Streams, Structures, Spaces, Scenarios, and Societies (5S): A Formal Digital Library Framework and Its Applications

Marcos André Gonçcalves

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fullfilment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

November 29, 2004
Blacksburg, Virginia

Keywords: digital libraries, ontology, semantic modeling, log standard, 5SL, 5SGen, 5SGraph, theory, quality.

Streams, Structures, Spaces, Scenarios, and Societies (5S):

A Formal Digital Library Framework and Its Applications

Marcos André Gonçcalves

Abstract

Digital libraries (DLs) are complex information systems and therefore demand formal foundations lest development efforts diverge and interoperability suffers. In this dissertation, we propose the fundamental abstractions of Streams, Structures, Spaces, Scenarios, and Societies (5S), which allow us to define digital libraries rigorously and usefully. Streams are sequences of arbitrary items used to describe both static and dynamic (e.g., video) content. Structures can be viewed as labeled directed graphs, which impose organization. Spaces are sets with operations that obey certain constraints. Scenarios consist of sequences of events or actions that modify states of a computation in order to accomplish a functional requirement. Societies are sets of entities and activities, and the relationships among them. Together these abstractions provide a formal foundation to define, relate, and unify concepts – among others, of digital objects, metadata, collections, and services – required to formalize and elucidate “digital libraries”. A digital library theory based on 5S is defined by proposing a formal ontology that defines the fundamental concepts, relationships, and axiomatic rules that govern the DL domain. The ontology is an axiomatic, formal treatment of DLs, which distinguishes it from other approaches that informally define a number of architectural invariants. The applicability, versatility, and unifying power of the 5S theory are demonstrated through its use in a number of distinct applications including: 1) building and interpreting a DL taxonomy; 2) informal and formal analysis of case studies of digital libraries (NDLTD and OAI); 3) utilization as a formal basis for a DL description language, digital library visualization and generation tools, and a log format specific for DLs; and 4) defining a quality model for DLs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Problem

Digital libraries may be extremely complex information systems. The proper concept of a digital library seems hard to completely understand and evades definitional consensus. Different views (*e.g.*, historical, technological) and perspectives (*e.g.*, from the library and information science, information retrieval, or human-computer interaction communities) have led to a myriad of differing definitions. Licklider, in his seminal work [130, pp. 36–39], visualized a collection of digital versions of the worldwide corpus of published literature and its availability through interconnected computers. More recently, Levy and Marshall gave a view of digital libraries as a polygamy of documents, technology, and work [128]. Lesk analyzed the relative weights of the words *digital* and *library* in recent efforts in the field, and concluded that many of those efforts are dissociated from an understanding of users' needs and their use of the resources being provided [126]. Borgman explicitly explored the competing visions of the digital library field, both from research and from practitioner communities, and showed the difficulty that this conflict imposes on activities like defining terms, characterizing terminologies, and establishing contexts [29]. A Delphi study of digital libraries coalesced a broad definition: organized collection of resources, mechanisms for browsing and searching, distributed networked environments, and sets of services objectified to meet users' needs [117]. The President's Information Technology Advisory Committee (PITAC) Panel on Digital Libraries discusses "digital libraries – the networked collections of digital text, documents, images, sounds, scientific data, and software that are the core of today's Internet and tomorrow's universally accessible digital repositories of all human knowledge" [165]. Underlying all of these is the consensus agreement that digital libraries are fundamentally complex.

Such complexity is due to the inherently interdisciplinary nature of this kind of system. Digital libraries integrate findings from disciplines such as hypertext, information retrieval, multimedia services, database management, and human-computer interaction [70]. The need to accommodate all these characteristics complicates the understanding of the underlying concepts and functionalities of digital libraries, thus making it difficult and expensive to construct new digital library systems. Designers of digital libraries are most often library technical staff, with little to no formal training in software engineering, or computer scientists with little background in the research findings about information retrieval or hypertext. Thus, digital library systems are usually built from scratch using home-grown architectures that do not benefit from digital library and software design experience. Wasted effort and poor interoperability can therefore ensue, raising the costs of digital libraries and reducing the fluidity of information assets in the future.

The broad and deep requirements of digital libraries demand new frameworks and theories in order to understand better the complex interactions among their components [81]. Supporting this claim, the summary report of the Joint NSF-European Union (EU) Working Groups on Future Directions of Digital

Libraries Research recommended that “new frameworks and theories be developed in order to understand the complex interactions between the various components in a globally distributed digital library” [185]. However, though the necessity for such an underlying theory has long been perceived and advocated, little if any progress has been made towards a formal framework or theory for digital libraries.

Formal frameworks and theories are crucial to specify and understand clearly and unambiguously the characteristics, structure, and behavior of complex information systems. It is not surprising that most of the disciplines related to digital libraries have underlying formal frameworks and theories that have steered them well: databases [44, 212, 21, 39, 3], information retrieval [177, 170, 211, 207, 108, 13], and hypertext and multimedia [132, 54]. A formal framework abstracts the general characteristics and common features of a set of systems developed for similar problems, explains their structures and processes, and strengthens common practice. Furthermore, formal frameworks for information systems can be used for the design of a real system, providing a precise specification of requirements against which the implementation can be compared for correctness. The lack of formal theories and frameworks lead to diverging efforts and has made interoperability one of the most important problems faced by the field.

1.2 Hypotheses and Research Questions

In this dissertation, we present the first comprehensive framework for digital libraries – 5S (Streams, Structures, Spaces, Scenarios, and Societies). The two main hypotheses of this dissertation are:

- A formal theory for DLs can be built based on 5S;
- The formal theory can serve as a basis for modeling and building high-quality DLs.

These two hypotheses lead to the following research questions, which we will try to answer in this dissertation.

- Can we formally elaborate 5S?
- How can we use 5S to formally describe digital libraries?
- What are the fundamental relationships among the Ss and high-level DL concepts?
- How can we allow digital librarians to easily express those relationships?
- Which are the fundamental quality properties of a DL? Can we use the formalized DL framework to characterize those properties?
- Where in the life cycle of digital libraries can key aspects of quality be measured and how?

1.3 Scope of the Dissertation

Figure 1.1 shows the scope of this dissertation which is concentrated in the top portion of the figure. Here we are interested in a theory of DLs which is abstracted from the commonalities existent among disparate DL systems/architectures. Also we are interested in how those theories can be represented (e.g., symbolic mathematics, markup languages) and instantiated to build real DL running systems. In terms of societal interactions (represented as dashed lines) we are only interested in those happening between actors (users of the DL) and (parts of) the system. Societal interactions among actors which do not go through the DL system are not covered here. Also, actors interact with objects of the real world (small balls). In this dissertation, we do not focus on these objects or how they are represented in the system. In other words, we focus mostly on (information) objects ‘born’ digital not in surrogates of objects existing in the real world. Finally, the quality portion of this work touches both theory and aspects of the running system (represented as a gray area marked with “Q”). But even this part has a more system-oriented perspective instead of focusing on usage issues.

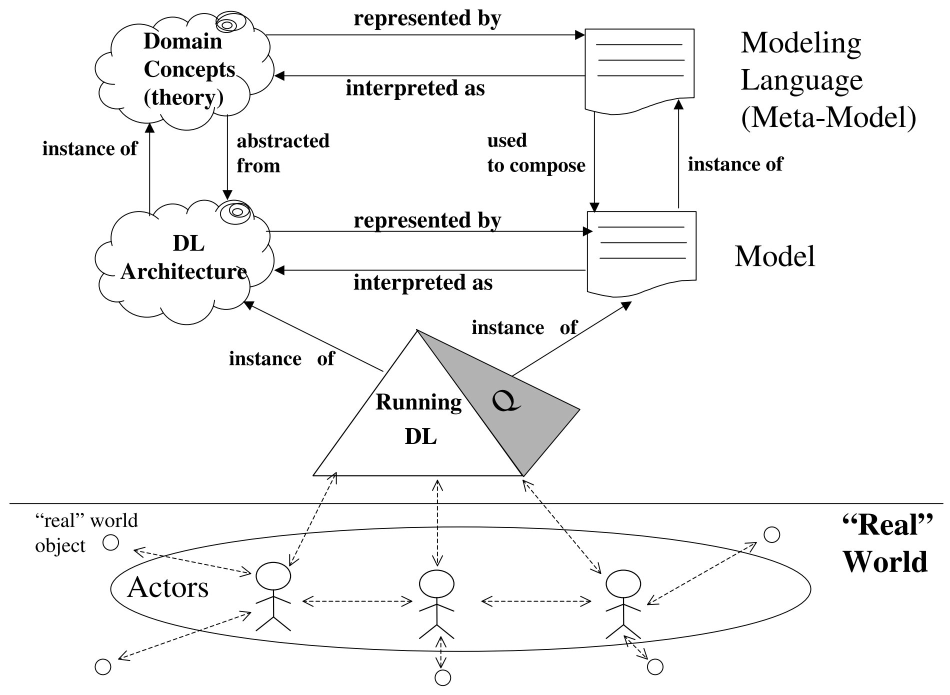


Figure 1.1: Scope of the dissertation

1.4 Outline of the Dissertation

This dissertation is logically organized in three ‘parts’. Part 1 deals with the theoretical aspects of the work. Part 2 shows how to put theory into practice by showing a number of applications/tools based on the theoretical framework. Part 3 focuses in a different type of application, namely a quality model for DLs. The chapters within the parts are organized as follows:

- Chapter 2: Introduction to 5S and to the formal framework, published in *ACM Transactions on Information Systems*, Vol. 22, No. 2, April 2004, with E. A. Fox, L.T Watson, and N. A. Kipp [90].

- Chapter 3: Introduction to the digital library ontology, early version published in *Proceedings of the ACM SIGIR Workshop in Mathematical Formal Methods in Information Retrieval*, July 29, 2004, Sheffield, England, with E. A. Fox, and L. T. Watson [88].

- Chapter 4: Introduction to the 5SL Language for Declarative Specification of DLs, published in *Proceedings of the Second Joint ACM / IEEE-CS Joint Conference on Digital Libraries*, July 14-18, 2002, Portland, with E. A. Fox [86]

- Chapter 5: Introduction to 5SGraph, a tool for visual semantic modeling of DLs, published in the *Proceedings of the 7th European Conference on Digital Libraries (ECDL 2003)*, 17-22 August, Trondheim, Norway, Springer LNCS 2769, 2003, with Q. Zhu, R. Shen, and E. A. Fox [236]. A demostration of the tool was conducted during the 3rd Joint ACM / IEEE-CS Joint Conference on Digital Libraries, May 27-31, 2003, Houston, with a summary being published in the proceedings of the conference [235].

- Chapter 6: Introduction to 5SGen, a tool for (semi-)automatic generation of DLs from scenario and societies models expressed in 5SL, published in the *7th European Conference on Digital Libraries (ECDL 2003)*, 17-22 August, Trondheim, Norway, Springer LNCS 2769, 2003, with R. Kelapure, and E. A. Fox [114].

- Chapter 7: Introduction to a proposal for an XML-based log standard for DLs and associated tools, published in the *Proceedings of the 6th European Conference on Digital Libraries (ECDL 2002)*, September 16-18, Rome. Italy, 2002, LNCS 2458, with M. Luo, R. Shen, M. F. Ali, and E. A. Fox [93]. Some extensions of the log format and the tools were also published as a short paper in *Proceedings of the Third Joint ACM / IEEE-CS Joint Conference on Digital Libraries*, May 27-31, 2003, Houston, with G. Panchanathan, U. Ravindranathan, A. Krowne, E. A. Fox, F. Jagodzinski, and L. Cassel [96].

- Chapter 8:Introduction to the proposed Quality Model for DLs, abstract published in the *Proceedings of the DELOS Workshop on the Evaluation of Digital Libraries*, October 4-5, Padova, Italy, 2004, with E. A. Fox, B. Zhang, and L. T. Watson [91].

- Chapter 9: Conclusions and Discussion into future work.

- Appendices

Related work is covered in the context of each chapter.

Part I

Theory

Chapter 2

Streams, Structures, Spaces, Scenarios, Societies (5S): A Formal Framework for Digital Libraries

In this chapter, we introduce 5S and formalisms for streams, structures, spaces, scenarios, and societies—as a framework for providing theoretical and practical unification of digital libraries. These formalisms are important for making sense of complexity and can ultimately serve as an aid for designers, implementers, and evaluators of digital libraries. These abstractions work with other known and derived definitions to yield a formal, rigorous framework of digital libraries.

This chapter is organized as follows. Section 2.2 presents an overview of 5S, including definitions and examples. Section 2.3 discusses two applications of 5S including: a) construction and interpretation of a DL taxonomy; and b) informal analysis of case studies of digital libraries. Sections 2.2 and 2.3 are purposely informal and introduce most of the key concepts in an intuitive manner without complete precision. The role of Section 2.4 is to formally define key information constructs that were introduced in the previous sections. Section 2.5 then builds on this framework to formally describe several DL higher level constructs and settings. Section 2.6 illustrates the application of the formal framework. Section 2.7 discusses related work.

2.1 5S Overview: Informal Definitions

2.1.1 Streams

Streams are sequences of elements of an arbitrary type (e.g., bits, characters, images, etc.). In this sense, they can model both static and dynamic content. The first includes, for example, textual material, while the later might be, for example, a presentation of a digital video, or a sequence of time and positional data (e.g., from a GPS) for a moving object.

A dynamic stream can represent an information flow—a sequence of messages encoded by the sender and communicated using a transmission channel possibly distorted with noise, to a receiver whose goal is to reconstruct the sender's messages and interpret message semantics [190]. Dynamic streams are thus important for representing whatever communications take place in the digital library. Examples of dynamic streams include video-on-demand delivered to a viewer, a timed sequence of news sent to a client, a timed sequence of frames that allows the assembly of a virtual reality scenario, etc. Typically, a dynamic stream is understood through its temporal nature. A dynamic stream then can be interpreted as a finite sequence of clock times and associated values[1] that can be used to define a stream algebra, allowing operations on diverse kinds of multimedia streams [134]. The synchronization of streams can be specified with Petri Nets [152] or other approaches.

In the static interpretation, the temporal nature is generally ignored or is irrelevant, and a stream corresponds to some information content that is interpreted as a sequence of basic elements, often of the same type. A popular type of static stream according to this view is text (sequence of characters). The type of the stream defines its semantics and area of application. For example, any text representation can be seen as a stream of characters, so that text documents, such as scientific articles and books, can be considered as structured streams.

2.1.2 Structures

A structure specifies the way in which parts of a whole are arranged or organized. In digital libraries, structures can represent hypertexts, taxonomies, system connections, user relationships, and containment – to cite a few. Books, for example can be structured logically into chapters, sections, subsections, and paragraphs; or physically into cover, pages, line groups (paragraphs), and lines [77]. Structuring orients readers within a document's information.

Markup languages (e.g., SGML, XML, HTML) have been the primary form of exposing the internal structure of digital documents for retrieval and/or presentation purposes [78, 45, 85]. Relational and object-oriented databases impose strict structures on data, typically using tables or graphs as units of structuring [21].

With the increase in heterogeneity of material continually being added to digital libraries, we find that much of this material is called "semistructured" or "unstructured". These terms refer to data that may have some structure, where the structure is not as rigid, regular, explicit, or complete as the structure used by structured documents or traditional database management systems [2]. Query languages and algorithms can extract structure from these data [119, 3, 147]. Although most of those efforts have a "data-centric" view of semi-structured data, works with a more "document-centric view" have emerged [12, 74, 73]. In general, humans and natural language processing systems can expend considerable effort to unlock the interwoven structures found in texts at syntactic, semantic, pragmatic, and discourse levels.

2.1.3 Spaces

A space is a set of objects together with operations on those objects that obey certain constraints. The combination of operations on objects in the set is what distinguishes spaces from streams and structures. Since this combination is such a powerful construct, when a part of a DL cannot be described well using another of the Ss, a space may well be applicable. Despite the generality of this definition, spaces are extremely important mathematical constructs. The operations and constraints associated with a space define its properties. For example, in mathematics, affine, linear, metric, and topological spaces define the basis for algebra and analysis [83]. In the context of digital libraries, Licklider discusses spaces for information [130, p. 62]. In the information retrieval discipline, Salton and Lesk formulated an algebraic theory based on vector spaces and implemented it in the SMART system [177]. "Feature spaces" are sometimes used with image and document collections and are suitable for clustering or probabilistic retrieval [169]. Spaces also can be defined by a regular language applied to a collection of documents. Document spaces are a key concept in many digital libraries.

Human understanding can be described using conceptual spaces. Multimedia systems must represent real as well as synthetic spaces in one or several dimensions, limited by some metric or presentational space (windows, views, projections) and transformed to other spaces to facilitate processing (such as compression

[1]These values are undefined or a value of type T, e.g., boolean, integer, text, or image.

[191, 237]). Many of the synthetic spaces represented in virtual reality systems try to emulate physical spaces. Digital libraries may model traditional libraries by using virtual reality spaces or environments [20, 148]. Also, spaces for computer-supported cooperative work provide a context for virtual meetings and collaborations [47, 161].

Again, spaces are distinguished by the operations on their elements. Digital libraries can use many types of spaces for indexing, visualizing, and other services they perform. The most prominent of these for digital libraries are measurable spaces, measure spaces, probability spaces, vector spaces, and topological spaces.

2.1.4 Scenarios

One important type of scenario is a story that describes possible ways to use a system to accomplish some function that a user desires. Scenarios are useful as part of the process of designing information systems. Scenarios can be used to describe external system behavior from the user's point of view [118]; provide guidelines to build a cost-effective prototype [204]; or help to validate, infer, and support requirements specifications and provide acceptance criteria for testing [103, 205, 123]. Developers can quickly grasp the potentials and complexities of digital libraries through scenarios. Scenarios tell what happens to the streams, in the spaces, and through the structures. Taken together the scenarios describe services, activities, tasks, and those ultimately specify the functionalities of a digital library.

For example, user scenarios describe one or more users engaged in some meaningful activity with an existing or envisioned system. This approach has been used as a design model for hypermedia applications [153]. Human information needs, and the processes of satisfying them in the context of digital libraries, are well suited to description with scenarios, including these key types: fact-finding, learning, gathering, and exploring [226]. Additionally, scenarios can aid understanding of how digital libraries affect organizations and societies, and how challenges to support social needs relate to underlying assumptions of digital libraries [128]. Scenarios also may help us understand the complexities of current publishing methods, as well as how they may be reshaped in the era of digital libraries, by considering publishing paths, associated participants, and publication functions [225].

The concepts of state and event are fundamental to understanding scenarios. Broadly speaking, a state is determined by what contents are in specified locations, as, for example, in a computer memory, disk storage, visualization, or the real world. The nature of the values and state locations related to contents in a system are granularity-dependent and their formal definitions and interpretations are out of the scope of this chapter; the reader is referred to [227] for a lengthy discussion. An event denotes a transition or change between states, for example, executing a command in a program. Scenarios specify sequences of events, which involve actions that modify states of a computation and influence the occurrence and outcome of future events. Dataflow and workflow in digital libraries can be modeled using scenarios.

2.1.5 Societies

A society is a set of entities and the relationships between them. The entities include humans as well as hardware and software components, which either use or support digital library services. Societal relationships make connections between and among the entities and activities.

Examples of specific human societies in digital libraries include patrons, authors, publishers, editors, maintainers, developers, and the library staff. There are also societies of learners and teachers. In a human society, people have roles, purposes, and relationships. Societies follow certain rules and their members play different roles—participants, managers, leaders, contributors, or users. Members of societies have activities and relationships. During their activities, society members often create information artifacts—art, history, images, data—that can be managed by the library. Societies are holistic—substantially more than the sums of their constituents and the relationships between them. Electronic members of digital library societies, i.e., hardware and software components, are normally engaged in supporting and managing services used by humans.

A society is the highest-level component of a digital library, which exists to serve the information needs of its societies and to describe the context of its use. Digital libraries are used for collecting, preserving, and sharing information artifacts between society members. Cognitive models for information retrieval [22, 59, 32], for example, focus on user's information-seeking behavior (i.e., formation, nature, and properties of a user's information need) and on the ways in which information retrieval systems are used in operational environments.

Several societal issues arise when we consider them in the digital library context. These include policies for information use, reuse, privacy, ownership, intellectual property rights, access management, security, etc. [165]. Therefore, societal governance (law and its enforcement) is a fundamental concern in digital libraries. Language barriers are also an essential concern in information systems and internationalization of online materials is an important issue in digital libraries, given their globally distributed nature [151].

Economics, a critical societal concern, is also key for digital libraries [109]. Collections that were "born electronic" are cheaper to house and maintain, while scanning paper documents to be used online can be relatively expensive. Internet access is widely available and in many settings is inexpensive. Online materials are seeing more use, including from distant locations. Since distribution costs of electronic materials are very low, digital delivery makes economic sense. However, it brings the problem of long-term storage and preservation, which must be adequately addressed if the information being produced today is to be accessible to future generations [131].

2.2 Example of Applications of 5S

In this section, we illustrate the expressiveness and unifying power of 5S through two different example applications. In the first, we build a taxonomy of DL concepts derived from the literature and characterize the result in light of the framework. The second application uses 5S as an analytical tool to understand and dissect a DL instance and a DL protocol for interoperability.

2.2.1 Digital Library Taxonomy

A taxonomy is a classification system of empirical entities with the goal of classifying cases according to their measured similarity on several variables [19]. Classifications are a premier descriptive tool and as such, they give a foundation towards an explanation for a phenomena. Classifications provide a terminology and vocabulary for a field and help to reduce complexity and achieve parsimony of description by logically arranging concepts through the identification of similarities and differences. We have built a taxonomy for digital libraries as a classification system of terms involved with the field. Our taxonomy describes the digital library field in conceptual terms and therefore its organization is amenable to be interpreted in the light of 5S. This interpretation aims toward a more informal conceptual understanding of the 'Ss' and corresponding DL components.

In the process of building such a taxonomy, we have considered the principles of taxonomies in social sciences, notably cluster analysis, and faceted classification schemes [213]. In particular we were guided by the idea that writing about a subject unequivocally reveals the appropriate facets for that subject [65], and that those facets are enough to describe the phenomenon [163]. We followed an agglomerative strategy using subjective relational concepts like association and correlation. During the construction of the taxonomy we tried to accommodate all the terms found in the literature and marginal fields, guarantee mutual exclusivity, and ensure consistency and clarity.

To collect the unstructured list of concepts, we went through the early literature to find all features, issues, and roles utilized, and identified specific terms [89]. As a starting point, we used an initial set of terms and phrases listed alphabetically in [67]. To this list we added other terms from various articles. When this was reasonably voluminous, we produced a grouping of terms of similar or related meaning into "notational families" known as facets. Each group was given a label that described the idea behind the homogeneity of the group or the main variable considered. From there, we grouped the clusters, and so on, until we achieved convergence into one unique facet called "digital library."

Once the initial taxonomy was complete, we noticed certain terms were missing or ambiguous, so we added terms and qualified them in each context. After several iterations of successive clustering, declustering, and reclustering, we released a more concrete and consistent working set for peer review and then improved the taxonomy based on comments received. The resulting taxonomy is shown in Figure 2.1.

We must point out that, as with any classification system, our taxonomy must evolve to accommodate changes in the digital library field. However, two factors should contribute to the stability of the taxonomy, and therefore to its relative longevity. First the taxonomy was derived from a significant corpus of digital library literature; therefore it is more stable than personal opinions. Second, the higher-level groupings are significantly abstract so that they may be applied to many fields, with possible additions or changes necessary only at the level of specific categories. Clearly, such changes are likely due to the youth and rapid development of the field. In the following we describe the main facets and sub-facets of the taxonomy, making use of 5S as an analytical tool.

Actors: Who interacts with/within DLs? In our context, actors are the users of a digital library. Actors interact with the DL through an interface design that is (or should be) affected by the actors' preferences and needs. Actors who have preferences and needs in common, display similar behavior in terms of services they use and interactions they practice. We say these actors form a *digital community*, the building blocks of a digital library society[2]. Communities—of students, teachers, librarians—interact with digital libraries and use digital libraries to interact, following pre-specified scenarios. Communities can act as a query-generator service, from the point of view of the library, and as a teaching, learning, and working service, from the point of view of other humans and organizations. Communications between actors and among the same and different communities occur through the exchange of streams. Communities of autonomous agents and computers also play roles in digital libraries. They instantiate scenarios upon requests by the actors of a DL. To operate, they need structures of vocabulary and protocols. They act by sending (possibly structured) streams of queries and retrieving streams of results.

Activities: What happens in DLs? Activities of digital libraries — abstracting, collecting, creating, disseminating, evaluating, modeling, organizing, personalizing, preserving, requesting, and selecting — all can be described and implemented using scenarios and occur in the DL setting as a result of actors using services. Furthermore, these activities make and characterize relationships within and between societies, streams, and structures. Each activity happens in a setting, arena, or space. The relationships developed can be seen in the context of larger structures (e.g., social networks [188, 112]).

Components: What constitutes DLs? Digital libraries can contain repositories of knowledge, information, data, metadata, relationships, logs, annotations, user profiles, and documents. They can be associated with higher-level structuring and organizational materials: term lists (e.g., authority files, dictionaries), classification tools (e.g., subject headings and taxonomies), thesauri, ontologies, and metadata catalogs. These

[2]Digital communities are formed by actors who interact with a DL possibly through the same interface paradigm. The actors might belong to distinct social communities of the real world. For instance, a digital community might be instantiated by the adoption of a particular architecture and interface for a DL (e.g., a chat room or MOO). This instantiation is somewhat arbitrary and artificial. Social communities, on the other hand, appear much more naturally as a result of complex social interactions.

Actors
- *Distributed Computers*
  - Clients
  - Servers
- *Electronic Agents*
  - Collection agents
  - Crawlers
  - Knowbots
  - Mediator agents
  - User agents
- *Humans*
  - Administrators
  - Authors
  - Annotators
  - Designers
  - Editors
  - Evaluators
  - Funders
  - Implementers
  - Learners
  - Librarians
  - Maintainers
  - Managers
  - Patrons
  - Publishers
  - Readers
  - Reviewers
  - Tool builders

Activities
- *Abstracting*
  - Cogitating (reflecting)
  - Comparing
  - Relating
  - Trusting
- *Collecting*
  - Acquiring
    - Digitizing/OCRing
    - Purchasing/Licensing
  - Crawling (focused)
  - Submitting
- *Creating*
  - Authoring
  - Transforming
- *Disseminating*
  - Collaborating
  - Filtering
  - Providing access
  - Publicizing
    - Marketing/Advertising
  - Translating
  - Using Email/Listservs
- *Evaluating/Assessing*
  - Analyzing logs
  - Certifying
  - Doing action research
  - Doing ethnographic/sociological study
  - Leading focus groups
  - Logging
  - Measuring
  - Surveying
  - Reviewing (peer)
- *Modeling*
  - Extracting
  - Linking
- *Organizing*
  - Analyzing
  - Annotating
    - Narrating
    - Rating
  - Cataloguing
  - Classifying
    - Training (classifier)
  - Clustering
  - Indexing
- *Personalizing*
  - Customizing
  - Recommending
- *Preserving/Archiving*
  - Conserving
  - Converting
  - Copying/Replicating
  - Emulating
  - Renewing
  - Translating
- *Requesting*
- *Selecting*
  - Binding
  - Browsing
  - Expanding (query)
  - Federating
  - Harvesting
  - Navigating
  - Searching
  - Visualizing

Components
- *Documents*
  - Books, digital
  - Courseware
  - Database
  - Diagrams
  - E-Artifacts
  - Hypermedia documents
  - Hypertext documents
  - Maps, digital
  - Multimedia documents
  - Music, digital
  - Photographs, digital
  - Semi-structured documents
  - Speech, digital
  - Structured documents
  - Versions
  - Video, digital
- *Handles*
  - DOIs
  - URI/URLs, PURLs, URNs
- *Knowledge Organization Sources*
  - Catalogs
    - Attributes
    - Metadata records
  - Relationship groups
    - Ontologies
    - Semantic networks
    - Thesauri
  - Term lists
    - Authority files
    - Classification schemes
    - Classification/Categorization tools
    - Dictionaries
    - Gazetteers
    - Glossaries
    - Subject headings
    - Taxonomies
- *Repositories*
  - Collections
    - Annotations
    - Data
    - Facts
    - Information
    - Knowledge
    - Logs
    - Relationships
      - Hyperlinks
    - User profiles
- *Substrate*
  - Communications
    - Internet
    - Networks
    - Protocols
    - Web
  - Modules
    - Distributed systems
    - File systems
    - Information retrieval systems
    - Multimedia systems
    - Object databases
    - Operating systems
    - Relational databases
    - Servers (e.g., audio, video)
    - User interfaces

Socio-economic/Legal Issues
- *Qualities*
  - Accessibility
  - Interoperability
  - Maintainability
  - Scalability
  - Sustainability
  - Usability
- *Policies*
  - Billing/Charging
    - Pay-per-view
  - Copyright clearances
  - Intellectual property
  - Multilingual access
  - Privacy
  - Rights management
  - Security
    - Authentication
    - Authorization
  - Special-needs access
  - Subscriptions
- *Standards*
  - Access
  - Description
  - Operation
  - Storage
  - Transmission

Environment
- *Academic Disciplines*
  - Anthropology
  - Archaeology
  - Business
    - Copyright law
    - Economics
    - Management
    - Marketing
    - Publishing
  - Computer science
    - Databases
    - Graphics
    - HCI
    - Hypertext
    - Information retrieval
    - Multimedia
    - Software engineering
  - Economics
  - Education
  - Engineering
  - Ethnography
  - Humanities
    - Museum science
  - Information science
  - Law
  - Library science
  - Philosophy
  - Sociology
- *Purposes*
  - Commercial service
  - Education support
  - National library
  - Public services
  - Training
- *Scope*
  - Company
  - Department
  - Discipline
  - Group
  - National
  - Personal
  - State
  - Worldwide

Figure 2.1: Taxonomy of digital libraries terms

knowledge organization sources are normally applied to collections of digital objects and support a number of services such as metadata-based resource discovery, query expansion with thesauri, hierarchical browsing with classification systems, and ontology-based crosswalks among disparate metadata formats and vocabularies. Finally, DLs are served by a substrate—a foundational complex amalgamation of different combinations of Ss that involves computers, network connections, file and operating systems, user interfaces, communication links, and protocols.

Socio-economic, Legal Aspects: What surrounds the DL? This facet is mainly related to the societal aspects of the DL and their relationships and interactions, including regulations, measures, and derivatives. It abstracts aspects surrounding the other DL issues and involves policies, economic issues, standards, and qualities. For example, policies may dictate that only certain communities have the right to use specific portions of a collection. Some of these DL issues can be established regarding normative structured documents. Policies and quality control also can be enforced by specific services, for example, authentication, authorization [82], encryption, and specific practices (scenarios) or protocols, which can involve other communication services and serialized streams.

Environment: In what contexts are DLs embedded? The environment involves a set of spaces (e.g., the physical space, or a concept space defined by the words of a natural language) that defines the use and the context of a DL. The environment also involves the society that sets up the DL and uses it. But the environment is also how the DL fits into the structure of community and its organization and dictates the scenarios by which its activities are performed. Those who pursue *Academic Disciplines* define a problem area "per se" and build a rational consensus of ideas and information about the problem that leads to a solution [182]. Thus they carve out a space for their approaches (e.g., in terms of concepts in a domain language, etc.), and structure some subject knowledge jointly with specific scenarios that define the methods or activities used to solve their specific problems. *Purposes* and *Scope* define types of societies served by the DL and determine a specific library structure.

2.2.2 DL Case Studies with 5S

In the last section, 5S was used to provide a better understanding of the DL field as a whole. The goals of this and the next section are threefold: 1) to show the use of 5S as an analytical tool that facilitates comprehension of specific DL phenomena; 2) to present the complex interplays that occur among 5S components and DL concepts in real DL applications; and 3) to illustrate the possibility of using 5S as an instrument for requirements analysis in DL development.

Case Study 1: Networked Digital Library of Theses and Dissertations (NDLTD)

The Networked Digital Library of Theses and Dissertations (NDLTD) [144, 68, 200, 201] is an international federation of universities, libraries, and other supporting institutions focused on efforts related to electronic theses and dissertations (ETDs). Many libraries and universities run their own programs and services, but consortial activities at the state (e.g., OhioLINK), regional (e.g., Catalunya, Spain), and national (e.g., Australia, Brazil, China, Germany, India, Korea, Portugal) levels exist. NDLTD allows institutions to cooperate and collaborate in a federated fashion, in a scalable and sustainable effort, especially since automation affords savings to both students and their universities relative to old paper-based approaches. As the distributed collection grows, and ultimately achieves critical mass, NDLTD has the potential to become one of the largest and most active digital libraries supporting education and research.

Societies The primary community addressed through the NDLTD society is graduate students. The project aims to enhance graduate education, particularly of those students who prepare either a thesis or dissertation. Consequently, a second community is implicated, namely those involved in administering graduate programs. Those who are deans or associate deans of graduate schools, and their supervisors (e.g., associate provosts or associate chancellors) and staff, as well as the members of related associations (e.g., Council of Graduate Schools in USA, or the Canadian Association of Graduate Schools), are key members of this important community, that often decides if a university will join NDLTD. Because some universities have distributed these responsibilities to colleges or faculties, or because some involved in graduate program administration are too busy to carefully study NDLTD, we expanded this second community to include those in colleges or departments that administer graduate programs, allowing them to have their respective units join NDLTD prior to an action by the entire university. The third community related to the NDLTD society includes those involved in related activities in university libraries. This often involves the director or dean of the university library, as well as those involved in automation, support of multimedia development, training, cataloging, preservation, or other similar roles.

A fourth community involved in NDLTD is that of faculty. They may encourage students to start early to experiment with electronic theses and dissertations (ETDs), and to prepare expressive works, using multimedia. They may assist by providing tools in their laboratories that help with production of an ETD. They may guide students to produce high-quality works, that, in turn, may encourage and help large numbers of interested readers. Faculty also assist students to grasp key issues regarding intellectual property and copyright, and to make their research results available to the widest community of readers possible given constraints relating to patents or publishers.

A fifth, whose importance to the project became obvious early in 1997, is that of publishers. Though NDLTD was developed as a university effort, there is linkage with scholarly publishers because thesis and dissertation work often relates to other writings involving those students, such as conference papers, journal articles, and monographs. Because of copyright laws and publisher policies, that may force editors to make judgements regarding prior publication, this important community must be considered. In cases like ACM, IEEE-CS, and Elsevier, there is strong support by way of policies encouraging ETDs, which has been highly beneficial.

Scenarios Each of the communities involved in the NDLTD society needs particular services from the digital library. They engage in various tasks and activities related to ETDs - each with corresponding scenarios. The NDLTD team has focused on training (through workshops, online materials, and help in media centers or library sites) to assist students with the authoring or creation of ETDs. Next, there is the process of submission, supported by workflow software to help students enter and edit the metadata about their ETDs. Staff in the graduate school and library also use other parts of the workflow software as they check, approve, catalog, and archive new ETDs. Library staff ensure that new works are added to the collection, and that the system affords access almost all the time. In terms of volume, the most active scenarios relate to use of the DL. First, there are simple (running) and advanced (prototype) interfaces that support accessing individual university sites (searching or browsing), federated search across multiple sites, and access to a union archive collection through ODL components [199] and Virtua [215] systems. There is experimental software to add annotation capabilities (the service selected as most important to add, based on focus groups to determine what other scenarios apply) [136]. There is also experimental software, extending SIFT (Stanford Information Filtering Tool) [231] to provide routing services based on stored user profiles, for those who wish to be notified whenever an interesting ETD arrives. As time proceeds, our work in interoperability with other digital library software like Greenstone [230] and Phronesis [80], or institutional repositories like DSpcae (www.dspace.org), may allow us to support other universities that choose to use those packages to provide access services for their local ETDs.

Spaces One space-related aspect of NDLTD is the physical location of members (a metric space) — now spread over parts of Africa, Asia, Australia, and Europe, as well as North, Central, and South America. The Internet provides the name space of machines, while the WWW provides the name space of servers. Vocabulary used in different NDLTD services relates to the conceptual space used in indexing. This will become more disciplined, as members use some version of MARC, Dublin Core, or the ETD-MS thesis and dissertation metadata standard [9], which is likely to provide the basic conceptual space for accessing the NDLTD collection. In addition, manual, semi-automatic, and automatic indexing and classification methods can be applied to place ETDs into conceptual spaces that relate to the Library of Congress or Dewey classifications, as well as discipline-specific thesauri (e.g., ACM's category system for computing) [95]. Another major space-related aspect of NDLTD deals with user interfaces. There are multiple graphical user interfaces that relate to our various software routines, including the ENVISION interface [102] and other visualization or personalization prototypes [157]. In addition, we have investigated how the library metaphor applies to using our CAVE (virtual reality environment) [148].

Streams NDLTD deals with a variety of streams. At the simplest level are streams of characters for text, and streams of pixels for images. Some students have included audio files, or digital video, with their ETDs. These present challenges regarding quality of service if played back in real time, or alternatively storage problems if downloaded and then played back from a local system. On the one hand, using standards like MPEG will make it easier to prolong the useful life of multimedia-rich ETDs, but on the other hand the representations that allow streaming of audio and video tend to be proprietary. This suggests that students probably should store both types of representation. The other class of streams related to NDLTD is that of network protocols. Those involve transmissions of serialized streams over the network. Federated search, harvesting, and hybrid services, using a number of protocols, like Dienst, Z39.50, the Harvest system, and the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), have been developed in the context of NDLTD [92].

Structures Structure plays many roles in NDLTD. A database management system is at the heart of the software for submission and workflow management developed at Virginia Tech. XML and SGML are ways to describe the structure of metadata, or of ETDs themselves. While only a small number of submissions at Virginia Tech have used such markup approaches, larger numbers are being collected in other locations, such as Germany. Moreover, NDLTD has developed and is promoting the Interoperability Metadata Standard for Electronic Theses and Dissertations (ETD-MS) as a standard descriptive metadata scheme for describing electronic theses and dissertations [9]. Structures in the form of *semantic networks* are used inside MARIAN to represent ETD collections and metadata and are explored through the services provided.

Case Study 2: Open Archives Initiative

The Open Archives Initiative (OAI) [121] is not a digital library by itself but a multi-institutional project to address interoperability of archives and digital libraries by defining simple protocols for the exchange of metadata. The current OAI technical infrastructure is defined by the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [150], which defines mechanisms for archives to expose and export their metadata. In the following, this technical infrastructure is analyzed from the 5S point of view.

Societies The main communities designed for the OAI society are electronic, namely active agents called harvesters and repositories, which interact through OAI-PMH. The other two kinds of communities emphasized by the initiative are the so-called *data providers* and *service providers*. The former may be the manager of an archive, acting on behalf of the authors submitting documents to the archive. The latter is a third party, creating end-user services based on data harvested from archives. Ultimately, we have those communities constituted by the final users of the services (including in some cases those engaged in self-archiving) and those involved with administrative aspects of repositories/archives.

Streams The main streams associated with the OAI are dynamic and include communications between harvester agents and the repository server. Those communications are organized as *requests* from the agent to the server, which occur through specific verbs (see Scenarios below) embedded in HTTP requests, and *responses* that are textual metadata, which must be encoded and serialized in XML streams. The Open Archives Initiative so far has not considered multimedia streams, except when they are encoded in XML as part of the metadata.

Structures Major structures of OAI are involved with *records*, *sets*, and *metadata formats*. OAI records can be considered containers [120], which encapsulate several kinds of descriptive metadata. Thus, OAI records obey a structure organized into:

- *Header*, which corresponds to information that is common to all records and includes a unique identifier and a datestamp – the date of creation, deletion, or latest date of modification of an item, the effect of which is a change in the metadata of a record disseminated from that item.
- A single manifestation of the metadata from an item. The OAI protocol supports multiple manifestations (structures) of metadata for any single item. At a minimum, repositories must be able to return records with metadata expressed in the Dublin Core format, without any qualification. Optionally, a repository also may be capable of disseminating other formats of metadata.
- *About*, an optional container to hold data about the metadata record itself, as opposed to the digital object associated with the metadata. Typically, this container is used to hold rights information regarding the metadata record, terms and conditions for usage, etc.

Sets are optional hierarchical structures for grouping items in a repository for the purpose of selective harvesting of records. Membership of records in *sets* is not mandatory, but *sets* can share common records.

Registries, with data about various OAI-compliant repositories, also are provided. This allows users or harvesters or service providers to find suitable collections.

Scenarios Regarding OAI repositories and the harvesting protocol, there is a fixed set of scenarios, namely those involved with requests and responses in the protocol conversations between harvesters and OAI archives. In a 5S analysis, we can associate each request-response pair with a scenario, involving an interaction between harvester and repository. Thus, in the OAI harvesting protocol there are scenarios for retrieving the identifiers of records in the repository restricted to specific sets (ListIdentifiers verb); to retrieve a particular record given an identifier and metadata format (GetRecord verb); to retrieve information about the repository, including administrative information (Identify verb); and to list all supported metadata formats, records and sets in the repository (respectively, ListMetadataFormats, ListRecords, and ListSets verbs)

Another extremely important set of services, which is not part of the OAI technical specifications itself, but is essential to its functionality, is provided by mediation middleware. This layer, which is placed between the repository and the OAI protocol itself, provides vertical communications, conversions, and translations from the OAI verbs and metadata organization to specific internal queries and operations on the underlying data representations of the repository. For example, if the repository is built upon a relational database, the mediation middleware is responsible for translating OAI requests to corresponding SQL queries.

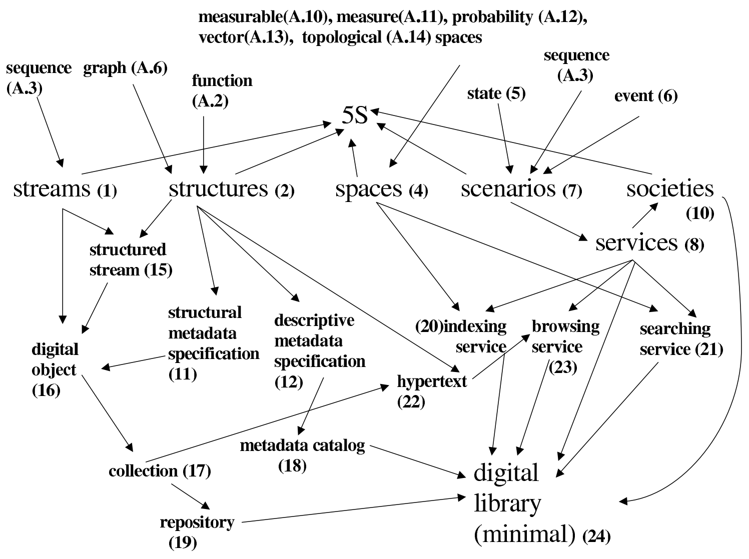


Figure 2.2: 5S map of formal definitions

Spaces The OAI framework is naturally distributed along the physical space. Service providers can build indexing spaces on the top of metadata spaces, a kind of document space, and make use of vector or probabilistic spaces for building services like searching and filtering.

