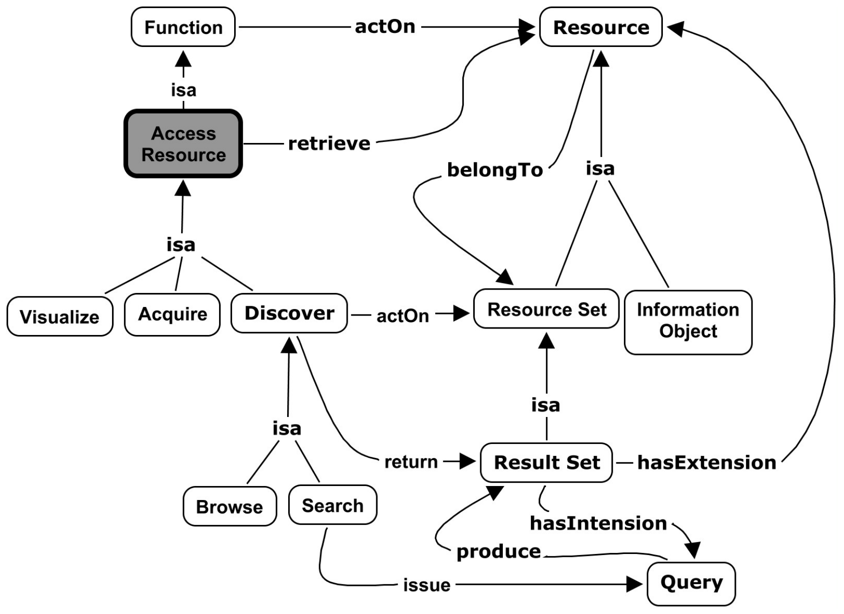

Figure II.2.11. Functionality Domain Concept Map: Access Resource Functions

The third type of prime action expected to be performed by *End-user* deals with conceiving the digital library service as a collaborative working environment. ***Collaborate*** is the family of *Functions* capturing all activities that allow multiple *Actors* to work together on top of a DL to achieve a common goal. It explicitly captures the main *Functions* falling in this domain including basic facilities, e.g., collaborative authoring via *Author Collaboratively*, and facilities promoting the collaboration, e.g. co-workers discovery via *Find Collaborator*.

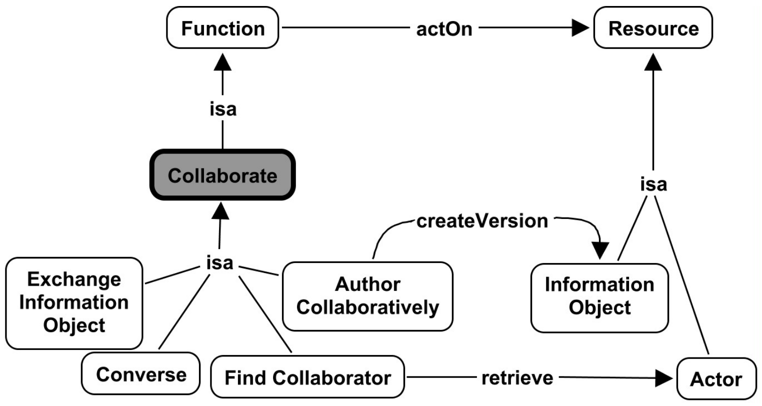

Figure II.2.12. Functionality Domain Concept Map: Collaborate Functions

The remaining two specialisations of the *Function* concept encompass all activities related to the 'system' as a whole and its management. These specialisation are *Manage DL* and *Manage & Configure DLS*. They are oriented to support the activities of the *Actors* requested to operate the digital library service – mainly, *Digital Librarians* and *DL Managers* as well as *DL Software Developers* – and are expected to be supported by (i) the Digital Library – for day-to-day management (Manage DL) and (ii) *Digital Library Management System* – for long-term management (*Manage & Configure DLS*).

Manage DL (Figure II.2.13) includes a wide variety of *Functions* that support the day-to-day management of the overall DL service. Because of this, it includes facilities for revising every aspect of the service from *Content* (e.g., *Collection* management) and *User* (e.g., *Group* management) -related characteristics to *Functionality*, *Policy* and *Quality* ones. These *Functions* are mainly associated with the role of *Digital Librarian*. However, part of them, can be associated with the role of *DL Designer*.

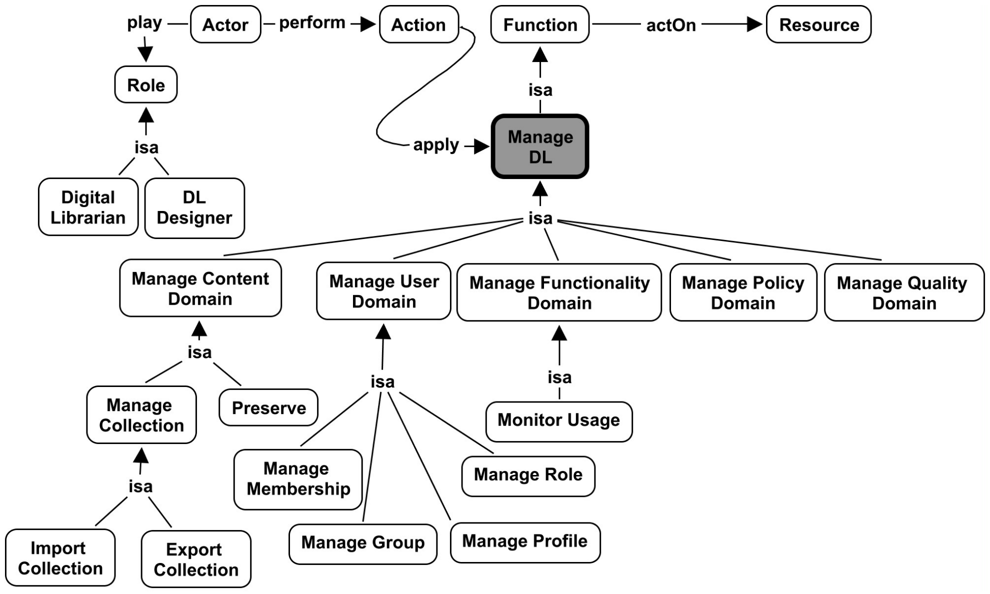

Figure II.2.13. Functionality Domain Concept Map: Manage DL Functions

Manage & Configure DLS (Figure II.2.14) contains *Functions* serving the *DL Manager* – in particular, the *DL System Administrator* – role with regard to setting up, configuring and monitoring the digital library service from a physical point of view, i.e., deploying the Digital Library System needed to implement and support the expected Digital Library.

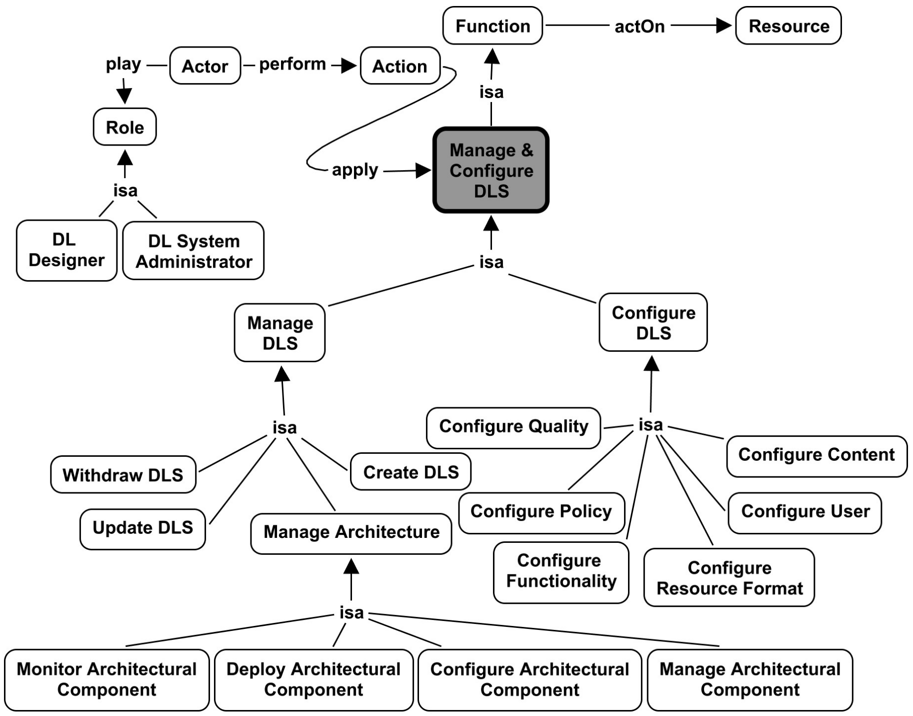

Figure II.2.14. Functionality Domain Concept Map: Manage DLS Functions

Functions realise what is usually called a 'business process' which is in the service of meeting specific 'business requirements' that satisfy a 'stakeholder need' (Lavoie, Henry, & Dempsey, 2006). As mentioned earlier, the *Functionality Domain* is probably one of the most dynamic of all fundamental *Domains* in the Digital Library Universe; hence, what is included in the present version of the Reference Model represents only a subset of *Functions* that one might imagine for digital libraries and corresponds to the concepts that are considered as most critical, i.e., *Functions* available in most of the existing DLs and necessary for supporting the interaction with its intended clients or *Functions* expected by DLMSs to deploy and operate the expected service.

II.2.5 Policy Domain

The *Policy Domain* represents the set of conditions, rules, terms or regulations governing the operation of any digital library 'system', i.e., DL, DLS and DLMS. Policy at large govern the operation of any kind of 'system' including our society or the Institution or Organisation that set up the Digital Library. Policies are always addressed to defined *Actors*. In fact, this domain is very broad and dynamic by nature. The representation provided by this model does not purport to be exhaustive, especially with respect to the myriad of specific rules each Institution would like to model and apply. The Policy domain captures the minimal set of relationships connecting it to the rest and presents the kind of rules that are considered as most critical in the Digital Library universe.

The most general policy concept is ***Policy*** (Figure II.2.15), the entity regulating the existence of a *Resource* with respect to a certain management point of view (<*regulatedBy*>). Each *Policy* is itself a *Resource* in this model and thus inherits all the characteristics of the former.

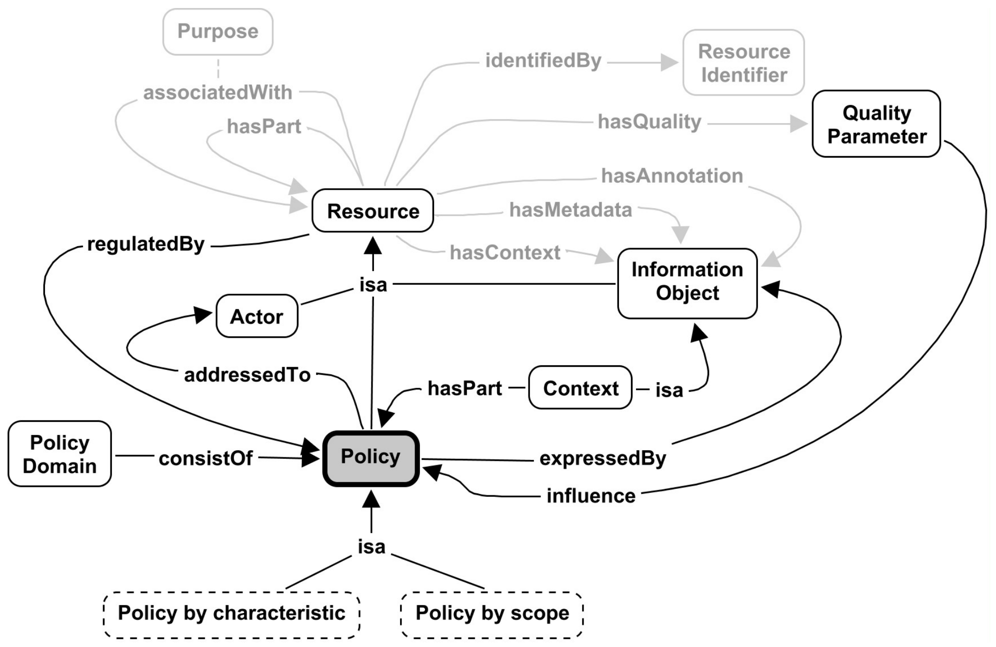

Figure II.2.15. Policy Domain Concept Map

Policy is actually a class of various types of policies (Figure II.2.16). For the purpose of this model, two abstract and orthogonal conceptual containers have been identified, i.e., ***Policy by characteristic*** and ***Policy by scope***.

Policy by characteristic is further specialised into eight subclasses, 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 and to clarify requirements at all levels across the boundaries of the three 'systems', DL, DLS and DLMS.

Policy by scope is further specialised into various classes, each representing a particular *Policy* with respect to (a) the system as a whole, e.g., *Resource Management Policy*; (b) a certain domain, e.g., *User Policy* or *Content Policy*. In some cases a *Policy* actually serves the needs of multiple domains, e.g. *Access Policy* is a *User Policy* and a *Functionality Policy* at the same time.

Recall that the model is extensible and does not intend to form an exhaustive list but rather a sample capturing some of the most important *Policies* governing the Digital Library universe. Among them, a special role is occupied by the *Digital Rights Management Policy* and *Digital Rights*.

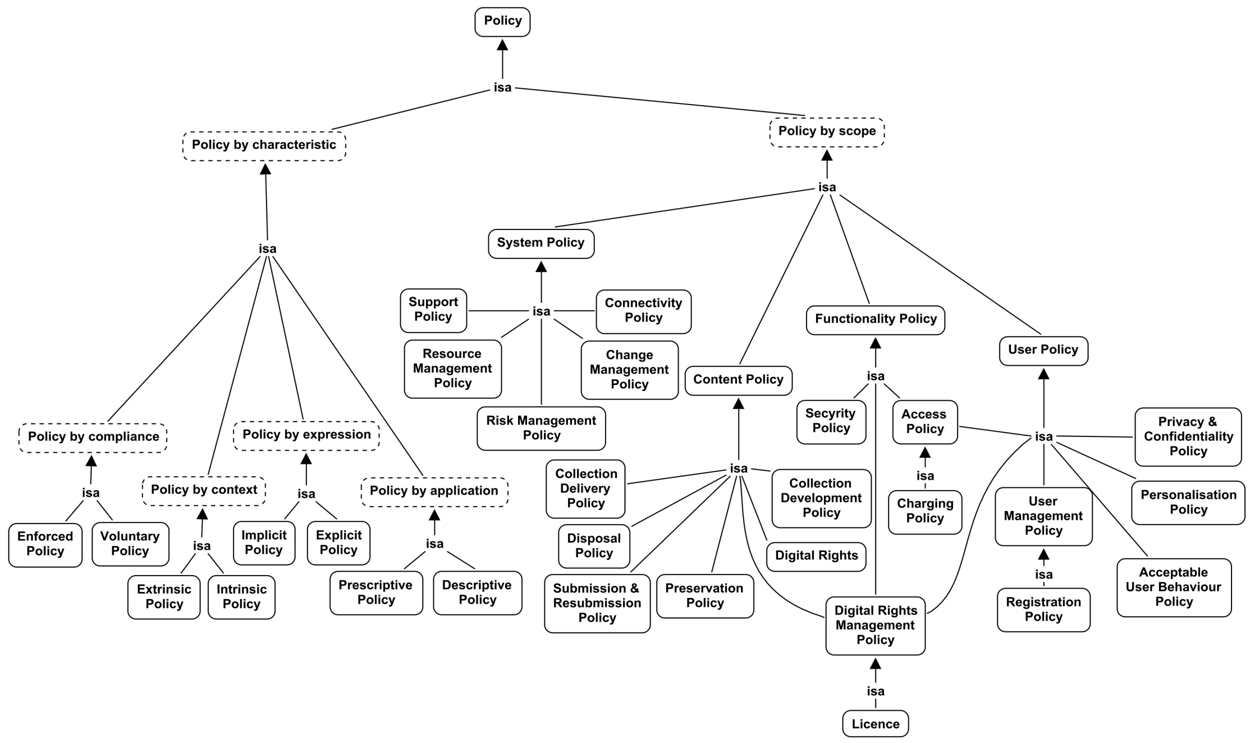

Figure II.2.16. Policy Domain Concept Map: Policies' Hierarchy

II.2.6 Quality Domain

The *Quality Domain* represents the aspects that permit considering any digital library 'system' 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* that can be either implicitly agreed, i.e., *Actors* simply have an understanding of what *Quality Parameters* are guaranteed, or explicitly formulated, i.e., there is a Quality of Service (QoS) agreement.

The most general quality concept is ***Quality Parameter*** (Figure II.2.17), 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*>). *Quality Parameters* express an assessment by an *Actor*, whether human or not, of the *Resource* under consideration. The *Quality Parameters* can be evaluated according to different *Measurements*, which provide alternative procedures for assessing different aspects of each *Quality Parameter* and assigning it a value. *Quality Parameters* are actually expressed by a *Measure*, which represents the value assigned to a *Quality Parameter* with respect to a selected *Measurement*.

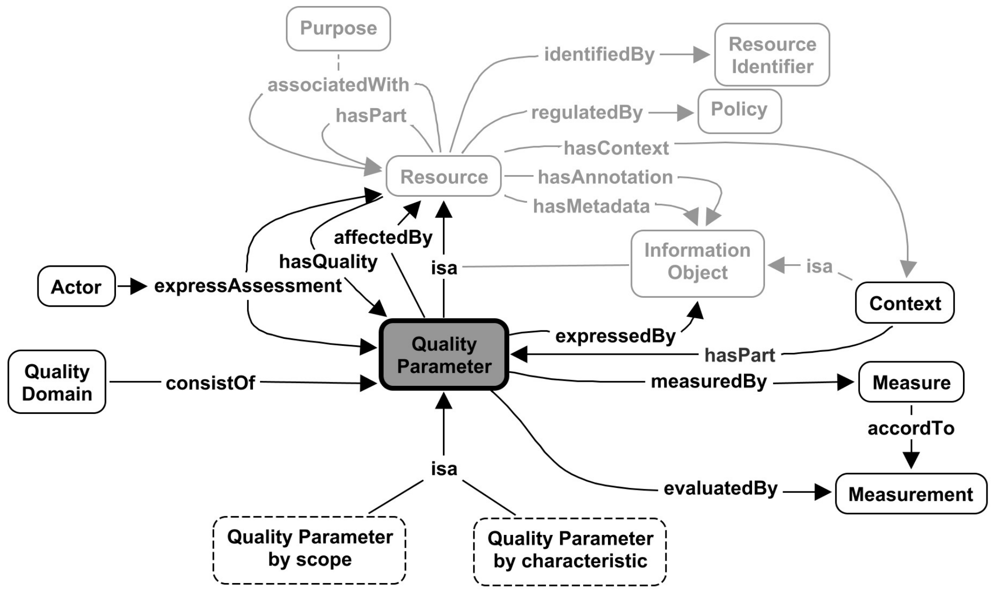

Figure II.2.17. Quality Domain Concept Map

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

The *Quality Domain* is very broad and dynamic by nature, extensible with respect to the myriad of specific quality facets each Institution would like to model. These parameters are grouped according to the *Resource* under examination, i.e., ***Quality Parameter by scope***, and to the characteristics of the Measurement, i.e., ***Quality Parameter by characteristic*** (Figure II.2.18).

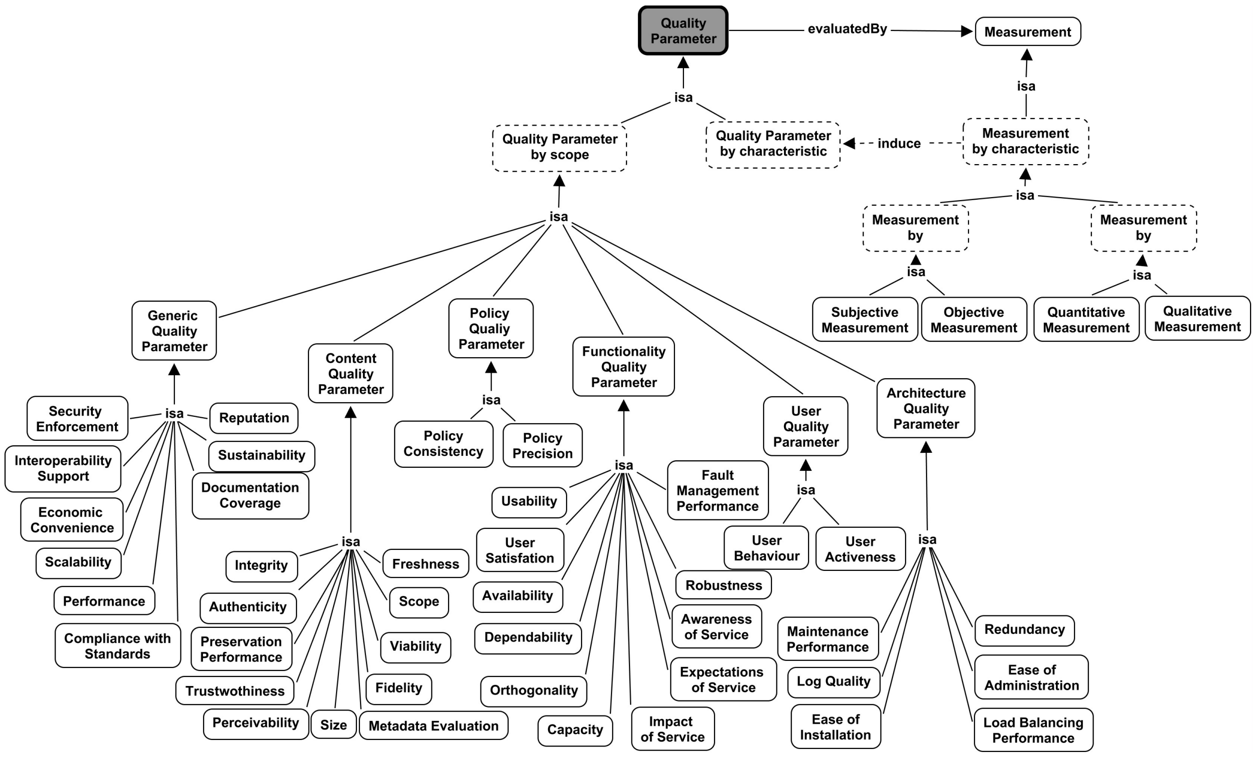

Figure II.2.18. Quality Domain Concept Map: Quality Parameters' Hierarchy

Quality Parameter by scope is further specialised in: ***Generic Quality Parameter*** – apply to any kind or most kinds of *Resources*; ***Content Quality Parameter*** – apply to *Resources* in the *Content Domain*, namely *Information Objects;* ***Functionality Quality Parameter*** – apply to *Resources* in the *Functionality Domain*, namely *Functions*; ***User Quality Parameter*** – apply to *Resources* in the *User Domain*, namely *Actors;* ***Policy Quality Parameter*** – apply to *Resources* in the *Policy Domain*, namely *Policies*; ***Architecture Quality Parameters*** – apply to *Resources* belonging to the *Architecture Domain*, namely *Architectural Components*. It is important to note that this grouping is made from the perspective of the *Resource* under examination, i.e., the main 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/she is an integral part of the definition of *Quality Parameter*. Therefore, ***User Satisfaction*** has been grouped under the *Functionality Quality Parameter* because it expresses how much an *Actor* (the subject who makes the assessment) is satisfied when he/she/it uses a given *Function* (the object of the assessment).

II.2.7 Architecture Domain

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

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

The architecture of a ‘software system’ is a concept easily understood by most engineers, system administrators, and developers, but it is not easily definable. In *An Introduction to Software Architecture* (Garlan & Shaw, 1993), 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 (IEEE, 2000), however, recognises that there is more than just structure in architecture, and defines it as ‘the highest-level concept of a system in its environment’. Thus, this Group’s understanding does not consider the architecture of a software system limited to an inner focus, but rather proposes to take into consideration the system as a whole in its usage and development environments.

The most general concept in the *Architecture Domain* is ***Architectural Component*** (Figure II.2.19), i.e., a system significant component. Thus, for the purposes of this Reference Model, the architecture of a software system (at a given point) is defined as the organisation or structure of its *Architectural Component* interacting with each other (<*use*>) through their interfaces (***Interface***). These components may in turn be composed of smaller and smaller components (<*composedBy*>); however, different *Architectural Components* may be incompatible with each other (<*conflictWith*>), i.e., cannot coexist in the context of the same system. When using the term ‘component’ the software industry and the literature 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.[16]

Each *Architectural Component* is a *Resource*, thus it inherits the *Resource*'s characterising aspects (cf. Section II.2.1), e.g., it is uniquely identified. Like any *Resource*, components have *Metadata* (***Component Profile***) which are expected to capture fundamental information for managing these kind of *Resource* including the implemented or supported *Functions*, the implemented *Interfaces*, their governing *Policies*, and the *Quality Parameters* characterising them.

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

Architectural Components are classified into ***Software Architecture Components*** and ***System Architecture Components***. These classes are used to describe the ***Software Architecture*** and the ***System Architecture*** of a software system respectively, where the former captures the organisation of the programs a software system consists of, while the latter captures the organisation of the processes and running units an operating software system consists of.

Software Architecture Components are realised by ***Software Components***. A *Software Component*, encapsulates the implementation of a portion of a software system and is regulated by (<*regulatedBy*>) particular *Policies* (***Licenses***). Moreover, 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*. Exemplars of Software Architecture Components are software packages implementing a specific Function, software artefacts supporting the implementation of a specific Functions, e.g. a Relational Database Management System (RDBMS).

System Architecture Components are realised by ***Hosting Nodes*** and ***Running Components***. A *Hosting Node* represents the (virtual) hardware environment hosting and running *Software Components*. A *Running Component* represents a running instance of a *Software Component* (<*realisedBy*>) active on a *Hosting Node*. Exemplars of System Architecture Components are servers that can host one or more of the DLS processes or running units, an operational Web Service partaking to the System Architecture of a DLS, a deployed RDBMS.

[16] The 'granularity' of the notion of 'component' is out of the scope of this model. The concepts and relations exploited are powerful and generic enough to capture this granularity at any level. Thus a 'component' is any part of a 'system' fulfilling a functionality.

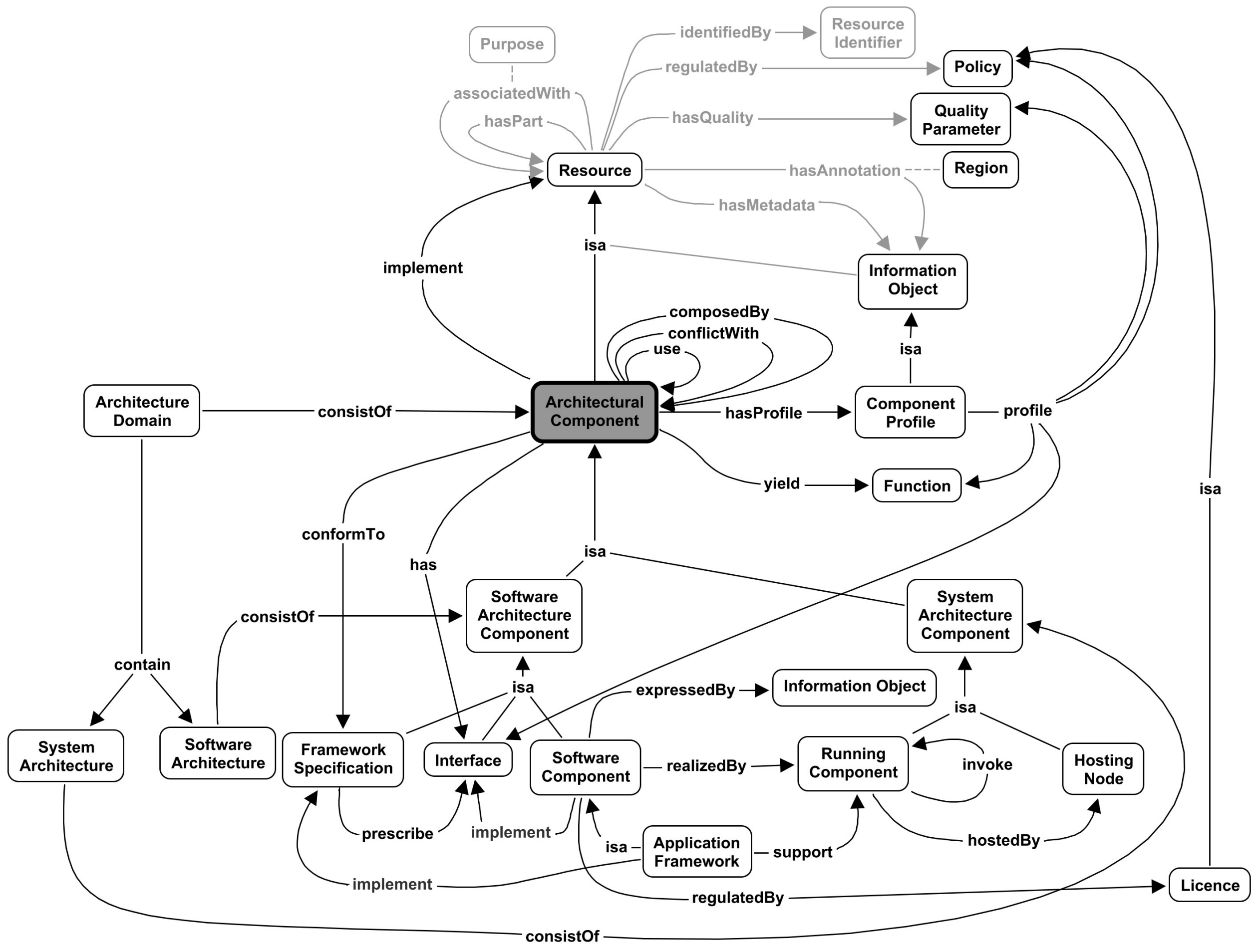

Figure II.2.19. Architecture Domain Concept Map

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

- The system is assembled from discrete executable components, which are developed and deployed somewhat independently of one another, and potentially by different players.
- The system may be upgraded with smaller increments, i.e., by upgrading some of the constituent components only. In particular, this aspect is one of the key points for achieving interoperability, as upgrading the appropriate constituents of a system enables it to interact with other systems.
- Components may be shared by systems; this creates opportunities for reuse, which contributes significantly to lowering the development and maintenance costs and the time to market.
- Though not strictly related to their being component-based, component-based systems tend to be distributed.

All these characteristics represent important features of current and future generations of DLSs and DLMSs.

II.3 Reference Model in Action

The Reference Model sets out to contribute to digital library foundations, but its value is not merely theoretical. It also provides a core instrument for a large variety of different concrete usages, as demonstrated by the feedback received since the release of its first draft version. The Manifesto, for example, has been exploited several times to clarify to stakeholders the complexity of the Digital Library universe and the value of the Digital Library 'systems' in the content production and management workflow. At a very different level, the detailed specification of the concepts and relationships that characterise a Digital Library has been largely exploited in designing a concrete software service that partially automates the process of creation of (virtual) digital libraries (Assante, et al., 2008). Through this service, the effort spent by DL Managers in performing their tasks is considerably reduced. The Reference Model has also been used as a basis for educational courses in digital libraries. Even if limited, the experience so far shows that the model provides a good integrated framework for introducing and explaining concepts. Starting from this framework, existing systems can easily be described and compared.

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

Among the many issues where the Reference Model has proved useful, two merit special attention: *interoperability* and *preservation*. These are closely related as preservation can be interpreted as 'interoperability over time'. They are discussed briefly in the next two subsections. Their initial treatment within the Reference Model appears very promising, but we expect that a more in-depth analysis will identify more systematic approaches and methodologies for handling these issues and suitable metrics to measure the degree of interoperability/preservation achieved.

II.3.1 Interoperability

Whenever two or more systems decide to operate together to better serve their clientele, *interoperability* issue comes up. So far, the Reference Model focuses on an individual Digital Library but the plan is to extend its scope to address multiple interoperating Digital Libraries as well, as this is fundamental for the development of current and future systems. This section provides initial thoughts on this problem and lists the Reference Model concepts deemed to be of particular importance for interoperability.

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

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

The results of exploiting the Reference Model to attack the interoperability issue have been collected in the Digital Library Technology and Methodology Cookbook (Athanasopoulos, et al., 2011). This represents an innovative artefact that collects and describes a portfolio of best practices and pattern solutions to common issues faced when developing interoperable Digital Library systems. It proposes an interoperability model that can be used to characterise – in a systematic way – the interoperability problem facets as well as the existing and forthcoming solutions and approaches as to have a framework for selecting and assessing them.

II.3.2 The Preservation Issue

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

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

Preservation aims to:

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

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

Preservation might also be viewed as interoperability over time.

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

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

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

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

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

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

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

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

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

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

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

- Generic Quality Parameters:
  - ***Security Enforcement*** (cf. Section III.3 C167)
  - ***Interoperability Support*** (cf. Section III.3 C165)
  - ***Documentation Coverage*** (cf. Section III.3 C169)
- Content Quality Parameters:
  - ***Integrity*** (cf. Section III.3 C177)
  - ***Authenticity*** (cf. Section III.3 C174)
  - ***Trustworthiness*** (cf. Section III.3 C175)
  - ***Preservation Performance*** (cf. Section III.3 C178)
  - ***Fidelity*** (cf. Section III.3 C181)
  - ***Dependability*** (cf. Section III.3 C193)
- Functionality Quality Parameters:
  - ***Fault Management Performance*** (cf. Section III.3 C190)
- Architecture Quality Parameters:
  - ***Compliance with Standards*** (cf. Section III.3 C172)

II.4 Related Work

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

II.4.1 The CIDOC Conceptual Reference Model

The CIDOC Conceptual Reference Model (CRM)[17] is an initiative whose goal is to provide a model, i.e., a formal ontology, for describing implicit and explicit concepts and relationships needed to describe cultural heritage documentation. This activity started in 1996 under the auspices of the ICOM-CIDOC Documentation Standard Working Group and since December 2006 it has been an official ISO standard (ISO 21127:2006, 2006).

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

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

The *Persistent Item* class can be compared to our notion of *Resource* as univocal identified entity (*Resource Identifier*). It is further specialised to form a hierarchy. *Thing* is the direct subclass and represents usable discrete, identifiable instances of persistent items documented as single units. At this point a complex hierarchy of *things* classes is introduced. In this hierarchy three classes need to be further explained, namely *Conceptual Object*, *Information Object* and *Collection*. A *Conceptual Object* is defined as 'non-material product of our minds, in order to allow for reasoning about their identity, circumstances of creation and historical implications'. It shares many commonalities with the IFLA-FRBR concept of Work (Madison, et al., 2009), while its counterpart in the Digital Library Reference Model is the *Information Object*. The CIDOC-CMR *Information Objects* are defined as 'identifiable immaterial items, such as poems, jokes, data sets, images, texts, multimedia objects, procedural prescriptions, computer program code, algorithm or mathematical formulae, that have an objectively recognisable structure and are documented as single units'. The CIDOC *Information Object* concept falls within the concept of *Information Object* of the Digital Library Reference Model. The CIDOC model takes care of complex *Information Objects* through the '*is composed of*' property as well as of rights ownership through the linking between *Legal Object*[18] and *Right*. *Collection* is defined as 'aggregation of physical items that are assembled and maintained by one or more instances of Actor over time for a specific purpose and audience, and accounting to a particular collection development plan'. Thus, differing from

[17] The CIDOC Conceptual Reference Model http://www.cidoc-crm.org

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

the Digital Library Reference Model, the CIDOC-CRM only refers collections to physical instantiation of such aggregative mechanism.

Actor, i.e., people who individually or as a group have the potential to perform actions of which they can be deemed responsible, is introduced as a specialisation of the *Persistent Item* class. This concept presents many commonalities with the one introduced in the Digital Library Reference Model and presented in Section II.2.2.

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

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

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

The 5S framework (Gonçalves, 2004; Gonçalves, Fox, Watson, & Kipp, 2004) is the result of an activity aimed at defining digital libraries in a rigorous manner. It is based on five fundamental abstractions, namely *Streams*, *Structures*, *Spaces*, *Scenario*s and *Societies*.

These five concepts are informally defined as follows:

- *Streams* are sequences of elements of an arbitrary type (e.g., bits, characters, images) and thus they can model both static and dynamic content. *Static streams* correspond to information content represented as basic elements, e.g., a simple text is a sequence of characters, while a complex object like a book may be a stream of simple text and images. *Dynamic streams* are used to model any information flow and thus are important for representing any communication that takes place in the digital library. Finally, streams are typed and the type is used to define their semantics and application area.
- *Structures* are the way through which parts of a whole are organised. In particular, they can be used to represent hypertexts and structured information objects, taxonomies, system connections and user relationships.
- *Spaces* are sets of objects together with operations on those objects conforming to certain constraints. This type of construct is powerful and, as suggested by the conceivers, when a part of a DL cannot be well described using another of the 5S concepts, space may well be applicable. Document spaces are the key concepts in digital libraries. However, spaces are used in various contexts – e.g., indexing and visualising – and different types of spaces are proposed, e.g., measurable spaces, measure spaces, probability spaces, vector spaces and topological spaces.
- *Scenarios* are sequences of events that may have parameters, and events represent state transitions. The state is determined by the content in a specific location but the value and the location are not investigated further because these aspects are system dependent. Thus a scenario tells what happens to the *streams* in *spaces* and through the *structures*. When considered together, the *scenarios* describe the services, the activities and the tasks representing digital library *functions*. DL workflows and dataflows are examples of scenarios.

- *Societies* are sets of entities and relationships. The entities may be humans or software and hardware components, which either use or support digital library services. Thus, society represents the highest-level concept of a Digital Library, which exists to serve the information needs of its *societies* and to describe the context of its use.

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

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

These concepts are of general purpose and represents low-level constructors. Using these concepts, Gonçalves *et al*. introduced the whole DL ontology reported in Figure II.4.1.

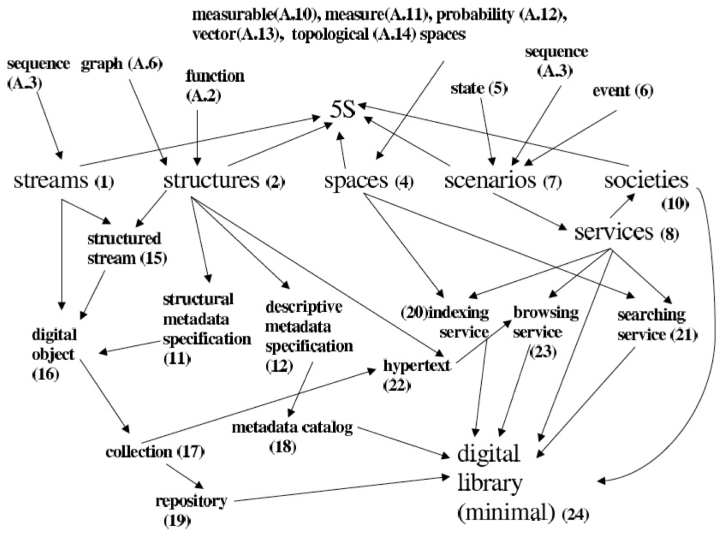

Figure II.4.1. 5S: Map of Formal Definitions[19]

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

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

- R is a repository, a service encapsulating a family of collections and specific services (get, store and del) to manipulate the collections;
- Cat is a set of metadata catalogues for all collections in the repository;
- Serv is a set of services containing at least services for indexing, searching and browsing; and

[19] Figure II.4.1 is extracted from Gonçalves, M. (2004). *Streams, Structures, Spaces, Scenarios, and Societies (5S): A Formal Model for Digital Library Framework and Its Applications.* Virginia Polytechnic Institute and State University.

- Soc is a society.

On top of this, a framework aimed at arranging the concepts and identifying the relationships between them has been proposed. It is depicted in Figure II.4.2.

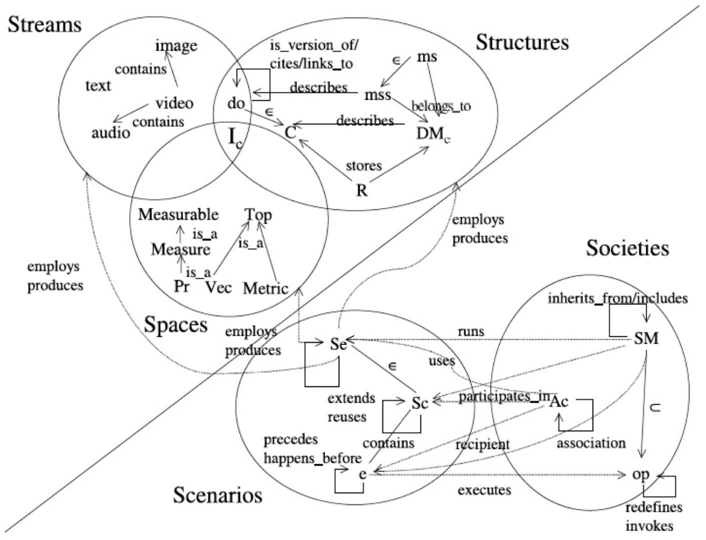

Figure II.4.2. 5S: DL ontology[20]

Figure II.4.3 shows a correspondence between the area covered by the 5S framework and the Reference Model: 5S basically covers what in the Reference Model have been called *Content*, *Functionality* and *User* main concepts; the *Quality* main concept has been addressed separately in the 5S Quality model (Gonçalves, Moreira, Fox, & Watson, 2007), while the *Policy* main concept has scarcely been dealt with in the 5S framework. Moreover, the degree of detail in the different areas can vary, since in some areas the 5S framework introduces very fine-grained concepts while in other areas it adopts a more high-level approach; similar considerations also hold for the Reference Model.

[20] Figure II.4.2 is extracted from Gonçalves, M. (2004). *Streams, Structures, Spaces, Scenarios, and Societies (5S): A Formal Model for Digital Library Framework and Its Applications.* Virginia Polytechnic Institute and State University.

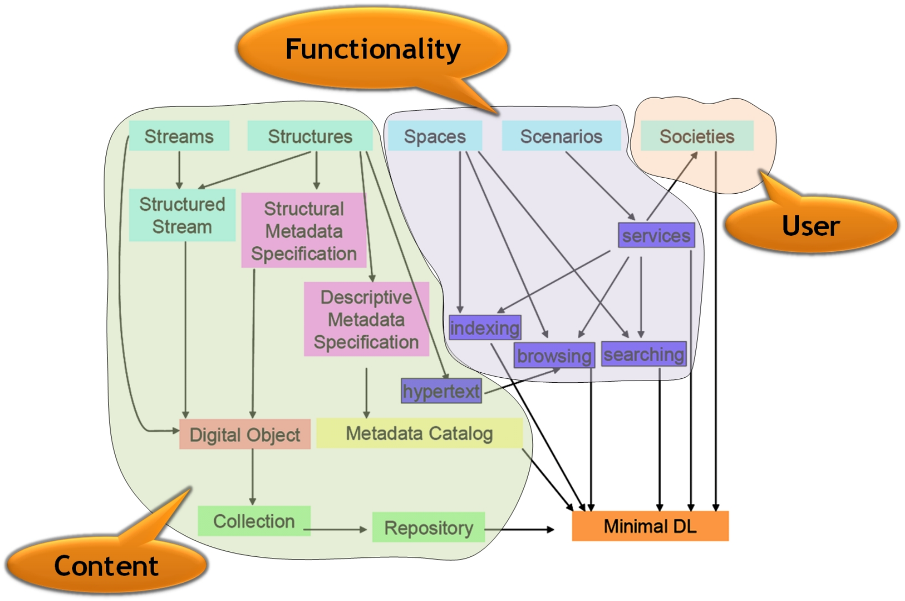

Figure II.4.3. 5S: Areas Covered by the Reference Model

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

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

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

II.4.3 The DELOS Classification and Evaluation Scheme

The DELOS Working Group dealing with the *evaluation of digital libraries* problem proposed a model (Fuhr, Hansen, Mabe, Micsik, & Solvberg, 2001; Fuhr, et al., 2006) that is broader in scope than the one usually adopted in the evaluation context. The aim is to be able to satisfy the needs of all DL researchers, either from the research community or from the library community.

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

- the definition of the set of users predefines the range and content of the collection relevant and appropriate for them;
- the nature of the collection predefines the range of technologies that can be used; and
- 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, 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 (Tsakonas, Kapidakis, & Papatheodorou, 2004). For each of these three aspects, techniques and principles for producing quantitative data and implementing their evaluation have been introduced.

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

II.4.4 DOLCE-based Ontologies for Large Software Systems

DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) is a foundational ontology developed to capture the ontological categories underlying natural language and human common sense. By relying on the basic constructs it identifies, a framework of a set of ontologies for modelling modularisation and communication in Large Software Systems has been developed (Oberle, Lamparter, Grimm, Vrandecic, Staab, & Gangemi, 2006).

This framework consist of three ontologies:

- the Core Software Ontology (CSO);
- the Core Ontology of Software Components (COSC); and
- the Core Ontology of Web Services (COWS).

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

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

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

II.5 Reference Model in a Nutshell: Concluding Remarks

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

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

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

PART III The Digital Library Reference Model Concepts and Relations

III.1 Introduction

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

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

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

III.2 Concepts' Hierarchy

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

C1 Resource

. C2 Resource Identifier

. C3 Resource Set

. . C4 Result Set (also <isa> Information Object)

. . C18 Collection

. . C23 Group (also <isa> Actor)

. C5 Resource Format

. C19 Query

. C20 Ontology

. [Content Resource][21]

. . 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 Provenance (see <hasProvenance> relation)

. . . . . C13 Context

. . . . . C14 Actor Profile

. . . . . C15 Action Log

. . . . . C16 Component Profile

. . . . C17 Annotation

. . . C18 Collection

. . . C4 Result Set (also <isa> Resource Set)

. [User Resource]

. . C22 Actor

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

. . . C23 Group (also <isa> Resource Set)
. . . . C24 Community
. . C25 Role
. . . C26 DL End-user
. . . . C27 Content Consumer
. . . . C28 Content Creator
. . . . C29 Digital Librarian
. . . C30 DL Manager
. . . . C31 DL Designer
. . . . C32 DL System Administrator
. . . C33 DL Software Developer
. . C14 Actor Profile (also <isa> Metadata)
. . C34 Action

. [Functionality Resource]
. . C36 Function
. . . C37 Access Resource
. . . . C38 Discover
. . . . . C39 Browse
. . . . . C40 Search
. . . . C41 Acquire
. . . . C42 Visualise
. . . C43 Manage Resource
. . . . C44 Create
. . . . C45 Submit
. . . . C46 Withdraw
. . . . C47 Update
. . . . C48 Preserve
. . . . C49 Validate
. . . . C50 Annotate
. . . . C51 Manage Information Object
. . . . . C52 Disseminate
. . . . . . C53 Publish
. . . . . C54 Author
. . . . . . C55 Compose
. . . . . C56 Process
. . . . . . C57 Analyse
. . . . C58 Linguistic Analysis
. . . . C59 Qualitative Analysis

. . . . C99 Manage Functionality
. . . . . C100 Monitor Usage
. . . . C101 Manage Quality
. . . . C102 Manage Policy Domain
. . . C103 Manage & Configure DLS
. . . . C104 Manage DLS
. . . . . C105 Create DLS
. . . . . C106 Withdraw DLS
. . . . . C107 Update DLS
. . . . . C108 Manage Architecture
. . . . . . C109 Manage Architectural Component
. . . . . . C110 Configure Architectural Component
. . . . . . C111 Deploy Architectural Component
. . . . . . C112 Monitor Architectural Component
. . . . C113 Configure DLS
. . . . . C114 Configure Resource Format
. . . . . C115 Configure Content
. . . . . C116 Configure User
. . . . . C117 Configure Functionality
. . . . . C118 Configure Policy
. . . . . C119 Configure Quality

. [Policy Resource]
. . C121 Policy
. . . [Policy by characteristic]
. . . . [Policy by context]
. . . . . C122 Extrinsic Policy
. . . . . C123 Intrinsic Policy
. . . . [Policy by expression]
. . . . . C124 Explicit Policy
. . . . . C125 Implicit Policy
. . . . [Policy by application]
. . . . . C126 Prescriptive Policy
. . . . . C127 Descriptive Policy
. . . . [Policy by compliance]
. . . . . C128 Enforced Policy
. . . . . C129 Voluntary Policy
. . . [Policy by scope]
. . . . C130 System Policy

. . . . . . C131 Change Management Policy
. . . . . . C132 Resource Management Policy
. . . . . . C133 Support Policy
. . . . . . C134 Connectivity Policy
. . . . . . C135 Risk Management Policy
. . . . . C136 Content Policy
. . . . . . C137 Disposal Policy
. . . . . . C138 Collection Development Policy
. . . . . . C139 Collection Delivery Policy
. . . . . . C140 Submission and Resubmission Policy
. . . . . . C142 Digital Rights
. . . . . . C144 Preservation Policy
. . . . . . C141 Digital Rights Management Policy
. . . . C143 License
. . . . . C145 User Policy
. . . . . . C146 User Management Policy
. . . . C147 Registration Policy
. . . . . . C148 Personalisation Policy
. . . . . . C149 Privacy and Confidentiality Policy
. . . . . . C150 Acceptable User Behaviour Policy
. . . . . C151 Functionality Policy
. . . . . . C152 Access Policy
. . . . C153 Charging Policy
. . . . . . C154 Security Policy

. [Quality Resource]
. . C156 Measurement
. . . C157 Objective Measurement
. . . C158 Subjective Measurement
. . . C159 Qualitative Measurement
. . . C160 Quantitative Measurement
. . C161 Measure
. . C162 Quality Parameter
. . . C163 Generic Quality Parameter
. . . . C164 Economic Convenience
. . . . C165 Interoperability Support
. . . . C166 Reputation
. . . . C167 Security Enforcement
. . . . C168 Sustainability

. . . . C208 Maintenance Performance
. . . . C209 Redundancy

. [Architectural Resource]
. . C211 Architectural Component
. . . C212 Software Architecture Component
. . . . C213 Software Component
. . . . . C214 Application Framework
. . . . C215 Interface
. . . . C216 Framework Specification
. . . C217 System Architecture Component
. . . . C218 Running Component
. . . . C219 Hosting Node
. . C16 Component Profile (also <isa> Metadata)
. . C143 License (also <isa> Policy)
. . C220 Software Architecture
. . C221 System Architecture

III.3 Reference Model Concepts' Definitions

C1 Resource

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

Relationships:

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

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

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

Examples:

- *Information Object* or a *Collection;*
- *Actor;*
- *Function;*
- *Policy;*
- *Ontology.*

C2 Resource Identifier

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

Relationships:

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

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

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

Examples:

- Uniform Resource Identifiers (URIs)[22];
- Internationalized Resource Identifiers (IRIs)[23];
- Archival Resource Keys (ARKs)[24];
- Digital Object Identifier (DOI)[25];
- Persistent handles.

