http://www.dlorg.eu/uploads/DL%20Reference%20Models/The%20Digital%20Library%20Reference%20Model_v1.0.pdf

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: The Digital Library Reference Model

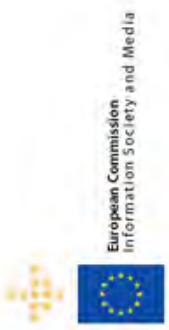
Submission Due Date: September 2009

Actual Submission Date: January 2010

Work Package: WP3

Responsible Partner: CNR

Deliverable Status: Final



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:	D3.2a
Deliverable title:	The Digital Library Reference Model
Editor(s):	L. Candela
Author(s):	G. Athanasopoulos, L. Candela, D. Castelli, 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: *(including cover)*	237
Keywords:	Reference Model, Content Domain Model, User Domain Model, Functionality Domain Model, Policy Domain Model, Quality Domain Model, Architecture Domain Model;

Table of Contents

List of Figures

Summary

This document represents the first 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. It is organised in three parts each forming a self contained artefact, i.e. the *Digital Library Manifesto*, the *Digital Library Reference Model in a Nutshell*, and the *Digital Library Reference Model Concepts and Relations*. The Digital Library Manifesto declares the intentions, motives, overall plans and views of the a long term initiative leading to the production of a foundational theory for Digital Libraries; it also 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. Finally, 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.

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 artefact 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 artefact identifies the set of concepts and relationships that characterise the essence of the Digital Library universe. This model should be considered as a roadmap allowing the various players involved in the Digital Library domain to follow the same route and share a common understanding when dealing with the entities of such a universe.

This document presents the 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 broad coverage of the Digital Library universe, its evolving nature, and the lack of any previous agreement on its foundations, the Reference Model is by necessity a dynamic framework, as is also this document. Continuous evolutions of the document are envisaged in order to obtain a number of well-formed and consolidated definitions, shared by the Digital Library community.

The volume is organised in three parts, each potentially constituting a document in its own. Each of the three parts describes the Digital Library universe from a different perspective that is driven by a trade-off between abstraction and concretisation. Thus each part is equally important in capturing the nature of 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 DELOS Digital Library Reference Model in a Nutshell' treats these notions in more detail by introducing the main concepts and relationships related to each of the aspects captured by the previous one; finally, 'PART III The DELOS Digital Library Reference Model Concepts and Relations' describes each of the identified concepts and relations in detail by explaining their rationale as well as presenting examples of their instantiation in concrete scenarios.

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

Section I.1 introduces 'PART I The Digital Library Manifesto' by providing the driving force pervading the whole activity. Section I.2 presents the relationships between 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. content, user, functionality, quality, policy and architecture. Section I.4 introduces the main roles actors may play within digital libraries, i.e. end-user, designer, administrator and application developer. Section I.5 describes the reference frameworks needed to clarify the DL universe at different levels of abstraction, i.e. the Digital Library Reference

Model and the Digital Library Reference Architecture. Section I.6 records concluding remarks on The Digital Library Manifesto.

Section II.1 introduces 'PART II The DELOS Digital Library Reference Model in a Nutshell' by summarising the content of the Manifesto and setting the basis for reading and using the rest of this part. Section II.2 presents the constituent domains by briefly describing their rationale and providing 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, it describes the problem by pointing out the instruments the Reference Model makes available for dealing with it. Section II.4 discusses related works. In particular, this section highlights the similarities and differences between this Reference Model and similar initiatives like the 5S Framework and the CIDOC Conceptual Reference Model. Section II.5 records concluding remarks on the Digital Library Reference Model as presented in PART II.

Section III.1 introduces 'PART III The DELOS Digital Library Reference Model Concepts and Relations' by highlighting the role of this part. Section III.2 presents the hierarchy of Concepts constituting the Reference Model. Section III.3 provides a definition for each of the 218 Concepts currently constituting the model. Each definition is complemented by the list of relations connecting the concept to the other concepts, the rationale for including this concept in the model, and examples of concrete instances of the concept in real-life scenarios. Section III.4 presents the hierarchy of the identified Relations. Section III.5 provides a definition for each of the 52 Relations currently constituting the model. Each definition is complemented by the rationale for including it in the model and some examples of concrete instances in real-life scenarios.

A concluding section summarised the entire artefact content and complete it.

The document comprises also four appendixes. Appendix A provides the concept maps of the Reference Model in A4 format to improve their readability. Appendix B provides the concept maps of the Reference Model expressed in terms of UML Class Diagrams to both demonstrate the equivalence of the Concept Maps and UML from the perspective of this model and provide readers accustomed to using UML with a representation that is familiar to them. Appendix C describes the main open issues affecting this artefact and some comments describing the discussion about them. Appendix D lists others contributors of this artefact along its lifetime and acknowledges them.

systems, e.g. a private *Information Object* in a DL is managed by a DLS service instructed to deliver that object only to the *Actor* that is its owner.[14]