2.3 The 5S Formal Framework

In this section, we precisely and unambiguously formalize most of the informal digital library concepts introduced in previous sections. Figure 2.2 shows a map of the most important concepts and formal definitions. Each concept is associated with the corresponding definition number of its formal definition; arrows indicate that a concept is formally defined in terms of previously defined concepts that point to it [3]. The mathematical preliminaries (Defs. A1–A14) are found in Appendix A.

2.3.1 5S Formalisms

Definition 1 *A **stream** is a $sequence$ whose codomain is a nonempty set.*

Definition 2 *A **structure** is a tuple $(G, L, \mathcal{F})$, where $G = (V, E)$ is a directed graph with vertex set V and edge set E, L is a set of label values, and $\mathcal{F}$ is a labeling function $\mathcal{F} : (V \cup E) \rightarrow L$.*

As a derivative of this definition, the next one follows.

Definition 3 *A **substructure** of a structure $(G, L, \mathcal{F})$ is another structure $(G', L', \mathcal{F}')$ where $G' = (V', E')$ is a subgraph of G, $L' \subseteq L$ and $\mathcal{F}' : (V' \cup E') \rightarrow L'$.*

Definition 4 *A **space** is a measurable space, measure space, probability space, vector space, topological space, or a metric space [4].*

[3]The notion of a tuple (def. A.4) is used in most definitions, so, for simplicity, we are not showing arrows coming out of that concept in Figure 2.2. Other popular definitions are treated likewise.

[4]See Appendix definitions 9-14 for formal definitions of each of these spaces.

Probability studies the possible outcomes of given events (or experiments) together with their relative likelihood and distributions. Probability is defined in terms of a **sample space** S, which is a set whose elements are called **elementary events**. More formally, in terms of a probability space, the set of possible events for an experiment consists of the σ-algebra $\mathbb{B}$ and a sample space is defined as the largest set $S \in \mathbb{B}$. The measure μ is called a probability distribution.

Probabilistic information retrieval (PIR) takes a more subjective interpretation of probability, called the *bayesian* interpretation, which sees probability as a statistical procedure which endeavors to estimate parameters of an underlying probability distribution based on the observed distribution. In PIR the sample space is the set $Q \times D$ of all possible queries and documents and the probability distribution tries to estimate, given a query $q \in Q$ the probability that a document $d \in D$ will be **relevant** to the query, using any evidence at hand. Normally the words in the documents and in the query are the major sources of evidence. A precise definition of probability of relevance is dependent on the definition of relevance and different PIR models have different interpretations [49].

Vector spaces are the basis for a widely used information retrieval model, the Vector Space Model (VSM) [179]. In this model, a document space D is a vector space where a document $d_i \in D$ is represented by a t-dimensional vector $d_i = (w_{i1}, w_{i2}, ..., w_{it})$, w_{ij} being the weight (a numerical value) of the jth index term t_j of d_i, $w_{ij} \geq 0$. An *index term* is normally a word (or variant), occurring in the text of the document, whose semantics helps in defining the document's main themes. However, in general, an index term may be any value describing some aspect of the document, such as a feature value (e.g., color, shape, elevation, temperature) or descriptor (e.g., element in a thesaurus or classification system), or concept, or complex linguistic expression (e.g., phrase, entry in a gazetteer). Furthermore, it is possible to use their representation vectors, i.e., their terms and term weights, to define a number of functions such as *degree of similarity* $s : D \times D \rightarrow \mathbb{R}$ between documents.

Vector spaces and measure spaces are often built on top of topological spaces, the latter being the more basic concept. Any use of the concept of distance implies an underlying **metric space**, which is a topological space whose open sets are defined by $\{y \mid d(x, y) < r\}$, where $d(x, y)$ is the distance between x and y.

Definition 5 *A* ***system state*** *(from now on, just state) is a function $s : L \rightarrow V$, from labels L to values V. A* ***state set*** *S consists of a set of state functions $s : L \rightarrow V$.*

Labels represent a logical *location* associated with some value in a particular state. Thus $s_i(X)$ is the value, or the contents, of location X in state $s_i \in S$. The nature of the values related to contents in a system is granularity-dependent and its definition is out of the scope of this chapter. Normally there are simple values of basic datatypes such as strings and numbers or higher-level DL objects such as digital objects and metadata specifications.

Definition 6 *A* ***transition event*** *(or simply* ***event****) on a state set S is an element $e = (s_i, s_j) \in (S \times S)$ of a binary relation on state set S that signifies the transition from one state to another. An event e is defined by a condition function $c(s_i)$ which evaluates a Boolean function in state s_i and by an action function p.*

This transition event is not a *probabilistic* event [46]. Rather, it is more like the events in networked operating systems theory [192], transitions in finite state machines [53], those modeled by the Unified Modeling Language (UML) [25], or transitions between places in Petri Nets [152].

The condition is used to describe circumstances under which a state transition can take place. An action models a reference to an operation, command, subprogram or method, responsible to perform the actual state transition. Events and actions can have parameters that abstract data items associated with attributes (labels) of a state.

Definition 7 *A* ***scenario*** *is a sequence of related transition events $\langle e_1, e_2, ..., e_n \rangle$ on state set S such that $e_k = (s_k, s_{k+1})$, for $1 \leq k \leq n$.*

We also can interpret a scenario as a path in a directed graph $G = (S, \Sigma_e)$, where vertices correspond to states in the state set S and directed edges are equivalent to events in a set of events Σ_e (and correspond to transitions between states). (Technically, G is a pseudodigraph [5], since loops (s_i, s_i) are possible as events.)

Definition 8 *A* ***service***, ***activity***, ***task***, *or* ***procedure*** *is a set of scenarios.*

Note that the scenarios defining a service can have shared states. Such a set of related scenarios has been called a "scenario view" [103] and a "use case" in the UML [25]. In this framework, a simple transmission service of streams can be formally specified as:

Definition 9 *Let $T = \langle t_1, t_2, ..., t_n \rangle$ be a stream. Let event $e_{t_i} = (s_{t_i}, d_{t_i}$[6]$)$ and event $a_{t_i} = (d_{t_i}, s_{t_{i+1}})$. A transmission of stream T is the scenario (sequence of related events) $e_T = \langle e_{t_1}, a_{t_1}, e_{t_2}, a_{t_2}, ...e_{t_n} \rangle$.*

Scenarios are *implemented* to make a working system and the so-called "specification-implementation" gap must be overcome [172]. Formally, the implementation of scenarios can be mapped to an abstract machine represented by a deterministic finite automaton (DFA). This automaton $M = (Q, \Sigma_e, \delta, q_0, F)$ is such that M is the user-perceived conceptual state machine of the system and accepts a language $L(M)$ over the set of events Σ_e. A grammar $G = (V, \Sigma_e, R, s_0)$ for the language $L(M)$ is such that the non-terminals set V corresponds to the state set S, the terminals are the finite set of events Σ_e, s_0 is a distinguished initial state initializing all locations in that state, and R is a finite set of rules. Each rule in R is of the form $s_i \rightarrow es_j$ and conveys the system from state s_i to s_j as a consequence of event e, or is of the form $s_i \rightarrow e$ when $s_j \in F$ is a final state. The grammar and the corresponding conceptual state machine make up the abstract formal model which the analyst uses to capture, represent, and display system behavior in terms of scenarios. Alternatively, denotational semantics [227] and object-oriented abstractions [171] offer a programming language perspective for the question of formal scenario implementation.

Definition 10 *A* ***society*** *is a tuple (C, R), where*

1. *$C = \{c_1, c_2, ..., c_n\}$ is a set of conceptual communities, each community referring to a set of individuals of the same class or type (e.g., actors, service managers);*

2. *$R = \{r_1, r_2, ..., r_m\}$ is a set of relationships, each relationship being a tuple $r_j = (e_j, i_j)$, where e_j is a Cartesian product $c_{k_1} \times c_{k_2} \times \cdots \times c_{k_{n_j}}$, $1 \leq k_1 < k_2 < \cdots < k_{n_j} \leq n$, which specifies the communities involved in the relationship and i_j is an activity (cf. Def. 8) that describes the interactions or communications among individuals.*

The second part of the definition emphasizes the collaborative nature of societies as in the case of users and service managers engaged in performing DL services. Scenarios describe the service behavior exactly in terms of interactions among the involved societies. For example, an ETD submission service involves interactions between graduate students and an ETD submission workflow manager (an electronic member of a service managers society).

2.4 5S Formal Definition of Digital Library

As pointed out in previous sections, there is no consensual definition of a digital library. This makes the task of formally defining this kind of application and its components extremely difficult. In this section, we approach this problem by constructively defining a "core" or a "minimal" digital library, i.e., the minimal

[5]A digraph which permits both loops and multiple edges between nodes.

[6]d_{t_i} is the state that indicates that the destination has received stream item t_i

set of components that make a digital library, without which, in our view, a system/application cannot be considered a digital library. Each component (e.g., collections, services) is formally defined in terms of an S construct or as combinations or compositions of two or more of them. The set-oriented and functional mathematical formal basis of 5S allows us to precisely define those components as functional compositions or set-based combinations of the formal Ss.

Informally, a digital library involves a managed *collection* of information with associated *services* involving *communities* where information is stored in digital formats and accessible over a network. Information in digital libraries is manifest in terms of *digital objects*, which can contain textual or multimedia content (e.g., images, audio, video), and *metadata*. Although the distinction between data and metadata often depends on the context, metadata commonly appears in a structured way and covering different categories of information *about* a digital object. The most common kind of metadata is *descriptive metadata*, which occurs in catalogs and indexes and includes summary information used to describe objects in a DL. Another common characteristic of digital objects and metadata is the presence of some internal structure, which can be explicitly represented and explored to provide better DL services. Basic services provided by digital libraries are indexing, searching, and browsing. Those services can be tailored to different communities depending on their roles, for example, creators of material, librarians, patrons, etc.

In the following we formally define those concepts of *metadata (structural and descriptive), digital object, collection, catalog, repository, indexing service, searching service, browsing service,* and finally *digital library*.

Definition 11 *A **structural metadata** specification is a structure.*

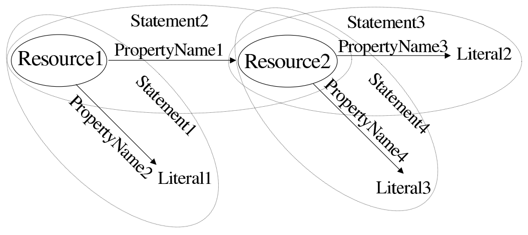
This simple definition emphasizes the role of structural metadata as a representation or abstraction of relationships between digital objects and their component parts (cf. Def. 16). The graph-based representation of this type of metadata can be explicitly expressed, as in the case of markup [45], or implicitly computed [143, 43].

The next definition, for **descriptive metadata specifications**, is inspired by developments in the metadata area, mainly those related to the *Semantic Web* [23] and the Resource Description Framework (RDF) [216], and emphasizes the semantic relationships implied by the labeling function in a structure. Figure 2.3(a) illustrates the basic constructs. Statements, which are triples corresponding to a specific resource (the thing being described) together with a named property about the resource plus the value of that property for that resource, are promoted to first-class concepts. Figure 2.3(b) shows an example of an instantiation of the construct for a descriptive metadata specification about an electronic thesis with four statements: Statement1 = (Thesis1, ‘author’, ‘M.A.Goncalves’), Statement2 = (Thesis1, ‘degree’, Degree1), Statement3 = (Degree1, ‘level’, ‘doctoral’), and Statement4 = (Degree1, ‘grantor’, ‘Virginia Tech’). Below we define the notions of **descriptive metadata specification** and **metadata format** more formally.

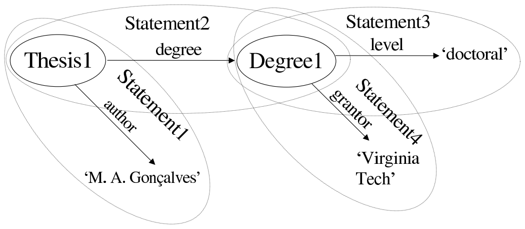
Definition 12 *Let $\mathcal{L} = \bigcup D_k$ be a set of literals defined as the union of domains D_k of simple datatypes (e.g., strings, numbers, dates, etc.). Let also $\mathcal{R}$ and $\mathcal{P}$ represent sets of labels for resources and properties respectively. A **descriptive metadata specification** is a structure $(G, \mathcal{R} \cup \mathcal{L} \cup \mathcal{P}, \mathcal{F})$, where:*

1. *$\mathcal{F} : (V \cup E) \rightarrow (\mathcal{R} \cup \mathcal{L} \cup \mathcal{P})$ can assign general labels $\mathcal{R} \cup \mathcal{P}$ and literals from $\mathcal{L}$ to nodes of the graph structure;*
2. *for each directed edge $e = (v_i, v_j)$ of G, $\mathcal{F}(v_i) \in \mathcal{R} \cup \mathcal{L}$, $\mathcal{F}(v_j) \in \mathcal{R} \cup \mathcal{L}$ and $\mathcal{F}(e) \in \mathcal{P}$;*
3. *$\mathcal{F}(v_k) \in \mathcal{L}$ if and only if node v_k has outdegree 0.*

*The triple $st = (\mathcal{F}(v_i), \mathcal{F}(e), \mathcal{F}(v_j))$ is called a **statement** (derived from the descriptive metadata specification), meaning that the resource labeled $\mathcal{F}(v_i)$ has property $\mathcal{F}(e)$ with value $\mathcal{F}(v_j)$ (which can be designated as another resource or literal).*



(a)



(b)

Figure 2.3: Overview of descriptive metadata with example

Definition 13 *Let $D_{\mathcal{L}_{MF}} = \{D_1, D_2, ..., D_i\}$ be the set of domains that make up a set of literals $\mathcal{L}_{MF} = \bigcup_{j=1}^{i} D_j$. As for metadata specifications, let $\mathcal{R}_{MF}$ and $\mathcal{P}_{MF}$ represent sets of labels for resources and properties, respectively. A* ***metadata format*** *for descriptive metadata specifications is a tuple $MF = (V_{MF}, \mathrm{def}_{MF})$ with $V_{MF} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_k\} \subset 2^{\mathcal{R}_{MF}}$ a family of subsets of the resource labels $\mathcal{R}_{MF}$ and $\mathrm{def}_{MF} : V_{MF} \times \mathcal{P}_{MF} \rightarrow V_{MF} \cup D_{\mathcal{L}_{MF}}$ is a property definition function.*

Therefore a metadata format, through the property definition function, constrains the kinds of resources that can be associated together in statements of a metadata specification as well as the basic datatype domains, which are associated with pairs (resource-property) related to literals [37]. For example, for any set of labels $\mathcal{R}$ for resources, the Dublin Core metadata format defines that $\mathrm{def}_{DC}(\mathcal{R}, \text{`}title') = String$ and $\mathrm{def}_{DC}(\mathcal{R}, \text{`}subject') = SubjectTerms$ where $SubjectTerms$ is a finite set of labels for Resources corresponding to controlled terms. The following definition follows from the previous two definitions:

Definition 14 *A descriptive metadata specification $MS = (G_{MS}, \mathcal{R}_{MS} \cup \mathcal{L}_{MS} \cup \mathcal{P}_{MS}, \mathcal{F}_{MS})$* ***conforms with*** *a metadata format $MF = (V_{MF}, \mathrm{def}_{MF})$ if $\mathcal{R}_{MS} \subseteq \mathcal{R}_{MF}$, $\mathcal{L}_{MS} \subseteq \mathcal{L}_{MF}$, $\mathcal{P}_{MS} \subseteq \mathcal{P}_{MF}$, and for every statement $st = (r, p, l)$ derived from MS, $r \in \mathcal{R}_k$ for some $\mathcal{R}_k \in V_{MF}$ and $p \in \mathcal{P}_{MS}$ implies $l \in \mathrm{def}_{MF}(\mathcal{R}_k, p)$.*

Definition 15 *Given a structure $(G, L, \mathcal{F})$, $G = (V, E)$ and a stream S, a* ***StructuredStream*** *is a function $V \rightarrow (\mathbb{N} \times \mathbb{N})$ that associates each node $v_k \in V$ with a pair of natural numbers (a, b), $a < b$, corresponding to a contiguous subsequence $[S_a, S_b]$ (segment) of the stream S.*

Therefore, a StructuredStream defines a mapping from nodes of a structure to segments of a stream. An example in a textual stream can be seen in Figure 2.4 . From the example, it can be deduced that several structures can be imposed over one stream and vice-versa. Also, it can be seen that segments associated with a node should include the segments of its children (in the case of a hierarchical tree), although it is not equal to the union of those, as "gaps" or "holes" can occur between child segments [143]. Finally, it should be noted that this definition works also for multimedia streams like audio, video, and images.

Definition 16 *A* ***digital object*** *is a tuple $do = (h, SM, ST, StructuredStreams)$ where*

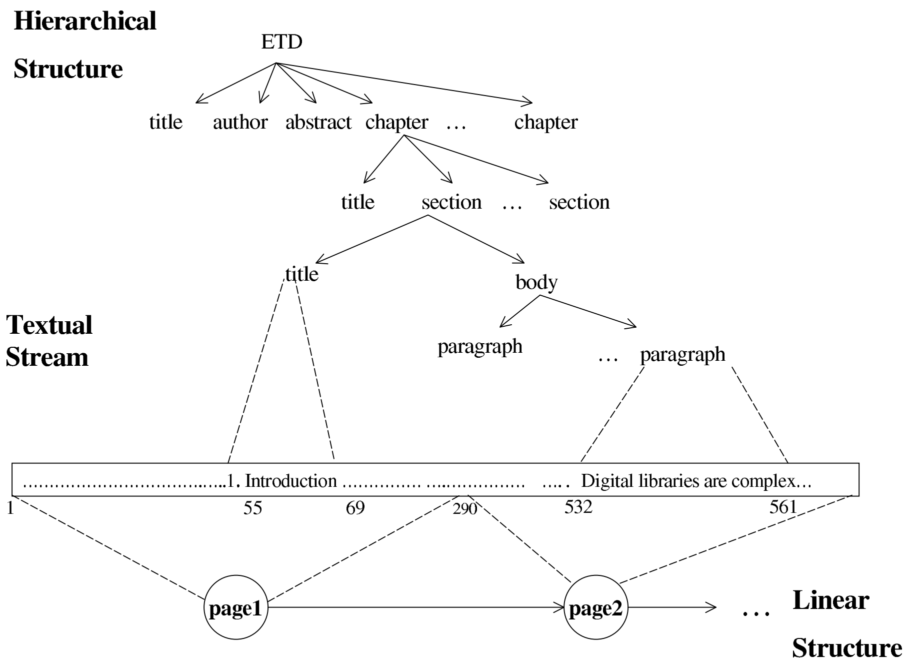


Figure 2.4: A StructuredStream for an ETD (adapted from [Navarro and Baeza-Yates 1997])

1. *$h \in H$, where H is a set of universally unique handles (labels);*
2. *$SM = \{sm_1, sm_2, \ldots, sm_n\}$ is a set of streams;*
3. *$ST = \{st_1, st_2, \ldots, st_m\}$ is a set of structural metadata specifications;*
4. *$StructuredStreams = \{stsm_1, stsm_2, \ldots, stsm_p\}$ is a set of StructuredStream functions defined from the streams in the SM set (the second component) of the digital object and from the structures in the ST set (the third component).*

Figure 2.5 shows an example of a very simple digital object with one structure and several streams. Two important aspects must be pointed out about this formal definition of a digital object:

1. Any real implementation does not need to enforce physical containment of the several component parts of a digital object; for example, we could have pointers to external streams.
2. The definition does not consider active behavior of digital objects [198, 145] which supports operations like different disseminations or exporting of subparts. While there is no explicit restriction regarding this, the definition conforms to our minimalist approach.

Definition 17 *A **collection** $C = \{do_1, do_2, \ldots, do_k\}$ is a set of digital objects.*

Definition 18 *Let C be a collection with k handles in H. A **metadata catalog** DM_C for C is a set of pairs $\{(h, \{dm_1, \ldots, dm_{k_h}\})\}$, where $h \in H$ and the dm_i are descriptive metadata specifications.*

Definition 19 *Let C be a collection with handles H. A **repository** is a tuple $(R, get, store, del)$, where $R \subset 2^C$ is a family of collections and the functions "get", "store," and "del" satisfy:*

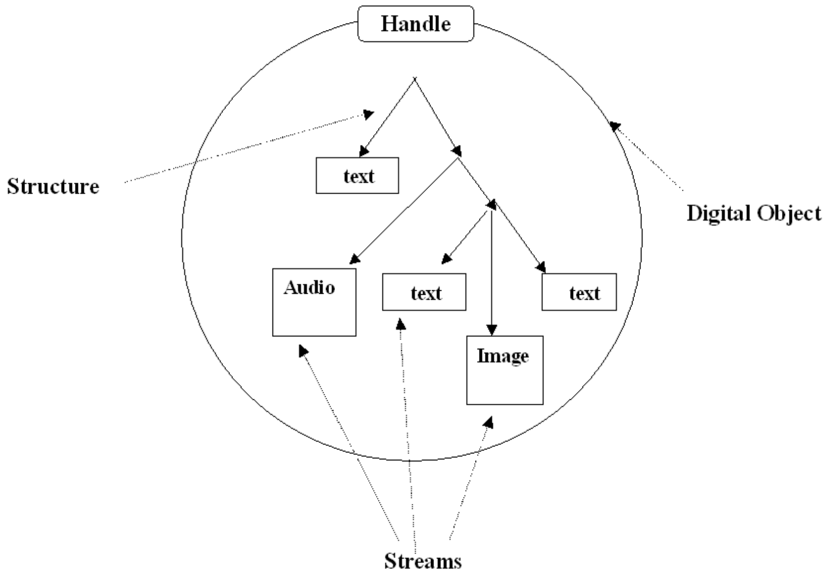


Figure 2.5: A simple digital object

1. *$get : H \rightarrow C$ maps a handle h to a digital object get(h).*
2. *$store : C \times R \rightarrow R$ maps $(do, \tilde{C})$ to the augmented collection $\{do\} \cup \tilde{C}$.*
3. *$del : H \times R \rightarrow R$ maps $(h, \tilde{C})$ to the smaller collection $\tilde{C} - \{get(h)\}$.*

Thus a repository encapsulates a set of collections and specific services to manage and access the collections.

Definition 20 *Let $I : 2^{\mathcal{T}} \rightarrow 2^{H}$ be an index function where $\mathcal{T}$ is a set of indexing features and H is a set of handles. An* ***index*** *is a set of index functions. An* ***indexing service*** *is a single scenario $\{\langle is_1, is_2, ..., is_n \rangle\}$ comprised of pipelined scenarios $is_1, is_2, ..., is_n$ in which the starting state s_{k_0} of the first event of the initial scenario is_1 has a collection $s_{k_0}(K) = C$ and/or a metadata catalog $s_{k_0}(Y) = DM_C$ for collection C as its values and the final state s_{k_f} of the final scenario is_n has an index $I_C = s_{k_f}(Z)$ as its value (K, Y, and Z being labels of the respective states).*

The interpretation of the index and the indexing service is dependent upon the underlying indexing space. Features of an indexing space can be words, phrases, concepts, or multimedia characteristics, like shape or color, appearing or associated with the content of a digital object (in its descriptive and structural metadata or streams). Normally, if a vector space is considered, terms are treated as unrelated, therefore defining orthogonal vectors that span a space $\mathcal{T}$ with dimension m. If a probabilistic space $p = (X, \mathbb{B}, \mu)$ is used, $\mathcal{T} = X$ is the set of distinct terms and is called a *sample space*. Also an index can be thought of as a mapping from an indexing space to a *document (digital object) space* defined by the collection.

The indexing service normally takes the shape of a *pipeline service* where scenarios themselves are executed in sequence and the final state of a scenario is the starting state of the next one. A very simple instance of such an indexing service is shown in Figure 2.6 for indexing of textual material. The indexing service is composed of three scenarios organized as a pipeline of the following scenarios: 1) tokenization, which identifies unique terms inside the textual streams; 2) stopword removal, which filters out terms not useful for retrieval; and 3) stemming, which removes affixes and allows retrieval of syntactic variations

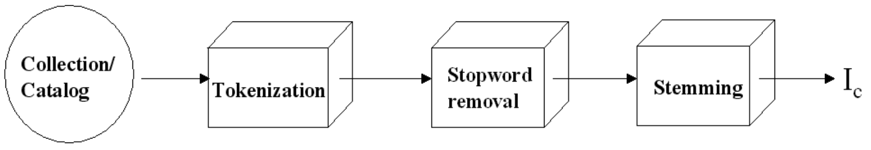


Figure 2.6: Simple indexing service

of query terms [13]. Each one of the scenarios can be thought of as doing some transformation in the representations of digital objects in order to produce the index function. Note again that we are making use of our minimalist approach by not considering complex indexes, for example, defining locations inside streams of digital objects for phrase, proximity, or structural queries.

Definition 21 *Let Q be a set of conceptual representations for user information needs, collectively called queries. Let $M_{I_C} : Q \times (C \times DM_C) \rightarrow \mathbb{R}$ be a matching function, associated with an index I_C, that associates a real number with a query $q \in Q$ and a digital object $do \in C$ and possibly its descriptive metadata specifications $ms \in DM_C$, indicating how well the query representation matches with the digital object, structurally, by content, or regarding the descriptive metadata specifications. A* ***searching service*** *is a set of searching scenarios $\{sc_1, sc_2, \ldots, sc_t\}$, where for each query $q \in Q$ there is a searching scenario $sc_k = \langle e_0, \ldots, e_n \rangle$ such that e_0 is the start event triggered by a query q and event e_n is the final event of returning the matching function values $M_I(q, d)$ for all $d \in C$.*

The components of a digital object do, are denoted by $do(1)$, $do(2)$, etc. Therefore, $do_k(2)$ denotes the second component, i.e., the stream set component of a digital object do_k, $do_k(3)$ its structural metadata set component (third component), and $do_k(4)$ its set of StructuredStreams functions (fourth component). Let also $G[v]$ denote the subgraph of a directed graph G containing node v and all points and edges reachable starting from v. A substructure defined by $G[v]$ inherits the labeling of the structure defined with G. Finally, let $f : A \rightarrow B$ and let $\mathcal{D}$ be any non-empty subset of A. The **restriction** of f to $\mathcal{D}$, denoted by $f|_{\mathcal{D}}$, is a subset of f and is a function from $\mathcal{D}$ to B. Then, for a collection C:

1. $AllStreams$ = $(\cup_{do_k \in C} do_k(2))$ and *AllSubStreams* = $\cup_{sm_t \in AllStreams} \{sm_t[i, j] \mid sm_t = \langle a_0 , a_1, \ldots, a_n \rangle, 0 \leq i \leq j \leq n)\}$ will be the set of all streams and substreams (segments of streams) of all digital objects in the collection C;

2. *AllSubStructuredStreams* = $\bigcup_{k,j} (SubStructuredStream_{k_j})$ where:

   (a) $d_k \in C$;

   (b) $G_{k_j} = (V_{k_j}, E_{k_j})$ is the first component of some structure $st_{k_j} \in d_k(3)$;

   (c) $\mathcal{H}_{k_j} = \{G_{k_j}[v_t] \mid vt \in V_{k_j}\}$ corresponds to the set of all substructures of st_{k_j};

   (d) $SubStructuredStream_{k_j} = \{\mathcal{S}|_{V'} \mid (V', E') \in \mathcal{H}_{k_j}$, $\mathcal{S} \in d_k(4)$ is a StructuredStream function defined from the structure st_{k_j}, and $\mathcal{S}|_{V'}$ is the restrition of $\mathcal{S}$ to $V'\}$.

   Therefore, $AllSubStructuredStreams$ corresponds to the set of all possible substructures and their corresponding connections to streams inside digital objects of the collection.

Definition 22 *Let $H = ((V_H, E_H), L_H, \mathcal{F}_H)$ be a structure and C be a collection. A* ***hypertext*** *$HT = (H, Contents, \mathcal{P})$ is a triple such that:*

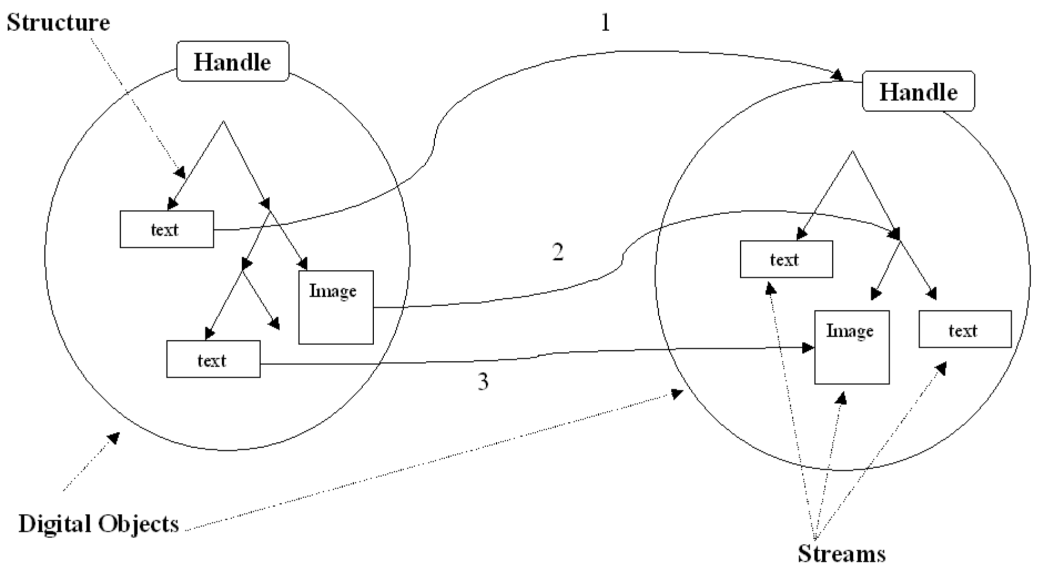


Figure 2.7: A simple hypertext

1. *$Contents \subseteq C \cup AllSubStreams \cup AllSubStructuredStreams$ is a set of contents that can include digital objects of a collection C, all of their streams (and substreams) and all possible restrictions of the StructuredStream functions of digital objects.*

2. *$\mathcal{P} : V_H \rightarrow Contents$ is a function which associates a node of the hypertext with the node content.*

A hyperlink is an edge in the hypertext graph. Source nodes of a hyperlink are called "anchors" and are generally associated via function $\mathcal{P}$ with segments of streams. Also, in this definition, two basic types of hyperlinks can be identified: *structural* and *referential* [222]. Structural hyperlinks allow navigation inside internal structures and across streams of digital objects. Referential hyperlinks usually have their target nodes associated with different digital objects or their subcomponents.

Figure 2.7 illustrates the definition. The hypertext is made by structural hyperlinks that follow the structural metadata and external referential links. Links originate from (segments of) streams. Link targets for, respectively, links 1, 2, and 3, are an entire digital object, a portion of its StructuredStream function (in the figure, represented by the subgraph pointed to by the link and the associated streams) and one of its streams, in this case an image.

An example of such a hypertext is the Web. The Web is a structure where hypertext links connect nodes that can be associated with: 1) complete HTML pages that can be considered digital objects; 2) substructures of a HTML page, for example, a section of the page; and 3) links to streams, e.g., images, audios, or text. The Distributed Graph Storage (DGS) system also implements similar ideas with structural and hyper-structural links representing, respectively, the internal structures of digital objects and hypertext constructs [189]. It should be noted that for the sake of brevity we are not describing links to services, for example, external plugins that can be invoked by browsers or Web forms.

Definition 23 *A **browsing service** is a set of scenarios $\{sc_1, \ldots, sc_n\}$ over a hypertext (meaning that events are defined by edges of the hypertext graph (V_H, E_H)), such that traverse link events e_i are associated with a function $TraverseLink : V_H \times E_H \rightarrow Contents$, which given a node and a link retrieves the content of the target node, i.e., $TraverseLink(v_k, e_{k_i}) = \mathcal{P}(v_t)$ for $e_{k_i} = (v_k, v_t) \in E_H$.*

Therefore, by this definition, every browsing service is associated with an underlying hypertext construct. This view unifies the three modes of browsing defined by Baeza-Yates and Ribeiro-Neto [13]: flat browsing, structured guided, and navigational mode. The third one is the most general case and fits exactly our framework. The first two can be considered special cases. In flat browsing the hypertext has a flat organization, for example, an ordered list of documents or a set of points in an image, and the graph structure of the hypertext corresponds to a disconnected bipartite graph. In the second one, which includes classification hierarchies and directories, the hypertext graph is a tree. Many semi-structured wrapper algorithms disclose this hypertext "hidden" structure in the Web. Once revealed, this structure can be recorded in databases or represented in other semi-structured models to allow queries or transformations. Methodologies like PIPE [162] make use of this information to personalize Web sites. Note also that more sophisticated kinds of hypertext can be defined by extending the current definition. For example, we could relax the function $\mathcal{P}$ to be a relation and associate different contents with the same node, which could be achieved by having different modes of traversing the same link in an extension of the *TraverseLink* function [7]. However, the present definition is simpler and serves well our minimalist approach [8].

Definition 24 *A **digital library** is a 4-tuple* $(\mathcal{R}, Cat, Serv, Soc)$*, where*

- $\mathcal{R}$ *is a repository;*
- $Cat = \{DM_{C_1}, DM_{C_2}, ..., DM_{C_K}\}$ *is a set of metadata catalogs for all collections* $\{C_1, C_2, ..., C_K\}$ *in the repository;*
- *Serv is a set of services containing at least services for indexing, searching, and browsing;*
- *Soc is a society.*

We should stress that the above definition only captures the syntax of a digital library, i.e., what a digital library is. Many semantic constraints and consistency rules regarding the relationships among the DL components (e.g., how the scenarios in $Serv$ should be built from $\mathcal{R}$ and Cat and from the relationships among communities inside the society Soc, or what the consistency rules are among digital objects in collections of $\mathcal{R}$ and metadata records in Cat) are not specified here. Those will be the subject of Chapter 3.

2.5 Example Applications of the 5S Formal Framework: Formal Treatment of Open Archives and the NDLTD Union Archive

2.5.1 Open Archives Initiative

The following formalizes the Open Archives Intitiative Protocol for Metadata Harvesting [121] (OAI-PMH).

Definition 25 *Let* $dl = (\mathcal{R}, Cat, Serv, Soc)$ *be a digital library. The digital library dl can be considered **OAI complaint** if:*

1. *there are two electronic members of the dl society,* $\{dp, hvt\} \subset Soc(1), Soc = dl(4)$*, called the data provider manager and the harvester;*
2. *there is a service* $OAI\_Harvesting \in dl(3) = Serv$ *whose behavior is defined below; and*

[7]This extended approach also generalizes the notion of link directionality where bi-directional links or non-directional links correspond just to different ways of traversing the link (e.g., SOURCE_TO_SINK, SINK_TO_SOURCE, BOTH).

[8]Note also that libraries can support *serendipity* or 'random links'.

3. $(\{dp\} \times \{hvt\}, OAI\_Harvesting) \in Soc(2)$

The data provider manager dp responds to requests of the harvester hvt. Conversations between the harvester and the data provider manager constitute the OAI harvesting service. $OAI\_Harvesting = \{$*Identify*, *ListMetadataFormats*, *ListSets*, *ListIdentifiers*, *ListRecords*, *GetRecord*$\} \in Serv$ is a service formally defined below as 6 scenarios:

1. Identify

   Goal: Returns general information about the archive (what in OAI terms corresponds to the repository $\mathcal{R}$ along with a metadata catalog $DM_{C_k} \in Cat$ for some C_k in the repository.

   Scenario: $\langle e_1 : p =$*identify*, $e_2 : p = response($*identification*$)\rangle$, where e_1 is an event generated by the harvester hvt invoking an operation in dp of dl, e_2 is the event corresponding to the response from the data provider dp, p : specifies the corresponding operation that is being invoked, and *identification* is a parameter of the response operation. The *identification* parameter is a descriptive metadata specification $= (G_{Ident}, \mathcal{R}_{Ident} \cup \mathcal{L}_{Ident} \cup \mathcal{P}_{Ident}, \mathcal{F}_{Ident})$ about the archive, where:

   (a) resource $\mathcal{R}_{Ident} = \{id\}$ is a unique identifier for the archive; and

   (b) properties $\mathcal{P}_{Ident} = \{$repositoryName, baseURL, protocolVersion, earliestDatestamp, deletedRecord, granularity$\}$

2. ListMetaFormats

   Goal: Lists metadata formats supported by the archive as well as their schema location.

   Scenario: $\langle e_1 : p = ListMetadataFormats, e_2 : p = response(metadata\_formats)\rangle$ and $oai\_dc \in metadata\_formats$, meaning the Dublin Core metadata format is mandatory.

3. ListSets

   Goal: Provides a hierarchical listing of sets in which records may be organized.

   Scenario: $\langle e_1 : p =ListSets(resumptionToken), e_2 : p= response($$archive_sets$, $resumptionToken_i$$)\rangle$ and $archive\_sets = \{set_1, set_2, ..., set_k\}$ where each set_i is a 3-tuple $(setSpec_i, setName_i, setDescription_i)$ and:

   (a) $setSpec_i$ is a colon [:] separated sequence of strings $\langle str_{1i} : str_{2i} : ... : str_{ni}\rangle$ indicating the path from the root of the set hierarchy to the respective node. Each string in the sequence must not contain any colons [:]. Since a setSpec forms a unique identifier for the set within the repository, it must be unique for each set. Flat set organizations have only sets with setSpec that do not contain any colons [:].

   (b) $setName_i$ – a short human-readable string naming set_i

   (c) $setDescription_i$ - an set of descriptive metadata specifications about set_i (metadata format not specified; Dublin Core suggested).

   The resumptionToken is a mechanism for flow control when returning an incomplete list of sets. Its exact format is not defined by the protocol. The only defined use of resumptionToken is as follows [150]:

$$\begin{cases} resumptionToken \neq \emptyset, \text{if archives\_sets list is incomplete;} \\ resumptionToken = \emptyset, \text{if archives\_sets list completes a previously received list;} \\ resumptionToken_i = resumptionToken_{i+1}, \text{ where } resumptionToken_{i+1} \\ \text{is the resumptionToken used in the next ListSets request and } resumptionToken_i \\ \text{is the resumptionToken received in the response of the previous request.} \end{cases}$$

4. ListRecords

   Goal: Retrieves metadata for multiple records.

   Scenario: $\langle e_1 : p =$ *ListRecords(from,until,set,metadataPrefix, resumptionToken),* e_2:$p = response$ $(\{oai\_record_1, \ldots, oai\_record_k\}, resumptionToken_i)\rangle$. Each oai_record_i is a 4-tuple $(header_i$, $metadata_i$, $about_i$, $status_i)$ where:

   (a) $header_i$ is a 3-tuple $(record\_id_i, datestamp_i, sets_i)$:

      i. $record\_id_i$ being a unique identifier for the oai_record_i,

      ii. $datestamp_i$, the date/time of creation, modification or deletion of the record for the purpose of selective harvesting;

      iii. $sets_i \subset archive\_sets$, the set membership of the item for the purpose of selective harvesting.

   (b) $metadata_i \in dm_j(2)$ for some $dm_j \in DM_{C_k}$;

   (c) $about_i$ is a descriptive metadata specification about the oai_record_i; metadata format not specified. Common examples of properties include *rights statements* and *provenance* information about the metadata record itself.

   (d) $status_i$ – an optional status attribute with a value of 'deleted' – indicates the withdrawal of availability of the specified metadata format for the item, dependent on the repository support for deletions.

   For every oai_record_i in the response set, the following set of constraints follows:

   (a) $from \leq datestamp_i \leq until$, i.e., datestamp corresponding to the record creation or modification is within the specified date range.

      If omitted the request parameter $from$ takes the value associated with the earliestDatestamp property of *identification* of the archive;

   (b) $set \in sets_i$;

   (c) *metadataPrefix* $\in$ *metadata_formats*; $metadata_i$ **conforms with** the metadata format defined in *metadataPrefix*;

   (d) and $resumptionToken$ fits within the sequence limits related to the flow control implemented by dp as discussed above.

5. ListIdentifiers

   Goal: Lists all unique handles (in OAI terms, identifiers) corresponding to digital objects in the repository.

   Scenario: $\langle e_1 : p =$ *ListIdentifiers(from,until,set,resumptionToken),* $e_2 : p = response(\{ record\_id_i$, ..., $record\_id_l\}, resumptionToken_i)\rangle$, where $\{record\_id_i$, ..., $record\_id_l\}$ is a set of identifiers (or handles) for OAI records $\{oai\_record_i$, ..., $oai\_record_l\}$. The same set of constraints for *ListRecords* applies to the *ListIdentifiers* response.

6. $GetRecord$

   Goal: Returns the metadata for a single identifier in the form of an OAI record.

   Scenario: $\langle e_1 : p =$*GetRecord(id,metadataPrefix)*, $e_2 : p = response(oai\_record_i)\rangle, id = record\_id_i$; other constraints apply as above.

2.5.2 NDLTD Union Archive

- A digital library federation is a set $DLF = \{dl_1, dl_2, ..., dl_f\}$ of independent and possibly heterogeneous digital libraries (DLs). NDLTD is a digital library federation where each independent DL $dl_k = (ETD\_\mathcal{R}_k,\ ETD\_Cat_k,\ ETD\_Serv_k,\ ETD\_Soc_k)$. $ETD\_\mathcal{R}_k$ is a repository having a collection $ETD\_Coll_{jk} = \{etd_{1jk},\ etd_{2jk},\ ...,\ etd_{njk}\}$ composed of a set of digital objects etd_{ijk} corresponding to electronic theses or dissertations (ETDs). The possible set of streams of an ETD, $etd_{ijk}(2)$, is normally limited to a small number of standard types (e.g., Unicode encoding for the character set, MPEG for videos) due to preservation concerns and technological limitations. NDLTD currently does not enforce (yet) any specific structural metadata for ETDs, but several projects for standardizing such a structure with XML Schemas and DTDs are under development in many locations including Finland, Germany, and USA. For each ETD $etd_{ijk} \in ETD\_Coll_{jk}$ there should be at least one $etd\_dm_{ijk} \in ETD\_DM_{jk_{ETD\_Coll_{jk}}}$, $ETD\_DM_{jk_{ETD\_Coll_{jk}}} \in ETD\_Cat_k$, $ETD\_DM_{jk_{ETD\_Coll_{jk}}}$ being a metadata catalog for the ETD collection ETD_Coll_{jk}.

- NDLTD promotes ETD-MS as the metadata format for ETD descriptive metadata specifications. For each dl_k in NDLTD, let:

  – $ETD\_IDs_k = \{h_{ijk} | h_{ijk} = etd_{ijk}(1), etd_{ik} \in ETD\_Coll_{jk}\}$, $NDLTD\_ETD\_IDs = \bigcup_{dl_k \in NDLTD} ETD\_IDs_k$ be the set of the handles of all the ETDs in the NDLTD federation collections;

  – $ETD\_Properties$ = {'title', 'creator', 'person', 'subject', 'description', 'publisher', 'contributor', 'date', 'type', 'format', 'identifier', 'language', 'coverage', 'rights', 'degree'};

  – $Degree = \{dg_1, dg_2, ..., dg_x\}$ a set of unique labels representing the degree portion of an ETD;

  – and $Degree\_Properties$ = {'name', 'level', 'discipline', 'grantor'}, a set of properties about the degree portion of an ETD.