C3 Resource Set

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

Relationships:

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

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

Examples:

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

C4 Result Set

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

Relationships:

- *Result Set <isa> Resource Set*

Rationale: A set of *Resources* returned by the system as the consequence of an *Actor* issuing a *Query*. *Result Set* is a group of *Resources* that are highly dynamic and time dependent, i.e., different *Result Sets* can be obtained by issuing the same *Query* in different time periods. This is a due to the changes in the *Resource Set* forming the search space, this set of resources evolves as a consequence of the `system' operation, e.g., new *Collections* can be created.

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

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

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

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

Examples:

- The set of *Information Objects* representing Picasso paintings retrieved by issuing a *Query like `Picasso'.*
- The set of *Resources* having `Leonardo' as keyword in their Metadata.

C5 Resource Format

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

Relationships:

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

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

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

Examples:

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

C6 Content Domain

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

Relationships:

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

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

Examples:

- In a DL containing medieval manuscripts the Content Domain would cover all aspects of the representation of these documents in the DL. The manuscripts would be represented as *Information Objects*, organized in *Collections* and maybe related to *Annotations* or *Metadata*.

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,* which are also represented as *Information Objects*. 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* is the main entity populating the *Content Domain*. The management of this kind of entities dedicated to capture any form of information is in fact the purpose of the Digital Library domain since the beginning. *Information Objects* are representations of raw data or organized information items that are stored in the Digital Library 'system' with the objective to provide its users (*Actors*) with the data they needs in an organised and seamless way.

An *Information Object* may be a simple text document, either scanned or in full-text format. It may also be a complex, multimedia and multi-type object with parts, such as a sound recording associated with a set of slides, a music score, political and economic data associated with interactive simulations, a PhD thesis which includes a representation of a performance, a simulation experiment and the experimental data set adopted, or a data stream representing the pool of data continuously measured by a sensor. This information is given in the *Resource Format* linked to the *Information Object* via a *<hasFormat>* relationship. Thanks to this relationship the mechanism identifying the boundaries and the structure of each *Information Object* is particularly flexible and powerful. For instance, it is possible to have a huge *Information Object* representing a soccer game, composed of 27 parts each containing the soccer game as captured by a particular camera. Another way to organise the same soccer game *Information Object* is to have a *Collection* of *Information Objects*, one for each of the highlights of the match; each of these *Information Objects* can be further broken down into parts, each representing the highlight as captured by a different camera, etc. Moreover, the notions of *Edition*, *View* and *Manifestation* represent yet another way of modelling *Information Objects* according to the semantics fixed by the IFLA-FRBR model

(Madison, et al., 2009). This model is particularly useful in dealing with 'document' *Information Objects* but can be extended and applied 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 (ISO 21127:2006, 2006), where it is used to refer to 'identifiable immaterial items, such as poems, jokes, data sets, images, texts, multimedia objects, procedural prescriptions, computer program code, algorithm or mathematical formulae, that have an objectively recognisable structure and are documented as single units'.

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

Examples:

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

C8 Edition[26]

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 modelling point of view, they are defined similarly to *Metadata* or *Annotations*, i.e., as derived concepts from a relation, in this case <*hasEdition*>.

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

Examples:

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

[26] This is a ***derived concept***, i.e. it is not depicted in any concept maps. Because of a modelling style, the notion of Edition – that is a fundamental one in the Content Domain – has been captured by the <*hasEdition*> Relation. The concept has been introduced here for the sake of simplicity, to make explicit this fact as well as to reflect the scope of Part III of this document that should provide its reader with a set of concepts characterising the Digital Library domain.

C9 View[27]

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

Relationships:

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

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

From a modelling point of view, they are defined similarly to *Metadata* or *Annotations*, i.e., as derived concepts from a relation, in this case <*hasView*>.

Edition and *View* together capture the expression concept of the IFLA-FRBR model (Madison, et al., 2009).

Examples:

- An example of View is that of an *Information Object* representing a data stream of an environmental sensor. This can be 'seen' in terms of its raw form as a series of numerical values or as a graph representing the evolution of the values measured by the sensor over time.
- Another example may be that of an *Information Object* representing the outcomes of a workshop; three different views of this object can be envisaged:
  - the 'full view' containing a preface prepared by the conference chair and the whole set of papers accepted and organised thematically;
  - the 'handbook view' containing the conference programme and the slides of each lecturer accompanied by the abstract of the papers organised per session; and
  - the 'informative view' reporting the goal of the workshop and the title list of the accepted papers together with the associated abstract.

[27] This is a ***derived concept***, i.e. it is not depicted in any concept maps. Because of a modelling style, the notion of View – that is a fundamental one in the Content Domain – has been captured by the <*hasView*> Relation. The concept has been introduced here for the sake of simplicity, to make explicit this fact as well as to reflect the scope of Part III of this document that should provide its reader with a set of concepts characterising the Digital Library domain.

C10 Manifestation[28]

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

Relationships:

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

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

From a modelling point of view, they are defined similarly to *Metadata* or *Annotations*, i.e., as derived concepts from a relation, in this case <*hasManifestation*>.

Examples:

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

C11 Metadata[29]

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*

[28] This is a ***derived concept***, i.e. it is not depicted in any concept maps. Because of a modelling style, the notion of Manifestation – that is a fundamental one in the Content Domain – has been captured by the <*hasManifestation*> Relation. The concept has been introduced here for the sake of simplicity, to make explicit this fact as well as to reflect the scope of Part III of this document that should provide its reader with a set of concepts characterising the Digital Library domain.

[29] This is a ***derived concept***, i.e. it is not depicted in any concept maps. Because of a modelling style, the notion of Metadata – that is a fundamental one in the Content Domain, actually in the Resource Domain – has been captured by the <*hasMetadata*> Relation. The concept has been introduced here for the sake of simplicity, to make explicit this fact as well as to reflect the scope of Part III of this document that should provide its reader with a set of concepts characterising the Digital Library domain.

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

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

There are a number of schemes for classifying metadata.

One of such categorisations classifies metadata according to the specific role they play:

- Descriptive metadata, i.e., metadata that provide a mechanism for representing attributes describing and identifying the *Resource*. Examples include bibliographical attributes (e.g., creator, title, publisher, date), format, list of keywords characterising the contents. The term 'descriptive' is used here in a consistent but broader sense than in 'descriptive cataloguing'.
- Administrative metadata, i.e., metadata for managing a *Resource*. This category of metadata may include metadata detailing: (*i*) technical characteristics of the Resource; (*ii*) the history of the operations performed on the Resource since its creation/ingest; (*iii*) means of access; and (*iv*) how the authenticity and integrity of the Resource can be verified.
- Preservation metadata, i.e., metadata designed to support the long-term accessibility of a Resource by providing information about its content, technical attributes, dependencies, management, designated community(ies) and change history. Preservation metadata have been identified as essential for the long-term management of digital objects. The Reference Model for an Open Archival Information System (OAIS) (ISO14721:2003, 2003) provides an excellent overview of the role of preservation metadata in the management over time of digital resources. PREservation Metadata: Implementation Strategies Working Group, commonly referred to as PREMIS, has defined a core set of preservation metadata elements that would provide support for the management of digital objects across systems and time. They acknowledged that, while they had identified the key aspects of the necessary preservation metadata, there was room for more work in the area of technical metadata and this might be necessary at the level of each DL *Resource* or *Collection*. 'Preservation metadata' encompasses technical elements necessary to enable access to, manipulation and/or rendering of a DL *Resource*, data about the structure and syntax of an *Information Object*, information to support semantic understanding of *Resources* and details of the responsibilities and rights governing the application of preservation actions to *Resources*.

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

- Syntactic metadata present data about the syntax or structure of the resource. They provide data such as time or space of *Resource* creation or inclusion into the *Digital_Library*, or size of *Resource*. Inherent metadata can be obtained from the *Resource* itself. They are dependent on the *Manifestation* used.
- Semantic metadata provide data about the *Resource* itself (the semantics of the *Resource*). These metadata could be explicitly or implicitly derived from the *Resource*, or generated by humans.
- Contextual metadata provide data related neither to the structure nor to the semantics of a *Resource* but to other issues within the context of the DL. They might be needed to understand the *Resource* or the ways of its possible use.

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

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

Examples:

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

C12 Provenance[30]

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

Relationships:

- *Provenance <isa> Information Object*
- *Resource <hasProvenance> Information Object* (*Provenance*)
- *Provenance <hasFormat> Resource Format* that is an <*expressionOf*> *Ontology* (inherited by *Resource*)

Rationale: *Provenance* (Gil, et al., 2010) is a record that describes entities and processes involved in producing and delivering or otherwise influencing that *Resource*. Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility. Provenance assertions are a form of contextual metadata and can themselves become important records with their own provenance.

Provenance and Metadata have many commonalities. Some Metadata of a *Resource* only becomes part of its Provenance when one also specifies its relationship to deriving the *Resource*. For example, an *Information Object* can have a *Metadata* that states its size. This is not typically considered *Provenance* since it does not relate to how the *Information Object* was created. The same *Information Object* can have *Metadata* regarding its creation date, which would be considered Provenance-relevant Metadata. However, even though a lot of *Metadata* potentially has to do with *Provenance*, both terms are not equivalent. In summary, *Provenance* is often represented as *Metadata*, but not all *Metadata* is necessarily *Provenance*.

Examples:

- A phenomenon within Twitter is the idea of retweeting, i.e., reposting someone else's tweet as your own message. Denoting a retweet with the prefix RT and the @user sign is an example of Provenance;

[30] This is a ***derived concept***, i.e. it is not depicted in any concept maps. Because of a modelling style, the notion of Provenance – that is a fundamental one in the Resource Domain – has been captured by the <*hasProvenance*> Relation. The concept has been introduced here for the sake of simplicity, to make explicit this fact as well as to reflect the scope of Part III of this document that should provide its reader with a set of concepts characterising the Digital Library domain.

C13 Context[31]

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

Relationships:

- *Context <isa> Information Object*
- *Resource <hasContext> Information Object* (*Context*)
- *Context <hasFormat> Resource Format* that is an <*expressionOf*> *Ontology* (inherited by *Resource*)
- *Context <hasPart>* Actor
- *Context <hasPart> Policy*
- *Context <hasPart> Quality Parameter*

Rationale: Context and Metadata – including *Provenance* – have many commonalities. Some Metadata of a *Resource* only becomes part of its Context when one also specifies its relationship to the settings surrounding the *Resource*. For example, an *Information Object* can have a *Metadata* that states its size. This is not typically considered *Context* since it does not relate to the circumstances leading to the *Information Object*. The same *Information Object* can have *Metadata* regarding its creation date, which would be considered Context-relevant Metadata. However, even though a lot of *Metadata* potentially has to do with *Context*, both terms are not equivalent. In summary, *Context* is often represented as *Metadata*, but not all *Metadata* is necessarily *Context*.

Examples:

- The *Information Objects* recording the creation period of a *Resource* and its most important traits, e.g., the events, the findings, the terminology in use.

C14 Actor Profile

Definition: An *Information Object* that models any entity (*Actor*) that interacts with any Digital Library 'system'. 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 <model> Actor*
- *Actor Profile <concern> Resource*
- *User Profiling <create/update> Actor Profile*
- *Actor Profile <influence> Action*

Rationale: An *Actor Profile* is an *Information Object* that models an *Actor* by potentially capturing a large variety of the *Actor*'s characteristics, which may be important for a particular Digital Library for allowing the *Actor* to use the 'system' and interact with it as well as with other *Actors*. It not only serves as a

[31] This is a ***derived concept***, i.e. it is not depicted in any concept maps. Because of a modelling style, the notion of Context – that is a fundamental one in the Resource Domain – has been captured by the <*hasContext*> Relation. The concept has been introduced here for the sake of simplicity, to make explicit this fact as well as to reflect the scope of Part III of this document that should provide its reader with a set of concepts characterising the Digital Library domain.

representation of *Actor* in the system but essentially captures *Policies* and *Roles* that govern which *Functions* an *Actor* is entitled to perform through the *Actor*'s lifetime. For example, a particular instance of *Actor* may be entitled to *Search* within particular *Collections* and *Collaborate* with certain other *Actors*. The characteristics captured in an *Actor Profile* vary depending on the type of *Actor*, i.e., human or non-human, and may include: identity information (e.g., age, residence or location for humans and operating system, web server edition for software components), educational information (e.g., highest degree achieved, field of study – only for humans) and preferences (e.g., topics of interest, pertinent for both human and software *Actors* that interact with the *Digital Library*).

Examples:

- Group Profile, i.e., the *Actor profile* capturing the characteristics of a *Group* as a single entity.
- Community Profile, i.e., the *Actor profile* capturing the characteristics of a *Community* as a single entity.
- Anne is an *Actor* that interacts with a Music Digital Library and has an *Actor Profile* which captures several characteristics, such as her full name, her date of birth, her address and her musical preferences.

C15 Action Log

Definition: A record of the *Actor's Activities* performed while interacting with the Digital Library or the Digital Library Management System.

Relationships:

- *Action Log <isSequenceOf> Action*
- *Action Log <isa> Information Object*

Rationale: Actors when interact with the Digital Library or the Digital Library Management System perform certain Actions that are captured in the Action Log. These records can be used to create or update the *Actor Profile*.

Examples:

- An Action Log is an Information Object that stores Mark activities in the Music Digital Library. Specifically, it contains the Functions that Mark executed and the songs that he listened to.

C16 Component Profile

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

Relationships:

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

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

Examples:

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

C17 Annotation

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

Relationships:

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

Rationale: *Annotations* can support cooperative work by allowing *Actors* to merge their intellectual work with the DL *Resources* provided by the DL to constitute a single working context. *Annotations* can be used in various contexts, e.g.,

- to express a personal opinion about an *Information Object*;
- to enrich an *Information Object* with references to related works or contradictory *Information Object*;
- to add personal notes about a retrieved *Information Object* for future use.

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

As *Annotations* are *Information Objects*, they may be in different formats, be expressed in different media, be associated with *metadata*, and can themselves be annotated. At present, in the literature there is an ongoing discussion as to whether *Annotations* should be considered as *Metadata* or as *Information Objects*. For the time being, an *Annotation* is modelled as an *Information Object* because (*i*) it has been considered as additional information that increases the existing content by providing an additional layer of elucidation and explanation, and (*ii*) because of this the *Annotation* itself takes the shape of an additional *Information Object* that can help people understand the annotated *Resource*. In fact, the status of *Annotation* is derived from the <*hasAnnotation*> relation linking *Resources*; this choice settles the long-standing issue of whether *Annotations* should be considered as *Information Objects* or as *Metadata*.

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

Examples:

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

C18 Collection

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

Relationships:

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

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

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

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

Examples:

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

C19 Query

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

Relationships:

- *Query <isa> Information Object*

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

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

Examples:

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

C20 Ontology

Definition: An *ontology* is a formal conceptualisation that defines the terms about a domain. *Ontologies* formalise a shared vocabulary about a domain (Guarino, 1998).

Relationships:

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

Rationale: The notion of *ontology* generalises that of schema or format, as well as related notions, such as thesaurus. *Ontologies* can refer to different aspects of *Information Objects*, such as their structure, their content, their preservation among others. Although a Digital Library might define and adopt its own proprietary formats, it is widely acknowledged that standard representation models (e.g., Dublin Core for descriptive metadata, MPEG for the structure of audio-visual objects, OAIS for preservation) enhance the interoperability and reuse of *Resources*. The emergence of rich schemas, such as CIDOC Conceptual Reference Model (CRM) (ISO 21127:2006, 2006), which enable content owners or holders to define articulated descriptions of their digital assets, and to exploit such descriptions in accessing the information or in managing complex applications around them demands greater flexibility at the level of generalisation. Semantic Web technologies, notably the Web Ontology Language (OWL), which builds upon Description Logics and the associated inferential capabilities, is another driver.

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

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

Examples:

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

C21 User Domain

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

Relationships:

- *Digital Library <definedBy> User Domain*

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

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

Examples: --

C22 Actor

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

Relationships:

- *Actor <isa> Resource*
- *Actor* is *<identifiedBy> Resource Identifier* (inherited from *Resource*)
- *Actor* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Actor <belongTo> Group*
- *Actor Profile <model> Actor*
- *Actor* is *<associatedWith> Actor* (inherited from *Resource*)
- *Actor <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Actor <has Annotation> Information Object* (inherited from *Resource*)
- *Actor <hasMetadata> Information Object* (inherited from *Resource*)
- *User Domain <consistOf> Actor*
- *Digital Library <serve> Actor*
- *Digital Library System <serve> Actor*
- *Digital Library Management System <serve> Actor*
- *Actor <play> Role*
- *Actor <perform> Action*
- *Group <isa> Actor*
- *Context <hasPart> Actor*

Rationale: An *Actor* captures any entity that interacts with a Digital Library 'system' or with other similar entities through the *Functions* offered by that 'system' and includes humans, and inanimate entities such as software programs or physical instruments. The latter may range from subscription services offered by external systems to portals and other digital libraries that pull content from, or push content to, the particular Digital Library. Although each distinct entity may be recognised in the system by a single *Resource Identifier*, it may play a different *Role* at different times, belong to more than one *Group* and be associated with more than one *Actor Profile*. For instance, an *Actor* may have one profile when assuming the *Content Creator* role and a different profile under the *Content Consumer* role. *Policies* that are associated with an *Actor* and are captured by 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 also produces and enforces policies. An *Actor* may be characterised by various *Quality Parameters*. For instance, a human may be distinguished on the basis of *Trustworthiness* and a software agent may be characterised by its *Robustness*. Such quality parameters may be used to guide or value an *Actor*'s interactions. For instance, in actor groupings, such as human cooperations or co-authorships or software component integrations, captured by instantiating the <*associatedWith*> relations, a more authoritative *Actor* can be trusted for sharing content from an *Actor* of disputable quality.

Examples:

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

C23 Group

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

Relationships:

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

Rationale: A *Group* represents an *Actor* population that exhibits cohesiveness to a large degree and can be considered as an *Actor* with its own profile and identifier. A *Group* is described by an *Actor Profile* that essentially specifies explicitly (through enumeration) or implicitly (through a set of desired characteristics) the members of the group, and specifies the *Roles* an *Actor* of the *Group* can take and the *Policies* that govern the *Actor* interactions in the system, such as permissible *Functions* and accessible *Resources*. A Group can also produce and enforce policies. 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:

- John, Mary and Paul are the *Actors* constituting the *Group* entitled to curate Leonardo da Vinci *Collection* disseminated through their University Digital Library. The *Group* has a *Profile* which specifies that John, Mary and Paul have the *Role* of Librarian as Actors of the Group.

C24 Community

Definition: A *Community* is a particular subclass of *Group*, which refers to a social group of humans with shared interests.

Relationships:

- *Community <isa> Group*

Rationale: 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. Examples of *Community* can be: a pre-existing group of people with shared interests, which is online in the Digital Library; a group gathered together by the DL; or a group that is formed as *Actors* in the Digital Library and that interacts 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. *Community*, as a *Group*, is a well-defined user community identified by a *Resource Identifier* and modelled by a specific *Actor Profile*. The *Profile* records permissible *Roles*, *Functions* and *Resources* according to specific *Policies*.

Examples:

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

C25 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*
- *Role <isa> Actor Profile*
- *DL End-user <isa> Role*
- *Content Creator <isa> Role*
- *Content Consumer <isa> Role*
- *Digital Librarian <isa> Role*
- *DL Manager <isa> Role*
- *DL Designer <isa> Role*
- *DL System Administrator <isa> Role*
- *DL Software Developer <isa> Role*

Rationale: The above definition comes from (Ferraiolo, Kuhn, & Chandramouli, 2003) and works in accordance with the policy mechanism pervading the *Policy Domain* (Section II.2.5). A role is a kind of pre-packaged generic profile and may be seen as a packet of statements identifying the kind of *Functions* an *Actor* is eligible to perform within the system. Thus, a role may be stored as a profile that represents an individual or (most likely) a population of users. *Roles* are also called stereotypes in user modelling. An *Actor* can be assigned to a *Role*; this means, the *Actor* inherits all the *Role* statements. Clearly, an *Actor* can play different *Roles* at different times or more than one *Role* at the same time. Apart from the three main *Actor Roles* defined (*DL End-user*, *DL Manager* and *DL Software Developer*), the following generic *Roles* are distinguished within a DL context and subsequently defined: *(i) Content Consumer*, *Content Creator* and *Digital Librarian*, which are sub-roles of the *End-user* role; and *(ii) DL Designer* and *DL System Administrator*, which are sub-roles of the *DL Manager* role. Apart from these roles and sub-roles, which are prototypically defined in the Reference Model, any digital library could, and should, define additional roles. A sub-role may be defined, providing it with some of the *Functions* of a generic *Role*. For example, a content annotator *Role* might be a sub-role of *Content Creator* that entitles *Actors* to annotate only existing *Information Objects*.

Examples:

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

C26 DL End-user

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

Relationships:

- *DL End-user <isa> Role*

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

Examples:

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

C27 Content Consumer

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

Relationships:

- *Content Consumer <isa> DL End-user*
- *Content Consumer <isa> Role*

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

Examples:

- John, the *DL End-user* of a scientific Digital Library, discovers the *Collections* containing the journal articles covering its research topics and processes this data with its novel procedure;
- Mary is the *DL End-user* of a Digital Library that can borrow resources, explore catalogs, search databases, cite bibliographic data, read articles and books, take notes and collaborate with others.

C28 Content Creator

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

Relationships:

- *Content Creator <isa> DL End-User*
- *Content Creator <isa> Role*

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

Examples:

- John, the *DL End-user* of a scientific Digital Library, uploads a new version of a working paper reporting on the latest results of its experimentation in a *Collection* shared with other colleagues working on a similar topic to prompt and informed feedback;
- Older classic medical articles are of particular importance to medical historians and to some students, researchers, and clinicians. Wider access to these classic print articles is now possible due to the availability of a scanning program which is the *DL End-user* that allows the production of digital copies of print material.

C29 Digital Librarian

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

Relationships:

- *Digital Librarian <isa> DL End-user*
- *Digital Librarian <isa> Role*

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

Examples:

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

C30 DL Manager

Definition: The *Role* of the *Actors* that by exploiting *Digital Library Management System* facilities are in charge of defining, customising and maintaining the *Digital Library* service.

Relationships:

- *DL Manager <isa> Role*

Rationale: *DL Managers* exploit DLMS facilities to define, customise and maintain the DL service. This is actually a class of *Actors* further subdivided in ***DL Designers*** – they define, customise and maintain the DL service – and ***DL System Administrators*** – they exploit DLMS facilities to create and operate the DLS realising the envisaged DL service.

Examples:

- John is the *DL Manager* in a University Digital Library which is requested to define and maintain the entire Digital Library service.
- Mary is another *DL Manager* of the same Digital Library of John which is requested to monitor the Digital Library service from an operational point of view as to guarantee a 24/7 service.

C31 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*
- *DL Designer <isa> Dl Manager*

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

Examples:

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

C32 DL System Administrator

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

Relationships:

- *DL System Administrator <isa> Role*

- *DL System Administrator <isa> DL Manager*

Rationale: DL System Administrators are in charge to select and deploy the *Architectural Components* (C211) needed to operate the *Digital Library System* implementing the expected *Digital Library*. Their choice of elements reflects the expectations that DL End-users and DL Designers have for the *Digital Library*, as well as the requirements the available resources impose on the definition of the DL. Moreover, it complies with the organisation mission and (business) goals of the institutions that set up the 'system'. DL System Administrators interact with the DLMS by providing architectural configuration parameters, such as the chosen software components and the selected hosting nodes. Their task is to identify the architectural configuration that best fits the DLS in order to ensure the highest level of quality of service. The value of the architectural configuration parameters can be changed over the DL lifetime. Changes of parameter configuration may result in the provision of different DL functionality and/or different levels of quality of service.

Examples:

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

C33 DL Software Developer

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

Relationships:

- *DL Designer <isa> Role*

Rationale: DL Application Developers develop the *Software Components* (C213) that will be used as constituents of the DLSs, to ensure that the appropriate levels and types of functionality are available.

Examples:

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

C34 Action

Definition: The *Actor's* activity that applies *Functions* and concerns *Resources*.

Relationships:

- *Actor <perform> Action*
- *Action <apply> Function*
- *Action <concern> Resources*

Rationale: Actors when interact with the *Digital Library* or the *Digital Library Management System* perform certain *Actions* that apply *Functions* and involve Resources.

Examples:

- Mark interacts with the Music Digital Library by performing certain *Actions* that include the execution of the Function Search concerning Digital Library's stored songs.

C35 Functionality Domain

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

Relationships:

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

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

Examples: --

C36 Function

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

Relationships:

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

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

Examples:

- Either an agent or a human user is submitting a query for the discovery of *Information Objects* contained by the Digital Library. The system responds back to the user request with a set of *Information Objects* that comply with the specified selection criteria.
- A user updates the information related to his/her profile that is maintained by the system.

C37 Access Resource

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

Relationships:

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

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

Examples:

- Ed submits a query for the discovery of information related to a specific subject (e.g., Picasso). The system discovers all the Information Objects matching Ed expectations (e.g., paintings and the biography of the famous painter) and returns these back to the user with a visualization of the corresponding content (e.g., a list of JPGs of all contained paintings and a PDF file with the biography).

C38 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:

- *Search* and *Browse* are two classical ways for performing the Discover function.

C39 Browse

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

Relationships:

- *Browse <isa> Discover*

Rationale: The *Browse* function allows an *Actor* to explore a digital library's *Resources* and may be used alternately with *Search* for this purpose. A digital library can be equipped with different *Browse* capabilities. For instance, it may provide a different ordering or grouping of *Resources*, such as browse per-author, when a *Collection* of publications is explored to search for the correct form of the name of an author, or through an ontology representing the underlying *Collection* of *Information Objects* or the set of permissible *Functions*. Alternatively, graphical representations of a *Resource Set* may be used for browsing DL *Resources*. For instance, it may be possible to have a digital library *Collection* depicted by using bubbles or areas of different size, each representing a certain topic, and then navigating among those bubbles in order to investigate on the content of each. Another example is that of a tag cloud,[32] i.e., a visual depiction of descriptors, namely tags, used to annotate Resources. Tags are typically listed alphabetically, and tag frequency is shown with font size or colour. The tags are usually hyperlinks that lead to a collection of items that are associated with that tag.

Examples:

- A user requests the display of all the Digital Library *Information Objects* (e.g., movies) per subject area (e.g., comedies, drama, sci-fi, etc);
- A user requests the display of all the Digital Library *Information Objects* per "author" in an alphabetical order.

C40 Search

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

Relationships:

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

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

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

characteristic of the *Search* functionality lies in which model is adopted in identifying the pertinence of the objects with respect to a query, e.g., the Boolean model or the vector-space model.

Examples:

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

C41 Acquire

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

Relationships:

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

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

Examples:

- A user downloads and locally stores *Information Objects* (e.g., video files, pdf files) from a list of *Information Objects* returned to him/her after performing a query.

C42 Visualise

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

Relationships:

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

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

Examples:

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

C43 Manage Resource

Definition: The class of *Functions* that supports the production, withdrawal or update, appraisal and preservation 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*
- *Appraise <isa> Manage Resource*
- *Preserve <isa> Manage Resource*

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

Examples:

- *Create*, *Update* and *Withdraw* are typical examples of *Manage Resource* functions.

C44 Create

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

Relationships:

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

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

Examples:

- An user registers within the system and creates a new profile for him/her.

C45 Submit

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

Relationships:

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

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

Examples:

- An author creates a record and uploads to the system his/her latest content e.g., a novel, movie, music recording.

C46 Withdraw

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

Relationships:

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

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

Examples:

- A user removes his/her profile from the system;
- An author removes his/her content from the system.

C47 Update

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

Relationships:

- *Update <isa> Manage Resource*

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

Examples:

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

C48 Preserve

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

Relationships:

- *Preserve <isa> Manage Resource*

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

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

Examples: --

C49 Validate

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

Relationships:

- *Validate <isa> Manage Resource*

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

Examples:

- A user rates the quality of the Digital Library Information Objects e.g., quality of multimedia files;
- A user rates the relevance of returned results to a submitted query.

C50 Annotate

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

Relationships:

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

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

Examples:

- A user inserts notes to a specific *Information Object*;
- A user tags *Information Objects* with a list of relevant labels.

C51 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: --

C52 Disseminate

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

Relationships:

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

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

Examples: --

C53 Publish

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

Relationships:

- *Publish <isa> Disseminate*

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

Examples:

- A Librarian approves/authorizes the publishing of a newly inserted *Information Object* to the system.

C54 Author

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

Relationships:

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

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

Examples:

- A musician creates a record for his/her latest music recording;
- A user creates a new profile.

C55 Compose

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

Relationships:

- *Compose <isa> Author*

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

Examples:

- A user combines two previous music recordings for the production of a new music file.

C56 Process

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

Relationships:

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

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

Examples:

- Analyze and Transform are two typical examples of this process.

C57 Analyse

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

Relationships:

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

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

Examples:

- Several types of analysis could be performed upon an Information Objects e.g., Linguistic, Statistical.

C58 Linguistic Analysis

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

Relationships:

- *Linguistic Analysis <isa> Analyse*

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

Examples:

- A typical example is grammar/spelling and syntactical analysis.

C59 Qualitative Analysis

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

Relationships:

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

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

Examples:

- Typical examples are bit-error-ratio, signal/noise ratio in audio/video files, file integrity, etc.

C60 Examine Preservation State

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

Relationships:

- *Examine Preservation State <isa> Qualitative Analysis*

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

Examples: --

C61 Statistical Analysis

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

Relationships:

- *Statistical Analysis <isa> Analyse*

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

Examples:

- E.g., statistics related to the accessing of contained information objects; user related statistics such geographical dispersion etc.

C62 Scientific Analysis

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

Relationships:

- *Scientific Analysis <isa> Analyse*

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

Examples: --

C63 Create Structured Representation

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

Relationships:

- *Create Structured Representation <isa> Analyse*

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

Examples: --

C64 Compare

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

Relationships:

- *Compare <isa> Analyse*

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

Examples:

- The system performs comparison between similar *information objects* (e.g., texts or multimedia files) to discover their degree of similarity.

C65 Transform

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

Relationships:

- *Transform <isa> Process*

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

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

Examples: --

C66 Physically Convert

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

Relationships:

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

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

Examples:

- Convert a 3D representation of a map to a 2D representation.

C67 Translate

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

Relationships:

- *Translate <isa> Physically Convert*

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

Examples:

- Translation of content from one language (e.g., English) to another (e.g., French).

C68 Convert to a Different Format

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

Relationships:

- *Convert to a Different Format <isa> Transform*

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

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

Examples:

- Information object transformation from a specific format to another one e.g., for text based information from MS-Word to PDF format, for video content from AVI to MPEG4 format.

C69 Extract

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

Relationships:

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

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

Examples: --

C70 Manage Actor

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

Relationships:

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

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

Examples: --

C71 Establish Actor

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

Relationships:

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

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

Examples: --

C72 Register

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

Relationships:

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

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

Examples:

- A new user called 'Ed' registers his profile within the system.

C73 Sign Up

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

Relationships:

- *Sign Up <isa> Register*

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

Examples: --

C74 Login

Definition: An *Establish Actor* function that enables an *Actor* to establish his/her identity in the DL and related rights.

Relationships:

- *Login <isa> Establish Actor*

Rationale: *Login* is performed by matching a set of qualities or characteristics that uniquely identify an *Actor*. Assurance of identification can be increased by a number of practices appropriate to the need. These practices range from passwords to tokens, smart cards, and public keys with Certificates. The system then performs authentication, and may also perform authorisation of the user. The execution of this *Function* should be regulated by *Policies*.

Examples:

- User Ed enters his authentication credentials, e.g., username and password, so as to be identified by the system.

C75 Personalise

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

Relationships:

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

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

Examples:

- User Ed states his preference in viewing only multimedia material/content and avoiding textual information objects.

C76 Apply Profile

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

Relationships:

- *Apply Profile <isa> Personalise*

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

Examples: --

C77 User Profiling

Definition: The *Manage Actor Function* that creates or updates the *Actor Profile*.

Relationships:

- *User Profiling <isa> Manage Actor*
- *Action Analysis <isa> User Profiling*
- *Sentiment Analysis <isa> User Profiling*
- *Explicit Declaration <isa> User Profiling*
- *User Profiling <create/update> Actor Profile*

Rationale: User Profiling is a Function that is used to create or update the Actor Profile.

Examples: --

C78 Action Analysis

Definition: The *User Profiling Function* that creates or updates the *Actor Profile* by relying on the *Action Log*.

Relationships:

- *Action Analysis <isa> User Profiling*

- *Action Analysis <create/update> Actor Profile*

Rationale: Action Analysis is a Function that is used to create or update the Actor Profile by relying on the Action Log.

Examples: --

C79 Sentiment Analysis

Definition: The *User Profiling Function* that creates or updates the *Actor Profile* by relying on natural language processing, computational linguistics, and text analytics to identify and extract subjective information from the *Resources* the *Actor* deals with, in particular those he/she owns.

Relationships:

- *Sentiment Analysis <isa> User Profiling*
- *Sentiment Analysis <create/update> Actor Profile*

Rationale: Sentiment Analysis is a Function that is used to create or update the Actor Profile by relying on the Action Log.

Examples: --

C80 Explicit Declaration

Definition: The *User Profiling Function* that creates or updates the *Actor Profile* by recording explicit statements and decisions made by the *Actor* to characterise himself/herself.

Relationships:

- *Explicit Declaration <isa> User Profiling*
- *Explicit Declaration <create/update> Actor Profile*

Rationale: Explicit Declaration is a Function that is used to create or update the Actor Profile. It relies on explicit declarations and statements the Actor creates to characterise himself/herself.

Examples: --

C81 Manage Function

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

Relationships:

- *Manage Function <isa> Manage Resource*

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

Examples: --

C82 Manage Policy

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

Relationships:

- *Manage Policy <isa> Manage Resource*

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

Examples: --

C83 Manage Quality Parameter

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

Relationships:

- *Manage Quality Parameter <isa> Manage Resource*

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

Examples:

- Define, update and withdraw quality parameters.

C84 Collaborate

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

Relationships:

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

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

Examples: --

C85 Exchange Information

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

Relationships:

- *Exchange Information <isa> Collaborate*

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

Examples:

- User 'A' submits a new *information object* that is published by the DL operator and accessed by another user (i.e., user 'B').

C86 Converse

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

Relationships:

- *Converse <isa> Collaborate*

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

Examples:

- User 'A' is able to submit messages e.g., instant messages or posts with other users.

C87 Find Collaborator

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

Relationships:

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

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

Examples: --

C88 Author Collaboratively

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

Relationships:

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

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

Examples:

- User 'A' and user 'B' are able to jointly edit an information object i.e., a textual description, meta-information, etc.

C89 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: --

C90 Manage Content

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

Relationships:

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

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

Examples: --

C91 Manage Collection

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

Relationships:

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

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

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

Examples:

- A librarian called 'Nick' requests the mass import of information entities contained in a file;
- The DL Administrator called 'Phil' decides to export the contents of the DL so as to migrate it to another instance of the DLS.

C92 Import Collection

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

Relationships:

- *Import Collection <isa> Manage Collection*

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

Examples:

- Harvesting the content of an OAI-PMH (Lagoze & Van de Sompel, 2001) compliant Repository through the underlying protocol is a kind of *Import Collection*.

C93 Export Collection

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

Relationships:

- *Export Collection <isa> Manage Collection*

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

Examples:

- Having the DL service compliant with the OAI-PMH (Lagoze & Van de Sompel, 2001) is a possible implementation of the *Export Collection* function.

C94 Manage User

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

Relationships:

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

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

Examples: --

C95 Manage Membership

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

Relationships:

- *Manage Membership <isa> Manage User*

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

Examples: --

C96 Manage Group

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

Relationships:

- *Manage Group <isa> Manage User*

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

Examples:

- Permissions are granted/revoked to a specific user group.

C97 Manage Role

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

Relationships:

- *Manage Role <isa> Manage User*

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

Examples:

- To facilitate the operations of a new group of users the DL administrators create a new user Role with appropriate rights and set of supported operations.

C98 Manage Actor Profile

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

Relationships:

- *Manage Role <isa> Manage User*

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

Examples: --

C99 Manage Functionality

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

Relationships:

- *Manage Functionality <isa> Manage DL*

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

Examples:

- A new form of user login needs to be added. This form of login will be based on the use of smart cards and this entails the addition of a new form of login functionality to the system.

C100 Monitor Usage

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

Relationships:

- *Monitor Usage <isa> Manage Functionality*

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

Examples: --

C101 Manage Quality

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

Relationships:

- *Manage Quality <isa> Manage DL*

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

Examples: --

C102 Manage Policy Domain

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

Relationships:

- *Manage Policy Domain <isa> Manage DL*

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

Examples: --

C103 Manage & Configure DLS

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

Relationships:

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

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

Examples: --

C104 Manage DLS

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

Relationships:

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

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

Examples: --

C105 Create DLS

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

Relationships:

- *Create DLS <isa> Manage DLS*

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

Examples: --

C106 Withdraw DLS

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

Relationships:

- *Withdraw DLS <isa> Manage DLS*

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

Examples: --

C107 Update DLS

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

Relationships:

- *Update DLS <isa> Manage DLS*

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

Examples: --

C108 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: --

C109 Manage Architectural Component

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

Relationships:

- *Manage Architectural Component <isa> Manage Architecture*

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

Examples: --

C110 Configure Architectural Component

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

Relationships:

- *Configure Architectural Component <isa> Manage Architecture*

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

Examples: --

C111 Deploy Architectural Component

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

Relationships:

- *Deploy Architectural Component <isa> Manage Architecture*

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

Examples: --

C112 Monitor Architectural Component

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

Relationships:

- *Monitor Architectural Component <isa> Manage Architecture*

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

Examples: --

C113 Configure DLS

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

Relationships:

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

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

Examples: --

C114 Configure Resource Format

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

Relationships:

- *Configure Resource Format <isa> Configure DLS*

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

Examples: --

C115 Configure Content

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

Relationships:

- *Configure Content <isa> Configure DLS*

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

Examples: --

C116 Configure User

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

Relationships:

- *Configure User <isa> Configure DLS*

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

Examples: --

C117 Configure Functionality

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

Relationships:

- *Configure Functionality <isa> Configure DLS*

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

Examples: --

C118 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: --

C119 Configure Quality

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

Relationships:

- *Configure Quality <isa> Configure DLS*

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

Examples: --

C120 Policy Domain

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

Relationships:

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

Rationale: The term *Policy* usually refers to a set of principles that describe the acceptable processes and/or procedures within an organisation.[33] *Policy Domain* affects how the complete system is designed and how it functions. This means that *Policies* should be incorporated in the *Architecture Domain*, implemented in the *Functionality Domain* and should be clear to *Actors* as they affects Actors' work with the *Content Domain* and influence their perception of the *Quality Domain*.

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

From the *Digital Library* perspective, *Policies* mean conditions, rules, terms and regulations governing the interaction between *Actors* and the *Digital Library*. They provide mechanisms to constrain operations that *Actors* may/may not perform in the context of the Digital Library on individual *Resources* at a given time. *Policies* at the *Digital Library* level reflect the management goals of the institution providing the *Digital Library* and should influence, rather than be influenced by, technical architecture, functionality, quality or information content.

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

[33] Digital libraries represent the confluence of vision, mandate and the imagined possibility of content and services constructed around the opportunity of use. Underpinning every digital library is a policy framework. It is the policy framework that makes the digital library viable - without a policy framework a digital library is little more than a container for content - even the mechanisms for structuring the content within a traditional library building as container (e.g., deciding what will be on what shelves where) are based upon policy. So policy governs how a digital library is instantiated and run; a library without policy therefore is similar to a Ferrari in a world without roads and populated only by blind drivers. The policy domain is therefore a meta-domain which is situated both outside the digital library and any technologies used to deliver it and with in the digital library. That is, policy exists as an intellectual construct that is deployed to frame the construction the digital library and its external relationships and then these and other more operational policies are represented in the functional elements of the digital library. Policy permeates the digital library from conceptualization through to operation.

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

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

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

While *Policy Domain* is a general term conceived to capture any kind of *Policy* and Policy-related feature in the Digital Library universe, specific 'rules', 'conditions', 'terms or 'regulations' within a single area are captured through the *Policy* concept and are manifested through a document which usually consists of policy statement, rationale, enforcement, responsible office (*Policy <expressedBy> Information Object*).

Examples: --

C121 Policy

Definition: A condition, rule, term or regulation governing the operation of any *Digital Library* `system'.

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*
- *Context <hasPart> Policy*

Rationale: A *Policy* regulates *Actors* exploiting *Resources* through *Functions* with respect to a validity interval (*Time Domain* can be used here). Each *Policy* is conceived to regulate a specific 'area', for example *Registration Policy* or *Preservation Policy*.

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

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

Examples:

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

C122 Extrinsic Policy

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

Relationships:

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

Rationale: *Extrinsic Policy* is a *Policy* imposed by a body outside the *Digital Library* (e.g., legal and regulatory frame works). According to the type of *Digital Library*, the regulatory framework might differ – a *Digital Library* 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.
- Accreditation Policy is an example of Extrinsic Policy, when the DL System is subjected to a certification process.

C123 Intrinsic Policy

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

Relationships:

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

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

Examples:

- A *Policy* within the *Policy* of the respective Digital Library is an *Intrinsic Policy*.
- Documentation of software specifications, codes and comments is routinely carried out using a production database where all staff is required to archive documents.

C124 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 the managing organisation wishes the users of the DL to interact with the DL. The implementation of an *Explicit Policy* at the Digital Library Management System level corresponds to the definition of *Actors* (and *Roles*) potential capabilities, i.e., to the definition of which kind of Resources can be exploited by a 'kind of' *Actor*.

Examples:

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

C125 Implicit Policy

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

Relationships:

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

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

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

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

Examples:

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

C126 Prescriptive Policy

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

Relationships:

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

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

Examples:

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

C127 Descriptive Policy

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

Relationships:

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

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

Examples:

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

C128 Enforced Policy

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

Relationships:

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

Rationale: An *Enforced Policy* is a *Policy* developed, deployed and strictly used in the DL.

Monitoring and reporting tools are necessary to follow up how the *Policy* is being applied.

Examples:

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

C129 Voluntary Policy

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

Relationships:

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

Rationale: *Voluntary Policy* basically means a *Policy* that is followed according to the decision of the *Actor*. This is valid for all *Policies* for which application is a matter of choice. In some cases, users may comply with *Policies* that are not officially communicated within the particular digital library – perhaps based on their previous experience with other digital libraries.

Examples:

- The *Collection Development Policy* might be outlined in broad terms, but not enforced in practice.

C130 System Policy

Definition: A *Policy* that concerns an aspect of a system as a whole, be it a *Digital Library*, a *Digital Library System* or a *Digital Library Management System*

Relationships:

- *System Policy <isa> Policy*
- *Change Management Policy <isa> System Policy*
- *Connectivity Policy <isa> System Policy*
- *Support Policy <isa> System Policy*
- *Resource Management Policy <isa> System Policy*

Rationale: This is a class of *Policies* governing generic processes within the digital library system in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- *System Policies* cover most general processes in any digital library `system', such as regulation of changes or management of resources.

C131 Change Management Policy

Definition: The purpose of the *Change Management Policy* is to regulate how changes are being carried out on the three levels and within the six domains of a digital library in a rational and consistent manner that would be effectively communicated to the *Actors* and would not harm their routine work.

Relationships:

- *Change Management Policy <isa> Policy*
- *Resource <regulatedBy> Change Management Policy*
- *Change Management Policy <govern> Manage DL Function*
- *Change Management Policy <govern> Manage DLS Function*
- *Change Management Policy <govern> Manage Resource*

Rationale: The aim of *Change Management Policy* in the DL is to ensure stability in the process of restructuring and assure coherence of actions on the three levels (DL, DLS and DLMS). The complexity of the DL could be approached when, in the process of change management, the issues relevant to the six basic areas – i.e., the issues of *Information Objects* in the *Content Domain*, the issues of *Actors* in the *User Domain*, the issues of *Functions* in the *Functionality Domain,* the issues of *Policies* in the *Policy Domain*, the issues of *Quality Parameters* in the *Quality Domain* and the issues of *Architectural Components* in the *Architecture Domain* – and the three levels (DL, DLS and DLMS) are addressed in a rational and consistent manner.

It is of the utmost importance to define roles and responsibilities in change management, and to consider in detail the change management process and the support the DLS and DLMS should provide for its smooth execution.

Examples:

- *Quality Parameter Measures* that demonstrate the change management progress may be part of the *Change Management Policy*.

C132 Resource Management Policy

Definition: *Policies* defining how *Resources* in the DL are allocated.

Relationships:

- *Resource Management Policy <isa> Policy*
- *Resource Management Policy <isa> System Policy*
- *Resource Management Policy <govern> Resource*

Rationale: Resource management is a key area within the organisation and use of *Resources* in the DL. *Resource Management Policy* is the *Policy* that describes the principles and procedures related to this field.

Since *Resources* may be of a different nature, this *Policy* would usually be a combination of different actions and procedures.