II.2.6 Quality Domain

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

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

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

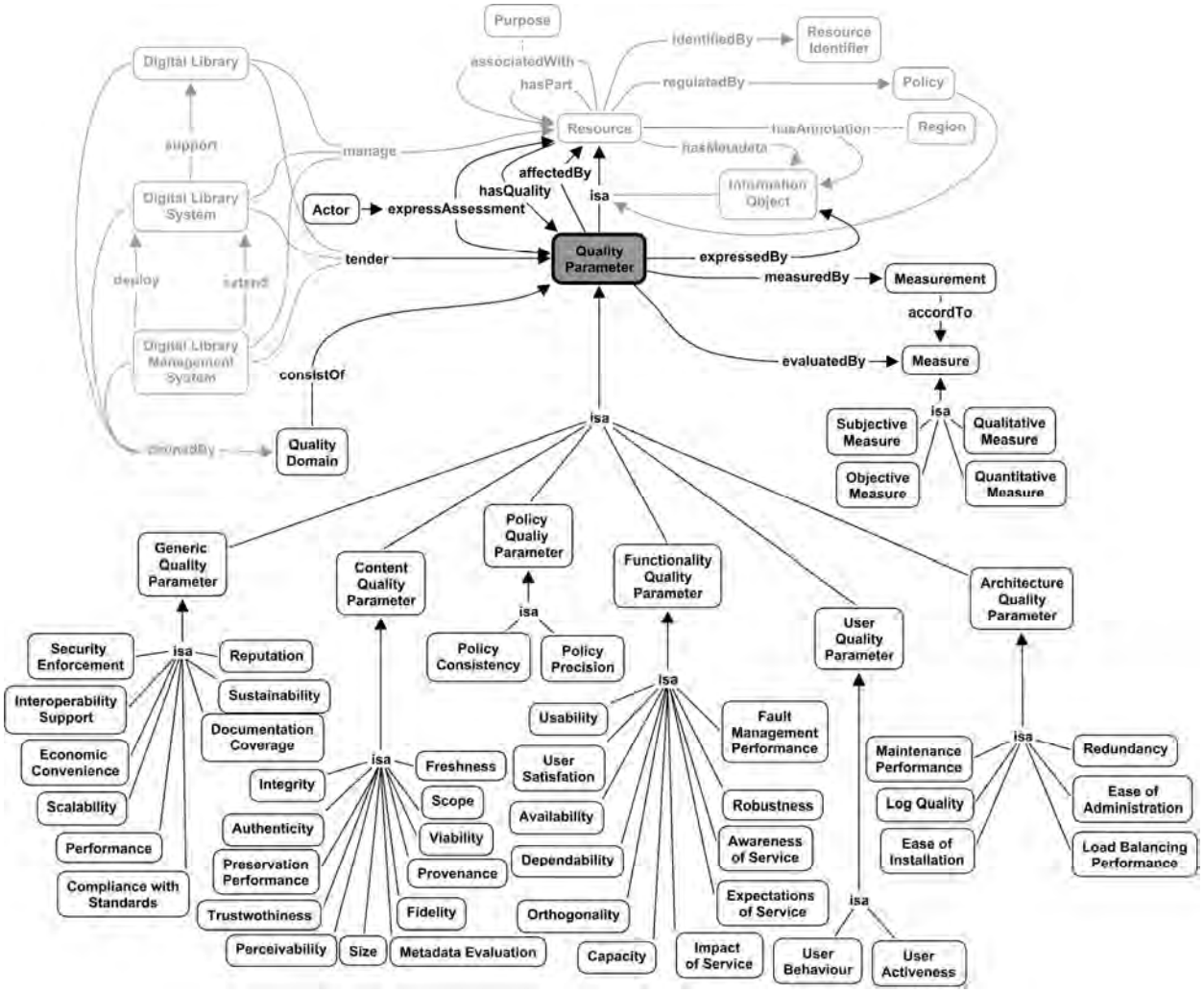


Figure II.2-16. Quality Domain Concept Map

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

II.2.7 Architecture Domain

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

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

The architecture of a 'software system' is a concept easily understood by most engineers, system administrators and developers, but it is not easily definable. In *An Introduction to Software Architecture* [92], 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 [119], however, recognises that there is more than just structure in architecture, and defines it as 'the highest-level concept of a system in its environment'. Thus, this Group's understanding does not consider the architecture of a software system limited to an inner focus, but rather proposes to take into consideration the system as a whole in its usage and development environments.

For the purposes of this Reference Model, the architecture of a software system (at a given point) is defined as the organisation or structure of the system's significant components (***Architectural Component***) interacting with each other (<*use*>) through their interfaces (***Interface***). These components may in turn be composed of smaller and smaller components (<*composedBy*>) (Figure II.2-17); however, different *Architectural Components* may be incompatible with each other (<*conflictWith*>), i.e. cannot coexist in the context of the same system. The software industry and the literature when using the term 'component' refer to many different concepts. Here, we use the term 'component' to mean an encapsulated part of a system, ideally a 'non-trivial', 'nearly independent', and 'replaceable' part of a system that fulfils a clear function in the context of a well-defined architecture.[15] Each *Architectural Component* is a *Resource*, thus it inherits the *Resource*'s characterising aspects (cf. Section II.2.1), e.g. it is uniquely identified. As any *Resource*, components have *Metadata* (***Component Profile***) which provide fundamental information for managing them. These *Metadata* specify characteristics like the implemented or supported *Functions*, the implemented *Interfaces*, their governing *Policies*, and the *Quality Parameters* that specify the various quality facets describing how and how well the component performs with respect to some viewpoint.

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

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

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

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

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

System Architecture Components are realised by ***Hosting Nodes*** and ***Running Components***. A *Hosting Node* encapsulates the implementation of the environment needed to host and run *Software Components*. A *Running Component* represents a running instance of a *Software Component* (<*realisedBy*>) active on a *Hosting Node*.

[15] 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.