  In formal terms, ETD-MS is a metadata format $(V_{ETD-MS}, def_{ETD-MS})$ for descriptive metadata specifications in ETD-MS $= (G_{ETD}, \mathcal{R}_{ETD} \cup \mathcal{L}_{ETD} \cup \mathcal{P}_{ETD},\ \mathcal{F}_{ETD})$, where resources $\mathcal{R}_{ETD} = (NDLTD\_ETD\_IDs \cup Degree)$, $V_{ETD-MS} =$
  $\{ETD\_IDs,\ Degree\}$, properties $\mathcal{P}_{ETD} = (ETD\_Properties \cup Degree\_Properties)$ and for all triples (r, p, z) (i.e. resource, property, value):

  1. $r \in NDLTD\_ETD\_IDs$ iff $p \in ETD\_Properties$,
  2. $r \in Degree$ iff $p \in Degree\_Properties$, and
  3. $def_{ETD}(NDLTD\_ETD\_IDs, \text{`}degree') = Degree$.

- Society ETD_Soc_k of dl_k is such that {Patron, Student, ETDReviewer, ETDCataloguer, ETDSearchManager, ETDWorkflowManager,...} $\subset ETD\_Soc_k(1)$.

- The NDLTD Union Archive is a tuple $(NDLTD\_Union, UA\_Harvester)$ where $NDLTD\_Union$ $= \bigcup_{dl_k \in NDLTD} ETD\_DM_{jk_{ETD\_Coll_{jk}}}, ETD\_Cat_k = dl_k(2)$,

$ETD\_DM_{jk_{ETD\_Coll_{jk}}} \in ETD\_Cat_k$, is the union of the metadata catalogs for the ETD collections of all NDLTD members and $UA\_Harvester$ is a manager, an electronic member of the NDLTD society, which participates in an OAI harvesting service that periodically harvests metadata records from the NDLTD members.

- Each DL dl_k in the union archive includes a data provider manager, $dp_k \in dl_k(4) = ETD\_Soc_k$, which responds to requests from the NDLTD harvester $UA\_Harvester$. Conversations between the *UA_Harvester* and dp_k are governed by the OAI-PMH and constitute an OAI harvesting service as defined in the previous section.

2.6 Related Work

Formal frameworks, which have supported research and development in most computer science subfields (e.g., programming languages, databases, information retrieval, hypermedia), are surprisingly missing in the digital library literature. One could conjecture that is due to the previously argued complexity of the field. Wang [220] provides one first attempt to fill this gap. His so-called "hybrid approach" defines a digital library as a combination of a special purpose database and a hypermedia-based user interface and builds upon this combination to formalize digital libraries in terms of the language Z [197]. Kalinichenko *et al.* [110] presented a canonical framework for information systems and a compositional approach that they applied to provide a partial solution for interoperability in DLs. Castelli *et al.* [37] have presented the closest work to ours so far. In the context of a multidimensional query language for digital libraries they have formalized the concepts of documents, based on the notions of views and versions, metadata formats and specifications, and a first-order logic based language. These approaches, clearly incomplete, are, as far as we know, the only attempts to provide some formalization for the digital libraries field.

The flexibility of 5S has been further demonstrated as an instrument for requirements analysis in DL development and as a basis for organizing a digital library taxonomy. While research in DL requirements analysis has been underrepresented with only small isolated case studies (e.g., [52, 58, 84, 128]), to the best of our knowledge there is no other comprehensive DL taxonomy published in the literature, other than that presented in [67]. Our taxonomy is an expanded version of that one.

]

Chapter 3

Towards a Digital Library Theory: A Formal Digital Library Ontology

Digital libraries have eluded definitional consensus and lack agreement on common theories and frameworks. This makes comparison of DLs extremely hard, promotes ad-hoc development, and impedes interoperability. In this chapter we propose a formal ontology for digital libraries (DLs) that defines the fundamental concepts, relationships, and axiomatic rules that govern the DL domain, therefore providing a frame of reference for the discussion of essential concepts of DL design and construction. The ontology is an axiomatic, formal treatment of DLs, which distinguishes it from other approaches that informally define a number of architectural variants. The process of construction of the ontology was guided by 5S, a formal framework for digital libraries. To test its expressibility we have used the ontology to create a taxonomy of DL services and reason about issues of reusability, extensibility, and composability.

3.1 Introduction

Research in digital libraries (DLs) has historically been very pragmatic. While much attention has been paid to design and implement systems and architectures [228, 38, 155, 199], create collections and services [149, 41], and improve algorithms and methods [99], very little has been done to understand the underlying fundamental concepts, their relationships, and the axiomatic rules that govern the DL domain, or in other words, to develop a theory of DLs. The necessity of such theory has long being advocated, from the origins of the field, illustrated by Licklider's call for a unified Computer Science (CS)/Library and Information Science (LIS) model [130], to recent workshops on the future of digital libraries [124]. The absence of such a theory makes comparison of different DL architectures and systems extremely hard, promotes ad-hoc development, and impedes interoperability. Its existence might enhance our ability to communicate about and identify new research areas [195].

In Chapter 2, we have presented a partial formal conceptualization of digital libraries by formally defining high-level DL concepts such as digital objects, collections, repositories, services, etc. We proceeded from basic mathematical concepts such as sets, graphs, functions, sequences, and so forth in a bottom-up manner. However, such conceptualization does not constitute a DL theory. A theory should make explicit the implicit relationships that exist among the defined formal DL concepts as well as provide a set of rules or axioms that precisely define and constrain the semantics of concepts and relationships in the theory. This type of formal conceptualization has elsewhere been called an ontology [57]. Ontologies specify relevant concepts – the types of things and their properties – and the semantic relationships that exist between those concepts in a particular domain. Formal specifications use a language with a mathematically well-defined syntax and semantics to describe such concepts, properties, and relationships precisely.

In this chapter, we define a formal, axiomatic ontology for digital libraries (DLs) that can serve as a frame of reference for the discussion of essential concepts of DL design. The process of construction of such an ontology was guided by 5S, a formal framework for digital libraries. We use the resulting ontology to provide answers for questions such as:

- how should DL services be built from the repository, its collections and metadata catalogs, and from the relationships among different societies that participate in the DL?;
- which are the dependencies and consistency rules that should follow in a DL theory?;
- which are the fundamental and elementary DL services and how can services be built/composed from other DL services?

This chapter is organized as follows. Section 3.2 summarizes our earlier results by giving a formal definition of DLs based on the 5S framework. Section 3.3 builds on the core definitions to create an axiomatic, formal ontology for digital libraries. Sections 3.4 illustrates the expressiveness of the ontology by applying it to create a taxonomy of DL services and to reason about issues of minimality, extensibility, and reusability.

3.2 Background: The 5S Framework for Digital Libraries

In this section, we summarize the results of Chapter 2. Accordingly, Let:

- $Streams$ be a set of streams, which are sequences of arbitrary types (e.g., bits, characters, pixels, frames);
- $Structs$ be a set of structures, which are tuples, $(G, L, \mathcal{F})$, where $G = (V, E)$ is a directed graph and $\mathcal{F} : (V \cup E) \rightarrow L$ is a labeling function;
- Sps be a set of spaces each of which can be a measurable, measure, probability, topological, metric, or vector space.
- $Scs = \{sc_1, sc_2, \ldots, sc_d\}$ is a set of scenarios where each $sc_k = \langle e_{1k}(\{p_{1k}\}), e_{2k}(\{p_{2k}\}), \ldots, e_{d_{kk}}(\{p_{d_{kk}}\})\rangle$ is a sequence of events that also can have a number of parameters p_{ik}. Events represent changes in computational states; parameters represent specific locations in a state and respective values.
- St^2 be a set of functions $\Psi : V \times Streams \rightarrow (N \times N)$ that associate nodes of a structure with a pair of natural numbers (a, b) corresponding to a segment of a stream.
- $Coll = \{C_1, C_2, \ldots, C_f\}$ be a set of DL collections where each DL collection $C_k = \{do_{1k}, do_{2k}, \ldots, do_{f_{kk}}\}$ is a set of digital objects. Each digital object $do_k = (h_k, Stm_{1k}, Stt_{2k}, \Omega_k)$ is a tuple where $Stm_{1k} \subseteq Streams$, $Stt_{2k} \subseteq Structs$, $\Omega_k \subseteq St^2$, and h_k is a handle which represents a unique identifier for the object.
- $Cat = \{DM_{C_1}, DM_{C_2}, \ldots, DM_{C_f}\}$ be a set of metadata catalogs for $Coll$ where each metadata catalog $DM_{C_k} = \{(h, mss_{hk})\}$, and $mss_{hk} = \{ms_{hk1}, ms_{hk2}, \ldots, ms_{hkn_{hk}}\}$ is a set of descriptive metadata specifications. Each descriptive metadata specification ms_{hki} is a structure with atomic values (e.g., numbers, dates, strings) associated with nodes.
- A repository $\mathcal{R} = \{C_i\}(i = 1 \text{ to } f)$ be a set of collections; it is assumed there exist operations to manipulate them (e.g., get, store, delete) (see Def. 19.) .

- $Serv = \{Se_1, Se_2, \ldots, Se_s\}$ be a set of services where each service $Se_k = \{sc_{1k}, \ldots, sc_{s_{kk}}\}$ is described by a set of related scenarios.

- $Soc = (C, R)$ where C be a set of communities and R is a set of relationships among communities $SM = \{sm_1, sm_2, \ldots, sm_j\}$, and $Ac = \{ac_1, ac_2, \ldots, ac_r\}$ are two such communities where the former is a set of service managers responsible for running DL services and the latter is a set of actors that use those services [1]. Being basically an electronic entity, a member sm_k of SM distinguishes itself from actors by defining or implementing a set of operations $\{op_{1k}, op_{2k}, \ldots, op_{nk}\} \subset sm_k$. Each operation op_{ik} of sm_k is characterized by a triple $(n_{ik}, sig_{ik}, imp_{ik})$, where n_{ik} is the operation's name, sig_{ik} is the operation's signature (which includes the operation's input parameters and output), and imp_{ik} is the operation's implementation. These operations define the capabilities of a service manager sm_k. For example, SearchManager $\supset$ match(q:query, C:collection)[2] indicates that a SearchManager defines an operation "match" with two parameters, a query and a collection.

According to the 5S formal framework a digital library is a 4-tuple *($\mathcal{R}$, Cat, Serv, Soc)*.

The above definition emphasizes syntactic aspects, i.e., how digital library concepts are composed or built from previously defined concepts. In the next section, we will explore semantic relations and rules of the DL domain.

3.3 Defining a DL Theory Through an Ontological Analysis of the 5S Framework

The crux of our contribution with the 5S framework was, departing from abstractions of many DL architectural settings, recognizing and formally defining the essential participating concepts in the digital library discourse. In this section, we extend those results to define a DL ontology by specifying the fundamental collaborations or relations that exist among the DL participants and the sets of rules (or axioms) which constrain the semantics of concepts and relations in the ontology.

We organize the presentation and development of the ontology according to 5S. For each S, we list the concepts and the relations in which they take part. We consider first intra-S relations, i.e., the relations that occur only among concepts of the same S, along with the corresponding axioms or rules. Afterwards, relations defined between concepts belonging to different Ss are defined representing inter-dependencies. It should be noticed in the discussion below that some concepts such as digital objects and indexes are inherently "cross-S" concepts, i.e., they are defined in terms of concepts belonging to more than one S. For presentation purposes, we will include those "cross-S" concepts within the discussion about the 'S' in which they share most of their relationships.

More formally, a domain is a set of objects of the same DL type. A DL type is characterized by a definition as in Chapter 2. An object is of a type X if its properties (e.g., internal components, organization) satisfy the definition of X. Examples of DL types include the basic Ss and derivative types such as collections, digital objects, etc. An ontological concept is a domain. For example, the statement $x \in$ **Digital Object** says that x is a digital object as defined in Chapter 2 and therefore describes x by the ontological concept Digital Object. An n-ary relation is a subset of the Cartesian product $C_1 \times C_2 \cdots \times C_n$ of the domains defined by the respective DL concepts. Let $R \subset A \times B$ be a relation. Then $R^{-1} = \{(b, a) | (a, b) \in R\} \subset B \times A$ is called the *inverse relation* of R. A predicate is a function from a Cartesian product to the Boolean values

[1] It is worthwhile to remind the reader that in this dissertation we will focus only on the relationships between actors and services managers, which correspond to interactions mediated by the DL. We will not focus on interactions which happen outside of the system.

[2] To simplify notation, we will represent an operation $op_x = (n_x, sig_x, imp_x)$ by $n_x(\{p_{xk}\})$ where $\{p_{xk}\}$ is the set of input parameters of op_x. The output parameters and implementation can be added when a fuller description of the operation is required.

true or **false**. A predicate $p(x)$ built over a relation among concepts is true if x is a member of the relation, false otherwise. We now proceed to define our meaning of a DL ontology.

Definition 26 *An ontology is a tuple (Ontol_Concepts, Ontol_Rels) where:*

1. *Ontol_Concepts is a family of ontological concepts,*
2. *Ontol_Rels is a family of relations.*

For notational purposes we will use **bold** to designate ontological concepts (or simply, concepts) and italics to define the corresponding predicate. We will use the *dot* "." notation to denote components of the definition of concepts, for example *x.h* specifies the handle of a digital object, *y.Img* specifies the image (or range) of events of scenario y, and *z.op* specifies the set of operations of Service Manager z. We also may refer to a component of a tuple-oriented concept by its position in the tuple, for example, z(2) specifies the set of descriptive metadata specifications of a member of a catalog. Finally, we will represent a relation $R \subset A \times B$ by A ***R*** B. The notation for 3-tuple relations will use similar variants, depending on the semantics of the relation.

Below we proceed to define the relations and rules of our DL ontology. The relations were developed by carefully analyzing all possible pairs of associations among concepts within the same and between Ss, and contextual information necessary to define some of these relations.

3.3.1 Intra-S Relationships

Streams

- Concepts: {**text**, **image**, **video**, **audio**}
- Relations:
  - contains $\subset$ **video** $\times$ **image** $\cup$ **video** $\times$ **audio**
    Streams define the basic content types over which digital objects are built, the latter being the ultimate carriers of the information in the DL. However some complex types of streams (e.g., video) may themselves be associated with simpler types of streams (e.g., images, audio). This relation indicates that a video contains a image as one of its frames, or contains a specific audio recording.

Structures

- Concepts: {**do**, **ms**, **mss**, **C**, DM_C, $\mathcal{R}$}. Key: do = digital object; ms = descriptive metadata specification; mss = set of descriptive metadata specifications; C = collection, DM_C = metadata catalog for collection C, $\mathcal{R}$ = repository.
- Relations:
  - is_version_of $\subset$ **do** $\times$ **do**
    Different manifestations of a digital object are versions, which normally differ structurally or in terms of their content (e.g., format, encoding, etc.). This relation indicates that a digital object is a version of another digital object. A digital object x is a slightly different version of digital object y in terms of their streams or structures. Note also that since handles are used as identifiers of digital objects they should be globally unique, so no two digital objects, version or not, share the same handle.

Rules. 1. Digital object handles are unique. 2. x is_version_of y for two digital objects x and y if they differ in the handle component and at least one other component, but share at least one other of their components (e.g., they have the same set of streams, set of structures, or set of structured_streams).

Symbolic rules. 1. $\forall x, y(do(x) \wedge do(y) \wedge (x.h_x = y.h_y) \Rightarrow x = y))$; 2. $\forall x, y\ (x$ *is_version_of* $y \iff do(x) \wedge do(y) \wedge (x.h \neq y.h) \wedge ((x.Stt \neq y.Stt) \vee (x.Stm \neq y.Stm) \vee (x.\Omega \neq y.\Omega)) \wedge ((x.Stt = y.Stt) \vee (x.Stm = y.Stm) \vee (x.\Omega = y.\Omega)))$.

- cites/links_to $\subset$ **do** $\times$ **do**

  Digital objects commonly contains references to other related digital objects in terms of citations or links. While there are discussions if citations and links have the same semantic meaning [30, 194] we will treat them likewise, since they ultimately connect documents together.

- belongs_to $\subset$ **ms** $\times$ DM_C $\cup$ mss $\times$ DM_C

  Digital objects can belong to many different collections. Similarly, descriptive metadata specifications can belong to many catalogs.

  Rule. x belongs_to y indicates that a metadata specification x is used to define an element of the metadata catalog y.

  Symbolic Rule. $\forall x, y(x$ *belongs_to* $y \iff (ms(x) \wedge DM_C(y) \wedge (\exists z \in y : x \in z(2))) \vee (mss(x) \wedge DM_C(y) \wedge (\exists z \in y : x = z(2)))$

- part_of $\subset$ **C** $\times$ **C** $\cup$ $DM_C \times DM_C$

  Many DL collections and metadata catalogs are built by aggregating smaller subcollections / subcatalogs. One good example is the National Science Digital Library (NSDL) union catalog which is basically an amalgamation of the metadata catalogs of all the participant projects.

  Rule. x part_of y indicates that collection x is a subset of collection y or metadata catalog x is a subset of metadata catalog y.

  Symbolic Rule. $\forall x, y(x$ *part_of* $y \iff ((C(x) \wedge C(y) \wedge x \subseteq y) \vee (DM_C(x) \wedge DM_C(y) \wedge x \subseteq y)))$

- describes $\subset$ **mss** $\times$ **do** $\cup$ $DM_C \times$ **C**;

  A digital object may potentially have many descriptive metadata specifications, for example, in standard formats (e.g., Dublin Core, MARC) for sharing purposes, or based on more detailed, community-oriented specific formats. Also qualitative properties of metadata catalogs such as completeness and consistency can be defined in terms of this relationship.

  Rules. 1. *x describes y* indicates that a set of descriptive metadata specifications x, belonging to some catalog q for collection p, describes the content of a digital object y, which belongs to that collection p. The set of metadata specifications x can describe only one digital object, therefore the describes relation between sets of metadata specifications and digital objects is a *function*.

  Symbolic rules. 1.1 $\forall x, y(x$ *describes* $y \wedge mss(x) \wedge do(y) \Rightarrow \exists p, q, h : C(p) \wedge DM_C(q) \wedge ((h, x) \in q) \wedge (y \in p) \wedge (y(1) = h))$;

  1.2 $\forall x, y, z(x$ *describes* $y \wedge x$ *describes* $z \wedge mss(x) \wedge do(y) \wedge do(z) \Rightarrow y = z)$

  Rules. 2. The relation q *describes* p, $(q, p) \in DM_C \times C$ indicates that a metadata catalog q describes a specific collection p. A complete catalog has at least one set of metadata specifications for each digital object in the collection it describes. In a consistent catalog, each set of metadata specifications describes (exactly) one digital object in the related collection. In other words, a complete *describes* relationship between a metadata catalog, and a collection defines a surjective partial function, and a consistent relationship defines a total function. Also note that it is very common that different metadata specifications (e.g., a Dublin Core and a MARC version) may describe the same digital object, so in most cases the *describes* function is not injective.

Symbolic Rules. 2.1 Catalog/Collection Consistency: $\forall x, y, z(C(y) \wedge DM_C(x) \wedge mss(z) \wedge x$ ***describes*** $y \wedge z$ ***belongs_to*** $x \Rightarrow \exists p \in y : z$ ***describes*** $p)$;
2.2 Catalog/Collection Completeness: $\forall x, y, z(C(y) \wedge DM_C(x) \wedge do(z) \wedge x$ ***describes*** $y \wedge z \in y \Rightarrow \exists m : (mss(m) \wedge m$ ***belongs_to*** $x \wedge m$ ***describes*** $z))$.

  – stores $\subset \mathcal{R} \times$ **C** $\times DM_C$
  Captures the fact that a physical linkage exists between a collection and the metadata catalog that describes it in the context of the collection's repository. In many DL systems, digital objects are physically stored with their metadatata specifications or with physical pointers to them (e.g., foreign keys, memory address). In our minimal DL, we were flexible enough to allow a catalog to exist outside of scope of the collection's repository, but this relationships make the physical connection explict.
  Rule. r stores (x,y) indicates that a repository r stores a collection x with the metadata catalog y which describes x.
  Symbolic Rule. $\forall x, y, z(x$ ***stores*** $(y, z) \Rightarrow \mathcal{R}(x) \wedge C(y) \wedge DM_C(z) \wedge z$ ***describes*** $y)$

Spaces

- Concepts: $\{$**Vec**, **Pr**, **Measurable**, **Measure**, **Metric**, **Top**$\}$. Key: Vec= vector space; Pr = probability space; Measurable = measurable space; Measure = measure space; Metric = metric space; Top = topological space.
- Relations
  – is_a $\subset$ **Measure** $\times$ **Measurable** $\cup$ **Pr** $\times$ **Measure** $\cup$ **Metric** $\times$ **Top** $\cup$ **Vec** $\times$ **Top**.
  x *is_a* y indicates that a space x has all the properties / constraints / operations associated with the definition of the space y and may include additional properties / constraints / operations. The *is_a* relationship is reflexive, transitive, and anti-symmetric, therefore mathematical spaces that participate in this relation define a partial order.

Scenarios

- Concepts: $\{$**Se**, **Sc**, **e**$\}$; Key: Se = service; Sc = scenario; e = event.
- Relations:
  – contains $\subset$ **Sc** $\times$ **e**
  Makes explicit the relationship that an event belongs to a sequence of some scenario of use of a DL service. Rule. sc_k *contains* e_{k_j} indicates that an event $e_{k_j} = sc_k(j)$ is a element of the image/range of a scenario sc_k, for some j belonging to the domain $\{1, 2, \ldots, d_k\}$ of sc_k. Recall that scenario is a sequence of events, i.e., it is a function from natural numbers to a set of events.
  Symbolic Rule. $\forall x, y(x$ ***contains*** $y \wedge Sc(x) \wedge e(y) \Rightarrow \exists j : (j \in x.Dom \wedge y = x(j)))$
  – precedes $\subset$ **e** $\times$ **e** $\times$ **Sc**; happens_before $\subset$ **e** $\times$ **e** $\times$ **Sc**
  A scenario of use represents a temporal sequence of events that a user (or another service manager) engages in while interacting with a DL service. The temporal ordering of events is captured by these relations.
  Rule 1. x $precedes_z$ y indicates that an event x occurs immediately before y in the context of scenario z. x $happens\_before_z$ y indicates that both x and y are elements of sequence z, and x

happens some time before y, i.e., the sequence value of x is smaller than the sequence value of y.

Symbolic Rule 1. $\forall x, y, z(x\ precedes_z\ y \wedge e(x) \wedge e(y) \wedge Sc(z) \Rightarrow \exists i, j : (z\ \textbf{contains}\ x \wedge z\ \textbf{contains}\ y \wedge x = z(i) \wedge y = z(j) \wedge i + 1 = j))$

Symbolic Rule 2. $\forall x, y, z(x\ happens\_before_z\ y \wedge e(x) \wedge e(y) \wedge Sc(z) \Rightarrow \exists i, j : (z\ \textbf{contains}\ x \wedge z\ \textbf{contains}\ y \wedge x = z(i) \wedge y = z(j) \wedge i < j))$

– reuses $\subset$ **Se** $\times$ **Se** $\cup$ **Sc** $\times$ **Sc**; extends $\subset$ **Se** $\times$ **Se** $\cup$ **Sc** $\times$ **Sc**

Services exposed by a DL can be classified either as elementary or composite. Elementary services provide the basic infrastructure for the DL. Examples include authoring, indexing, rating, and linking. Composite services can be composed of other services (elementary or composed) by reusing or extending them. For example, searching and browsing services use indexing and linking services, a relevance feedback service extends the capabilities of a basic searching service, and a lesson plan building service may use already existing searching, browsing, and binding services to find and organize relevant resources. The problem of composability of services has gained considerable attention recently, mainly in the Web Services community [33, 50]. However, DL services are restricted to certain specific types with constrained inputs and outputs, therefore making the problem more manageable and amenable to domain specific techniques. Since DL services are described by correlated, generally slightly variant scenarios of use, similar notions can be applied to those scenarios. For example, consider scenario $sc_1 = \langle search(q, C), results(\{(do_i, w_i)\})\rangle$ for a search service where q represents a query, C a DL collection, do a digital object, and w a weight. The scenario $sc_2 = \langle search(q, C)$, $results(\{(do_i, w_i)\})$, $relevant\_docs\{do_i\}$, $expanded\_query(eq, \{do_j\})$, $search(eq, C)$, $results(\{(do_k, w_k)\})\rangle$ is an extension of sc_1 representing a relevance feedback search.

Rule 1. Let $sc_1 = \langle e_1, e_2, \ldots, e_n\rangle$ be a scenario. A scenario $sc_2 = \langle e_{2x}, \ldots, e_{2y}\rangle$ *reuses* scenario sc_1 if it contains all events of sc_1 in the same order they appear, i.e., if event e_i precedes event e_j in sc_1, the same relationship holds in scenario sc_2, or, in other words, sc_2 *reuses* sc_1 only if sc_1 is a *consecutive subsequence* of sc_2.

Symbolic Rule 1. $\forall x, y(x\ \textbf{reuses}\ y \wedge Sc(x) \wedge Sc(y) \Rightarrow (\forall z : e(z) \wedge y\ \textbf{contains}\ z \Rightarrow x\ \text{contains}\ z) \wedge (\forall p, q : e(p) \wedge e(q) \wedge p\ precede_y\ q \Rightarrow p\ precedes_x\ q))$

Rule 2. A service Se_1 reuses service Se_2 if it includes all its scenarios, i.e., if $Se_2 \subseteq Se_1$.

Symbolic Rule 2. $\forall x, y(x\ \textbf{reuses}\ y \wedge Se(x) \wedge Se(y) \Rightarrow y \subseteq x)$.

Rule 3. Let $sc_1 = \langle e_1, e_2, \ldots, e_n\rangle$ be a scenario. A scenario $sc_2 = \langle e_{2x}, \ldots, e_{2y}\rangle$ *extends* scenario sc_1 if it contains all events of sc_1 in the same relative order they appear, i.e., if event e_i happens before event e_j in sc_1, the same relationship holds in scenario sc_2, or, in other words, sc_2 *extends* sc_1 only if sc_1 is a subsequence of sc_2.

Symbolic Rule 3. $\forall x, y(x\ \textbf{extends}\ y \wedge Sc(x) \wedge Sc(y) \Rightarrow (\forall z : e(z) \wedge y\ \textbf{contains}\ z \Rightarrow x\ \textbf{contains}\ z) \wedge (\forall p, q : e(p) \wedge e(q) \wedge p\ happens\_before_y\ q \Rightarrow p\ happens\_before_x\ q))$.

Rule 4. A service Se_2 *extends* service Se_1 if Se_2 includes all of Se_1's scenarios, and Se_2 has new scenarios, i.e., there exist scenarios in Se_2 which are not elements of Se_1, or there exist scenarios of Se_2 which extend scenarios of Se_2.

Symbolic Rule 4. $\forall x, y(x\ \textbf{extends}\ y \wedge Se(x) \wedge Se(y) \Rightarrow y \subseteq x \wedge (x \neq y \vee \exists p, q : Sc(p) \wedge Sc(q) \wedge p \in x \wedge q \in y \wedge p\ \textbf{extends}\ q))$.

Societies

- Concepts: $\{$**SM**, **Ac**, **op**$\}$; Key: SM = service Manager; Ac = actor; op = operation.

- Relations
  - redefines $\subset$ **op** $\times$ **op**

    A common reason to redefine or override an operation is to provide more specific functionality for a service manager which inherits an operation from another service manager (see below).

    Rule. A redefined operation has the same name, and often (but not necessarily) the same signature, but a different implementation.

    Symbolic Rule. $\forall x, y(x$ ***redefines*** $y \wedge op(x) \wedge op(y) \iff x.n = y.n \wedge x.imp \neq y.imp)$.
  - includes $\subset$ **SM** $\times$ **SM**; inherits_from $\subset$ **SM** $\times$ **SM**

    Aggregation and generalization are two special types of relationships between service managers that foster reusability and extensibility. Aggregation, captured in the includes relation, models a 'whole/part' relationship in which one manager as a whole has other managers as parts, or, in other words, if service manager x includes service manager y, it implies that y is required in order to use service manager x. Generalization, captured by the *inherits_from* relation, means that a manager has all the capabilities defined by another manager, potentially has additional ones, and can redefine others (polymorphism). For example, LessonPlanBuilding includes Binding Manager indicates that a service manager LessonPlanBuilding includes operations of a Binding Manager. Similarly, RelevanceFeedbackSearch Manager *inherits_from* Search Manager indicates that a RelevanceFeedbackSearch Manager has the same capabilities as the Search Manager as well as additional ones (e.g., for query expansion).

    Rule 1. x includes y indicates that a service manager x has all operations defined in service manager y plus others not defined in y.

    Symbolic Rule 1 .$\forall x, y(x$ ***includes*** $y \wedge SM(x) \wedge SM(y) \Rightarrow y.op \subseteq x.op \wedge y.op \neq x.op)$.

    Rule 2. x inherits_from y indicates that a service manager x has all operations from the service manager y and defines additional operations, or x redefines some operations of y.

    Symbolic Rule 2. $\forall x, y(x$ ***inherits_from*** $y \wedge SM(x) \wedge SM(y) \Rightarrow (y.op \subseteq x.op \wedge y.op \neq x.op) \vee (\forall z \in y.op - x.op : \exists w \in x.op : w$ ***redefines*** $z))$.
  - invokes: **op** $\times$ **op**

    It is generally useful to specify dependencies between operations when discussing issues of extensibility and reusability. For example, search_similar(do) *invokes* match(q:query, C:collection) indicates that a search_similar operation invokes a match operation, defined in a Service Manager x or in another manager that x inherits from or includes.

    Rule. 1. f invokes g indicates that operation f may invoke operation g, namely, that within the body of operation f there is an expression whose evaluation invokes g (g is a subfunction of f). The operation f defined in a service manager x may only invoke an operation g, if g also is defined in x or in another manager that x includes or inherits from.

    Symbolic Rule. $\forall f, g(f$ ***invokes*** $g \wedge op(f) \wedge op(g) \wedge (\exists p : SM(p) \wedge f \in p \wedge g \in p) \iff$ g is a subfunction of $f \iff \exists$ functions $r, s : g = r \circ f \circ s$
  - association: **Ac** $\times$ **Label** $\times$ **Ac**

    A generic relationship between actors without a pre-defined semantics, this one captures generic societal relationships between communities of actors. For example, the relation (Professor, "teaches", Learner) is self-explanatory.

3.3.2 Inter-S Relationships

In this section, we identify several relations that cross the borders of Ss. Our emphasis here is on the relationships between the dynamic aspects of the DL, characterized by societies and scenarios, and the more “static” aspects of the DL, characterized by concepts in the other Ss. We also further explore other relationships among the three static Ss.

Scenarios and Societies

- Relations:
  - executes $\subset$ **e** $\times \langle op \rangle$

    The changes of computational states which are triggered by events in a scenario are computationally realized by invoking operations defined on service managers. Let $\langle op \rangle$ be the set of finite sequences from op. e_k *executes* $\langle op \rangle_j$ indicates that the list of operations $\langle op \rangle_j = \langle op_{1j}, op_{2j}, \ldots, op_{n_{jj}} \rangle$ is executed as the result of the occurrence of event e_k. Also if P_k is the set of event parameters of e_k and P_j is the union of all parameters of all operations in $\langle op_j \rangle$, $P_j \subseteq P_k$. For example, search(q,C) *executes* match(q,C) states that an event *search* executes an operation *match* (probably defined in a Searching Manager) between a query q and the set of digital objects in the collection C.
  - recipient $\subset \{$**SM** $\cup$ **Ac**$\} \times$ **e**

    In a scenario it is normally useful to identify the societal members that receive events for the purpose of checking consistency, security, etc. For example, the following two relationships specify recipients of events in a simple searching scenario: Search Manager *recipien*t search(q,C); Researcher *recipient* results($\{(do_i, w_i)\}$).

    Rule. recipient $\subset \{$SM $\cup$ Ac$\} \times$ e indicates that a specific service manager or actor is the receiver of an event in a scenario. Any actor can be the receiver of any event. If the event has an *execute* relationship with some operation, the receiver must be a Service manager which should have this operation.

    Symbolic Rule. $\forall x, y, z(x$ ***recipient*** $y \wedge y$ ***executes*** $z \wedge SM(x) \wedge e(y) \Rightarrow \forall w \in z.Img : w \in x.op)$.
  - participates_in $\subset \{$**SM** $\cup$ **Ac**$\} \times$ **Sc**

    This relation makes explicit the societal entities interacting in a scenario.

    Rule. Indicates that a service manager or actor x participates in a specific scenario y of a DL service by being a recipient of an event z of scenario y.

    Symbolic Rule. $\forall x, y(x$ ***participates_in*** $y \wedge (SM(x) \vee Ac(x)) \wedge Sc(y) \Rightarrow \exists z : e(z) \wedge y$ ***contains*** $z \wedge x$ ***recipient*** $z))$.

    For Service Managers, a consequence of the defined relations is that only operations defined in the participating managers should be associated with events of the scenarios in the service. This gives rise to the following consistency rule between a scenario and a society.

    Symbolic Rule. $\forall x, y, z, w(Sc(x) \wedge e(y) \wedge op(z) \wedge x$ ***contains*** $y \wedge y$ ***executes*** $w \wedge z \in w.Img \Rightarrow \exists p : SM(p) \wedge p$ ***partipates_in*** $x \wedge z \in p.op)$.
  - uses $\subset$ **Ac** $\times$ **Se**

    In many real DL settings it is useful to specify that only specific kinds of Actors may be allowed to use certain services. For example, while a researcher should be allowed to use all information seeking services, services such as ‘lesson plan building’ and ‘dissertation submission approval’. should be used only by teachers and archivists, respectively.

    Rule. Indicates that an Actor is allowed to use a specific service by participating in some of the services’ scenarios.

    Symbolic Rule. $\forall x, y(x$ ***uses*** $y \wedge Se(y) \wedge Ac(x) \Rightarrow \exists z : Sc(z) \wedge SM(w) \wedge z \in y \wedge x$ ***participates_in*** $z)$.

– runs $\subset$ **SM** $\times$ **Se**

  Rule. Service Manager x runs service y if all operations executed in all scenarios of y are defined on x or in managers that x includes or inherits from.

  Symbolic Rule. $\forall x, y(x \textit{ runs } y \wedge SM(x) \wedge Se(y) \Rightarrow \forall z, p, q, r : (Sc(z) \wedge e(p) \wedge op(r) \wedge z \in y \wedge z \textit{ contains } p \wedge p \textit{ executes } q \wedge r \in q.Img \Rightarrow r \in x.op)$.

Structures, Streams and Spaces

- Relations:

  – $I_C \subset \theta \times VPM \times H$

    Let $C \in Coll$ be a collection, H be the set of all handles of digital objects in C, $\theta \subset \bigcup_{do \in C} do(4)$ be a set of all triples (node, stream, interval) associated to digital objects in the collection, where interval is a pair of natural numbers (a,b) corresponding to a portion of the stream (or a substream) and $VPM = \bigcup_{sp \in \{Vec \cup Pr \cup Metric\}} sp$. An type of index I_C is a relation that maps specific substreams associated with nodes of specific digital object structures and elements of a vector, probability, or metric space representing those substreams to handles of digital objects. Normally, the elements of these spaces are built by extracting *statistical features* (e.g., number of specific text terms, histograms) from the respective substream. In the case of a probability space, the elements of H are mapped from a finite set with a discrete measure assigning positive probabilities to elements of that set. If an index fuunction I in defintion 20 of Chpater 2 only mapped feature singletons to handle singletons, and if the index were the **union** of these pairs, rather than a set of set of pairs, then the index from Chapter 2 would be the same as I_C here.

    Symbolic Rule. $\forall x(x \in I_{C_i} \Rightarrow \exists y, z : do(y) \in C_i \wedge x(1) = y(1) \wedge z = x(2) \wedge z \in y(4))$. $\forall C(Coll(C) \wedge (h, s, v) \in I_C \wedge (h, s, v') \in I_C \Rightarrow v = v')$.

Scenarios and (Streams, Structures, Spaces)

- Relations:

  – employs $\subset$ **Se** $\times$ S^3; produces $\subset$ **Se** $\times$ S^3

    Let S^3 = **Streams** $\cup$ **Structures** $\cup$ **Spaces** be the union of all concepts of the respective Ss. DL services manipulate, transform, and return instances of the concept types defined in S^3 or their component parts. For example, the notion of distance (as defined by a metric space) or probability (as defined by a probability space) are essential to services which need to compute a similarity measure between objects in the DL or between a patron's intrinsically vague information need and objects in the DL. Examples of services that normally employ spaces to compute these measures include searching, filtering, recommending, visualizing, classifying, and clustering. Also, services exist that transform DL objects (digital objects, metadata specifications, structures, streams) into different types of spaces for many purposes. Examples include services such as indexing, which transforms structured streams into elements of a vector or probability space, rendering or visualizing, which normally takes collections and transforms into a 2D/3D-metric space, or customizing, that normally transforms a space (e.g., a user interface (UI) or a distance function) into another personalized space (e.g., a customized UI or a personalized distance function [60]).

    Due to the complexity and number of possible instances of this relation, we will postpone the discussion to the next section, where we will further characterize the relationships between services and the other "static" Ss by making explicit employed inputs and produced outputs of

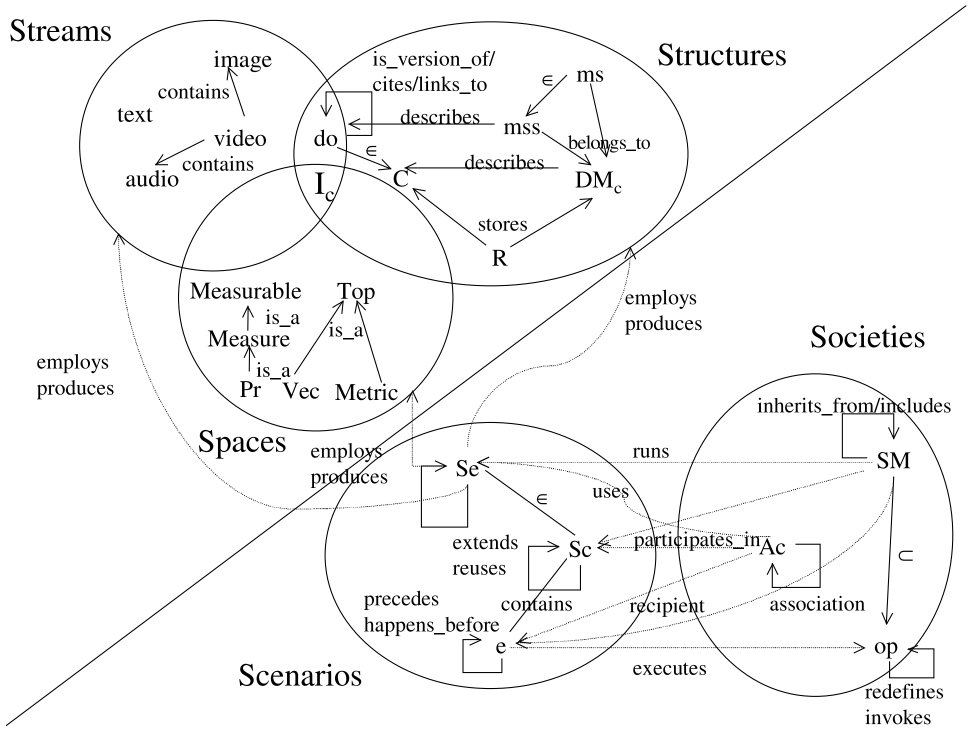


Figure 3.1: DL ontology

events in these services as well as pre- and post-conditions that constrain when and how these services can be evoked and combined.

The resulting ontology is graphically depicted as shown in Figure 3.1. Each S is represented as a circle containing the respective concepts. Normal lines represent inter-S relations while dotted lines correspond to inter-S relationships. Arrows linked to a whole indicate that the relationship can exist among all concepts in an S.

3.4 Example Application of the Ontology: Expanding the “Minimal DL” to the “Typical DL”: A Taxonomy of DL Services

Our objective in this section is to further explore some of the most important types of relations in the DL ontology, namely the “employs” and “produces” relationships between services and the other static “Ss” and the “extends” and “reuses” relationships among services. More specifically, in this section we want to answer questions such as:

1. Which DL elements are employed or produced by the different DL services?
2. Which are the fundamental DL services?
3. Which kinds of service compositions are possible or valid?
4. Which DL services are elementary or composite?

By answering these questions we make two contributions:

- we expand the definition of the “minimal digital library” by formally characterizing a number of typical digital libraries services, other than the ones defined in Chapter 2; and
- we demonstrate how to use these characterizations to reason about how DL services can be built from other DL components such as repositories and societal interactions, as well as be composed with other services by extension or reuse.

Table 3.1 shows a set of activities or services derived from a shortened list of the DL taxonomy’s activities presented in Chapter 2, along with their informal definitions. From that list of activities/services, we chose only those with explicit “employs/produces” relationships with some static components of the DL, thus removing entries which:

- had to do with the user’s mind and perception and did not correspond to any tangible DL service (e.g., those under ‘abstracting’);
- had to do with pure societal interactions(e.g., ‘collaborating’, ‘publicizing/advertising’)
- had to do with evaluation of services (e.g., ‘analyzing logs’, ‘leading focus groups’);
- had to do with operational aspects of the DL (e.g., ‘renewing’);
- had to do with specific implementation or architectural issues of the DL (e.g., ‘federating’ which deals with the aspect of distributing queries over a network).

It is very hard to argue for completeness/suficiency of both the set of chosen services and their definitions. An argument similar to the one made for the completeness of the taxonomy in Chapter 2 can be made regarding the representativeness of this set of services as the most common ones in typical DLs, since those were driven from the analysis of the DL literature for a relatively long period of time.

In Tables 3.2, 3.3, and 3.4 each service is characterized by parameters (input, output) of the initial and final events of the scenarios that compose those services[3] as well as pre- and post-conditions that should hold before and after the respective events. Correctness of the entries in Tables 3.2, 3.3, and 3.4 can be assessed in terms of if they correctly capture the semantics of the informal definition of each service in Table 3.1, asssuming that those are correct. Finally, it is also worth noticing that we are not extending the proper concept of ‘Service’ in the ontology, but we are *instantitating* or characterizing a number of members of that concept in the context of the ontology’s relationships. All other previous definitions and keys apply here. Those definitions are complemented with the following ones.