Examples:

- Checking the consistency of *Resource Identifiers* may be a task from the *Resource Management Policy*.
- A Digital Library organisation defines the use of Version Identification Framework (VIF), which provides guidance and solutions for repository managers, content creators and software developers about identifying versions of any type of digital object.
- A Resource Management Policy has been created by the organisation's Digital Librarian for supporting the evaluation, promotion, and management of all print, electronic and media information resources.
- A range of resources are stored by the Digital Library or held in separate locations and referenced by the Digital Library. The Resource Management Policy might define that access to some resources is controlled, all items are individually tagged with differing rights permissions and conditions, and whenever possible resources held by the organisation's Digital Library are made available under the terms of defined Creative Commons licences.[34]

C133 Support Policy

Definition: *Policies* describing the kinds of support *Actors* can expect when using the DL system and the *Resources* it contains.

Relationships:

- *Support Policy <isa> Policy*
- *Support Policy <isa> System Policy*
- *Support Policy <isa>* (should be) *Explicit Policy*
- *Support Policy <isa>* (should be) *Descriptive Policy*
- *Support Policy <isa>* (should be) *Intrinsic Policy*
- *Support Policy <govern> Actor*
- *Resource <regulatedBy>Support Policy*

Rationale: *Support Policy* refers to the technical and educational support on issues arising from the exploitation of a *Digital Library Management System*. In this case, the *Support Policy* should clearly describe what services are offered. Sometimes it is also helpful to include a list of excluded services.

[34] This example is valid also for C136 Content Policy and C138 Collection Development Policy.

Support Policy should be explicit (*Explicit Policy*), descriptive (*Descriptive Policy*) and intrinsic (*Intrinsic Policy*).

In some circumstances, policies related to support might be prescriptively enforced (*Prescriptive Policy* and *Enforced Policy*).

The procedure to be followed in order to request a service, and the conditions for its provision (charges, prioritising in the case of several simultaneous requests, and timing) should be clearly expressed in the *Support Policy*.

Examples:

- Priorities (critical requests receive higher priority than standard requests) may be a component of *Support Policy*.
- People increasingly and daily use a variety of computing devices not all of which are continuously connected to a network. This growing proliferation of palm devices, hand-held computers, disconnected laptops, and embedded processors (e.g., "smart" mobile telephones) offer opportunities for the creation of personalized information spaces – digital libraries with collections and services that correspond to targeted needs and situations. The Organisation's Digital Library Connectivity Policy supports individuals in exploiting the mobility of these devices in terms of capability and connectivity, and using them for storage, access, and update of selected information resources when network access is impractical or impossible.
- Especially in the context of public and national libraries, the organisation's Digital Library Connectivity Policy defines key policy issues that include the nature of sufficient bandwidth and broadband, the perpetuation of the digital divide of Internet access in libraries, the role of libraries as e-government access points, the complexities of funding Internet access, the impacts and contradictions of filtering, and the effect of homeland security legislation.[35]

C134 Connectivity Policy

Definition: *Policy* assuring maximum access to DL *Resources*.

Relationships:

- *Connectivity Policy <isa> Policy*
- *Connectivity Policy <isa> System Policy*
- *Connectivity Policy <govern> Actor*

Rationale: Connectivity can be defined as an organisation's capacity for communicating with itself and with its global environment through the use of ICT.

Connectivity Policy should promote all means that enable *Actors* from various environments to access *Resources*. The DL should be accessible via various communication channels, including mobile devices.

Examples:

- The digital divide is one of the threats that can be addressed within the *Connectivity Policy.*

C135 Risk Management Policy

Definition: A *Policy* that explains the approach within the DL towards various identified risks, the likelihood of their occurrence and the strategy for risk management.

[35] This example is valid also for C136 Content Policy.

Relationships:

- *Risk Management Policy <isa> System Policy*
- *Risk Management Policy <isa> Intrinsic Policy*

Rationale: This *Policy* should identify and provide an evaluation of, and correcting actions for, the risks within the six DL domains.

Examples:

- Good risk management would contribute to increasing *Quality Parameters* and in particular the *Trustworthiness* of the DL.
- The digital library organisation created a policy to pursue a structured approach to the effective management of risk in pursuit of its business objectives. This approach and the framework for its achievement is set out in the policy, which covers the continuous process of integrated activities by which the potential impact of risks to the achievement of organisation's objectives are managed. The Risk Management Policy indicates to adopt good practices and toolkits such DRAMBORA in the identification, evaluation and cost effective control of risks to ensure that they are eliminated where possible, reduced to an acceptable level or managed and contained; and to embed risk management practices within management and planning activities. It is the responsibility of Directors and Managers at all levels of the organisation to ensure that risks are understood and appropriately managed in accordance with this policy. At all levels of the organisation, risk management, reporting and auditing processes will reflect the requirements set out in the Risk Management Policy.

C136 Content Policy

Definition: *Policy* regulating the *Content* domain.

Relationships:

- *Content Policy <isa> Policy*
- *Disposal Policy <isa> Content Policy*
- *Collection Delivery Policy <isa> Content Policy*
- *Collection Development Policy <isa> Content Policy*
- *Digital Rights Management Policy <isa> Content Policy*
- *Preservation Policy <isa> Content Policy*
- *Submission and Resubmission Policy <isa> Content Policy*

Rationale: This is a class of *Policies* that govern processes related to the *Content* domain within the Digital Library 'system' in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- The issues of strategic planning and development of the *Content* of a Digital Library are addressed in the *Collection Development Policy*.
- The Content Policy of an organisation can be oriented to allow harvesting of metadata, in order to allow creation of layers of services as OAI Harvesters or Google Scholar services useful to citation analysis too. These metadata and also these contents are catch up and use by commercial sellers (e.g., Scopus of Elsevier) which offer to users the OA contents inside their databases.

C137 Disposal Policy

Definition: *Policy* concerning de-accession of DL material.

Relationships:

- *Disposal Policy <isa> Policy*
- *Disposal Policy <isa> Content Policy*
- *Disposal Policy <isa>* (may be a) *Prescriptive Policy*
- *Disposal Policy <isa>* (may be a) *Descriptive Policy*
- *Disposal Policy <govern> Actor*

Rationale: *Policies* defining de-accession of DL material (any *Information Object*) from the DL collections. They are both prescriptive (*Prescriptive Policy*) and descriptive (*Descriptive Policy*).

Examples:

- De-accession of a *Resource* that has not been requested for a certain period of time is part of a *Disposal Policy*.

C138 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* is *<regulatedBy> Collection Development Policy*
- *Information Object* is *<regulatedBy> Collection Development Policy*
- *Collection* is *<regulatedBy> Collection Development Policy*

Rationale: The institution(s) taking care of the DL development make their vision on the further development of the DL publicly available through their *Collection Development Policy*.

These intentions may reflect different aspects – for example, number of *Resources*, *Resource sets*, *Collections*. *Collection Development Policy* can also affect issues of subject areas, genres, data formats, or services related to better use of the *Collection* and adding value to its *Content*.

Basically, they should describe:

- access to what *Resources* are provided and how *Resources* will be enriched over time – in the short-, mid- and long-term future.
- information on formats, encodings, and recommendations for use of software tools for consulting *Resources* (*Resource Formats*).
- guidance on handling or tracking new *Editions*.

Collection Development Policies are of help to *Actors* in comparing different DLs in terms of what they offer and how relevant they are to their purposes.

Collection Development Policies can assist institutions developing DLs as they help to find and demonstrate the unique standing of particular DLs.

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

Examples:

- Estimation of current coverage of a DL is part of the *Collection Development Policy*.
- Acquisition Policy is an example of Collection Development Policy. The Acquisition Policy is an official statement detailing the types of materials the library accepts and the terms that affect the

acquisition of materials. The policy provides guidance for library staff as well as organizations considering donating their materials. Acquisition policies typically address elements such as the scope, how duplicate materials are handled, specific collection themes desired, an overview of how their appraisal process works, why a library may refuse a collection, and approximately how often the Acquisition Policy is reviewed.

- Appraisal is an example of Collection Development Policy. Appraisal is the process of assigning a value to records for the purpose of determining whether a library should accession them or not. It also can be used to determine how long records should be maintained. Establishing an appraisal policy allows a library to utilize a specific, consistent process when assessing the merits of a collection.[36]

C139 Collection Delivery Policy

Definition: *Policy* encompassing the constraints affecting how *Collections* will be delivered, under what conditions and for what purposes.

Relationships:

- *Collection Delivery Policy <isa> Policy*
- *Collection Delivery Policy <isa> Content Policy*
- *Collection Delivery Policy <govern> Actor*
- *Resource <regulatedBy> Collection Delivery Policy*
- *Collection Delivery Policy <govern> Browse Function*
- *Collection Delivery Policy <govern> Visualise Function*

Rationale: *Collection Delivery Policy* covers methods of providing access to the DL – through Internet services, removable memory, stand-alone computers, mobile devices, print on demand services.

The *Collection Delivery Policy* should also specify the conditions for the delivery (free of charge or paid; conditions for purchase of single items or through use licenses).

The *Collection Delivery Policy* may also specify the acceptable uses of *Resources*.

Examples:

- Purchasing a DVD with selected *Resources*.
- Offering a print-on-demand service for selected *Resources*.
- Defining the conditions for commercial use of images from illuminated manuscripts.
- Announcing free use of material for education and research purposes.

C140 Submission and Resubmission Policy

Definition: *Policies* regulating submission and resubmission of *Resources* to the DL.

Relationships:

- *Submission and Resubmission Policy <isa> Policy*
- *Submission and Resubmission Policy <isa> Content Policy*
- *Submission and Resubmission Policy <isa> Explicit Policy*

[36] This appraisal example is valid also for C137 Disposal Policy.

- *Submission and Resubmission Policy <isa> Prescriptive Policy*
- *Submission and Resubmission Policy <isa> Intrinsic Policy*

Rationale: *Submission and Resubmission Policies* govern which *Actors* can submit and resubmit *Information Objects* to the DL.

They should be explicit (*Explicit Policy*), prescriptive (*Prescriptive Policy*) and intrinsic (*Intrinsic Policy*).

Time constraints may be part of a *Submission and Resubmission Policy*.

Examples:

- *Actors* may have the right to submit but not to edit *Resources*.
- Policies regarding the annotations are Submission and Resubmission Policies. They can define the modalities within which a registered user views the annotations of the documents from other registered users; or how alerting software (RSS feeds) notify registered users that the documents have been annotated.[37]

C141 Digital Rights Management Policy

Definition: *Policy* that explains what technologies control how *Conten*t is used within the DL.

Relationships:

- *Digital Rights Management Policy <isa> Policy*
- *Digital Rights Management Policy <isa> Content Policy*
- *Digital Rights Management Policy <isa> User Policy*
- *Digital Rights Management Policy <isa> Functionality Policy*
- *Digital Rights Management Policy <govern> Function*
- *Digital Rights Management Policy <govern> Configure User Function*
- *Digital Rights Management Policy <govern> Digital Rights*

Rationale: Digital rights management (DRM) is the technological framework which should guarantee persistent access and use restrictions to *Content Resources,* i.e., *Information Objects*. *Digital Rights Management Policy* is the *Policy* explaining how the DL manages digital rights from the perspective of both the content creator/originator/owner and the *Actor*, and which technologies control the use of *Content* within the DL. While DRM regulates the types of actions that can be performed with information (for example, view, save, print, modify certain *Manifestation*), *Digital Rights Management Policy* explains DRM.

Digital rights management has been developed so that copyright holders on digital content have exclusive rights of copyright (e.g., the right to make a copy or the right to distribute a work to the public). Copyright holders, however, cannot control how digital content is used (e.g., the right to view, save, print, read or modify a work). Traditional library materials are better protected from unauthorised use because of their 'physical' nature. The development of digital content along with electronic publishing and the Internet, which gives access to *Manifestation* of the *Resources* in the digital environment, is creating new issues in the area of copyright regulations.

[37] This example is valid also for C136 Content Policy, C145 User Policy, C146 User Management Policy and C147 Registration Policy.

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

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

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

From the *Digital Library Management System* perspective, the emphasis is on the capabilities to implement the elements of the *Digital Rights Management Policy*. This means that the system should be capable of tracing certain actions undertaken by the user and of reacting correctly.

Examples:

- The *Actors* belonging to a specific *Group* can view *Resources* but not save local copies.
- Viewing *Resources* expires within a specific period of time.
- A DRM Policy can specify that the withdrawal of a paper deposited into a institutional repository is not allowed within a prescribed set of conditions.

C142 Digital Rights

Definition: *Policy* defining the rights of use of *Information Objects*.

Relationships:

- *Digital Rights <isa> Policy*
- *Digital Rights <isa> Content Policy*
- *Digital Rights <isa> Descriptive Policy*
- *Digital Rights <govern> Information Object*

Rationale: *Digital Rights* define the specific rights of use of digital objects. In this sense, they are a *Descriptive Policy* regulating the possible uses of *Information Objects*. The practical implementation of the *Digital Rights* falls within *Digital Rights Management Policy*.

A broader understanding of *Digital Rights* defines them as all human rights that are affected by technology, including the rights to use computers, communication networks and resources.

Examples:

- The right to access knowledge is affected by digital technology and not all people have equal opportunities in this respect.
- The right to use without a license is another example.

C143 License

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

Relationships:

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

Rationale: A *License* is the agreement by which the owner of intellectual property permits its use. In digital libraries, a *License* may be issued for specific uses of *Resources*, or for designated functionality features that should be downloaded and installed by the users.

Examples:

- GPL (GNU General Public License), a popular license for free software; GNU LGPL (Lesser General Public License); and BSD (Berkeley Software Distribution or Berkeley System Distribution) are examples of software licenses.

C144 Preservation Policy

Definition: *Policy* defining the approach to preservation taken by the DL.

Relationships:

- *Preservation Policy <isa> Policy*
- *Preservation Policy <isa> Content Policy*
- *Resource <regulatedBy> Preservation Policy*

Rationale: *Preservation Policy* prescribes how to implement actions assuring long-term preservation of *Resources*, such as decision making on archival needs, archiving practices, timing issues, access to archived materials, subsequent preservation measures for already archived materials, maintaining preservation metadata, issues of interoperability of preserved materials.

Examples:

- Reuse of preserved materials is part of the *Preservation Policy*.
- The Reference Model for an Open Archival Information System (ISO14721:2003, 2003) contains a set of *Preservation Policy*.
- The conditions applied to the storage over time of digital objects are part of the *Preservation Policy* of a Digital Library. E.g., for an academic Digital Library which manages an institutional repository some conditions are necessary to plan what would happen to the stored digital papers when/if the repository stops operation.[38]
- Digital Libraries and Archives need to replicate (or backup) their content both for access continuity and as part of a preservation strategy, when that is a requirement of the library. Technically, there are many options for how to do that, including local or remote, accessible of access-controlled, identical or different formats, with or without associate metadata, and so on. These choices should be specified by the library's policy. A Digital Preservation Policy could state that there must exist three copies of each Item, one stored locally and two stored in different jurisdictions. These copies must remain synchronized.[39]

C145 User Policy

Definition: *Policy* regulating the *User domain*.

Relationships:

- *User Policy <isa> Policy*

[38] This example is also valid also for C138 Collection Development Policy.

[39] This example is also valid also for C138 Collection Development Policy.

- *Digital Rights Management Policy <isa> User Policy*
- *User Management Policy <isa> User Policy*
- *Acceptable User Behaviour Policy <isa> User Policy*
- *Personalisation Policy <isa> User Policy*
- *Privacy and Confidentiality Policy <isa> User Policy*
- *Access Policy <isa> User Policy*

Rationale: This is a class of *Policies* governing processes related to the *User domain* within the Digital Library 'system' in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- All *Policies* that regulate issues regarding digital rights and user behaviour.
- The User Policy may explain the Digital Library's online information practices and the choices that the user can make about the information he/she share with the Digital Library. The policy also expresses how user's personal information is handled.

C146 User Management Policy

Definition: *Policy* defining how user management is handled.

Relationships:

- *User Management Policy <isa> Policy*
- *User Management Policy <isa> User Policy*
- *User Management Policy <govern> Actor*

Rationale: The *User Management Policy* makes it possible to execute *Functions* such as issuing, managing, changing, sharing accounts; administration rights; sharing resources between multiple users.

Examples:

- Account management is part of the *User Management Policy.*

C147 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*.

C148 Personalisation Policy

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

Relationships:

- *Personalisation Policy <isa> Policy*
- *Personalisation Policy <isa> User Policy*
- *Personalisation Policy <govern> Actor*
- *Personalisation Policy <govern> Personalise Function*

Rationale: The *Personalisation Policy* has two roles; on the one hand it makes it possible to recognise the user and his/her access rights, and on the other hand it enables the DL to serve its *Actors*, guaranteeing better *Quality Parameters* by offering *Information Objects* (generally *Resources*) that are in line with user preferences. In the DLS the *Functions* used to assure personalisation are *Apply Profile*, *Customise*, *Login* and *Subscribe*.

Examples:

- The choice of representation layout based on statistics of user behaviour is an example of *Personalisation Policy*.
- The possibility to use alerting tools like RSS feeds every time a Resource has been annotated is regulated by a Personalisation Policy

C149 Privacy and Confidentiality Policy

Definition: A *Policy* outlining the terms by which the organisation that manages the DL will handle personal information on its *Actors*.

Relationships:

- *Privacy and Confidentiality Policy <isa> Policy*
- *Privacy and Confidentiality Policy <isa> User Policy*

Rationale: *Policies* prescribing *Actor* details from application and enrolment information through to actor interaction data will be handled by the DL and the organisation that manages the DL.

Typically, the DL should only maintain personal information on *Actors* that is relevant to its better functioning and services.

Data about the *Actors* could be entered directly by them (e.g., user names, passwords) or obtained automatically (e.g., IP address).

The personal data collected should be protected against unauthorised access, destruction, misuse, modification, improper disclosure and loss.

Different rules may be applied to the use of various types of personal information, e.g., e-mail addresses, postal address, log-in names and passwords, users' opinions entered via web pages.

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

Examples:

- The use of the e-mail addresses of the *Actors* to announce new DL collections may be justified as a part of the *Privacy and Confidentiality Policy*.
- Selling or sharing with other organisations lists of e-mail addresses of the *Actors* is typically not in line with the *Privacy and Confidentiality Policy*, unless the users have agreed to this.

C150 Acceptable User Behaviour Policy

Definition: *Policy* covering how the *Actors* may or may not interact with the DL.

Relationships:

- *Acceptable User Behaviour Policy <isa> Policy*
- *Acceptable User Behaviour Policy <isa> User Policy*

Rationale: *Acceptable User Behaviour Policy* presents rules and regulations for appropriate use of the DL content and services, prescribing what a user can do and what he/she should refrain from doing.

Examples:

- Regulations on copying material from a DL are part of the *Acceptable User Behaviour Policy*.
- Rules for citation of the source of material from a DL are part of the *Acceptable User Behaviour Policy*.
- Rules on downloading images of workstations for within-institutional use of a DL are part of the *Acceptable User Behaviour Policy*.

C151 Functionality Policy

Definition: *Policy* regulating the *Functionality* domain.

Relationships:

- *Functionality Policy <isa> Policy*
- *Access Policy <isa> Functionality Policy*
- *Security Policy <isa> Functionality Policy*

Rationale: This is a class of *Policies* governing processes related to the *Functionalit*y domain within the Digital Library 'system' in its entirety on the three levels (DL, DLS and DLMS).

Examples:

- Taking care of the security of the Digital Library is a serious concern, for which the practical implementation would be a *Security Policy*.

C152 Access Policy

Definition: *Policy* regulating permission or denial of use of *Resources* by *Actors* in any *Digital Library* 'system'.

Relationships:

- *Access Policy <isa> Policy*
- *Access Policy <isa> User Policy*
- *Access Policy <isa> Functionality Policy*
- *Charging Policy <isa> Access Policy*

Rationale: *Access Policy* regulates the use of *Resources* (permission or denial of use) by *Actors*. It should guarantee that *Resources* are accessed by their intended *Actors* and not by others who might harm them unintentionally or deliberately. *Access Policy* belongs to both *Functionality* and *User* domains, as on the one hand it prescribes what *Functions* are possible, and on the other hand regulates the work of the *Actors*.

Examples:

- Access to *Resources* provided on the basis of IP address identification is an example of *Access Policy*.

C153 Charging Policy

Definition: *Policy* defining how charging schemes will be implemented and handled by the DL.

Relationships:

- *Charging Policy <isa> Policy*
- *Charging Policy <isa> Access Policy*
- *Charging Policy <govern> Actor*

Rationale: The *Charging Policy* explains what mechanisms are applied for collecting payments.

There are various models that could be applied: services provided on the basis of a longer time period; micro-payments; exchange of use of content for uploading user's own content into the DL.

Examples:

- Institution has unlimited access to all high-quality images stored in a DL based on an annual fee. *Actors* not coming from such an institution only have access to low-quality images.

C154 Security Policy

Definition: *Policy* regulating how a system provides security and protects *Resources* within the DL.

Relationships:

- *Security Policy <isa> Policy*
- *Security Policy <isa> Functionality Policy*
- *Resource <regulatedBy> Security Policy*

Rationale: *Security Policies* address the protection of the Digital Library. They implement the rules and tools that assure the security of services and integrity of the Digital Library.

Examples:

- Ingest of *Resources* into the library on the basis of virus checking is an example of *Security Policy*.
- The Security Policy of the Digital Library defines measures to protect its collections and assets from theft and deliberate or reckless damage, and to protect all its assets from unauthorized intrusion and vandalism.[40]

C155 Quality Domain

Definition: One of the six main concepts characterising the Digital Library universe. It captures the aspects that permit considering digital library `systems' from a quality point of view, with the goal of judging and evaluating them with respect to specific facets. 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*

[40] This example is valid also for C144 Preservation Policy.

Rationale: The *Quality Domain* concept represents the various facets used to characterise, evaluate and measure *Digital Libraries*, *Digital Library Systems*, *Digital Library Management Systems* and their *Resources* from a quality point of view. *Digital Library, Digital Library System* and *Digital Library Management System* <*tenders*> a certain level of *Quality Parameters* to its *Actors*, which can be either implicitly agreed or explicitly formulated by means of a Quality of Service (QoS) agreement.

Examples: --

C156 Measurement

Definition: A process for computing and assigning a value (*Measure*) to a *Quality Parameter*.

Relationships:

- *Quality Parameter* is <*evaluatedBy*> *Measurement*
- *Subjective Measurement* <*isa*> *Measurement*
- *Objective Measurement* <*isa*> *Measurement*
- *Qualitative Measurement* <*isa*> *Measurement*
- *Quantitative Measurement* <*isa*> *Measurement*
- *Measure* is assigned according to *(<accordTo>) a Measurement*

Rationale: See *Quality Parameter*.

Examples:

- See Quality Parameter.

C157 Objective Measurement

Definition: A *Measurement* that is well-defined and does not depend on individual perception.

Relationships:

- *Objective Measurement* <*isa*> *Measurement*

Rationale: *Objective Measurements* could be obtained by taking measures and using an analytical method to estimate the quality achieved. They could also be based on processing and comparing measures between a reference sample and the actual sample obtained by the system.

The distinction between *Objective Measurement* and *Subjective Measurement* is due to the fact that *Quality Parameters* can involve measure methods that can either be independent of the subject who is conducting them or, on the other hand, express the viewpoint and perception of the subject.

Examples:

- Examples of objective factors related to the perception of audio recordings in a *Digital Library* are: noise, delay and jitter.

C158 Subjective Measurement

Definition: A *Measurement* based on, or influenced by, personal feelings, tastes or opinions.

Relationships:

- *Subjective Measurement* <*isa*> *Measurement*

Rationale: *Subjective Measurements* 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 Measurement* and *Subjective Measurement* is due to the fact that *Quality Parameters* can involve measure methods that can either be independent of the subject who is conducting them or, on the other hand, express the viewpoint and perception of the subject.

Examples:

- Examples of factors related to the subjective perception of audio recordings in a *Digital Library* are: listening quality, loudness, listening effort.

C159 Qualitative Measurement

Definition: A *Measurement* based on a unit of measure that is not expressed via numerical values.

Relationships:

- *Qualitative Measurement <isa> Measurement*

Rationale: *Qualitative Measurements* 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 measurements are exploratory while quantitative measurements usually play a confirmatory role. Methods of *Qualitative Measurement* that could be applied to a DL are direct observation; participant observation; interviews; auditing; case study; collecting written feedback.

Examples:

- The opinions of the users expressed in a DL forum or blog can be used as a source for *Qualitative Measurement* of important issues for the users (content analysis is one of the popular techniques for analysing texts).

C160 Quantitative Measurement

Definition: A *Measurement* based on a unit of measure that is expressed via numerical values.

Relationships:

- *Quantitative Measurement <isa> Measurement*

Rationale: *Quantitative Measurements* are based on collecting and interpreting numerical data. There is a wide range of statistical methods for their analysis.

Examples:

- *Quantitative Measurement* is applied when collecting data and calculating the mean time spent by users in locating content.

C161 Measure

Definition: The action of, and the value obtained by, measuring a *Quality Parameter* in accordance with a selected *Measurement*.

Relationships:

- *Quality Parameter <measuredBy> Measure*
- *Measure* is assigned according to (<*accordTo*>) a *Measurement*

Rationale: See *Quality Parameter*.

Examples:

- See Quality Parameter.

C162 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 Domain <consistOf> Quality Parameters*
- *Resource <hasQuality>* with respect to *Quality Parameter*
- *Actor <expressAssessment>* about *Resources* according to *Quality Parameters*
- *Quality Parameter* is *<evaluatedBy> Measurement*
- *Quality Parameter* is *<measuredBy> Measure*
- *Quality Parameter* is *<affectedBy> Resource*
- *Quality Parameter* is *<expressedBy> Information Object*
- *Generic Quality Parameter <isa> Quality Parameter*
- *Content Quality Parameter <isa> Quality Parameter*
- *Functionality Quality Parameter <isa> Quality Parameter*
- *User Quality Parameter <isa> Quality Parameter*
- *Policy Quality Parameter <isa> Quality Parameter*
- *Architecture Quality Parameter <isa> Quality Parameter*
- *Digital Library <tender> Quality Parameter*
- *Digital Library System <tender> Quality Parameter*
- *Digital Library Management System <tender> Quality Parameter*
- *Context <hasPart> Quality Parameter*

Rationale: *Quality Parameters* serve the purpose of expressing the different facets of *Quality Domain* and provide information about how and how well a *Resource* performs with respect to a particular viewpoint. They express the assessment of an *Actor*, be it human or not, about the *Resource* under examination. They can be evaluated according to different *Measures*, which provide alternative procedures for assessing different aspects of a *Quality Parameter* and assigning it a value. *Quality Parameters* are actually measured by a *Measurement*, which represents the value assigned to a *Quality Parameter* with respect to a selected *Measure*.

Note that the *Resource* under examination in a *Quality Parameter* can be either a singleton *Resource*, as in the case of the *Integrity* of an *Information Object*, or a *Resource Set*, as in the case of the *Orthogonality* of a set of *Functions*.

Finally, a *Quality Parameter* may be affected by other *Resources*, such as other *Quality Parameters*, *Policies* or *Functions*; this allows us to create a 'chain' of *Resources* which leads to the determination of the *Quality Parameter* in question. For example, *Availability* is affected by *Robustness* and *Fault Management*: in fact, when a *Function* is both robust and able to recover from error conditions, it is probable that its *Availability* is also increased. As a further example, *Economic Convenience* may be affected by *Charging Policy*, since the latter is responsible for the definition of the charging strategies.

Note that, being a *Resource*, a *Quality Parameter* may have *Metadata* and *Annotations* linked to it; the former can provide useful information about the provenance of a *Quality Parameter*, while the latter can offer the possibility to add comments about a *Quality Parameter*, interpreting the obtained values, and proposing actions to improve it.

Please note that the groupings of *Quality Parameters* in broad categories, such as *Content Quality Parameter*, are made from the perspective of the *Resources* under assessment, in the case of the example mainly *Information Objects*. This means that *User Quality Parameter* does not concern issues such as *User Satisfaction* or *Usability*, where the *Actor* is the subject who makes the assessment, but in this group the *Actor* is the object of the assessment from different points of view, such as *User Behaviour*. Nevertheless, the active role of an *Actor* in expressing an assessment is always preserved in the *Quality Parameter* by the fact the *Actor <expressAssessment>* about a *Resource* in each *Quality Parameter*.

The definition of *Quality Parameter* complies with the notion of quality dimension used in (Batini & Scannapieco, 2006) and (Gonçalves, Moreira, Fox, & Watson, 2007).

Examples:

- In order to clarify the relationship between *Quality Parameter*, *Measure* and *Measurement*, we can take an example from the information retrieval field. One of the main *Quality Parameters* in relation to an information retrieval system is its effectiveness, meaning its capability to answer user information needs with relevant items. This *Quality Parameter* can be evaluated according to many different *Measures*, such as precision and recall (Salton & McGill, 1983): precision evaluates effectiveness in the sense of the ability of the system to reject useless items, while recall evaluates effectiveness in the sense of the ability of the system to retrieve useful items. The actual values for precision and recall are *Measurements* and are usually computed using standard tools, such as trec_eval,[41] which are *Actors*, but in this case not human.

C163 Generic Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of a 'system' as a whole, be it a *Digital Library*, a *Digital Library System* or a *Digital Library Management System*.

Relationships:

- *Generic Quality Parameter <isa> Quality Parameter*
- *Reputation <isa> Generic Quality Parameter*
- *Economic Convenience <isa> Generic Quality Parameter*
- *Sustainability <isa> Generic Quality Parameter*
- *Security Enforcement <isa> Generic Quality Parameter*
- *Interoperability Support <isa> Generic Quality Parameter*
- *Documentation Coverage <isa> Generic Quality Parameter*
- *Performance <isa> Generic Quality Parameter*
- *Scalability <isa> Generic Quality Parameter*
- *Compliance With Standard <isa> Generic Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the 'system' in its entirety, in particular the *Digital Library*, the *Digital Library System* and the *Digital Library Management System*.

[41] http://trec.nist.gov/trec_eval/

Examples:

- A Digital Library operating in the research environment is going to be sold within the commercial market. Few big information providers are interested in buying it and need to assess it as a whole in order to negotiate the estimate. They will take primarily into account its Generic Quality Parameter, establishing the overall value and impact within its specific context.

C164 Economic Convenience

Definition: A *General Quality Parameter* reflecting how favourable the economic efficiency is when using a *Digital Library*.

Relationships:

- Economic Convenience <*isa*> *Generic Quality Parameter*
- Economic Convenience <*affectedBy*> *Charging Policy*

Rationale: This parameter evaluates the economic conditions for using the *Digital Library* in order to determine if they are sufficiently advantageous.

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

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

Examples:

- An institution may find it advantageous to pay a moderate subscription for offering access to standard functionalities to all of its users and then pay an extra amount of money for access to more advanced functionalities for a restricted set of users who actually need them.
- As another example, consider the possibility of paying a basic fee for subscription to a set of standard *Collections* of a *Digital Library* and pay on a per-*Information Object* basis when you access *Information Objects* belonging to a *Collection* you are not subscribed to.

C165 Interoperability Support

Definition: A *Generic Quality Parameter* reflecting the capability of a *Digital Library* to inter-operate with other *Digital Libraries*.

Relationships:

- *Interoperability Support* <*isa*> *Generic Quality Parameter*
- *Interoperability Support* is <*affectedBy*> *Connectivity Policy*
- *Interoperability Support* is <*affectedBy*> *Compliance to Standards*

Rationale: This parameter concerns the capability of interoperating with other *Digital Libraries* as well as the ability to integrate with legacy systems and solutions. As discussed in Section II.3, this is a very prominent issue in the Digital Library universe and this parameter can help in expressing the 'degree of interoperability' among *Digital Libraries* and/or *Resources*.

Connectivity Policy may affect *Interoperability Support* since it defines and controls how, and to what extent, a *Digital Library* should be accessible.

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

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

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

Examples:

- A relevant example of effort to offer interoperability at the data level is the OAI-PMH protocol[42] and the OAI-ORE initiative;[43] examples of interoperability efforts at the service level are the SRU/SRW[44] protocol and the Web Services.[45]

C166 Reputation

Definition: A *Generic Quality Parameter* reflecting the trustworthiness of a *Digital Library*.

Relationships:

- *Reputation <isa> Generic Quality Parameter*
- *Reputation* is *<affectedBy> Authenticity*
- *Reputation* is *<affectedBy> Trustworthiness*
- *Reputation* is *<affectedBy> Integrity*
- *Reputation* is *<affectedBy> Preservation Performance*
- *Reputation* is *<affectedBy> Documentation Coverage*
- *Reputation* is *<affectedBy> Usability*
- *Reputation* is *<affectedBy> Robustness*
- *Reputation* is *<affectedBy> Fidelity*
- *Reputation* is *<affectedBy> Viability*
- *Reputation* is *<affectedBy> Availability*
- *Reputation* is *<affectedBy> Dependability*
- *Reputation* is *<affectedBy> Fault Management Performance*

Rationale: *Reputation* concerns the 'good name' of a *Digital Library*, the credit it has gained from the user community, and its ability as a point of reference.

Other *Quality Parameters* may greatly affect the *Reputation* and we may consider it as a sort of overall indicator of the appreciation of a *Digital Library*.

Examples:

- Examples of aspects that influence the *Reputation* of a *Digital Library* are whether a *Digital Library* provides *Resources* that can be regarded as true, real, impartial, credible and conveying the right information.
- Examples of Quality Parameters that influence Reputation are: Economic Convenience, *Usability*, Dependability, and so on.

[42] http://www.openarchives.org/pmh/

[43] http://www.openarchives.org/ore/

[44] http://www.loc.gov/standards/sru/

[45] http://www.w3.org/2002/ws/

C167 Security Enforcement

Definition: A *Generic Quality Parameter* reflecting the level and kind of security features offered by a *Digital Library*.

Relationships:

- *Security Enforcement <isa> Generic Quality Parameter*
- *Security Enforcement <affectedBy> Digital Rights Management Policy*
- *Security Enforcement <affectedBy> Access Resource*
- *Security Enforcement <affectedBy> Configure DL*
- *Security Enforcement <affectedBy> User Behaviour*

Rationale: This parameter reflects the capability of the *Digital Library* to support the management of different levels of security as expected by users, content depositors, rights owners and librarians themselves.

Security Enforcement can be affected by both *Policies* and *Functions*. In particular, the *Digital Rights Management Policy* affects the level of *Security Enforcement* of a *Digital Library*, since it defines how the content has to be controlled. The *Access Resources* functions and their implementation influence *Security Enforcement*, since they provide *Actors* with mechanisms for consuming *Information Objects*; the *Configure DL* functions impact *Security*, since the possibility of correct and careful configuration of the *Digital Library* is a prerequisite for security; finally, *User Behaviour* can affect the *Security Enforcement*, since an *Actor* may compromise security, for example by careless use of username and password.

Examples:

- An example of a factor that influences Security Enforcement is the capability to prevent unauthorised access to content or the saving of local copies of copyrighted material. Within the Policy domain the regulations should be clearly stated in the Digital Rights Management Policy.

C168 Sustainability

Definition: A *Generic Quality Parameter* reflecting the prospects of durability and future development of a *Digital Library*.

Relationships:

- *Sustainability <isa> Generic Quality Parameter*
- *Sustainability <affectedBy> Change Management Policy*
- *Sustainability <affectedBy> Collection Development*
- *Sustainability <affectedBy> Compliance with Standards*
- *Sustainability <affectedBy> Maintenance*

Rationale: *Sustainability* should take into consideration various factors, such as the organisational and economic aspects of a *Digital Library*, as well as its capability of ensuring the preservation of its *Content* and of keeping pace with future innovations.

Sustainability may be affected by the *Policies* adopted by the *Digital Library*, such as the *Change Management Policy* or the *Collection Development Policy*.

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

Examples:

- Examples of factors that influence *Sustainability* are: the funding scheme that ensures the economic conditions for carrying on the *Digital Library*; the skills and willingness within the organisation that provides for the *Digital Library*; the presence of accurate development plans for the collections held by the *Digital Library*, as well as for the software and hardware resources needed for the *Digital Library System* and the *Digital Library Management System*.

C169 Documentation Coverage

Definition: A *Generic Quality Parameter* measuring the accuracy and clarity of the documentation describing a given *Resource*.

Relationships:

- *Documentation Coverage <isa> Generic Quality*

Rationale: This *Quality Parameter* addresses the quality of the written documentation of a *Resource*. The importance of documentation associated to *Resources* of any form is usually underestimated. On the contrary, having a valuable documentation reflects in an optimal usage of the available Resources.

Examples:

- Manuals explaining the use of *Function*s are typical examples of *Documentation Coverage*.
- Other examples are the accuracy of online help, better if contextual, or the selection provided by the Frequently Asked Question sections.

C170 Performance

Definition: A *Generic Quality Parameter* measuring the capabilities a *Resource* when observed under particular conditions.

Relationships:

- *Performance <isa> Generic Quality Parameter*
- *Performance* is *<affectedBy> Capacity*
- *Performance* is *<affectedBy> Robustness*
- *Performance* is *<affectedBy> Dependability*
- *Performance* is *<affectedBy> Fault Management Performance*
- *Performance* is *<affectedBy> Availability*
- *Performance* is *<affectedBy> Integrity*
- *Performance* is *<affectedBy> Size*
- *Performance* is *<affectedBy> Perceivability*
- *Reputation* is *<affectedBy> Viability*

Rationale: This *Generic Quality Parameter* provides an overall assessment of how well a *Resource* performs from different points of view, e.g., efficiency, effectiveness, efficacy and so on.

Examples:

- The response time upon invocation of a *Function* is an example of a generic *Performance* indicator.
- The presence of delays and/or jitter is an example of *Performance* indicators more tailored to the multimedia and streaming contexts.
- Precision and recall are widely used *Performance* indicators in the information retrieval field.

C171 Scalability

Definition: A *Generic Quality Parameter* measuring the capability of increasing *Capacity* as much as needed.

Relationships:

- *Scalability <isa> Generic Quality Parameter*
- *Scalability* is *<affectedBy> Size*
- *Scalability* is *<affectedBy> Load Balancing Performance*
- *Scalability* is *<affectedBy> Redundancy*
- *Scalability* is *<affectedBy> Maintenance Performance*
- *Scalability* is *<affectedBy> Capacity*
- *Scalability* is *<affectedBy> Availability*

Rationale: *Scalability* denotes the ability of a system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and/or to be susceptible to enlargement; it is a desirable attribute of a network, system or process. This is a very wide concept that affects many entities in the Digital Library universe and it is often difficult to define precisely and formally.

Examples:

- The ability of a DLS to support a growing number of users and/or provide access to (massively) growing collections without deterioration in performance.
- Another example is the ability to increase the number of requests served by a *Function* while keeping response time reasonable.

C172 Compliance with Standards

Definition: A *Generic Quality Parameter* measuring the degree to which standards have been adopted in developing a *Resource*.

Relationships:

- *Compliance with Standards <isa> Generic Quality Parameter*

Rationale: Standards represent one of the most common and well recognized approach to attack interoperability issues at any level and in any domain. This parameter captures the exploitation of standards while developing or implementing a *Resource*. Potentially, standards are everywhere, i.e., a standard can be exploited to develop every single aspect of a *Resource*. This parameter influences *Interoperability Support*, since the adoption of standards increases the ease of interoperation with other entities. It influences also the *Sustainability* of a *Digital Library*, since open standards support keeping the *Resource* up-to-date with future technological developments.

Examples:

- An Architectural Component implementing the Access Resource Function through the OAI-PMH protocol has an high Compliance with Standards Measurement.
- An Architectural Component implementing the Search Function through the SRU/SRW protocol has an high Compliance with Standards Measurement.
- A Metadata having Dublin Core and its Resource Format has an high Compliance with Standards Measurement.

C173 Content Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *Content* main concept.

Relationships:

- *Content Quality Parameter <isa> Quality Parameter*
- *Authenticity <isa> Content Quality Parameter*
- *Integrity <isa> Content Quality Parameter*
- *Provenance <isa> Content Quality Parameter*
- *Freshness <isa> Content Quality Parameter*
- *Preservation Performance <isa> Content Quality Parameter*
- *Size <isa> Content Quality Parameter*
- *Scope <isa> Content Quality Parameter*
- *Trustworthiness <isa> Content Quality Parameter*
- *Fidelity <isa> Content Quality Parameter*
- *Perceivability <isa> Content Quality Parameter*
- *Viability <isa> Content Quality Parameter*
- *Metadata Evaluation <isa> Content Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *Content*, in particular *Information Objects*, in a *Digital Library*.

Examples:

- Content quality is in a sense a moving target, but the requirements on the level of quality of various materials in the *Digital Library* and its scope have to be presented in the *Collection Development Policy*.

C174 Authenticity

Definition: A *Content Quality Parameter* reflecting whether an *Information Object* retains the property of being what it purports to be.

Relationships:

- *Authenticity <isa> Content Quality Parameter*

Rationale: The capability to measure to what extend an *Information Object* is actually 'what' it is declaring to be is fundamental in order to properly use it to produce/derive new knowledge. The definition takes into account the results and experience of the InterPARES I project[46].

Examples:

- The methods for data protection are key to assuring authenticity of *Resources*. Document sealing engines which timestamp and sign digitally every item in the *Digital Library* are an example of a solution that creates the proof that the documents have not been modified from the original.

[46] http://www.interpares.org/

C175 Trustworthiness

Definition: A *Content Quality Parameter* measuring the trustfulness and credibility of a *Resource* based on the reliability of the creator of the *Resource*.

Relationships:

- *Trustworthiness <isa> Content Quality Parameter*
- *Trustworthiness <affectedBy> Provenance*

Rationale: *Trustworthiness* concerns the reliability and believability of a given *Resource*, meaning the possibility of both placing the *Actor*'s trust in it and resting assured that the trust will not be betrayed. It may be helpful to compare digital libraries that have a similar or identical scope where one might be more trustworthy than the other.

Provenance may affect *Trustworthiness*, since knowing the lineage and history of a *Resource* may improve its reliability and credibility.

Examples:

- NISO Z39.7 Library Statistics and ISO 11620 Library Performance Indicators suggest measures of usage especially for libraries; in this context, *Trustworthiness* could be measured by estimating the number of visitors (general number or different users). Another possibility is to gather transaction information (number of downloads and printouts).
- After the ingestion of a digital object into a repository, Digital libraries can use digital signatures as a method to preserve the digital object trustworthiness.

C176 Freshness

Definition: A *Content Quality Parameter* measuring the *Information Object* quality of being current and promptly updated.

Relationships:

- *Freshness <isa> Content Quality Parameter*

Rationale: This parameter evaluates whether an *Information Object* and the information it carries are fresh and updated with respect to the task in hand.

Examples:

- A stream of data coming from a sensor that monitors the temperature and blood pressure of a patient should be updated at regular intervals in order to provide meaningful information for a physician.
- Another relevant example is a *Digital Library* keeping weather forecast information, where it is important to know if this information is updated and reflects the current weather conditions. *Information Objects* might be replicated in order to increase their availability. When a replicated *Information Object* is updated, these changes have to be propagated to all replicas. The *Freshness* value of a replica denotes how up-to-date it is, i.e., how many update operations on this *Information Object* are still outstanding.

C177 Integrity

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

Relationships:

- *Integrity <isa> Content Quality Parameter*

Rationale: This parameter encompasses the extent to which an *Information Object* is of sufficient breadth, depth and scope for the task in hand, as pointed out in (Batini & Scannapieco, 2006). Integrity expresses in what degree the content is complete and correct. The integrity of the content ensures the users that the documents they retrieve are the most appropriate ones. Integrity measurements can help Digital Libraries to assess the completeness and trustworthiness of their collections.

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.
- User A downloads an image file from a DL but he discovers it's not readable as the file is corrupted.

C178 Preservation Performance

Definition: The *Content Quality Parameter* is used to evaluate the need to undertake actions that would ensure that the digital resources will be accessible over the long term.

Relationships:

- *Preservation Performance <isa> Content Quality Parameter*

Rationale: The *Preservation Performance* parameter helps to monitor the need to apply digital curation actions to the separate resources, collections and *Digital Library* as a whole.

Examples:

- If the policy of the *Digital Library* is to make copies of content stored on DVDs every five years, a *Preservation Performance* parameter would help to comply with this requirement.

C179 Scope

Definition: A *Content Quality Parameter* measuring the areas of coverage of the *Content* and/or *Resources* of the *Digital Library*.

Relationships:

- *Scope <isa> Content Quality Parameter*

Rationale: The *Scope* parameter helps to understand the coverage of a *Digital Library* both in the sense of *Content* and in the sense of *Functionality*. While the *Size* provides quantitative insight, *Scope* is more qualitatively oriented.

Examples:

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

C180 Size

Definition: A *Content Quality Parameter* measuring the magnitude of *Resource*, *Collection* or a *Digital Library* as a whole.

Relationships:

- *Size <isa> Content Quality Parameter*
- *Size <isa> Quantitative Measure*

Rationale: Sizes can be provided according to different measures: for example, numbers of items, pages, bytes, articles, words, images, multimedia files. The evaluation of the size of a *Digital Library* helps the user to get an idea about the resources. *Size* is also an important parameter for the architecture and functionality of the DL.

Examples:

- The physical size of a collection calculated in bytes is important for estimating the migration effort.

C181 Fidelity

Definition: A *Content Quality Parameter* measuring the accuracy with which an electronic system reproduces a given *Resource*.

Relationships:

- *Fidelity <isa> Content Quality Parameter*

Rationale: The *Fidelity* parameter is used to evaluate to what degree a particular representation of a given *Resource* is different from its original representation.

Examples:

- The rendition of a text document may be identical to its original appearance in the word processing software used at the time of creating the document, but may significantly differ from its original appearance especially in layout – this difference is expressed through *Fidelity*.

C182 Perceivability

Definition: A *Content Quality Parameter* measuring the effort an *Actor* needs to invest in order to understand and absorb a *Resource*.

Relationships:

- *Perceivability <isa> Content Quality Parameter*

Rationale: The *Perceivability* parameter is used to evaluate how easily an *Actor* would understand and retain the information/knowledge within a *Resource* from the *Content* domain. This quality parameter is essential for evaluating which *Resources* are most likely to be well understood within a specific target group of users.

Examples:

- When numerous *Resources* in the *Digital Library* represent the same topic, perceivability may help to choose those that are most likely to be quickly understood. Quite often, images might be found to have higher perceivability than texts. Perceivability can also be used to answer the needs of special groups of users, for example providing audio content to visually impaired users.

C183 Viability

Definition: A *Content Quality Parameter* measuring whether the *Resource*'s bit stream is intact and readable with the existing technology.

Relationships:

- *Viability <isa> Content Quality Parameter*

Rationale: *Viability* is essential for preservation activities within a *Digital Library*. It would estimate whether a digital object could be read and manipulated with the existing hardware and software.

Examples:

- The minimum time specified by the supplier for the media's viability under prevailing environmental conditions.

C184 Metadata Evaluation

Definition: A *Content Quality Parameter* measuring characteristics of *Metadata*.

Relationships:

- *Metadata Evaluation <isa> Content Quality Parameter*

Rationale: *Metadata Evaluation* is essential for various processes in the *Digital Library*, and most specifically in tasks related to access, preservation and operability. According to a functionality-oriented definition of Guy, Powell and Day, 'high quality metadata supports the functional requirements of the system it is designed to support'. Metadata evaluation could be as simple as checking whether metadata (or specific metadata elements) are available, or it could be a more sophisticated evaluation of incomplete, inaccurate or inconsistent metadata elements. In the most detailed case, *Metadata Evaluation* would be a compound parameter consisting of several others – for example, Completeness, Accuracy, Provenance, Conformance to Expectations, Timeliness, User Satisfaction, Perceivability. This combination would depend on the purpose of the *Metadata Evaluation*.

Examples:

- Completeness in the context of *Metadata evaluation* could be used to measure whether a minimal required set of elements is available in the metadata records;
- A DL requires metadata evaluation to ensure that digital objects can be correctly identified, located and retrieved. Quality metadata is also essential for enabling the content of the Digital Library to be managed, and the access to that content. Compliance to appropriate standards facilitates the interoperability support parameter across Digital Libraries, which in turn facilitates the scalability parameter. Evaluation of the quality Digital Library's metadata should assess the support the metadata gives to each of the content quality parameters across the different classes of metadata – each of which is required to fulfil all the necessary functions.