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

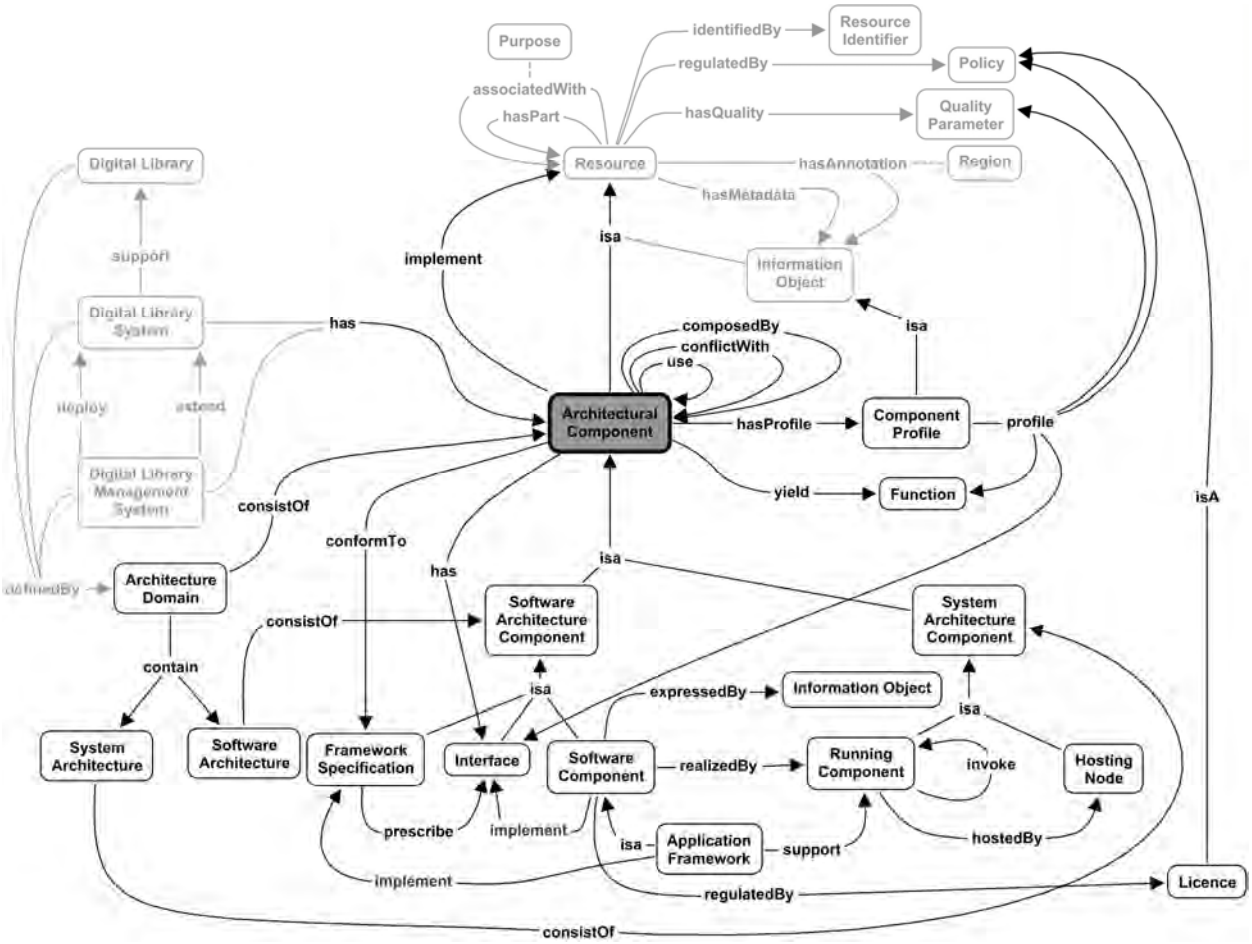


Figure II.2-17. Architecture Domain Concept Map

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

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

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

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

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

II.3 Reference Model in Action

The Reference Model sets out to contribute to digital library foundations, but its value is not merely theoretical. It also provides a core instrument for a large variety of different concrete usages, as demonstrated by the feedback received since the release of its first draft version. The Manifesto, for example, has been exploited several times to clarify to stakeholders the complexity of the Digital Library universe and the value of the Digital Library 'systems' in the content production and management workflow. At a very different level, the detailed specification of the concepts and relationships that characterise a Digital Library has been largely exploited in designing a concrete software service [94] that partially automates the process of creation of (virtual) digital libraries. Through this service, the effort spent by digital library designers and system administrators in performing this task is considerably reduced. The Reference Model has also been used as a basis for educational courses on digital libraries. Even if limited, the experience so far shows that the model provides a good integrated framework for introducing and explaining concepts. Starting from this framework, existing systems can easily be described and compared.

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

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

II.3.1 The Interoperability Issue

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

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

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

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

Alternative degrees of interoperability are often put in place. For instance, in the case of searching across multiple data sources provided by diverse organisations (digital libraries), usually three different approaches, characterised by a different level of engagement of the sources, are realised: the federated, the harvesting, and the gathering approach. In the federated approach the participating organisations agree on a set of protocols and standards to be applied in delivering the search, e.g. each source implements the SRU/SRW protocol because the federation imposes it. In this case, the semantic interoperability is at the level of *query* and *result set*. In the case of the harvesting approach (a notable example is represented by OAI-PMH [149]), the participating organisations make their content ready to be used by third parties according to a certain standard. Thus, no imposition comes from the potential consumers. Here, semantic interoperability is usually based on the use of a common *ontology*, e.g. Dublin Core. The third case, i.e. the gathering approach, is the least demanding of the three. In this case, no source takes care of its potential consumers, as the exposition of its content so that it can easily be used by third parties is not a requirement; in other words, in this case the *resources* are not required to be interoperable.