Definition 27 *A query q is a (possibly structured) representation of a user interest or information need. The exact format of a query is left unspecified here since it is system-dependent.*

Definition 28 *An annotation ann_{ik} is a descriptive metadata specification that exists only in reference to a digital object do_i; $ans_{ij} = \{ann_{i1}, ann_{i2}, \ldots, ann_{ik_j}\}$ is a set of annotations describing do_i.*

Definition 29 *Hyptxt is a hypertext (see formal definition in Chapter2); anchor is a node of a hypertext.*

Definition 30 *A personal binder bi_{uk} is a subset of some collection $C \in Coll$ for an actor $ac_u \in Ac \in Soc(1)$.*

Definition 31 *A log_entry is a descriptive metadata specification about an event of a scenario.*

[3]In fact, some scenarios of a service can have different initial and final output. In this case it is assumed that the most ‘typical’ scenario of a service can be identified and that we can use the initial and final events of that scenario for the purposes of characterizing the service.

Definition 32 *$tfr \subset S^3 \times Spaces$ is a function that transforms any element of a concept in S^3 into a space. $Transformers = \{tfr_1, tfr_2, \ldots, tfr_n\}$ is a set of such functions.*

Definition 33 *Let $\{do_i : i \in I = \{i_1, i_2, \ldots, i_{n_I}\}\}$ be a set of digital objects and $Ct = \{c_1, c_2, \ldots, c_m\}$ be a set of labels for m categories. A classifier $class_{Ct} : \{do_i : i \in I\} \rightarrow 2^{Ct}$ is a function that maps a digital object to a set of categories.*

Definition 34 *A cluster clu_k is a subset of a set of digital objects.*

Service	**Informal Definition**
Acquiring	Takes a set of digital objects, belonging to a collection not in the DL, and incorporates them into some collection of the DL
Annotating	Incorporates an annotation to a set of annotations of a digital object; this set of annotations describes the digital object
Authoring	Creates a digital object and incorporates it into some collection of the DL
Binding	Incorporates a set of digital objects into a personal binder of some actor
Browsing [a]	Given an anchor of a hypertext returns a set of digital objects
Cataloging	Incorporates a metadata specification into a set of metadata specifications describing a digital object
Classifying	Takes a digital object and a classifier and assigns a number of possible categories to the object from a finite set of possibilities
Clustering	Takes a set of digital objects and produces a number of subsets; the union of those subsets should be equal to the original set.
Conserving	Takes a collection and produces a similar one, i.e., another collection with all objects of the previous one.
Converting	Takes a digital object and produces a version of it (by changing its streams, structures, or structured streams)
Copying/ Replicating	Takes a digital object, produces an identical one, and incorporates it into some collection of the DL
Crawling (focused)	Given a collection, produces a subset or a copy of that collection
Customizing (interface)	Transforms (using a transformer) the appearance of a user interface, i.e., transforms a metric space into a different one
Describing	Produces a description of a digital object (in terms of a metadata specification) and incorporates this description into the object's set of metadata specifications
Digitizing	Produces a new digital object from a hard-copy version
Disseminating	Given a set of handles of digital objects in a collections, returns the respective objects
Evaluating	Given a digital object, produces an evaluation (i.e., a real number) for it
Expanding (query)	Given a query, an index for a collection, and a set of digital objects returned by the original query and marked as relevant/pertinent to that query by a user, produces a new, modified query
Extracting (structure)	Given a stream, produces a structured stream from it.
Filtering	Given a set of digital objects, and a query, a threshold (a real number), and an index, or a category and a classifier, produces a subset of the original set, in which the objects either match the query with a weight higher than the threshold or belong to the specified category
Harvesting (metadata)	Given a set of handles of digital objects, returns the set of metadata specifications for those objects
Indexing	Given a collection, produces an index for it
Linking	Includes a digital object into a hypertext
Logging	Produces a log entry from some event from a scenario of some service
Measuring	Produces a measure for a specific digital object
Rating	Given a digital object, produces a rate (real number) for it
Recommending	Given a collection and an actor, and a set of ratings for objects in that collection produced by others or the same actor, recommends (produces a subset of that collection) for that particular actor
Requesting	Given a handle of a digital object, returns the respective object
Reviewing (peer)	Produces a revision (i.e., a real number or a metadata specification) for a digital object
Searching	Given a query, a collection, and an index for that collection, returns for each object in the collection a real number indicating how well the query matches with the object
Submitting	Either incorporates 1) a new object into the collections of the DL; 2) a new metadata specification into the set of metadata specifications of a digital object; or) a new operation into the set of operations of a service manager
Training (classifier)	Produces a classifier given a set of digital objects along with their associated categories
Translating (format/ language)	Produces a version of a digital object by changing its format or language
Visualizing	Given a colllection produces a visualization (i.e., a metric space) for it

[a]The definition of a browsing service in Chapter 2 includes a number of different outputs for browsing events over a hypertext, including internal structures of digital objects and their structured streams. For the sake of simplicity in this discussion we will consider browsing services whose events's output include only a set of digital objects.

Table 3.1: Informal services definitions

Service	**User input**	**Other Service Input**	**Output**
Acquiring	$\{do_i : i \in I\}$	none	C_j
Annotating	do_i, ann_{ik}	(h_i, ans_{ip})	(h_i, ans_{iq})
Authoring	none[a]	none	do_i
Binding	$\{do_i : i \in I\}, bi_{um}$	$\{do_j : j \in J\}$	bi_{un}
Browsing	anchor	$Hyptxt_j$	$\{do_i : i \in I\}$
Cataloging	h_i, ms_{ik}	(h_i, mss_{ip})	(h_i, mss_{iq})
Classifying	do_i	$class_{Ct}$	$(do_i, \{c_x, \ldots, c_y\})$
Clustering	$\{do_i : i \in I\}$	none	$\{clu_k : k \in K\}$
Conserving	C_i	none	C_k
Converting	do_i	none	do_j
Copying/ Replicating	do_i	none	do_j
Crawling (focused)	C_i	none	C_k
Customizing (interface)	ac_i , tfr_k	sp_q	sp_j
Describing	none	do_i	ms_{ik}
Digitizing	none [b]	none	do_i
Disseminating	$\{h_x, \ldots, h_y\}$	none	$\{do_x, ..., do_y\}$
Evaluating	do_i	none	(do_i, w_i)
Expanding (query)	$\{do_i : i \in I\}$	I_C, $q_i, \{do_j : j \in J\}$	q_k
Extracting (structure)	stm_i	none	(st_j, Ψ_{ij})
Filtering	q, t, $\{do_i : i \in I\}$	I_{C_i}	$\{do_j : j \in J\}$
	c_k, $\{do_i : i \in I\}$	$class_{Ct}$	$\{do_k : k \in K\}$
Harvesting (metadata)	$\{h_x, \ldots, h_y\}$	none	$\{(h_x, mss_{h_x k}), \ldots, (h_y, mss_{h_y t})\}$
Indexing	C_i	none	I_{C_i}
Linking	do_i	$Hyptxt_j$	$Hyptxt_q$
Logging	none	e_i	log_entry_i
Measuring	do_i	sp_j	(do_i, w_i)
Rating	do_i, ac_j	none	(do_i, ac_j, r_{ij})
Recommending	ac_i, C_k	$\{(do_i, ac_j, r_{ij}) : i \in I, j \in J\}$	$\{do_m : m \in M\}$
Requesting	h_i	none	do_i
Reviewing	do_i	none	(do_i, w_i)
	do_i	none	ms_{ik}
Searching [c]	q, C_i	I_{C_i}	$\{(do_k, w_{qk}) : k \in K\}$
Submitting	do_i, C_k	none	C_j
	do_i, ms_{ik}, DM_{C_j}	none	DM_{C_t}
	op_k, sm_i	none	sm_j
Training (classifier)	$\{do_i, i \in I\} \times 2^{Ct}$	none	$class_{Ct}$
Translating (format/ language)	do_i	none	do_j
Visualizing	C	tfr_k	sp_j

[a]User's mind is outside scope.

[b]Analog/Real world items are outside scope.

[c]In fact, this and some other services that take digital objects (or sets of them, or collections) as inputs could also receive digital object surrogates such as metadata specifications as inputs as well. For sake of simplicity we chose not to specify those here and leave them as extensions for future work.

Table 3.2: Services inputs and outputs

Service	Pre-conditions	Post-conditions
Acquiring	$\exists C_t \notin Coll : \{do_i : i \in I\} \subseteq C_t$;	$C_j \in Coll; \exists C_k \in Coll : C_k \cup \{do_i : i \in I\} = C_j \wedge C_k \cap \{do_i : i \in I\} = \emptyset$
Annotating	$\exists C \in Coll : do_i \in C; h_i = do_i(1)$; $\exists x \in Cat : (h_i, ans_{ip}) \in x$	$\exists y \in Cat : (h_i, ans_{iq}) \in y \wedge ann_{ik} \in ans_{iq} \wedge$ ans_{iq} describes do_i
Authoring	none	$\exists C \in Coll : do_i \in C$
Binding	$\exists C \in Coll : \{do_i : i \in I\} \subseteq \{do_j : j \in J\} \subseteq C; bi_{um} \in Coll$	$\{do_i : i \in I\} \cup bi_{um} = bi_{un} \in Coll$
Browsing	anchor $\in Hyptxt_j$	$\exists C \in Coll : \{do_i : i \in I\} \subseteq C$
Cataloging	$\exists C \in Coll : do_i \in C \wedge do_i : h_i = do_i(1); \exists x \in Cat : (h_i, mss_{ip}) \in x$	$\exists y \in Cat : (h_i, mss_{iq}) \in y \wedge ms_{ik} \in mss_{iq} \wedge$ mss_{iq} describes do_i
Classifying	$\exists C \in Coll : do_i \in C$	$\{c_x, \ldots, c_y\} \subseteq Ct$
Clustering	$\exists C \in Coll : \{do_i : i \in I\} \subseteq C$	$\bigcup_k clu_k = \{do_i : i \in I\}$
Conserving	$C_i \in Coll$	$C_i = C_k \in Coll$
Converting	$\exists C \in Coll : do_i \in C$	$\exists C \in Coll : do_j \in C \wedge do_j$ is_version_of do_i
Copying/ Replicating	$\exists C \in Coll : do_i \in C$	$\exists C \in Coll : do_j \in C \wedge do_i = do_j$
Crawling (focused)	$C_i \in Coll$	$C_i \supseteq C_k \in Coll$
Customizing (interface)	$sp_q \in Metric$; $ac_i \in Ac$; $tfr_k \in Transformers$	$sp_j \in Metric$; $sp_j = tfr_k(sp_q)$
Describing	$\exists C \in Coll : do_i \in C$	$\exists x \in Cat, y : y$ belongs_to $x \wedge ms_{ik} \in y \wedge$ y describes do_i
Digitizing	none	$\exists C \in Coll : do_i \in C$
Disseminating	$\forall h_i \in \{h_x, \ldots, h_y\} : \exists do_i, C : do_i \in C \wedge h_i = do_i(1)$	$\forall do_i \in \{do_x, \ldots, do_y\} : (h_i = do_i(1))$
Evaluating	$\exists C \in Coll : do_i \in C$	$w_i \in [a, b] \subset R$
Expanding (query)	$\exists C \in Coll : \{do_i : i \in I\} \subseteq \{do_j : j \in J\} \subseteq C$; $\exists x, y, op(x), SM(y) : x \in y \wedge op(q, C) = \{do_i : i \in I\}$	none

Table 3.3: Services pre- and post-conditions (Part 1)

Service	**Pre-conditions**	**Post-conditions**
Extracting (structure)	$stm_i \in Streams$	$st_j \in Structs$; $\Psi_{ij} \in St^2$; $stm_i \in \Psi_{ij}.Dom$; $st_j.V \in \Psi_{ij}.Dom$
Filtering	$\exists C \in Coll : \{do_i : i \in I\} \subseteq C$; $t \in [a,b] \subset R$	$\exists x, y, op(x), SM(y) : x \in y \wedge op(q, \{do_i : i \in I\}) = \{(do_j, w_j) : j \in J\}$; $\forall (do_g, w_g) \in \{(do_j, w_j) : j \in J\}, w_g > t$; $\{do_j : j \in J\} \subseteq \{do_i : i \in I\}$
	$\exists C \in Coll : \{do_i : i \in I\} \subseteq C$; $c_k \in Ct$	$\{do_k : k \in K\} \subseteq \{do_i : i \in I\}$; $\forall do \in \{do_k : k \in K\} : c_k \in class_{Ct}(do)$
Harvesting (metadata)	$\forall h_i \in \{h_x, \ldots, h_y\}, \exists C \in Coll, do_i \in C : h_i = do_i(1)$	$\exists DM \in Cat : \{(h_x, mss_{h_x k}), \ldots, (h_y, mss_{h_y t})\} \subset DM$
Indexing	$C_i \in Coll$	none
Linking	$\exists C \in Coll : do_i \in C$	$do_i \in Hyptxt_q$
Logging	$\exists x, y : Se(x) \wedge Sc(y) \wedge y \in x \wedge e_i \in y$;	none
Measuring	$sp_j \in Measure$	$w_i \in sp_j$
Rating	$\exists C \in Coll : do_i \in C$; $ac_j \in Ac$	$r_{ij} \in [a,b] \subset R$
Recommending	$\{do_i : i \in I\} \in C_k$; $\{ac_j, j \in J\} \subseteq Ac$; $C_k \in Coll$; $r_{ij} \in [a,b] \subset R$	$\{do_m : m \in M\} \subseteq C_k$
Requesting	$\exists C \in Coll : do_i \in C$	$h_i = do_i(1)$
Reviewing	$\exists C \in Coll : do_i \in C$	$w_i \in [a,b] \subset R$
	$\exists C \in Coll : do_i \in C$	$\exists x \in Cat, y : y$ belongs_to $x \wedge ms_{ik} \in y \wedge y$ describes do_i
Searching	$C_i \in Coll$	$\{do_k : k \in K\} \subseteq C_i$, $w_{qk} \in [a,b] \subset R$
Submitting	$C_k \in Coll$	$\{do_i\} \cup C_k = C_j \in Coll$
	$DM_{C_j} \in Cat$	ms_{ik} belongs_to $DM_{C_t} \in Cat$; $\exists x \in DM_{C_t} : ms_{ik} \in x \wedge x$ describes do_i
	$sm_i \in SM$	$op_k \in sm_j \in SM$
Training (classifier)	$\exists C \in Coll : \{do_i : i \in I\} \subseteq C$;	none
Translating (format/language)	$\exists C \in Coll : do_i \in C$	$\exists C \in Coll : do_j \in C \wedge do_j$ is_version_of do_i
Visualizing	$C \in Coll$	$tfr_k(C) = sp_j \in Metric$

Table 3.4: Services pre- and post-conditions (Part 2)

Table 3.5 shows an organized taxonomy of DL services featured in Tables 3.1 through 3.4, derived from a deep analysis of the entries in that table. The taxonomy was created by grouping together all services with similar I/O behaviour. In the highest level, we make a distinction between Infrastructure and Information Satisfaction Services. The latter is distinguished from the former by always receiving as a user input either an active, personal item (e.g., a binder, a personal transformer, a set of documents considered relevant/pertinent) or a personal representation of an information need (see Chapter 8 for a discussion on this) or interest (e.g., a query, an anchor, a profile, a handle of a desired object, a category) [4].

Infrastructure services can be further sub-divided into Repository-Building and Add_Value services. Repository-Building services are characterized by producing as output uniquely the four basic items of the "minimal DL" related to a Repository: collections, digital objects, catalogs, and/or metadata specifications. All Add_Value services produce distinct outputs, with the exception of the translating services. Preservational services are distinguished from Creational by producing outputs that are equal to their inputs or slightly variations of them, i.e., versions in the case of converting/translating services, for preservation purposes. Add_Value services either aggregate value/information to their inputs (e.g., annotations, evaluations, structures, indexes, measures, log entries, rates, reviews, translations, visualizations) or connect objects together (e.g., by training and classifying, clustering, indexing, linking).

Infrastructure Services			**Information Satisfaction Services**
Repository-Building		*Add_Value*	
Creational	Preservational		
Acquiring	Conserving	Annotating	Binding
Authoring	Converting	Classifying	**Browsing**
Cataloging	Copying/Replicating	Clustering	Customizing
Crawling (focused)	Translating (format)	Evaluating	Disseminating
Describing		Extracting	Expanding (query)
Digitizing		**Indexing**	Filtering
Harvesting		**Linking**	Recommending
Submitting		Logging	**Requesting**
		Measuring	**Searching**
		Rating	
		Reviewing (peer)	
		Surveying	
		Training (classifier)	
		Translating (language/format)	
		Visualizing	

Table 3.5: A taxonomy of DL services/activities

In this taxonomy, we further characterize a **fundamental** service (denoted by **bold**) as either:

1. one that produces as an output an element of the basic concepts belonging to our minimal definition of a DL, such as digital objects, metadata specifications, collections, and catalogs, without which a

[4]For for the sake of simplicity, in the tables, for some services such as customizing, rating, recommending, we did not make an explicit distinction between an actor and a *representation* of an actor. Despite being an important distinction, since services can only take representations as inputs, those representations can take multiple forms (e.g., a simple identifier, a complex profile), all which can in a way or another be represented as some kind of structure.

DL cannot exist;

2. one that belongs to the minimal set of DL services (i.e., indexing, searching and browsing) proposed in Chapter 2;

3. one that supports the three former services in terms of extension or reuse. In case of item 1, this includes all the Creational services, with the exception of crawling and harvesting, which produces collections and catalogs from existing inputs in the same or another DL.

In case of item 1, this includes all the Creational services, with the exception of crawling and harvesting, which produces collections and catalogs from existing inputs in the same or another DL. In case of item 3, this includes linking, fundamental to produce an hypertext which supports the browsing service. Similarly a composite (denoted by underlining) DL service is one that takes input from some other service (as per column 3 of Table 3.2); otherwise the service is called elementary. For example, it can be seen from Table 3.2 that all Preservational services and all the Creational services but 'describing' (which takes a digital object as an input from another service) are elementary.

One interesting application of the taxonomy is as a tool to help reason about issues of reuse/extension of services. Services that produce outputs that can be employed as input by other services have a high potential to be (re-)used by the latter and services that have similar behaviors have also a large potential for reuse/extension. For example, Figure 3.2 graphically depicts **fundamental** services, split into Infrastructure (Creational + Add_Value) and Information Satisfaction services. Normal arrows represent relationships, where those marked with 'p' and 'e', denote 'produces' and 'employs' relationships respectively. Dotted arrows represent the transformation of a user interest/information need into a representation useful as an input of a service, in the context of the 'uses' relationship between actors and services. From the figure is easy to see that: 1) the output of (fundamental) infra-structure services is normally the input of either another Creational service or a (fundamental) Information Satisfaction service; and 2) there are many potential reuse/extend inter-dependencies among the Creational services. Authoring and digitizing produce digital objects that when submitted produce DL collections and when described/catalogued produce catalogs. Collections are also produced by acquiring services. An indexing service takes a collection and produces an index used by searching services while linking services produce hypertexts used in browsing services. These along with requesting services give ultimately access to the digital objects of the DL to the interested actor.

Another example is shown in Figure 3.3 which expands on the right portion of Figure 3.2 by depicting input/output interactions (and therefore potential for reuse/extension) between fundamental and composite information services and between the latter and some non-fundamental Add_Value services. Common to the composite information satisfaction services depicted in the figure is the fact that all of them take a set of digital objects or a collection as input. Recommending takes an actor's representation, a collection, and the output of a rating service (i.e., triples (digital object, actor, rating)), and produces a subset of the original collection [156]. Filtering also takes a user interest representation expressed as a query or a category of interest, either an index or a classifier (produced by a training service), and a set of objects to produce a subset of the original set. Similarly, binding takes the output produced by searching or browsing services and produces a new binder which contains (a subset of) those objects. Visualizing produces a space out of a collection and a transformer, while expanding a query takes the original query submitted to a Searching service, an index for a collection, and a subset of the response set (i.e., relevant and/or non-relevant documents) and produces a modified query.

Many other possible compositions are possible and can be seen by analyzing the entries in Tables 3.1-3.4 and the similarities implied by the taxonomy. Without wanting to be exhaustive at all, some examples include:

- Recommending, Filtering, Binding, and Expanding query, among others, produce a set of digital objects; those sets can be further indexed for searching/browsing purposes.

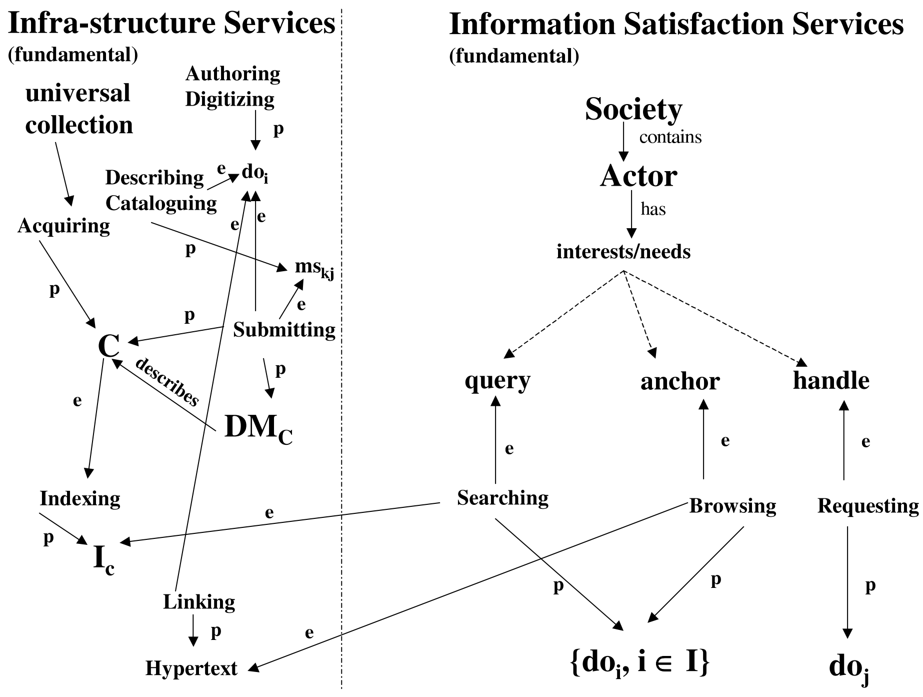


Figure 3.2: Several examples of compositions among infrastructure and information satisfaction DL services

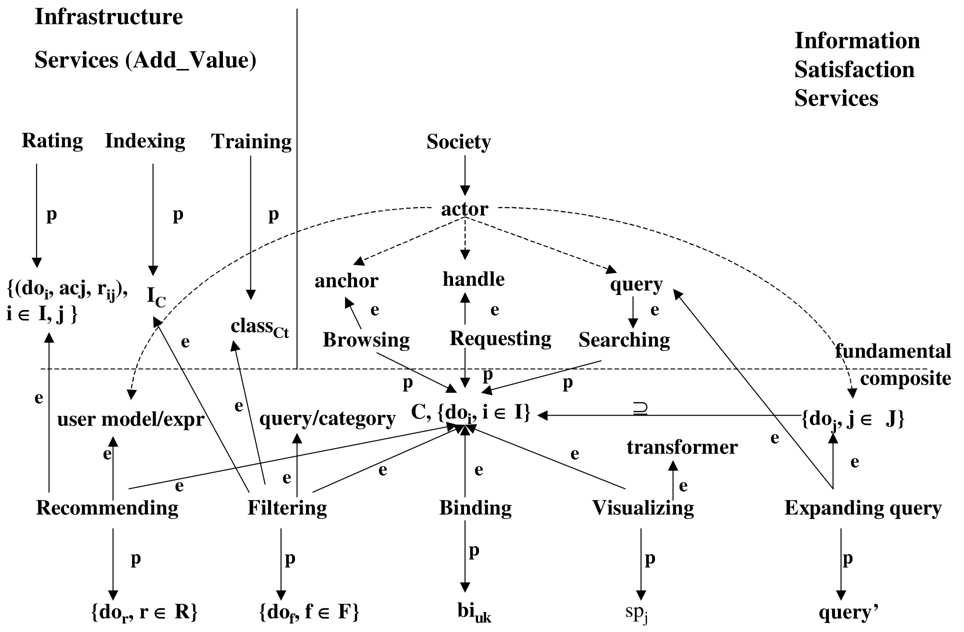


Figure 3.3: Examples of compositions of services

- Conserving should be able to reuse a copy/replicating service multiple times to produce a copy of a collection.
- Classifying could be extended to implement filtering, by simply including events to check if a specified category is in the returned set of categories for an object.
- Translating should be able to be re-used by converting for some types of conversion.
- An advanced searching service may reuse extracting (structure) and extend a Searching service to provide support for structured queries [87].

The reader is invited to continue exploring other possibilities. To finalize, one can devise an interesting practical application for the ontology/taxonomy where all those relatiohsips are materialized in terms of design patterns/object-oriented hierarchies/component pools to serve as a backbone for the construction of the IR/DL systems, therefore promoting reuse of code and saving time and money.

Part II

Practical Applications/Tools

Chapter 4

Language – Declarative Specification of DLs: the 5SL Language

Digital libraries (DLs) have emerged as an important research and application area, facilitated by advances in information technology, especially the Internet and the World Wide Web (WWW). There is strong demand for new and varied DL systems capable of dealing with all kinds of mixed-mode, multimodal, and multimedia digital content. As hundreds of DLs are created by colleges, universities, associations, and diverse other organizations to deal with content they create or digitize, and with local collections of information from numerous remote sources, strong pressure will be exerted to tailor each to special requirements of use as well as content.

The process of building a digital library involves specification of the content to be stored; how that content is organized, structured, described, and accessed; which services are offered by the library (e.g., searching, browsing, personalizing, collaborating); and how patrons (and automated agents) ultimately use those services and interact with each other in the DL environment. Thus, by their own nature, DLs are complex and inter-disciplinary information systems.

This complexity makes it difficult and expensive to construct new systems. Nowadays DLs are built within monolithic, tightly integrated, and generally inflexible systems – or by assembling disparate components in an ad-hoc way, with resulting problems in interoperability and adaptability. Ad hoc construction is not sufficient to meet that demand. More importantly, conceptual modeling, requirements analysis, and methodological approaches are rarely supported, making it extremely difficult and expensive to tailor DL content and behavior to particular communities' interests and needs. The general trend has been to develop tools to solve small parts of the problem, whereas the root of the problem – the lack of specific DL patterns, models, methodologies, formalisms, and languages – is almost completely ignored.

To address these challenges, we propose a novel digital library modeling language for conceptual DL design, based on 5S, called 5SL. 5SL is a high-level, domain-specific language, which allows declarative specification of a number of DL features that are often considered in isolation. Domain specific languages are explicitly designed to address a particular class of problems by offering specific abstractions and notations for the domain at hand. Thus, the main contributions of this chapter are: 1) the raising of the level of abstraction in digital library specification and modeling, through a DL design methodology which is model-driven and use-case based; 2) an illustration of how the various DL design issues may be combined in a coherent framework that enriches, extends, and customizes classical models for database, hypertext, information retrieval, and software engineering.

This chapter is organized as follows. Section 4.1 gives an overview of the language and its foundations. Section 4.2 details and exemplifies the use of each of the five models (in 5SL). Section 4.3 compares our approach to related research.

4.1 5SL General Organization

5SL is a domain-specific, declarative language with a formal semantics for conceptual modeling of digital libraries. The formal semantics is understood in terms of a translation of language constructs into the 5S-formalized framework. Its formal basis provides an unambiguous and precise DL specification tool, which can facilitate prototyping, allow proofs of assertions and aid validation of implementations.

With 5SL, the specification of a digital library consists of 5 complementary perspectives. Figure 4.1 presents a UML-based graphical representation of that **5SL meta-model**, corresponding to the portion of the 5S ontology used in the language. A modeling language is a representation of some theory and serves as a meta-model whose instantiation produces models of specific (DL) systems. As in normal UML notations, boxes represent Concepts or classes. Lines with diamonds diamonds represent “whole/part” aggregation relationships (e.g., services are composed of scenarios which are composed of events) and lines with large arrowheads pointing to a (parent) class represent a “is-kind-of” generalization relatiohsiphs (e.g., actors and service managers are two kinds of communities). Besides showing a “view” of the meta-model which is more familiar to system designers, this representation also help to focus on the parts of the DL ontology utilized in the language. It also shows some relationships not represented directly in the ontology, but derivable from some of its relationships such as ‘employs’, ‘uses’, and ‘runs’. One example is the dependecy relationships (dashed lines) which show that one class uses another class as an argument in their events/operations.

To improve acceptability and interoperability, 5SL makes extensive use of existing standard specification sublanguages for representing DL concepts, when that turns out to be possible. The need for the integration of multiple languages is a key aspect of the domain-specific language approach [10]. A domain typically consists of multiple subdomains, each of which may require its own particular language. This is particularly true for digital libraries but the aggregative nature of 5S matches this requirement quite well.

5SL utilizes an XML syntax. The abundance of XML supporting software tools facilitates the construction of DL generators (see Chapter 6). Most of the 5SL model primitives are defined as XML elements, which can enclose other sublanguages that help to define DL concepts. In more detail, MIME types constitute the basis for encoding streams. XML Schema [218] and/or RDF Schema [217] are the primary tools for describing structures. And finally, an adapted and extended version of UXF [206], an XML serialization of UML [25], is used with the Societal and Scenario Models.

4.1.1 Running Example

Throughout this chapter, we employ a simplified running example (scenario) to illustrate the features and use of 5SL.

> A Networked Digital Library of Theses and Dissertations (NDLTD) digital library manages collections of electronic theses and dissertations (ETDs). Students produce ETDs as a result of their graduate studies and are responsible for submitting their work along with metadata. In the process of creating ETDs, students are encouraged to fully apply new multimedia and hypermedia technologies. The submission service is controlled by a workflow process, which includes a review phase and a cataloguing phase. In the review phase, the university staff checks ETD files, the metadata submitted by the student, and payment of appropriate fees. In the cataloguing phase, MARC, ETD-MS (a metadata standard for ETDs), and other metadata formats are produced from the workflow process, possibly complemented, and distributed for several other catalogs. The DL services for patrons include fulltext and keyword-based searching as well as browsing by author and department. Optionally other services like topical or hierarchical browsing or recommendations can be offered.

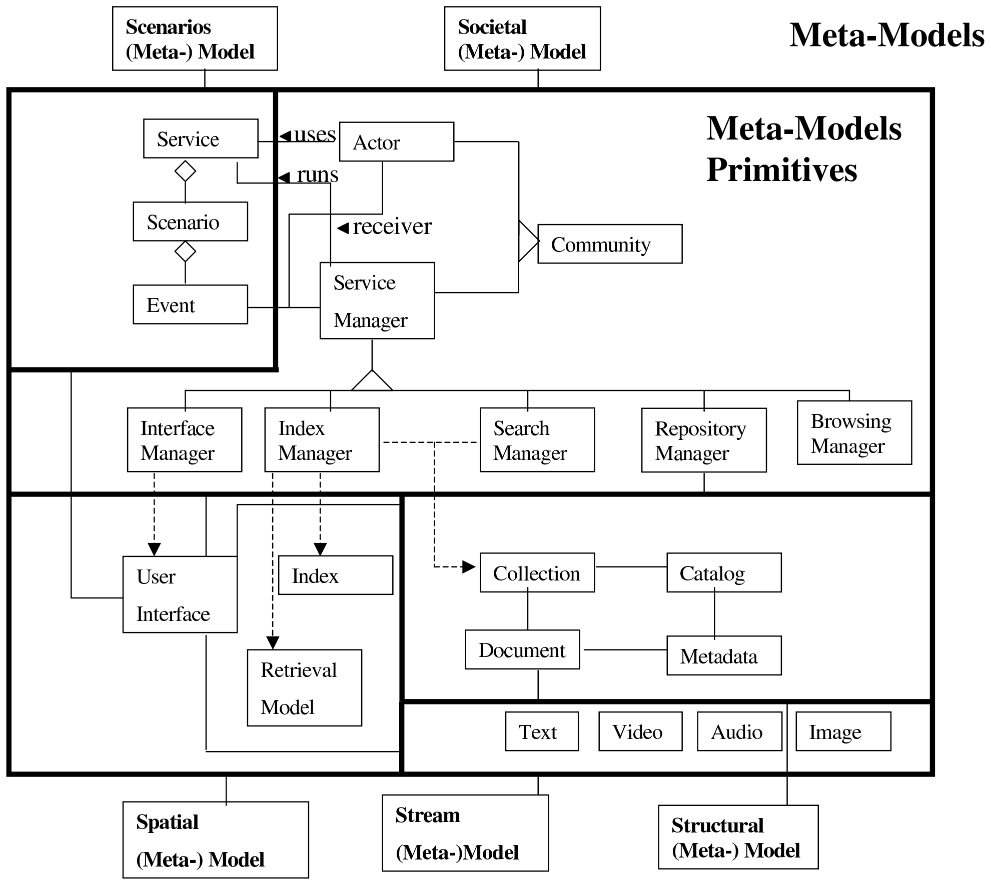


Figure 4.1: 5SL metamodel for minimal DL

```
<streams>
   <text name='ETDText'>
     <content-type>text/xml
         <charset>UTF-8</charset>
     </content-type>
     <content-type>application/pdf</content-type>
     <lang>ENG</lang>
   </text>
   <audio name='ETDAudio'>
     <content-type>audio/x-aiff</content-type>
   </audio>
   <video name ='ETDVideo'>
     <content-type>video/mpeg</content-type>
   </video>
    . . .
 </streams>
```

Figure 4.2: Stream model example

4.2 5SL Models

4.2.1 Stream Model

The stream model specifies the kinds and formats of multimedia content supported by the digital library. Preservation concerns and technological limitations influence the specification of this model. With the objective of promoting reusability and standardization, we have based this model in the Web standard of MIME types. A MIME type consists of a type (e.g., text), a subtype (e.g., plain, xml) and, in the case of textual data, a character set, which corresponds to the encoding and language. A three-character Z39.53 language code (e.g., ENG, JPN) is used to indicate the language.

Figure 4.2 shows a subset of the kinds of streams supported by an NDLTD DL. It includes English XML texts using UTF-8 encoding, PDF files, and several audio and video formats.

4.2.2 Structural Model

The structural model considers multiple sources for organization of a digital library. In this model, one describes the internal structure of digital objects (documents), metadata standards, properties of collections and catalogs, as well as knowledge organization tools, which impose organization upon collections, catalogs, and sets of concepts.

Figure 4.3 presents the portion of the Document Type Definitions (DTDs) (i.e., context-free grammars that define the logical structure of acceptable XML documents) for the structural model, which shows how these different aspects of the DL organization are arranged in a 5SL description.

The internal structure of digital objects (or documents) is defined with the use of XML Schema. Similarly, structures for descriptive metadata and knowledge organization structures can be described either with XML or RDF Schema. XML Schema, a standard promoted by the W3C, was conceived to improve on the deficiencies of DTDs.

A typical description of a document is shown in Figure 4.4, which presents a skeleton of an ETD document specification.

```
<!ELEMENT Structure (Document, Metadata,
                     Collection, Catalog)>
<!ELEMENT Metadata (Descriptive,
                    Administrative?)>
```

Figure 4.3: DTD for structural model

```
<document name='ETD'>
  <stream_enumeration>
    <stream value='ETDText'>
    <stream value='ETDAudio'>
     ...
  </stream_enumeration>
  <structured_stream>
     %XMLSchema%
  <structured_stream>
</document>
```

Figure 4.4: Document definition example

Documents are defined by imposing structures over (sets of) streams or by using the structure to provide some organization among them. In other words, a document is seen as a structural composition of streams. The set of streams that enter in the composition of the documents (as reflected by the relationship between a document and the entire Stream Model in Figure 4.1) is specified in the <stream_enumeration> subsection. An XML Schema inside the section <structured_stream> defines the internal organization of the document as a structuring of the streams.

Properties of collections of documents and catalogs of metadata such as name, creator, maintainer, availability, and semantics are similarly defined in the Structural Model. Figure 4.5 shows the 5SL encoding of a NDLTD DL catalog which supports ETD-MS and Dublin Core as metadata formats.

```
<catalog name='VT-ETDCatalog'>
  <creator='fox@vt.edu' >
  <maintainer='mgoncalv@vt.edu'>
  <public ='true'>
  <metadata_format = 'ETD-MS'
   schema='http://www.ndltd.org/standards/metadata/
          etdms/1.0/etdms.xsd'/>
  <metadata_format = 'dublin_core'
   schema='http://www.openarchives.org/OAI/1.1/dc.xsd'/>
  ....
</catalog>
```

Figure 4.5: Catalog definition

4.2.3 Spatial Model

The Spatial Model specifies a framework for modeling logical representations and operations of several DL components. In particular, this model gives details of the underlying DL retrieval models(e.g., use of vectorial or probability spaces), including detais for indexes, and the user interface appearance (i.e., sets of metric spaces).

```
<Spatial_Model name='NDLTD_Space_Model'>
     <UI name='NDLTD_UI'>
            <Rendering>HTML</Rendering>
     </UI>
     <IR_Model>
         <Retrieval_Space>Vector</Retrieval_Space>
         <Index>
                <Stemming>Porter</Stemming>
                <Stopwords>NDLTDStopWords.txt</Stopwords>
         </Index>
     </IR_Model>
<Spatial_Model>
```

Figure 4.6: Spatial model example

Figure 4.6 especifies that the rendering of the NDLTD user interface is HTML-based (instead of Java-based), that a vector space is used as basis for retrieval (instead of a probability or metric space), that the Porter's algorithm is used for stemming (instead of, for example, a morphological algorithm), and that the set of utilized stopwords for removal from the index is listed in the specified file.

Currently these properties are described only for documentation purposes and are not being used in the DL generation process described in Chapter 5, but we intend to use them in the future for configuration of some of the components that implement these funcionailites.

4.2.4 Societal Model

The fundamental concept in the societal model is that of a 'community', a set of entities (human or computer) that share the same characteristics and behavior. In the Societal model of 5SL, we are concerned about identifying the different communities that interact within the DL environment, the main functionalities associated with them, and their semantic relationships. The model is based on the classical object-oriented paradigm, and uses the concepts of attributes, methods, and relationships.

In 5SL, as in 5S, we distinguish two main types of communities: service managers and actors. Service managers administer DL services while actors explore those services to fulfill their information needs. Service managers also define and implement basic methods or operations. The exact behavior and functionality of the DL services are described in the Scenario Model with sequences of events (and corresponding invocations of operations) representing interactions or collaborations between the communities. Note, though, that the functional description of societies as described by their operations/methods could be automatically generated from the Scenarios Model, and their implementation also could be generated by forward engineering [25] or algorithmic model transformations [186]. We have decided to explicitly model at least the interfaces of those operations/methods in the Societal Model as a design choice for consistency purposes, to aid explicit modeling of relationships and constraints, and because of its possible impact on the DL generation process. This decision can be reviewed in the future as we gain more insight with the additional

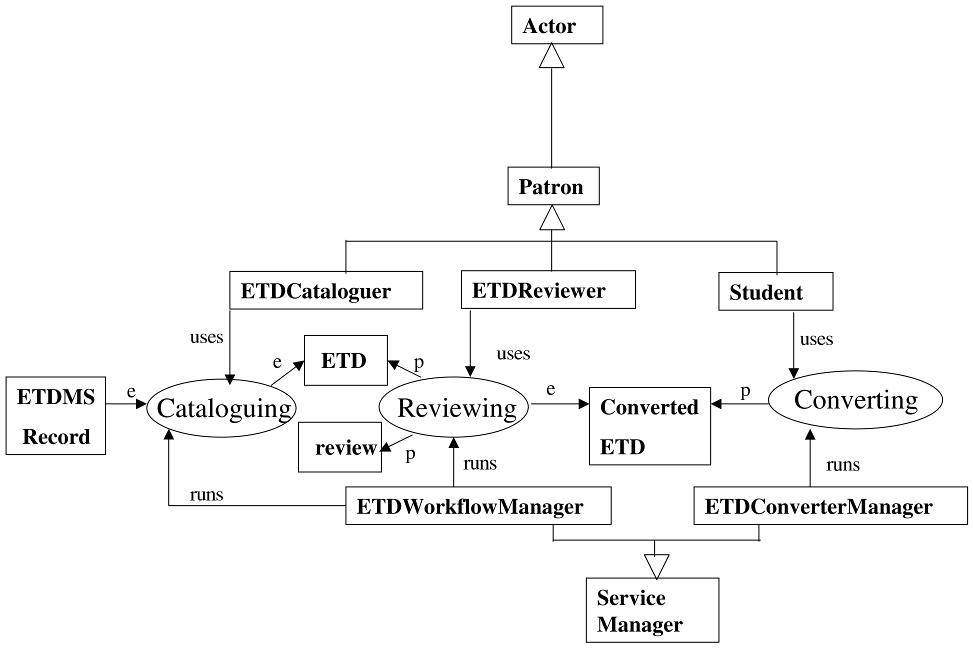


Figure 4.7: Example of a societal schema for an NDLTD site (in UML)

use of the language. Figure 4.7 shows a simple societal schema for an ETD site that captures the semantics described in the running example (described with UML notation). This schema consists of four actors (Patron, Student, ETDReviewer, and ETDCataloger) and two managers, the ETDWorkflowManager and the ETDConverterManager, and three services, converting, reviewing, and cataloguing [1]. In particular, the ETDConverterManager is responsible for converting ETD files into standard formats as specified in the Stream Model.

The XML code of Figure 4.8 represents part of the 5SL encoding of the societal schema (corresponding to dark boxes in Figure 4.7).

Core Service Managers Model

The core service managers model is formed by a set of pre-defined managers, whose supported services constitute the minimal functionality that a digital library should support. There are five of these core managers, as shown in Figure 4.1. The InterfaceManager is responsible for the active aspects of the user interface as for receiving and passing events for all the appropriate Service Managers. The SearchManager executes the search strategy as described in the retrieval model. It takes a query representation and a collection, and returns documents in the collection along with associated weights, where the weights specify how well the document representation matches with (or implies) the query. The BrowsingManager contains operations to manage hypertexts and to implement run-time navigation activities based on the hypertext topology. The IndexManager is responsible for, given a collection of documents, producing an Index as described in the Spatial Model. The Repository Manager manages collections and potentially catalogs (if it stores them).

[1]We chose to represent services with oval symbols. Besides differetiate them from the other Societal Concepts it is also consistent with the representation of UML Use cases, to which Services are related.

```
<Society>
   <Actor>
    <Community name='Patron' IsAbstract='true'>
      <Attribute name='name' type='String' visibility='private'/>
      <Attribute name='ID' type='Integer' visibility='private'/>
     </Community>
    <Community name='Student'>
      <Service>Converting</Service>
    </Community>
    <Community name='ETDReviewer'>
      <Service>Reviewing</Service>
    </Community>
    <Community name='ETDCataloguer'>
      <Service>Cataloguing</Service>
    </Community>
   </Actor>
    <Generalization>
      <Parent>Patron</Parent>
      <Child>Student</Child>
      <Child>ETDReviewer</Child>
      <Child>ETDCataloguer</Child>
    </Generalization>
  <Manager>
    <Community name='ETDWorkflowManager'>
      <Attribute name='ETDFiles' type='set(File)' visibility='private'/>
      <Attribute name='MetaRecord' type='ETDMSRecord' visibility='private'/>
      <Operation name='getETDFile' visibility='public' returnType='void'/>
      <Operation name='getETDMSRecord' visibility='public' returnType='void'/>
      ...
      <Service>Cataloguing</Service>
      <Service>Reviewing</Service>
    </Community>
  </Manager>
</Society>
```

Figure 4.8: 5SL-XML societal encoding of the NDLTD schema

Basic operations of the Repository Manager include adding, deleting, and retrieving documents from a collection. As we will see in Chapter 5, each of this ‘logical’ classes are materialized as software components in our DL component Pool.