  Metadata evaluation can vary according to the metadata classes:
  - Metadata structure standards
    - Are the chosen metadata standards in compliance with policy?
    - Are the chosen metadata standards appropriate for the discipline?
    - Do the chosen metadata standards support the Content Quality Parameters?
    - How closely are the standards complied with?
    - Do application profiles support the Content Quality Parameters and the purpose stated in the policy?
    - Are at the minimum Simple Dublin Core elements included, to enable harvesting using the OAI-PMH protocol?
    - Are there appropriate XML schemas for the chosen standards?
    - Are the standards chosen monitored for updates, additions and changes to community practice?
  - Metadata content standards
    - Is a persistent identifier used?
    - Are appropriate content standards used to ensure consistency?
    - Are there project specific content standards in use and how fit for purpose are these?
    - Are appropriate thesauri, word lists, ontologies or authority files used to ensure consistency?
    - Is their a set of rules for adding to thesauri, word lists, ontologies or authority files as new situations arise?
  - Metadata Creation

- To what extent have elements been completed?
- How closely have content standards been complied with?
- How closely have appropriate thesauri, word lists, ontologies or authority files been complied with
- Are automation tools available for technical metadata
- Are links between digital objects recorded correctly
- Can you afford to create all the metadata required?

C185 Functionality Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *Functionality* main concept.

Relationships:

- *Functionality Quality Parameter <isa> Quality Parameter*
- *Usability <isa> Functionality Quality Parameter*
- *User Satisfaction <isa> Functionality Quality Parameter*
- *Availability <isa> Functionality Quality Parameter*
- *Dependability <isa> Functionality Quality Parameter*
- *Robustness <isa> Functionality Quality Parameter*
- *Fault Management Performance <isa> Functionality Quality Parameter*
- *Capacity <isa> Functionality Quality Parameter*
- *Orthogonality <isa> Functionality Quality Parameter*
- *Awareness of Service <isa> Functionality Quality Parameter*
- *Expectations of Service <isa> Functionality Quality Parameter*
- *Impact of Service <isa> Functionality Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *Functionality*, in particular *Functions*, of a *Digital Library*.

Examples:

- *User* interacts with the *Digital Library System* using its functions, e.g., submitting a query to retrieve a set of digital objects. The system will retrieve the information according to the selection criteria specified in the query, which can be descriptive or semantic. The Functionality Quality Parameter determines the overall quality of this interaction.

C186 Availability

Definition: A *Functionality Quality Parameter* indicating the ratio of the time a *Function* is ready for use to the total lifetime of the system.

Relationships:

- *Availability <isa> Functionality Quality Parameter*
- *Availability <affectedBy> Robustness*
- *Availability <affectedBy> Fault Management*
- *Availability <affectedBy> Capacity*

Rationale: *Availability* is a fundamental parameter for assessing the quality of a *Function*, as *Actors* may be very disappointed when they try to use a *Function* and it is not available.

Availability may be affected by other parameters, such as *Robustness* and *Fault Management*: the former guarantees that a *Function* will continue to work and be available even in the case of bad input; the latter guarantees that a *Function* will be able to recover from an error condition and thus continue to be available. Finally, *Capacity* may also affect *Availability*, as, in the case of starvation of resources, a *Function* may stop being available.

Availability typically parallels *Dependability*.

Examples:

- In the telephone services, high levels of availability are demanded – the well-known 'five-nines', the 99.999% of uptime of the system – since nobody expects to pick up the receiver and not hear the signal.

C187 Awareness of Service

Definition: A *Functionality Quality Parameter* measuring how well the *Actors* of a *Digital Library* are aware of its existence and *Functions*.

Relationships:

- *Awareness of Service <isa> Functionality Quality Parameter*

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

Examples:

- *Awareness of Service* for target user groups is an important component of the current information literacy.
- Libraries build and offer information literacy online tutorials to increase the Awareness of Service. Qualitative methods help Digital Libraries to measure this parameter, such as online questionnaires.

C188 Capacity

Definition: A *Functionality Quality Parameter* representing the limit to the number of requests a *Function* can serve in a given interval of time.

Relationships:

- *Capacity <isa> Functionality Quality Parameter*
- *Capacity* is *<affectedBy> Scalability*
- *Capacity* is *<affectedBy> Redundancy*
- *Capacity* is *<affectedBy> Load Balancing Performance*

Rationale: *Capacity* determines how many concurrent requests can be served successfully.

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

Examples:

- The number of *Information Objects* that an information access component can index in a certain unit of time is an example of *Capacity*, as is the maximum number of users that can connect to the portal of a *Digital Library* at the same time.

C189 Expectations of Service

Definition: A *Functionality Quality Parameter* measuring what *Actors* believe a *Function* should offer.

Relationships:

- *Expectations of Service <isa> Functionality Quality Parameter*

Rationale: The *Expectations of Service* from the point of view of the digital library service can be clarified through user agreements on the Quality of Service (QoS), which outline the actual service and the existing framework to the user. However, users might have different expectations based on their experience with other DLs or other digital services. User expectations could be studied through surveys.

Examples:

- Users expect that clicking on an image thumbnail will open up a larger size and higher quality image file.

C190 Fault Management Performance

Definition: A *Functionality Quality Parameter* measuring the ability of a *Function* to react to and recover from failures in a transparent way.

Relationships:

- *Fault Management Performance <isa> Functionality Quality Parameter*
- *Fault Management Performance <affectedBy> Robustness*

Rationale: *Fault Management Performance* reflects the capacity of a *Function* to recover from error conditions, thus avoiding the interruption of the service provided.

It may be affected by *Robustness*, meaning the capacity to recover from faulty inputs.

Examples:

- Consider the case of a *Function* that crashes due to some problem but is able, during its functioning, to save its state and seamlessly restart from the last valid state.
- As a further example, consider the capability of switching to another *Architectural Component* with similar capabilities if the one being used stops working.

C191 Impact of Service

Definition: A *Functionality Quality Parameter* measuring the influence that the service offered by a *Function* has on the *Actor*'s knowledge and behaviour.

Relationships:

- *Impact of service <isa> Functionality Quality Parameter*

Rationale: The user of *Digital Libraries* does not have static skills; in the ideal case, his or her knowledge is increased and the practical skills of exploring digital collections are improved over time. This parameter has special importance if we consider the applications of digital libraries in the educational area, in particular e-Learning applications using *Digital Libraries*.

Examples:

- The user who has experience with a specific visual interface will generally be able to use another similar interface. Since the user has mastered how to use a specific set of functionalities organised in a particular interface, his expectation of service is also different.

C192 Orthogonality

Definition: A *Functionality Quality Parameter* indicating to what extent different *Functions* are independent of each other, i.e., do not affect each other.

Relationships:

- *Orthogonality <isa> Functionality Quality Parameter*

Rationale: *Orthogonality* measures whether sets of *Functions* are independent of each other. DLs with full functional orthogonality, or at least pronounced orthogonality, will usually be much more intuitive for their users than DLs with a high degree of functional overlap.

Orthogonality may affect *Usability* and may also affect *User Satisfaction*, when the usage of the DL might become too complicated.

Examples:

- In a well designed Digital Library, Functions having different scope, e.g., Manage Information Object and manage Actor, should have an high degree of Orthogonality, e.g., the Actor performing them should perceived the differences and the effects of them.
- The Orthogonality of Manage Resource and Manage Information Object is low since the latter is a special kind of the former.

C193 Dependability

Definition: A *Functionality Quality Parameter* measuring the ability of a DL to perform a *Function* under stated conditions for a specified period of time.

Relationships:

- *Dependability <isa> Functionality Quality Parameter*
- *Dependability* is *<affectedBy> Capacity*

Rationale: *Dependability* reflects whether a given *Function* works correctly without producing errors.

Capacity may affect *Dependability*, since in the case of starvation of resources a *Function* may not work properly.

Examples:

- When an *Actor* types the URL of a portal that gives access to a *Digital Library*, he/she expects the address to be correctly resolved and to be redirected to the correct site and not to an incorrect one.

C194 Robustness

Definition: A *Functionality Quality Parameter* measuring the resilience to ill-formed input or incorrect invocation sequences of a *Function*.

Relationships:

- *Robustness <isa> Functionality Quality Parameter*

Rationale: *Robustness* is a key parameter that may affect other *Quality Parameters*, such as *Security Enforcement* or *Availability*. Indeed, many kinds of attack that compromise the functioning of a service or gain unauthorised access to services are based on ill-formed input, such as buffer overflows.

Examples:

- Consider the capacity of preventing buffer overflows, which are often exploited to gain unauthorised access to a system.

C195 Usability

Definition: A *Functionality Quality Parameter* that indicates the ease of use of a given *Function*.

Relationships:

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

Rationale: *Usability* records to what extent a given *Function* makes it easy for an *Actor* to achieve its goals.

It can be evaluated by using different *Measures*: for example, the *Actor* can indicate on a subjective scale the degree of *Usability* of a *Function*; alternatively, the time needed to complete a task can be measured.

Examples:

- *Usability* concerns many different aspects of a *Digital Library*, ranging from the user interface, the facility in finding and accessing relevant information, the presentation of search results, to support for facilitating complex or difficult tasks, such as the provision of query-by-example functionalities or browsing and navigation facilities for complex metadata schemas or ontologies.

C196 User Satisfaction

Definition: A *Functionality Quality Parameter* indicating to what extent an *Actor* is satisfied with a given *Function*.

Relationships:

- *User Satisfaction <isa> Functionality Quality Parameter*
- *User Satisfaction <affectedBy> Usability*
- *User Satisfaction <affectedBy> Expectations of Service*
- *User Satisfaction <affectedBy> Documentation Coverage*
- *User Satisfaction <affectedBy> Performance*
- *User Satisfaction <affectedBy> Availability*
- *User Satisfaction <affectedBy> Dependability*
- *User Satisfaction <affectedBy> Orthogonality*

Rationale: The *User Satisfaction* parameter reflects to what extent an *Actor* is satisfied by the capabilities offered by a given *Function*. Many factors can influence *User Satisfaction*, such as *Usability*, *Expectations of Service*, *Documentation Coverage*, *Performance*, *Availability*, *Dependability* and so on.

Examples:

- *User Satisfaction* can be explicitly assessed by making use of surveys and questionnaires where the user's opinion is explicitly requested, or it may be implicitly deduced by observing how much a given *Function* is used and preferred over other similar ones.

C197 User Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *User Domain* main concept.

Relationships:

- *User Quality Parameter <isa> Quality Parameter*
- *User Behaviour <isa> User Quality Parameter*

- *User Activeness <isa> User Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *User Domain*, in particular *Actors*, of a *Digital Library*.

Examples:

- How and how much users interact with Digital Libraries. E.g., e-journals usage statistics give information on the number of monthly downloads and on the format preferred (HTML or PDF).

C198 User Activeness

Definition: A *User Quality Parameter* that reflects to what extent an *Actor* is active and interacts with a *Digital Library*.

Relationships:

- *User Activeness <isa> User Quality Parameter*

Rationale: This parameter concerns whether and how much an *Actor* is active with respect to the *Content* and *Functionality* offered by a *Digital Library*.

Examples:

- Factors that influence this parameter are, for example, whether an *Actor* frequently contributes his own *Content* to the *Digital Library* or whether an *Actor* often participates in discussions with other *Actors*, perhaps by using *Annotations*.

C199 User Behaviour

Definition: A *User Quality Parameter* that reflects how an *Actor* behaves and interacts with a *Digital Library*.

Relationships:

- *User Behaviour <isa> User Quality Parameter*

Rationale: This parameter concerns whether and how much an *Actor* abides by the *Policies* and regulations of a *Digital Library*.

Examples:

Factors that influence this parameter are, for example, whether an *Actor* respects the copyright on the *Resources* of a *Digital Library* or if he/she makes unauthorised copies of such material.

C200 Policy Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the top-level *Policy* concept.

Relationships:

- *Policy Quality Parameter <isa> Quality Parameter*
- *Policy Consistency <isa> Policy Quality Parameter*
- *Policy Precision <isa> Policy Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of a set of *Policies*.

Examples:

- A DL gives access to a digital collection including the multimedia works of a living artist. These works are protected by copyright; however the DL doesn't provide a specific policy that clearly states the artist's collection limitations and terms of use.

C201 Policy Consistency

Definition: A *Policy Quality Parameter* that characterises the extent to which a set of *Policies* are free of contradictions.

Relationships:

- *Policy Consistency <isa> Policy Quality Parameter*

Rationale: This parameter concerns whether or not a set of *Policies* (each of them well defined) are free of contradictions. Because of the fact *Policies*, being *Resources*, might be composed of 'sub'-Policy, this *Quality Parameter* captures also the case of *Policies* whose parts are inconsistent.

Examples:

- *Digital Rights* is a policy regulating rights of use of digital objects. *Digital Rights Management Policy* governs the *Functions* that implement rights issues in the use of *Resources*. These two policies have to be consistent in their approach to rights issues.

C202 Policy Precision

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

Relationships:

- *Policy Precision <isa> Policy Quality Parameter*

Rationale: *Architecture*, *Functionality* and the underlying technologies need to be well understood when designing DL *Policies*. A lack of knowledge of the technology used may lead to undesired DLS behaviour. Since *Digital Libraries* are such a complex field, we would like to stress the importance of understanding the reasons that cause unexpected behaviour. It might be the fault of the *Policy*, if aspects it should govern have not been envisaged in the necessary detail (in this case, precision of policy is not sufficient). Other causes of deviant behaviour might be found in insufficient knowledge of technology, or inadequate reflection of architecture or software in the policy design. Because of the fact *Policies*, being *Resources*, might be composed of 'sub'-Policy, this *Quality Parameter* captures also the case of Policies whose parts are defined in a precise way.

Examples:

- A policy limiting the rate of sending data over a network cannot be enforced in a DL if the underlying DLS does not provide some means for adjusting the data transmission rate; this could be of special importance in very large digital libraries or for institutions that have limited resources and need to keep the bandwidth consumption low.
- A policy is precise when it is detailed and defined enough to deal properly with its consequences. The co-operation between DLs implies the support of a wide range of policies, i.e., policies can be defined to constrain many different behaviours. Successful co-operations will make compromises based on providing sufficient generality to define most useful policies but enough limitations to make efficient and reliable enforcement feasible.

C203 Architecture Quality Parameter

Definition: A *Quality Parameter* that concerns an aspect of the *Architecture Domain* main concept.

Relationships:

- *Architecture Quality Parameter <isa> Quality Parameter*
- *Redundancy* <isa> *Architecture Quality Parameter*

- *Ease of Administration <isa> Architecture Quality Parameter*
- *Load Balancing Performance <isa> Architecture Quality Parameter*
- *Ease of Installation <isa> Architecture Quality Parameter*
- *Log Quality <isa> Architecture Quality Parameter*
- *Maintenance Performance <isa> Architecture Quality Parameter*
- *Compliance to Standards <isa> Architecture Quality Parameter*

Rationale: This is a family of *Quality Parameters* reflecting the variety of facets that characterise the quality of the *Architecture Domain*, in particular *Architectural Components*, of a *Digital Library System*.

Examples:

- A System Administrator is considering the possibility to change the DLMS software since the one currently exploited to realise the DL is characterised by *Architecture Quality Parameters* (e.g., Maintenance Performance) that hinder the evolution of the DL in line with the expectations (e.g., the deployment of a new System Component impose an overall DL downtime).

C204 Ease of Administration

Definition: An *Architecture Quality Parameter* measuring the presence and ease of use of tools for configuring, administering and monitoring *System Architecture Components*.

Relationships:

- *Ease of Administration <isa> Architecture Quality Parameter*

Rationale: The presence of good administration tools is crucial for configuring and monitoring the functioning of complex and distributed systems, which *Digital Library Systems* potentially are.

Examples:

- A DLS which supports dynamic (re-)configuration by adding or removing *Software Components* without the need to recompile the system after each change.
- The presence of automatic procedures for installing software and patches in a networked and distributed context, or of tools for informing and alerting administrators in the case of malfunctioning are another example of factors that influence the *Ease of Administration*.

C205 Ease of Installation

Definition: An *Architecture Quality Parameter* measuring the ease of installation and configuration of *Software Components*.

Relationships:

- *Ease of Installation <isa> Architecture Quality Parameter*

Rationale: The *Ease of Installation* parameter concerns the presence of tools and procedures for seamlessly installing and deploying *Software Components*, as well as adding new *System Architecture Components* to an operating *Digital Library System*.

Examples:

- The presence of intuitive wizards for installing new components or the possibility of adding components without restarting the whole system are examples of factors that influence *Ease of Installation*.

C206 Load Balancing Performance

Definition: An *Architecture Quality Parameter* measuring the capacity to spread and distribute work evenly across *System Architecture Components*.

Relationships:

- *Load Balancing Performance <isa> Architecture Quality Parameter*

Rationale: *Load Balancing Performance*, together with *Redundancy*, may help in improving the overall performance and responsiveness of a *Digital Library System*.

Examples:

- For a DLS on top of a Grid environment, which takes into account several instances of *Architectural Components*, *Load Balancing Performance* includes the ability of the system to distribute requests equally among different components of the same type within the system. In particular, this capability consists in selecting *Hosting Nodes* according to their workload or moving a job from one *Hosting Node* to another in order to achieve optimal *Resource* utilisation so that no *Resource* is over/under-utilised.

C207 Log Quality

Definition: An *Architecture Quality Parameter* measuring the presence and accuracy of logs which monitor the activity and functioning of *System Architecture Components*.

Relationships:

- *Log Quality <isa> Architecture Quality Parameter*

Rationale: The presence of accurate logs is crucial for understanding, analysing, debugging and improving the functioning of a *Digital Library System*.

Furthermore, log analysis can be an effective means of understanding *Actor* behaviour and personalising the *Digital Library System* accordingly; therefore, logs can provide useful input for the *Personalise* functions and for creating *Actor Profiles*.

Examples:

- There are various standards for creating logs. For example, in the case of the Web, there is W3C Extended Log Format.

C208 Maintenance Performance

Definition: An *Architecture Quality Parameter* addressing the design and implementation of software and hardware maintenance plans for *Architectural Components*.

Relationships:

- *Maintenance Performance <isa> Architecture Quality Parameter*
- *Maintenance Performance <affectedBy> Change Management Policy*

Rationale: *Maintenance Performance* concerns the design of plans for keeping *Architectural Components* updated with research and technological advances.

Change Management Policy may affect *Maintenance Performance*, since it regulates the change process in a *Digital Library*.

It may influence *Sustainability*, as it involves keeping the current system functioning properly and evolving it to face future technological developments.

Examples:

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

C209 Redundancy

Definition: An *Architecture Quality Parameter* measuring the degree of (partial) duplication of *System Architecture Components* to decrease the probability of a system failure.

Relationships:

- *Redundancy <isa> Architecture Quality Parameter*

Rationale: A redundant architecture helps in improving the overall performance of a system and may improve the *Availability*, *Dependability* and *Robustness* of a *Digital Library System*.

Examples:

- Availability of a system can be increased by *Redundancy* of *Architectural Components*. In the event that one component fails, another component of the same type is able to take over.

C210 Architecture Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents the various aspects related to the software systems that concretely realise the Digital Library universe.

Relationships:

- *Digital Library <definedBy> Architecture Domain*
- *Digital Library System <definedBy> Architecture Domain*
- *Digital Library Management System <definedBy> Architecture Domain*
- *Architecture Domain <consistOf> Architectural Component*

Rationale: The *Architecture Domain* encompasses concepts and relationships characterising the two software systems that play an active role in the DL universe, i.e., DLSs and DLMSs. Unfortunately, the importance of this fundamental concept has been largely underestimated in the past. The importance of the domain and its modelling is described in Section II.2.7.

Examples: --

C211 Architectural Component

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

Relationships:

- *Architectural Component <isa> Resource*
- *Architectural Component <yield> Function*
- *Architectural Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Architectural Component* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Architectural Component <hasProfile> Component Profile*
- *Architectural Component <conformTo> Framework Specification*
- *Architectural Component <use> Architectural Components*
- *Architectural Component <composedBy> Architectural Components*

- *Architectural Component <conflictWith> Architectural Components*
- *Architectural Component <has> Interface*
- *Software Architecture Component <isa> Architectural Component*
- *System Architecture Component <isa> Architectural Component*

Rationale: The notion of *Component* has been introduced in modern software systems to represent 'elements that can be reused or replaced'. By exploiting such an approach, systems gain the potential to be:

- flexible – users' needs change over time, even while the system is being developed. It is important to be able to apply changes to the system at a later stage. Moreover, it should be possible/easy to fix the bugs;
- affordable – both to buy and to maintain. Reuse and replacement features of the component-oriented approach contribute to reducing 'costs'.

An *Architectural Component* is a *Resource* in the Digital Library universe. In particular, this kind of *Resource* becomes relevant in the context of *Digital Library Systems* and *Digital Library Management Systems*, which are responsible for concretely realising the *Digital Library*. As a *Resource* to be managed, such components should have a description, i.e., *Component Profile*, characterising them and promoting their correct usage. This description may assume diverse forms ranging from human-oriented description, e.g., a textual description in natural language, to a machine-understandable one, e.g., WSDL, as in the case of Web Services. Neither statements nor constraints are imposed on the *Component Profile* associated with each *Architectural Component*.

Examples:

- *Architectural Components* are classified in two main categories: *Software Architecture Components* and *System Architecture Components*. These components are the constituents of a *Software Architecture* and *System Architecture* respectively. Examples of *Software Architecture Components* and *System Architecture Component* are presented in the respective sections.

C212 Software Architecture Component

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

Relationships:

- *Software Architecture Component <isa> Architectural Component*
- *Software Architecture Component <isa> Resource* (inherited from *Architectural Component*)
- *Software Architecture Component <yield> Function* (inherited from *Architectural Component*)
- *Software Architecture Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Software Architecture Component* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Software Architecture Component <hasProfile> Component Profile* (inherited from *Architectural Component*)
- *Software Architecture Component <conformTo> Framework Specification* (inherited from *Architectural Component*)
- *Software Architecture Component <use> Software Architecture Components* (inherited from *Architectural Component*)
- *Software Architecture Component* is *<composedBy> Software Architecture Components* (inherited from *Architectural Component*)

- *Software Architecture Component <conflictWith> Software Architecture Components* (inherited from *Architectural Component*)
- *Software Architecture Component <has> Interface* (inherited from *Architectural Component*)
- *Software Component <isa> Software Architecture Component*
- *Interface <isa> Software Architecture Component*

Rationale: The notion of *Component* has been introduced in modern software systems to represent 'elements that can be reused or replaced'. The advantages of such an approach in implementing software systems are introduced in Section II.2.7 Architecture Domain.

This notion may have different manifestations in present-day systems. In particular, due to the fact that *Software Architecture Components* (being *Architectural Components*) may in turn be composed of smaller and smaller parts (<*composedBy*>), it is possible to model *Software Architecture Components* at different levels of abstraction. For instance, a *Software Architecture Component* implementing a Web Service responsible for providing a range of *Functions* may consist of smaller *Software Architecture Components* (usually logical components in which the whole service is organised), each implementing specific sub-tasks needed to carry out the expected component functions. Each of such smaller *Software Architecture Components* is in turn organised in packages and classes (smaller *Software Architecture Components*), effectively containing the code (program instructions, data structures, etc.) that implements a constituent piece of the main *Software Architecture Component*.

Examples:

- A service in a system following the Service Oriented Architecture.
- A software library, i.e., one or several files that either are necessary for the execution/running of the *Software Architecture Component* or add features to it once co-deployed on the same *Hosting Node*.
- A software package in object-oriented programming. It is a named group of related classes (another example of *Software Architecture Component*). Classes are groups of methods (set of instructions) and variables.

C213 Software Component

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

Relationships:

- *Software Component <isa> Software Architecture Component*
- *Software Component <isa> Architectural Component* (inherited from *Software Architecture Component*)
- *Software Component <isa> Resource* (inherited from *Architectural Component*)
- *Software Component <yield> Function* (inherited from *Architectural Component*)
- *Software Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Software Component <regulatedBy> Policy* (inherited from *Resource*)
- *Software Component <regulatedBy> License*
- *Software Component <hasProfile> Component Profile* (inherited from *Architectural Component*)
- *Software Component <conformTo> Framework Specification* (inherited from *Architectural Component*)
- *Software Component <use> Software Components* (inherited from *Architectural Component*)

- *Software Component* <*composedBy*> *Software Components* (inherited from *Architectural Component*)
- *Software Component* <*conflictWith*> *Software Components* (inherited from *Architectural Component*)
- *Software Component* <*has*> *Interface* (inherited from *Architectural Component*)
- *Software Component* <*implement*> *Interface*
- *Software Component* <*representedBy*> *Information Object*
- *Software Component* <*realisedBy*> *Running Component*

Rationale: The *Software Component* is the core of the component-oriented approach when applied to software systems. This approach promotes software reuse and replacement, and thus makes system development potentially inexpensive.

Examples:

- A Java class implementing a specific Function.

C214 Application Framework

Definition: A *Software Architecture Component* representing middleware, i.e., software that connects and supports the operation of other *Software Architecture Components* available at the *Hosting Nodes*. It provides the runtime environment for the *Running Component*.

Relationships:

- *Application Framework* <*isa*> *Software Component*
- *Application Framework* <*support*> *Running Component*
- *Application Framework* <*implement*> *Framework Specification*

Rationale: The middleware guarantees proper operation of *Architectural Components*. The application framework influences the way in which components are implemented. It must be provided before the deployment and configuration of the components. For instance, in the case of components relying on an application framework that offers a SOAP library, the components are implemented expecting that such a library is available on the *Hosting Node*.

Examples:

- Apache Tomcat (http://tomcat.apache.org/)

C215 Interface

Definition: A *Software Architecture Component* representing a set of methods and parameters implemented by an *Architectural Component (Software Component).* The client of such an *Architectural Component* may rely on them while interacting with it.

Relationships:

- *Interface* <*isa*> *Software Architecture Component*
- *Architectural Component* <*has*> *Interface*
- *Framework Specification* <*prescribe*> *Interface*
- *Component Profile* <*profile*> *Interface*
- *Software Component* <*implement*> *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 (Lagoze & Van de Sompel, 2001) prescribed the *Interface* an *Architectural Component* acting as an OAI compliant data provider must implement in order to serve an *Architectural Component* willing to act as an OAI application provider.

C216 Framework Specification

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

Relationships:

- *Framework Specification <isa> Software Architecture Component*
- *Architectural Component <conformTo> Framework Specification*
- *Application Framework <implement> Framework Specification*
- *Framework Specification <prescribe> Interface*

Rationale: The notion of *Framework Specification* is needed to capture the operational context in which an *Architectural Component* has been designed to operate.

Examples:

- Enterprise JavaBeans
- Component Object Model

C217 System Architecture Component

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

Relationships:

- *System Architecture Component <isa> Architectural Component*
- *System Architecture Component <isa> Resource* (inherited from *Architectural Component*)
- *System Architecture Component <yield> Function* (inherited from *Architectural Component*)
- *System Architecture Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *System Architecture Component <regulatedBy> Policy* (inherited from *Resource*)
- *System Architecture Component <hasProfile> Component Profile* (inherited from *Architectural Component*)
- *System Architecture Component <conformTo> Framework Specification* (inherited from *Architectural Component*)
- *System Architecture Component <use> Architectural Components* (inherited from *Architectural Component*)
- *System Architecture Component <composedBy> System Architecture Components* (inherited from *Architectural Component*)
- *System Architecture Component <conflictWith> System Architecture Components* (inherited from *Architectural Component*)
- *System Architecture Component <has> Interface* (inherited from *Architectural Component*)
- *Running Component <isa> System Architecture Component*

- *Hosting Node <isa> System Architecture Component*

Rationale: The notion of *Component* has been introduced in modern software systems to represent 'elements that can be reused or replaced'. The advantages of such an approach in implementing software systems are given in Section II.2.7 Architecture Domain as well as discussed in the *Architectural Component* definition.

Examples:

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

C218 Running Component

Definition: An *Architectural Component* realising a *Software Component*.

Relationships:

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

Rationale: The concrete realisation of the code captured by the notion of *Software Component* in a concrete hardware, i.e., it corresponds to the standard notion of 'software process'.

Examples:

- The operational web server implementing the user interface of the DELOS Digital Library. (http://www.delos.info)

C219 Hosting Node

Definition: A hardware device providing computational and storage capabilities such that (*i*) it is networked, (*ii*) it is capable of hosting components, and (*iii*) its usage is regulated by *Policies*.

Relationships:

- *Hosting Node <isa> System Architecture Component*
- *Running Component <hostedBy> Hosting Node*

Rationale: *Hosting Nodes*, being *System Architecture Components* (and thus *Architectural Components*), should be equipped with *Component Profiles* that represent their description. An example of the usage of such information is the automatic matchmaking process used to assign a *Software Component* to the most appropriate *Hosting Node* for its deployment (i.e., the creation of the *Running Component*) by relying on its descriptive characteristics.

Examples:

- The server equipped with the bundle of software needed to host and run the *Software Component* implementing the user interface of the DELOS Digital Library. (http://www.delos.info)

C220 Software Architecture

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

Relationships:

- *Software Architecture <consistOf> Software Architecture Component*

Rationale: Each software system is characterised by a set of software pieces organised in a structure that enables them to work together. This organised set of software is the *Software Architecture*. To help software engineers design their systems, a set of well-proven generic schemes for the solution of recurring design problems have been identified, i.e., *Software Architecture* patterns (Buschmann, Meunier, Rohnert, Sommerla, & Stal, 1996). Patterns capture existing, well-proven experience in software development and help to promote good design practice. The *Reference Architecture* envisaged in Section I.5 and constituting an important part of the Digital Library development framework is a pattern for *Digital Library Systems*. Similarly to patterns, it is important to recall that many *Reference Architectures* can be designed, each dealing with a specific and recurring problem in designing or implementing DLSs. Moreover, different *Reference Architectures* can be used to construct DLSs with specific properties.

Examples:

- Client-Server Architecture
- Service-oriented Architecture

C221 System Architecture

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

Relationships:

- *System Architecture* <*consistOf*> *System Architecture Component*

Rationale: Each software system is characterised by the set of its constituents. This Reference Model classifies the constituents of a software system along two dimensions, that of the *Software Architecture* and that of the *System Architecture*. The *System Architecture*, as an architecture, is an organised set of constituents. In this case, constituents are *System Architecture Components*, namely *Running Instances* and *Hosting Nodes*. Because of (*i*) the strong relations between *Running Instances* and *Software Components*, i.e., a *Running Component* is the result of the deployment of a *Software Component*, and (*ii*) the fact that *Software Components* are the main constituents of the *Software Architecture* of the system, there is a strong relation between *Software Architecture* and *System Architecture*. A *System Architecture* is one of the possible instances that are obtainable according to the *Software Architecture* of the system in use. It is well known that, by exploiting a software system developed according to a monolithic application pattern, it is not possible to realise a system with a distributed *System Architecture*. The more flexible the *Software Architecture* a system adopts, the larger will be the potential range of application scenarios that can be successfully exploited.

Examples:

- The set of servers and services realising the DELOS Digital Library. (http://www.delos.info)

C222 Purpose

Definition: The motivation characterising the <*associatedWith*> relationship.

Relationships:

- *Resource* <*associatedWith*> *Resource* to a certain *Purpose*

Rationale: The <*associatedWith*> relation is one of the powerful ones enabling the building of compound *Resources*, i.e., *Resources* obtained by combining existing constituent *Resources* so as to form a new knowledge bundle that has a value added with respect to the single *Resources* when considered as a single island of information. Various kinds of associations are possible, and this diversity is captured by the *Purpose* concept attached to each instance of the <*associatedWith*> relation.

Examples:

- An *Information Object* representing an experiment (itself composed of various *Information Objects* representing, for example, the dataset the experiment is carried on, the dataset representing the outcomes, the description of the procedure adopted) is <*associatedWith*> the *Information Object* representing the scientific publication in an outstanding Journal in the field with the <*Purpose*> of scholarly dissemination.

C223 Region

Definition: A contiguous portion of a given *Resource* with the desired degree of granularity identified in order to anchor a given *Annotation* to it.

Relationships:

- *Resource* <*hasAnnotation*> *Annotation* about a *Region*

Rationale: The idea of 'contiguous portion' of a *Resource* resembles and complies with the concept of segment introduced in (Navarro & Baeza-Yates, 1997). The granularity of such a kind of 'identifier' can vary according to the meaningful ways of locating a part of a *Resource*, which depend on the actual specialisation of the *Resource* we are dealing with. As a consequence, we can have *Regions* that anchor an *Annotation* to the whole *Resource*, as well as *Regions* that can anchor an *Annotation* to a specific part of a *Resource*.

Examples:

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

C224 Digital Library

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

Relationships:

- *Digital Library* <*manage*> *Resource*
- *Digital Library* <*manage*> *Information Object*
- *Digital Library* <*serve*> *Actor*
- *Digital Library* <*offer*> *Function*
- *Digital Library* <*agreeWith*> *Policy*
- *Digital Library* <*tender*> *Quality Parameter*
- *Digital Library System* <*support*> *Digital Library*
- *Digital Library* is <*definedBy*> *Resource Domain*
- *Digital Library* is <*definedBy*> *Content Domain*
- *Digital Library* is <*definedBy*> *User Domain*
- *Digital Library* is <*definedBy*> *Functionality Domain*
- *Digital Library* is <*definedBy*> *Policy Domain*
- *Digital Library* is <*definedBy*> *Quality Domain*

Rationale: Digital Library is a complex universe and usually the term 'digital library' is used with many different semantics. This Reference Model introduces three notions of systems (cf. Section I.2) active in this universe, i.e., *Digital Library*, *Digital Library System* and *Digital Library Management System*. The first is the most abstract of the three and represents the set of *Information Objects*, *Actors*, *Functions*, *Policy* and *Quality Parameters* forming the *Digital Library* and perceived by *End-users* as the service they can exploit. This service is supported by a running system, i.e., the *Digital Library System*.

Examples:

- The DELOS Digital Library (http://www.delos.info)
- The European Library (http://www.theeuropeanlibrary.org)
- The National Science Digital Library (http://nsdl.org)

C225 Digital Library System

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

Relationships:

- *Digital Library System <support> Digital Library*
- *Digital Library System* is *<definedBy> Resource Domain*
- *Digital Library System* is *<definedBy> Content Domain*
- *Digital Library System* is *<definedBy> User Domain*
- *Digital Library System* is *<definedBy> Functionality Domain*
- *Digital Library System* is *<definedBy> Policy Domain*
- *Digital Library System* is *<definedBy> Quality Domain*
- *Digital Library System* is *<definedBy> Architecture Domain*
- *Digital Library System <has> Software Architecture*
- *Digital Library System <has> System Architecture*

Rationale: This Reference Model introduces three notions of systems (cf. Section I.2) active in the universe, i.e., the *Digital Library*, the *Digital Library System* and the *Digital Library Management System*. The *Digital Library System* is the running software system serving the *Digital Library*. Like any running software system, it is characterised by two facets, its *Software Architecture* and its *System Architecture*. The former consists of a set of *Software Architecture Components*, i.e., *Software Components* and *Interfaces* that compose the software implementing the system. The *System Architecture* is the set of *System Architecture Components* that form the running system, namely the servers, *Hosting Nodes* and the processes, *Running Components*, resulting from the deployment of the *Software Components*.

Examples:

- The set of servers, services and software realising the DELOS Digital Library (http://www.delos.info)
- The set of servers, services and software realising The European Library (http://www.theeuropeanlibrary.org)
- The set of servers, services and software realising the National Science Digital Library (http://nsdl.org)

C226 Digital Library Management System

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

Relationships:

- *Digital Library Management System <deploy> Digital Library System*
- *Digital Library Management System <extend> Digital Library System*
- *Digital Library Management System* is *<definedBy> Resource Domain*
- *Digital Library Management System* is *<definedBy> Content Domain*
- *Digital Library Management System* is *<definedBy> User Domain*
- *Digital Library Management System* is *<definedBy> Functionality Domain*
- *Digital Library Management System* is *<definedBy> Policy Domain*
- *Digital Library Management System* is *<definedBy> Quality Domain*
- *Digital Library Management System* is *<definedBy> Architecture Domain*

Rationale: This Reference Model introduces three notions of systems (cf. Section I.2) active in the universe, i.e., the *Digital Library*, the *Digital Library System* and the *Digital Library Management System*. The *Digital Library Management System* (*DLMS*) is the system that provides *DL Designers*, *DL System Administrators* and *DL Application Developers* with *Functions* supporting their tasks (cf. Section I.4). Depending on the set of *Functions* with which the *DLMS* provides *Actors*, different types of such systems can be implemented (cf. Section I.2).

A Digital Library Management System belongs to the class of 'system software'. As is the case in other related domains, such as operating systems, databases and user interfaces, DLMS software generation environments may provide mechanisms to be used as a platform to produce Digital Library Systems. Depending on the philosophy it follows, a DLMS may belong to one of the following three types:

- **Extensible Digital Library System**

  A complete Digital Library System that is fully operational with respect to a defined core suite of functionality. DLs are constructed by instantiating the DLMS and thus obtaining the DLS. Thanks to the open software architecture, new software components providing additional capabilities can be easily integrated. The DelosDLMS (Ioannidis, Milano, & Schek, 2008) is a prototypical example of a system based on this philosophy.

- **Digital Library System Warehouse**

  A collection of software components that encapsulate the core suite of DL functionality and a set of tools that can be used to combine these components in a variety of ways (in Lego®-like fashion) to create Digital Library Systems offering a tailored integration of functionalities. New software components can be easily incorporated into the Warehouse for subsequent combination with those already there. D4Science (Assante, et al., 2008) is a prototypical example of systems that are based on this philosophy.

- **Digital Library System Generator**

  A highly parameterised software system that encapsulates templates covering a broad range of functionalities, including a defined core suite of DL functionality as well as any advanced functionality that has been deemed appropriate to meet the needs of the specific application domain. Through an initialisation session, the appropriate parameters are set and configured; at the end of that session, an application is automatically generated, and this constitutes the Digital Library System ready for installation and deployment. The MARIAN framework equipped with the 5SL specification language represents an example of this process (Gonçalves & Fox, 2002).

Examples:

- OpenDLib[47] (Castelli & Pagano, 2003): the *DLMS* used to create and maintain the DELOS Digital Library.
- D4Science[48] (Assante, et al., 2008): a prototypical *DLMS* capable of deploying *Digital Library Systems* by relying on a set of *Resources* ranging from *Software Components* to *Hosting Nodes* dynamically gathered through Grid technologies.
- The DelosDLMS (Ioannidis, Milano, & Schek, 2008): a *DLMS* built by integrating software and services developed by DELOS partners.

[47] www.opendlib.com

[48] www.d4science.eu

III.4 Relations' Hierarchy

[Generic Relations][49]

.   R1 isa

.   [Resource Relations]

.   .   R2 identifiedBy

.   .   R3 hasFormat

.   .   .   R4 expressionOf

.   .   .   R5 conformTo

.   .   R6 hasQuality

.   .   R7 regulatedBy

.   .   R8 hasMetadata

.   .   .   R9 describedBy

.   .   .   R10 hasProvenance

.   .   .   R11 hasContext

.   .   .   R12 model (isa User Relation)

.   .   .   R13 hasProfile (isa Architecture Relation)

.   .   R14 hasAnnotation

.   .   R15 expressedBy

.   .   R16 hasPart

.   .   .   R17 composedBy (isa Architecture Relation)

.   .   .   R12 isSequenceOf (isa User Relation)

.   .   R18 associatedWith

.   .   .   R19 use (isa Architecture Relation)

.   .   .   R20 conflictWith (isa Architecture Relation)

.   .   .   R21 invoke

.   .   R1 belongTo

.   .   R2 hasIntension

.   .   R3 hasExtension

.   .   R7 profile

.   [Content Relations]

.   .   R4 hasEdition

.   .   R5 hasView

.   .   R6 hasManifestation

.   [User Relations]

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

. . R8 perform

. . R9 apply

. . R10 concern

. . R11 play

. . R12 model (isa hasMetadata)

. . R1 belongTo (isa Resource Relation)

. . R12 isSequenceOf

. [Functionality Relations]

. . R13 interactWith

. . R14 influencedBy

. . R15 actOn

. . R16 create

. . . R17 createAnnotation

. . . R18 createVersion

. . . R19 createView

. . . R20 createManifestation

. . . R21 create/update

. . R22 retrieve

. . . R23 return

. . R24 produce

. . R25 issue

. [Policy Relations]

. . R7 regulatedBy (isa Resource Relation)

. . R26 govern

. . . R27 prescribe

. . R28 addressedTo

. . R29 antonymOf

. . R30 influence

. [Quality Relations]

. . R31 expressAssessment

. . R32 evaluatedBy

. . R33 measuredBy

. . R34 affectedBy

. . R35 accordTo

. [Architecture Relations]

. . R36 implement

. . . R37 realisedBy

. . . R38 support

. . . R39 hostedBy

.   .   R17 composedBy (isa hasPart)

.   .   R19 use (isa associatedWith)

.   .   R20 conflictWith (isa associatedWith)

.   .   R13 hasProfile (isa hasMetadata)

.   .   R5 conformTo (isa hasFormat)

.   .   .   R27 prescribe (isa govern)

.   .   R7 profile (isa belongTo)

III.5 Reference Model Relations' Definitions

R1 isa

Definition: The relation connecting any concept to the concept it is a sub-concept of. "A isa B" means that A is a 'specialisation' of B and B is a 'generalisation' of A.

Rationale: This relation is borrowed from domains like linguistics, knowledge representation and object-oriented programming where it is of common use. In linguistics, it is used to express hyponyms, i.e. words or sentences whose semantic field is included within that of another word. In knowledge representation and computer science in the large, it is used to capture subsumption among classes of concepts as well as inheritance, i.e. that the objects connected to the object 'A' inherit from it attributes and behaviour.

Examples:

- An Information Object is a Resource;
- Apple is a fruit;

R2 identifiedBy

Definition: The relation connecting a *Resource* to its *Resource Identifier*.

Rationale: The issue of univocally identifying the constituents of a system is a fundamental task for their management. This relation captures the *Resource Identifier* attached to each *Resource* for this identification purpose. Various types of *Resource Identifiers* have been proposed; for a discussion of these, please refer to the *Resource Identifier* concept.

Each *Resource* must have at least one *Resource Identifier*. Each *Resource Identifier* can be assigned to one *Resource* only.

Examples:

- The paper 'Setting the Foundations of Digital Libraries: The DELOS Manifesto' is identified by the DOI 10.1045/march2007-castelli.

R3 hasFormat

Definition: The relation connecting a *Resource* to its *Resource Format*, which establishes the attributes or properties of the *Resource*, their types, cardinalities and so on.

Rationale: A *Resource* must have one format only, whereas the same format can (obviously) be used by many *Resources*.

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

Examples:

- The paper 'Setting the Foundations of Digital Libraries: The DELOS Manifesto' has a *Resource Format* 'text/html'.
- The electronic version of this volume has a *Resource Format* that is a pdf file;

- The OAI-ORE[50] Resource Map is the Resource Format of an OAI-ORE Aggregation.

R4 expressionOf

Definition: The relation connecting a *Resource Format* to the *Ontology* that defines the terms of the schema and states the main constraints on them.

Rationale: A schema gives a concrete status to the terms abstractly defined in an ontology, by establishing implementation details such as the data type of the primitive concepts of the ontology. The schema must retain the constraints expressed in the ontology and may consistently add other constraints, reflecting the implementation decisions taken in the schema.

The same ontology can be expressed in many schemas, while a schema is preferably the expression of a single ontology, even though for practical reasons it is possible for a schema to borrow from a number of ontologies. In this latter case, the schema performs a sort of integration of several ontologies.

Examples:

- The Multipurpose Internet Mail Extensions (MIME) is an Ontology of possible types for Information Objects.

R5 conformTo

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

Rationale: The *Framework Specification* is the *Software Architecture Component* that describes the design of the set of *Architectural Components* planned to form the *Software Architecture* of a system. A *Framework Specification* <*prescribe*> the *Interfaces* that *Architectural Components* should implement. Compliance with the *Framework Specification* guarantees that the *Architectural Components* will by design be interoperable with the other *Architectural Components* of the same system.

Examples:

- A *Framework Specification* may <*prescribe*> the publish/subscribe *Interface* that each *Software Component* of a system must implement in order to conform to the publish/subscribe mechanism planned for such a system.

R6 hasQuality

Definition: The relation connecting a *Resource* to its *Quality Parameters*.

Rationale: A *Resource* will have as many *Quality Parameters* as the number of quality features with which it is associated. The same *Quality Parameter* can be associated with many *Resources*.

Examples:

- A *Function* may be assessed with regard to *User Satisfaction* to understand to what extent the needs of the *Actors* using it are fulfilled.

R7 regulatedBy

Definition: The relation connecting *Resources* to the *Policies* regulating them.

[50] http://www.openarchives.org/ore/

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*.

R8 hasMetadata

Definition: The relation connecting *Resources* to *Information Objects* for management purposes.

Rationale: In classic *Digital Library* models, *Metadata* is a concept that is a primary notion modelling a clearly defined category of objects in the domain of discourse. However, it depends from the context whether an object is or is not *Metadata*. For instance, a relational tuple describing an event (such as an artistic performance) can be a primary *Information Object* in some contexts (e.g., in a database storing the programme of the theatre season) and *Metadata* in a different context (e.g., in a repository storing a digital representation of the performance). For this reason, the notion of *Metadata* is more clearly seen as a role that an *Information Object* plays to another *Information Object* (more precisely, to a resource), hence it is defined as a relation.

From this relation, the notion of *Metadata* is then derived as meaning any *Information Object* that is *Metadata* of a *Resource*. In so doing, we are following the same linguistic convention that, in everyday speech, leads to the usage of the word 'father' as a noun.

Each *Resource* can be associated with zero or more *Information Objects* implementing *Resource's Metadata*. An *Information Object* can be the *Metadata* on zero or one *Resource*.

Examples:

- This volume is an *Information Object* associated with another *Information Object* representing its Dublin Core metadata record via the <*hasMetadata*>.

R9 describedBy

Definition: The relation connecting *Resources* to *Information Objects* describing them.

Rationale: This is a specialisation of the <*hasMetadata*>. A *Resource* can be associated with many descriptive *Information Objects*. A descriptive *Information Object* is associated with one *Resource*.

Examples:

- This volume is associated with an *Information Object* implementing a summary of the volume content for advertisement actions.

R10 hasProvenance

Definition: The relation connecting *Resources* to the *Information Objects* capturing their *Provenance*.

Rationale: This is a specialisation of the <*hasMetadata*> devised to link the *Resource* to the *Information Object* recording its *Provenance* (Gil, et al., 2010), i.e., a record that describes entities and processes involved in producing and delivering or otherwise influencing that *Resource*. Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility. Provenance assertions are a form of contextual metadata and can themselves become important records with their own provenance.

Examples:

- The *Information Object* recording the process leading to the creation of a *Resource*.

R11 hasContext

Definition: The relation connecting *Resources* to the *Information Objects* capturing the situation surrounding the *Resource*.

Rationale: *Context* is a very important yet fuzzy and broad concept. It is the set of all contextual information that can be used to characterize the situation of a *Resource*. By relying on this information it is possible to better understand the *Resource* itself and to use it according to the proper circumstances.

Examples:

- The *Information Object* recording the background underlying the *Resource* including the characteristics of the creator, the "place" the Resource was been created, the "period" the *Resource* has been created.

R12 model

Definition: The relation connecting *Actors* to *Actor Profiles* representing them.

Rationale: This is a specialisation of the <*hasMetadata*>. An *Actor* may have many *Actor Profiles*. An *Actor Profile* must be associated with one *Actor*.

Examples:

- John is associated with an *Information Object* containing John's full name, date of birth, address and reading preferences.
- John, Mary and Paul are the *Actors* constituting the *Group* of a Music Digital Library which is entitled to curate 'The Beatles Collection'. The Group Profile is an *Actor Profile* that specifies through enumeration the three members of the group and defines that an *Actor* of the Group has the Role of Librarian being entitled to *Manage* the Resource of 'The Beatles Collection'.