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

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

- ***Resource <hasMetadata> Information Object*** makes it possible to capture any Metadata for supporting interoperability.
- ***Resource <hasFormat> Resource Format*** makes it possible to capture the *Resource Format* with which a *Resource* is compliant. The notion of format is important for the correct interpretation of a *Resource*. For instance, in order for DL A to use an *Information Object* from DL B, DL A must either be

- 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.[36]

C146 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*

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: --

C147 Measure

Definition: A process for computing and assigning a value (*Measurement*) to a *Quality Parameter*.

Relationships:

- *Quality Parameter* is <*evaluatedBy*> *Measure*
- *Subjective Measure <isa> Measure*
- *Objective Measure <isa> Measure*
- *Qualitative Measure <isa> Measure*
- *Quantitative Measure <isa> Measure*
- *Measurement* is assigned according to *(<accordTo>) a Measure*

Rationale: See *Quality Parameter*.

Examples:

- See Quality Parameter.

C148 Objective Measure

Definition: A *Measure* that is well-defined and does not depend on individual perception.

Relationships:

[36] This example is valid also for C135 Preservation Policy.

- *Objective Measure <isa> Measure*

Rationale: *Objective Measures* could be obtained by taking measurements and using an analytical method to estimate the quality achieved. They could also be based on processing and comparing measurements between a reference sample and the actual sample obtained by the system.

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

Examples:

- Examples of objective factors related to the perception of audio recordings in a *Digital Library* are: noise, delay and jitter.

C149 Subjective Measure

Definition: A *Measure* based on, or influenced by, personal feelings, tastes or opinions.

Relationships:

- *Subjective Measure <isa> Measure*

Rationale: *Subjective Measures* involve performing opinion tests, user surveys and user interviews which take into account the inherent subjectivity of the perceived quality and the variations between individuals. The perceived quality is usually rated by means of appropriate scales, where the assessment is often expressed in a qualitative way using terms such as bad, poor, fair, good, excellent to which numerical values can be associated to facilitate further analyses.

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

Examples:

- Examples of factors related to the subjective perception of audio recordings in a *Digital Library* are: listening quality, loudness, listening effort.

C150 Qualitative Measure

Definition: A *Measure* based on a unit of measurement that is not expressed via numerical values.

Relationships:

- *Qualitative Measure <isa> Measure*

Rationale: *Qualitative Measures* are applied when the collected data are not numerical in nature. Although qualitative data can be encoded numerically and then studied by quantitative analysis methods, qualitative measures are exploratory while quantitative measures usually play a confirmatory role. Methods of *Qualitative Measure* that could be applied to a DL are direct observation; participant observation; interviews; auditing; case study; collecting written feedback.

Examples:

- The opinions of the users expressed in a DL forum or blog can be used as a source for *Qualitative Measure* of important issues for the users (content analysis is one of the popular techniques for analysing texts).

C151 Quantitative Measure

Definition: A *Measure* based on a unit of measurement that is expressed via numerical values.

Relationships:

- *Quantitative Measure <isa> Measure*

Rationale: *Quantitative Measures* are based on collecting and interpreting numerical data. There is a wide range of statistical methods for their analysis.

Examples:

- *Quantitative Measure* is applied when collecting data and calculating the mean time spent by users in locating content.

C152 Measurement

Definition: The action of, and the value obtained by, measuring a *Quality Parameter* in accordance with a selected *Measure*.

Relationships:

- *Quality Parameter <measuredBy> Measurement*
- *Measurement* is assigned according to (<*accordTo*>) a *Measure*

Rationale: See *Quality Parameter*.

Examples:

- See Quality Parameter.

C153 Quality Parameter

Definition: A *Resource* that indicates, or is linked to, performance or fulfilment of requirements by another *Resource*. A *Quality Parameter* is evaluated by (<*evaluatedBy*>) a *Measure*, is <*measuredBy*> a *Measurement*, and expresses the assessment (<*expressAssessment*>) of an *Actor*.

Relationships:

- *Quality Parameter <isa> Resource*
- *Quality Domain <consistOf> Quality Parameters*
- *Resource <hasQuality>* with respect to *Quality Parameter*
- *Actor <expressAssessment>* about *Resources* according to *Quality Parameters*
- *Quality Parameter* is *<evaluatedBy> Measure*
- *Quality Parameter* is *<measuredBy> Measurement*
- *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*

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 [22] and [101].

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 [188]: 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,[37] which are *Actors*, but in this case not human.

[37] http://trec.nist.gov/trec_eval/

C154 Generic Quality Parameter

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

Relationships:

- *Generic Quality Parameter <isa> Quality Parameter*
- *Reputation <isa> Generic Quality Parameter*
- *Economic Convenience <isa> Generic Quality Parameter*
- *Sustainability <isa> Generic Quality Parameter*
- *Security Enforcement <isa> Generic Quality Parameter*
- *Interoperability Support <isa> Generic Quality Parameter*
- *Documentation Coverage <isa> Generic Quality Parameter*
- *Performance <isa> Generic Quality Parameter*
- *Scalability <isa> Generic Quality Parameter*
- *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*.

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.

C155 Economic Convenience

Definition: A *General Quality Parameter* reflecting how favourable the economic efficiency is when using a *Digital Library*.

Relationships:

- Economic Convenience <*isa*> *Generic Quality Parameter*
- Economic Convenience <*affectedBy*> *Charging Policy*

Rationale: This parameter evaluates the economic conditions for using the *Digital Library* in order to determine if they are sufficiently advantageous.