4.2.5 Scenario Model

The purpose of the Scenario Model is to describe the behavior of the DL services. The description is realized as a set of sequences of events that the actor and managers engage to yield an observable result of value to members of the DL society. UML interaction diagrams provide a visual tool to help with this description. 5SL provides a specific way to serialize these graphical representations in XML.

Figure 4.9 exemplifies the use of a UML sequence diagram for describing a simple scenario of a searching service in an NDLTD DL. A patron searching for ETDs about a particular topic expresses her interests as a fulltext query. The request is passed through the InterfaceManager to the SearchManager that executes a search procedure, according with the strategy described in the retrieval model (see Spatial Model). The SearchManager then returns a weighted set of ETD identifiers where weights specify how well the corresponding ETD representations match with the query. The set is presented to the user by the InterfaceManager as an ordered list of titles, which allows her to scan to judge them for relevance. She finds a particular title appealing and requests the ETD. The request is carried out by passing the corresponding identifier to the Repository Manager, which retrieves the particular ETD for the researcher.

The corresponding 5SL code for this scenario is shown in Figure 4.10. The 5S theory describes scenarios as sets of events, which can be associated with operations and conditions [2]. The operation associated with

[2]In Tables 3.3 and 3.4 we have pre- and post-conditions only for the initial and final events, but potentially all events can have associated conditions.

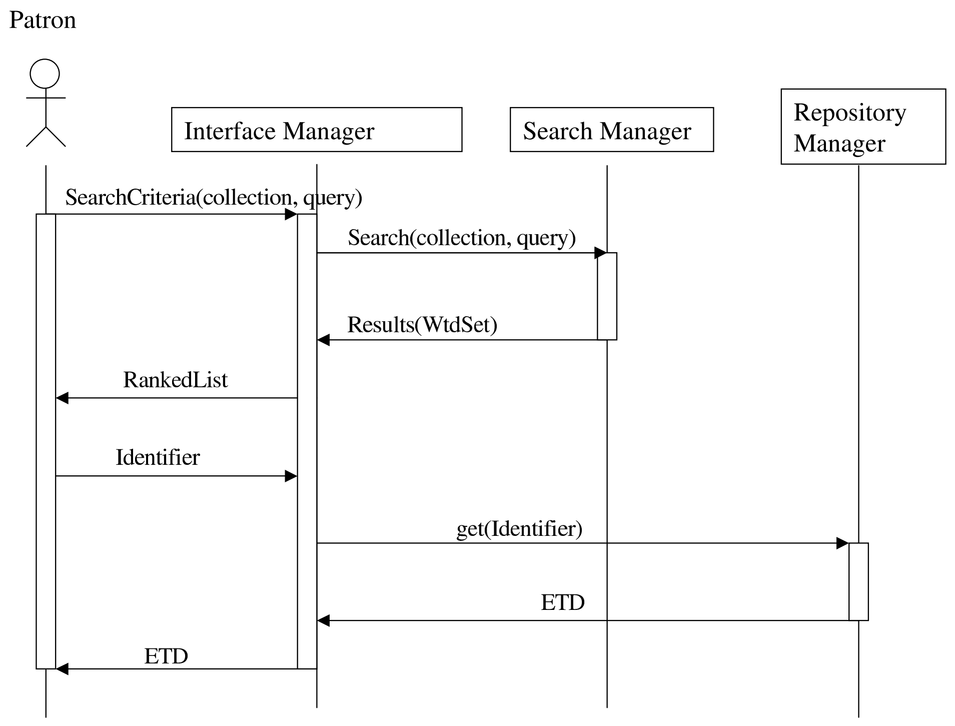


Figure 4.9: A simple search scenario for an ETD site

```
<SERVICE name ='Searching'>
  <SCENARIO name='SimpleSearching'>
  <NOTE>Simple scenario for an NDLTD
        site searching service</NOTE>
    <EVENT>
     <SENDER>Patron</SENDER>
     <RECEIVER>InterfaceManager</RECEIVER>
     <OPERATION name=SearchCriteria/>
       <PARAMETER>collection</PARAMETER>
       <PARAMETER>query</PARAMETER>
    </EVENT>
    <EVENT>
     <SENDER>InterfaceManager</SENDER>
     <RECEIVER>SearchManager</RECEIVER>
     <OPERATION name='Search'/>
      <PARAMETER>collection</PARAMETER>
      <PARAMETER>query</PARAMETER>
    </EVENT>
    <EVENT>
     <SENDER>SearchManager</SENDER>
     <RECEIVER>InterfaceManager</RECEIVER>
     <PARAMETER name='Results'>WtdSet
     </PARAMETER>
    </EVENT>
    <EVENT>
     <SENDER>InterfaceManager</SENDER>
     <RECEIVER>Patron</RECEIVER>
     <PARAMETER>RankedList</PARAMETER>
    </EVENT>
    <EVENT>
     <SENDER>Patron</SENDER>
     <RECEIVER>InterfaceManager
     </RECEIVER>
     <PARAMETER>Identifier</PARAMETER>
    </EVENT>
    <EVENT>
     <SENDER>InterfaceManager
     </SENDER>
     <RECEIVER>Repository
     </RECEIVER>
     <OPERATION name='get'/>
       <PARAMETER>Identifier
      </PARAMETER>
    </EVENT>
    <EVENT>
     <SENDER>Repository</SENDER>
     <RECEIVER>InterfaceManager
     </RECEIVER>
     <PARAMETER>ETD</PARAMETER>
    </EVENT>
    <EVENT>
     <SENDER>InterfaceManager
     </SENDER>
     <RECEIVER>Patron</RECEIVER>
     <PARAMETER>ETD
     </PARAMETER>
    </EVENT>
  </SCENARIO>
</SERVICE>
```

Figure 4.10: 5SL-XML serialization of the scenario depicted in Figure 4.9

the event, which may result in a change of state, is an executable statement that forms an abstraction of a computational procedure as defined in the societal model. The XML textual sequence of event descriptions corresponds to the temporal sequence in the scenario. Note also that the XML element ‘receiver’ in the code corresponds to the recipient ontological relationship. For completeness and to facilitate generation of code, we also introduced the ‘sender’ element.

A more complex service description is illustrated in Figure 4.11. For brevity, we omit events related to the InterfaceManager. The scenario depicts the Graduate School perspective of the ETD Submission service. The ETD reviewer logs into the system and checks for the lastest submitted ETDs. For each of those ETDs the reviewer repeats the same process: he chooses an ETD to review; the workflow manager responds with an ETD review page which lists ETD files, metadata, and options; the reviewer downloads ETD files for review and verifies the metadata; and finally he checks if the student has returned all forms and has paid all appropriate fees. The repetition is shown in the picture by an enclosing box with an associated condition; all events in the box are repeated if the reviewer chooses to review another ETD. The reviewer can then choose to accept the submission, so the ETD goes into the library collection. If the reviewer rejects it, an email goes to both student and advisor explaining the reasons for rejection. In the picture, the alternatives are presented by multiple arcs exiting from different boxes representing distinct states; the first arc carries the condition that led to the respective state.

Part of the 5SL encoding is shown in Figure 4.12. Two language-specific constructs were also introduced in the language to faciliate specification of different flows of control and are shown in the figure. Repetition is captured by the <INTERACTION> element. All events defined inside this element are repeated according to the order of their definition. Alternatives are defined inside a <BRANCHING> section. Each <WHEN> subelement with respective condition inside the branching defines an alternative path of execution.

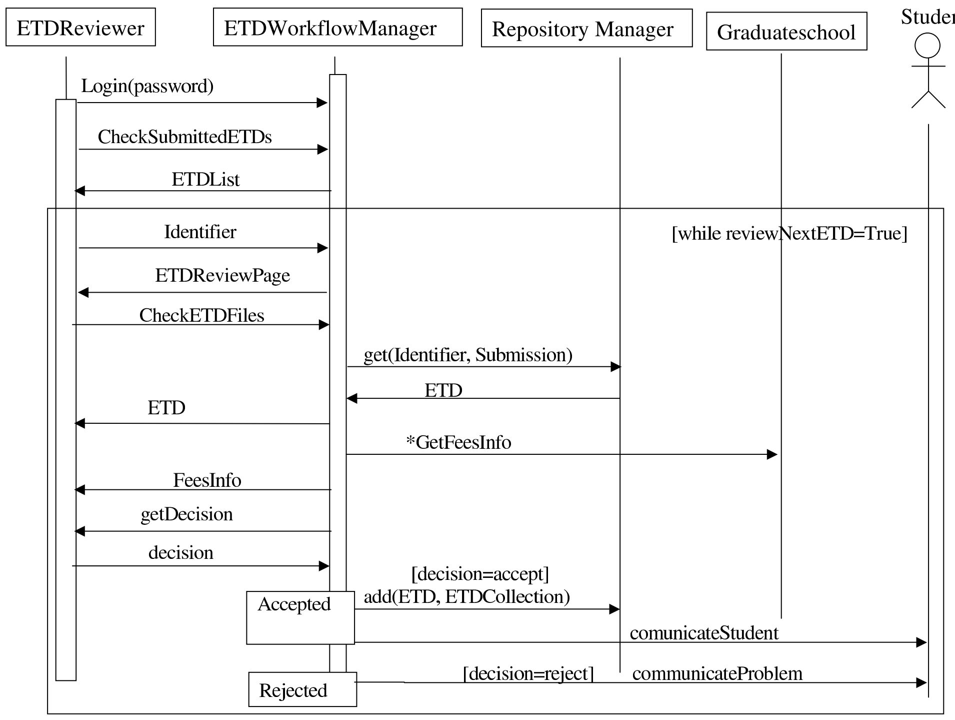


Figure 4.11: The review scenario for the submission service

```
<SERVICE name='Submission'>
  <SCENARIO name='ETDReview'>
   ...
     <INTERATION condition='while reviewNextETD=true'>
        <EVENT>
           <SENDER>ETDReviewer</SENDER>
           <RECEIVER>ETDWorkflowManager</RECEIVER>
           <OPERATION name='get'>
             <PARAMETER>Identifier</PARAMETER>
             <PARAMETER>Submission</PARAMETER>
        </EVENT>
       ...
       <BRANCHING>
          <WHEN condition='decision=accept'>
           <EVENT>
              <SENDER>ETDReviewer</SENDER>
              <RECEIVER>Repository</RECEIVER>
                 <OPERATION name='add'/>
                   <PARAMETER>ETD</PARAMETER>
                   <PARAMETER>ETDCollection</PARAMETER>
                 </OPERATION>
           </EVENT>
          ...
          </WHEN>
          <WHEN condition='decision=reject'>
            ...
          </WHEN>
        </BRANCHING>
     </INTERATION>
   </SCENARIO>
</SERVICE>
```

Figure 4.12: Portion of the 5SL-XML serialization of the scenario depicted in Figure 4.11

4.3 Related Work

Recent research, developed mainly in the database and hypertext communities, has been investigating declarative approaches and representations for specific kinds of information systems, mainly Web and e-commerce sites. Strudel [62], Tiramisu [6], and Active Views [1], are examples of systems that have this data-centered perspective of a Web site. The common objective is to separate Web site structure, data management, and page presentation, and to provide some query mechanism to allow manipulation of their representations (normally graph-based).

The hypertext/hypermedia community also has a long tradition of developing rich abstraction models and decompositions for hypermedia systems. Examples include OOHDM [187], Web2000 [17], and Autoweb [71]. An interesting approach close to ours is described in [176, 40]. The WebML modeling language and its supporting tool, Torii, provide powerful abstractions to describe and generate the hypertext and navigation structure of Web sites.

In comparison, 5SL factors information architectures at a finer granularity, which provides more expressiveness as well as more specific DL constructs and abstractions not present in WebML or any of the cited works.

An even closer approach is described in [238]. The Digital Library Definition Language (DLDL) is focused on describing external behavior of DLs for purposes of supporting interoperability in terms of federated searching. A similar approach is defined in [160]. These approaches however are limited in scope and in their ability to cover most of the challenges in the construction of complex digital libraries.

A remarkable system that shares many objectives with our approach is the New Zealand Greenstone DL system [229, 15]. Greenstone allows the construction of complex DLs and tailoring of many parts of DLs to specific domains and needs. However to achieve these goals Greenstone utilizes (in early versions) heterogeneous machinery including Perl modules, proprietary markup languages, and macros, CORBA, Standard Template Library (STL) in C++, etc [3]. In contrast, our approach presents more uniform, high level, and abstract way to deal with all these aspects without committing to any particular implementation or architecture.

In sum, although 5SL shares many of the goals exposed by the research community, none of the implemented methods presents such a comprehensive, homogeneous, and integrated DL oriented approach to cover almost all aspects of digital library design and construction. Moreover, in contrast with most of the related work, 5SL is deeply grounded is a rich formal theory for digital libraries.

[3]The latest version of Greenstone just released uses Java.

Chapter 5

Visualization – Visual Semantic Modeling of Digital Libraries: the 5SGraph Tool

The current interest from non-experts who wish to build digital libraries (DLs) is strong worldwide. However, since DLs are complex systems, it usually takes considerable time and effort to create and tailor a DL to satisfy specific needs and requirements of target communities/societies. What is needed is a simplified modeling process and rapid generation of DLs. To enable this, DLs can be modeled with descriptive domain-specific languages as 5SL. In such languages, models are made up of elements representing concepts, rules, and terminology that are part of the domain world, as opposed to the world of code or of generic modeling languages (e.g., UML [26]). Despite its advantages, domain-specific languages are sometimes hard to learn and master. A visual tool would be helpful to non-experts so they may model a DL without knowing the theoretical foundations and the syntactic details of the descriptive language. In this chapter, we present a domain-specific visual DL modeling tool, 5SGraph. It employs a metamodel that describes DLs using the 5S theory. 5SGraph presents the metamodel in a structured toolbox, and provides a top-down visual building environment for designers. The visual proximity of the metamodel and instance model facilitates requirements gathering and simplifies the modeling process. The output from 5SGraph is a DL model that is an instance of the metamodel, expressed in the 5S description language. Furthermore, 5SGraph maintains semantic constraints specified by a 5S metamodel and enforces these constraints over the instance model to ensure semantic consistency and correctness. 5SGraph enables model reuse to reduce the time and effort of designers. 5SGraph also is designed to accommodate and integrate several other complementary tools reflecting the interdisciplinary nature of DLs. The 5SGraph tool has been tested with real users and several modeling tasks in a usability experiment, and its usefulness and learnability have been demonstrated.

This chapter is organized as follows. Section 5.1 describes 5SGraph: its design, functionality, key features, and visualization properties. Section 5.2 presents design, measures, and results of a usability experiment to evaluate the tool.

5.1 The 5SGraph Modeling Tool

5.1.1 Motivation

With 5SL, a DL designer does not need to be an expert in software engineering or information science; she only needs to have a clear conceptual picture of the needed DL and be able to transform the conceptual picture to 5SL files. This greatly reduces the burden on designers, speeds up the building process, and increases the quality of the DLs built. However, 5SL has its own problems and limitations:

1. The designer must understand 5SL well enough to be able to write a 5SL file and to correctly use it to

express his/her ideal digital library.

2. The 5SL file, which describes a DL, consists of five sub-models (Stream model, Structural model, Spatial model, Scenarios model, and Societal model). Although all of the five sub-models are expressed in XML, they use different sets of concepts and have different semantics. Thus, the 5SL specification is compatible and extensible, because many existing standard formats can be used within the 5SL language. Yet, to build one DL, the designer needs to understand the five or more different semantic specifications that are required to express the system.

3. When large and complex digital libraries are to be built, it is very hard even for experts to manually write those XML files without any assistance from a tool.

4. It is very difficult to obtain the big picture of a DL just from a huge set of XML files. This may cause trouble for maintenance, upgrade, or even understanding of an existing system.

5. A number of semantic constraints exist between (inter-model constraints) and within (intra-model constraints) the sub-models. Designers need extra effort to ensure consistency in the whole model.

5.1.2 Requirements

Reflecting on the above mentioned disadvantages of 5SL, we consider the following four functions of a modeling tool based on the 5S/5SL framework to be essential: to help DL designers to 1) understand 5S quickly and easily; 2) build their own DLs without difficulty; 3) transform their models into complete, correct, and consistent 5SL files automatically; 4) understand, maintain, and upgrade existing DL models conveniently.

Accordingly, our 5SGraph modeling tool supports these functions as it provides an easy-to-use graphical interface. It automatically generates desired 5SL files for the designer. Since visualization often helps people understand complex models, 5SGraph is able to load and graphically display DL metamodels [1]. The visual model shows the structure and different concepts of a DL and the relationship among these concepts. 5SGraph also provides a structured toolbox to let the designer build a DL by manipulation and composition of visual components (see Figure 5.1). The structured toolbox provides all the visual components of the metamodel, and shows the relationships among these components. The visualization thus provides guidance while the designer is building the model. The designer only needs to deal with a graphical interface and pull visual components together. It is not required to memorize the details of the syntax and semantics of 5SL. Cognitive load is reduced. Typing effort and typing errors are reduced. Furthermore, correctness and consistency can be automatically guaranteed by 5SGraph; thus it yields correct and consistent 5SL XML files according to the visual model built by the designer. As such, 5SGraph eliminates the disadvantages of working with raw 5SL.

The concept of metamodel is very important to assure flexibility. The metamodel, which is a representation of the 5S theory, describes a generic DL. The model for a specific DL is an instance of the metamodel, which in our case is a domain-specific metamodel, i.e., specific to the domain of building DLs. Since the 5S framework is still under development, it is expected that more changes and additions will be made in the future, especially to 5SL. Fortunately, when given a new metamodel, the tool can be used with future versions of 5SL as well as more application-oriented specializations of it. One example of the latter is the 5S-based metamodel for archaeological DLs currently being developed in the ETANA project [164]

[1]Diffent metamodels can exist. For example, the current metamodel can evolve generating new versions or metamodels for more application-oriented types of digital libraries (e.g., ETANA – an archaeological DL) can be created.

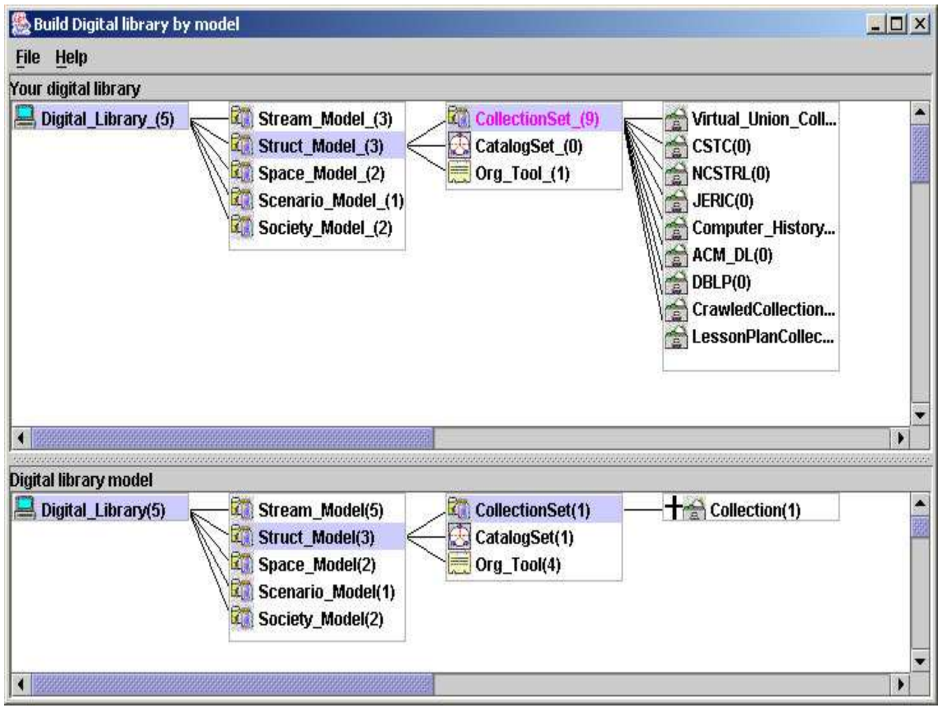


Figure 5.1: 5SGraph sample interface with structured toolbox (bottom part) and workspace (upper part); figure shows modeling of collections for the CITIDEL project (www.citidel.org)

5.1.3 Key Features.

Some of the major features of the tool include:

- Flexible and extensible architecture

  5SGraph is a domain-specific modeling tool. Thus, the model is made up of elements that are part of the domain world, not the whole entity world. 5SGraph is tailored to accommodate a certain domain metamodel for 5S. Methods that are appropriate only to 5S can be used to optimize the modeling process. Reuse in such a specific domain is realistic and efficient, because the models in that domain have many characteristics in common.

  The 5SL language extensively uses existing standards. The reason is that the specification of a DL involves many sub-domains, and there are various standard specifications for each sub-domain. There also are many well-developed tools for those sub-domains. For example, metadata is an important element in 5S. Several existing metadata editors can be used to view and edit metadata. Another example concerns the scenario part of 5S. A specific scenario can be modeled and described by UML sequence diagrams, so existing UML modeling tools could be used for this purpose.

  The 5SGraph tool should not “re-invent the wheel”. Therefore, the tool is designed to be a super-tool, which means it provides an infrastructure based on 5S and calls existing tools as needed. In the interest of brevity, however, this chapter focuses on how 5SGraph helps with modeling a DL, rather than on how 5SGraph calls other tools to create customized components.

- Reuse of sub-models

  In 5SGraph, model reusability means that models designed for one digital library instance can be saved and reused in other DL models. Reusability saves time and effort. There are models that are common for different DL systems. For example, many DLs share the same data formats, and the same descriptive metadata standards. The models representing streams or metadata can be built once and reused in different DLs. When a new model is needed, the user does not need to build a new one from scratch. He/she loads a similar (sub-)model and spends relatively less time by making minor changes to customize it (see Figure 5.2). Of course, not all models are designed to be reusable. A reusable model should be self-contained and independent.

- Synchronization between the model and the metamodel

  There are two views in the tool. One is for the toolbox (metamodel); the other is for the user model. These two views are related through the concept type/instance relationships between concepts in the toolbox and instances in the user model. When a user selects an instance in the workspace (user model), 5SGraph is able to synchronize the view of the toolbox by showing a visible path from the root to the corresponding type of selected concept (see Figure 5.1). The convenience of synchronization is that: 1) the user does not need to manually search all the components in the toolbox to find the correct type; and 2) The tool helps the user focus on the most important relationships of the type. The child parts that can be added to the current component are within easy reach.

- Enforcing of semantic constraints

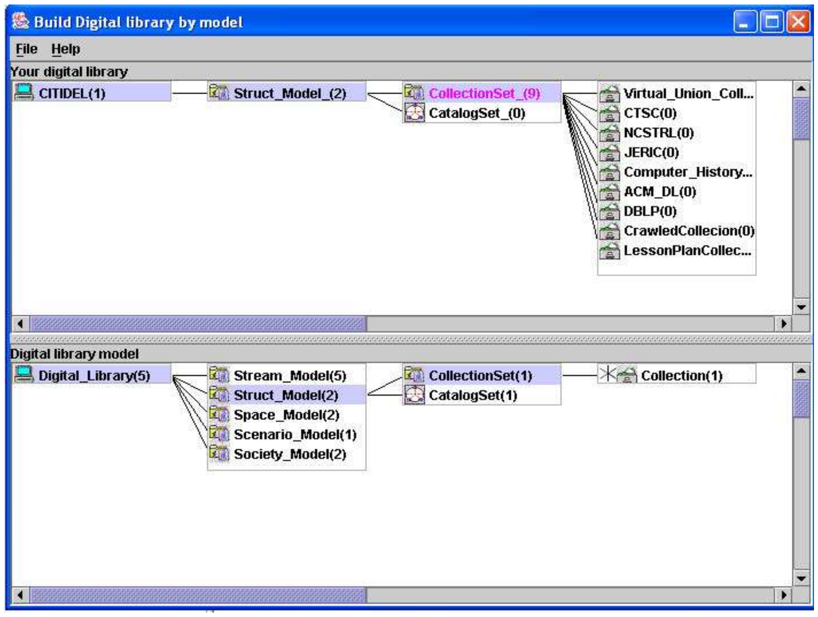
  Certain inherent semantic constraints exist in the hierarchical structure of 5S. These constraints in 5S are divided into two categories. Value constraints specify the range of possible values of an element, while association constraints define the relationships among different components. Examples of such constraints include:

  1. The streams used in the definition of a digital object (document) are predefined in the Stream Model.
  2. A collection consists of different kinds of documents. A catalog describes a collection, since a catalog collects administrative or descriptive metadata that apply to the digital objects in the collection. A catalog, therefore, is dependent on a collection.
  3. The services that the actor (a member of the Society Model) uses or a service manager (another member of the Society Model) runs can be drawn only from the services already defined in the Scenario Model.

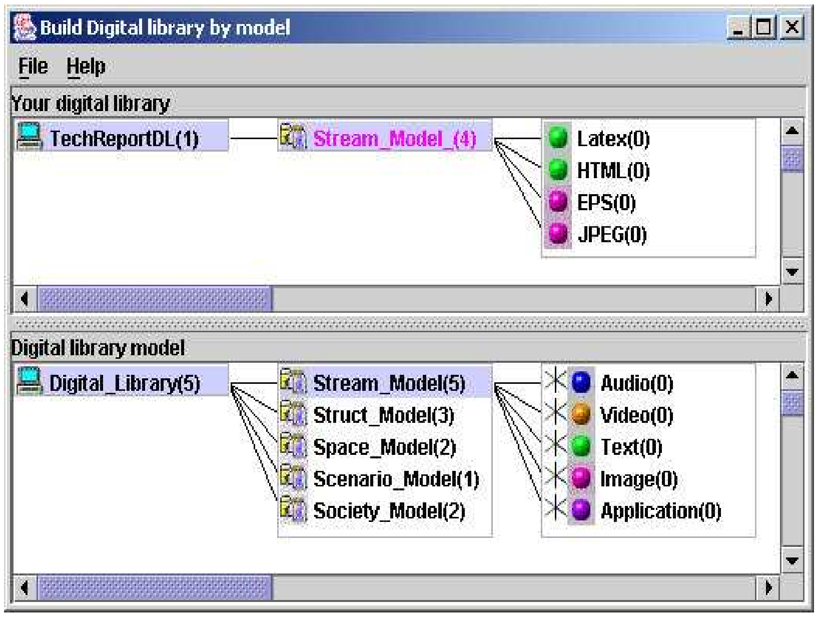
The 5SGraph tool is able to manage these constraints. For example, an actor only can use services that have been defined in the Scenario Model. For example, as occurs in CITIDEL, the declaration of an actor, Teacher, is shown in Figure 5.3(a). In order to associate actors with the services they use, the designer browses back to the Scenario Model to first define services: metadata search, multi-scheme browsing, profile filtering, browsing, cataloging, focused crawling, lesson plan building, and lesson plan customized browsing (this one with four scenarios: unordered and ordered browsing, guided path, and slide show - as supported by the VIADUCT manager). When the designer browses back to Actor in the Scenario Model in the metamodel, he/she finds that the created set of services are automatically added into the metamodel under the node ‘Actor’ (see Figure 5.3(b), structured toolbox), allowing him/her to connect the defined services with the actors that use them. In the example, Learner is connected to all but two services (focused crawling, run by the ‘crawlifier’ manager, and lesson plan building, used only by teachers).

5.2 Evaluation

We conducted a pilot usability test of 5SGraph. The questions to be answered were: 1) Is the tool effective in helping users build DL models based on the 5S theory? 2) Does the tool help users efficiently describe

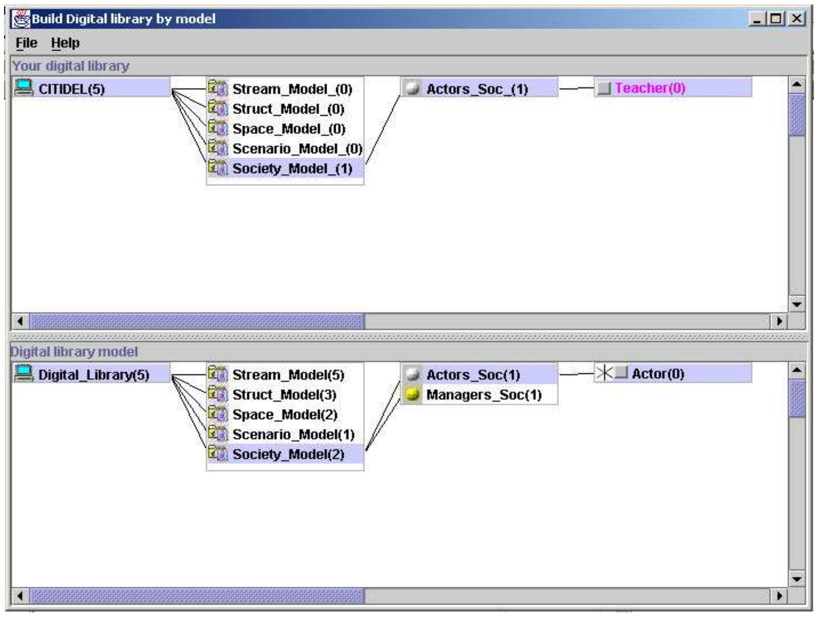


(a)

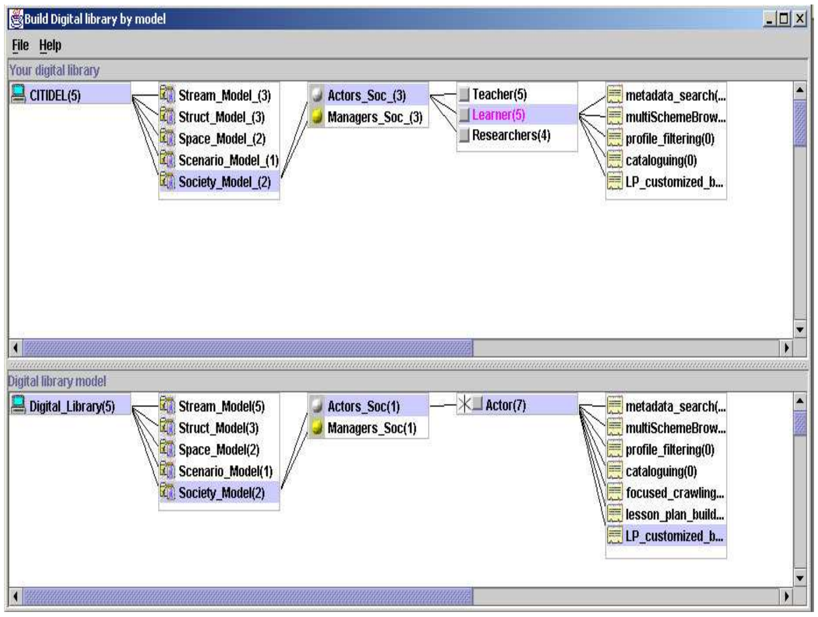


(b)

Figure 5.2: Reuse of models before and after loading



(a)



(b)

Figure 5.3: Enforcing semantic constraints in the CITIDEL system. See (top) teacher as actor and (bottom) learner services

DL models in the 5S language? 3) Are users satisfied with the tool? Participants of this preliminary test included seventeen volunteers from a graduate level Information Storage and Retrieval class, or from the DL research group of Virginia Tech. We choose participants who have basic knowledge of DLs and have the motivation to create DLs. These types of people are some of the target users of the tool. Three representative tasks with different levels of difficulty were selected:

- Task 1: build a simple model of a Technical Report Digital Library using reusable sub-models

  The difficulty level of this task is low. Its purpose is to help the participants to get familiar with 5S and the 5SGraph tool.

- Task 2: finish an existing partial model of CITIDEL (Computing and Information Technology Interactive Digital Educational Library, www.citidel.org)

  The difficulty level of this task is medium.

- Task 3: build a model of NDLTD (Networked Digital Library of Theses and Dissertations, www.ndltd.org) from scratch.

  The difficulty level of this task is high.

The procedures were as follows:

1. the participant was asked to read some background documents about 5S and the modeling methodology;
2. the participant was given an introductory presentation on 5SGraph;
3. the participant was given a description of task 1 and we recorded how he/she completed it;
4. after the participant finished each task, he/she was given the next task description immediately;
5. after the participant finished all the tasks, he/she was given a questionnaire form to fill out.

5.2.1 Measures

We use the following test measures:

- Effectiveness
  - Completion rate: percentage of participants who complete each task correctly.
  - Goal achievement: extent to which each task is achieved completely and correctly.
- Efficiency
  - Task time: time to complete each task.
  - Closeness to expertise: minimum task time divided by task time.
- Satisfaction Satisfaction is measured using a subjective 10-point bipolar rating scale, where 1 is the worst rating and 10 is the best rating. After each participant finishes all three tasks, he/she is given a questionnaire and asked to rate the overall learnability, effectiveness, and satisfaction based on his/her observation.

	Task 1	Task 2	Task 3
Completion Rate (%)	100	100	100
Mean Task Time (min)	11.3	11.4	15.1
Mean Closeness to Expertise	0.483	0.752	0.712
Mean Goal Achievement (%)	97.4	97.4	98.2

Table 5.1: Overall performance results for three tasks

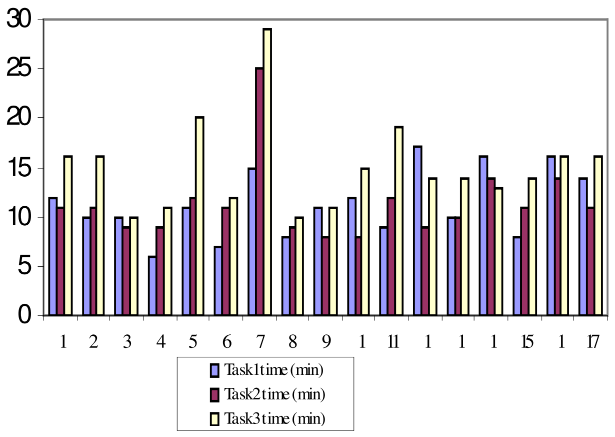


Figure 5.4: Task time

5.2.2 Results

- Effectiveness: The high completion rate and the high goal achievement rate demonstrate the effectiveness of 5SGraph (see Table 5.1).
- Efficiency: Most participants finish tasks in less than 20 minutes (see Figure 4); the generated 5SL files are very accurate.
- Closeness to Expertise reflects the learnability of the tool (see Table 2, Figure 5). There are three observations, which have been confirmed statistically (t test: 0.05).
  1. Observation 1: the mean Closeness to Expertise in task 2 is significantly greater than that in task 1.
  2. Observation 2: the mean Closeness to Expertise in task 3 is significantly greater than that in task 1.
  3. Observation 3: the mean Closeness to Expertise in task 3 is not significantly different from that in task 2.

The results suggest that the tool is easy to learn and use. Working on a simple and short task such as task 1 is enough for users to become familiar with the tool. User proficiency is quite close to expert performance level after they use the tool the first time. In fact, there are some participants (#9 and #10) with good computer skills who achieved a completion speed very close to the expert's in tasks 2 and 3. Observation 3 indicates that users have similar performance in tasks 2 and 3. The reason may be that users have become highly familiar with the tool after task 1. The remaining difference between the participants and the expert

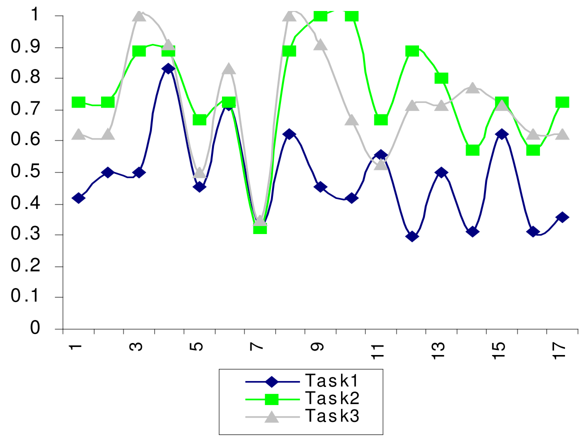


Figure 5.5: Closeness to expertise

may be due to other factors, e.g., familiarity with the tasks, typing speed, reading speed, and skill in using computers. The average rating of user satisfaction is 9.1 and the average rating of tool usefulness is 9.2. From these numbers, it does appear that our participants are satisfied with the tool and consider it useful for building DLs based on 5S.

Chapter 6

Generation – (Semi-)Automatic Generation of Componentized DLs: the 5SGen Tool

In the previous chapters, we have introduced languages and tools to allow customized modeling of DLs. In this chapter we describe a new generic DL generator based on the 5S framework, focusing on support for two key aspects of DLs: 'societies' and 'scenarios' [35]. The principal contribution of this work is the development, implementation, and deployment of a generic DL generator that can be used by DL designers to semi-automatically produce tailored DL services from models of societies and scenarios. By doing this the generator attempts to bridge the gap between DL models and system implementation, i.e., between concept and execution, therefore partially validating the formal theory of 5S. We demonstrate the feasibility of this approach and substantiate our claims by providing two examples that illustrate the features of the generator. This chapter is organized as follows. Section 6.2 describes our approach and development environment. Section 6.3 is the core of the chapter and details examples, architecture, and implementation of our digital library generator. The 'Examples' subsection focuses on extensibility and reusability. Section 6.4 deals with related work.

6.1 Summary of the Approach

Our objective is to cover the whole process of DL development, from requirements to analysis, analysis to design, design to implementation. We aim to generate 'tailored' DL software satisfying the particular requirements of specific DL societies. The basic idea is to develop models, languages, and tools able to capture the rich set of DL requirements and properties of particular settings and to automatically convert these 'patterns' into different representations by properly compiling, transforming, and mapping models in different levels and phases of the DL development process. The assumption is that automatic transformations and mappings diminish the risk of inconsistency and increase productivity. This view will be supported by:

1. Having a model based approach that allows the DL designer to describe: 1) the kinds of multimedia information the DL supports (Stream Model); 2) how that information is structured and organized (Structural Model); 3) different logical and presentational properties and operations of DL components (Spatial Model); 4) the behavior of the DL (Scenario Model); and 5) the different societies of actors and managers of services that act together to carry out the DL behavior (Societal Model). We have organized and formalized these and other DL notions into the 5S (Streams, Structures, Spaces, Societies, Scenarios) framework. This formal framework provides a foundation for the DL generator.

2. Using a domain-specific language based on 5S, 5SL, for declarative specification and automatic generation of DLs. Domain specific languages enable applications to be programmed with domain abstrac-

tions, thereby allowing compact, clear, and machine-processable specifications to replace detailed and abstruse code [18].

3. Using scenario based design for defining the behavior of a system. Scenarios keep design discussion focused on user activities, more specifically, they keep design discussion focused on the level of task organization that actors experience in their tasks. In 5S, we envision scenarios as sequences of events that modify states of a computation in order to accomplish some functional requirement. We use scenarios to describe the behavior of DL services and societal interactions.

4. Implementing a code generator that allows a DL designer to provide a modeling specification in terms of scenarios and societies. This generates implementations using precise transformations/mappings. The generated DL makes use of well defined components that each carry out key DL functions interacting with one another using lightweight protocols [1]. We draw heavily upon work with the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [150] and Open Digital Libraries (ODL [199]).

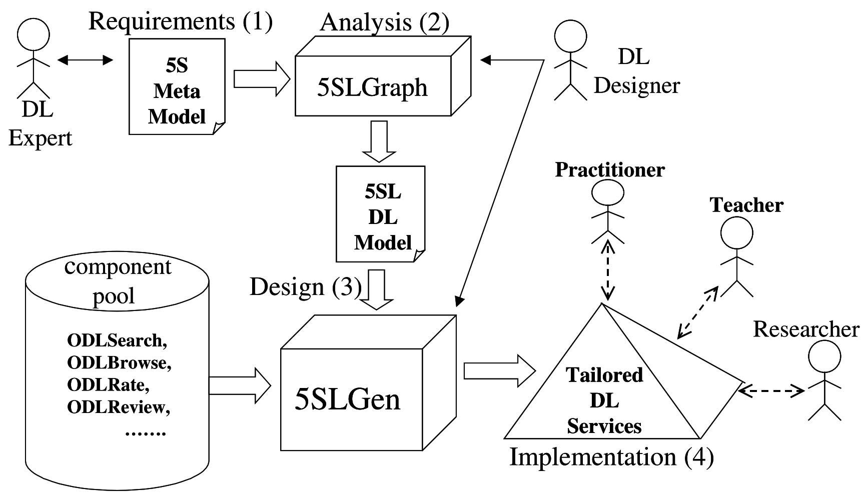


Figure 6.1: Overview of the architecture for DL modeling and generation

We adopt an approach shown to be highly effective in other areas of computing: develop powerful theories and (meta)models (i.e., 5S framework); use them to develop formal specifications (i.e., 5SL), and generate tailored systems from those specifications (using 5SLGen). We explain the approach in the context of the classical software engineering process (see Figure 6.1). During requirements gathering (see 1 in Figure 6.1) the DL designer captures all ‘societal’ conditions and capabilities to which the DL must conform. 5S provides a common ground terminology and domain model that is close to the DL world and furnishes precisely defined concepts so that the resulting description is understandable by end users. The role of the DL expert is to design a metamodel for DLs based on 5S, which will be used for modeling the DL. In the

[1]An earlier and different tool following similar principles but targeted towards a monolothical system (MARIAN) based on a object-oriented class hierarchy was presented in [94, 90]

analysis phase (see 2 in Figure 6.1), the requirements are formally captured in 5SL. The DL designer must be aware of functional requirements – what services a community needs and what form of interaction these services should have with the users of the DL: publishers, searchers, administrators, etc. Modeling such a complex system using only an XML-based language requires a great deal of knowledge of the 5S theory and language syntax. Accordingly, we introduced 5SGraph, a visual modeling tool that helps designers to model a DL instance without knowing the theoretical foundations and the syntactical details of 5SL. The focus of the design phase (see 3 in Figure 6.1) is to produce models that are closer to the implementation and the target architecture, but still preserve the structure of the system as captured by the analysis model. 5SLGen produces design models from 5SL models by transforming higher-level 5SL concepts into object-oriented classes and workflows. This transformation involves scenario analysis and scenario synthesis. Finally in the implementation phase (see 4 in Figure 6.1), 5SLGen uses the produced design models to generate running DL services by integrating components from pools, mapping models to specific target platforms and languages (e.g., Java, Perl), and compiling and producing new components and subsystems. This digital library generator, 5SLGen, is the focus of this chapter.