R13 hasProfile

Definition: The relation connecting *Architectural Components* to *Component Profiles* representing them.

Rationale: This is a specialisation of the <*hasMetadata*>. An *Architectural Component* can be associated with *Component Profiles*. A *Component Profile* must be associated with one *Architectural Component*.

Examples:

- A Web service is associated with an *Information Object* implementing its WSDL document, i.e., a description of the web service in terms of the operations performed and the messages, the data types and the communication protocols used.

R14 hasAnnotation

Definition: The relation connecting *Resources* to *Information Objects* to add an interpretative value to a certain *Region*.

Rationale: This relation is analogous to <*hasMetadata*>. *Annotations* are sometimes modelled as concepts; however, they are more clearly seen as roles that *Information Objects* play to *Resources* in specific contexts. Hence, *Annotation* (cf. Section III.3 C17) is defined as the range of the <*hasAnnotation*> relation. Fortunately, this definition settles the long-standing issue as to whether *Annotations* are to be considered as *Objects* or as *Metadata*.

Each *Resource* can be associated with zero or more *Information Objects* implementing *Resource's Annotations*. An *Information Object* can be the *Annotation* on zero or one *Resource*.

Examples:

- Each note John, a reader of this volume, will produce can be linked to the version he holds and the published as a the 'The DELOS Digital Library Reference Model annotated by John'.

R15 expressedBy

Definition: The relation connecting *Resources* to *Information Objects* materialising them.

Rationale: This relation has been introduced to capture the materialisation of otherwise abstract *Resources*. It is intended mainly for the materialisation of *Resources* such as *Policy* and *Quality Parameter* but can be applied to any type of *Resource*.

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

Examples:

- The *Information Object* recording the 'Mean Average Precision' *Measure* of a *Performance Quality Parameter*.

R16 hasPart

Definition: The relation connecting *Resources* to their constituent *Resources*.

Rationale: The relation where a *Resource* 'child' is a subset or part of the 'parent' *Resource*. This 'part of' association may have two different natures: the aggregative and the compositional one. In the aggregative nature, the single parts stand by themselves and may be constituents of any number of *Resources*. In the compositional nature, the whole strongly owns its parts, i.e., if the whole *Resource* is copied or deleted, its parts are copied or deleted along with it.

Examples:

- A book has parts: the preface, the chapters, the bibliography.

R17 composedBy

Definition: The relation connecting *Architectural Components* to constituent *Architectural Components*.

Rationale: This is the specialisation of the <*hasPart*> relation in the case of *Architectural Components*. Also, in this case the relation can implement the aggregative and the compositional nature of the 'part of'.

An *Architectural Component* can be comprised of many *Architectural Components*. The same *Architectural Component* can be a component of many *Architectural Components*.

Examples:

- A Fedora[51] Repository is comprised of a federation of services among wich the Fedora Search Service, the Preservation Integrity Service and the Fedora Repository Service.

[51] http://www.fedora.info

R18 associatedWith

Definition: The relation connecting a *Resource* to the *Resources* that are linked to the former according to a certain *Purpose*.

Rationale: In addition to the explicitly identified pool of relations connecting_*Resources*, this relation makes it possible to specify cross-resource links with respect to a well-known *Purpose*.

No constraints regarding the cardinality of this relation are established, i.e., a *Resource* may be connected to zero or more *Resources* through the <*associatedWith*> with a certain *Purpose*; a *Resource* may be, or may appear as, the second term of an <*associatedWith*> relationship with a certain *Purpose*.

Examples: --

R19 use

Definition: The relation connecting *Architectural Components* to *Architectural Components* they use.

Rationale: *Architectural Components* are the constituents of the architectures of *Digital Library System*. Thus, despite this model permit to represent monolithic systems, i.e., system composed by a single Architectural Component, it is recommended that systems exploit the component-oriented approach because of its benefits. The <*use*> relations capture the usage relationships between the constituents of compound systems.

An *Architectural Component* can rely on the functions of zero or more *Architectural Components* and can be used by zero or more other *Architectural Components*.

Examples: --

R20 conflictWith

Definition: The relation connecting *Architectural Components* to *Architectural Components* they conflict with.

Rationale: In software systems exploiting the component oriented approach each architectural component must fit with the characteristics of the environment in which it have to operate. The <*conflictWith*> relation captures incompatibilities between *Architectural Components* preventing the coexistence of such *Architectural Components* in the same system. In particular, this relation is particularly useful in the case of Digital Library Systems dynamically deployed, i.e., the set of constituent *Architectural Components* is automatically aggregated by the *DLMS* in order to implement a *Digital Library* (and thus deploying the *DLS* realising the *DL*) matching the *DL Designer* criteria.

An *Architectural Component* can conflict with zero or more *Architectural Components*.

Examples:

- The *Software Component* A conflicts with the *Software Component* B; thus A and B cannot coexist in the same system.
- The *Software Component* A conflicts with the *Hosting Node* B; thus A cannot be deployed on B.

R21 invoke

Definition: The relation connecting *Running Components* to *Running Components* they use to accomplish their task.

Rationale: In software systems exploiting the component oriented approach a lot of relations hold between the constituents. Some of these relations are static, i.e., established at design time and valid in each environment the connected components appear, other are dynamic, i.e., they can evolve along the time and exist in the specific environment in which they have been defined. Usually, dynamic relations are a consequence (more precisely an instantiation) of static relations in an operational context. The <*invoke*> relation represents a dynamic dependency between *Running Components*. In particular, this relation is used to capture the run-time dependency between a *Running Component* and the set of other *Running Components* it uses to deliver its expected functions.

Examples:

- The *Running Component* A invokes the *Running Component* B to obtain the set of *Information Object* it must process to serve a specific request.

R1 belongTo

Definition: The relation connecting *Resources* to the *Resource Sets* in which they belong. A specialisation of this is the relation connecting *Information Objects* to the *Collections* that defines which *Collections* an *Information Object* belongs to. Another specialisation of this is the relation connecting an *Actor* to a *Group* that defines which user group an actor belongs to.

Rationale: A *Resource* may be a member of any number of *Resource Sets* and, conversely, a *Resource Set* may include any (finite) number of *Resources*.

Examples:

- An *Information Object* belongs to one or more *Collections*;
- An *Actor* belongs to one or more *Groups*.
- John is the *End-user* of a Music Digital Library that can belong to the Group of Librarians entitled to curate The Beatles Collection, but also can belong to the Group of Librarians entitled to curate Elvis Presley Collection.

R2 hasIntension

Definition: The relation connecting *Resource Set* to the *Query* describing the criterion underlying the *Resource Set*.

Rationale: In logic, the intension of an expression is its sense, as distinguished by the reference (or denotation) of the expression, called the extension of the expression. This distinction was first made by G. Frege, for whom the sense of an expression corresponded to what we intuitively think of as the meaning of the expression. R. Carnap later suggested that the sense of an expression is a function that gives, for each state of affairs, the extension of the expression. S. Kripke, upon defining a semantics for modal logic, finally established the notion of ‘possible world as state of affairs’ (Dowty, Wall, & Peters, 1980). Davidson argued that giving the meaning of a sentence is equivalent to stating its ‘truth conditions’.

The intension of a *Resource Set* can thus be understood as a statement of what must be true of a *Resource* for it to be a member of the *Resource Set*.

Examples:

- The *Collection* of ‘Leonardo Da Vinci’ works <*hasIntension*> the *Query* ‘author=Leonardo Da Vinci’.

R3 hasExtension

Definition: The relation connecting *Resource Set* to the *Resources* belonging to it.

Rationale: In logic, the extension of an expression is its denotation. For a proposition, this is the truth value in the considered interpretation; the extension of a predicate is the set of objects that are denoted by the predicate in the considered interpretation.

Examples:

- The *Collection* of 'Leonardo Da Vinci' works <*hasExtension*> the set of *Information Objects* authored by Leonardo Da Vinci.

R4 hasEdition

Definition: The relation connecting *Information Objects* to the *Information Objects* that realise them along the time dimension.

Rationale: In classic *Digital Library* models, *Editions* represent the different states of an *Information Object* during its lifetime, i.e., they play the role usually assigned to versions.

Versioning usually creates a tree, because an object may be the version of at most one other object. However, in the Digital Library world a more liberal approach may be appropriate, allowing an *Information Object* to be the *Edition* of possibly many different *Information Objects*. The resulting structure will be a directed graph, which must be acyclic to avoid unintuitive situations.

Examples: An *Information Object* representing a study:

- <*hasEdition*> another *Information Object* representing the draft version of such a study;
- <*hasEdition*> another *Information Object* representing the 'submitted version' of such a study;
- <*hasEdition*> another *Information Object* representing the 'version published in the conference proceedings' with colour images.

R5 hasView

Definition: The relation connecting *Information Objects* to the *Information Objects* that are *Views* of them.

Rationale: The concept of *View* captured by this relation fits very well with those used in the database world. In this context, a view is a virtual or logical table expressed as a query providing a new organisational unit to support some application. Similarly, *Information Object Views* are introduced to provide multiple presentations of the information represented/captured by the *Information Object*, which may prove useful in specific application contexts.

The same *Information Object* may have different *Information Objects* linked through the <*hasView*> relation. Conversely, an *Information Object* may or may not be a *View*, that is the second term of a <*hasView*> relationship.

Examples:

- An *Information Object* representing a data stream of an environmental sensor
  - <*hasView*> the *Information Object* consisting of the raw data, i.e., a series of numerical values measured by the sensor
  - <*hasView*> the *Information Object* consisting of a picture representing the graph of the evolution of the values measured by the sensor over time.

- An *Information Object* representing the outcomes of a workshop
    - <*hasView*> the *Information Object* representing the 'full view' and containing a preface prepared by the conference chair and the whole set of papers accepted and organised thematically
    - <*hasView*> the *Information Object* representing the 'handbook view' and containing the conference programme and the slides of each lecturer accompanied by the abstract of the papers organised per session, and
    - <*hasView*> the *Information Object* representing the 'informative view' and reporting the goal of the workshop and the title list of the accepted papers together with the associated abstract.

R6 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.

R7 profile

Definition: The relation connecting *Component Profiles* to the *Architectural Components* they describe. *Component Profiles* should describe (at least) *Functions*, *Policies*, *Quality Parameters* and *Interfaces* inherent in *Architectural Components* with which they are associated via the <*hasProfile*> relation.

Rationale: *Component Profile* is the *Metadata* associated with each *Architectural Component* for its management. The <*profile*> relation captures the aspects that are expected to be captured by this kind of *Metadata*.

Examples:

- The *Functions* yielded by the *Architectural Component* are typical information expected to be included in the *Component Profile.*
- The *Quality Parameters* guaranteed by the *Architectural Component* are typical information expected to be included in the *Component Profile.*

R8 perform

Definition: The relation connecting *Actors* to Actions that apply *Functions* and concern *Resources* in order to accomplish their digital library-related activities.

Rationale: *Functions* have no meaning by themselves if no *Actor* is executing them. This relation is fundamental to a DL/DLMS, as it expresses the interaction of the DL/DLMS with the *Actors* through specific *Functions* in order to achieve their goals.

Examples:

- Alex is the *Content Consumer* of a scientific Digital Library that can perform an Action than applies the *Search* function in order to explore all the papers on a given research topic created by a certain author and published in a specific period of time.

R9 apply

Definition: The relation connecting *Action* to *Function*.

Rationale: This relation is used to indicate that an Actor's activity in the Digital Library is performed through the execution of a specific Function.

Examples:

- The Content Consumer Mark is acting in the Music Digital Library by applying the Search Function to find specific songs.

R10 concern

Definition: The relation connecting *Action* to *Resource* as well as *Actor Profile* to *Resource.*

Rationale: This relation is used to indicate that an Actor's activity in the Digital Library as well as its Profile is related to other Resources.

Examples:

- The Content Consumer Mark is acting in the Digital Library for finding specific Information Objects. His Profile indicates the various Policies that govern his actions, the Quality characteristics, and the specific Information Objects that he is interested in.

R11 play

Definition: The relation connecting an *Actor* to a *Role* that defines the *Role*(s) of the *Actor*.

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

Examples:

- Mary is a postgraduate student that can be an *End-user* of her department's Digital Library, having the Role of *Content Consumer* when searches the library to find specific papers. She can also have the Role of *Content Creator* when uploads to the Digital Library her own papers.

R12 isSequenceOf

Definition: The relation connecting *Action Log* to *Action*.

Rationale: This relation indicates that an Action Log consists of a set of Actions.

Examples:

- Mark's Action Long in the Digital Library consists of a set of Functions that Mark executed and the various Resources involved in his activities.

R13 interactWith

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

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

Examples:

- The *Acquire* function may interact with *Transform* function so as to facilitate the extraction of *information objects* in an appropriate format requested by the user e.g., content exported into a PDF format.

R14 influencedBy

Definition: The relation connecting *Functions* to *Actor Profiles* that expresses the fact that *Functions* are influenced by specific user characteristics.

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

Examples:

- The *Search Function* is influenced by the *Actor Profile* of the *Actor* that perform it by personalising the returned *Result Set*.

R15 actOn

Definition: The relation connecting *Functions* to *Resources* on which they operate.

Rationale: This relation expresses the connection between specific *Functions* and the *Resources* they interact with, either to manage or access them. A *Function* in most cases produces a result to be presented to the *Actor*; it represents an action performed on one of the DL constituents, which are not only primary *Information Objects* but also *Actor* profiles, *Policies*, etc.

Examples:

- The *Publish Function* acts on the *Collections* the *Actor* performing it requests to submit the *Information Object*.

R16 create

Definition: The relation connecting the *Create Functions* to *Resources* they create. A specialisation of this is the relation connecting the *Author Functions* to the *Information Objects* created.

Rationale: This connection expresses the relation of the creation of *Resources* to the *Resource* created by the *Function*. Note that in this case the new *Resource* is not actually inserted in the library until a *Submit Function* has been performed.

Examples:

- Create a new user;
- Create a new policy rule.

R17 createAnnotation

Definition: The relation connecting *Annotate Functions* to the *Information Objects* they create.

Rationale: This relation expresses the relation of the creation process of an *Annotation* with its end-result.

Examples:

- Create appropriate meta-information for the description of an *information object* (e.g., painting).

R18 createVersion

Definition: The relation connecting *Author Collaboratively Functions* to *Information Objects* they create. These *Information Objects* are linked to the originating *Information Objects* via the <*hasEdition*> relation.

Rationale: This relation expresses the fact that a by-product of collaborative authoring is the creation of different versions of the authored *Information Object*.

Examples:

- Upon the execution of an update function on a specific *information object* e.g., annotations of a specific object, a new version of the *information object* is automatically created.

R19 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:

- Transformation of a video file from the AVI to the MPEG4 format.

R20 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:

- Textual *information objects* should be transformed to an HTML format so as to import them within a system supporting such a kind of manifestations only.

R21 create/update

Definition: The relation connecting *User Profiling Function* to the *Actor Profile.*

Rationale: This relation indicates that a *User Profiling Function* is used to create or update an Actor's Profile.

Examples:

- A *User Profiling Function* is used to analyse Mark's activity in the Digital Library and find his preferences in order to add them to his Profile.

R22 retrieve

Definition: The relation connecting *Access Resource Functions* to *Resources* they find. A specialisation of this relation connects *Find Collaborator Functions* to *Actors* they find. Another specialisation is the relation <*return*> which connects the *Function Discover* and *Result Set.*

Rationale: This *Function* connects a retrieval function to the retrieved result.

Examples: --

R23 return

Definition: The relation connecting *Discover Functions* to *Result Sets* they find. It is a specialisation of the <*retrieve*> relation connecting *Access Resource Functions* to *Resources*.

Rationale: This *Function* connects a *Discover Function* to the *Result Set* it returns.

Examples: --

R24 produce

Definition: The relation connecting *Queries* to *Result Sets* they characterise.

Rationale: When a *Query* is issued as the input to a *Search Function*, it produces a *Result Set*.

Examples: --

R25 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: --

R26 govern

Definition: The relation connecting *Policies* to the *Resources* they control/govern. It is the inverse relation of <*regulatedBy*>.

Rationale: Each *Policy* to be effectively implemented must be applied to *Resources*. This relation captures those *Resources* for which each *Policy* is designed to influence the actions and conduct.

Examples:

- Digital Rights Management Policy governs Functions, while Digital Rights govern Information Objects.
- System Policy governs Functions.

R27 prescribe

Definition: The relation connecting *Framework Specifications* to the *Interfaces* they state as a rule that should be carried out by *Architectural Components* that <*conformTo*> it.

Rationale: *Framework Specification* is the *Software Architecture Component* that describes the design of the set of *Software Architecture Components* designed to form the *Software Architecture* of a system. By establishing the *Interfaces* each *Software Architecture Component* (actually a *Software Component*) is expected to implement, it is possible to guarantee by design that the set of *Software Architecture Components* forming a *Software Architecture* will work successfully collaboratively so as to form a whole.

Examples:

- A *Framework Specification* may <*prescribe*> the publish/subscribe *Interface* each *Software Component* of a system must implement in order to conform to the publish/subscribe mechanism planned for such a system.

R28 addressedTo

Definition: The relation connecting *Policy* to the *Actor(s)* the *Policy* is conceived for.

Rationale: Every Policy is conceived to drive the behaviour of the Actor(s). This relation is conceived to capture the "target" Actor each Policy is devised for.

Examples:

- *Collection Development Policy* is addressed to *DL End-user(s)*, namely *Content Creator(s)*.

R29 antonymOf

Definition: The relation connecting *Policy* to *Policy* with opposite meaning.

Rationale: This relation is used when we have a set of two *Policies* (generally *Resources*) with opposite meaning. It is introduced in order to facilitate the understanding of bipolar sets of concepts.

Examples:

- *Extrinsic Policy* and *Intrinsic Policy* form a pair where each concept is the <*antonymOf*> the other concept.

R30 influence

Definition: The relation connecting *Quality Parameters* to *Policy* they affect.

Rationale: This Reference Model does not present the digital library as a static entity but also highlights the processes within the functioning of a digital library. An important aspect is how decisions for applying specific *Policies* could be taken within the DL. This relation captures those cases in which the decision is based on *Quality Parameters*.

Examples:

- The value of the *Security Enforcement Quality Parameter* supported by a *Digital Library System* will influence the *Digital Right Management Policy*.
- *Content Quality Parameter* influences *Preservation Policy*
- *Integrity* influences *Preservation Policy*

R31 expressAssessment

Definition: The relation connecting *Quality Parameters* to the *Actors* who are expressing an assessment of a *Resource*.

Rationale: The *expressAssessment* relation models the fact that a *Quality Parameter* serves the purpose of recording the judgement of an *Actor*, who is the active subject expressing an opinion about some feature of a *Resource*, which is the object under examination.

Examples:

- See *Quality Parameter*, *Actor* and *Resource*.

R32 evaluatedBy

Definition: The relation connecting *Quality Parameters* to the *Measurements* according to which they are evaluated.

Rationale: The <*evaluatedBy*> relation defines the process followed to determine the assessment of a *Resource* with respect to the specific feature taken into consideration by a *Quality Parameter*. This relation takes into account that different *Measurements* can be used for assessing the same *Quality Parameter*.

Examples:

- For example, an *Objective Measurement* of the *Performance* of a given *Function* is its response time; an *Objective Measurement* of the *Usability* of a given *Function* is the time needed to complete a task, while a *Qualitative Measurement* and *Subjective Measurement* of it is a score expressed by a user on a like-dislike scale.

R33 measuredBy

Definition: The relation connecting *Quality Parameters* to the *Measures* that assign them a value.

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

Examples:

- See *Quality Parameter* and *Measure*.

R34 affectedBy

Definition: The relation connecting *Quality Parameters* to other *Resources* that influence their determination.

Rationale: *Quality Parameters* and *Resources* are interrelated concepts not only in the sense that each *Resource* can be associated with one or more *Quality Parameters* that characterise their features, as expressed by the <*hasQuality*> relation, but also that *Resources* may affect and impact the assessment expressed by a *Quality Parameter*.

Examples:

- *Security Enforcement* is affected by other *Quality Parameters*, such as *User Behaviour*, by *Policies*, such as *Digital Rights Managements Policy*, and by *Functions*, such as *Access Information*.
- *Integrity* (C177) is affected by *Policy consistency* (C202)
- *Integrity* (C177) is affected by *Preservation performance* (C178)
- *Provenance* (C12) is affected by *Policy precision* (C202)
- *Metadata evaluation* (C184) is affected by *Sustainability* (C168)
- *Metadata evaluation* (C184) is affected by *Maintenance Performance* (C208)
- *Interoperability support* (C165) is affected by *Compliance with Standards* (C172)
- Reputation (C166) is affected by Trustworthiness (C175)

R35 accordTo

Definition: The relation connecting *Measure* to the *Measurements* that define how they should be obtained.

Rationale: The <*accordTo*> relation associates an actual value computed for assessing a *Quality Parameter*, i.e., a *Measure*, with the procedure adopted for estimating it in order to make clear and traceable how each value has been produced.

Examples:

- When evaluating information access components, you may need to specify that the value 0.3341 (the *Measure*) is the Mean Average Precision computed by the trec_eval[52] tool, which truncates the computation after 1,000 retrieved documents (the *Measurement*).

R36 implement

Definition: The relation connecting *Architectural Components* to the *Resources* they realise.

Rationale: The <*implement*> relation associates notion of *Resource* with the *Architectural Component* (another *Resource*) that makes it real, put it into effect. This notion of 'implementation' covers the whole spectrum of possible interpretations ranging from the implementation of a *Policy* or a *Function* (meaning that the *Architectural Component* contains the logic to put into effect the *Policy* or the *Function*) to the implementation of an *Information Object* (meaning that through the *Architectural Component* it is possible to have access to the *Information Object*).

Two notable instances of this relation are in the Architecture Domain: a *Software Component* (C213) implements one or more *Interface* (C215) and an *Application Framework* (C214) implements a *Framework Specification* (C216).

The same *Resource* can be implemented by many *Architectural Components* as well as an *Architectural Component* can implement many *Resources*.

Examples: --

R37 realisedBy

Definition: The relation connecting *Software Components* to the *Running Components* realising them.

Rationale: This relation is a specialisation of the <*implement*> relation in the context of *Software Components* and *Running Components*. In particular, it is used to capture the fact that a *Running Component* implements one or more *Software Components* thus it is the process putting into effect what is coded in the software artefact.

A *Software Components* can be put in action by zero or more *Running Components*; a *Running Component* can put in action one or more *Software Components.*

Examples: --

R38 support

Definition: The relation connecting *Application Frameworks* to the *Running Components* that support the operation.

Rationale: This relation is a specialisation of the <*implement*> relation in the context of *Application Frameworks* and *Running Components*. In particular, it is used to capture the fact that a *Running Component* implements zero or more *Application Frameworks,* thus it is the process putting into effect

[52] http://trec.nist.gov/

the software connecting and supporting the operation of other *Software Components* forming the system.

An *Application Framework* can be put in action by zero or more *Running Components*; a *Running Component* can put in action zero or more *Application Frameworks.*

Examples:

- The gCube application framework support the operation of all the gCube services.

R39 hostedBy

Definition: The relation connecting *Running Components* to the *Hosting Nodes* physically hosting them.

Rationale: This relation is a specialisation of the <*implement*> relation in the context of *Running Components* and *Hosting Nodes*. In particular, it is used to capture the fact that a *Hosting Node* hosts zero or more *Running Components* and thus by providing them with the environment supporting their operation it puts them into action.

A *Running Component* is hosted on one *Hosting Node* a time; a *Hosting Node* can put in action zero or more *Running Components.*

Examples:

- The D4Science portal is hosted by a set of servers (for replication issues).

PART IV Digital Library Reference Model Conformance Checklist

IV.1 Introduction

In a wide range of domains from aviation to construction and from healthcare to project management checklists are increasingly common as mechanism to control process quality (e.g. by reducing errors), to ensure compliance with performance guidelines, to provide transparent mechanisms for understanding and using complex systems, and to facilitate consistency of action between practitioners. They enable audit consistency, and in providing a method for understanding complex systems. The DL.org project has elaborated this Digital Library Reference Model Conformance Checklist. In this instance, this checklist provides a set of statements that will enable assessors to determine whether or not their library is compliant with the *Digital Library Reference Model* (DLRM), to enable those designing a new digital library to determine whether or not their planned library application is compliant with the DLRM, and to make it feasible for those who would like to use a digital library to hold their content, as a resource, or for any other purpose to establish its compliance. The structured nature of the checklist reduces ambiguity, a common aspect of assessments of this kind. Within the realm of digital libraries The *Digital Library Reference Model* delivers a common vocabulary and model to communication about digital libraries and their characteristics. The DL.org Checklist supports assessment of compliance of digital libraries and systems with the model and comparisions between different digital libraries.

IV.2 Scope and Beneficiaries of the Checklist

This checklist has been designed to be used by assessors, from a system designer to a digital librarian or from a funder to a digital library content contributor who seeks to determine whether or not their digital library, or a specific digital library service or system, conforms to the *Digital Library Reference Model (DLRM)*. It will help DL designers involved in building new digital library services or systems to assess whether or not their design will deliver a digital library management system that conforms to the *DLRM*. The checklist will allow an auditor (or researcher) to internally or externally assess information systems – which claim to be digital libraries – for conformance with the *DLRM*. Digital Library depositors and users will be able to make their own assessments with the checklist. It is expected that these roles overlap. While we intend that the users of the checklist should be varied we recognise that only staff (or auditors) with broad access to the digital library at several core levels will be able to complete all the checklist sections, and that a complete assessment will require the participation of more than one DL actor.

There will be many ways to use results of applying the checklist. For instance, a registry of assessed digital libraries might be created and maintained to make available the conformance checklist results; such a registry would help policy makers and DL managers to identify the key steps towards the implementation or development of a digital library, or even specific components or services to strengthen and innovate. Alternatively, DL Designers might use the Checklist in an inspirational way to test whether or not the DL that they are proposing developing conforms to the model.

The checklist – in conjunction with the *Digital Library Reference Model* – can also be used as an educational tool; the process of employing the *Digital Library Reference Model* requires the user to ask questions and to develop an appreciation of the Reference Model's attributes and subtleties. With the checklist in place, teachers will be able to use it in conjunction with the DLRM to enable students to study different digital libraries and to develop an understanding of their attributes and their processes.

IV.3 Developing the Reference Model Conformance Checklist

IV.3.1 Identifying the Criteria

The checklist criteria were derived from the *Digital Library Reference Model* concepts and relationships structuring those requirements into groups of properties which were seen to be either mandatory, recommended, or nice to have. In listing them as criteria, we considered domain-related concepts and relationships within each domain and cross-domains. The checklist does not have a one to one correspondence with the *Digital Library Reference Model*, but it does link each criterion that it has included to the model itself.

The criteria development process involved the identification of:

- "essential" features, i.e. characteristics that a 'digital library'[53] must have (<MUST>). The "must" criteria is mandatory for any 'digital library';
- features that characterize "good" 'digital libraries' (<SHOULD>). The "should" criteria is a good practice according to the *Digital Library Reference Model;*
- "optional" features (<MAY>). The "may" criteria is related to property that can distinguish a 'digital library' from another one. These characteristics make a 'digital library' more appropriate to one purpose or another or add unique functionality to it.

IV.3.2 Checklist Criteria Characterisation

The *Digital Library Reference Model's* concepts and relationships have been analysed from the different perspectives and prioritised with respect to their role in defining the 'digital library'. From these, a checklist that covered all such features has been developed. For each criterion statements describing the entity as well as a range of other characteristics are given. These assist the user in contextualising the criterion by helping them to understand why a particular criterion is important, and by linking it to the *Digital Library Reference Model*. In particular, the checklist statement defining a criterion in the checklist consists of the following elements (see Table 1):

- ***Criterion*** – it consists of a statement which describes a particular property or aspect and states whether or not its presence is mandatory (must), recommended (should), or considered a worthy optional addition (may). In creating this structure, the IETF guidelines "RFC 2119"[54] have been used:
    - *MUST* – This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification;
    - *MUST NOT* – This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification;
    - *SHOULD* – This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course;

[53] In the remainder of this document the term digital library is used in the most abstract term to refer to the system that is time to time target of the checklist, i.e. it might correspond to one or more of the types of systems introduced in the Reference Model: Digital Library, Digital Library System and Digital Library Management System.

[54] http://www.ietf.org/rfc/rfc2119.txt

  - *SHOULD NOT* – This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label;
  - *MAY* – This word, or the adjective "OPTIONAL", mean that an item is truly optional.
- ***Why is it needed*** – it encapsulates in a couple of lines why this particular property is mandatory, recommended, or an optional element of a digital library or digital library management system.
- ***DLRM Concept Identifier(s)*** – it supports the linking of the particular criterion to the Reference Model through the addition of the number of the relevant DLRM entity (e.g. Part III.3 numeric identifier(s)).
- ***DLRM Relation Identifier(s)*** – it supports the linking of the particular criterion to the DLRM through the addition of the number of the relevant DLRM relation (e.g. Part III.5 numeric identifier(s)).
- ***Examples*** – Taken alone criterion can be difficult to understand or contextualise. So for each criterion the example provides guidance to assist assessors in conceptualising the specific criterion (e.g. Archaeology Data Service Charging Policy at http://ads.ahds.ac.uk/project/userinfo/charging.html).
- ***Type of Evidence*** – it provides indications as to where the checklist user might find an indication of the presence of the property within the digital library. Properties tend to be apparent either through analysis of the ***documentary materials*** (e.g. manuals, reports), ***observational evidence*** (the auditor observes users interacting with the 'digital library'), or ***testimony*** (after talking to the system designer, the assessor verified the system uses unique resource identifiers). When completing the checklist as well as noting the presence or absence of a particular property the assessor should also note the kind of evidence used to verify that the system complies with the particular criterion.

Table 1. Checklist Criterion With explanatory definition material

Criterion:	[i.e., statement of the criterion, a single sentence]
Why it is needed:	[i.e., explanatory text of a couple of sentences describing the criterion and why it is significant]
DLRM Concept Identifier(s):	[i.e., corresponding *Digital Library Reference Model* numeric identifier(s)]
DLRM Relation Identifier(s):	[i.e., corresponding *Digital Library Reference Model* numeric identifier(s)]
Example:	[i.e., for each criterion at least one example is given. The example illustrates how the criterion can be met]
Suggested Type of Evidence:	[i.e., suggestion as to whether the evidence is likely to be found through analysis of documentary materials, observational evidence, or testimony]

IV.4 Applying the Reference Model Conformance Checklist

The Reference Model conformance checklist is designed to be used by assessors. It is best used with complete access to the underlying digital library managers, documentation and back-of-the-house systems because determining conformance with the criteria requires detailed understanding of the system itself.

In using the checklist the assessor should read each criterion and where necessary the assessor should consider the statement as to why the criterion is needed, the examples provided, and, if necessary, use the DLRM Concept Identifier to find the correct place in the Reference Model where the concept that this criterion references is described. Then the assessor should examine the 'digital library', its associated documentation, conduct observations or interviews to determine whether or not the criterion is met by the system. Once the assessment has been made the assessor should enter the type of evidence (e.g. *observational*, *documentary*, or *testimonial*) used to make the determination as to whether the criterion is satisfied and then under the heading 'Source of the evidence' the assessor should list what specific piece of evidence was used to determine whether or not the criterion was satisfied or not. For example, in an assessment as to whether the Archaeology Data Service (ADS) satisfied the Criterion "The Policy managed by the Digital Library may be enforced " the assessor might provide the following supporting record:

- ***Type of Evidence***: Documentary
- ***Source of Evidence***: Archaeology Data Service (ADS) Charging Policy available at: (e.g. in a particular publication or at a specific URL)

Finally, the assessor makes a determination as to whether or not the criterion is satisfied by the system. The result of the analysis is a table showing whether the criterion has been met and the evidence used to make that determination (see Table 2).

Table 2. Reference Model Conformance Checklist Assessment Form - To be completed by the Assessors

DL auditor / DL assessor:	[i.e., statement documenting the person that is filling the form]
Criterion fulfilment:	[i.e., statement documenting whether or not the criterion is satisfied by the 'digital library'.]
Type of evidence:	[i.e., statement documenting the type of evidence that has been used to assert whether or not the criterion is satisfied by the 'digital library'. Possible values are: • Documentary – the auditors used the 'digital library' documentation, e.g. reports, plan, handbooks; • Observational – the auditor observes users interacting with the 'digital library'; Testimonial – after talking to the system designer, the auditor verified the system behaviour;]
Source of the evidence:	[i.e. statement documenting the piece of evidence that has been used to determine the 'Criterion fulfilment' statement above.]

Table 3 below shows a complete example for the assessment of a single criterion.

Table 3. Example of Reference Model Conformance Checklist Assessment Form

Criterion:	Every **Information Object** must have at least one element of **Metadata** (**hasMetadata**) associated with it.
Why it is needed:	This ensures that each Information Object is equipped with data supporting its management and use..
DLRM Concept Identifier(s):	C7 Information Object C11 Metadata
DLRM Relation Identifier(s):	R8 hasMetadata
Example:	Keywords are metadata which describe the content of a resource.
Suggested Type of Evidence:	Documentary Observational
DL auditor / DL assessor:	John Smith (john.smith@nowhere.org)
Criterion fulfilment:	The criterion is fulfilled. In particular, in this digital library there is a DublinCore metadata record associated to every Information Object.
Type of evidence:	Documentary: the fact that the digital library supports DublinCore metadata records is reported in the digital library handbook.
Source of the evidence:	The digital library handbook (Doe, 2010), Section 3, pages 121-123.

Once all the criteria have been considered the assessor reviews them and determines whether or not the mandatory criteria are all met and uses that evidence combined with the breath of recommended and optional criteria to make statements about the conformity of a particular digital library to the *Digital Library Reference Model* and about its fitness for purpose.

IV.5 Identifying the Reference Model Conformance Criteria

The following set of criteria results from an analysis of the Digital Library Reference Model concepts and relationships. These criteria have been selected because of their discriminating power with respect to defining whether a 'digital library' conforms to the characterisation of such systems as envisaged by the *Digital Library Reference Model*. The presentation of the criteria is structured according to the six domains characterising the digital library service (Content, User, Functionality, Policy, Quality and Architecture) for the sake of usability and interoperability between the Checklist and the model.

IV.5.1 Content-oriented Criteria

The following criteria have been selected to verify whether or not the 'digital library' conforms to the Digital Library Reference Model from the Content domain point of view.

MANDATORY

Regardless of the type of Content a 'digital library' was conceived to hold, it must meet at least the following criteria:

- The Digital Library must manage a set of **Information Object**s and the set can not be empty.

  By definition the purpose of a digital library is to collect, manage and preserve in perpetuity digital content.
- Every **Information Object** must have (i**dentifiedBy**) a unique identifier (**Resource Identifier**).

  This guarantees that each Information Object managed by the 'digital library' is distinguishable from the remaining ones in the context of the same 'digital library'.
- Every **Information Object** must have at least one element of **Metadata** (**hasMetadata**) associated with it.

  This ensures that each Information Object is equipped with data supporting its management and use.
- Every **Information Object** must belong (**belongTo**) to at least one **Collection.**

  This guarantees that the overall set of Information Objects managed by the 'digital library' pertains to an organized body.
- Every **Collection** must have (**identifiedBy**) a unique identifier (**Resource Identifier**).

  This establishes that each Collection managed by the Digital Library is distinguishable from any others in the context of the same Digital Library.
- Every **Collection** must have at least one element of **Metadata** (**hasMetadata**) associated with it.

  This asserts that each Collection is equipped with data supporting its management and use.

RECOMMENDED

Additional desired properties of a 'digital library' are:

- Every **Information Object** should conform to (**hasFormat**) an explicit and known format (**Resource Format**).

  This guarantees that the system is aware of the "structure" each Information Object conforms to and that this structure is publicly declared thus making the Information Object usable by third party actors whether human or machine. The notion of Resource Format is wide and might range from an abstract one (e.g. "enhanced publication") to a concrete one (e.g. PDF).

- Every **Metadata** should conform to (**hasFormat**) an explicit and known format (**Resource Format**).

  This criterion – a specialization of the previous one – ensures that the system is aware of the "structure" the metadata object conforms to and that this structure is publicly declared so that it can be used by third party actors whether human or machine. In this case the notion of Resource Format corresponds to the notion of metadata schema.
- Every **Annotation** should conform to (**hasFormat)** an explicit and known format (**Resource Format**).

  This criterion guarantees that the system is aware of the particular "structure" to which the annotation object conforms. Being publicly declared, this structure can be used by third party actors whether human or machine.
- Every **Collection** should have a well-defined intension, i.e., the set of criteria characterising **Collection** membership (**hasIntension**), and should have a well-defined extension, i.e., the set of **Information Objects** belonging to the collection (**hasExtension**).

  The collection concept is fundamental to keep the set of Information Objects organised. Because of this, it is recommended that both the set of Information Objects belonging to a collection and the criteria driving the membership of an information object into a collection are clearly defined.
- Every **Information Object** should be regulated (**regulatedBy**) by **Policies**.

  Policies are essential to establish conditions, rules, terms or regulations governing the management of information objects.

OPTIONAL

Finally, a 'digital library' may also meet the following set of criteria:

- An **Information Object** may have multiple **Editions** (**hasEdition**) each represented by a different related **Information Object**.

  A 'digital library' might be employed to manage multiple editions of the same work. In these cases it is important to deal effectively with the edition concept.
- An **Information Object** may have multiple **Views** (**hasView**) each represented by a different related **Information Object**.

  A 'digital library' might be called to manage multiple "views"/"expressions" of the same conceptual work. In these cases it is important to properly deal with the view concept.
- An **Information Object** may have multiple **Manifestations** (**hasManifestation**) each represented by a different related **Information Object**.

  A 'digital library' might be called to manage multiple "items" of the same conceptual work or view. In these cases it is important to properly deal with the manifestation concept.
- An **Information Object** may be compound (**hasPart**), i.e., it may consist of multiple **Information Objects**.

  Modern 'digital libraries' are usually expected to deal with emerging forms of "documents". Very often such a "documents" consists of aggregates of other objects (of different media).
- An **Information Object** may be associated (**associatedWith**) with other **Information Objects** for a certain **Purpose**.

  Managing compound objects may require links other objects. The motivations leading to linking are diverse and context specific, e.g., citation and lineage.
- An **Information Object** may have multiple elements of **Metadata** (**hasMetadata**) associated with it.

Metadata are a type of Information Object intended to support the management and use of the Information Objects to which they are attached. Different metadata can be conceived to support diverse needs. The majority of 'digital libraries' tend to deal with a single metadata format.

- An **Information Object** may be associated with multiple **Annotations** (**hasAnnotation**).

  Annotations are kinds of Information Objects that are attached to existing Information Objects for various purposes including objects enrichment and cooperative working.
- A **Collection** may be associated with multiple **Metadata** (**hasMetadata**).

  According to the Reference Model, Collections are a type of Information Object. Because of this, they inherit all the features of Information Objects and benefit of multiple metadata.
- A **Collections** may be associated with multiple **Annotations** (**hasAnnotation**).

  According to the Reference Model, Collections are a type of Information Object. Because of this, they inherit all the features of Information Objects and benefits of multiple annotations.

IV.5.2 User-oriented Criteria

The following criteria have been selected to verify whether or not the 'digital library' conforms to the Digital Library Reference Model from the User domain point of view.

MANDATORY

Regardless of the type of Users a 'digital library' is conceived for, it must meet at least the following criteria:

- The Digital Library must serve a clearly identified set of **Actors** *and this can not be an empty set*.

  Actors represent the entities that interact with any digital library 'system', i.e., humans and inanimate entities such as software programs or physical instruments. This guarantees that they exist, i.e., there is no 'digital library' without users interacting with it.
- Every **Actor** must have (**identifiedBy**) a unique **Resource Identifier**.

  This guarantees that every Actor is distinguishable from other Actors in the context of the same 'digital library'.
- Every **Actor** must be described (**model**) by at least one **Actor Profile**.

  This guarantees that every Actor uses the 'digital library' and interacts with it as well as with other Actors in a personalised and codified way.
- Every **Actor** must act (**play**) with at least one **Role**.

  This guarantees that an Actor cannot interact with a Digital Library if its role is not specified.
- The set of managed **Roles** must include the **DL Manager** Role.

  DL Managers are Actors that exploit DLMS facilities to define, customise, and maintain the DL service.
- The set of managed **Roles** must include the **DL Software Developer** Role.

  DL Software Developers exploit Digital Library Management System facilities to create and customise the constituents of the Digital Library Systems.
- The set of managed **Roles** must include the **End-user** Role.

  This guarantees that the digital library supports at least typical end-user roles, like content consumers, content creators or digital librarians.

RECOMMENDED

Additionally, a Digital Library should meet the following criteria:

- Every **Actor** should perform (**perform**) **Actions** that apply (**apply**) **Functions** and concern (**concern**) **Resources**.

  Every Actor that interacts with a digital library should be able to perform certain Actions that involve the application of Functions and specific Resources.

OPTIONAL

Finally, a Digital Library may meet the following criteria:

- **Actors** may belong to (**belongTo**) more than one **Group**.

  During the interaction of an Actor with a Digital Library, the Actor may communicate or collaborate with other Actors that belong to various Groups; thus, the specific Actor may participate in different Groups. The concept of Group in the User domain has commonalities with the concept of Collection in the Content domain, it is a mechanism to organise Actors.

IV.5.3 Functionality-oriented Criteria

The following criteria have been selected to verify whether or not the 'digital library' conforms to the Digital Library Reference Model from the Functionality domain point of view.

MANDATORY

Regardless of the type of Functionality a 'digital library' is conceived for, it meets at least the following criteria:

- The Digital Library must offer a clearly identified set of **Functions** and this can not be an empty set.

  The purpose of the DL is to offer functions, i.e., a particular processing task that can be realised on a Resource or a Resource Set as the result of an Action of a particular Actor.
- Every **Function** must have (**identifiedBy**) a unique identifier (**Resource Identifier**).

  A Function is a Resource, thus it must be identified by a persistent identifier if it is to be distinguished from other Functions managed by the DL.
- Every **Function** must be performed **(perform)** by **Actors.**

  DL Functions are the implementations of functions and services enabling Actors to interact with the DL.
- Every **Actor** must be provided with (**perform**) **Functions** to **Access Resources**.

  The DL must implement functions to enable actors to access, e.g., discover, acquire and visualize, all types of Resources (Information Objects, Actors Profiles).
- Every **Actor** must be provided with (**perform**) **Functions** to **Discover Resources**.

  Actors must be able to find the desired Information Objects, search and access not only the DL Content, but also other Actors or Functions.
- Every **DL System Administrator** must be provided with (**perform**) **Functions** to **Manage & Configure DLS**.

  DL must implement functions for handling the DLS and configuring its settings.

RECOMMENDED

Additionally, a Digital Library should meet the following criteria:

- Every **Function** should be able to interact with (**intertactWith**) other **Functions**.

  DL functions should exchange information with other functions regulating their behaviour and performance.
- **Functions** to **Acquire** (**actOn**) **Resources** should be provided.

  DL functionality should enable Actors to retain Resources e.g., Information Objects and Actor Profiles, in existence past their interaction with the Digital Library System.
- **Functions** to **Browse** (**actOn**) the **Resources** should be provided.

  DL should implement services enabling Actors (virtual or real) to browse the available DL content, user profiles, policies, etc.
- **Functions** to **Search** (**actOn**) the **Resources** should be provided.

  Actors should be able to look for specific objects held within the DL by expressing queries and by entering specific keywords and constraints.
- **Functions** to **Visualize** (**actOn**) the Actor's requested **Resources** should be provided.

  A DL should deliver to Actor the requested information using the appropriate visualizations to produce comprehensive and well-presented objects, lists and query result sets.
- **Functions** to **Manage Information Object**(s) (**actOn**) should be provided.

  A DL should implement functions to handle, i.e., disseminate, publish, process, analyze and transform, the Content of the DL, i.e., Information Objects.
- **Functions** to **Manage Actor**(s) (**actOn**) should be provided.

  A DL should implement Functions to establish registered actors, personalize their preference and apply user profiles.
- **Functions** to **Manage DL** specific domains in a large scale should be provided.

  The DL should implement services and mechanisms to handle DL domains as a whole, e.g., Manage (import, export) all the Content of DL rather than handling each Information Object individually.

OPTIONAL

Finally, a Digital Library may meet the following criteria:

- **Functions** may depend on (**influencedBy**) the **Actor's Profile** who invokes them.

  DL Functions that are offered to the Actor(s) may be customized according to his/her profile, DLS role and rights and /or personal preferences.
- **Functions** may consist of other parts (**hasPart**), i.e., sub-functions.

  Functions may be organized in arbitrarily complex workflows, based on composition and linking facilities.
- **Functions** may be enriched with **Metadata** (**hasMetadata**) and **Annotation** (**hasAnnotation**).

  DL Functions may have a description, which tells what the function does and how a system or human can interact with it.
- **Functions** may enable (**actOn**) **Actors** (virtual or real) to **Collaborate** with each other.

  Actors (virtual or real) may act as peers who are able to communicate, share and exchange information collaboratively.

IV.5.4 Policy-oriented Criteria

The following criteria have been selected to verify whether or not the 'digital library' conforms to the Digital Library Reference Model from the Policy domain point of view.

MANDATORY

Regardless of the type of Policy a 'digital library' is conceived for, it must meet at least the following criteria:

- The Digital Library must be regulated by (**regulatedBy**) a clearly defined set of **Policies** and this can not be an empty set.

  Policies are set of conditions, rules, terms or regulations governing the operation of the DL;
- **Access Policies** must regulate (**regulatedBy**) the use of the Digital Library by **Actors**.

  Access Policies are essential to establish conditions, rules, terms or regulations governing the interactions between the Digital Library and Actors.
- Every **Policy** must be addressed (**govern**) at least to an **Actor** (**regulatedBy**).

  Defining the recipients of a Policy ensures the interaction between the Digital Library and its Actor(s).
- Every **Policy** must have clearly defined scope(s) and characteristics (**Policy Quality Parameter**).

  A Policy must have defined objectives and consequences affecting the DL system as a whole, a certain domain, a specific task or entity.

RECOMMENDED

Additional desired properties of 'digital library' policies are:

- Every **Policy** should be expressed by (**expressedBy**) an **Information Object**.

  The digital representation of a Policy ensures its controlled description, management and use within the Digital Library. This representation enables automatic enforcing. Moreover, it is a prerequisite for a series of other automatic actions including policy comparison, policy reconciliation and policy interoperability.
- Every **Policy** should have (**identifiedBy**) a unique identifier (**Resource Identifier**).

  The use of a persistent identifier ensures that each DL Policy is distinguishable from the others in the context of the same Digital Library.
- Every **Policy** should have (**hasFormat**) a known format (**Resource Format**).

  The implementation of a Policy in a known format guarantees that the system is aware of which "structure" each Policy conforms to and that this structure is publicly declared as to be used by third party actors whether human or machine.

OPTIONAL

Finally, a Policy governing a 'digital library' may also meet the following criteria:

- A **Policy** may regulate (**regulatedBy**) the service of the system as a whole (**System Policy**).

  Generic processes within the 'digital library' may be regulated by policies.
- A **Policy** may regulate (**regulatedBy**) functionalities related to Content (**Content Policy**).

  A Policy may regulate processes related to the Content domain.
- A **Policy** may regulate (**regulatedBy**) DL Functions (**Functionality Policy**).

DL Functions' lifetime and behaviour may be governed by specific policies.

- A **Policy** may regulate (**regulatedBy**) User profiles and behaviour (**User Policy**).

  A Policy may regulate processes related to the User domain.

- A **Policy** may be extrinsic (**Extrinsic Policy**).

  A Policy may be imposed from outside the organisation domain of the 'digital library', e.g., by wider organisations regulating the Digital Library itself, by national and international laws, or by customs.

- A **Policy** may be intrinsic (**Intrinsic Policy**).

  The Policy governing the Digital Library may be defined and determined by the Digital Library organisation itself. Intrinsic Policy manifests the Policy principles implemented in the DL. A Policy that is defined by the DL or its organisational context that reflects the organisation's mission and objectives, the intended expectations as to how Actors will interact with the DL, and the expectations of Content Creators as to how their content will be used.