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

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

Examples:

- An institution may find it advantageous to pay a moderate subscription for offering access to standard functionalities to all of its users and then pay an extra amount of money for access to more advanced functionalities for a restricted set of users who actually need them.

- As another example, consider the possibility of paying a basic fee for subscription to a set of standard *Collections* of a *Digital Library* and pay on a per-*Information Object* basis when you access *Information Objects* belonging to a *Collection* you are not subscribed to.

C156 Interoperability Support

Definition: A *Generic Quality Parameter* reflecting the capability of a *Digital Library* to inter-operate with other *Digital Libraries*.

Relationships:

- *Interoperability Support* <*isa*> *Generic Quality Parameter*
- *Interoperability Support* 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 [149] and [150] and the OAI-ORE initiative;[38] examples of interoperability efforts at the service level are the SRU/SRW[39] protocol and the Web Services.[40]

C157 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*

[38] http://www.openarchives.org/ore/

[39] http://www.loc.gov/standards/sru/

[40] http://www.w3.org/2002/ws/

- *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.

C158 Security Enforcement

Definition: A *Generic Quality Parameter* reflecting the level and kind of security features offered by a *Digital Library*.

Relationships:

- *Security Enforcement* <*isa*> *Generic Quality Parameter*
- *Security Enforcement* <*affectedBy*> *Digital Rights Management Policy*
- *Security Enforcement* <*affectedBy*> *Access Resource*
- *Security Enforcement* <*affectedBy*> *Configure DL*
- *Security Enforcement* <*affectedBy*> *User Behaviour*

Rationale: This parameter reflects the capability of the *Digital Library* to support the management of different levels of security as expected by users, content depositors, rights owners and librarians themselves.

Security Enforcement can be affected by both *Policies* and *Functions*. In particular, the *Digital Rights Management Policy* affects the level of *Security Enforcement* of a *Digital Library*, since it defines how the content has to be controlled. The *Access Resources* functions and their implementation influence *Security Enforcement*, since they provide *Actors* with mechanisms for consuming *Information Objects*; the *Configure DL* functions impact *Security*, since the possibility of correct and careful configuration of the *Digital Library* is a prerequisite for security; finally, *User Behaviour* can affect the *Security Enforcement*, since an *Actor* may compromise security, for example by careless use of username and password.

Examples:

- An example of a factor that influences Security Enforcement is the capability to prevent unauthorised access to content or the saving of local copies of copyrighted material. Within the Policy domain the regulations should be clearly stated in the Digital Rights Management Policy.

C159 Sustainability

Definition: A *Generic Quality Parameter* reflecting the prospects of durability and future development of a *Digital Library*.

Relationships:

- *Sustainability <isa> Generic Quality Parameter*
- *Sustainability <affectedBy> Change Management Policy*
- *Sustainability <affectedBy> Collection Development*
- *Sustainability <affectedBy> Compliance with Standards*
- *Sustainability <affectedBy> Maintenance*

Rationale: *Sustainability* should take into consideration various factors, such as the organisational and economic aspects of a *Digital Library*, as well as its capability of ensuring the preservation of its *Content* and of keeping pace with future innovations.

Sustainability may be affected by the *Policies* adopted by the *Digital Library*, such as the *Change Management Policy* or the *Collection Development Policy*.

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

Examples:

- Examples of factors that influence *Sustainability* are: the funding scheme that ensures the economic conditions for carrying on the *Digital Library*; the skills and willingness within the organisation that provides for the *Digital Library*; the presence of accurate development plans for the collections held by the *Digital Library*, as well as for the software and hardware resources needed for the *Digital Library System* and the *Digital Library Management System*.

C160 Documentation Coverage

Definition: A *Generic Quality Parameter* measuring the accuracy and clarity of the documentation describing a given *Resource*.

Relationships:

- *Documentation Coverage <isa> Generic Quality*

Rationale: This *Quality Parameter* addresses the quality of the written documentation of a *Resource*. The importance of documentation associated to *Resources* of any form is usually underestimated. On the contrary, having a valuable documentation reflects in an optimal usage of the available Resources.

Examples:

- Manuals explaining the use of *Function*s are typical examples of *Documentation Coverage*.
- Other examples are the accuracy of online help, better if contextual, or the selection provided by the Frequently Asked Question sections.

C161 Performance

Definition: A *Generic Quality Parameter* measuring the 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.

C162 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 [34]. 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 [116].

Examples:

- The ability of a DLS to support a growing number of users and/or provide access to (massively) growing collections without deterioration in performance.
- Another example is the ability to increase the number of requests served by a *Function* while keeping response time reasonable.

C163 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.

C164 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*.

C165 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[41] [75][76].

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.

C166 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.

[41] http://www.interpares.org/

C167 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.

C168 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 [22]. 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.

C169 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.

C170 Provenance

Definition: A *Content Quality Parameter* recording how well the origins and history of an *Information Object* are known and traced.

Relationships:

- *Provenance <isa> Content Quality Parameter*
- *Provenance <affectedBy> Metadata*
- *Provenance <affectedBy> Annotation*
- *Provenance <affectedBy> Preservation Policy*
- *Provenance <affectedBy> Information Object*

Rationale: This *Quality* parameter is aimed at determining how far it is possible to reconstruct the history and evolution of an *Information Object* in order to know if it fits the purpose. An *Information Object* may be derived from other *Information Objects* (e.g. due to a merger and/or transformations); tracing its provenance is not always a trivial task.

In particular, when we are dealing with scientific data, *Provenance* of data must be traced since a scientist needs to know where the data came from and what cleaning, rescaling or modelling was done to arrive at the data to be interpreted [1].