6.2 The 5SLGEN Digital Library Generator

6.2.1 The Tool Underlying Model

As argued in Chapter 3, we envision the services exposed by a DL to be either of the composite or elementary type. Elementary services provide the basic infrastructure for the DL. Examples include collecting, indexing, rating, and linking. Composite services can be composed of other services (elementary or composed) by reusing or extending them. For example, a relevance feedback service extends the capabilities of a basic search service while a lesson plan building service can use already existing searching, browsing, and binding services to find and organize relevant resources. The problem of composability of services has been studied recently, mainly in the Web community [7, 50]. However, DL services are restricted to certain specific types with constrained inputs and outputs, therefore making the problem more manageable and possible to be treated with domain specific techniques. Figure 6.2 shows a UML-based (meta-)model for the services exposed by the tailored DL produced by 5SLGen. The model defines composite services recursively as an aggregation of other services, composite or elementary. Elementary services do not rely on other services to fulfill their responsibilities while composite services act like umbrella structures that bring together other services, which collaborate to implement certain functionality. The application logic of a composite service is described by a workflow, i.e., a combination of control and data flows that mirror the behavior defined in the services scenarios, including invocations of other services. Statecharts [26] and Petri nets [167] are possible notations for formally representing workflows. In our implementations we chose statecharts to represent the workflow of a service. Statecharts, introduced by Harel [101], represent a compact way of describing the dynamic aspects of the system. Statecharts connect events and states. When an event is received, the system leaves its current state, initiates the actions specified for the transition and enters a new state. The next state depends on the current state as well as the event.

The distinct aspects of this model are: 1) the combination of an explicit workflow and service aggregation to support composite services; 2) the emphasis on scenario-based modeling of services and the automatic generation of workflows from them; and 3) the role of the service manager (a societal member) as the binding point for societal relationships, scenario interactions, and spatial visualizations. From an architectural and implementation point of view, point 1 becomes significant, since combining a small set of basic DL services and managers gathered from a pool of DL components should allow a designer to model and generate most digital libraries (at least from the behavioral point of view) with a minimum amount of coding. The only situations when coding is unavoidable are, for example, when a specific behavior of a composite service (e.g., Rocchio based expansion of a query in relevance feedback) is not implemented as

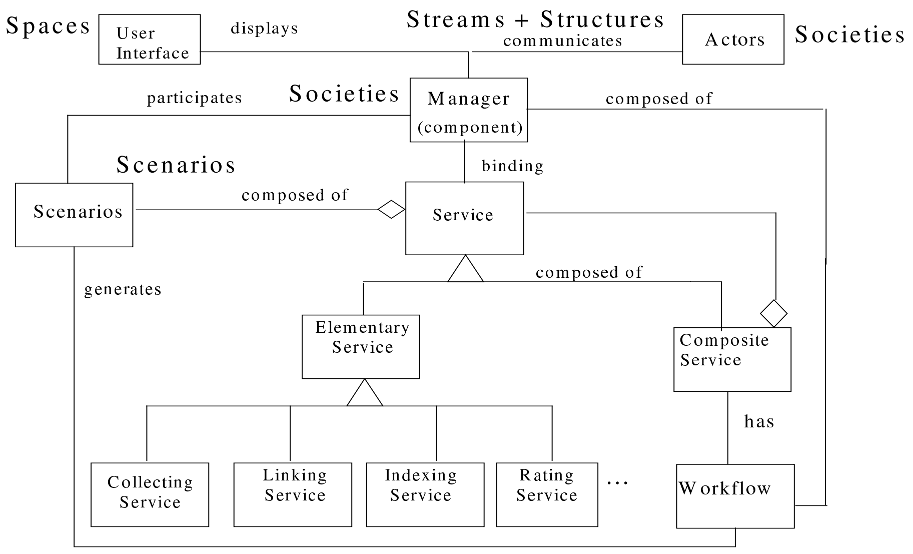


Figure 6.2: DL service composition pattern

operations by any component in the core pool or cannot be reused (e.g., due to incompatibility of interfaces). More importantly, the model also shows how the 5 Ss help when defining all components of a real, implemented DL. Services are implemented as components taken from the pool or automatically generated from the scenarios and their interactions/relationships. Service managers define the context or functionality of the service in terms of its operations and the data it expects, and are associated with a spatial (presentational) model of a user interface. It is interesting to notice the connections between the service manager roles and the classical Model-View-Controller (MVC) architecture of user interfaces [34], which explicitly separates functionality, behavior, and presentation and has helped facilitate the development of user interfaces that are modular and extensible. Service managers and actors communicate through streams (e.g., protocols) and structures (e.g., structured streams such as metadata specifications and digital objects). Finally the model provides the basic architectural underpinnings for the creation of DL generators, as described in Section 6.3.

6.2.2 Extensiblity and Reusability

In this section we present examples of two services, a Relevance Feedback Search service and a Lesson Plan Building service, implemented using 5SLGen. The services exposed follow the model explained in Figure 6.2, and illustrate reusability and extensibility. More formally a service Y reuses a service X if the behavior of Y incorporates the behavior of X. A service Y extends a Service X if it subsumes the behavior of X and potentially includes conditional sub-flows of events. We start each subsection with the main scenario for the particular service.

Extensibility: A Relevance Feedback Service

Scenario 1: Relevance feedback is a well known technique to improve quality of search services. A relevance feedback service extends a basic search service by allowing the user to choose from the results of a search the documents that are relevant. The selected relevant documents are then used by the Relevance Feedback Manager to construct an expanded query (using the Rocchio method, for example), which is then run to retrieve the next set of documents to be presented to the user. Figure 6.3 shows the relationship between the relevance feedback service and the search service (left part) and describes the relevance feedback scenario in terms of a UML sequence diagram (middle part). A sequence diagram focuses on the time ordering of events between members of societies. These members appear along the top margin of a dashed line that represents a timeline. Events can be associated with actions that the service managers perform to provide a given functionality. For the sake of brevity we do not show the corresponding 5SL modeling in XML. The ⟨⟨extends⟩⟩ relationship specifies that an instance of a relevance feedback search service includes the behavior of search service and adds specific events, subject to specific conditions (e.g., the set of relevant documents cannot be empty). The scenario shows that all the events associated with the basic search scenario occur in the relevance feedback scenario, with the addition of the expandQuery event and synchronous response. The statechart for the Relevance Feedback Manager derived from the scenario is shown in the figure too. There are only two states: the system transitions from the default to the .expanded query. state after reception of the expandQuery event (if the condition is true) and immediately transitions back to the default state where it can receive other requests.

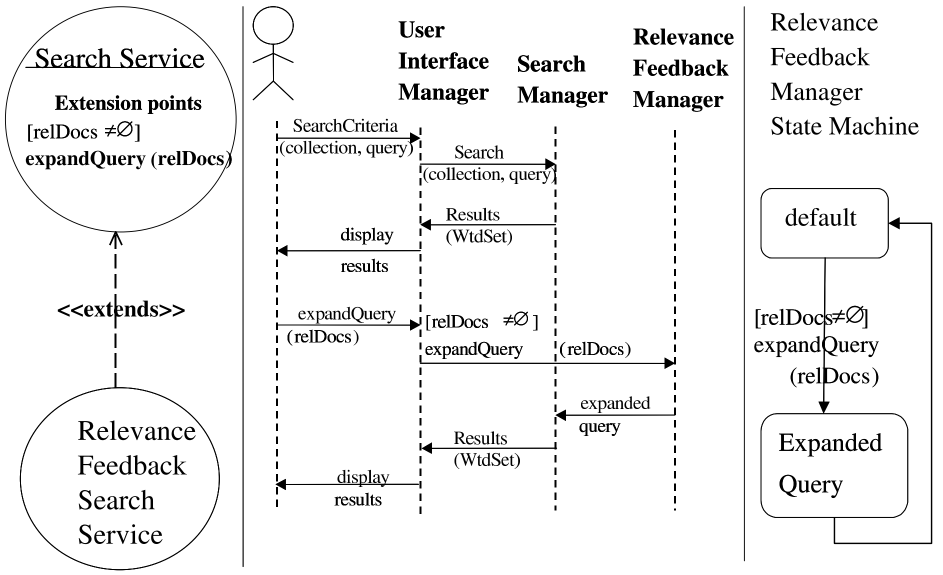


Figure 6.3: Relationships between services for relevance feedback

Reusability: Lesson Plan Building Service

Scenario 2: A lesson plan aggregates specific educational metadata and correlated resources (e.g., papers, simulations) available in the Computing and Information Technology Interactive Digital Electronic Library (CITIDEL) into a coherent package useful for some CS teaching activity. A specific service manager called VIADUCT supports this service, which can be used only by teachers registered with CITIDEL. To build a lesson plan, the teacher uses the information-seeking services of CITIDEL (i.e., searching and browsing) to look for relevant resources to a specific lesson, choses among those using any subjective criteria (e.g., by relevance, by date), assembles a number of the chosen resources together using a binding service, and associates descriptive metadata such as typical DC-based ones like author, identifier, language, as well as specific ones such as topic area, target audience, and time required for the whole lesson plan object. The teacher has to explicitly publish the lesson plan to allow students to view it. To allow a select group of people to view the lesson plan, the teacher saves the plan, returns to the main VIADUCT user information page, re-opens the project, and gives the project URL to whomever she wishes.

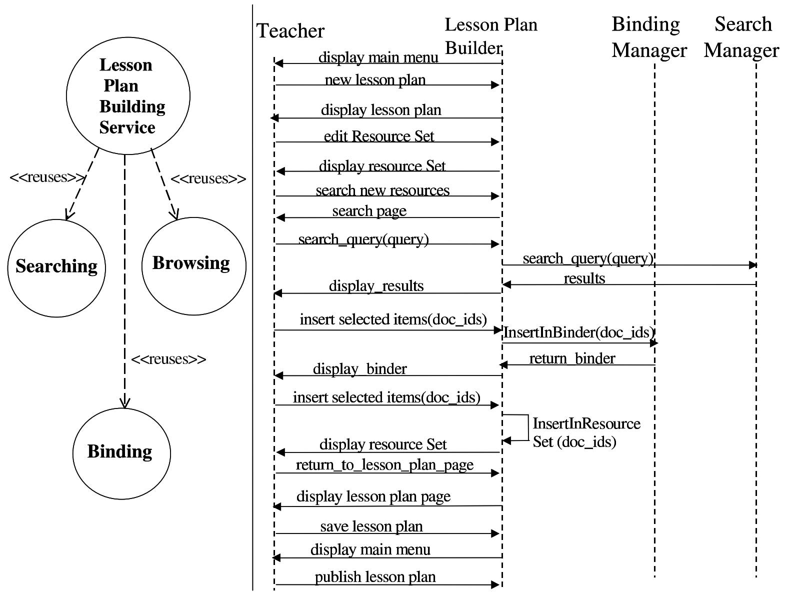


Figure 6.4: Relationships between the services for the lesson plan in VIADUCT

Figure 6.4 shows that the lesson plan building (LPB) service reuses three other services: Searching, Browsing, and Binding, as well as the main scenario of the LPB service. Figure 6.5 presents the statechart generated from scenario synthesis of the main scenario with other related scenarios of this service (not shown for brevity). The teacher starts the construction of a new lesson plan from the main menu (see 1 in Figure 6.5). The lesson plan edit page allows the teacher to fill out basic metadata about the plan and organize a number of related resources together (see 2 in Figure 6.5). To locate relevant resource the teacher can either search or browse (see 3 and 4 in Figure 6.5) the collection according to some criteria and sorting order. Having the results of an initial search/browse activity the user can either: 1) search for a similar document (see 5 in Figure 6.5); 2) browse a particular entry for details (see 6 in Figure 6.5); 3) perform another search (see 7 in Figure 6.5); or 4) select a number of items to put in her binder (see 8 in Figure 6.5). If the user chooses the latest option the binder is shown and she can transfers a number of resources from the binder to the resource set of the current plan (see 9 in Figure 6.5). Once the plan is ready the teacher can save it and publish to the students (see 10 and 11 in Figure 6.5).

6.3 5SLGen Architecture and Implementation

The architecture of 5SLGen is shown in Figure 6.6. The generated system is organized around the idea of a clean separation between service managers, that implement operations and carry data; views, for displaying all or a portion of the data; and controllers, for handling events that affect the data or the view(s) [34]. In

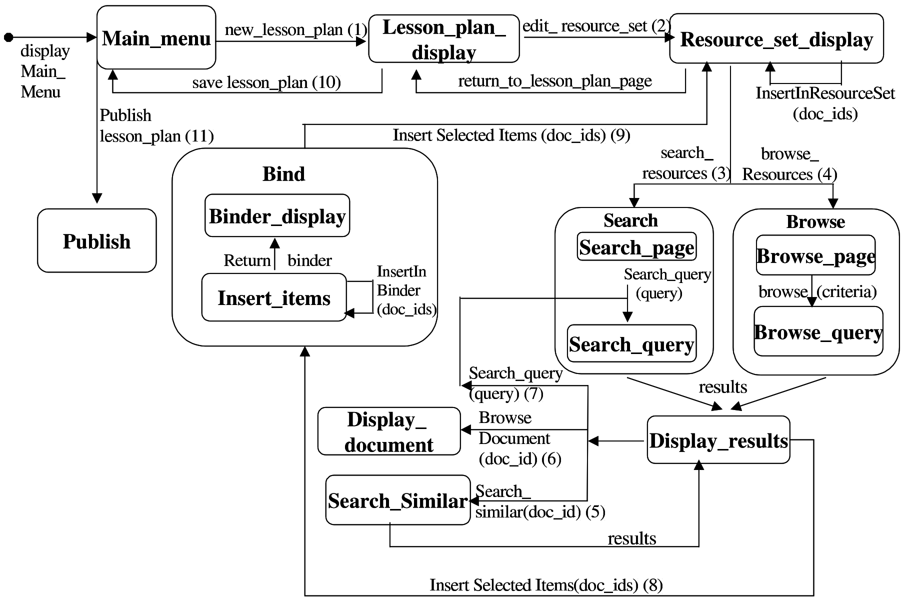


Figure 6.5: Statechart diagram of the VIADUCT system

the context of 5SLGen the service managers are either represented by one or more components in the pool or are generated from the 5SL-Societies model. The generated service managers may contain skeleton code for operations and capabilities not defined in any component of the pools; this code needs to be provided by the designer. Our current component pool consists of ODL components that communicate through a family of lightweight protocols based on OAI [150]. The ODL components, originally implemented in Perl, have been encapsulated through a Java interface, allowing them to be imported by the Java classes for the service managers. The controller maps onto the workflow of the system generated from the 5SL-Scenarios model. The view corresponds to the user interface presentation. The DL designer binds the presentation elements with the service managers to complete the implementation of the generated DL services.

This architecture for the generated DL services is achieved through the following process: The DL designer captures the structural and behavioral aspects of DL services through the 5SL-Societies model and 5SL-Scenarios model. 5SL-Societies captures the relationships among actors (those who use services) and services managers, whereas 5SL-Scenarios capture their dynamic interactions. The specifications of the DL captured in 5SL (- Societies and Scenarios) undergo a series of transformations (explained below) with the DL designer providing input at certain stages to generate the Java classes corresponding to the implementations of the service managers and the workflow of the DL. Once the presentation elements (views) are coupled with the controller, the generation of the tailored DL service is complete.

6.3.1 Generating Static Contextual Structure

The 5SL-Societies model is realized based on the relationships among actors and service managers and the set of operations that define the services. capabilities. In order to generate Java classes from the 5SL-Societies model we have chosen an intermediate step of transforming (see 2 in Figure 6.6) the 5SL-Societies XML Model into a XMI [154] representation model (see 3 in Figure 6.6) using the JDOM and XPATH XML APIs. XMI is an XML based industry standard to enable easy interchange of metadata between

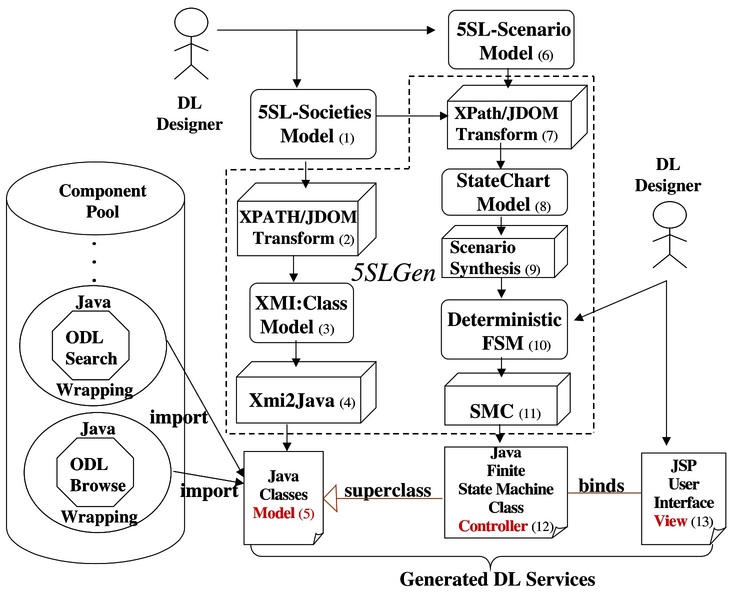


Figure 6.6: Architecture of 5SLGen based on MVC (expanding part of Figure 6.1)

modeling tools and between tools and metadata repositories. Many CASE tools serialize UML diagrams to XMI. Generation of XMI files for the 5SL-Societies model enables the exchange of the 5S-Societies model among various UML modeling tools supporting modeling as well as forward and reverse engineering. Moreover, existing freeware tools (see 4 in Figure 6.6) enable the generation of Java code from the serialized XMI Model (XMI2Java). We use an open source XMI2Java implementation to generate Java classes that implement the service managers (see 5 in Figure 6.6) for the generated DL.

6.3.2 Generating Dynamic Behavior

The 5SL-Scenarios model is used to capture the dynamic behavior of services (e.g., see figures 6.3 and 6.4) as scenarios. In order to describe the whole behavior of a DL service, a great multitude of scenarios is required to capture the complete set of possible societal interactions. Scenarios can contain other scenarios and in many cases are only small variations of others. This requires an approach for scenario integration in order to capture the whole behavior of the system. Also, in order to be able to generate an implementation from the scenarios, the level of abstraction needs to be reduced to a more concrete model in terms of computational actions and change of states that occur during scenario execution. These problems can be addressed by generating a statechart model (see 8 in Figure 6.6) from the scenarios (see 6 in Figure 6.6). The mapping from scenarios to statecharts is performed according to the following rules: For any object in a sequence diagram, incoming arrows represent events received by the object and they become transitions (see Section 3.1). Outgoing arrows are actions and they become actions of the transitions leading to the states. The intervals between events become states. The object starts in the default state specified in the 5SL-Societies model [208]. This transformation (see 7 in Figure 6.6) is achieved by parsing the 5SL-Scenarios modeled in XML with the JDOM and XPATH XML APIs [107, 42] and implementing the rules mentioned above. Again, since scenarios represent partial descriptions of the system behavior, an approach for scenarios composition is needed to produce a more complete specification of the service. As each scenario is mapped to a statechart we synthesize the statecharts derived in the previous step to perform scenario composition. The statecharts are synthesized (see 9 in Figure 6.6) according to the following rules [208]: 1) if a transition is common to the two statecharts, it is taken only once into the final statechart; 2) if at a certain moment in time either one or another scenario is executed, the statecharts are combined with sequential (object can be in only one state) substates within a composite state; and 3) if two scenarios are executed at the same time they are combined with concurrent substates (object can be in more than one state) within a composite state. A statechart extends traditional a finite-state machine (FSM) with notions of hierarchy and concurrency. A FSM represents a mathematical model of a system that attempts to reduce the model complexity thereby providing a powerful manner to describe the dynamic behavior of systems and components. The synthesized statechart (Figures 6.3 and 6.5) generated using the above rules represents the FSM/workflow for the DL service (see 10 in Figure 6.6). To generate code from the FSM we have extended an open source state machine compiler (see 11 in Figure 6.6) that compiles the annotated FSM to generate code (Java classes – see 12 in Figure 6.6) for the controller of the DL services. Before compilation the FSM is annotated by providing component specific implementation details by the DL designer (see Figure 6.6). There are many techniques of implementing state machines; the most common implies some sort of switch or if-else statements for implementing state dependent behavior; however this solution is not scalable; therefore we chose to implement FSM using the state design pattern from [79]. The state pattern localizes state-specific behavior in an individual class for each state, and puts all the behavior for that state in a single state object eliminating the necessity for a set of long, look-alike conditional statements. In the context of 5SLGen when the service manager class receives an event, it delegates the request to its state object, which provides the appropriate state specific behavior. The lesson plan building and relevance feedback service have been implemented using the above generation process. Manual intervention is required for: first annotating the FSM with component specific implementation details and second providing the views for the data. For more detais on implemenation, algorithms, etc., the reader is referred to [113]

As a proof of concept, 5SGen has been used to create prototypes of several DLs including CITIDEL, VIADUCT, the NDLTD Union Archive [113], and the Brazilian Digital Library in Computing [55].

6.4 Related Work

The first work to advocate a goal-oriented requirements analysis approach for digital libraries is [24], but that work does not propose any development tool or environment. The closest approach to our DL generator is the collection services and plug-in architecture of Greenstone [228]. However their architecture covers only portions of the Stream and Structural models of 5S with little support for modeling and generation of customized DL services (other than basic searching and browsing). While much attention has been paid to digital library architectures and systems, very few works tackle the problem of integrated DL design, conceptual modeling, and requirements gathering. Examples of work on DL architectures and systems include: monolithic systems (e.g., Greenstone [228], MARIAN [94], componentized architectures (e.g, [199], [37]), agent-based architectures (e.g., UMDL [224]), and layered architectures (e.g., Alexandria [72]). Our declarative/generative approach should be generalizable for any of those systems/architectures by taking whole or portions of those systems as part of our component pools. There is no reason why those systems and their components can not be incorporated in our component pools, given that they export clear, reusable software interfaces with accessible entry points. Most research done in the area of code generation from requirements has not been directed towards specific domains such as DLs. Most CASE tools do not address issues raised by research in scenario-based requirements analysis and design such as scenario generation, management of scenario descriptions, analysis and integration of scenarios, and bridging the gap between scenario descriptions and software designs and implementation. We have attempted to tackle the above problems and to bridge the model system gap through the 5S framework.

6.4.1 Systems Comparison

Digital Library systems can be compared in many ways. Table 6.1 shows a comparative analysis of several systems and DL generators based on the goals and requirements associated with the 5S family of tools.

Feature/ System	Greenstone	OpenDLib	EPrints	Dspace	5S*
Model-based	-	+	-	+	+
Theory-based	-	+	-	+	+
Support for Multiple Architectures	-	-	-	-	+
Graphical Environment for Modeling	+ -	-	-	-	+
Support for Requirement Analysis	+ -	+ -	+ -	+ -	+
Support for Workflows	-	+ -	+ -	+ -	+
Reusability	+ -	+	-	-	+
Extensibility	+ -	+	-	-	+
Large User Base	+	+	+	+	-

Table 6.1: Systems comparison (+ indicates support of a feature, - indicates lack of support, + - indicates partial support)

OpenDLib [37] and DSpace [209] are the only systems with a clear data model behind them. The former has a formal model for documents and metadata based on the notion of views and versions and a proprietary query language; the latter has a simple E-R model based on the notions of communities, collections, and aggregators. These are the only systems with a theoretical basis, albeit incomplete. The theoretical model for OpenDLib supports a set of axioms and constraints for distributed DLs, while DSpace is based on OAIS (Open Archival Information System), a non-formal reference model for digital repositories focused on preservation [175].

The 5S family of tools is the only one with potential to support multiple architectures. This has been proved by building generators for monolithic [94] as well as componentized architectures. This is due to the clear separation between the DL model, visualization, and code generation. The same DL model can be implemented in different systems and architectures given the corresponding generator. Such a separation has become a fundamental concept in databases and Web development, but has not been investigated widely in DL systems.

Greenstone has the best set of graphical interfaces for organization of resources and collections in a sense that is complementary to the 5S tools discussed in this work [14]. While the other systems have simple user interfaces for ingesting and revising resources, for administrative tasks, and for searching and browsing, interfaces for support of the complete modeling and personalization of all aspects of a digital library for specific communities do not exist in these systems.

Requirements analysis and workflow services are only partially supported by some of these systems. Eprints (www.eprints.org) and DSpace only allow the configuration of fixed workflows for ingesting and reviewing. OpenDLib is the most flexible in this sense but also does not support declarative description of the collective behavior of the DL (elsewhere called "services choreography [27]"). None of them supports requirements analysis for different types of services in the form of descriptions of scenarios of use and automatic generation of workflows based on the integration of these scenarios. For the same reasons, reusability and extensibility are only partially supported by Greenstone and OpenDLib, the latter being the most flexible and configurable of all these systems in terms of services.

On the other hand, all the above systems have a larger base of users than our tools, due mainly to the amount of financial and technical support they have.

Chapter 7

Logging: An XML Log Standard for DLs

Log analysis is a primary source of knowledge about how digital library patrons actually use DL systems and services and how systems behave while trying to support user information seeking activities. Log recording and analysis allows evaluation assessment and opens opportunities to improvements and enhanced new services. Indeed, the benefits of logging are numerous, including improving performance by recording effective evaluation data [16], helping in designing and testing of user interfaces [135], and better allocation of resources [111].

Conventional libraries have a long history of concern for privacy [133]. While circulation statistics are widely available, storage of patron-related information is rare in such libraries. The introduction of On-Line Public Access Catalogs (OPACs) has changed the picture and allowed some degree of log recording and analysis to improve library services [31, 180, 111, 158]. More recently, Web servers and proxy caching servers have made Web log analysis become common place, recording each and every access to their documents. These, along with the advance of techniques in Web log mining, have made possible a number of new and enhanced services such as customization and personalization [168].

Digital libraries differ from the Web in many ways. Firstly, digital library collections are explicitly organized, managed, described, and preserved. Secondly, Web sites and Web search engines assume very little about the users, tasks, and data they deal with. Digital libraries normally have much more knowledge of their users and tasks since they are built to satisfy specific needs of interested communities. And thirdly, the digital objects in DL collections tend to be much more structured than the information presented in the Web. Therefore, digital library logging should offer much richer information and opportunities. Despite the fact that many current DL systems do some kind of logging, they tremendously differ in the format in which they record the information and even the sort of information that is recorded. Interoperability, reuse of log analysis tools, and comparability of log analysis results are major problems.

In this chapter, we propose an XML-based standard digital library log format based on 5S that captures a rich, detailed set of system and user behaviors supported by current digital library services. The proposed standard is implemented in a generic log component tool, which can be plugged into any digital library system to produce the specified format. The focus of this work is on interoperability, reusability, and completeness. Specifications, implementation details, and examples of use are described.

This chapter is organized as follows. Section 7.2 covers related work and analyzes associated problems. Section 7.3 describes the DL log format and motivation for design. Section 7.4 presents the log tool, its implementation and some examples. Section 7.5 discusses the implementation of a prototype analysis tool. Section 7.6 shows some examples of real log entries.

7.1 Related Work

Most current Web servers store log files in the Common Log Format (CLF)- a simplistic format which reflects the stateless nature of the HTTP protocol by recording just individual server events. Apache, perhaps the most used Web server, uses an extension of CLF called Combined Log Format, which tries to keep some state information by recording the links between resources.

A sample of CLF is given in Figure 7.1 (from the csgrad.cs.vt.edu server). The fields are host; rfc931, i.e., information returned regarding identity of the person, otherwise '-'; authuser, if a userid is sent for authentication, otherwise '-'; day; month; year; hour; minutes; seconds; request; the first line of the HTTP request as sent by the client; ddd, status code returned by the server, otherwise '-'; and bbb, the number of bytes sent (not including the HTTP/1.0 header), otherwise '-'.

```
bbn-cache-3.cisco.com - - [22/Oct/1998:00:20:21 -0400] "GET
/~harley/courses.html HTTP/1.0" 200 1734
bbn-cache-3.cisco.com - - [22/Oct/1998:00:20:22 -0400] "GET
/~harley/clip_art/word_icon.gif HTTP/1.0" 200 1050
www4.e-softinc.com - - [22/Oct/1998:00:20:27 -0400] "HEAD
/ HTTP/1.0" 200 0
user-38ldbam.dialup.mindspring.com - - [22/Oct/1998:00:20:48 -0400] "GET
/~lhuang/junior/capehatteras.html HTTP/1.0" 200 328
user-38ldbam.dialup.mindspring.com - - [22/Oct/1998:00:20:48 -0400] "GET
/~lhuang/junior/PB2panforringed.mirror.gif HTTP/1.0" 200 20222
eger-dl01.agria.hu - - [22/Oct/1998:00:20:51 -0400] "GET
/~tjohnson/pinouts/ HTTP/1.0" 200 26994
```

Figure 7.1: Example of Apache log format

Distinct from simple Web servers, which focus primarily on browsing behavior, Web search engines and digital libraries also record data about search and other information seeking behaviors. The following (Figure 7.2))is a sample (from the VT Web site) of a query transaction submitted through the OpenText search engine. It shows the search terms and operations, but also records a good deal of internal cryptic information about how the system operates internally.

Digital library systems, most probably for historical reasons, usually implement logs that resemble Web log formats or utilize proprietary formats. As an example, Figure 7.3 shows an annotated sample of a portion of the log of the Greenstone digital library system [228]. Greenstone is a comprehensive, open-source digital library system, which enables logging by setting a specific flag in the configuration file. Each line in the sample user log contains: (a) the IP address of the user's computer; (b) a timestamp in square brackets; (c) the CGI arguments in parentheses; and, (d) the name of the user's browser (Netscape is called "Mozilla").

The last CGI argument, "z", is an identification code or "cookie" generated by the user's browser: it comprises the user's IP address followed by the timestamp when they first accessed the digital library. The log file usage.txt is placed in the /etc directory in the Greenstone file structure.

Other digital library log formats that we analyzed include those associated with the Dienst protocol (used by the old NCSTRL-Networked Computer Science Technical Reference Library [122]), and MARIAN digital library systems [94].

Problems with existing DL logs

A careful analysis of the logs of the Web and DL systems discussed above reveals a common set of problems. These include:

```
Mon Sep 28 17:48:42 1998
----- Starting Search -----
Mon Sep 28 17:48:42 1998
{Transaction Begin}
Mon Sep 28 17:48:42 1998
{RankMode Relevance1}
Mon Sep 28 17:48:42 1998
"Bacillus thuringiensis "
Mon Sep 28 17:48:42 1998
P0 = "Bacillus thuringiensis "
Mon Sep 28 17:48:42 1998
R = (*D including (*P0))
Mon Sep 28 17:48:42 1998
R = (((*R rankedby *P0)))
Mon Sep 28 17:48:42 1998
S = (subset.1.10 (*R))
Mon Sep 28 17:48:42 1998
SL0 = (region "OTSummary" within.1 (*S))
Mon Sep 28 17:48:42 1998
(*SL0 within.1 ( subset.1.1 *S ))
Mon Sep 28 17:48:42 1998
(*SL0 within.1 ( subset.2.1 *S ))
Mon Sep 28 17:48:42 1998
{Transaction End}
Mon Sep 28 17:48:42 1998
----- Ending Search -----
```

Figure 7.2: Example of OpenText log format

```
ADMINISTRATION 37
/fast-cgi-bin/niupepalibrary
(a) its-www1.massey.ac.nz
(b) [Thu Dec 07 23:47:00 NZDT 2000]
(c) (a=p, b=0, bcp=, beu=, c=niupepa, cc=, ccp=0, ccs=0, cl=, cm=,
cq2=, d=, e=, er=, f=0, fc=1, gc=0, gg=text, gt=0, h=, h2=, hl=1,
hp=, il=l, j=, j2=, k=1, ky=, l=en, m=50, n=, n2=, o=20, p=home,
pw=, q=, q2=, r=1, s=0, sp=frameset, t=1, ua=, uan=, ug=,
uma=listusers, umc=, umnpw1=, umnpw2=, umpw=, umug=, umun=, umus=,
un=, us=invalid, v=0, w=w, x=0, z=130.123.128.4-950647871)
(d) "Mozilla/4.08 [en] (Win95; I ;Nav)"
```

Figure 7.3: Example of Greenstone log format

1. **Disorganization:** Barring a few, most of the system logs were very poorly organized and structured.

2. **Complexity of analysis:** Lack of proper thought in recording the log information makes log analysis a hard problem. Indeed, complex data mining techniques are currently needed to extract some useful information from Web and similar types of logs [196, 140].

3. **Incompleteness:** Important information that would be necessary for analysis was omitted from some logs. As an example, most of the logs failed to record the client postal and email address, information that is essential in any user-based study of the system. Note , however, that for private reasons, such information may need to be kept separate, and have restricted access.

4. **Incompatibility:** Each of the systems had their own log formats, making it difficult to use the same tools to analyze logs from different systems for the same kind of study.

5. **Ambiguity:** Many of the log entries and their semantics were not properly and precisely specified in the log format itself, which could lead to ambiguity in analyzing them.

6. **Inflexibility:** The logs recorded a good deal of system specific information which would not be applicable to other systems. This information was recorded in conjunction with other information that was system independent.

7. **Verboseness:** Many of the logs looked just like code dumps used for debugging by the implementers of the system, rather than containing clear and precise self-describing information about system usage and behavior.

The above problems were found across virtually the whole set of logs that we analyzed. In the next section, we present our standardized digital library log format design, which attempts to solve many of those problems.

7.2 The Digital Library Standardized Log Format

As per the previous analysis, current Web and digital library logging has a number of problems. Our solution is to propose an XML-based DL standard format which is comprehensive, reflective of the actual DL system behavior, easily readable, precise, flexible to accommodate in varying systems, and succinct enough to be easily implemented.

7.2.1 DL Log Standard Design

As a first step in creating the DL log format, we collected an extensive, flat set of attributes that we felt were necessary to be recorded in the DL log. The next step was to organize these attributes in a fashion that was logical and structured and could be easily represented and implemented. We chose to produce an XML Schema [218] to describe the syntax and semantics of our DL log format. XML provides a standard syntax for the log format; different XML element tags represent different semantic attributes to be registered in the log. As a matter of fact, a similar use of XML to guarantee structural quality of Web logs is reported in [199]. XML Schema provides an equivalent to a grammar in XML syntax to specify the structure of the log format. Also, XML log files produced by our tool can be validated against the schema for correctness. Besides that, XML Schema has a rich set of basic types, such as those for numbers, dates, and times, which further contribute to standardization. And finally, the abundance of XML parsers and other related software helps in the construction of analysis tools.

The DL log format had to be reflective of how a generic DL system behaves. We achieve this goal in two ways:

1. By using the 5S framework as guidance for how to organize the log structure and define the semantics of the DL components whose behavior would be logged.

   To be useful, the DL log format has to be reflective of how a generic DL system behaves. Accordingly, we have designed and organized the log structure in accordance with our 5S framework. In 5S, services are composed of scenarios, which describe service behavior through sequences of specific user and system events. Since we are mostly interested in understanding user interactions and the perceived value of responses, we have chosen to record only the initial user input and final service output events along with corresponding parameters (modeled as XML sub-elements of events), and ignore most of the internal system communications (except administrative information). According to the 5S philosophy, extensions regarding new service events are to be modeled by analyzing user inputs and system outputs. Table 7.1 shows the current (bold) and in-development services and input events supported by the log format.

Service	**XML log event and sub-elements**
Searching	Search (Collection/Catalog, SearchBy(Field), QueryString)
Browsing	Browse(anchor, DocInfo(PathName,DocID))
Submitting	Update(AddInfo(DocInfo))
Annotating	Annotate (AnnotateInfo (AnnotationID, DocInfo))
Filtering	Filter (criteria (query,category), userId)

Table 7.1: Current and in-development services in the log format

2. By having the notion of a “transaction” as the basic unifying entity of the log format.

   Basically everything that occurs in a DL system could be broken down to the level of a transaction, either as interaction between users and the system or among the system components themselves. Simple examples of a transaction in our format would be a search query submitted by a user, the registering of a new user, or the recording of some system failure. This may be an isolated transaction in a system that does not have the notion of an explicit “session”, or it might be a part of a group of transactions that define a session. However, most of the current DL log formats, such as CLF, record just one or a few kinds of events or transactions. All or most of the entries in those log files have similar semantics. Our log format is designed to record a number of different kinds of transactions. Examples of distinct transactions are search, browse, session start, etc.

7.2.2 DL Log format structure

Figure 7.4 shows the higher-level organization of the DL log format. Each DL log file consists of a number of log entries, each entry representing a type of transaction. Transactions could be categorized as being related to session creation, user registration, user and system events associated with the use of DL services, administration activities, errors, and user-responses. An important and essential feature of the format should be to identify each transaction precisely. To achieve this, we record the timestamp at which it occurred and also associate a unique ID with each transaction. This ID should ideally be monotonically increasing across one server to provide a logical representation of successive transactions. Additionally, in case we are dealing with a non-session based system, we need a way to identify the user. One way to do this is to associate the location (IP address) from which the user is interacting with the system. Each transaction is then associated with a specific statement. A partial XML Schema of the high level organization is shown in Figure 7.5.

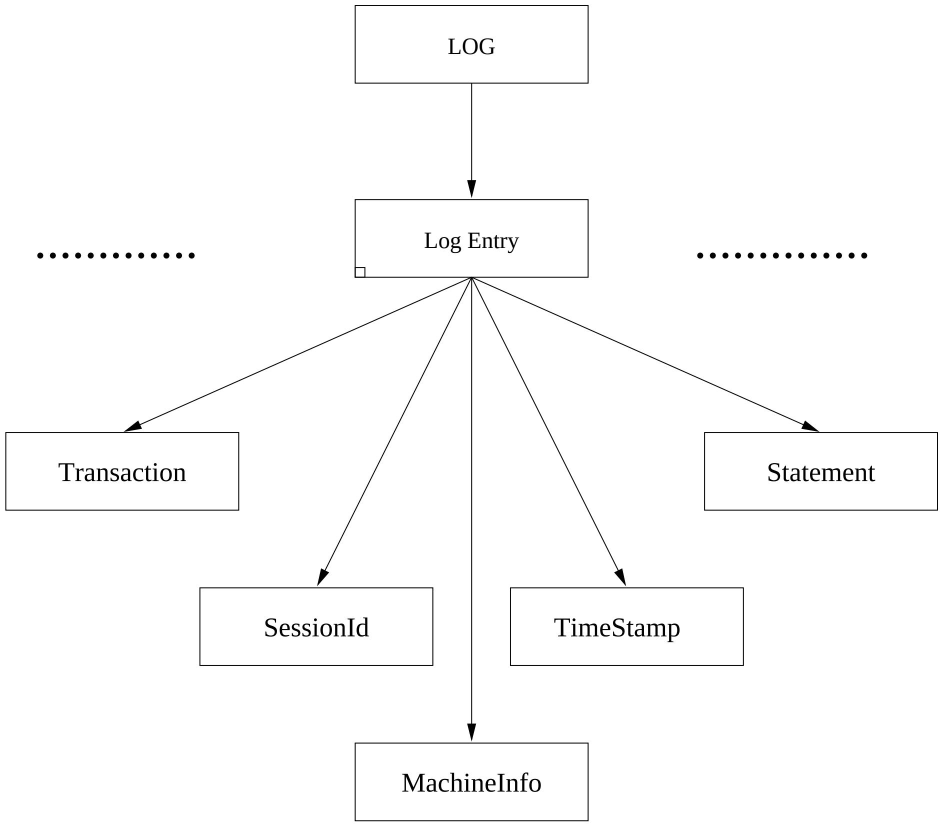


Figure 7.4: Top level hierarchy

There are basically two kinds of statements: 1) those related to specific user and system events associated to DL services; and 2) general statements related to administrative and other general activities. In more detail (Figure 7.6), the following types of statements are defined:

- **SessionInfo:** In the case of an explicit session based system, the session start and end times, as well as the user's and associated information, need to be recorded. We also assign a globally unique ID to each session. Using this ID, it is very easy to group together all the transactions that occur within this session.

- **Registration info:** In many session-based systems, users have to register themselves with the system when they use it for the first time. They usually have to select a user-ID, password, and possibly provide their identifying and demographic information. This information can latter be used to identoify the user and her patterns of usage of DL services.

- **Administration information:** Most systems record administration activities like system startup, shutdown, backup start, backup end, etc. This transaction type is provided to record such information.

- **Error Information:** This element is related to errors or failures that may occur anytime in the system. Invalid query, document not found, etc., are examples of error and failure information that need to be

```
<xsd:complexType name="LogType">
   <xsd:element name="LogEntry" minOccurs="0" maxOccurs="unbounded">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="Transaction">
         <xsd:complexType>
           <xsd:attribute name="ID" type="xsd:int/>
         </xsd:complexType>
       </xsd:element>
       <xsd:element name="TimeStamp" type="xsd:dateTime"/>
       <xsd:element name="MachineInfo">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="IPAddress" type="xsd:string"/>
             <xsd:element name="Port" type="xsd:int" minOccurs="0"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
       <xsd:element name="SessionId" type="xsd:string" minOccurs="0"/>
       <xsd:element name="Statement" type="StatementType"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:complexType>
```

Figure 7.5: XML schema for log format

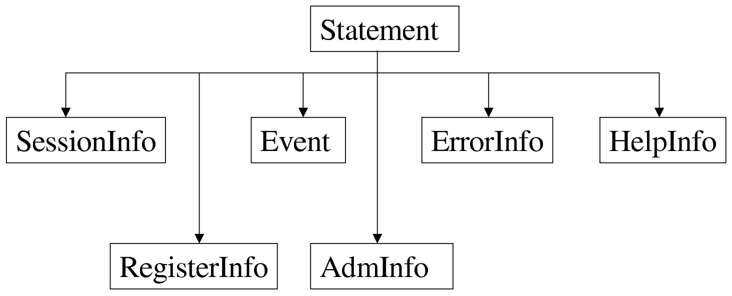


Figure 7.6: Decomposition of statement into different types

recorded. If the user forgets to explicitly logout in a session based system the automatic system time out can be recorded.

- **Help Information:** Some DL systems provide help facilities to aid the user. Use of this feature should be considered to be separate from the other actions described above. Our log format considers this to be another type of transaction. It can be an interesting investigation to find out which kinds of help are frequently used.