- A **Policy** may be implicit (**Implicit Policy**).

  The Policy governing the Digital Library may be inherent by accident or design. Implicit Policies usually arise as a result of ad-hoc decisions taken at system development level or as a consequence of the inadequate testing of a DLS that results in an interaction of Policies leading to unintended policy deployment.

- A **Policy** may be explicit (**Explicit Policy**).

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

- A **Policy** may be prescriptive (**Prescriptive Policy**).

  The Policy governing the Digital Library may constrain the interactions between DL Actors (virtual or real) and the DL. Prescriptive Policies can cover a broad range of Policies from the kind of Function to which specific types of Actors can have access, to those that govern Collection development.

- A **Policy** may be descriptive (**Descriptive Policy**).

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

- A **Policy** may be enforced (**Enforced Policy**).

  The Policy governing the Digital Library may be deployed and strictly applied within the DL. An Enforced Policy is a Policy applied consistently and strictly in the DL. Monitoring and reporting tools are necessary to follow up how the Policy is being applied.

- A **Policy** may be voluntary (**Voluntary Policy**).

  The Policy governing the Digital Library may be monitored by an actor (human or machine). Voluntary Policy basically means a Policy that is followed according to the decision of the Actor. This is valid for all Policies for which its application is a matter of choice. In some cases, users may comply with Policies that are not officially communicated.

- A **Policy** may be compound (**hasPart**).

  A Policy may be organised in arbitrarily complex and structured forms. A compound policy may be obtained by properly combining constituent Policies.

IV.5.5 Quality-oriented Criteria

The following criteria have been selected to verify whether or not the ‘digital library’ conforms to the Digital Library Reference Model from the Quality domain point of view.

MANDATORY

Regardless of the type of Quality a ‘digital library’ is conceived for, it must meet at least the following criteria:

- A Digital Library (actually its **Resource(s)**) must be characterised by a set of **Quality parameter(s)** (**hasQuality**) and this can not be an empty set.

  Any DL can be considered from a quality point of view by a DL Actor. The expression of the Actor’s assessment is the Quality Parameter.

- Every **Quality Parameter** must represent the assessment of a Digital Library **Actor**, whether human or machine, on a **Resource** (**expressAssessment**).

  A Quality Parameter is always the expression of an assessment made by an Actor on a Resource.

RECOMMENDED

Additional desired properties of a ‘digital library’ are:

- Every **Quality Parameter** should be identified by (**identifiedBy**) a unique identifier (**Resource Identifier**).

  The use of a persistent identifier ensures that each Quality Parameter is distinguishable from the remaining ones in the context of the same ‘digital library’.

- Every **Quality Parameter** should be expressed by (**expressedBy**) an **Information Object**.

  The digital representation of a Quality Parameter ensures its controlled description, management and use within the Digital Library. This representation is a prerequisite for a series of other automatic actions including the assessment of Digital Library content and services, and quality interoperability.

- Every **Quality Parameter** should be evaluated by (**evaluatedBy)** specific **Measurements**.

  In accordance with a selected Measurement, a Quality Parameter should have a specific value (e.g. the Measure).

- Every **Quality Parameter** should be measured (**measuredBy**) by a **Measure**.

  Any Quality Parameter should be managed by the Digital Library according to different Measurements, which provide procedures for assessing different aspects of each Quality Parameter and assigning it a value.

- Every **Quality Parameter** should be specified (**regulatedBy**) by **Policies**.

  The Digital Library should have policies governing the evaluation and the assessment of its systems and facets.

OPTIONAL

Finally, a ‘digital library’ may also meet the following set of criteria:

- A **Quality Parameter** may be compound (**hasPart**).

  A Quality Parameter may be organised in arbitrarily complex and structured forms, e.g. a Quality Parameter may be the compound of other specific Quality Parameters.

- A **Quality Parameter** may be evaluated by (**evaluatedBy**) a **Quantitative Measurement**.

Quantitative Measurements are based on collecting and interpreting ordinal data.

- A **Quality Parameter** may be evaluated by (**evaluatedBy**) a **Qualitative Measurement**.

  Qualitative Measurements are applied when the collected data are categorical 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 Measurements that could be applied to a DL are direct observation; participant observation; interviews; auditing; case study; collecting written feedback.
- A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL system as a whole (**Generic Quality Parameter**).

  This is a family of Quality Parameters reflecting the variety of facets that characterise the quality of the 'system' in its entirety, in particular the Digital Library, the Digital Library System and the Digital Library Management System.
- A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL Content (**Content Quality Parameter**).

  A Quality Parameter which reflects the variety of facets that characterise the quality of the Content, in particular Information Objects, in a Digital Library.
- A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL Functions (**Functionality Quality Parameter**).

  A Quality Parameter which reflects the variety of facets that characterise the quality of the Functionality, in particular Functions, of a Digital Library.
- A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL User (**User Quality Parameter**).

  A Quality Parameter may assess Actor profiles and User behaviour of a Digital Library.
- A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL Policies (**Policy Quality Parameter**).

  A Quality Parameter which reflects the variety of facets that characterise the quality of a set of Policies.
- A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the Architecture of the DLS (**Architecture Quality Parameter**).

  A Quality Parameter may assess the aspects related to the Digital Library System Architecture. The presence of good administration tools is crucial for configuring and monitoring the functioning of complex and distributed systems.

IV.5.6 Architecture-oriented Criteria

The following criteria have been selected to verify whether or not the 'digital library' conforms to the Digital Library Reference Model from the Architecture domain point of view.

MANDATORY

Regardless of the content, user, functionality, policy and quality characteristics of the 'digital library', the **Digital Library System** supporting its operation **must meet** at least the following criteria:

- The **Digital Library System** underlying the 'digital library' must have a well-defined **Software Architecture.**

The Software Architecture describes the digital library system enabling software by clearly defining how it is structured in components, i.e. programmes, how they communicate and are interrelated to offer the digital library service.

- The **Digital Library System** underlying the 'digital library' must have a well-defined **System Architecture.**

  The System Architecture is the conceptual model that defines the organisation and relations between the Hosting Nodes, i.e. the (virtual) hardware environments hosting and running the Software Components, and the Running Component, i.e. the running instances of a Software Component active on a Hosting Node.

- Every **Architectural Component** must have a unique identifier (**Resource Identifier**, **identifiedBy**).

  The use of a persistent identifier ensures that each DL Architecture Component is distinguishable from the remaining ones in the context of the same Digital Library System.

- The **Software Architecture** must consist of (**consistOf**) at least one well identified **Software Architecture Component**.

  The Software Architecture must include at least one Component, i.e. a software package, a web service, or a module, with well-defined interfaces, that encapsulates a set of related functions (or data).

- The **System Architecture** must consist of (**consistOf**) at least one **Hosting Node** and one **Running Component.**

  The System Architecture of a DLS is implemented by a set of components (System Component) running on servers which act as Hosting Nodes. The resulting system organisation (i.e., Software Components used and resulting Running Components and Hosting Nodes) can evolve over the time. A single Running Component hosted by a single Hosting Node corresponds to the minimal System Architecture structure.

RECOMMENDED

Additional desired properties of a 'digital library' (its Digital Library System) are:

- The 'digital library' service is deployed and operated by means of a **Digital Library Management System**.

  The Digital Library Management System facilitates the set up and maintenance of DL Systems by offering facilities for their production and administration. These facilities also assure a well-defined Quality of Service for the managed DL Systems.

- Every **Software Component** should be regulated by (**regulatedBy**) a **Licence.**

  The Licence is particular policy which specifies the permission on use, re-use and modification of the Software Component.

- The **Software Architecture** should be composed of more than one identifiable **Software Architecture Components**.

  A component-oriented approach for digital library systems offers many advantages with respect to system building, openness, and evolution, and it is thus preferable to other solutions especially for large systems.

- The **System Architecture** should be composed of more than one identifiable **System Architecture Components**.

A System Architecture based on a number of running components distributed on different hosting nodes offers many advantages with respect to system building, openness, and evolution, and it is thus preferable to other monolithic solutions especially when dealing with large systems.

- Every **Architectural Component** should conform to (**conformTo**) a **Framework Specification**.

  Architectural Components should interact through a Framework Specification. The Framework Specification prescribes the set of Interfaces to be implemented by the components and the protocols governing how components interact with each other. In so doing, it facilitates components composition and interoperability.

OPTIONAL

Finally, a 'digital library' (its Digital Library System) may also meet the following set of criteria:

- Every **Architectural Component**, be it a **Software Architecture Component** or a **System Architecture Component**, may exploit (**use**) one or more other not conflicting (**conflictWith**) **Architectural Components**.

  The exploitation of functionality offered by other components is a very common practice in software engineering. It reduces the complexity of the problem to be dealt with and favour reusability.

IV.6 Characterising the Reference Model Conformance Criteria

This section describes each criterion listed in the previous section according to the characterization introduced in Section IV.3.2. The overall set of criteria is grouped in three main subsets: mandatory criteria (cf. Section IV.6.1), recommended criteria (cf. Section IV.6.2) and optional criteria (cf. Section IV.6.3).

IV.6.1 Mandatory Features

IV.6.1.1 Content-oriented Mandatory Features

Criterion:	MF1. The Digital Library must manage a set of **Information Objects** and the set can not be empty.
Why it is needed:	By definition the purpose of a digital library is to collect, manage and preserve in perpetuity digital content.
DLRM Concept Identifier(s):	C7 Information Object
DLRM Relation Identifier(s):	N/A
Example:	A DL can offer access to different types of information objects, such as textual documents, metadata records, images, etc.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	MF2. Every **Information Object** must have (**identifiedBy**) a unique identifier (**Resource Identifier**).
Why it is needed:	This guarantees that each Information Object managed by the 'digital library' is distinguishable from the remaining ones in the context of the same 'digital library'.
DLRM Concept Identifier(s):	C2 Resource Identifier C7 Information Object
DLRM Relation Identifier(s):	R2 identifiedBy
Example:	A Uniform Resource Identifier (URI) is a string of characters used to identify DL resources. Such identification enables interaction with representations of the resource over DL networks using specific protocols.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	MF3. Every **Information Object** must have at least one element of **Metadata** (**hasMetadata**) associated with it.
Why it is needed:	This ensures that each Information Object is equipped with data supporting its management and use.

DLRM Concept Identifier(s):	C7 Information Object C11 Metadata
DLRM Relation Identifier(s):	R8 hasMetadata
Example:	Keywords are metadata which describe the content of a resource.
Suggested Type of Evidence:	Documentary Observational

Criterion:	MF4. Every **Information Object** must belong (**belongTo**) to at least one **Collection**.
Why it is needed:	Collections represent a traditional mechanism to organise Information Objects and to provide focused views of the Digital Library Information Object Resource Set. These focused views enable Actors to access thematic parts or specific types of the whole content.
DLRM Concept Identifier(s):	C7 Information Object C18 Collection
DLRM Relation Identifier(s):	R1 belongTo
Example:	The set of Information Objects characterized by having the same format, e.g. a video collection.
Suggested Type of Evidence:	Documentary

Criterion:	MF5. Every **Collection** must have (**identifiedBy**) a unique identifier (**Resource Identifier**).
Why it is needed:	This establishes that each Collection managed by the Digital Library is distinguishable from any others in the context of the same Digital Library.
DLRM Concept Identifier(s):	C2 Resource Identifier C18 Collection
DLRM Relation Identifier(s):	R2 identifiedBy
Example:	Digital Object Identifiers (DOIs) are specifications of Uniform Resource Identifiers (URIs), which are used to identify electronic resources managed by DLs such as e-journal articles.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	MF6. Every **Collection** must have at least one element of **Metadata** (**hasMetadata**) associated with it.
Why it is needed:	This asserts that each Collection is equipped with data supporting its management and use.
DLRM Concept Identifier(s):	C11 Metadata

	C18 Collection
DLRM Relation Identifier(s):	R8 hasMetadata
Example:	Information about the collection can concern its topics, its curators, etc.
Suggested Type of Evidence:	Documentary Observational

IV.6.1.2 User-oriented Mandatory Features

Criterion:	MF7. The DL must serve a clearly identified set of **Actors** and this can not be an empty set.
Why it is needed:	Actors represent the entities that interact with any digital library 'system', i.e., humans and inanimate entities such as software programs or physical instruments. This guarantees that they exist, i.e., there is no 'digital library' without users interacting with it.
DLRM Concept Identifier(s):	C22 Actor
DLRM Relation Identifier(s):	N/A
Example:	An academic DL offers its content and functionalities to the university staff and students in order to serve their research needs.
Suggested Type of Evidence:	Documentary Observational

Criterion:	MF8. Every **Actor** must have (**identifiedBy**) a unique **Resource Identifier**.
Why it is needed:	This guarantees that every Actor managed by the Digital Library is distinguishable from other Actors in the context of the same 'digital library'.
DLRM Concept Identifier(s):	C2 Resource Identifier C22 Actor
DLRM Relation Identifier(s):	R2 identifiedBy
Example:	User Identifiers (UIDs) and Group Identifiers (GID) are managed by DLs in order to authenticate Actors and (if they successfully authenticate) provide them access to their content and functionalities.
Suggested Type of Evidence:	Testimonial

Criterion:	MF9. Every **Actor** must be described (**model**) by at least one **Actor Profile**.
Why it is needed:	This guarantees that every Actor uses the 'digital library' and interacts with it as well as with other Actors in a personalised and codified way.

DLRM Concept Identifier(s):	C14 Actor Profile C22 Actor
DLRM Relation Identifier(s):	R12 model
Example:	An academic DL creates a group profile in order to capture the personal information of doctoral students in order to offer them specialized functionalities and advanced access to restricted materials
Suggested Type of Evidence:	Documentary Observational

Criterion:	MF10. Every **Actor** must act (**play**) with at least one **Role**.
Why it is needed:	This guarantees that an Actor cannot interact with a Digital Library if its role is not specified.
DLRM Concept Identifier(s):	C22 Actor C25 Role
DLRM Relation Identifier(s):	R11 play
Example:	The DL Designer of a Digital Library is in charge to identify the set of Collections and Functions constituting the Digital Library and define the characteristics of the User Domain (e.g. Roles and Groups), Policy Domain (e.g. establish the Content Policy) and Quality Domain (e.g. establish the Generic Quality Parameters) instances.
Suggested Type of Evidence:	Documentary Observational

Criterion:	MF11. The set of managed **Roles** must include the **DL Manager** Role.
Why it is needed:	DL Managers are Actors that exploit DLMS facilities to define, customise, and maintain the DL service.
DLRM Concept Identifier(s):	C25 Role C30 DL Manager
DLRM Relation Identifier(s):	N/A
Example:	A DL Manager in a University Digital Library is in charge to define and maintain the entire Digital Library service.
Suggested Type of Evidence:	Observational

Criterion:	MF12. The set of managed Roles must include the DL Software Developer Role.
Why it is needed:	DL Software Developers exploit Digital Library Management System facilities to create and customise the constituents of the Digital Library Systems.

DLRM Concept Identifier(s):	C25 Role C33 DL Software Developer
DLRM Relation Identifier(s):	N/A
Example:	A DL Software Developer of a scientific Digital Library is in charge to develop a new Software Component for managing Annotations of specific Information Objects.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	MF13. The set of managed **Roles** must include the **End-user** Role.
Why it is needed:	This guarantees that the digital library supports at least typical end-user roles, like content consumers, content creators or digital librarians.
DLRM Concept Identifier(s):	C25 Role C26 DL End-user
DLRM Relation Identifier(s):	N/A
Example:	In setting up the journal annual subscriptions, an academic DL needs to consider the needs of the faculty staff and students
Suggested Type of Evidence:	Documentary Observational

IV.6.1.3 Functionality-oriented Mandatory Features

Criterion:	MF14. The Digital Library must offer a well defined set of **Functions** and this can not be an empty set.
Why it is needed:	The purpose of the DL is to offer functions, i.e., a particular processing task that can be realised on a Resource or Resource Set as the result of an Action of a particular Actor.
DLRM Concept Identifier(s):	C36 Function
DLRM Relation Identifier(s):	N/A
Example:	A DL website gives access to the different DL functionalities, e.g. the search and browse functions
Suggested Type of Evidence:	Observational Documentary

Criterion:	MF15. Every **Function** must have (**identifiedBy**) a unique identifier (**Resource Identifier**).
Why it is needed:	A Function is a Resource, thus it must be identified by a persistent identifier if it is to be distinguished from other Functions managed by the DL.
DLRM Concept Identifier(s):	C2 Resource Identifier

	C36 Function
DLRM Relation Identifier(s):	R2 identifiedBy
Example:	A function identifier is a unique string, literal or arithmetic expression, representing the function. This identifier could then be used as a placeholder to indicate the function itself.
Suggested Type of Evidence:	Testimonial Documentary

Criterion:	MF16. Every **Function** must be performed (**perform**) by **Actors.**
Why it is needed:	DL Functions are the implementations of functions and services enabling Actors to interact with the DL.
DLRM Concept Identifier(s):	C22 Actor C36 Function
DLRM Relation Identifier(s):	R8 perform
Example:	A Content Consumer of a scientific Digital Library performs a Search action by invoking the implementation of a Search function using as constraints the topic created by a certain author in a specific period of time.
Suggested Type of Evidence:	Observational Documentary

Criterion:	MF17. Every **Actor** must be provided with (**perform**) **Functions** to **Access Resources**.
Why it is needed:	The DL must implement functions to enable actors to access, e.g., discover, acquire and visualize, all kind of Resources (Information Objects, Actors Profiles).
DLRM Concept Identifier(s):	C22 Actor C36 Function C37 Access Resource
DLRM Relation Identifier(s):	R8 perform
Example:	A DL End-user submits a query for the discovery of information related to a specific subject The system discovers all the Information Objects matching user's expectations and returns these back to the user with a visualization of the corresponding content.
Suggested Type of Evidence:	Documentary Observational

Criterion:	MF18. Every **Actor** must be provided with (**perform**) **Functions** to **Discover Resources**.
Why it is needed:	Actors must be able to find the desired Information Objects,

	search and access not only the DL Content, but also other Actors or Functions.
DLRM Concept Identifier(s):	C1 Resource C22 Actor C36 Function C38 Discover
DLRM Relation Identifier(s):	R8 perform
Example:	A user can *Search* using specific keywords for a specific subject or *Browse* the content from a list, a drop down menu or a set of links connected to the subject.
Suggested Type of Evidence:	Documentary Observational

Criterion:	MF19. Every **DL System Administrator** must be provided with (**perform**) **Functions** to **Manage & Configure DLS**.
Why it is needed:	DL must implement functions for handling the DLS and configuring its settings.
DLRM Concept Identifier(s):	C32 DL System Administrator C36 Function C103 Manage & Configure DLS
DLRM Relation Identifier(s):	R8 perform
Example:	Configuring the settings of the DLS, creating, withdrawing or updating components and managing architecture are operations performed in managing DLS.
Suggested Type of Evidence:	Documentary Observational Testimonial

IV.6.1.4 Policy-oriented Mandatory Features

Criterion:	MF20. The DL must be regulated by a well defined set of **Policies** (**regulatedBy**) and this can not be an empty set.
Why it is needed:	Policies are set of conditions, rules, terms or regulations governing the operation of DL.
DLRM Concept Identifier(s):	C121 Policy
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	When setting a browse function a DL will need to have a policy defining general issues of how the browse function will be presented on the DL website and how it will be accessed by the End-users.
Suggested Type of Evidence:	Documentary Observational

	Testimonial

Criterion:	MF21. **Access Policies** must regulate the use of the Digital Library by **Actors** (**regulatedBy**).
Why it is needed:	Access Policies are essential to establish conditions, rules, terms or regulations governing the interactions between the Digital Library and Actors.
DLRM Concept Identifier(s):	C22 Actor C152 Access Policy
DLRM Relation Identifier(s):	R7 regulatedBy
Example(s):	Access to Resources provided on the basis of IP address identification is an example of Access Policy.
Suggested Type of Evidence:	Documentary Observational

Criterion:	MF22. Every **Policy** must be addressed **(govern**) at least to an **Actor** (**regulatedBy).**
Why it is needed:	Defining the recipients of a Policy ensures the interaction between the Digital Library and its Actor(s).
DLRM Concept Identifier(s):	C22 Actor C121 Policy
DLRM Relation Identifier(s):	R7 regulatedBy R26 govern
Example(s):	Offering a print-on-demand service for selected Resources is an example of Collection Delivery Policy, which will specify to whom that service is offered.
Suggested Type of Evidence:	Documentary

Criterion:	MF23. Every **Policy** must have clearly defined scope(s) and characteristics (**Policy Quality Parameter**).
Why it is needed:	A Policy must have defined objectives and consequences affecting the DL system as a whole, a certain domain, a specific task or entity.
DLRM Concept Identifier(s):	C121 Policy C200 Policy Quality Parameter
DLRM Relation Identifier(s):	N/A
Example(s):	A lack of knowledge of the technology used may lead to undesired DLS behaviour and unexpected consequences to the System Policy. A policy limiting the rate of sending data over a network cannot be enforced in a DL if the underlying DLS does not provide

	some means for adjusting the data transmission rate.
Suggested Type of Evidence:	Documentary Observational

IV.6.1.5 Quality-oriented Mandatory Features

Criterion:	MF24. A Digital Library (actually its **Resource(s)**) must be characterised by a set of **Quality parameter(s)** (**hasQuality**) and this can not be an empty set.
Why it is needed:	Any DL can be considered from a quality point of view by a DL Actor. The expression of the Actor's assessment is the Quality Parameter.
DLRM Concept Identifier(s):	C1 Resource C162 Quality Parameter
DLRM Relation Identifier(s):	R6 hasQuality
Example:	One of the main Quality Parameters in relation to an information retrieval system is its effectiveness, meaning its capability to respond to user information needs with relevant items. This Quality Parameter can be evaluated according to many different Measures, such as precision and recall: precision evaluates effectiveness in the sense of the ability of the system to reject useless items, while recall evaluates effectiveness in the sense of the ability of the system to retrieve useful items. The actual values for precision and recall are Measurements and are usually computed using standard tools, such as trec_eval, which are Actors, but in this case not human.
Suggested Type of Evidence:	Documentary

Criterion:	MF25. Every **Quality Parameter** must represent the assessment of a Digital Library **Actor**, whether human or machine, on a **Resource** (**expressAssessment**).
Why it is needed:	A Quality Parameter is always the expression of an assessment made by an Actor on a Resource.
DLRM Concept Identifier(s):	C1 Resource C22 Actor C162 Quality Parameter
DLRM Relation Identifier(s):	R31 expressAssessment
Example:	User Satisfaction can be explicitly assessed by making use of surveys and questionnaires addressed to the End-Users.
Suggested Type of Evidence:	Documentary

IV.6.1.6 Architecture-oriented Mandatory Features

Criterion:	MF26. The **Digital Library System** underlying the 'digital library' must have a well-defined **Software Architecture.**
Why it is needed:	The Software Architecture describes the digital library system enabling software by clearly defining how it is structured in, Components, i.e. programmes, how they communicate and are interrelated to offer the digital library service.
DLRM Concept Identifier(s):	C220 Software Architecture C225 Digital Library System
DLRM Relation Identifier(s):	N/A
Example:	The software packages implementing the DLS specific function, their interfaces and the dependencies among them.
Suggested Type of Evidence:	Documentary

Criterion:	MF27. The **Digital Library System** underlying the 'digital library' must have a well-defined **System Architecture.**
Why it is needed:	The System Architecture is the conceptual model that defines the organisation and relations between the Hosting Nodes, i.e. the (virtual) hardware environments hosting and running the Software Components, and the Running Component, i.e. the running instances of a Software Component active on a Hosting Node.
DLRM Concept Identifier(s):	C221 System Architecture C225 Digital Library System
DLRM Relation Identifier(s):	N/A
Example:	The set of running Web Service implementing the functionality provided by the DLS and the servers hosting them.
Suggested Type of Evidence:	Documentary

Criterion:	MF28. Every **Architectural Component** must have a unique identifier (**Resource Identifier**, **identifiedBy**).
Why it is needed:	The use of a persistent identifier ensures that each DL Architecture Component is distinguishable from the remaining ones in the context of the same Digital Library System.
DLRM Concept Identifier(s):	C2 Resource Identifier C211 Architectural Component
DLRM Relation Identifier(s):	R2 identifiedBy
Example:	URI and URN are exemplars of identifiers that might be used to refer to Architectural Components.
Suggested Type of Evidence:	Documentary

Criterion:	MF29. The **Software Architecture** must consist of (**consistOf**) at

	least one well identified **Software Architecture Component**.
Why it is needed:	The Software Architecture must include at least one Component, i.e. a software package, a web service, or a module, with well-defined interfaces, that encapsulates a set of related functions (or data).
DLRM Concept Identifier(s):	C212 Software Architecture Component C220 Software Architecture
DLRM Relation Identifier(s):	N/A
Example:	A Software Architecture which distinguishes components for the storage, access and visualization of the content.
Suggested Type of Evidence:	Documentary

Criterion:	MF30. The System Architecture must consist of (consistOf) at least one Hosting Node and one Running Component.
Why it is needed:	The System Architecture of a DLS is implemented by a set of components (System Component) running on servers which act as Hosting Nodes. The resulting system organisation (i.e., Software Components used and resulting Running Components and Hosting Nodes) can evolve over the time. A single Running Component hosted by a single Hosting Node corresponds to the minimal System Architecture structure.
DLRM Concept Identifier(s):	C218 Running Component C219 Hosting Node C221 System Architecture
DLRM Relation Identifier(s):	N/A
Example:	A system architecture whose Running Components are distributed on Hosting Nodes of different organizations responsible for the sustainability and maintenance of the DLS.
Suggested Type of Evidence:	Documentary

IV.6.2 Recommended Features

IV.6.2.1 Content-oriented Recommended Features

Criterion:	RF1. Every **Information Object** should conform to (**hasFormat**) an explicit and known format (**Resource Format**).
Why it is needed:	This guarantees that the system is aware of the “structure” each Information Object conforms to and that this structure is publicly declared thus making the Information Object usable by third party actors whether human or machine. The notion of Resource Format is wide and might range from an abstract one (e.g. “enhanced publication”) to a concrete one (e.g. PDF).

DLRM Concept Identifier(s):	C5 Resource Format C7 Information Object
DLRM Relation Identifier(s):	R3 hasFormat
Example:	A HTML webpage is a compound information object conforming to the structure defined as "webpage" and to the HTML format.
Suggested Type of Evidence:	Documentary

Criterion:	RF2. Every **Metadata** should conform to (**hasFormat**) an explicit and known format (**Resource Format**).
Why it is needed:	This criterion – a specialization of the previous one – ensures that the system is aware of the "structure" the metadata object conforms to and that this structure is publicly declared so that it can be used by third party actors whether human or machine. In this case the notion of Resource Format corresponds to the notion of metadata schema.
DLRM Concept Identifier(s):	C5 Resource Format C11 Metadata
DLRM Relation Identifier(s):	R3 hasFormat
Example:	A set of information objects share the common schema called "thesis" which defines the set of properties of those information objects.
Suggested Type of Evidence:	Documentary

Criterion:	RF3. Every **Annotation** should conform to (**hasFormat)** an explicit and known format (**Resource Format**).
Why it is needed:	This criterion guarantees that the system is aware of the particular "structure" to which the annotation object conforms. Being publicly declared, this structure can be used by third party actors whether human or machine.
DLRM Concept Identifier(s):	C5 Resource Format C17 Annotation
DLRM Relation Identifier(s):	R3 hasFormat
Example:	A set of annotations share the common schema called "comment" which defines the set of properties of those annotations.
Suggested Type of Evidence:	Documentary

Criterion:	RF4. Every **Collection** should have a well-defined intension, i.e., the set of criteria characterising **Collection** membership (**hasIntension**), and should have a well-defined extension, i.e., the set of **Information Objects** belonging to the collection (**hasExtension**).

Why it is needed:	The collection concept is fundamental to keep the set of Information Objects organised. Because of this, it is recommended that both the set of Information Objects belonging to a collection and the criteria driving the membership of an information object into a collection are clearly defined.
DLRM Concept Identifier(s):	C7 Information Object C18 Collection
DLRM Relation Identifier(s):	R2 hasIntension R3 hasExtension
Example:	The statement "information objects related to events happened in the period 1900 -1950" is the intention of a collection. The set of information objects for which the above statement is true is the extension of that collection.
Suggested Type of Evidence:	Documentary

Criterion:	RF5. Every **Information Object** should be regulated (**regulatedBy**) by **Policies**.
Why it is needed:	Policies are essential to establish conditions, rules, terms or regulations governing the management of information objects.
DLRM Concept Identifier(s):	C7 Information Object C121 Policy
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	Access to the digital reproduction of a portrait is regulated by copyright policy.
Suggested Type of Evidence:	Documentary Observational

IV.6.2.2 User-oriented Recommended Features

Criterion:	RF6. Every **Actor** should perform (**perform**) **Actions** that apply (**apply**) **Functions** and concern (**concern**) **Resources**.
Why it is needed:	Every Actor that interacts with a digital library should be able to perform certain Actions that involve the application of Functions and specific Resources.
DLRM Concept Identifier(s):	C1 Resource C22 Actor C34 Action C36 Function
DLRM Relation Identifier(s):	R8 perform R9 apply

	R10 concern
Example:	A Content Consumer of a scientific Digital Library can perform an Action than applies the Search function in order to explore all the papers on a given research topic created by a certain author and published in a specific period of time.
Suggested Type of Evidence:	Observational

IV.6.2.3 Functionality-oriented Recommended Features

Criterion:	RF7. Every **Function** should be able to interact with (**intertactWith**) other **Functions**.
Why it is needed:	DL functions should exchange information with other functions regulating their behaviour and performance.
DLRM Concept Identifier(s):	C36 Function
DLRM Relation Identifier(s):	R13 interactWith
Example:	The Acquire function may interact with Transform function so as to facilitate the extraction of information objects in an appropriate format requested by the user, e.g., content exported into a pdf format.
Suggested Type of Evidence:	Testimonial

Criterion:	RF8. **Functions** to **Acquire** (**actOn**) **Resources** should be provided.
Why it is needed:	DL functionality should enable Actors to retain Resources e.g., Information Objects and Actor Profiles, in existence past their interaction with the Digital Library System.
DLRM Concept Identifier(s):	C1 Resource C36 Function C41 Acquire
DLRM Relation Identifier(s):	R15 actOn
Example:	A user downloads and locally stores Information Objects (e.g., video, pdf files) from a list of Information Objects returned to him /her after performing a query.
Suggested Type of Evidence:	Documentary Observational

Criterion:	RF9. **Functions** to **Browse** (**actOn**) the **Resources** should be provided.
Why it is needed:	DL should implement services enabling Actors (virtual or real) to browse the available DL content, user profiles, policies, etc.
DLRM Concept Identifier(s):	C1 Resource

	C36 Function C39 Browse
DLRM Relation Identifier(s):	R15 actOn
Example:	A user requests the display of all the Digital Library Information Objects (e.g. movies) per subject area (e.g., comedies, drama).
Suggested Type of Evidence:	Documentary Observational

Criterion:	RF10. **Functions** to **Search** (**actOn**) the **Resources** should be provided.
Why it is needed:	Actors should be able to look for specific objects held within the DL by expressing queries and by entering specific keywords and constraints.
DLRM Concept Identifier(s):	C1 Resource C36 Function C40 Search
DLRM Relation Identifier(s):	R15 actOn
Example:	The "Query by Example" of a Music DL searches for Resources similar to a provided sample audio file.
Suggested Type of Evidence:	Documentary Observational

Criterion:	RF11. **Functions** to **Visualize** (**actOn**) the Actor's requested **Resources** should be provided.
Why it is needed:	A DL should deliver to Actor the requested information using the appropriate visualizations to produce comprehensive and well-presented objects, lists and query result sets.
DLRM Concept Identifier(s):	C1 Resource C36 Function C42 Visualise
DLRM Relation Identifier(s):	R15 actOn
Example:	Animation and the drawing of diagrams are examples of the Visualise function.
Suggested Type of Evidence:	Documentary Observational

Criterion:	RF12. **Functions** to **Manage Information Object**(s) (**actOn**) should be provided.
Why it is needed:	A DL should implement functions to handle, i.e., disseminate, publish, process, analyze and transform, the Content of the DL, i.e.,

	Information Objects.
DLRM Concept Identifier(s):	C7 Information Object C36 Function C51 Manage Information Object
DLRM Relation Identifier(s):	R15 actOn
Example:	A Librarian approves the publishing of a newly inserted Information Object to the system. Disseminating, transforming, and composing are some other examples of content management.
Suggested Type of Evidence:	Documentary Observational Testimonial

Criterion:	RF13. **Functions** to **Manage Actor**(s) (**actOn**) should be provided.
Why it is needed:	A DL should implement Functions to establish registered actors, personalize their preference and apply user profiles.
DLRM Concept Identifier(s):	C22 Actor C36 Function C70 Manage Actor
DLRM Relation Identifier(s):	R15 actOn
Example:	A DL User's registration and logging in to the DLS by entering the authentication credentials (username and password) are some examples of Manage Actor functions.
Suggested Type of Evidence:	Documentary Observational Testimonial

Criterion:	RF14. **Functions** to **Manage DL** specific domains in a large scale should be provided.
Why it is needed:	The DL should implement services and mechanisms to handle DL domains as a whole, e.g., Manage (import, export) all the Content of DL rather than handling each Information Object individually.
DLRM Concept Identifier(s):	C36 Function C89 Manage DL
DLRM Relation Identifier(s):	N/A
Example:	The mass import of provenance metadata contained in a collection is an example of the Manage DL function.
Suggested Type of Evidence:	Documentary Observational Testimonial

IV.6.2.4 Policy-oriented Recommended Features

Criterion:	RF15. Every **Policy** should be expressed by (**expressedBy**) an **Information Object**.
Why it is needed:	The digital representation of a Policy ensures its controlled description, management and use within the Digital Library. This representation enables automatic enforcing. Moreover, it is a prerequisite for a series of other automatic actions including policy comparison, policy reconciliation and policy interoperability.
DLRM Concept Identifier(s):	C7 Information Object C121 Policy
DLRM Relation Identifier(s):	R15 expressedBy
Example:	A License is a Digital Rights Management Policy which may be issued for specific uses of DL Resources, or for designated functionality features that should be downloaded and installed by the users. GPL (GNU General Public License) is an example of license for free software.
Suggested Type of Evidence:	Documentary

Criterion:	RF16. Every **Policy** should have (**identifiedBy**) a unique identifier (**Resource Identifier**).
Why it is needed:	The use of a persistent identifier ensures that each DL Policy is distinguishable from the others in the context of the same Digital Library.
DLRM Concept Identifier(s):	C2 Resource Identifier C121 Policy
DLRM Relation Identifier(s):	R2 identifiedBy
Example:	Certificate Policies require object identifiers (OID) to name every object type in public key certificates such as X.509.
Suggested Type of Evidence:	Documentary Observational Testimonial

Criterion:	RF17. Every **Policy** should have (**hasFormat**) a known format (**Resource Format**).
Why it is needed:	The implementation of a Policy in a known format guarantees that the system is aware of which “structure” each Policy conforms to and that this structure is publicly declared as to be used by third party actors whether human or machine.
DLRM Concept Identifier(s):	C5 Resource Format C121 Policy
DLRM Relation Identifier(s):	R3 hasFormat

Example:	Access policies are usually published on the DL websites as HTML documents.
Suggested Type of Evidence:	Documentary

IV.6.2.5 Quality-oriented Recommended Features

Criterion:	RF18. Every **Quality Parameter** should be identified by (**identifiedBy**) a unique identifier (**Resource Identifier**).
Why it is needed:	The use of a persistent identifier ensures that each Quality Parameter is distinguishable from the remaining ones in the context of the same 'digital library'.
DLRM Concept Identifier(s):	C2 Resource Identifier C162 Quality Parameter
DLRM Relation Identifier(s):	R2 identifiedBy
Example:	Key Performance Indicator (KPI) Identifiers are used to manage KPI values.
Suggested Type of Evidence:	Documentary

Criterion:	RF19. Every **Quality Parameter** should be expressed by (**expressedBy**) an **Information Object**.
Why it is needed:	The digital representation of a Quality Parameter ensures its controlled description, management and use within the Digital Library. This representation is a prerequisite for a series of other automatic actions including the assessment of Digital Library content and services, and quality interoperability.
DLRM Concept Identifier(s):	C7 Information Object C162 Quality Parameter
DLRM Relation Identifier(s):	R15 expressedBy
Example:	User Satisfaction can be expressed in natural language or with numerical values.
Suggested Type of Evidence:	Documentary

Criterion:	RF20. Every **Quality Parameter** should be evaluated by (**evaluatedBy)** specific **Measurements**.
Why it is needed:	In accordance with a selected Measurement, a Quality Parameter should have a specific value (e.g. the Measure).
DLRM Concept Identifier(s):	C156 Measurement C162 Quality Parameter
DLRM Relation Identifier(s):	R32 evaluatedBy
Example:	The measure of a DL total reference activity is constituted by the

	total number of reference questions received.
Suggested Type of Evidence:	Documentary

Criterion:	RF21. Every **Quality Parameter** should be measured (**measuredBy**) by a **Measure**.
Why it is needed:	Any Quality Parameter should be managed by the Digital Library according to different Measurements, which provide procedures for assessing different aspects of each Quality Parameter and assigning it a value.
DLRM Concept Identifier(s):	C161 Measure C162 Quality Parameter
DLRM Relation Identifier(s):	R33 measuredBy
Example:	Examples of measurements are data collection of full-text downloads.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	RF22. Every **Quality Parameter** should be specified (**regulatedBy**) by **Policies**.
Why it is needed:	The Digital Library should have policies governing the evaluation and the assessment of its systems and facets.
DLRM Concept Identifier(s):	C121 Policy C162 Quality Parameter
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	A list of objectives of a DL published on its website which includes a regular follow-up of user satisfaction is a specific quality policy.
Suggested Type of Evidence:	Documentary

IV.6.2.6 Architecture-oriented Recommended Features

Criterion:	RF23. The 'digital library' service is deployed and operated by means of a **Digital Library Management System**.
Why it is needed:	The Digital Library Management System facilitates the set up and maintenance of DL and the Digital Library Systems by offering facilities for their production and administration. These facilities also assure a well-defined Quality of Service for the managed DL Systems.
DLRM Concept Identifier(s):	C226 Digital Library Management System
DLRM Relation Identifier(s):	N/A
Example:	The DLS supporting the "Scuola Normale Digital Library" http://opendlib.sns.it/ is operated through the OpenDLib Digital

	Library Management System.
Suggested Type of Evidence:	Documentary

Criterion:	RF24. Every **Software Component** should be regulated by (**regulatedBy**) a **Licence.**
Why it is needed:	The Licence is particular policy which specifies the permission on use, re-use and modification of the Software Component.
DLRM Concept Identifier(s):	C143 License C213 Software Component
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	European Union Public Licence (EUPL), GNU General Public License (GPL), Apache Licence.
Suggested Type of Evidence:	Documentary

Criterion:	RF25. The **Software Architecture** should be composed of more than one identifiable **Software Architecture Components**.
Why it is needed:	A component-oriented approach for digital library systems offers many advantages with respect to system building, openness, and evolution, and it is thus preferable to other solutions especially for large systems.
DLRM Concept Identifier(s):	C212 Software Architecture Component C220 Software Architecture
DLRM Relation Identifier(s):	N/A
Example:	Exemplars of Software Architecture Components are software packages implementing a specific Function, software artefacts supporting the implementation of a specific Functions, e.g. a Relational Database Management System (RDBMS).
Suggested Type of Evidence:	Documentary

Criterion:	RF26. The **System Architecture** should be composed of more than one identifiable **System Architecture Components**.
Why it is needed:	A System Architecture based on a number of running components distributed on different hosting nodes offers many advantages with respect to system building, openness, and evolution, and it is thus preferable to other monolithic solutions especially when dealing with large systems.
DLRM Concept Identifier(s):	C217 System Architecture Component C221 System Architecture
DLRM Relation Identifier(s):	N/A
Example:	Usually, in a federated digital library system the running components supporting the storage and curation of the content

	are hosted on servers local to the participating organizations while the running components for search and visualization of the results are hosted on the servers of the organization responsible for the DLS operation.
Suggested Type of Evidence:	Documentary

Criterion:	RF27. Every **Architectural Component** should conform to (**conformTo**) a **Framework Specification**.
Why it is needed:	Architectural Components should interact through a Framework Specification. The Framework Specification prescribes the set of Interfaces to be implemented by the components and the protocols governing how components interact with each other. In so doing, it facilitates components composition and interoperability.
DLRM Concept Identifier(s):	C211 Architectural Component C216 Framework Specification
DLRM Relation Identifier(s):	R5 conformTo
Example:	The Web Service Resource Framework provides specification for Web Services.
Suggested Type of Evidence:	Documentary

IV.6.3 Optional Features

IV.6.3.1 Content-oriented Optional Features

Criterion:	OF1. An **Information Object** may have multiple **Editions** (**hasEdition**) each represented by a different related **Information Object**.
Why it is needed:	A 'digital library' might be employed to manage multiple editions of the same work. In these cases it is important to deal effectively with the edition concept.
DLRM Concept Identifier(s):	C7 Information Object C8 Edition
DLRM Relation Identifier(s):	R4 hasEdition
Example:	A scientific report is represented in its draft version and in its version as submitted for publication.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF2. An **Information Object** may have multiple **Views** (**hasView**) each represented by a different related **Information Object**.

Why it is needed:	A ‘digital library’ might be called to manage multiple “views”/“expressions” of the same conceptual work. In these cases it is important to properly deal with the view concept.
DLRM Concept Identifier(s):	C7 Information Object C9 View
DLRM Relation Identifier(s):	R5 hasView
Example:	The outcomes of a workshop represented in a “full view” containing a preface and the whole set of the accepted papers organized per sessions, as well as in an “informative view” reporting the goal of the workshop and the title list of the accepted papers.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF3. An **Information Object** may have multiple **Manifestations** (**hasManifestation**) each represented by a different related **Information Object**.
Why it is needed:	A ‘digital library’ might be called to manage multiple “items” of the same conceptual work or view. In these cases it is important to properly deal with the manifestation concept.
DLRM Concept Identifier(s):	C7 Information Object C10 Manifestation
DLRM Relation Identifier(s):	R6 hasManifestation
Example:	The information object representing a conference lecture is embodied into a PDF file as well as into a MPEG file containing the video recording of that lecture.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF4. An **Information Object** may be compound (**hasPart**), i.e., it may consist of multiple **Information Objects**.
Why it is needed:	Modern ‘digital libraries’ are usually expected to deal with emerging forms of “documents”. Very often such a “documents” consists of aggregates of other objects (of different media).
DLRM Concept Identifier(s):	C7 Information Object
DLRM Relation Identifier(s):	R16 hasPart
Example:	A web page is a compound information object which includes text, images, and hyperlinks.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF5. An **Information Object** may be associated (**associatedWith**) with other **Information Objects** for a certain **Purpose**.
Why it is needed:	Managing compound objects may require links other objects. The motivations leading to linking are diverse and context specific, e.g., citation and lineage.
DLRM Concept Identifier(s):	C7 Information Object C222 Purpose
DLRM Relation Identifier(s):	R18 associatedWith
Example:	Annotations including other works' citations link the content of different Information Objects.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF6. An **Information Object** may have multiple elements of **Metadata** (**hasMetadata**) associated with it.
Why it is needed:	Metadata are a type of Information Object intended to support the management and use of the Information Objects to which they are attached. Different metadata can be conceived to support diverse needs. The majority of 'digital libraries' tend to deal with a single metadata format.
DLRM Concept Identifier(s):	C7 Information Object C11 Metadata
DLRM Relation Identifier(s):	R8 hasMetadata
Example:	An object of a digital art collection has descriptive metadata as well as preservation metadata which record provenance information and planned preservation actions.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF7. An **Information Object** may be associated with multiple **Annotations** (**hasAnnotation**).
Why it is needed:	Annotations are kinds of Information Objects that are attached to existing Information Objects for various purposes including objects enrichment and cooperative working.
DLRM Concept Identifier(s):	C7 Information Object C17 Annotation
DLRM Relation Identifier(s):	R14 hasAnnotation
Example:	An online collection of pictures can be annotated by authenticated users which can also comment other users' annotations.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF8. A **Collection** may be associated with multiple **Metadata** (**hasMetadata**).
Why it is needed:	According to the Reference Model, Collections are a type of Information Object. Because of this, they inherit all the features of Information Objects and benefit of multiple metadata.
DLRM Concept Identifier(s):	C11 Metadata C18 Collection
DLRM Relation Identifier(s):	R8 hasMetadata
Example:	A collection requiring different metadata for different categories of users.
Suggested Type of Evidence:	Documentary

Criterion:	OF9. A **Collections** may be associated with multiple **Annotations** (**hasAnnotation**).
Why it is needed:	According to the Reference Model, Collections are a type of Information Object. Because of this, they inherit all the features of Information Objects and benefits of multiple annotations.
DLRM Concept Identifier(s):	C17 Annotation C18 Collection
DLRM Relation Identifier(s):	R14 hasAnnotation
Example:	A collection accrued under the responsibility of different curators, each giving the reasons of their choices in a different annotation.
Suggested Type of Evidence:	Documentary

IV.6.3.2 User-oriented Optional Features

Criterion:	OF10. **Actors** may belong to (**belongTo**) more than one **Group**.
Why it is needed:	During the interaction of an Actor with a Digital Library, the Actor may communicate or collaborate with other Actors that belong to various Groups; thus, the specific Actor may participate in different Groups. The concept of Group in the User domain has commonalities with the concept of Collection in the Content domain, it is a mechanism to organise Actors.
DLRM Concept Identifier(s):	C22 Actor C23 Group
DLRM Relation Identifier(s):	R1 belongTo
Example:	An End-user of a Music Digital Library that can belong to the Group of Librarians entitled to curate The Beatles Collection, but also can belong to the Group of Librarians entitled to curate Elvis Presley Collection.

Suggested Type of Evidence:	Observational

IV.6.3.3 Functionality-oriented Optional Features

Criterion:	OF11. **Functions** may depend on (**influencedBy**) the **Actor's Profile** who invokes them.
Why it is needed:	DL Functions that are offered to the Actor(s) may be customized according to his/her profile, DLS role and rights and /or personal preferences.
DLRM Concept Identifier(s):	C14 Actor Profile C22 Actor C36 Function
DLRM Relation Identifier(s):	R14 influencedBy
Example:	The Search function is influenced by the Actor Profile of the Actor that performs it by personalizing the returned Result Set.
Suggested Type of Evidence:	Observational Testimonial

Criterion:	OF12. **Functions** may consist of other parts (**hasPart**), i.e., sub-functions.
Why it is needed:	Functions may be organized in arbitrarily complex workflows, based on composition and linking facilities.
DLRM Concept Identifier(s):	C36 Function
DLRM Relation Identifier(s):	R16 hasPart
Example:	The Acquire function has parts the Search, Transform and Visualize subfunctions.
Suggested Type of Evidence:	Testimonial

Criterion:	OF13. **Functions** may be enriched with **Metadata** (**hasMetadata**) and **Annotation** (**hasAnnotation**).
Why it is needed:	DL Functions may have a description, which tells what the function does and how a system or human can interact with it.
DLRM Concept Identifier(s):	C11 Metadata C17 Annotation C36 Function
DLRM Relation Identifier(s):	R8 hasMetadata R14 hasAnnotation
Example:	Function A, which is implemented by Software Company B, is enriched with the administration online help on the company website.