Note that this parameter resembles what [22] calls 'interpretability'.

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

Provenance <affectedBy> Annotation, since *Annotations* allow us to trace the provenance and flow of data, report errors or remarks about a piece of data, and describe the quality or the security level of a piece of data [28]. In addition, [103] uses annotations in interactive visualisation systems as a means of both capturing the history of user interaction with the visualisation system and keeping track of the observations that a user may make while exploring the visualisation.

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

There are several initiatives related to provenance and, as discussed in Appendix C, provenance has a larger coverage that those captured here. For instance, the W3C Provenance Incubator Group (2009-2010) is working to develop a roadmap in the area of provenance for Semantic Web technologies, development, and possible standardization. Provenance helps to track authorship, enforce intellectual property rights, validate the integrity and authenticity of work and its quality, and to reproduce the work. When databases share heterogeneous data, it is crucial to have information about their history, as they are moved or copied from place to place, or evolve over time. This form of provenance is especially important in settings, such as bioinformatics. Researchers will need to know where those data come from and if they have been modified since they were obtained.

Examples:

- Consider a bioinformatics DL, which supports the analysis of gene expressions. This usually requires a set of tools that need to be applied to raw data in a certain order by a dedicated workflow. Since reproduction of results is a very important requirement in this domain, not only the result of a

workflow but also all intermediate steps of this workflow, including the configuration of tools and algorithms, need to be kept as part of the preservation metadata of the *Information Object* that represents the final result.

C171 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.

C172 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.

C173 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*.

C174 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.

C175 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.

C176 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?

C177 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.

C178 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.

C179 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.

C180 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.

C181 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.

C182 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.

C183 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.

C184 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.

C185 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.

C186 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.

C187 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.

C188 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.

C189 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).

C190 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*.

C191 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.

C192 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.

C193 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.

C194 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.

C195 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).

C196 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*.

C197 Ease of Installation

Definition: An *Architecture Quality Parameter* measuring the ease of installation and configuration of *Software Components*.

Relationships:

- *Ease of Installation <isa> Architecture Quality Parameter*

Rationale: The *Ease of Installation* parameter concerns the presence of tools and procedures for seamlessly installing and deploying *Software Components*, as well as adding new *System Architecture Components* to an operating *Digital Library System*.

Examples:

- The presence of intuitive wizards for installing new components or the possibility of adding components without restarting the whole system are examples of factors that influence *Ease of Installation*.

C198 Load Balancing Performance

Definition: An *Architecture Quality Parameter* measuring the capacity to spread and distribute work evenly across *System Architecture Components*.

Relationships:

- *Load Balancing Performance <isa> Architecture Quality Parameter*

Rationale: *Load Balancing Performance*, together with *Redundancy*, may help in improving the overall performance and responsiveness of a *Digital Library System*.

Examples:

- For a DLS on top of a Grid environment, which takes into account several instances of *Architectural Components*, *Load Balancing Performance* includes the ability of the system to distribute requests equally among different components of the same type within the system. In particular, this capability consists in selecting *Hosting Nodes* according to their workload or moving a job from one

Hosting Node to another in order to achieve optimal *Resource* utilisation so that no *Resource* is over/under-utilised.

C199 Log Quality

Definition: An *Architecture Quality Parameter* measuring the presence and accuracy of logs which monitor the activity and functioning of *System Architecture Components*.

Relationships:

- *Log Quality <isa> Architecture Quality Parameter*

Rationale: The presence of accurate logs is crucial for understanding, analysing, debugging and improving the functioning of a *Digital Library System*.

Furthermore, log analysis can be an effective means of understanding *Actor* behaviour and personalising the *Digital Library System* accordingly; therefore, logs can provide useful input for the *Personalise* functions and for creating *Actor Profiles*.

Examples:

- There are various standards for creating logs. For example, in the case of the Web, there is W3C Extended Log Format [110].

C200 Maintenance Performance

Definition: An *Architecture Quality Parameter* addressing the design and implementation of software and hardware maintenance plans for *Architectural Components*.

Relationships:

- *Maintenance Performance <isa> Architecture Quality Parameter*
- *Maintenance Performance <affectedBy> Change Management Policy*

Rationale: *Maintenance Performance* concerns the design of plans for keeping *Architectural Components* updated with research and technological advances.

Change Management Policy may affect *Maintenance Performance*, since it regulates the change process in a *Digital Library*.

It may influence *Sustainability*, as it involves keeping the current system functioning properly and evolving it to face future technological developments.

Examples:

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

C201 Redundancy

Definition: An *Architecture Quality Parameter* measuring the degree of (partial) duplication of *System Architecture Components* to decrease the probability of a system failure.

Relationships:

- *Redundancy <isa> Architecture Quality Parameter*

Rationale: A redundant architecture helps in improving the overall performance of a system and may improve the *Availability*, *Dependability* and *Robustness* of a *Digital Library System*.

Examples:

- Availability of a system can be increased by *Redundancy* of *Architectural Components*. In the event that one component fails, another component of the same type is able to take over.

C202 Architecture Domain

Definition: One of the six main concepts characterising the Digital Library universe. It represents the various aspects related to the software systems that concretely realise the Digital Library universe.

Relationships:

- *Digital Library <definedBy> Architecture Domain*
- *Digital Library System <definedBy> Architecture Domain*
- *Digital Library Management System <definedBy> Architecture Domain*
- *Architecture Domain <consistOf> Architectural Component*

Rationale: The *Architecture Domain* encompasses concepts and relationships characterising the two software systems that play an active role in the DL universe, i.e. DLSs and DLMSs. Unfortunately, the importance of this fundamental concept has been largely underestimated in the past. The importance of the domain and its modelling is described in Section II.2.7.