- **Event:** We consider this to be the heart of the DL log format. User or system events occur as a result of users performing information seeking activities and using digital library services, or as a system response to those activities. Each event is associated with an action type, which encompasses the main operations associated with DL services such as searching, browsing, updating, and recording of system information related to these three operations. Each of those actions is performed over a collection of digital objects or a catalog of metadata. User events also have a status code that is based on the outcome of the action (e.g., success, failure, etc.). Four different kinds of actions are currently defined (Figure 7.7):

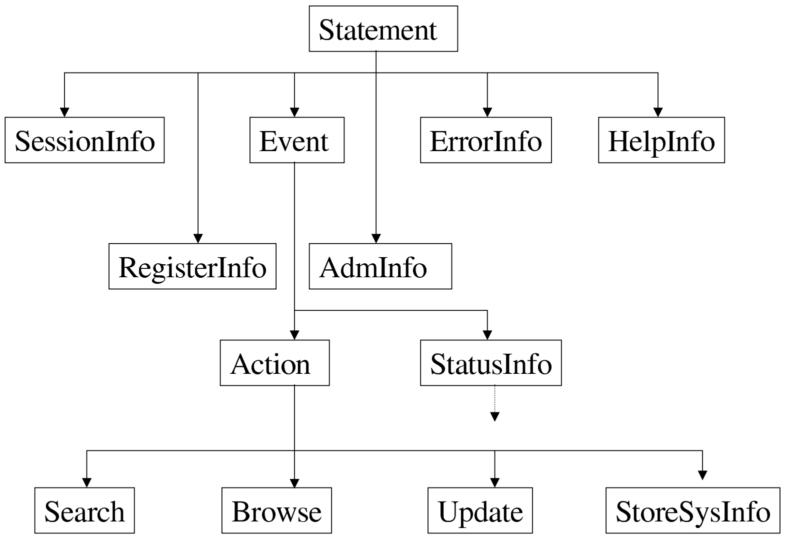


Figure 7.7: Decomposition of an event into different types

1. **Search:** Searching is a fundamental DL service. Different systems implement a number of different query languages and search schemes based on the underlying retrieval model they use. Two of the common models are boolean and ranked retrieval [13, 108]. Each of these systems also can provide additional features like selection of collection(s), structure related information such as which fields the search concerns (author, title, subject, etc.), and some contextual information such as the duration of activities, or some way to indicate whether this search operation is to be performed in the context of a previous, larger search or some profile or filter. Systems also can provide options to the users to select how they want to view the results from their queries,

including sort option and maximum number of results to be presented. The details of the search element are presented in the portion of the Log Schema below (Figure 7.8).

Specific types of objects can be searched include generic digital objects and metadata records. SearchBy is used in structured queries and covers specific (metadata/structural) fields under which the query will be performed. The value of SearchType is set to persistent if the search is to be performed over the result of a previous search. Since query syntax is heavily dependent on the specific DL system and underlying retrieval model, we only record the exact query string used. Log analysts will consider this information in the context of the particular system for their studies. It is obvious that here we are considering an extended set of of inputs/outputs for the Searching service events than those considered in Tables 3.1-3.4, more specifically metadata specifications and catalogs, and structured fields (i.e., labels for nodes of structures). Here the goal is completeness and abundance of information while there was simplicity, precision, and correctness. The PresentationInfo includes contextual and user's preference information such as presentation format (e.g., list, threaded, tabular), which type of sort to apply (e.g., estimated relevance, by a specific field), number of results per page, and cut off threshold.

2. **Browse:** Browsing services can be performed by navigation through lists of search results, indexes organized by specific fields, and generic hypertexts. In the browse section we include identifiers of nodes or anchor's text for links navigated, targeted documents, and presentation information.
3. **Update:** Some systems also provide facilities to allow an administrator or user to add, modify or edit some part(s) of collections and/or catalogs resident in a repository. This corresponds to the submitting services in Tables 3.1-3-4.
4. **Store information:** This action allows us to record the data associated with the search and browse actions from the point of view of the system. So, basically actions 1, 2 and 3 above record the user's data, while this action records the system's response data. After any action the system needs to record some information like number of bytes transferred, response time of the action, highest and lowest ranked item, etc.

7.3 DL Log Tool and its Implementation

7.3.1 The First Generation

The first version of the DL XML Log Tool was implemented using generic Java classes and was designed to be used by any digital library or analysis system. There were mainly two classes in the log tool implementation, XMLLogData.java, used for storing data, and XMLLogManager.java, which provided methods to write and read log information according to our DL log format. XMLLogData.java basically provided a structure to hold private data with Set and Get methods to set and get values. For example it had one attribute String SessionInfo for session-based systems and it had SetSessionInfo() and GetSessionInfo() methods to set and get the value of SessionInfo. All the read and write methods were synchronized to avoid conflicts and inconsistences. The most difficult part was how to plug-in the tool into the target system. That should be done by calling specific methods of the XMLLogManager wherever a specific type of transaction occurs. Since this was heavily dependent on the target system architecture and implementation, that should be done by developers or administrators.

First tests were performed on the MARIAN digital library system [94]. MARIAN had a resource management mechanism, which administered and allocated all the system resources such as class managers and searchers. In the MARIAN system, we only had one XMLLogManager Java object in memory, created as

```
<xsd:complexType name="SearchType">
  <xsd:sequence>
    <xsd:element name="Collection"
                 type= "xsd:string" minOccurs="0"/>
    <xsd:element name="MetadataCatalog" type= "xsd:string" minOccurs="0"/>
    <xsd:element name="ObjectType">
       <xsd:complexType>
        <xsd:element name="DigitalObject"
                     type="xsd:string" minOccurs="0"/>
        <xsd:element name="MetadataRecord"
                     type="xsd:string" minOccurs="0"/>
       </xsd:complexType>
    </element>
    <xsd:element name="SearchBy"
                 type= "xsd:string" minOccurs="0"/>
    <xsd:element name="SearchType">
       <xsd:complexType>
        <xsd:element name="persistent"
                     type="xsd:string" minOccurs="0"/>
        <xsd:element name="non-persistent"
                     type="xsd:string" minOccurs="0"/>
       </xsd:complexType>
    <xsd:element>
    <xsd:element name="QueryString"
                 type= "xsd:string" minOccurs="0"/>
    <xsd:element name="TimeOut"
                 type= "xsd:string" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="StartDate" type="xsd:date"/>
          <xsd:element name="EndDate" type="xsd:date"/>
        </xsd:sequence>
      </xsd:complexType>
    </element>
    <xsd:element name="PresentationInfo"
                 type= "PresentationInfoType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Figure 7.8: XML schema for log format (Search)

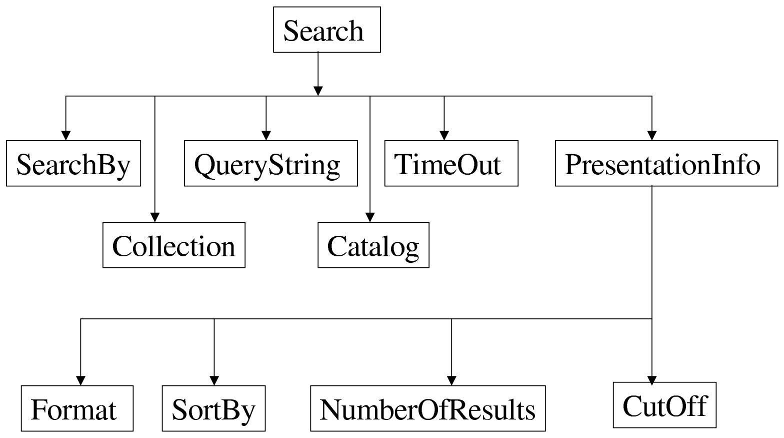


Figure 7.9: Search attributes

an attribute of the ResourceClassManager. Whenever information needed to be logged the client called the corresponding method of the XMLLogManager instance of the ResourceManager.

7.3.2 The Second Generation

The first version had a monolithic architecture, which was strongly coupled within the target system. Whenever DL events needed to be logged, the client invoked the corresponding methods of the log tool, since specific calls had been inserted within the target system. This implementation revealed two major drawbacks: 1) small changes in the format required complex changes of the DL logger code and complete recompilation of the tool and target system, therefore preventing extensibility; and 2) the Java-based implementation and close coupling required a deep understanding of the target tool architecture and caused problems in connecting the tool with DLs implemented in other languages (e.g., Perl), therefore preventing wide-spread adoption.

Our second generation implementation solved those problems by 1) re-implementing the tool with an OO hierarchical, bottom-up design that mimics the organization of the XML schema of the log format, therefore making internal communications clearer and isolating points of communication and modification; and 2) de-taching the tool from the target DL system by using connectionless sockets. For socket communication, we devised a simple, ad-hoc datagram packet format.

7.4 Examples

We have included examples of some log transactions in MARIAN captured from real use of the system. In the examples, we use the Dirline collection, a U.S. National Library of Medicine's online digital library containing location and descriptive information about a wide variety of information resources including

```
<Transaction ID = "3452">
   <SessionId > 987654usr3 </SessionId>
   <SessionInfo>
      <SessionStart> Start </SessionStart>
      <LoginInfo>
         <UserId> mhabib <UserId>
      </LoginInfo>
   </SessionInfo>
   <TimeStamp> 2002-05-31T20:10:55.000-05:00 </TimeStamp>
   <MachineInfo>
      <IPAddress> 128.173.244.56 <IPAddress>
      <Port> 8000 </Port>
   </MachineInfo>
</TransId>
```

Figure 7.10: XML log entry - example 1

organizations and projects concerned with health and biomedicine.

1. Login to the System:

2. Query on all Dirline records enties about “low back pain” in any part of the record.

3. Browse an item of the ranked list returned as a answer for the previous search.

7.5 Discussion

As expected with any newly proposed standard, evolution to cope with results of the early stages of experimentation is expected. Accordingly, our formats and tools have evolved to deal with the results of such experiments. With the interest demonstrated by many DLs and institutions (e.g., CiteSeer [125], MyLibrary [141], Daffodil [76]) in adopting the format and tools, we expect soon to release stable versions of both. Once this phase is achieved, other research issues will become the focus of future efforts, such as richer analysis and evaluation, and efficient use of distributed storage.

```
<Transaction ID = "3455">
  <SessionId > 987654usr3 </SessionId>
   <TimeStamp> 2002-05-31T20:11:07.000-05:00 </TimeStamp>
   <MachineInfo>
     <IPAddress> 128.173.244.56 <IPAddress>
     <Port> 8000 </Port>
   </MachineInfo>
  <Statement>
    <Event>
      <Action>
        <Search>
         <Collection>Dirline</Collection>
         <ObjectType>CommunityRecord</ObjectType>
         <SearchBy>SearchByAnyParts</SearchBy>
         <SearchType>NonPersistant</SearchType>
         <QueryString>low back pain</QueryString>
         <TimeFrame>
           <StartTime>2002-05-31T20:11:07.000-05:00</StartTime>
           <EndTime>2002-05-31T20:11:09.000-05:00</EndTime>
         </TimeFrame>
         <PresentationInfo>
           <Format>List</Format>
           <SortBy>ByRank</SortBy>
           <NumberOfResults>217</NumberOfResults>
           <Cutoff>20</Cutoff>
         </PresentationInfo>
       </Search>
     </Action>
     <StatusInfo>successful</StatusInfo>
   </Event>
  </Statement>
</Transaction>
```

Figure 7.11: XML log entry - example 2