Suggested Type of Evidence:	Testimonial

Criterion:	OF14. **Functions** may enable (**actOn**) **Actors** (virtual or real) to **Collaborate** with each other.
Why it is needed:	Actors (virtual or real) may act as peers who are able to communicate, share and exchange information collaboratively.
DLRM Concept Identifier(s):	C22 Actor C36 Function C84 Collaborate
DLRM Relation Identifier(s):	R15 actOn
Example:	User John Doe submits a new Information Object that is published by the DL operator and accessed by User John Smith.
Suggested Type of Evidence:	Observational Testimonial

IV.6.3.4 Policy-oriented Optional Features

Criterion:	OF15. A **Policy** may regulate (**regulatedBy**) the service of the system as a whole (**System Policy**).
Why it is needed:	Generic processes within the 'digital library' may be regulated by policies.
DLRM Concept Identifier(s):	C121 Policy C130 System Policy
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	System Policies cover most general processes in the digital library, such as regulation of changes or management of resources.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF16. A **Policy** may regulate (**regulatedBy**) functionalities related to Content (**Content Policy**).
Why it is needed:	A Policy may regulate processes related to the Content domain.
DLRM Concept Identifier(s):	C121 Policy C136 Content Policy
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	The issues on strategic planning and development of the Content of a Digital Library are addressed in the Collection Development Policy.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF17. A **Policy** may regulate (**regulatedBy**) DL Functions (**Functionality Policy**).
Why it is needed:	DL Functions' lifetime and behaviour may be governed by specific policies.
DLRM Concept Identifier(s):	C121 Policy C151 Functionality Policy
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	Taking care of the security of the Digital Library is a serious concern, for which the practical implementation would be a Security Policy.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF18. A **Policy** may regulate (**regulatedBy**) User profiles and behaviour (**User Policy**).
Why it is needed:	A Policy may regulate processes related to the User domain.
DLRM Concept Identifier(s):	C121 Policy C145 User Policy
DLRM Relation Identifier(s):	R7 regulatedBy
Example:	All Policies that regulate issues regarding digital rights and user behaviour.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF19. A **Policy** may be (**isa**) extrinsic (**Extrinsic Policy**).
Why it is needed:	A Policy may be imposed from outside the organisation domain of the 'digital library', e.g., by wider organisations regulating the Digital Library itself, by national and international laws, or by customs.
DLRM Concept Identifier(s):	C121 Policy C122 Extrinsic Policy
DLRM Relation Identifier(s):	R1 isa
Example:	Legal and regulatory frameworks of a specific country applied to a Digital Library developed by a local body.
Suggested Type of Evidence:	Documentary Observational Testimonial

Criterion:	OF20. A **Policy** may be (**isa**) intrinsic (**Intrinsic Policy**).

Why it is needed:	The Policy governing the Digital Library may be defined and determined by the Digital Library organisation itself. Intrinsic Policy manifests the Policy principles implemented in the DL. A Policy that is defined by the DL or its organisational context that reflects the organisation's mission and objectives, the intended expectations as to how Actors will interact with the DL, and the expectations of Content Creators as to how their content will be used.
DLRM Concept Identifier(s):	C121 Policy C123 Intrinsic Policy
DLRM Relation Identifier(s):	R1 isa
Example:	The mission defined by the DL reflects the organisation's mission and objectives, the intended expectations as to how Actors will interact with the DL, and the expectations of Content Creators as to how their content will be used.
Suggested Type of Evidence:	Documentary Observational Testimonial

Criterion:	OF21. A **Policy** may be (**isa**) implicit (**Implicit Policy**).
Why it is needed:	The Policy governing the Digital Library may be inherent by accident or design. Implicit Policies usually arise as a result of ad-hoc decisions taken at system development level or as a consequence of the inadequate testing of a DLS that results in an interaction of Policies leading to unintended policy deployment.
DLRM Concept Identifier(s):	C121 Policy C125 Implicit Policy
DLRM Relation Identifier(s):	R1 isa
Example:	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.
Suggested Type of Evidence:	Observational Testimonial

Criterion:	OF22. A **Policy** may be (**isa**) explicit (**Explicit Policy**).
Why it is needed:	Explicit Policy is a Policy defined by the DL managing organisation and reflecting the objectives of the DL and how it wishes its users to interact with the DL. The implementation of an Explicit Policy at the Digital Library Management System level corresponds to the definition and Actor expectations.
DLRM Concept Identifier(s):	C121 Policy C124 Explicit Policy

DLRM Relation Identifier(s):	R1 isa
Example:	Limitation for upload of files over a specified size, e.g. over 1 MB can be clearly stated on the DL website, and explained within the Submission and Resubmission Policy.
Suggested Type of Evidence:	Documentary

Criterion:	OF23. A **Policy** may be (**isa**) prescriptive (**Prescriptive Policy**).
Why it is needed:	The Policy governing the Digital Library may constrain the interactions between DL Actors (virtual or real) and the DL. Prescriptive Policies can cover a broad range of Policies from the kind of Function to which specific types of Actors can have access, to those that govern Collection development.
DLRM Concept Identifier(s):	C121 Policy C126 Prescriptive Policy
DLRM Relation Identifier(s):	R1 isa
Example:	Termination of file upload, if the file is of a format that is not permitted, is an example of action taken as a result of a Prescriptive Policy.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF24. A **Policy** may be (**isa**) descriptive (**Descriptive Policy**).
Why it is needed:	Descriptive Policies are used to present the aspects of a particular Policy in the form of explanation. A Descriptive Policy is a Policy that describe modes of behaviour, expectations of Actor interaction, collecting and use guidelines, but which do not manifest themselves through the automated application of rules, as a Prescriptive Policy does.
DLRM Concept Identifier(s):	C121 Policy C127 Descriptive Policy
DLRM Relation Identifier(s):	R1 isa
Example:	The Collection Development Policy describes the scope and coverage of the DL.
Suggested Type of Evidence:	Documentary

Criterion:	OF25. A **Policy** may be (**isa**) enforced (**Enforced Policy**).
Why it is needed:	The Policy governing the Digital Library may be deployed and strictly applied within the DL. An Enforced Policy is a Policy applied consistently and strictly in the DL. Monitoring and reporting tools are necessary to follow up how the Policy is being applied.
DLRM Concept Identifier(s):	C121 Policy

	C128 Enforced Policy
DLRM Relation Identifier(s):	R1 isa
Example:	A Charging Policy, which has been introduced into the DL, is an example of Enforced Policy.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	OF26. A **Policy** may be (**isa**) voluntary (**Voluntary Policy**).
Why it is needed:	The Policy governing the Digital Library may be monitored by an actor (human or machine). Voluntary Policy basically means a Policy that is followed according to the decision of the Actor. This is valid for all Policies for which its application is a matter of choice. In some cases, users may comply with Policies that are not officially communicated.
DLRM Concept Identifier(s):	C121 Policy C129 Voluntary Policy
DLRM Relation Identifier(s):	R1 isa
Example:	The Collection Development Policy might be outlined in broad terms, but not enforced in practice
Suggested Type of Evidence:	Documentary Observational Testimonial

Criterion:	OF27. A **Policy** may be compound (**hasPart**).
Why it is needed:	A Policy may be organised in arbitrarily complex and structured forms. A compound policy may be obtained by properly combining constituent Policies.
DLRM Concept Identifier(s):	C121 Policy
DLRM Relation Identifier(s):	R16 hasPart
Example:	A Security Policy may be the combination of Access Policy, Privacy Policy and Risk Management Policy.
Suggested Type of Evidence:	Documentary Observational

IV.6.3.5 Quality-oriented Optional Features

Criterion:	OF28. A **Quality Parameter** may be compound (**hasPart**).
Why it is needed:	A Quality Parameter may be organised in arbitrarily complex and structured forms, e.g. a Quality Parameter may be the compound of other specific Quality Parameters.

DLRM Concept Identifier(s):	C162 Quality Parameter
DLRM Relation Identifier(s):	R16 hasPart
Example:	Total Reference Activity can be measured by combining the Digital Reference Questions Received and the Digital Reference Questions received digitally but not answered or responded to by completely digital means.
Suggested Type of Evidence:	Documentary Observational

Criterion:	OF29. A **Quality Parameter** may be evaluated by (**evaluatedBy**) a **Quantitative Measurement**.
Why it is needed:	Quantitative Measurements are based on collecting and interpreting ordinal data.
DLRM Concept Identifier(s):	C160 Quantitative Measurement C162 Quality Parameter
DLRM Relation Identifier(s):	R32 evaluatedBy
Example:	Quantitative Measurement is applied when collecting data and calculating the mean time spent by users in locating content.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	OF30. A **Quality Parameter** may be evaluated by (**evaluatedBy**) a **Qualitative Measurement**.
Why it is needed:	Qualitative Measurements are applied when the collected data are categorical 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 Measurements that could be applied to a DL are direct observation; participant observation; interviews; auditing; case study; collecting written feedback.
DLRM Concept Identifier(s):	C159 Qualitative Measurement C162 Quality Parameter
DLRM Relation Identifier(s):	R32 evaluatedBy
Example:	Methods of Qualitative Measurements that could be applied to a DL are direct observation; participant observation; interviews; auditing; case study; collecting written feedback.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	OF31. A **Quality Parameter** may evaluate (**evaluatedBy**,

	hasQuality) the DL system as a whole (**Generic Quality Parameter**).
Why it is needed:	This is a family of Quality Parameters reflecting the variety of facets that characterise the quality of the 'system' in its entirety, in particular the Digital Library, the Digital Library System and the Digital Library Management System.
DLRM Concept Identifier(s):	C162 Quality Parameter C163 Generic Quality Parameter
DLRM Relation Identifier(s):	R6 hasQuality R32 evaluatedBy
Example:	Performance provides an overall assessment of how well a Resource performs from different points of view, e.g. efficiency, effectiveness, efficacy, etc.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	OF32. A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL Content (**Content Quality Parameter**).
Why it is needed:	A Quality Parameter which reflects the variety of facets that characterise the quality of the Content, in particular Information Objects, in a Digital Library.
DLRM Concept Identifier(s):	C162 Quality Parameter C173 Content Quality Parameter
DLRM Relation Identifier(s):	R6 hasQuality R32 evaluatedBy
Example:	Metadata Evaluation is essential for various processes in the Digital Library, and most specifically in tasks related to access, preservation and operability. 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.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	OF33. A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL Functions (**Functionality Quality Parameter**).
Why it is needed:	A Quality Parameter which reflects the variety of facets that characterise the quality of the Functionality, in particular Functions, of a Digital Library.
DLRM Concept Identifier(s):	C162 Quality Parameter C185 Functionality Quality Parameter

DLRM Relation Identifier(s):	R6 hasQuality R32 evaluatedBy
Example:	Usability records to what extent a given Function makes it easy for an Actor to achieve its goals. Usability concerns many different aspects of a Digital Library, ranging from the user interface, the facility in finding and accessing relevant information, the presentation of search results, to support for facilitating complex or difficult tasks, such as the provision of query-by-example functionalities or browsing and navigation facilities for complex metadata schemas or ontologies.
Suggested Type of Evidence:	Documentary Testimonial

Criterion:	OF34. A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL User (**User Quality Parameter**).
Why it is needed:	A Quality Parameter may assess Actor profiles and User behaviour of a Digital Library.
DLRM Concept Identifier(s):	C162 Quality Parameter C197 User Quality Parameter
DLRM Relation Identifier(s):	R6 hasQuality R32 evaluatedBy
Example:	User Activeness is a User Quality Parameter which reflects to what extent an Actor is active and interacts with a Digital Library. 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.
Suggested Type of Evidence:	Documentary Observational Testimonial

Criterion:	OF35. A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the DL Policies (**Policy Quality Parameter**).
Why it is needed:	A Quality Parameter which reflects the variety of facets that characterise the quality of a set of Policies.
DLRM Concept Identifier(s):	C162 Quality Parameter C200 Policy Quality Parameter
DLRM Relation Identifier(s):	R6 hasQuality R32 evaluatedBy
Example:	Policy Consistency: Digital Rights is a policy regulating rights of use of digital objects. Digital Rights Management Policy governs the

	Functions that implement rights issues in the use of Resources. These two policies have to be consistent in their approach to rights issues.
Suggested Type of Evidence:	Documentary

Criterion:	OF36. A **Quality Parameter** may evaluate (**evaluatedBy**, **hasQuality**) the Architecture of the DLS (**Architecture Quality Parameter**).
Why it is needed:	A Quality Parameter may assess the aspects related to the Digital Library System Architecture. The presence of good administration tools is crucial for configuring and monitoring the functioning of complex and distributed systems.
DLRM Concept Identifier(s):	C162 Quality Parameter C203 Architecture Quality Parameter
DLRM Relation Identifier(s):	R6 hasQuality R32 evaluatedBy
Example:	Load Balancing Performance measures the capacity to spread and distribute work evenly across System Architecture Components. For a DLS on top of a Grid environment, which takes into account several instances of Architectural Components, Load Balancing Performance includes the ability of the system to distribute requests equally among different components of the same type within the system. In particular, this capability consists in selecting Hosting Nodes according to their workload or moving a job from one Hosting Node to another in order to achieve optimal Resource utilisation so that no Resource is over/under-used.
Suggested Type of Evidence:	Documentary Testimonial

IV.6.3.6 Architecture-oriented Optional Features

Criterion:	OF37. Every **Architectural Component**, be it a **Software Architecture Component** or a **System Architecture Component**, may exploit (**use**) one or more other not conflicting (**conflictWith**) **Architectural Components**.
Why it is needed:	The exploitation of functionality offered by other components is a very common practice in software engineering. It reduces the complexity of the problem to be dealt with and favour reusability.
DLRM Concept Identifier(s):	C211 Architectural Component C212 Software Architecture Component C217 System Architecture Component
DLRM Relation Identifier(s):	R19 use

	R20 conflictWith
Example:	In a federated DLS the Search component may rely on the Index Components running on the servers of the participating organisations.
Suggested Type of Evidence:	Documentary

Conclusions

This report has presented version 2.0 of the Reference Model for Digital Libraries. It has been produced by using the DELOS Digital Library Reference Model released by the DELOS Network of Excellence as firm starting point. Because of this, the structure as well as the content of the DELOS Digital Library Reference Model has been inherited by this previous document. It consists of separate parts that illustrate the model from different perspectives and at different levels of abstraction. This structure has been introduced to accommodate the needs of the many different players of the Digital Library universe who are interested in understanding Digital Library 'systems' at different levels of detail.

The model presented is the result of a joint effort initiated by DELOS and continued by DL.org aimed at contributing to the ambitious process of laying foundations for Digital Libraries as a whole. Research on 'digital libraries' addresses many different areas. The lack of any agreement on the foundations for this broad research field has led to a plethora of systems, methodologies and results that are difficult to combine and reuse to produce enhanced outcomes.

The model illustrated draws on the understanding of Digital Library Systems acquired by several European research groups active in the Digital Library field for many years, initially within the DELOS Network of Excellence and now in the Digital Library Community operated by DL.org. In such a context, it is intended as a collective effort by the Digital Library community to agree on common ground. This is meant to be useful not only for current activities but also as a springboard for future work.

The model presented is an abstract framework for understanding significant relationships among the entities of the Digital Library Universe, and for the development of consistent standards or specifications supporting the different elements of this universe. It aims at providing a common semantics that can be used unambiguously across and between different application areas both to explain and organise existing Digital Library 'systems' and to support the evolution of research and development in this area.

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

The framework introduced so far does not aim to cover every specific aspect of the Digital Library universe. Rather, its objective is to provide a core context representing the main aspects of these systems. Other specific aspects can easily be modelled by building on this core part and by introducing more detailed concepts and relationships. We expect that in the future many more focused, fine-grained models, developed by other communities, will be progressively integrated into the present model, thus creating an increasingly richer framework capable of capturing more and more aspects of the Digital Library universe.

Appendix A. Concept Maps in A4 Format

A.1 DL Resource Domain Concept Map

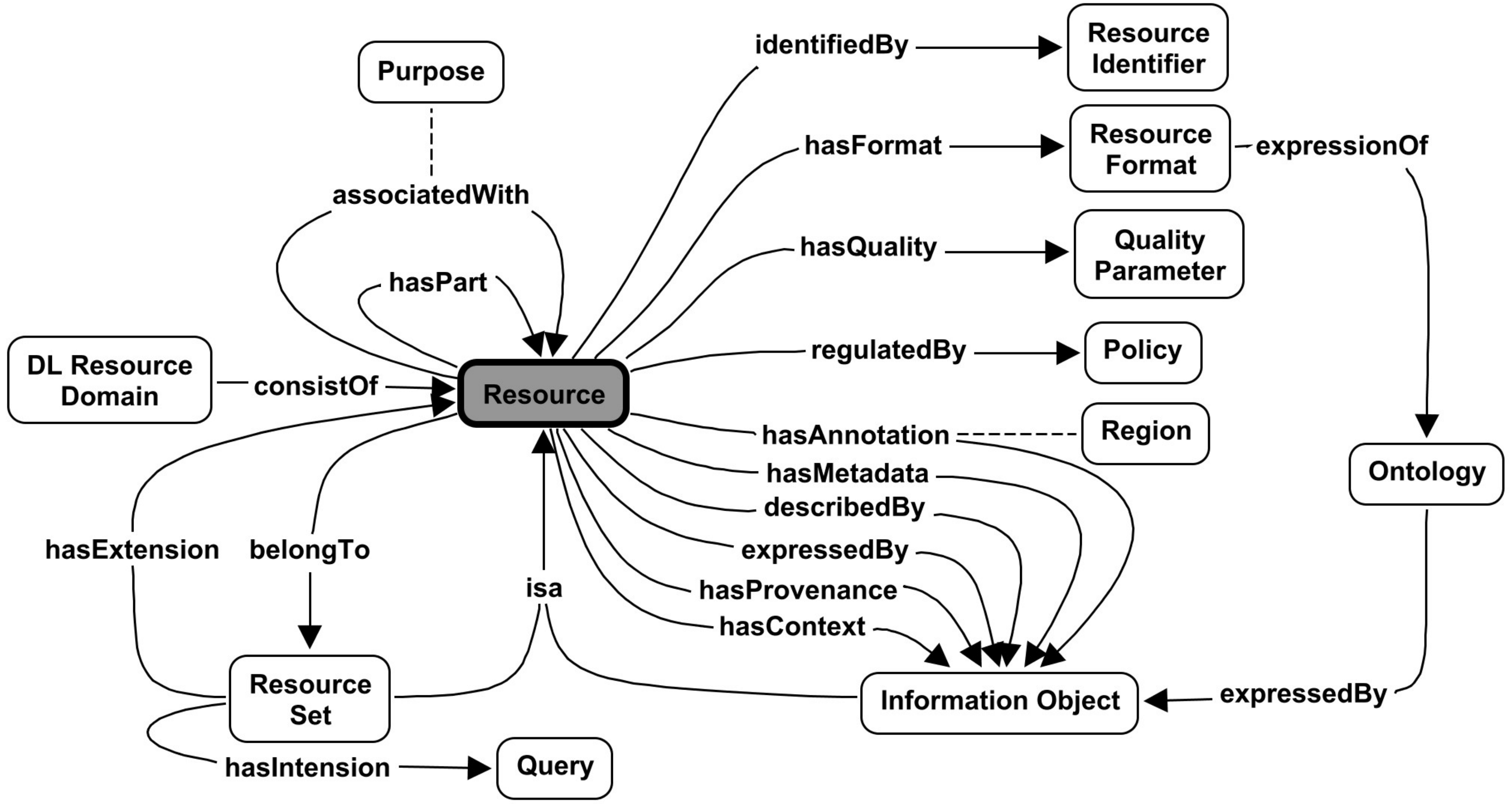

Figure A-1. Resource Domain Concept Map (A4 format)

A.2 Content Domain Concept Map

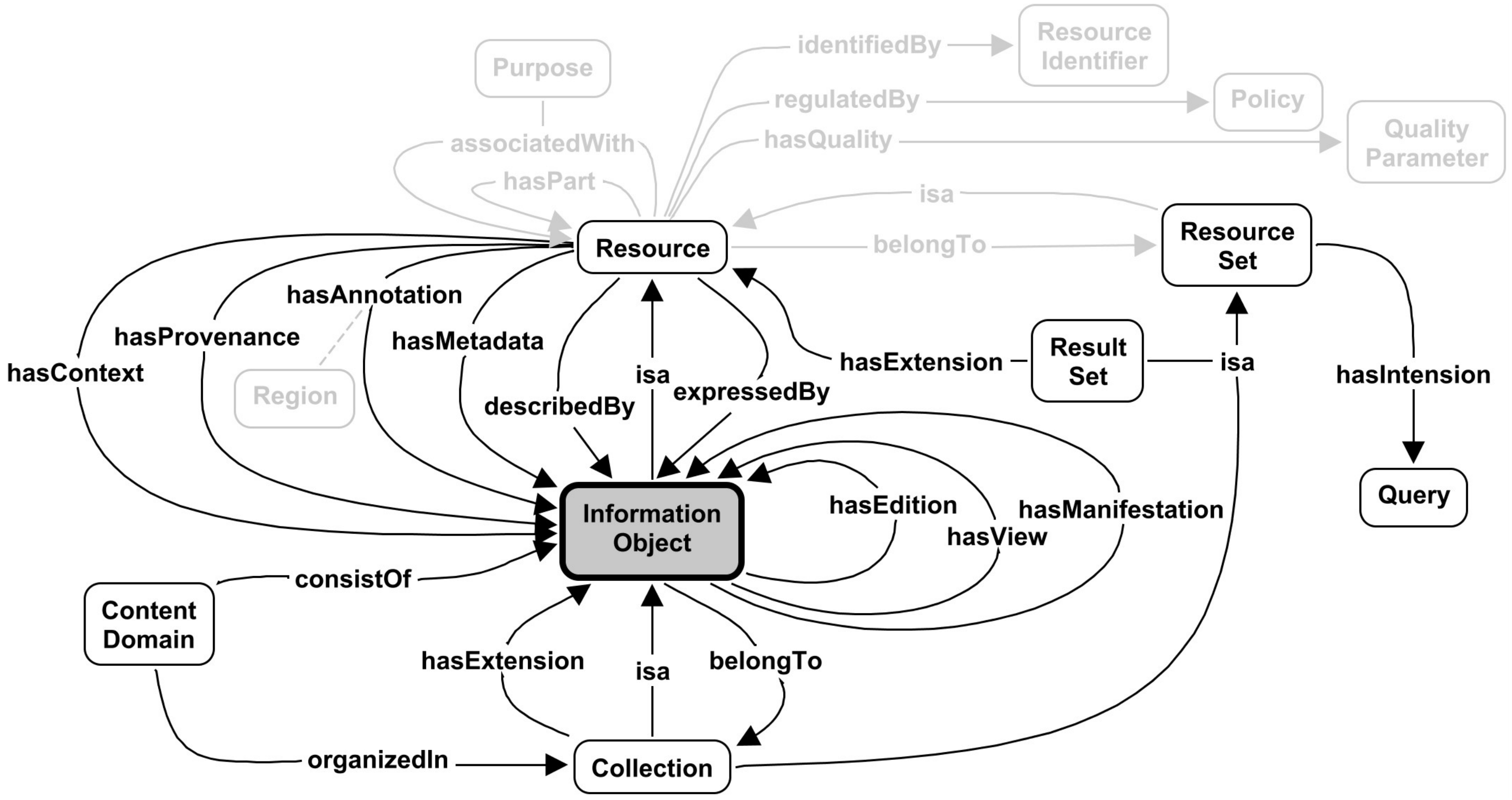

Figure A-2. Content Domain Concept Map (A4 format)

A.3 User Domain Concept Map

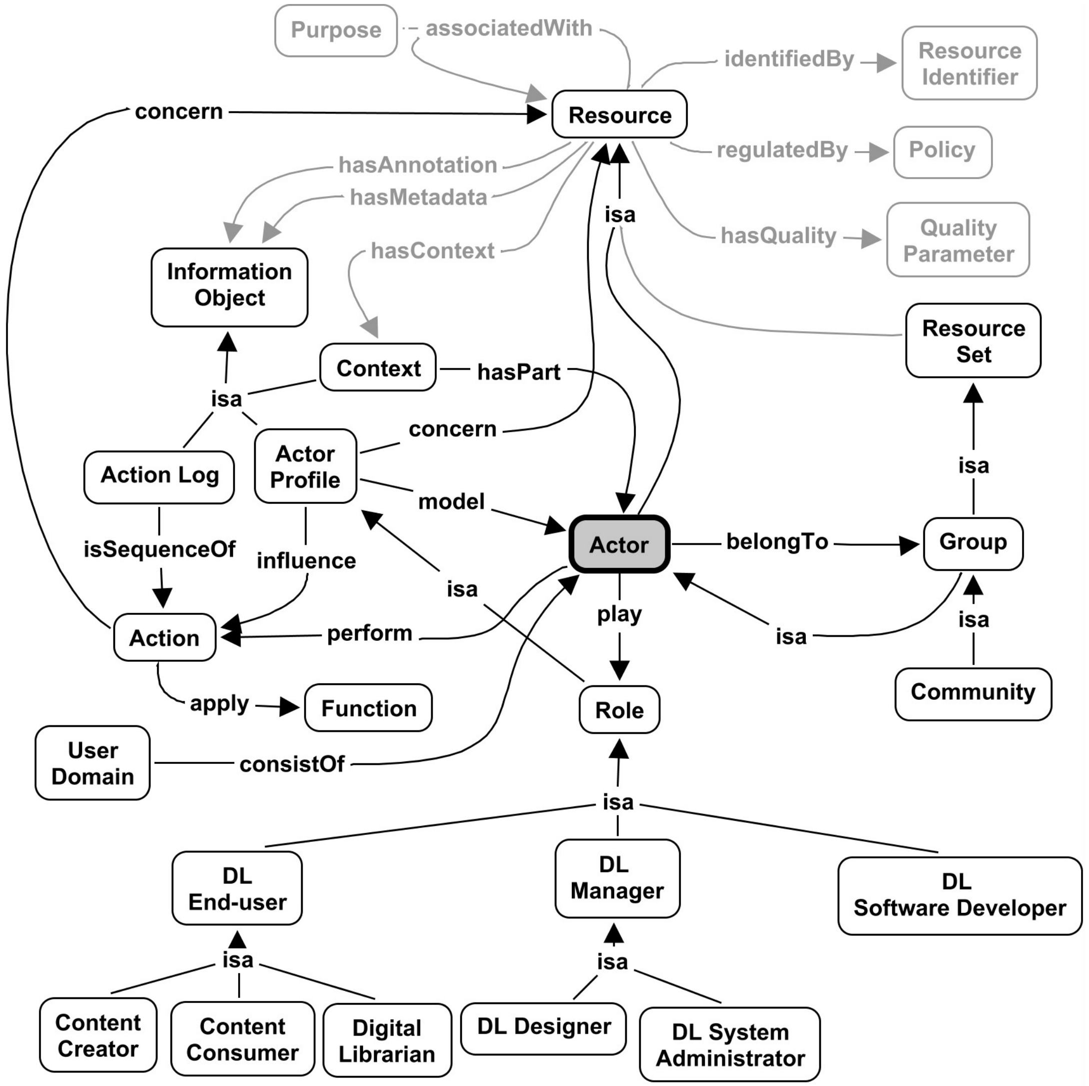

Figure A-3. User Domain Concept Map (A4 format)

A.4 Functionality Domain Concept Map

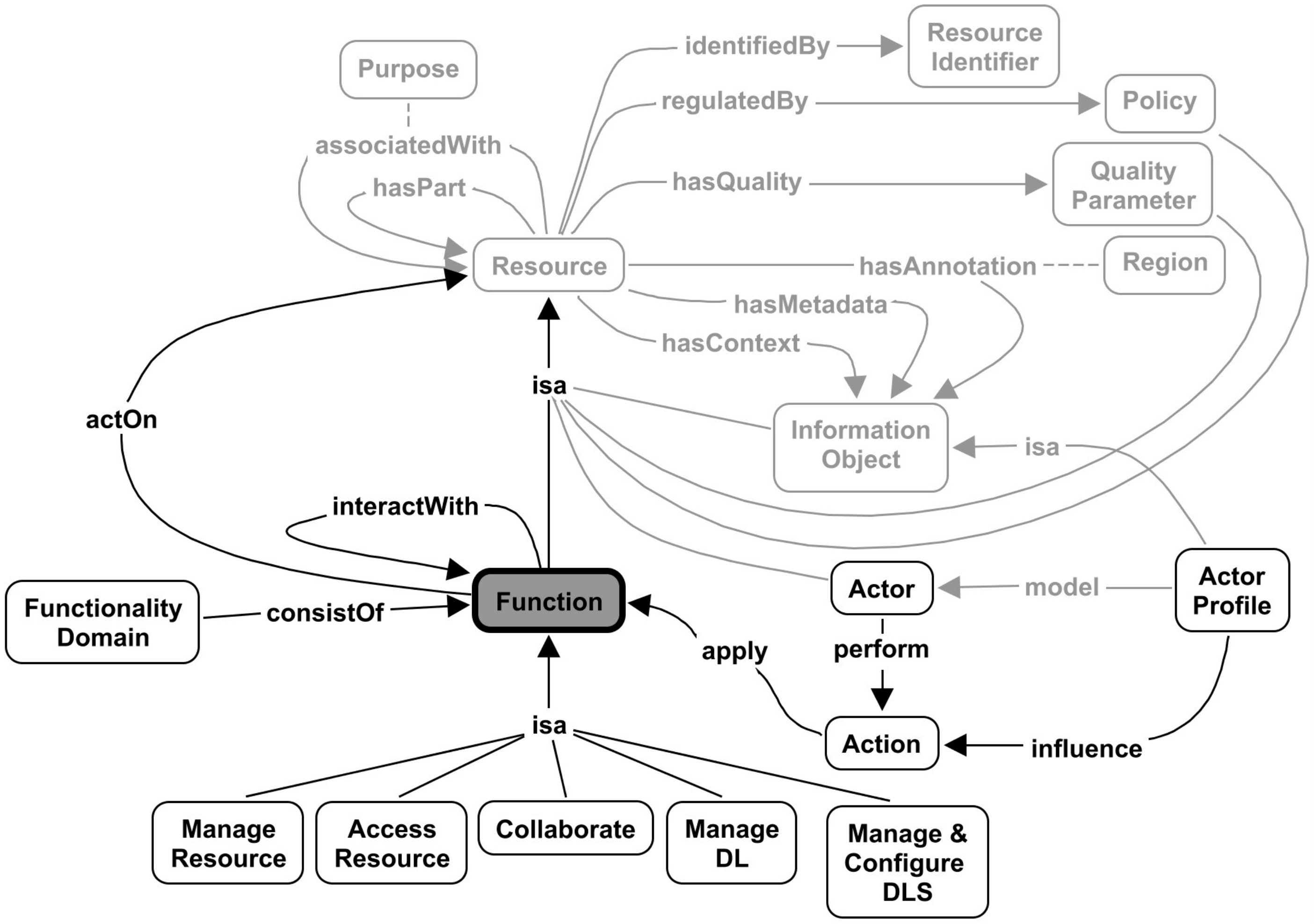

Figure A-4. Functionality Domain Concept Map (A4 format)

A.5 Functionality Domain Concept Map: Access Resource Functions

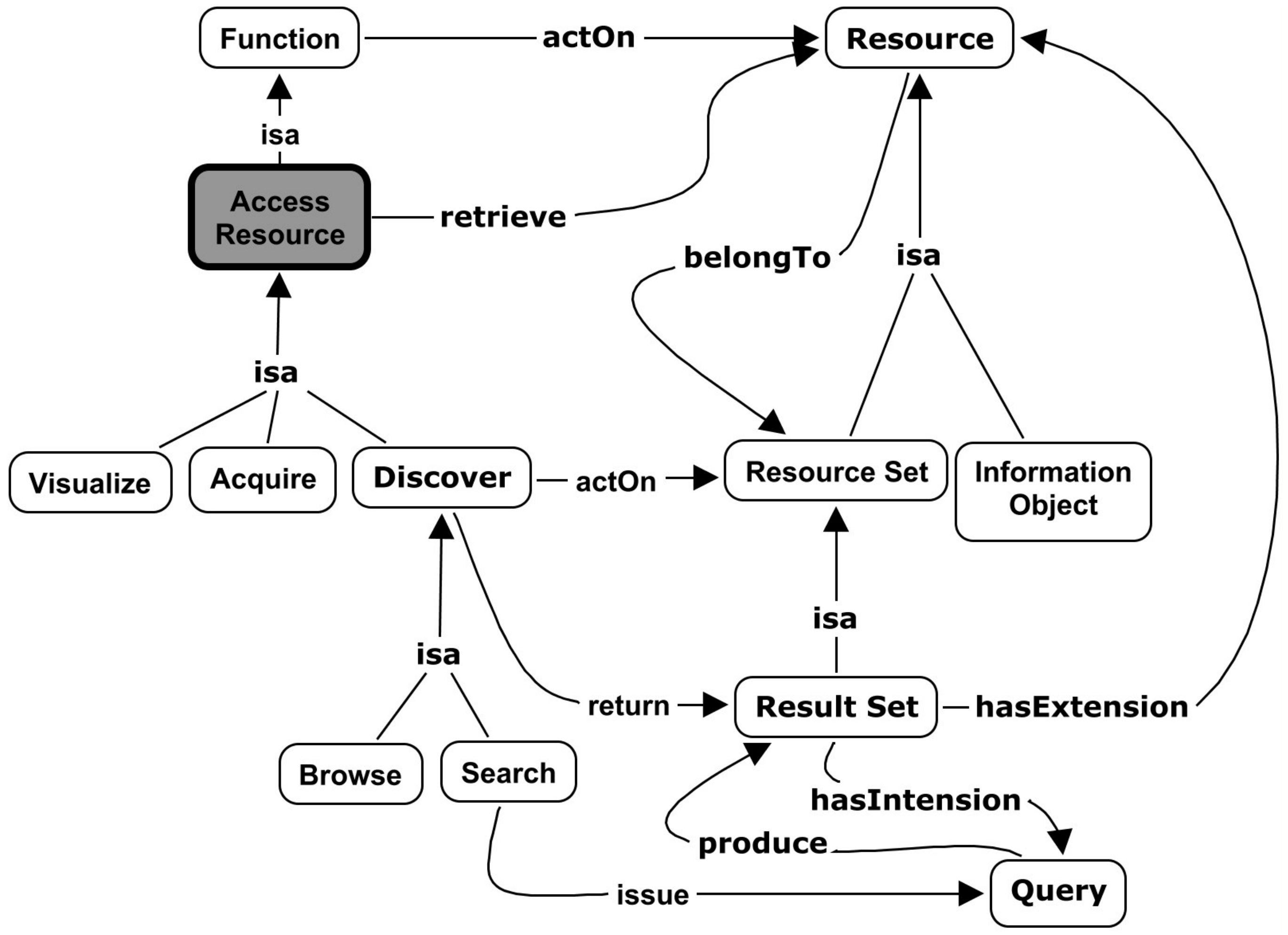

Figure A-5. Functionality Domain Concept Map: Access Resource Functions (A4 format)

A.6 Functionality Domain Concept Map: Manage Resource Functions

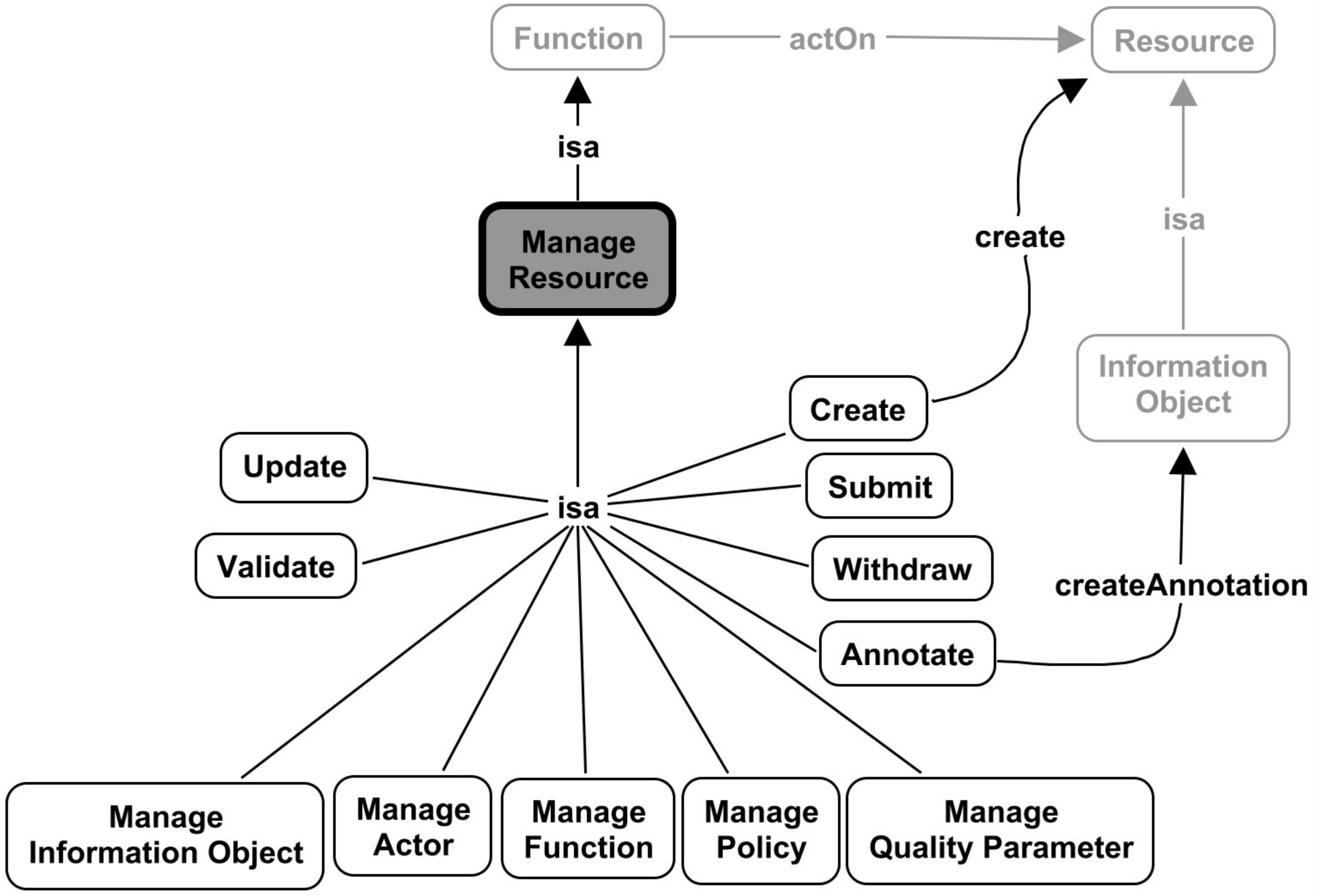

Figure A-6. Functionality Domain Concept Map: General Manage Resource Functions (A4 format)

A.7 Functionality Domain Concept Map: Manage Information Object Functions

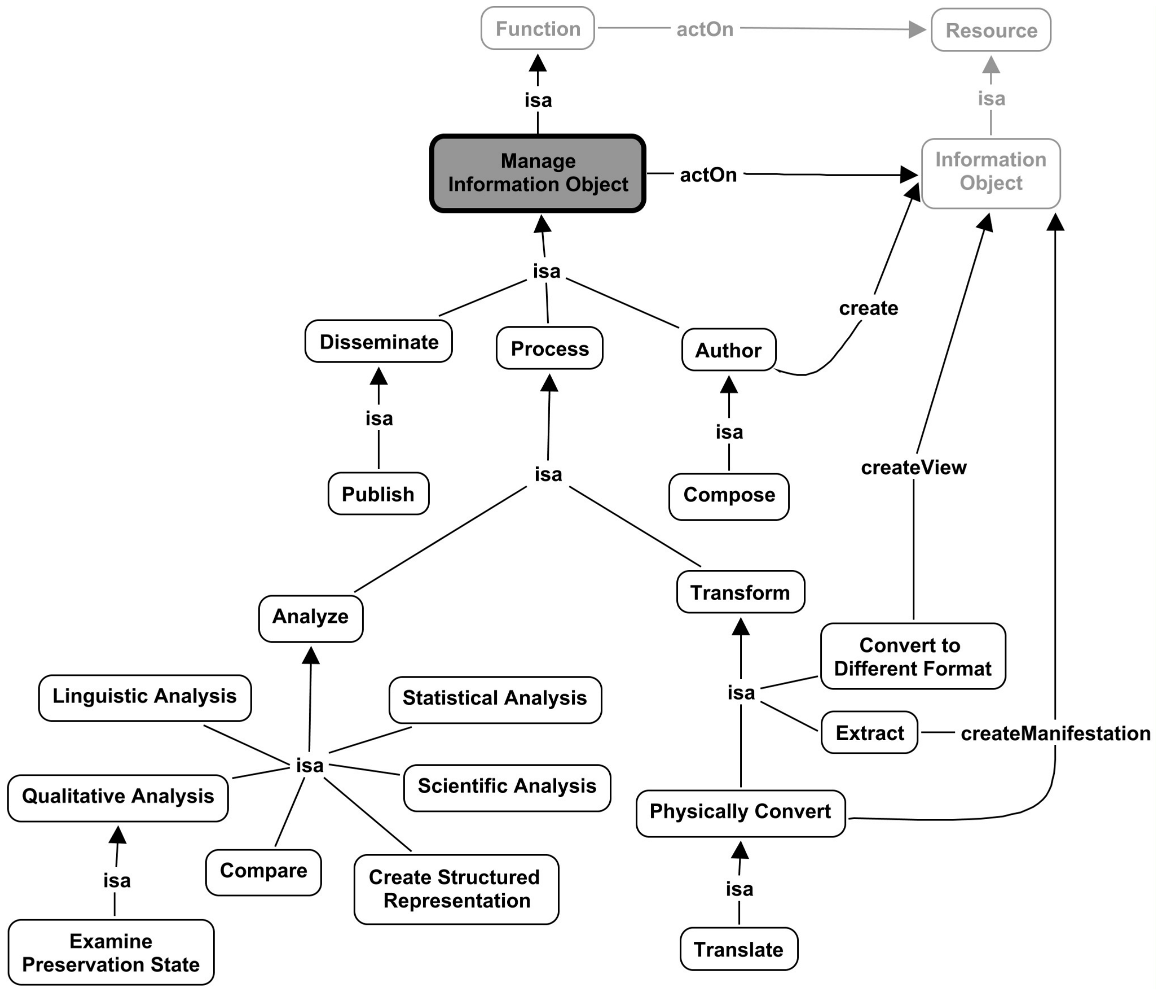

Figure A-7. Functionality Domain Concept Map: Manage Information Object Functions (A4 format)

A.8 Functionality Domain Concept Map: Manage Actor Functions

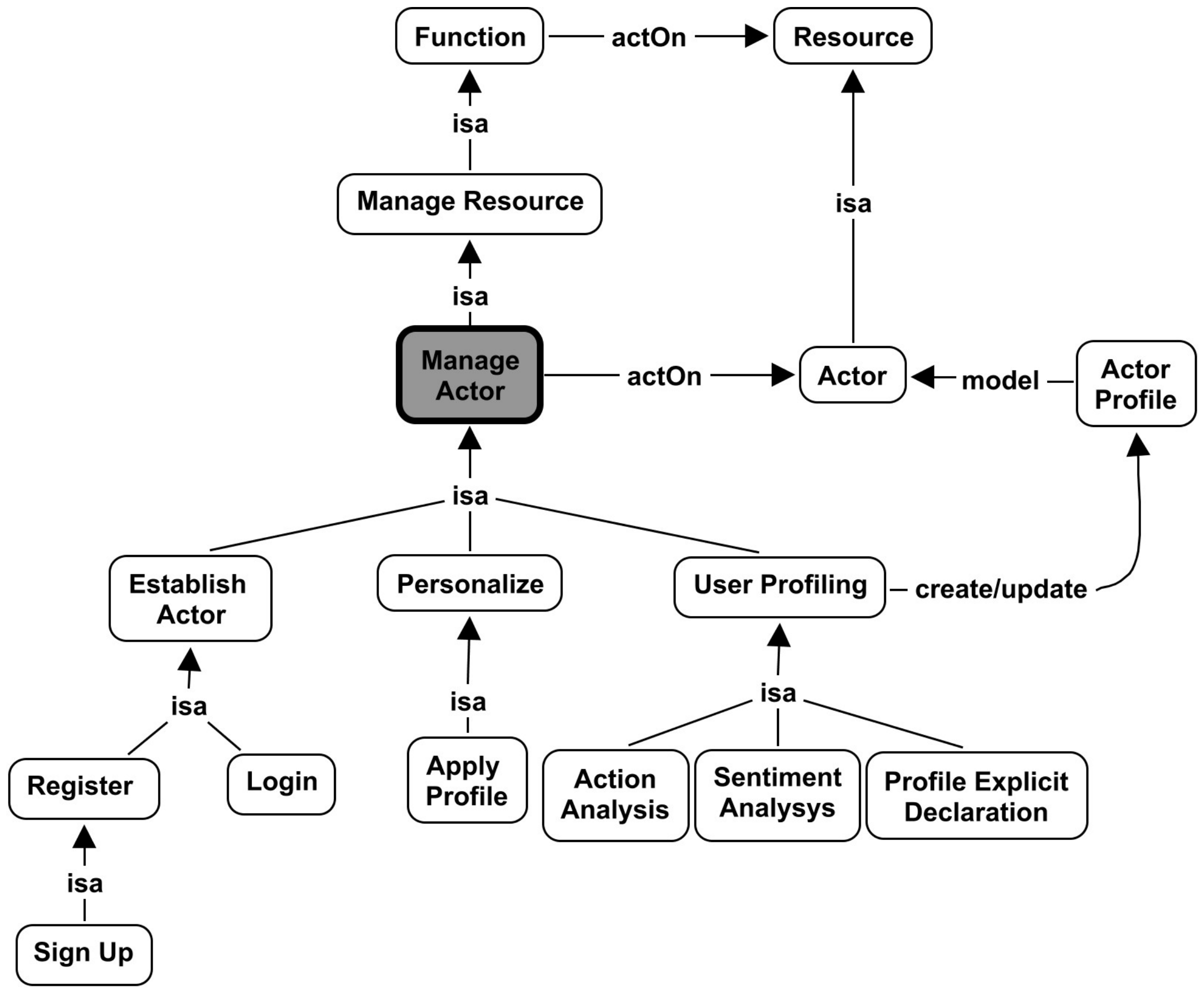

Figure A-8. Functionality Domain Concept Map: Manage Actor Functions (A4 format)