Examples: --

C203 Architectural Component

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

Relationships:

- *Architectural Component <isa> Resource*
- *Architectural Component <yield> Function*
- *Architectural Component <hasQuality> Quality Parameter* (inherited from *Resource*)
- *Architectural Component* is *<regulatedBy> Policy* (inherited from *Resource*)
- *Architectural Component <hasProfile> Component Profile*
- *Architectural Component <conformTo> Framework Specification*
- *Architectural Component <use> Architectural Components*
- *Architectural Component <composedBy> Architectural Components*
- *Architectural Component <conflictWith> Architectural Components*
- *Architectural Component <has> Interface*
- *Software Architecture Component <isa> Architectural Component*
- *System Architecture Component <isa> Architectural Component*

Rationale: The notion of *Component* has been introduced in modern software systems to represent 'elements that can be reused or replaced'. By exploiting such an approach, systems gain the potential to be:

- flexible – users' needs change over time, even while the system is being developed. It is important to be able to apply changes to the system at a later stage. Moreover, it should be possible/easy to fix the bugs;

A.15 Quality Domain Concept Map

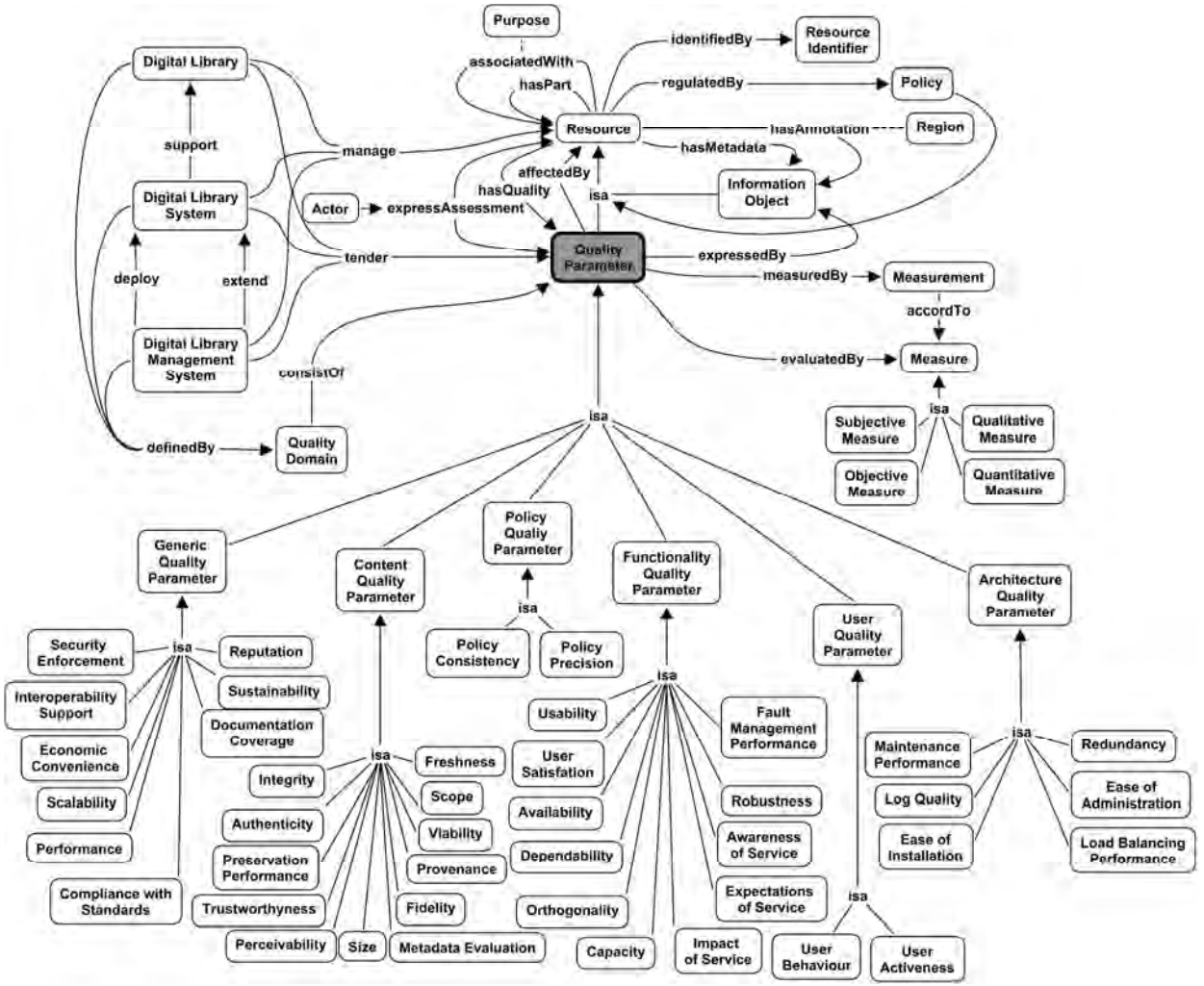


Figure A-15. Quality Domain Concept Map (A4 format)