A.9 Functionality Domain Concept Map: Collaborate Functions

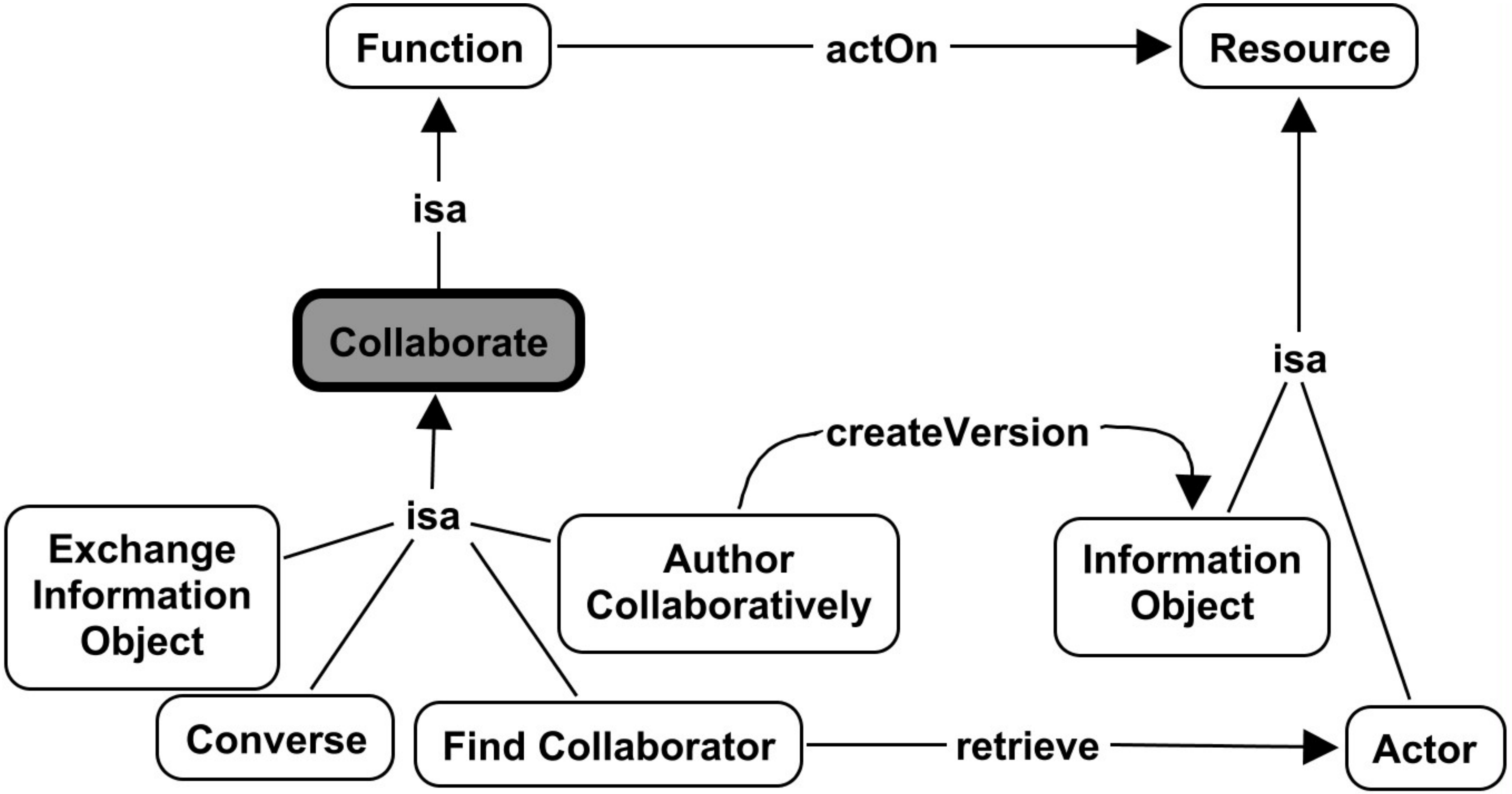

Figure A-9. Functionality Domain Concept Map: Collaborate Functions (A4 format)

A.10 Functionality Domain Concept Map: Manage DL Functions

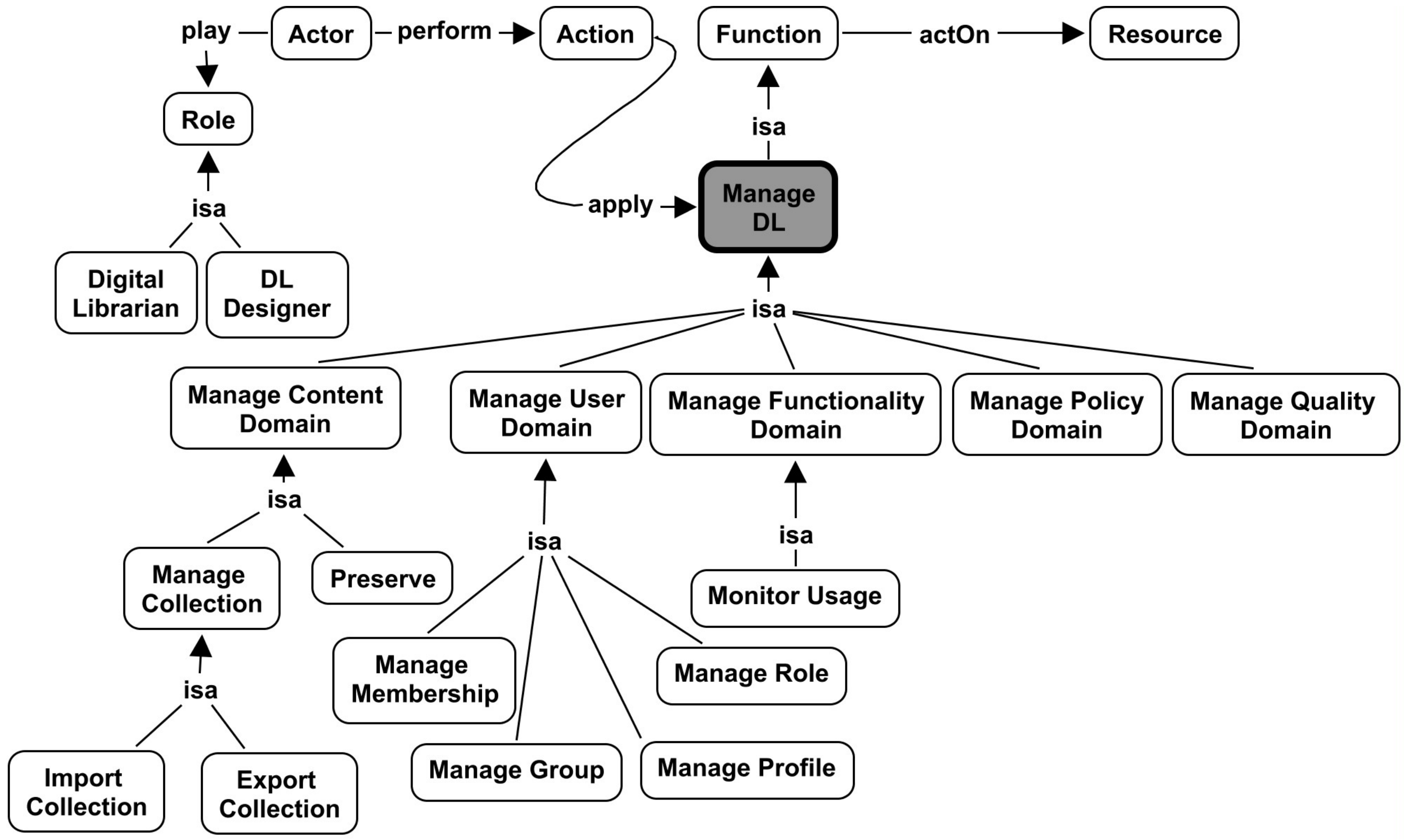

Figure A-10. Functionality Domain Concept Map: Manage DL Functions (A4 format)

A.11 Functionality Domain Concept Map: Manage & Configure DLS Functions

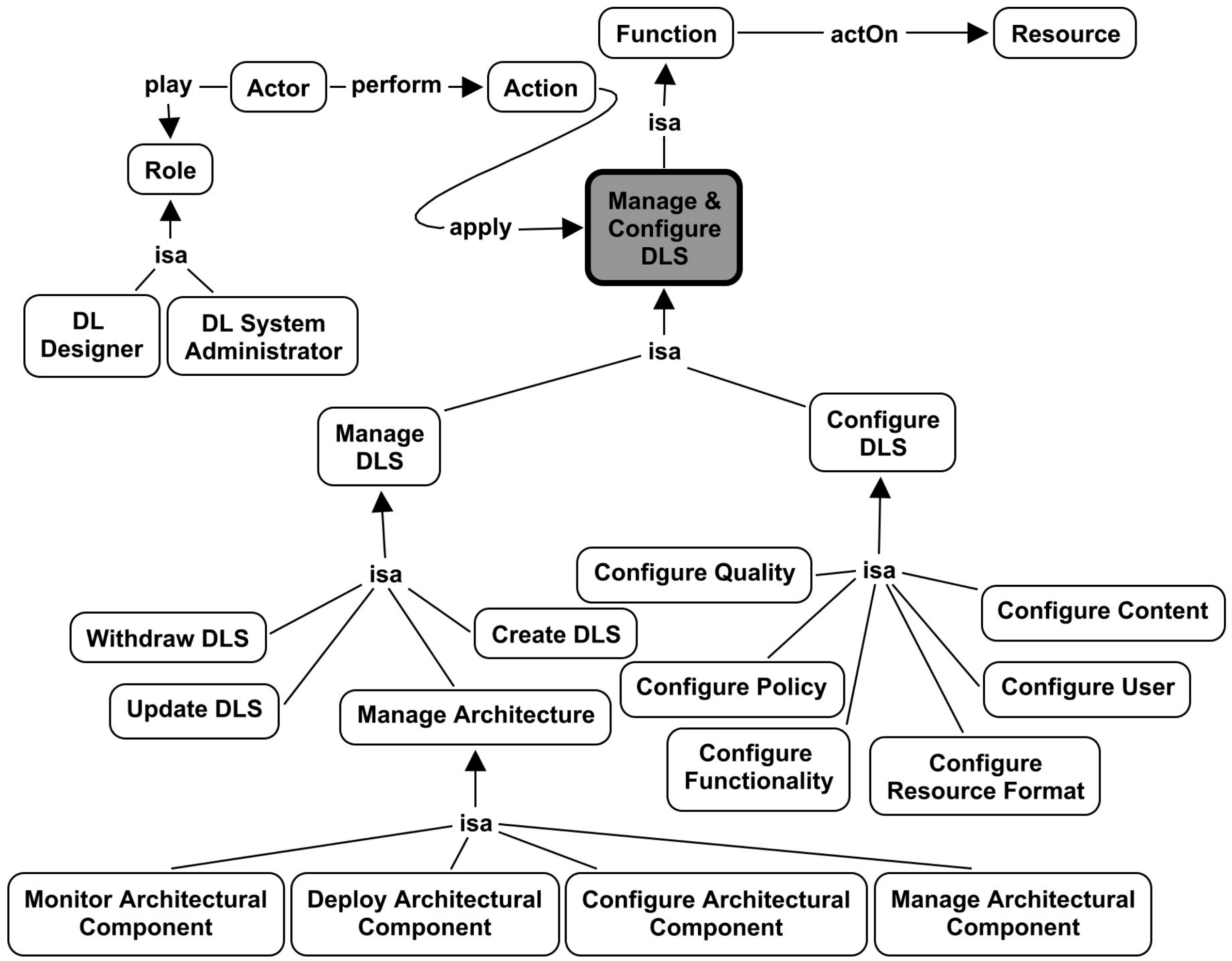

Figure A-11. Functionality Domain Concept Map: Manage & Configure DLS Functions (A4 format)

A.12 Policy Domain Concept Map

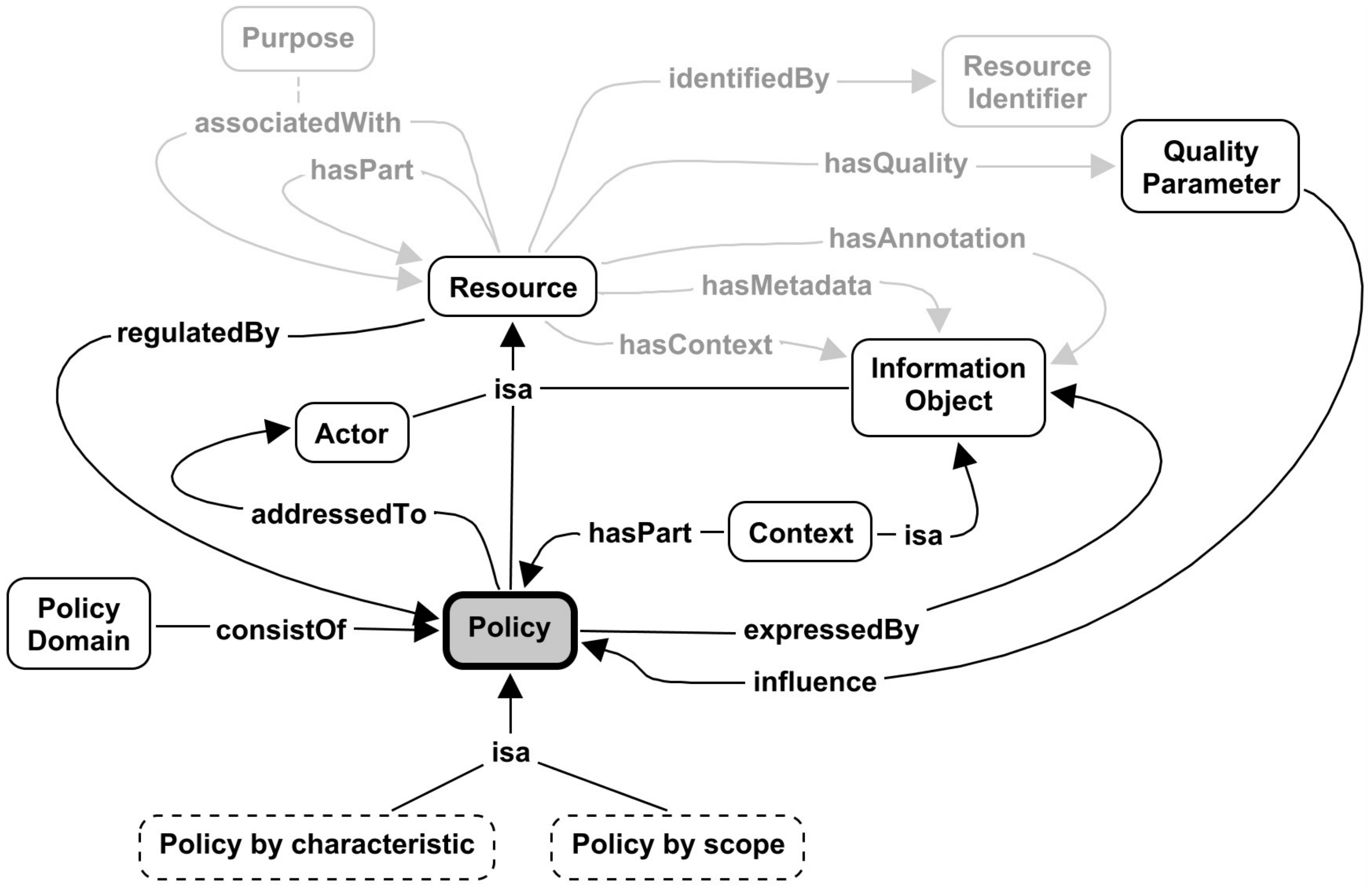

Figure A-12. Policy Domain Concept Map (A4 format)

A.13 Policy Domain Concept Map: Policies' Hierarchy

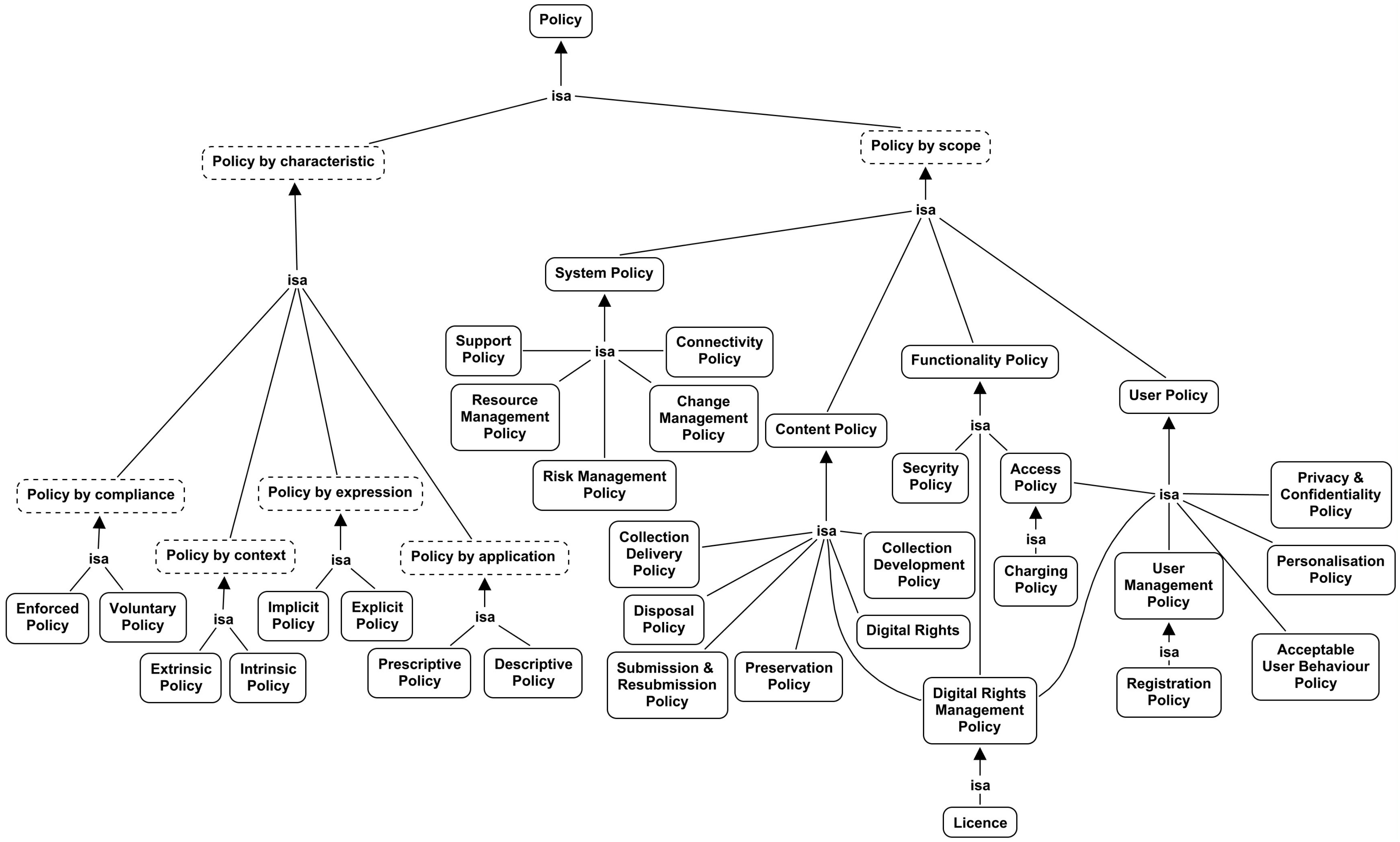

Figure A-13. Policy Domain Concept Map: Policies' Hierarchy (A4 format)

A.14 Quality Domain Concept Map

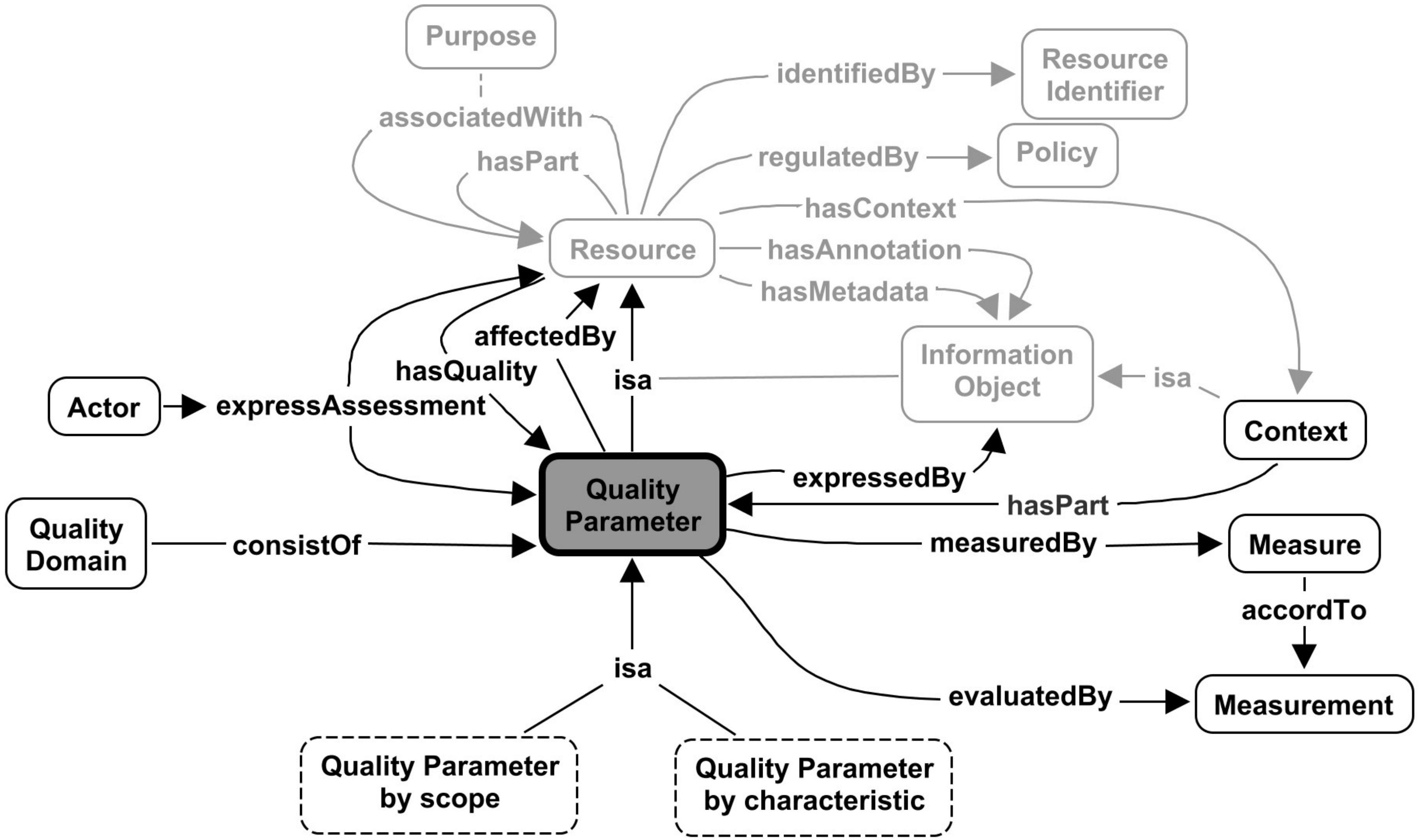

Figure A-14. Quality Domain Concept Map (A4 format)

A.15 Quality Domain Concept Map: Quality Parameters' Hierarchy

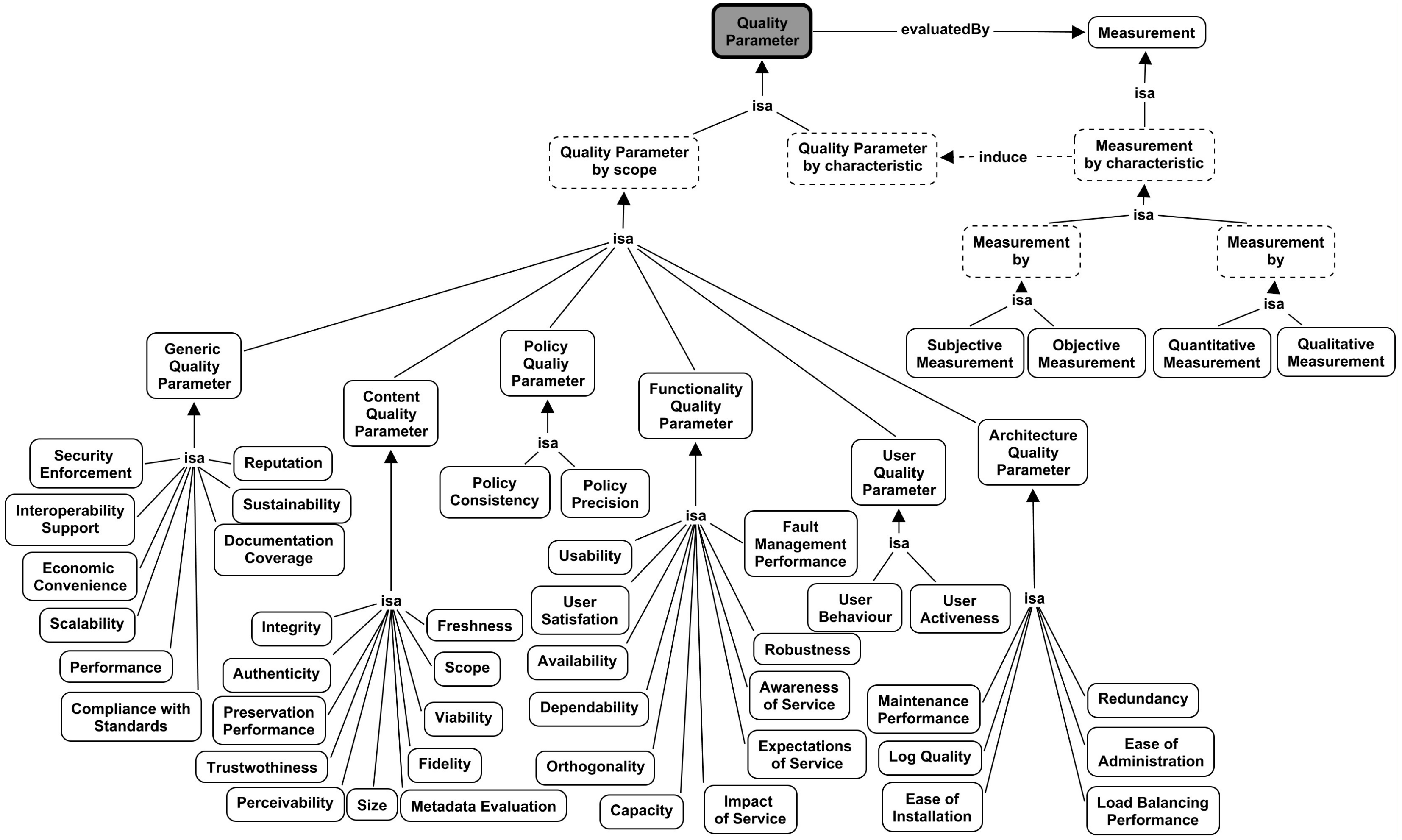

Figure A-15. Quality Parameters' Hierarchy Concept Map (A4 format)

A.16 Architecture Domain Concept Map

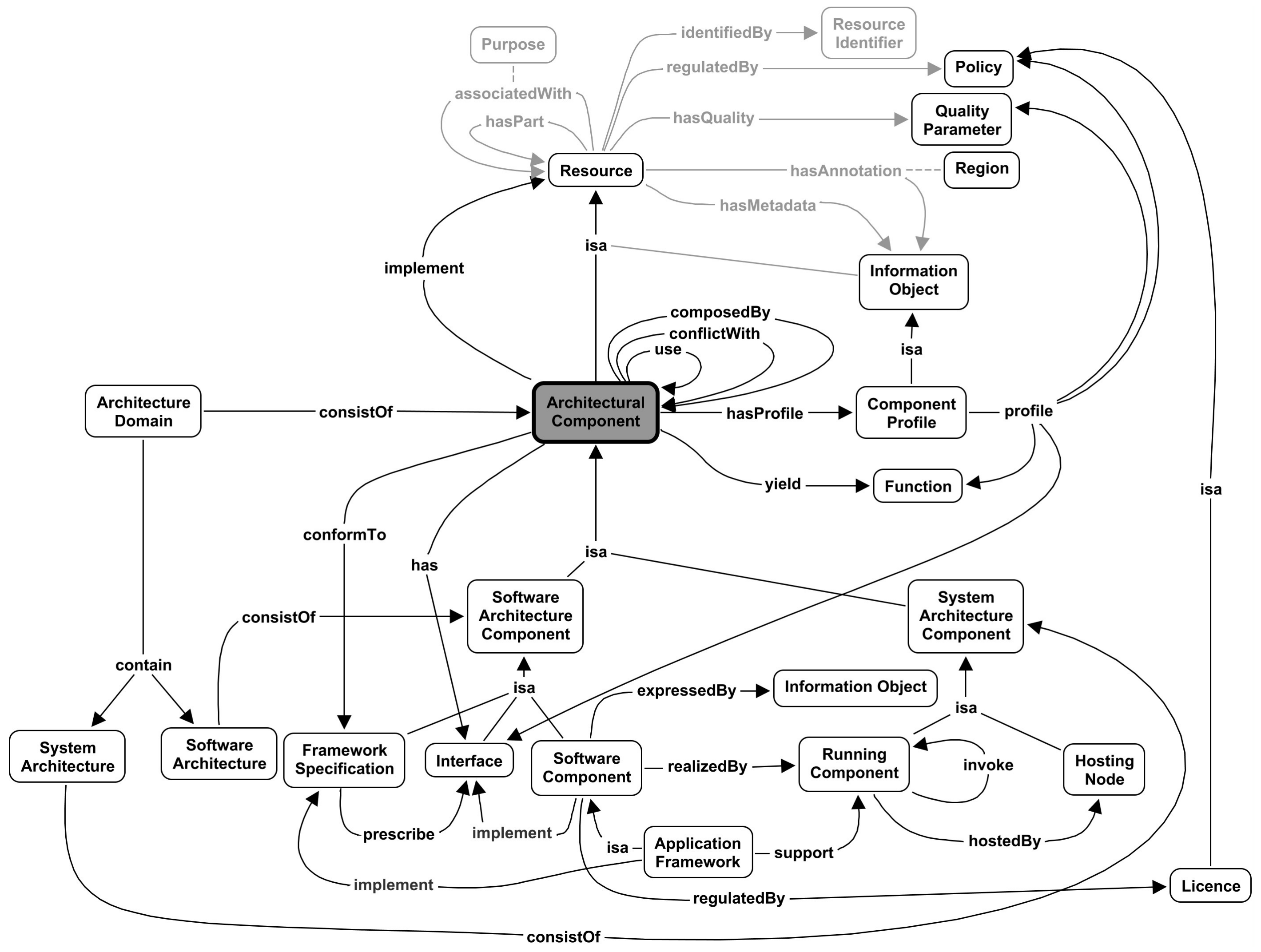

Figure A-16. Architecture Domain Concept Map (A4 format)

Appendix B. Acknowledgements

Many people have contributed to the production of this artefact at different levels and in different periods.

First, a big thank goes to the DELOS Network of Excellence on Digital Libraries Consortium. This Reference Model would have not come into life without the support of this Consortium who gave to the authors the chance to work on this fascinating topic.

Second, the current version of this artefact benefit from the comments and contributions received by the Digital Library community built and operated in the context of the DL.org Coordination Action. This initiative is of paramount importance for guaranteeing the continuation of the path initiated by DELOS Network of Excellence toward the production of a Digital Library Reference Model powerful enough to match the needs of the Digital Library community in the large.

In addition to DELOS and DL.org, various events and people played a key role to produce the current version of this artefact.

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

This document has also benefited from the comments and ideas discussed during three workshops entitled 'Foundations of Digital Libraries'. The three workshops were held, respectively, in conjunction with the ACM IEEE Joint Conference on Digital Libraries (JCDL 2007), the 11th European Conference on Research and Advanced Technologies on Digital Libraries (ECDL 2007) and the 12th European Conference on Research and Advanced Technologies on Digital Libraries (ECDL 2008). As it is not possible to list all the participants individually, thanks goes to all of them collectively and only explicitly to mention the particularly helpful comments received from Edward Fox (Virginia Tech), Geneva Henry (Rice University) and Marianne Backes (Centre Virtuel de la Connaissance sur l'Europe).

Thanks to Perla Innocenti (HATII) for having revised an initial version of the sections on Policy and highlighting the lack of the time dimension.

Thanks also to Professor Stavros Christodoulakis (Technical University of Crete) for actively participating in some of the DELOS Reference Model internal meetings and raising useful comments about the management of multimedia content, and to Professor Hans-Jörg Schek (University of Konstanz), leader

Index of Concepts and Relations

Bibliography

Abiteboul, S., Agrawal, R., Bernestein, P., Carey, M., Ceri, S., Croft, B., et al. (2005). The Lowell Database Research Self-Assessment. *Communications of the ACM , 48* (5), 111-118.

Ackerman, M. (1994). Providing Social Interaction in the Digital Library. *Proceedings of the First Annual Conference on the Theory and Practice of Digital Libraries.*

Agosti, M., & Ferro, N. (2007). A Formal Model of Annotations of Digital Content. *ACM Transactions on Information Systems , 26* (1), 1-57.

Agosti, M., & Ferro, N. (2005). A System Architecture as a Support to a Flexible Annotation Service. *Volume 3664 of Lecture Notes In Computer Science Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures: 6th Thematic Workshop of the EU Network of Excellence DELOS. Revised Selected Papers* (pp. 147-166). Springer-Verlag.

Agosti, M., & Ferro, N. (2005). Annotations as Context for Searching Documents. *Volume 3507 of Lecture Notes In Computer Science, Proceedings 5th International Conference on Conceptions of Library and Information Science – Context: nature, impact and role* (pp. 155-170). Springer-Verlag.

Agosti, M., Albrechtsen, H., Ferro, N., Frommholz, I., Hansen, P., Orio, N., et al. (2005). DiLAS: a Digital Library Annotation Service. *Proceedings of International Workshop on Annotation for Collaboration – Methods, Tools, and Practices (IWAC 2005)* , (pp. 91–101).

Agosti, M., Berretti, S., Brettlecker, G., Del Bimbo, A., Ferro, N., Fuhr, N., et al. (2007). DelosDLMS – The Integrated DELOS Digital Library Management System. *C. Thanos, F. Borri, L. Candela (eds): Digital Libraries: Research and Development, First International DELOS Conference, Pisa, Italy, 13–14 February 2007, Revised Selected Papers. Volume 4877 of Lecture Notes in Computer Science.* Springer.

Agosti, M., Ferro, N., & Orio, N. (2005). Annotating Illuminated Manuscripts: an Effective Tool for Research and Education. *Proceedings of the 5th ACM/IEEE 2005 Joint Conference on Digital Libraries (JCDL 2005)*, (pp. 121-130).

Agosti, M., Ferro, N., Frommholz, I., & Thiel, U. (2004). Annotations in Digital Libraries and Collaboratories: Facets, Models and Usage. *Volume 3232 of Lecture Notes In Computer Science, Research and Advanced Technology for Digital Libraries: Proceedings 8th European Conference, ECDL 2004.* (pp. 244–255). Springer-Verlag.

Alves, M., Viegas Damásio, C., Olmedilla, D., & Nejdl, W. (2006). Distributed tabling algorithm for rule based policy systems. *7th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2006).* IEEE Computer Society. .

Andrews, J., & Law, D. (2004). *Digital Libraries: Policy, Planning and Practice.* Ashgate.

Antoniou, G., Baldoni, M., Bonatti, P., Nejdl, W., & Olmedilla, D. (2007). Rule-based policy specification. *Ting Yu and Sushil Jajodia (eds), Secure Data Management in Decentralized Systems.* Springer.

Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., et al. (2005). *Web Services Business Process Execution Language Version 2.0.* OASIS. OASIS.

Assante, M., Candela, L., Castelli, D., Frosini, L., Lelii, L., Manghi, P., et al. (2008). An Extensible Virtual Digital Libraries Generator. *B. Christensen-Dalsgaard, D. Castelli, B. A. Jurik, and J. Lippincott, editors, 12th European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2008, Aarhus, Denmark, September 14-19, volume 5173 of Lecture Notes in Computer Science* , 122-134.

Athanasopoulos, G., Candela, L., Castelli, D., El Raheb, K., Innocenti, P., Ioannidis, Y., et al. (2011). *Digital Library Technology and Methodology Cookbook.* DL.org.

Athanasopoulos, G., Tsalgatidou, A., & and Pantazoglou, M. (2006). Interoperability among Heterogeneous Services. *Proceedings of the 2006 IEEE International Conference on Services Computing (IEEE SCC 2006)* (pp. 174-181). IEEE Computer Society.

Avancini, H., Candela, L., & Straccia, U. (2007). Recommenders in a Personalized, Collaborative Digital Library Environment. *Journal of Intelligent Information Systems , 28* (3), 253–283.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2002). *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press.

Banwell, L., Ray, K., Coulson, G., Urquhart, C., Lonsdale, R., Armstrong, C., et al. (2004). The JISC User Behaviour Monitoring and Evaluation Framework. *Journal of Documentation , 60* (3), 302-320.

Barlas, C., Cunard, J., & Hill, K. (2003). *Current Developments in the Field of Digital Rights Management.* World Intellectual Property Rights Organization.

Batini, C., & Scannapieco, M. (2006). *Data Quality.* Springer-Verlag.

Beall, J. (2005). Metadata and Data Quality Problems in the Digital Library. *Journal of Digital Information , 6* (3).

Becker, M., & Sewell, P. (2004). Cassandra: Distributed access control policies with tunable expressiveness. *Proceedings of 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)* (pp. 159–168). IEEE Computer Society.

Belkin, N. (1999). Understanding and Supporting Multiple Information Seeking Behaviors in a Single Interface Framework. *Proceedings of the Eighth DELOS Workshop: User Interfaces in Digital Libraries* (pp. 11-18). European Research Consortium for Informatics and Mathematics.

Belkin, N., Oddy, R., & Brooks, H. (1982). ASK for Information Retrieval: part 1. Background and Theory. *Journal of Documentation , 38* (2), 61-71.

Bertino, E., Casarosa, V., Crane, G., Croft, B., Del Bimbo, A., Fellner, D., et al. (2001). *Digital Libraries: Future Directions for a European Research Programme.* San Cassiano: DELOS.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide (2nd Edition).* Addison Wesley Longman.

Borbinha, J. L., Kunze, J., Spinazzè, A., Mutschke, P., Lieder, H.-J., Mabe, M., et al. (2005). Reference models for digital libraries: actors and roles. *International Journal of Digital Libraries , 5* (4), 325-330.

Borgman, C. (1999). What are digital libraries? Competing visions. *Information Processing and Management , 35* (3), 227-243.

Buschmann, F., Meunier, R., Rohnert, H., Sommerla, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture.* John Wiley & Sons.

Bush, V. (1945). As We May Think. *Atlantic Monthly* (176), 101-108.

Candela, L., Castelli, D., & Pagano, P. (2003). A Service for Supporting Virtual Views of Large Heterogeneous Digital Libraries. *Research and Advanced Technology for Digital Libraries, 7th European Conference on Digital Libraries (ECDL 2003)* (pp. 362-373). Springer.

Candela, L., Castelli, D., & Pagano, P. (2011). History, Evolution and Impact of Digital Libraries. In I. Iglezakis, T.-E. Synodinou, & S. Kapidakis, *E-Publishing and Digital Libraries: Legal and Organizational Issues* (pp. 1-30). IGI Global.

Candela, L., Castelli, D., Ioannidis, Y., Koutrika, G., Pagano, P., Ross, S., et al. (2006). *The Digital Library Manifesto.* DELOS.

Castelli, D., & Pagano, P. (2003). A System for Building Expandable Digital Libraries. *Proceedings of ACM/IEEE 2003 Joint Conference on Digital Libraries (JCDL 2003)*, (pp. 335–345).

Chen, P. P.-S. (1976). The Entity-Relationship Model, Toward a Unified View of Data. *ACM Transactions on Database Systems , 1* (1), 9-36.

Dowty, D. R., Wall, R., & Peters, S. (1980). *Introduction to Montague Semantics* (Vol. 11). Springer.

Duranti, L. (2005). The long-term preservation of accurate and authentic digital data: the INTERPARES project. *Data Science Journal , 4*, 106-118.

Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Pal, K., et al. (2004). *Patterns: Service-Oriented Architecture and Web Services.* IBM Redbook.

Ferraiolo, D., Kuhn, D., & Chandramouli, R. (2003). *Role-based Access Control.* Artech House.

Fox, E. A., Akscyn, R. M., Furuta, R. K., & Leggett, J. J. (1995). Digital Libraries. *Communications of the ACM , 38* (4), 23-28.

Fox, E., & Marchionini, G. (1998). Toward a Worldwide Digital Library. *Communications of the ACM , 41* (4), 29-32.

Fuhr, N., Hansen, P., Mabe, M., Micsik, A., & Solvberg, I. (2001). Digital Libraries: A Generic Classification and Evaluation Scheme. *ECDL'01: Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries* (pp. 187-199). London, UK: Springer.

Fuhr, N., Tsakonas, G., Aalberg, T., Agosti, M., Hansen, P., Kapidakis, S., et al. (2006). Evaluation of Digital Libraries. *International Journal of Digital Libraries* .

Garlan, D., & Shaw, M. (1993). An Introduction to Software Architecture. *Advances in Software Engineering and Knowledge Engineering. 2*, pp. 1-39. Singapore: World Scientific Publishing Company.

Gil, Y., Cheney, J., Groth, P., Hartig, O., Miles, S., Moreau, L., et al. (2010). *Provenance XG Final Report.* W3C.

Gladney, H. (2006). Principles for Digital Preservation. *Communication of the ACM , 49* (2), 111-116.

Gonçalves, M. (2004). *Streams, Structures, Spaces, Scenarios, and Societies (5S): A Formal Model for Digital Library Framework and Its Applications.* Virginia Polytechnic Institute and State University.

Gonçalves, M., & Fox, E. (2002). 5SL - A Language for Declarative Specification and Generation of Digital Libraries. *Proceedings of the 2nd ACM/IEEE Joint Conference on Digital Libraries (JCDL'02).*

Gonçalves, M., Fox, E., Watson, L., & Kipp, N. (2004). Streams, Structures, Spaces, Scenarios, Societies (5S): A Formal Model for Digital Libraries. *ACM Transactions on Information Systems (TOIS) , 22* (2), 270 –312.

Gonçalves, M., Moreira, B., Fox, E., & Watson, L. (2007). What is a good digital library? – A quality model for digital libraries. *Information Processing & Management , 43* (5), 1416-1437.

Griffin, S., Peters, C., & Thanos, C. (2005). Towards the new-generation digital libraries: recommendations of the NSF/EU-DELOS working groups. *International Journal of Digital Libraries , 5* (4), 253 –254.

Guarino, N. (1998). Formal Ontology and Information Systems. *Proceedings of the 1st International Conference on Formal Ontology in Information Systems (FOIS'98)*, (pp. 3-15). Trento.

Harvey, R. (2005). *Preserving Digital Materials.* K.G. Saur.

Heery, R., & Patel, M. (2000). Application profiles: mixing and matching metadata schemas. *Ariadne* (25).

IEEE. (2000). *IEEE Recommended Practice for Architectural Description for Software-Intensive Systems.* IEEE.

Ioannidis, Y. (2005). Digital libraries at a crossroads. *International Journal of Digital Libraries , 5* (4), 255-265.

Ioannidis, Y., Maier, D., Abiteboul, S., Buneman, P., Davidson, S., Fox, E. A., et al. (2005). Digital library information-technology infrastructures. *International Journal of Digital Libraries , 5* (4), 266-274.

Ioannidis, Y., Milano, D., & Schek, H. a. (2008). DelosDLMS. *International Journal on Digital Libraries , 9* (2), 101-114.

ISO 21127:2006. (2006). *Information and documentation -- A reference ontology for the interchange of cultural heritage information.* International Standards for Business, Government and Society.

ISO14721:2003. (2003). *Space data and information transfer systems - Open archival information system - Reference model.* International Standards for Business, Government and Society.

Kuny, T., & Cleveland, G. (1996). The Digital Library: Myths and Challenges. *Proceedings 62nd IFLA General Conference.*

Lagoze, C., & Van de Sompel, H. (2001). The open archives initiative: building a low-barrier interoperability framework. *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries, Roanoke, Virginia, United States, JCDL '01*, (pp. 54–62).

Lagoze, C., Payette, S., Shin, E., & Wilper, C. (2006). Fedora: an architecture for complex objects and their relationships. *International Journal of Digital Libraries , 6* (2), 124-138.

Lavoie, B., Henry, G., & Dempsey, L. (2006). A Service Framework for Libraries. *D-Lib Magazine , 12* (7/8).

Lesk, M. (1999). Expanding Digital Library Research: Media, Genre, Place and Subjects. *Proceedings of the International Symposium on Digital Libraries1999, ISDL'99*, (pp. 51-57). Tsukuba, Ibaraki, Japan.

Licklider, J. (1965). *Libraries of the Future.* Cambridge: The MIT Press.

MacKenzie, M., Laskey, K., McCabe, F., Brown, P., & Metz, R. (2006). *Reference Model for Service Oriented Architecture.* OASIS.

Madison, O., Byrum, J., Jouguelet, S., McGarry, D., Williamson, N., Witt, et al. (2009). *Functional Requirements for Bibliographic Records.* International Federation of Library Associations and Institutions. International Federation of Library Associations and Institutions.

Navarro, G., & Baeza-Yates, R. (1997). Proximal nodes: a model to query document databases by content and structure. *ACM Transactions on Information Systems (TOIS) , 15* (4), 400-435.

Novak, J. D., & Cañas, A. J. (2008). *The Theory Underlying Concept Maps and How to Construct and Use Them.* Institute for Human and Machine Cognition. Florida Institute for Human and Machine Cognition.

Novak, J., & Gowin, D. (1984). *Learning How to Learn.* Cambridge University Press.

Oberle, D., Lamparter, S., Grimm, S., Vrandecic, D., Staab, S., & Gangemi, A. (2006). Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems. *Journal of Applied Ontology* .

Paepcke, A., Chang, C., Winograd, T., & García-Molina, H. (1998). Interoperability for digital libraries worldwide. *Communications of the ACM , 41* (4), 33-42.

Ross, S., & Hedstrom, M. (2005). Preservation research and sustainable digital libraries. *International Journal of Digital Libraries , 5* (4), 317-324.

Salton, G., & McGill, M. (1983). *Introduction to Modern Information Retrieval.* McGraw-Hill.

Saracevic, T. (2000). Digital Library Evaluation: Toward Evolution of Concepts. *Library Trends , 49* (2), 350-369.

Smeaton, A., & Callan, J. (2005). Personalisation and recommender systems in digital libraries. *International Journal of Digital Libraries , 5* (4), 299-308.

Snowdon, D., Churchill, E., & Frecon, E. (2004). *Inhabited Information Spaces - Living with your Data.* London: Springer.

Soergel, D. (2002). A Framework for Digital Library Research. *DLib Magazine , 8* (12).

Sturges, P., Davies, E., Dearnley, J., Iliffe, U., Oppenheaim, C., & Hardy, R. (2003). User privacy in the digital library environment: an investigation of policies and preparedness. *Library Management , 24* (1/2), 44-50.

Sturges, P., Teng, V., & Iliffe, U. (2001). User privacy in the digital library environment: a matter of concern for information professionals. *Library Management , 22* (8/9), 364-370.

Tansley, R., Bass, M., & Smith, M. (2003). DSpace as an Open Archival Information System: Current Status and Future Directions. *Research and Advanced Technology for Digital Libraries, 7th European Conference on Digital Libraries (ECDL 2003).*

Tedd, L. L. (2005). *Digital libraries: principles and practice in a global environment.* K.G. Saur.

Tsakonas, G., Kapidakis, S., & Papatheodorou, C. (2004). *Evaluation of User Interaction in Digital Libraries.* In: Agosti, M.; Fuhr, N. (eds): Notes of the DELOS WP7 Workshop on the Evaluation of Digital Libraries, Padua, Italy.

Van de Sompel, H., Lagoze, C., Bekaert, J., Liu, X., Payette, S., & Warner, S. (2006). An Interoperable Fabric for Scholarly Value Chains. *D-Lib Magazine , 12* (10).

Van der Sluijs, K., & Houben, G.-J. (2006). A Generic Component for Exchanging User Models between Web-based Systems. *International Journal of Continuing Engineering Education and Life-Long Learning , 16* (1/2), 64-76.

Van der Sluijs, K., & Houben, G.-J. (2005). Towards a generic user model component. *Proceedings of the Workshop on Decentralized, Agent Based and Special Approaches to User Modelling, in International Conference on User Modelling*, (pp. 43-52).

Walter, V. (1997). Becoming digital: Policy implications for library and youth services. *Library Trends , 45* (4), 585-601.

Weiser, P. (2003). The Internet, Innovation, and Intellectual Property Policy. *Columbia Law Review , 103*, 534–613.

Wiederhold, G. (1992). Mediators in the Architecture of Future Information Systems. *Computer , 25* (3), 38-49.

Wiederhold, G., Wegner, P., & Ceri, S. (1992). Toward Megaprogramming. *Communication of the ACM , 38* (11), 89-99.

Wilson, T. (1999). Exploring Models of Information Behaviour: the Uncertainty Project. *Information Processing and Management , 35* (6), 839–849.

Wilson, T. (1997). Information Behaviour: An Interdisciplinary Perspective. *Information Processing and Management , 33* (4), 551–572.

Witten, I., Bainbridge, D., & Boddie, S. (2001). Power to the People: End-user Building of Digital Library Collections. *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries* (pp. 94–103). ACM Press.

Wormell, I. (. (1990). *Information quality: Definitions and dimensions.* Taylor Graham.

Zachman, J. A. (1987). A Framework For Information Systems Architecture. *IBM Systems Journal , 26* (3), 276-292.