B.9 Quality Domain

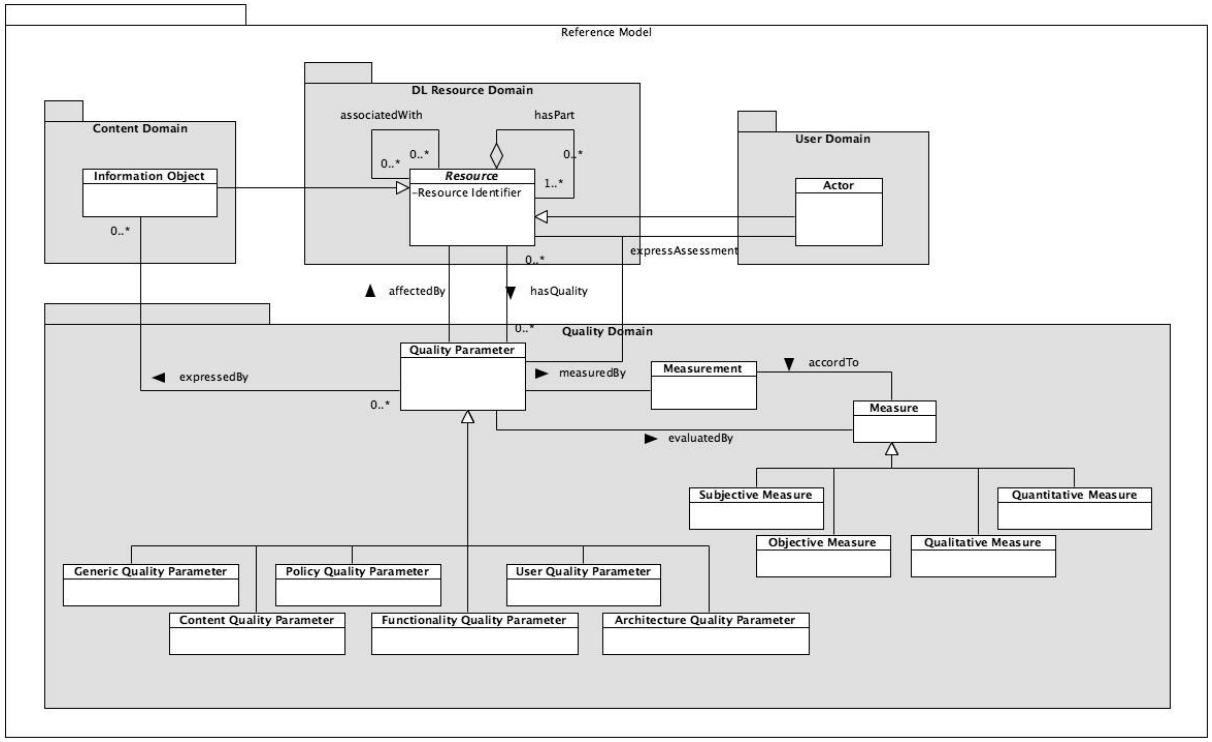


Figure B-9. Quality Domain UML Class Diagram

B.10 Quality Parameter Hierarchy

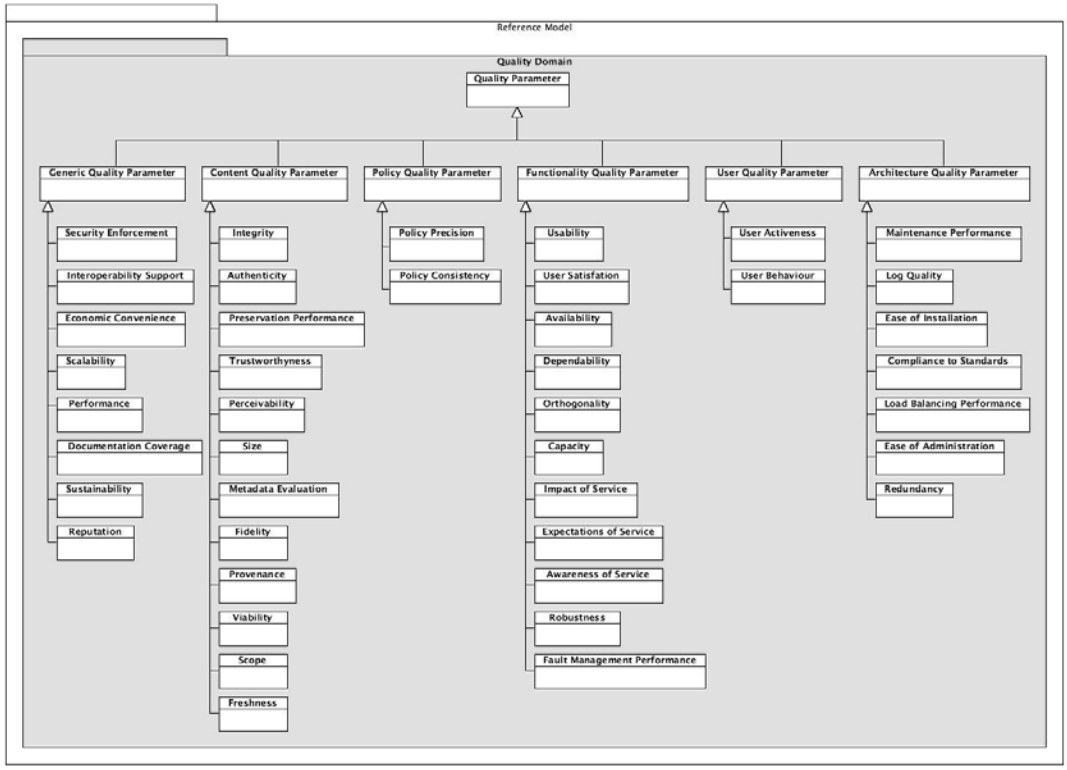


Figure B-10. Quality Parameter Hierarchy UML Class Diagram