```
<Transaction ID = "3456">
  <SessionId > 987654usr3 </SessionId>
  <TimeStamp> 2002-05-31T20:11:15.000-05:00 </TimeStamp>
  <MachineInfo>
    <IPAddress> 128.173.244.56 <IPAddress>
    <Port> 8000 </Port>
  </MachineInfo>
  <Statement>
    <Event>
      <Action>
        <Browse>
          <DocID> 5114 </DocID>
          <DocName>University of Washington School of
           Medicine Multidisciplinary Pain Center ( UWPC )
          </DocName>
        </Browse>
     </Action>
    </Event>
  </Statement>
</Transaction>
```

Figure 7.12: XML log entry - example 3

Part III

Quality

Chapter 8

Defining a Quality Model for Digital Libraries

In this chapter, we elaborate on the meaning of quality in digital libraries (DLs) by proposing a quality model which is deeply grounded in the 5S formal theory for digital libraries (see Chapters 2-3). We move the theory one step further to define: "What is a good digital library?". For each major DL concept and ontological relationship we can formally define a number of dimensions of quality and propose a set of numerical indicators for those quality dimensions. In particular, we consider key concepts of a minimal DL: catalog, collection, digital object, metadata specification, repository, and services. Regarding quality dimensions, we consider: accessibility, accuracy, completeness, composability, conformance, consistency, effectiveness, efficiency, extensibility, impact factor, pertinence, preservability, relevance, reliability, reusability, significance, similarity, and timeliness. Regarding measurement, we consider characteristics like: response time (with regard to efficiency), cost of migration (with respect to preservability), and number of service failures (to assess reliability). For key DL concepts, each pair, of a quality dimension and its corresponding numerical indicator, can be illustrated through application to a number of "real-world" digital libraries such as the ACM Digital Library [4], the Computing and Information Technology Interactive Digital Educational Library (CITIDEL) [41], and the Networked Digital Library of Theses and Dissertations (NDLTD) [144].

We also discuss connections between the proposed dimensions of DL quality and an expanded version of a workshop's consensus view of the life cycle of information in digital libraries [28]. Such connections can be used to determine when and where quality issues can be measured, assessed, and improved — as well as how possible quality problems can be prevented, detected, and eliminated.

The main contributions of this work are twofold:

1. a proposal of new formalizations for quality dimensions and indicators for digital libraries in the context of our 5S framework; and

2. a contextualization of these indicators within the DL information cycle.

8.1 Introduction

What is a good digital library? As was pointed out by Fuhr et al. [75], the answer to this question depends on whom you ask. It can be considered that what differentiates a good DL from a not so good one is the quality of its services and content. Since one of the main goals of our work with the 5S formal theory is to try to answer (at least partially) the question of "What is a digital library," our hypothesis in this chapter is that further development on the theory will allow us to define critical dimensions and indicators of DL quality. In contrast to its physical counterpart, the "digital" nature of DLs allows automatic assessment and enforcement of those quality properties, therefore supporting prevention and elimination of quality problems. 5S gives a standard terminology to discuss these issues in a common framework. Moreover, the formal nature of our DL theory allows us to add precision as we define specific DL quality dimensions and corresponding numeric indicators.

In this chapter, we will follow the standard terminology used in *social sciences* [11]. We will use the term *composite quality indicator* [1] (or in short quality indicator) to refer to the proposed equations instead of the stronger term *quality measures*. Only after one has a number of indicators, and they are validated [2] and tested for reliability [3], can they be composed into reliable "measures". Despite partial tests of validity (for example, through focus groups [4]) the proposed quality indicators do not qualify as measures yet. Also, it should be stressed that the proposed equations are only approximations of or give quantified indication of a quality dimension. They should not be interpreted as a complete specification of a quality dimension, since more factors/variables can enter in their compositions which were not specified here. We will, however, reserve the right to use the term "measure" when talking about standard measures that have long been used by the CS / LIS communities. The distinction should be clear in context.

Table 8.1 shows a summary of candidate dimensions of quality for some of the most important DL concepts and factors affecting the measurement of the corresponding quality dimensions. Most of these dimensions of quality can be characterized in the context of several DL semantic relationships defined in our Digital Library Ontology (see Chapter 3). Table 8.2 shows some of the S concepts and ontological relationships involved in the definition of quality dimensions and indicators [5]. The following subsections explain these indicators in detail by:

1. motivating them and discussing their meaning/utilization;
2. formally defining them and specifying their corresponding numerical computation; and
3. illustrating their use by applying the indicators/metrics in the context of some "real-world" DLs (e.g., ACM, CITIDEL, NDLTD).

This chapter is organized as follows. Sections 8.2 through 8.5 present all the dimensions of quality, the proposed indicators, and their applications to the respective DL concepts. Section 8.6 deals with the connections between the proposed dimensions and Borgman et al.'s Information Life Cycle [28]. Section 8.7 covers related work and Section 8.8 concludes the chapter.

8.2 Digital Objects

8.2.1 Accessibility

A digital object is accessible by a DL actor or patron, if it exists in the collections of the DL, the repository is able to retrieve the object, and: 1) an overly restrictive rights management property of a metadata specification does not exist for that object; or 2) if such exists, the property does not restrict access for the particular society to which the actor belongs or to that actor in particular. A quality indicator for calculating accessibility is a function, which depends on all those factors and the granularity of the rules (e.g., entire

[1] An indicator composed of two or more simpler indicators or variables.

[2] According to [11], validity refers to the extent to which a specific measurement provides data that relate to commonly accepted meanings of a particular concept. There are numerous yardsticks for d etermining validity: face validity, criterion-validity, content validity, and construct validity.

[3] Also according to [11], reliability refers to the likelihood that a given measurement procedure will yield the same description of a given phenomena if that measurement is repeated.

[4] A type of face validity.

[5] Each S in which the DL concept in the first column is inserted also is involved.

DL Concept	Dimension of Quality	Factors/Variables in Measuring
Digital object	Accessibility	Collection, # of structured streams, rights management metadata, communities
	Pertinence	Context, information, information need
	Preservability	Fidelity (lossiness), migration cost, digital object complexity, stream formats
	Relevance	Term (feature) frequency, inverse document frequency, document size, document structure, query size, collection size
	Similarity	Same as in relevance, citation/link patterns
	Significance	Citation/link patterns
	Timeliness	Age, time of latest citation, collection freshness
Metadata Specification	Accuracy	Accurate attributes, # of attributes in the record
	Completeness	Missing attributes, schema size
	Conformance	Conformant attributes, schema size
Collection	Completeness	Collection size; size of the 'ideal collection'
	Impact Factor	Size of the collection; number of citations
Catalog	Completeness	# of digital objects without a set of metadata specifications; size of the described collection
	Consistency	# of sets of metadata specifications per digital object
Repository	Completeness	# of collections
	Consistency	# of collections in repository; Catalog/Collection pairwise consistency
Services	Composability	Extensibility, reusability
	Efficiency	Response time
	Effectiviness	Precision/recall (search); F1 measure (classification)
	Extensibility	# of extended services; # of services in the DL; # of. lines of code per service manager
	Reusability	# of reused services; # of services in the DL; # of lines of code per service manager
	Reliability	# of service failures; # of accesses

Table 8.1: DL high-level concepts and corresponding DL dimensions of quality with respective metrics

DL Concept	**Dimension of quality**	**Some ‘S’ Concepts and Relationships Involved**
Digital object	Accessibility	Societies(actor), Structures (metadata specification), Streams + Structures (structured streams)
	Pertinence	Societies (actor), Scenarios (task)
	Preservability	Streams, Structures (structural metadata), Scenarios (process (e.g., migration))
	Relevance	Streams + Structures (structured streams), Structures (query), Spaces (Metric, Probabilistic, Vector)
	Similarity	Same as in relevance, Structures (citation/link patterns)
	Significance	Structures (citation/link patterns)
	Timeliness	Streams (time), Structures (citation/link patterns)
Metadata Specification	Accuracy	Structure (properties, values)
	Completeness	Structure (properties, schema)
	Conformance	Structure (properties, schema)
Collection	Completeness	none
	Impact Factor	Structure (citation/link patterns)
Catalog	Completeness	Structure (describes(Collection))
	Consistency	Structure (describes(Collection))
Repository	Completeness	Structure (describes(Collection))
	Consistency	Structure (describes(Catalog, Collection))
Services	Composability	see Extensibility, reusability
	Efficiency	Streams (time), Spaces (operations, contraints)
	Effectiviness	see Pertinence, Relevance
	Extensibility	Societies + Scenarios (extends, inherits_from, redefines)
	Reusability	Societies + Scenarios (includes, reuses)
	Reliability	Societies + Scenarios (uses, executes, invokes)

Table 8.2: Dimensions of quality and Ss involved in their definitions

object; structured streams). It should be noted that digital object accessibility as defined here is different from the common view of “Web site accessibility”, which is concerned with creating better ways to provide Web content to users with disabilities [202, 174].

Let *access constraint* be a property of some metadata specification of a digital object do_x whose values include the sets of communities that have the right to access specific (structured) streams within the object. Also let $struct\_streams(do_x) = \bigcup_{t \in do_x(4)} t$ be the set of structured streams of do_x. The accessibility $acc(do_x, ac_y)$ of a digital object do_x to an actor ac_y is:

- 0, if there is no collection C in the DL such that $do_x \in C$;
- otherwise $acc = (\sum_{z \in struct\_streams(do_x)} r_z(ac_y))/|struct\_streams(do_x)|$, where:
  - $r_z(ac_y)$ is a rights management rule defined as an indicator function:
    - 1, if
      - z has no access constraints; or
      - z has access constraints and $ac_y \in cm_z$, where $cm_z \in Soc(1)$ is a community that has the right to access z; and
    - 0, otherwise

Example of application. Virginia Tech’s Digital Library of Electronic Theses and Dissertations.

At Virginia Tech, a student can choose, at the moment of submission, to allow her electronic thesis or dissertation to be viewed worldwide, by the originating university, or not at all. The “mixed” case occurs when some portions (e.g., particular chapters or multimedia files) have restricted access while others are more widely available. The majority of Virginia Tech students choose their documents to be viewable worldwide [69] – but some initially choose not to grant worldwide access, because of concerns regarding patents or publication of results in journals/conferences. To address this concern, there are ongoing discussions with publishers to help them understand the goals and benefits of NDLTD (http://www.ndltd.org/publshrs/). An additional concern is faculty advisors. Author exit surveys indicate that many of the authors who chose to restrict access based their decisions on the advice of their faculty committee. Anecdotal evidence indicates that committee chairs feel very protective of these future academicians and are concerned about their chances of publishing in other venues.

Therefore the accessibility $acc(etd_x, ac_y)$ of a Virginia Tech ETD etd_x is:

- 0, if etd_x does not belong to the VT-ETD collection;
- otherwise $(\sum_{z \in struct\_streams(etd_x)} r_z(ac_y))/|struct\_streams(etd_x)|$, where:
  - $r_z(ac_y)$ is a rights management rule defined as an indicator function:
    - 1, if
      - etd_x is marked as “worldwide access” or
      - etd_x is marked as “VT only” and $ac_y \in VT_{cmm}$, where VT_{cmm} is the community of Virginia Tech ID holders accessing z through a computer with a Virginia Tech registered IP address.
    - 0, otherwise

Table 8.3 shows the number of unrestricted (worldwide, accessibility = 1 to everybody), restricted to VT campus (accessibility = 0 worldwide, 1 to members of VT_{cmm}), mixed, along with the degree of accessibility $acc(etd_x, ac_y)$ of the mixed ETDs for non-VT_{cmm} members ac_y, as of March 25, 2003. For example, five out of the six chapters (structured streams) of the third mixed ETD under the letter A were available only to VT. The rights management rule therefore is 0 for all those chapters, thus making its overall acessibility to non-VT actors 1/6 or 0.167.

First letter of author's name	**Unrestricted**	**Restricted**	**Mixed**	**Degree of accessibility for users not in the VT community**
A	164	50	5	mix(0.5, 0.5, 0.167, 0.1875, 0.6)
B	286	102	3	mix(0.5,0.5, 0.13)
C	231	108	7	mix (0.11, 0.5, 0.5, 0.5, 0.33, 0.09, 0.33)
D	159	54	2	mix(0.875, 0.666)
E	67	26	1	mix(0.5)
F	88	39	2	mix(0.375, 0.09)
G	166	72	2	mix(0.666,0.5)
H	225	91	3	mix(0.66, 0.5, 0.235)
I	20	8	1	mix(0.5)
J	84	36	2	mix(0.5, 0.6)
K	166	69	2	mix(0.5, 0.5)
L	189	68	6	mix(0.153, 0.33, 0.5, 0.5, 0.94)
M	299	115	9	mix(0.5, 0.5, 0.5, 0.041, 0.5, 0.5, 0.5, 0.117, 0.5)
N	74	16	1	mix(0.8)
O	45	19	2	mix(0.5, 0.125)
P	172	71	3	mix(0, 0, 0.33)
Q	13	6	0	mix = none
R	158	71	3	mix(0.66, 0.5, 0.5)
S	398	159	8	mix(0.66, 0.5, 0.5, 0.6, 0.33, 0.66, 0.33, 0.6)
T	111	49	1	mix(0.13)
U	9	7	0	mix = none
V	63	20	0	mix = none
W	191	81	5	mix (0.5, 0.22, 0.38, 0.875, 0.5)
X	11	5	0	mix = none
Y	38	9	3	mix(0.5, 0.5, 0.125)
Z	47	17	2	mix(0.5, 0.5)
All	3474	1368	73	

Table 8.3: Accessibility of VT-ETDs (first column corresponds to the first letter of author's name)

8.2.2 Pertinence

Pertinence is one of the most "social" quality indicators since it is a relation between the information carried by a digital object and an actor's information need, and depends heavily on the actor's knowledge, background, current task, etc.

Let $Inf(do_i)$ represent the "information" [6] (not physical) carried by a digital object or any of its (metadata) descriptions, $IN(ac_j)$ be the information need [7] of an actor and $Context_{jk}$ be an amalgam of societal factors which can impact the judgment of pertinence of do_i by ac_j at time k. These include, among others, task, time, place, the actor's history of interaction, and a range of other factors that are not given explicitly but are implicit in the interaction and ambient environment. A complete formalization of context is out of the scope of this work. The reader is referred to a recent workshop on "Context in Information Retrieval" for a number of papers on the subject [105].

Also, let's define two sub-communities of actors, *users* and *external-judges* $\subset Ac$, as:

- *users*: set of actors with an information need who use DL services to try to fulfill/satisfy that need
- *external-judges*: set of actors responsible for determining the relevance (see Section 8.2.4) of a document to a query. Let's also constrain that a member of external-judges can not judge the relevance of a document to a query representing her own information need, i.e., at the same point in time $users \cap$ *external-judges* $= \emptyset$.

The pertinence of a digital object do_i to a user ac_j at a time k is an indicator function[8] $Pertinence(do_i, ac_j) : Inf(do_i) \times IN(ac_j) \times Context_{jk}$ defined as:

- 1, if $Inf(do_i)$ is judged by ac_j to be informative with regards to $IN(ac_i)$ in context $Context_{jk}$;
- 0, otherwise.

Since pertinence is an implicit, subjective judgment made by a *user* in a particular context it can ultimately only be assessed by the user herself.

Example of use. We will use pertinence as a quality indicator to evaluate the effectiveness of information satisfaction services (see Section 8.5.1).

8.2.3 Preservability

Preservability is a quality property of a digital object which reflects a state of the object that can vary due to changes in hardware (e.g., new recording technologies), software (e.g., release of a new version of the software used to create/display the object), representation formats (e.g., new image standard such as JPEG 2000), and processes to which the object is submitted (e.g., migration).

The four main technical approaches to digital preservation are:

1. Migration: transforming from one digital format to another format, normally a successive subsequent one (e.g., from JPEG to JPEG 2000) [48]

[6] Information and information need, by themselves, are hard notions to formally define. One comprehensive attempt is presented in [138].

[7] Certain authors such as Taylor [210] and Mizzaro [139] make a distinction between the "real" and the "perceived" information need. We will not make this distinction here, in the interest of brevity.

[8] Voorhees [214], Greisdorf [98], and others argue for non-binary pertinence/relevance functions. We will leave such extensions for future work.

2. Emulation: re-creating the original operating environment by saving the original programs and or creating new programs that can emulate the old environment [173];

3. Wrapping: packaging the object to be preserved with enough human readable metadata to allow it to be decoded in the future [223];

4. Refreshing: copying the stream of bits from one location to another, whether the physical medium is the same or not [127].

Note that here we are not considering physical deterioration of the medium in which the object is stored since this is a property of the medium itself, not the object. However we acknowledge that this is an important problem, for which "refreshing" is the normally used approach [8].

For cost, operational, and technical reasons, migration is the most widely used of the three techniques mentioned above [223]. However the ideal solution should be some combination of all the techniques [223, 107]. One example which applies such a combination is the UVC-based approach [131]. Nonetheless, for the purpose of the discussion below we will concentrate on migration issues.

A digital object's preservability can be affected by its obsolescence and the fidelity of the migration process (see Figure 8.1). Obsolescence reflects the fact that a very obsolete object is really hard and costly to migrate, given the difficulty of finding appropriate migration tools and the right expertise. Fidelity reflects the differences between the original and the migrated object or, in other words, reflects the distortion or the loss of information inherent in the migration process that is absorbed by the object. The more obsolete and the less faithful the migration process, the lower the object's preservability. Preservability also is affected by contextual issues of specific DLs. For example, while it is desirable to always use the most faithful migration process, a DL may not have sufficient resources (money, storage, personnel) to apply it to its digital objects during migration. Based on the above discussion and on the fact that these two factors are orthogonal, we can define the preservability of a digital object do_i in a digital library dl as a tuple:

$$preservability(do_i, dl) = (\textsf{fidelity of migrating}(do_i, format_x, format_y), obsolescence(do_i, dl)). \tag{8.1}$$

As mentioned before, obsolescence is a complex notion to capture, that depends on many contextual factors. Since the choice of how to deal with obsolescence generally depends on resources at the disposal of the DL, one possible idea is to approximate its value by the actual cost of migrating the object [181]. While a complete cost model for preservability/obsolescence is beyond the scope of this work, we recognize many factors that can impact the cost, including:

- Capital direct costs
  - Software development/acquiring or license updating for newer versions
  - Hardware (for preservation processing or storage)
- Indirect Operating Costs
  - Identifing candidate digital objects
  - Evaluating/examining/negotiating intellectual property issues and rights
  - Process
  - Storage
  - Staff training (on software / procedures)

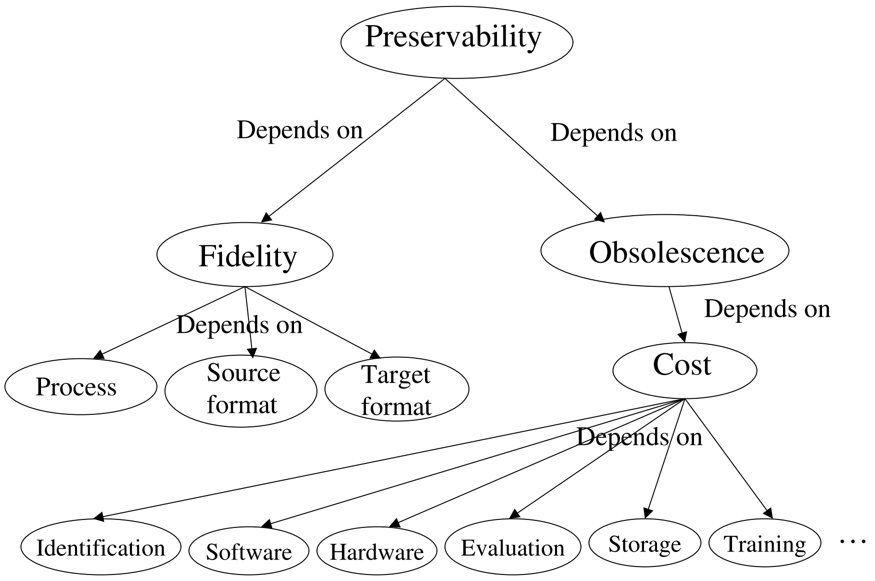


Figure 8.1: Factors in preservability

Obsolescence then can be defined as obsolescence(do_i, dl) = cost of converting/migrating the digital object within the context of the specific dl.

The fidelity of the migration process p of a digital object do_i from $format_x$ to $format_y$ can be defined as the inverse of the distortion or noise introduced by the process p, i.e., fidelity(do_i, $format_x$, $format_y$) = $\frac{1}{distortion(p(format_x, format_x))+0.5}$. Distortion can be computed in a number of ways depending on the type of object and transformation [184]. One very common measure, when converting between similar formats, is the *mean squared error (mse)*. In the case of a digital object, *mse* can be defined as follows. Let $\{x_n\}$ be a stream of a digital object do_i and $\{y_n\}$ be the converted/migrated stream; the mean squared error $mse(\{x_n\}, \{y_n\})$ = $\frac{1}{N} * \sum_{n=1}^{N}(x_n - y_n)^2$, where N is the size of the streams. The mean square error for the whole object do_i can be calculated as the average of *mse* for all its streams.

Example of Use. Let's consider the following scenario adapted from [104]. In 2004, a librarian receives an email notifying her that a special collection [9] of 1,000 digital images, stored in TIFF version 5.0, is in danger of becoming obsolete, due to the fact that the latest version of the display software no longer supports TIFF 5.0. The librarian decides to migrate all digital photos to JPEG 2000 which now has become the *de facto* image preservation standard, recommended by the Research Libraries Group (RLG).

The librarian does a small search for possible migration options and finds a tool, costing \$500, which converts TIFF 5.0 directly to JPEG 2000. Let's consider that the amount of time taken by the librarian and the system administrator to order the software, install it, learn it, and apply it to all digital images combined takes 20 hours. Assume also that the hourly rate in this DL is \$66.6 per hour per employee [10]. In order to

[9]Preservation of a collection, instead of a digital object, also may involve preserving all the structures (e.g., classification schemes, etc.) imposed on the collection.

[10]1800 is the number of hours in a work-year (37.5 hrs/wk * 48 wks/yr) and \$110,000 the total annual cost of an employee working for this DL, based on salary, benefits, expenses, etc.

save space, the librarian chooses to use in the migration a compression rate which produces an average *mse* of 9 per image. In this scenario, the preservability of each digital image would correspond to: preservability (image-TIFF 5.0, dl) = (1/9, ($500 + $66.6 * 20) /1000) = (0.11, $1.83).

8.2.4 Relevance

A digital object is *relevant* [182] in the context of an expression of an information need (e.g., a query) or interest (e.g., profile) and a service (e.g., searching, recommending). A role of an information satisfaction service is to provide ways to find the most relevant information for the user, which in case of DLs is carried by digital objects and their metadata specifications.

The relevance of a digital object to a query is an indicator function $Relevance(do_i, q)$ defined as:

- 1, if do_i is judged by a ***external-judge*** to be relevant to q;
- 0 otherwise

The most common measures for relevance estimates/predictions are based on statistical properties of the streams of the digital object such as term (feature) frequency and the collection (e.g., collection size, inverse document frequency). For example, to estimate the relevance as $rel(do_i, q)$ of a document do_i to a query q in the vector space model (see Chapter 2), the cosine of the angle between the vectors representing the two entities is normally used. Therefore

$$rel(do_i, q) = \frac{do_i^{\rightarrow} \times q^{\rightarrow}}{|do_i^{\rightarrow}| * |q^{\rightarrow}|} \tag{8.2}$$

Note that differently from pertinence, relevance is a relation between a **representation** of a document and a **representation** of an information need (i.e., query). Also, it is supposed to be an objective, public, and social notion that can be established by a general consensus in the field, not a subjective, private judgment between the actor and her information need [64, 115].

Example of use. Examples can be found in any information retrieval work using the vector space model [178, 13]. There are literally thousands of these, the most prominent being published in the annual ACM SIGIR conference proceedings and in journals such as the ACM Transactions on Information Systems.

8.2.5 Significance

Significance of a digital object can be viewed from two perspectives – relative to its relevance to a user need (as in Section 8.2.4) or in absolute terms, irrespective of particular user requirements. Absolute significance can be calculated based on *raw citedness* – the number of times a document do_i is cited, or the frequency of occurence of citations whose target is do_i. Other factors may play a role in the significance of a document such as the prestige of the journal publishing the work, its use in courses, awards given, etc., but these are very hard to quantify/measure.

Example of Use. The ACM Digital Library

We used 98,000 documents from ACM DL, which corresponded to approximately 1,093,700 (outgoing) citations (average of 11.53 citations per document). Table 8.4 shows the top 9 documents in the ACM collection with the highest values of significance while Figure 8.2 shows the distribution of significance values in the same collection.

Document	**Publication**	**Year**	**Significance**
Computer programming as art	CACM	1974	279
A generalized processor sharing approach to flow control in integrated services networks: the single-node case	IEEE/ACM Transactions on Networking (TON)	1993	138
The entity-relationship model – toward a unified view of data	ACM Transactions on Database Systems	1976	130
A relational model of data for large shared data banks	CACM	1970	121
Revised report on the algorithmic language scheme	ACM SIGPLAN Notices	1986	116
Parallel discrete event simulation	CACM	1990	108
Can programming be liberated from the von Neumann style?: a functional style and its algebra of ...	CACM	1994	107
A case for redundant arrays of inexpensive disks (RAID)	ACM SIGMOD Record	1988	107
Time, clocks, and the ordering of events in a distributed system	CACM	1978	105

Table 8.4: Documents with highest degree of significance

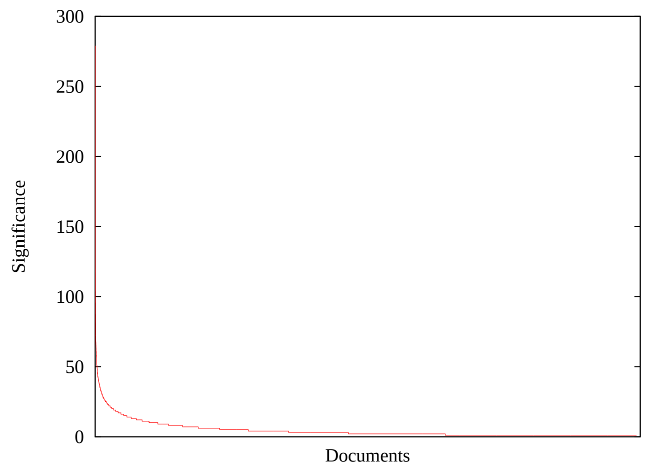


Figure 8.2: Significance in the ACM Digital Library

8.2.6 Similarity

Similarity metrics reflect the relatedness between two or more digital objects. Similarity can be measured based on the digital object's content (streams) (e.g., use and frequency of words), the digital object's internal organization (structures), or citations/linking patterns. For example, similarity between two documents can be calculated using a similar idea as for relevance estimates in Section 8.2.4, where a document vector (and its term frequency vector) substitutes for the query vector. This idea can be expanded to calculate similarity between corresponding structured streams of documents (e.g., using their title and abstract texts). Other measures can be used to calculate similarity as well. Assuming again that documents are represented as vectors of terms or features, two such measures are "bag-of-words" and Okapi, both defined below. For vectors d_i and d_j, $d_i \cap d_j$ is the set of terms t that are components of both d_i and d_j. $|d_i|$ means the dimension of the vector d_i (number of terms in the document).

$$\textit{Bag-of-words}(d_i, d_j) = |d_i \cap d_j| / |d_j| \tag{8.3}$$

$$Okapi(d_i, d_j) = \sum_{t \in d_i \cap d_j} \frac{3 * tf(t, d_j)}{0.5 + 1.5 * \frac{len(d_j)}{len_{avg} + tf(t, d_j)}} * \log \frac{N - df(t) + 0.5}{df(t) + 0.5} * tf(t, d_i) \tag{8.4}$$

In Equation 8.4, $tf(t, d)$ is the frequency of term t in document d and $df(t)$ is the document frequency of the term t in the whole collection. N is the number of documents in the whole collection, $len(d)$ is the length of document d, and len_{avg} is the average length of all documents in the collection. Note that both measures are asymetric.

Similarity measures also may use link or citation information to compute the relatedeness of two objects. Among the most popular citation-based measures of similarity are: co-citation, bibliographic coupling, and the Amsler measure, all of which we explain below.

Co-citation was first proposed by Small [193]. Two documents are co-cited if a third paper has citations to both of them. This reflects the assumption that the author of a scientific paper will cite only papers related to his own work. The extension for digital objects and hyperlinks is straightforward. More formally, let d_i be a digital object and let Pd_i be the set of documents that cite or link to d_i, called the parents of d_i. The co-citation similarity $cocit(d_i, d_j)$ between two documents d_i and d_j is defined as:

$$cocit(d_i, do_j) = \frac{|Pd_i \cap Pd_j|}{max|P|} \tag{8.5}$$

where $max|P|$ is the maximum number of parents for any object in the whole collection. If both Pd_i and Pd_j are empty, we define the co-citation similarity as zero. Equation 8.5 tells us that, the more parents d_i and d_j have in common, the more similar they are. However, co-citation is a measure between pairs of digital objects. The absolute degree of co-citation of document d_i in collection C is defined as $\sum_{d_j \in C - \{d_i\}} cocit(d_i, d_j)$.

Also with the goal of determining how related two documents are, Kessler [116] introduced the measure of bibliographic coupling. Two documents share one unit of bibliographic coupling if both cite a same document. The idea is based on the notion that authors who work on the same subject tend to cite the same documents. More formally, let d_i be a digital object. We define Cd_i as the set of documents that d_i links to, also called the children of d_i. Bibliographic coupling $bibcoup(d_i, d_j)$ between two pages d_i and d_j is defined as

$$bibcoup(d_i, d_j) = \frac{|Cd_i \cap Cd_j|}{max|C|} \tag{8.6}$$

where $max|C|$ is the maximum value of children for any object in the whole collection. According to Equation 8.6, the more children in common document d_i has with document d_j, the more related they are.

This value is normalized by the total set of children, to fit between 0 and 1. If both Cd_i and Cd_j are empty, we define the bibliographic coupling similarity as zero. The absolute degree of bibliographic coupling of a document d_i in collection C is defined as $\sum_{d_j \in C - \{d_i\}} bibcoup(d_i, d_j)$.

Finally, in order to take advantage of both types of information, Amsler proposed a measure that combines co-citation and bibliographic coupling. According to Amsler, two documents d_i and d_j are related if (1) d_i and d_j are cited by the same document, (2) d_i and d_j cite the same document, or (3) d_i cites a third document d_k that cites d_j. Thus, let Pd_i be the set of parents of d_i, and let Cd_i be the set of children of d_i. The Amsler similarity between two pages d_i and d_j is defined as:

$$Amsler(d_i, d_j) = \frac{|(Pd_i \cup Cd_i) \cap (Pd_j \cup Cd_j)|}{max(|P \cup Cd|)} \tag{8.7}$$

Eq. 8.7 tells us that, the more links (either parents or children) d_i and d_j have in common, the more they are related. The absolute Amsler degree of a document d_i in collection C is defined as $\sum_{d_j \in C - \{d_i\}} Amsler(d_i, d_j)$.

Examples of use. The ACM Digital Library and Automatic Classification

First, we illustrate the use of the citation-based similarity measures. Tables 8.5, 8.6, and 8.7 show the top 9 documents in the ACM collection with the highest absolute values of co-citation, bibliographic coupling, and Amsler, respectively.

Document	**Publication**	**Year**	**Cocit**
A unified lattice model for static analysis of programs by construction or approximation of fixpoints	4th ACM SIGACT-SIGPLAN	1977	37.97
Active messages: a mechanism for integrated communication and computation	19th annual int. symposium on Computer architecture	1992	36.92
Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers	17th Annual Int. Symposium on Computer Architecture	1990	31.12
Computer programming as an art	CACM	1974	30.07
The SPLASH-2 programs: characterization and methodological considerations	22nd Annual Int. symposium on Computer architecture	1995	29.36
ATOM: a system for building customized program analysis tools	ACM SIGPLAN '94	1994	27.90
Analysis of pointers and structures	Proceedings of the conference on Programming language design and implementation	1990	27.53
Revised report on the algorithmic language scheme	ACM SIGPLAN Notices (Issue)	1986	25.99
The directory-based cache coherence protocol for the DASH multiprocessor	17th annual international symposium on Computer Architecture	1990	25.87

Table 8.5: Documents with highest absolute degree of co-citation

Figures 8.2(a), (b), (c) show the distribution of absolute values (sorted by decreasing magnitude) of co-citation, bibliographic coupling, and Amsler in the ACM DL collection. It can be seen, as expected, that the values for Amsler and bibliographic coupling are higher than for co-citation due to the fact that documents cite more than they are cited. It also can be seen that the values drop really quickly in all measures.

Similarity measures can be used within a number of DL services. One example is automatic classification. Table 8.8 shows the performance of several classifiers based on the nearest neighbor approach and different types of similarity, using the macro-F1 measure (see Section 8.5.1)[11]. The collection utilized was

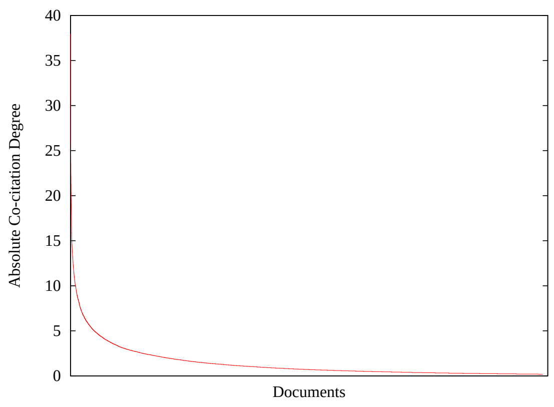
[11]Macro-F1 is computed by averaging the F1 value obtained by the classifiers for each candidate category.

Document	**Publication**	**Year**	**Bibcoup**
Query evaluation techniques for large databases	CSUR	1993	70.49
Compiler transformations for high-performance computing	CSUR	1994	63.82
On randomization in sequential and distributed algorithms	CSUR	1994	47.77
External memory algorithms and data structures: dealing with massive data	CSUR	2001	45.29
A schema for interprocedural modification side-effect analysis with pointer aliasing	TOPLAS	2001	44.69
Complexity and expressive power of logic programming	CSUR	2001	44.39
Computational geometry: a retrospective	ACM symposium on Theory of computing	1994	41.29
Research directions in object-oriented database systems	ACM SIGACT-SIGMOD-SIGART symposium	1990	40.17
Cache coherence in large-scale shared-memory multiprocessors: issues and comparisons	CSUR	1993	35.07

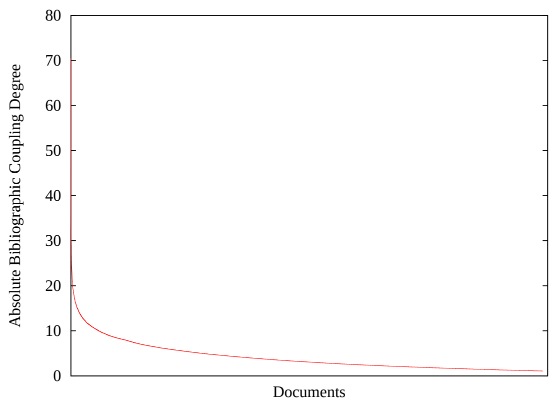
Table 8.6: Documents with highest absolute degree of bibliographic coupling

Document	**Publication**	**Year**	**Amsler**
Computer programming as an art	CACM	1974	69.15
Compiler transformations for high-performance computing	CSUR	1994	64.31
Analysis of pointers and structures	Prog. language design and implementation	1990	62.56
Query evaluation techniques for large databases	CSUR	1993	59.81
A schema for interprocedural modification side-effect analysis with pointer aliasing	TOPLAS	2001	57.90
Context-sensitive interprocedural points-to analysis in the presence of function pointers	ACM SIGPLAN ’94	1994	56.59
Efficient flow-sensitive interprocedural computation of pointer-induced aliases and side effect	20th ACM SIGPLAN-SIGACT	1993	55.72
Efficiently computing static single assignment form and the control dependence graph	TOPLAS	1991	54.50
Extensibility safety and performance in the SPIN operating system	ACM Symp. Operating systems principles	1995	53.84

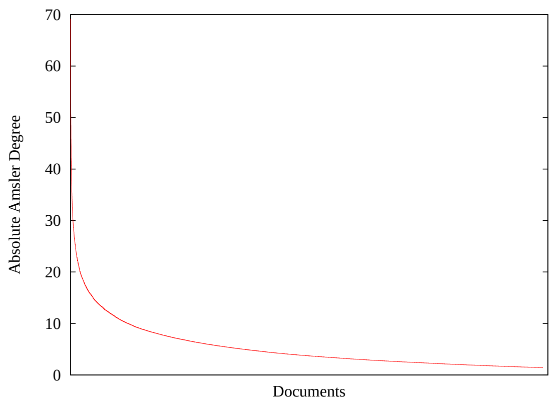
Table 8.7: Documents with highest absolute Amsler degree



(a)



(b)



(c)

Figure 8.3: Distribution of absolute values of citation-based similarity in the ACM DL

a subset of the ACM digital library (approximately 30,000) documents, in particular those classified under only one category of the ACM classification scheme (first level, 11 categories), with 30% of the collection used for training and the remainder for testing. These classifiers assign a category label to a test document based on the categories attributed to the k most similar documents in the training. The most widely used algorithm was introduced by Yang and is referred to as *kNN* [232]. The *kNN* algorithm was chosen since it is simple and makes direct use of similarity information. In the *kNN* algorithm, to a given test document d_i is assigned a score associating d_i with each candidate category c. This score is defined as

$$\sum_{d' \in Nk(d_i)} similarity(d_i, d') * f(c, d') \tag{8.8}$$

where $Nk(d_i)$ are the k nearest neighbors of d_i (the most similar documents in the training set) and $f(c, d')$ is an indicator function that returns 1 if document d' belongs to categocy c, and 0 otherwise. For the similarity function we used 10 different measures:

1. similarity based on the bag-of-words method using only the title of the document (tit_bag),
2. similarity based on the cosine method using only the title of the document (tit_cos),
3. similarity based on the Okapi measure using only the title of the document (tit_okapi),
4. similarity based on the bag-of-words method using only the abstract of the document (abs_bag),
5. similarity based on the cosine method using only the abstract of the document (abs_cos),
6. similarity based on the Okapi measure using only the abstract of the document (abs_okapi),
7. similarity based on the three citation-based measures.

Evidence	Macro F1(30%)
Abstract_BagOfWords	0.195
Co-citation	0.273
Abstract_Okapi	0.339
Abstract_Cosine	0.343
Bib_Coup	0.347
Amsler	0.412
Title_BagOfWords	0.492
Title_Cosine	0.525
Title_Okapi	0.525

Table 8.8: Macro F1 on individual evidence

From Table 8.8 it can be seen that, for this collection, the best similarity measures for use in the kNN algorithm are the title-based ones, followed by the Amsler measure. Surprisingly, for this task, these measures are significantly better than the measures based on abstracts.

8.2.7 Timeliness

Timeliness of a digital object is the extent to which it is sufficiently up-to-date for the task at hand [159]. It can be a function of the time when the digital object was created, stored, accessed, or cited.

Since the timeliness of an object is directly related to the information it carries, which can still be timely even if the object is “old”, a good quality indicator of this quality dimension is the time of the latest citation, since it’s a measure that:

1. captures the fact that the information carried by the object is still relevant by the time the citing object was published;
2. is independent from the actor that receives the object and the time the object is delivered;
3. reflects the overall importance of the object inside its community of interest.

As it is known that many documents are never cited, an alternative is to consider the age of the object itself. Therefore the timeliness of a digital object do_i can be defined as:

- (current time or time of last freshening) - time of the latest citation, if object is ever cited
- age = (current time or time of last freshening) - (creation time or publication time), if object is never cited.

Time of last freshening, which is defined as the time of the creation/publication of the most recent object in the collection to which do_i belongs, may be used instead of current time if the collection is not updated frequently.

Example of use. ACM Digital library

Figure 8.5 shows the distribution of degree of timeliness (0 through 10) for documents in the ACM Digital Library with citations. Time of last freshness is 2002, representing years. It can be seen, discounting the first set of values (timeliness=0), that there is a inverse relation between timeliness/size of the set of documents with that value: the smaller the value, the bigger the set, meaning that as time passes there is less chance that a document will be cited.

8.3 Metadata Specifications and Metadata Format

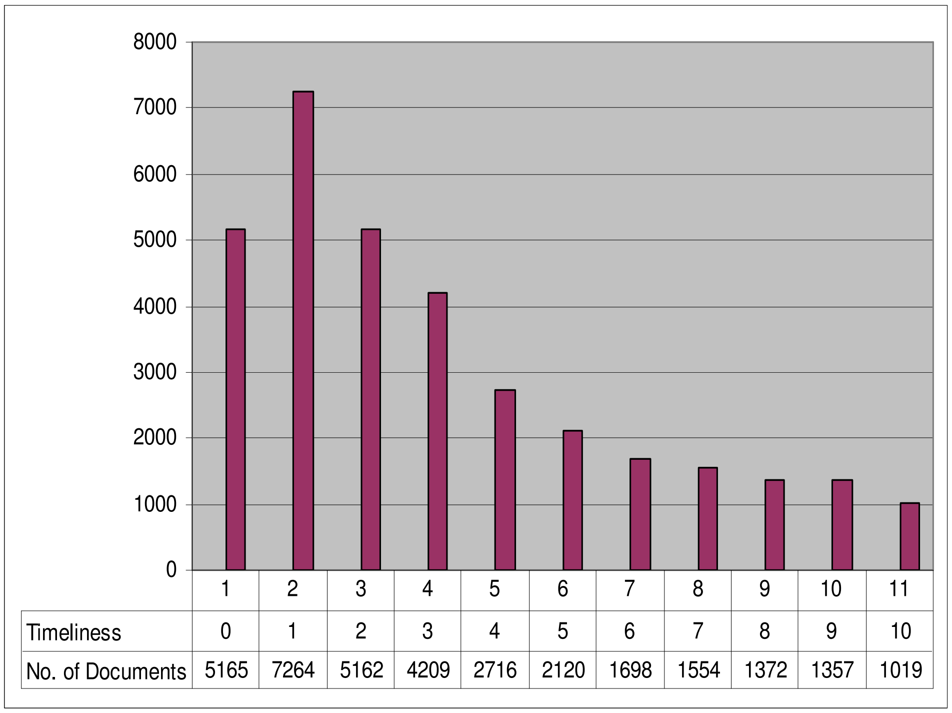
Three main dimensions of quality can be associated with metadata specifications and metadata formats: accuracy, completeness, and conformance.

8.3.1 Accuracy

Accuracy is defined in terms of properties of a metadata specification of a digital object. Accuracy of a triple $\langle r, p, v\rangle$ (i.e., $\langle resource, property^{12}, value\rangle$) refers to the nearness of the value v to some value v' in the attribute domain which is considered as the correct one for the pair: resource r and property p [166]. A metadata specification for a digital object is completely accurate with respect to a digital object if all the triples are accurate, assuming some appropriate accuracy threshold. The degree of accuracy of attribute att_{xy} can be defined as an indicator function or with specific rules for a particular schema/catalog. It is dependent on several factors, including the attribute’s domain, intended use, etc. Examples are given below. Thus, the degree of accuracy $acc(ms_x)$ of a metadata specification ms_x can be defined as

$$acc(ms_x) = \frac{\sum_{\forall \text{attribute } att_{xy} \text{ of } ms_x} \text{degree of accuracy of } att_{xy}}{\text{total number of attributes of } ms_x} \quad (8.9)$$

[12] In this work we will use the terms ‘metadata property’, ‘metadata attribute’, and ‘metadata field’ interchangeably.



	1	2	3	4	5	6	7	8	9	10	11
Timeliness	0	1	2	3	4	5	6	7	8	9	10
No. of Documents	5165	7264	5162	4209	2716	2120	1698	1554	1372	1357	1019

Figure 8.4: Timeliness in the ACM Digital Library

Example of Use. OCLC's WebCat

To illustrate the application of such a indicator we used OCLC's WebCat Union. We chose Webcat because of its numerous problems regarding metadata accuracy, observed while creating a collection for filtering experiments [234]. For example, author information is very commonly found in the title field (e.g., "The concept of the church in the thought of Rudolf Bultmann – by Paul E. Stambach.") and sometimes the abstract contains all kinds of information (see below) but not the thesis/dissertation's summary. We defined the following rules for the dc.author [13], dc.title, and dc.abstract fields:

- Degree of accuracy of dc.title for OCLC = 1, if it does not contain author information; 0.5 otherwise.
- Degree of accuracy of dc.abstract = (number of abstract fields in which the field corresponds to the thesis or dissertation's summary)/ (number of dc.abstract fields). The decision of whether a dc.abstract field corresponds to a summary or not was based on the size of the text and a number of heuristics, for example: 1) if dc.abstract is equal to "Thesis" or "Dissertation", it is not a summary; 2) if dc.abstract contains phrases like "Title from *" (e.g., "Title from first page of PDF file"); "Document formatted into pages", "Includes bibliographical references", "Mode of access", among others, it is not a summary.

Using these two rules the average OCLC accuracy for all its metadata records (approx. 14,000 records, in September 2003 [14]) was calculated as around 0.79, assuming a maximum of 1.

[13]The author field in the Dublin Core standard

[14]Approximately 129K in Sept. 2004

8.3.2 Completeness

Completeness is a pervasive quality dimension which is associated with many of the DL concepts (as can be seen in Table 8.1). The general notion of completeness can be defined as: number of units of a particular concept / ideal number of units of that concept. This notion can be adapted or instantiated to specific DL concepts.

Completeness of metadata specifications refers to the degree to which values are present in the description, according to a metadata standard. As far as an individual property is concerned, only two situations are possible: either a value is assigned to the property in question, or not. The degree of completeness of a metadata specification ms_x can be defined as[15]

$$Completeness(ms_x) = 1 - \frac{\text{no. of missing attributes in } ms_x}{\text{total no. of attributes of the schema to which } ms_x \text{ conforms}} \tag{8.10}$$

Example of application. OCLC NDLTD Union Catalog

Figure 8.5 shows the average of completeness of all metadata specifications (records) of the NDLTD Union Archive administered by OCLC as of February 23, 2004, regarding to the Dublin Core metadata standard (15 attributes). Table 8.9 shows the corresponding institution and number of metadata records per institution.

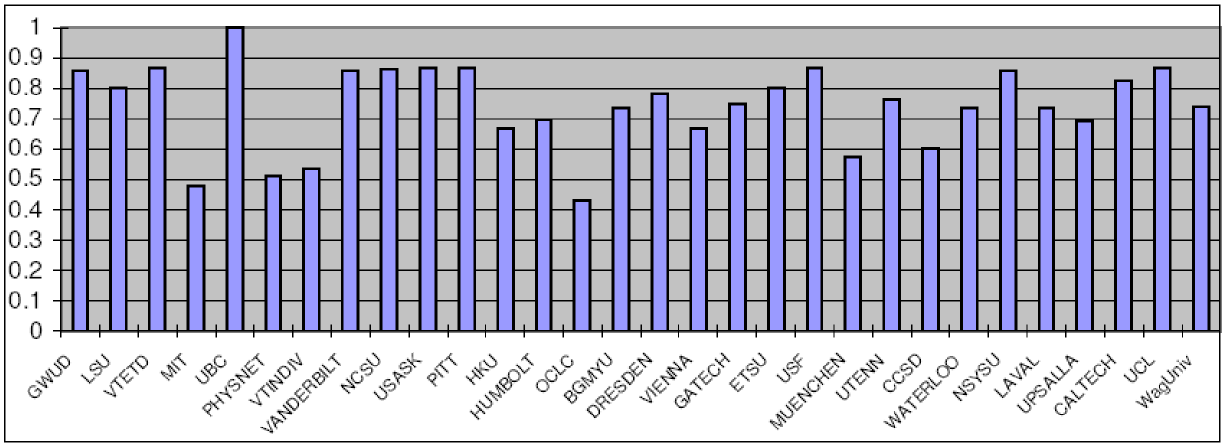


Figure 8.5: Average completeness of catalogs in NDLTD (as of February 2004)

8.3.3 Conformance

The conformance of a metadata specification to a metadata standard/format/schema has been formally defined in Chapter 2, Definition 14. In that definition a value of an attribute is conformant to its schema if it belongs to the defined domain of the attribute (e.g., string, date, number, etc.). That definition can be extended to include cardinality (i.e., considering mandatory/optional fields) and multiplicity (i.e., considering repeatable fields) issues.

A metadata specification ms_x is *cardinally conformant* to a metadata format if:

1. it conforms with its schema in terms of the domain of its attributes according to Definition 14 in Chapter 2;

[15]According to definition of conformance in Chapter 2

Institution	# of records	Institution	# of records	Institution	# of records
GWUD	8	PITT	200	MUENCHEN	181
LSU	646	HKU	9243	UTENN	3751
VTETD	4711	HUMBOLT	346	CCSD	59
MIT	8686	OCLC	14160	WATERLOO	154
UBC	3	BGMYU	76	NSYSU	502
PHYSNET	224	DRESDEN	4	LAVAL	2
VTINDIV	23	VIENNA	261	UPSALLA	1218
VANDERBILT	43	GATECH	482	CALTECH	802
NCSU	1473	ETSU	367	UCL	30
USASK	64	USF	212	WagUniv	2837
				TOTAL	**50768**

Table 8.9: Institutions in the OCLC NDLTD Union Catalog with corresponding number of records (as of February 2004)

2. each attribute att_{xy} of ms_x appears at least once if att_{xy} is marked as mandatory in the schema; and
3. att_{xy} does not appear more than once if it is not marked as repeatable in the schema.

From now on, we will use conformance to refer to the stronger definition of *cardinally conformant*. Differently from completeness, an attribute may be missing in a record, but still can be considered conformant, if it is not marked as mandatory. The degree of conformance of a metadata specification ms_x can be defined as

$$\text{Conformance}(ms_x) = \frac{\sum_{\forall \text{ attribute } att_{xy} \text{ of } ms_x} \text{degree of conformance of } att_{xy}}{\text{total number of attributes}} \tag{8.11}$$

The degree of conformance of att_{xy} is an indicator function defined as 1 if att_{xy} obeys all conditions specified in the above definition; 0 otherwise.

Example of use. The ETD Union Archive

Figure 8.6 shows the average conformance of the metadata records in the NDLTD Union Archive, related to the ETD-MS metadata standard for electronic theses and dissertations. ETD-MS, differently from Dublin Core in which all fields are optional, defines six mandatory fields: dc.title, dc.creator, dc.subject, dc.date, dc.type, dc.identifier. Also the domain for the dc.type is defined as the set {‘Collection’, ‘Dataset’, ‘Event’, ‘Image’, ‘InteractiveResource’, ‘Software’, ‘Sound’, ‘Text’, ‘PhysicalObject’, ‘StillImage’, ‘MovingImage’, ‘Electronic Thesis or Dissertation’}. If any value other than these words is used for the attribute, it is defined as non-conformant.

8.4 Collection, Metadata Catalog, and Repository

8.4.1 Collection Completeness

A complete DL collection is one which contains all the existing digital objects that it should hold. Measuring completeness of a collection can be extremely hard or virtually impossible in many cases when there is no way to determine the ideal real-world collection such as in the Web or in hidden databases. Advanced judicious sampling or probing of alternative repositories whose completeness has been established manually can give crude estimates [142, 106]. An example could be to approximate a measure of the completeness of a

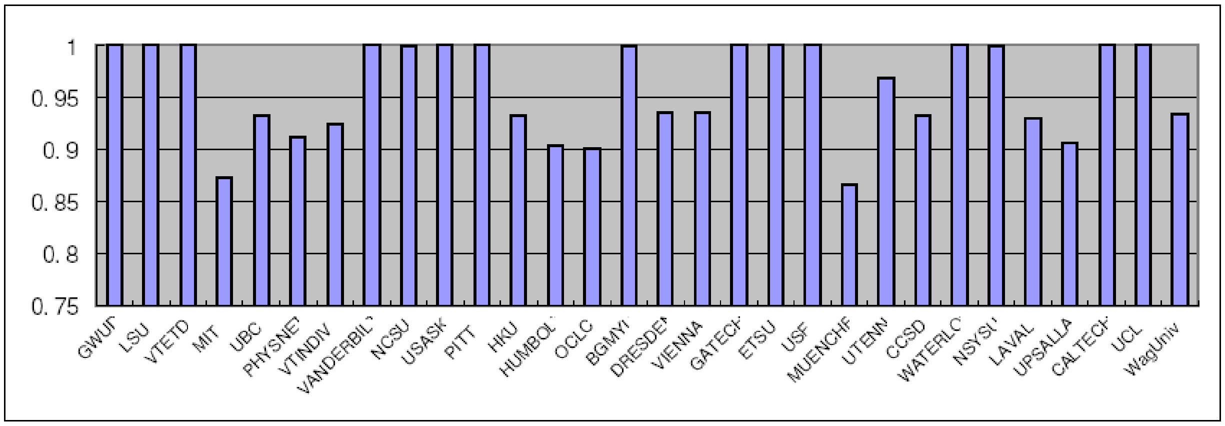


Figure 8.6: Average conformance of catalogs in NDLTD

computer science collection of papers on a specific topic by sampling the ACM or IEEE-CS digital libraries, DBLP, and some other commercial publishers' on-line databases. In other cases such as for harvested or mirrored collections those estimates are easier to establish. More formally, Completeness(C_x) of a collection C_x, can be defined as the ratio between the size of C_x and the ideal real-world collection, i.e.,

$$Completeness(C_x) = \frac{|C_x|}{|\text{ideal collection}|} \tag{8.12}$$

Example of use. Computing collections

The ACM Guide is a collection of bibliographic references and abstracts of works published by ACM and other publishers. Statistics on the number of entries and types of works are shown in Tables 8.10 and 8.11 (as of March 28, 2004). The Guide can be considered a good approximation of an ideal computing collection for a number of reasons including the fact that it contains most of the different types of computing-related literature and for each type it can be considered fairly complete. For example, the set of theses in the Guide comes from Proquest-UMI, which receives copies of almost all dissertations defended in USA or Canada; the number of tech reports is close to that of NCSTRL (http://www.ncstrl.org), the largest repository of CS technical reports, and it contains large numbers of records from many of the most important publishers in computer science (see Table 8.11). Table 8.12 shows the degree of completeness of several CS-related collections [16] when compared with the Guide.

8.4.2 Catalog Completeness and Consistency

Catalog completeness and consistency have been formally defined in Chapter 3 (in the context of the *describes* relationship). The degree of completeness of a catalog DM_C can be defined accordingly as

$$Completeness(DM_C) = 1 - \frac{\text{no. of do's without a metadata specification}}{\text{size of the described collection}} \tag{8.13}$$

Consistency, on the other hand, is an indicator function defined as:

- 0, if there is at least one set of metadata specifications assigned to more than one digital object;
- 1, otherwise.

[16] All subsets of the Guide.

ACM Guide: Types of Works	**No. of entries**
Journal (articles)	256527
Proceeding (papers)	299850
Book(chapters)	107870
Theses/Dissertations	46098
Tech. Reports	25081
Bibliography	2
Play	1
Total	735429

Table 8.10: ACM Guide partitioned by genre.

IEEE	J(29,018)	**Springer**	J(3,524)	**ACM**	J(4,017)	**Elsevier**	J(7,549)	**Kluwer**	J(17,253)
123,844	P(90,760)	108,193	P(96,203)	107,887	P(96,203)	87,777	P(76,335)	27,320	P(3,756)
	B(2,071)		B(7,886)		B(7,886)		B(1,527)		B(3,532)
SIAM	J(12,321)	**Wiley**	J(5,569)	**Pergamon**	J(9,170)	**MIT**	J(3,252)		
15,032	P(1,627)	14,486	P(367)	11,482	P(1,419)	9,593	P(1,770)		
	B(7)		B(7,271)		B(70)		B(4,179)		

Table 8.11: ACM Guide sizes of subsets partitioned by genre (key: J= journal; P = proceedings, B = book) and publisher.

Collection	**Degree of Completeness**
ACM Guide	1
DBLP	0.652
CITIDEL(DBLP(partial) + ACM(partial) + NCSTRL + NDLTD-CS)	0.467
IEEE-DL	0.168
ACM-DL	0.146

Table 8.12: Completeness of several collections

Example of Use. In April 2004, the NDLTD Union catalog administered by OCLC tried to harvest data from the Brazilian Digital Library of Electronic Theses and Dissertations (BDTD). Because of problems in BDTD's implementation of the OAI protocol and problems with the Latin character set handling by OCLC, only 103 records were harvested from the repository. The BDTD collection contained 4446 records. Therefore the completeness of the harvested catalog for BDTD in OCLC would be completeness(BDTD in OCLC Union Catalog) = 1 - (4446 - 103)/4446 = 0.023

8.4.3 Repository Completeness and Consistency

A repository is complete if it contains all collections it should have. The degree of completeness of a repository Rep is defined as

$$Completeness(\mathcal{R}) = \frac{\text{number of collections in the repository}}{\text{ideal number of collections}} \tag{8.14}$$

If the repository *stores* collections with their respective metadata catalogs, its consistency can be defined in terms of these two components. Therefore, repository consistency is an indicator function defined as:

- 1, if the consistency of all catalogs with respect to their described collections is 1;
- 0, otherwise.

Example of use. CITIDEL

We will use the ACM Guide as the ideal collection. Not considering the Bibliography and Play sub-collections of the Guide and considering each publisher as a different sub-collection, the completeness of CITIDEL can be calculated as 4 (ACM + IEEE + NCTRL + NDLTD-CS) / 11 (total number of collections) or 0.36.

8.4.4 Collection Impact Factor

Building upon an analogy between DL collections and journals, the *impact factor* CIF of a DL collection C can be measured as the number of citations or links to digital objects in C [17]. The absolute external collection impact factor eCIF is a variant which does not count self-citations, i.e., citations/links from digital objects in C to other objects in C are not counted.

Example of use. ACM and DBLP

As an example, we applied this metric to the ACM digital library and the DBLP collection, both of which are partially covered by CITIDEL. Table 8.13 shows the CIF for ACM and DBLP (which covers ACM) and eCIF for DBLP alone (removing ACM). Considering that the total number of citations in ACM used in this computation was 1,094,108, it is worth noticing that the CIF for ACM implies that approximately 20.4% of all citations in ACM point to ACM itself. That fact along with the eCIF of DBLP (Approximately 13.48%), which includes many other CS collections like IEEE and the LNCS series, reinforce the high importance of the ACM collection in the computer science domain.

8.5 DL Services

Dimensions of quality for DL services can be classified as external or internal [219]. The external view is related to information satisfaction services and is concerned with the use and perceived value of these

[17] If known, this value can be normalized by the total number of citations in the DL.

	ACM	DBLP
CIF	223,198	369,557
eCIF	none	146,359

Table 8.13: Impact Factor for the ACM DL and DBLP

services from the point of view of societies of end users. The internal view addresses the construction and operation necessary to attain the required functionality given a set of requirements that reflect the external view. Issues in system construction, operation, design, and implementation should be considered here.

8.5.1 Effectiveness and Efficiency

The two most obvious external quality indicators of DL services, as perceived by end users, are efficiency and effectiveness. Efficiency is most commonly measured in terms of speed, i.e., the difference between request and response time. Infrastructure services also may be subject to specific efficiency metrics, for example, speed and space saving for indexing. More formally, let $t(e)$ be the time of an event e, and let e_{ix} and e_{fx} be the initial and the final events of service se_x. The efficiency of service se_x is defined as [18]:

$$\text{Efficiency}(se_x) = t(e_{fx}) - t(e_{ix}) \tag{8.15}$$

Effectiveness is normally related to information satisfaction services and can be measured by batch experiments with test collections or through experiments with real users. Different types of information services can use different metrics, the most common ones being precision and recall [13], extensively used to assess quality of searching or filtering services. Let $|C|$ be the size of a collection and q a query expressing a patron's information need. A searching service processes q and returns a document set A. Let $|A|$ be the size of this set. Moreover, let R_q be the set of all documents relevant to q in C (as judged by external-judges), or pertinent to the information need represented by the query (as judged by a user), and $Ra = A \cap R_q$. Precision P is the fraction of the relevant/pertinent documents retrieved by q, i.e., $P = |Ra|/|A|$. It is useful as an indication of how accurate the service is when retrieving the answers to the user's query. Recall R is the percentage of all the relevant/pertinent items that were retrieved, i.e., $R = |Ra|/|Rq|$. It indicates if the system is able to retrieve all of the relevant/pertinent items. High recall is especially useful when the user needs to be certain that all relevant/pertinent information will be found. These two measures can be combined into a single value providing a simple way of evaluating the service's overall effectiveness, for example the F1 measure combines precision and recall with equal weights and is defined as $F1 = 2PR/(P + R)$. More recently, other quality criteria indicators such as separability and connectivity have been proposed for information satisfaction services such as image retrieval and recommendation [51, 137].

Example of use. Effectiveness: CITIDEL and Structured Queries

Table 8.14 shows the effectiveness of a pure vector space search engine, ESSEX, which implements searching services for CITIDEL and PlanetMath, and a modified, experimental version of the system. In its basic form, ESSEX takes queries as a simple bag of words. The system is also able to handle structured or fielded queries (e.g., author:fox abstract:digital, libraries). The experimental version is able to automatically find a suitable set of structured queries from the original 'bag-of-words' version, ranked by the probability they better represent the user's information need [87]. These following results are based on an experiment conducted using the CITIDEL collection (then with approx. 440,000 records), with 20 subjects [19], each

[18]Here, the smaller the value (i.e., the faster), the more efficient a service is.

[19]Since subjects evaluated items returned by queries representing their own information need, it is reasonable to say that a pertinence view was used here.

issuing 5 bag-of-words queries, with half of them also having to issue manually structured versions of the queries. More details on this experiment and results can be found in [87]. The third row in Table 8.14 reports the performance of the best of the top 5 automatically structured queries. To present the results, besides F1, we also consider two forms of precision: 10-precision and R-precision. The 10-precision measure indicates the precision for the first 10 items retrieved by the system. This measure is important in practice since it is known that users tend to only look at the top results in a ranked answer set. The R-precision measure indicates the precision when all relevant/pertinent documents were retrieved (recall=1). It is a measure of how many spurious results the user has to look at before all relevant/pertinent results are seen. Both measures are useful in determining not only if the system is able to show relevant/pertinent results at the top of the list of retrieved items, but also if it can discover all relevant/pertinent information while still keeping the noise level to a minimum.

Average	**F1**	**10-precision**	**R-precision**
Bag of words	28.9	31.1	29.4
Manually structured query	55.6	72.5	70.2
Best of the top 5 automatically structured queries	59.8	83.4	84.7

Table 8.14: Effectiveness of structured queries

Example of Use. Efficiency: The WISE search engine

Table 8.15 shows efficiency figures for indexing and searching services provided by an experimental search engine – WISE (Web Intelligent Search Engine) [233], built to provide a plataform for experiments with adaptive ranking functions.

	WISE
Indexing	0.40 Gigabytes/hour
Searching	1.2 seconds/query

Table 8.15: Efficiency of indexing and searching services of a search engine

8.5.2 Extensibility and Reusability

Regarding design and implementation of DL services, there are two main classes of quality properties: 1) those regarding composability of services; and 2) those regarding qualitative aspects of the models and implementations. The latter include issues such as completeness, consistency, correctness, and soundness. In this work we will concentrate on composability aspects but we acknowledge the importance and complexity of the latter issues.

Composability can be defined in terms of reusability and extensibility, both properties being formally defined in terms of DL relationships in Chapter 3. In short, a service Y reuses a service X if the behavior of Y incorporates the behavior of X. A service Y extends a service X if it subsumes the behavior of X and potentially includes conditional subflows of events. A composed service either extends or reuses another service. A composable service has to satisfy a number of requirements including exporting clear interfaces, providing mechanisms/protocols for connections and passing of parameters, offering gateway or mediator services to convert between disparate document formats and protocols [61], and satisfying circumstantial conditions such as satisfaction of any pre-condition based on the service's current state and input values to any called service. All of these make it very hard to quantify the composability of a service. However, even if an indicator of composability can be determined, a service is still only potentially reusable and extensible. One more pragmatic indicator of the actual composability is to check from a set of services and service managers that run or implement those services, which managers are actually inherited from or included by others. Therefore given a set of services $Serv$ and a set of service managers SM that run those services, two quality indicators of extensibility and reusability can be defined.

- Macro-Extensibility(Serv) = $\frac{\sum_{se_i \in Serv} extended(se_i)}{|Serv|}$, where $Serv$ is the set of services of the DL and $extended(se_i)$ is an indicator function defined as :
  - 1, if $\exists se_j \in Serv$: se_j extends s_i;
  - 0, otherwise.
- Micro-Extensibility(Serv) = $\frac{\sum_{sm_x \in SM, se_i \in Serv} LOC(sm_x) * extended(se_i)}{\sum_{sm \in SM} LOC(sm)}$, where LOC corresponds to the number of lines of code of all operations of a service manager and sm_x *runs* se_i.
- Since reuse/inclusion has a different semantics of extension, reusability can accordingly be defined as Macro-Reusability(Serv) = $\frac{\sum_{se_i \in Serv} reused(se_i)}{|Serv|}$, where $reused(se_i)$ is an indicator function defined as :
  - 1, if $\exists se_j \in Serv$: se_j reuses s_i;
  - 0, otherwise.
- Micro-Reusability(Serv) = $\frac{\sum_{sm_x \in SM, se_i \in Serv} LOC(sm_x) * reused(se_i)}{\sum_{sm \in SM} LOC(sm)}$, where LOC corresponds to the number of lines of code of all operations of a service manager and sm_x *runs* se_i.

Example of use. ETANA-DL services

Table 8.16 shows the lines of code (LOC) needed to implement service managers which run several services in the ETANA archaeological digital library. Let's assume a 1:1 cardinality between the set of services and set of service managers. Reused services (and included service managers) are implemented as ODL components [199]. These services are searching, annotating, recommending, and (union) cataloging.

The wrapping services, the ones that really reuse and provide the services offered by the DL components, are necessary in order to deal with issues such as invoking operations, parsing results, and interfacing with other components (like the user interface). However, the additional code for those wrappers is only a very small percentage of the total lines of code required for implementing the components. In the current ETANA-DL prototype, only a few important services are componentized and therefore reused (Macro-Reusability(ETANA DL Services) = 4/16 = 0.25. However, Micro-Reusability = 3630/11910 = 0.304 makes it clear that we can re-use a very significant percentage of DL code by implementing common DL services as components. Moreover, as more service managers get componentized, more code and managers are potentially inherited from/included by more DLs.

8.5.3 Reliability

Regarding operation, the most important quality criterion is reliability. Service reliability can be defined as the probability that the service will not fail during a given period of time [100]. We define the reliability of a service se_x as:

$$Reliability(se_x) = 1 - \frac{\text{no. of failures}}{\text{no. of accesses}} \tag{8.16}$$

Service	**Component Based**	**LOC for implementing service**	**LOC reused from component**	**Total LOC**
Searching . Back-end	Yes	-	1650	1650
Search Wrapping	No	100	-	100
Recommending	Yes	-	700	700
Recommend Wrapping	No	200	-	200
Annotating . Back-end	Yes	50	600	600
Annotate Wrapping	No	50	-	50
Union Catalog	Yes	-	680	680
User Interface Service	No	1800	-	1600
Browsing	No	1390	-	1390
Comparing (objects)	No	650	-	650
Marking Items	No	550	-	550
Items of Interest	No	480	-	480
Recent Searches/Discussions	No	230	-	230
Collections Description	No	250	-	250
User Management	No	600	-	600
Framework Code	No	2000	-	2000
	Total	8280	3630	11910

Table 8.16: Analysis of prototype using the metric of Lines of Code

A failure is characterized as an event that:

1. was supposed to happen in a scenario but did not; or
2. did happen but did not execute some of its operations; or
3. did happen, where the operations were executed, but the results were not the expected ones.

Example of use. CITIDEL services

Table 8.17 shows reliability figures for the most popular services of CITIDEL, according to a log analysis done on April 1, 2004. The low reliability for the *structured searching* service can be explained by the fact that it was an experimental one, which ran only for a short period of time. However, entry points and links to this service were not removed after the experiments, and users kept using it without getting answers. This also shows how flaws in design can be found with such quality-oriented analysis.

CITIDEL service	**No. of failures/No. of accesses**	**Reliability**
searching	73/14370	0.994
browsing	4130/153369	0.973
requesting (getobject)	1569/318036	0.995
structured searching	214/752	0.66
contributing	0/980	1

Table 8.17: Reliability of CITIDEL services

8.6 Quality and the Information Life Cycle

Given the fact that information in digital libraries is carried by digital objects and their respective metadata specifications, the proposed dimensions of quality for these two concepts can be connected to the life cycle of information in digital libraries [28]. Such connections can be used to determine when and where quality issues can be measured, assessed, and improved – as well as how possible quality problems can be prevented, detected, and eliminated.

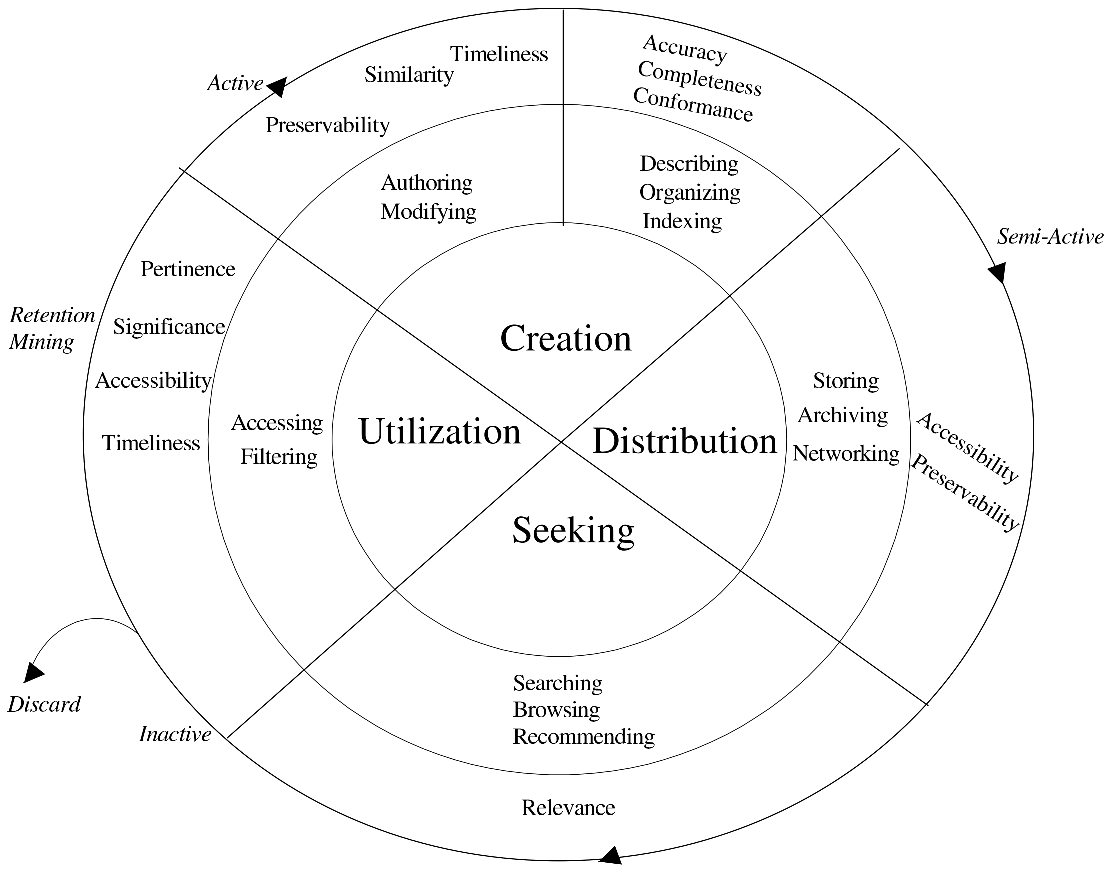


Figure 8.7: Information Life Cycle (adapted from [28]) with respective dimensions of quality added for each major phase and related activities.

The connections are shown in Figure 8.7. The cycle (see inner portion) has four major phases: information creation, distribution, discovery, and utilization. The outer arrows show in which stage information is active, semi-active, or inactive with regard to the phases. Each major phase (see next inner portion) is connected to a number of activities or services. Finally (see outer ring), each dimension of quality is associated with the corresponding set of activities/services.

Similarity to other digital objects or versions can be assessed at time of creation and modification. Preservability and timeliness (in relation to modifications) also are related to this phase. The next subphase in the cycle deals with metadata creation and information organization and description; therefore all quality dimensions associated with metadata specifications are located here. Special metadata quality processes such as enforcing filling out of mandatory fields and use of specific formats (e.g., for dates) as well schema validations, should be applied to related activities to guarantee quality (e.g., accuracy, completeness, conformance).

In the next phase of archiving and distribution, the aspects of accessibility and preservability (e.g., means of storage, format, position in an organizational scheme, etc.) should be taken into consideration. In the discovery (e.g., searching) phase, relevance of information as returned by the several information satisfaction services can be measured. Finally most of the dimensions associated with the perceived value of the information (pertinence, significance) can be assessed during the utilization phase.

8.7 Evaluation: Focus Group

In order to assess the potential practical utility of the proposed quality model in the library and information science (LIS) world, we conducted a focus group with three librarians with experience in practical library work and digital libraries.

This focus group meeting included a presentation of approximately 60 minutes duration about 5S and the proposed quality indicators (with examples) by the researchers /moderators followed by a 30-minute discussion. Questions, comments, and discussions were allowed during the presentation. In particular, the discussions were focused around these questions:

- Are you able to understand the 5S model?
- How does it relate to (your) library world?
- How do the proposed indicators relate to your practices in the library?
- Would you be willing to apply these measures to your (digital) libraries?

8.7.1 Presentation – Discussions

Discussions during the presentation centered around 5S itself and the utility of the quality model. The discussion started with the raising of the basic question of "What's a DL?" by one of the participants. The moderator emphasized that to precisely define DLs was one of the main goals of the work.

When first confronted with 5S, some of the participants felt that the framework made sense but the concepts of scenarios and spaces were the least intuitive. Regarding streams, it was felt that intuitively they seem active and dynamic but the ones used in this work were more static, which was a bit counter-intuitive. One of the participants also said that she had a hard time seeing how the 5S terminology maps to the concepts normally used in the library world. A mapping of conventional LIS terms into the 5S framework was suggested. For example, another participant raised the question of "what a database means in this framework?". After learning that for that participant a database means a storage of documents, we concluded that it corresponded to their notions of collections and repositories.

One of the most discussed aspects was the "minimalist" approach we took in this work. It was almost consensus that this was the right way to go. One of the participants suggested to add "reference service" – essential in the library world – to the minimal DL. This raised concerns that we were dealing with DLs as something outside of the library: "shouldn't the DL be part of the library?" asked one of the participants. It was explained why in a field with so many constructs we could be more elegant and precise to clarify terms with our approach. In the end it was the consensus that the framework was OK, if the focus is on DLs, not (conventional) libraries. It was agreed that it would be challenging to do a metamodel for (conventional) libraries. Another participant also raised concerns about the higher level constructs being understandable by the general public and librarians. Finally some of the relationships among the services (e.g., between indexing and searching) and additional (traditional) DL services were discussed. The moderator did not present the taxonomy of DLs services to the participants in the slides. When it was shown that portions of the dissertation dealt with such issues, they felt happy that this was being covered.

In the second phase of the presentation, discussion concentrated around the quality model. One of the first questions raised was the difference between pertinence and relevance. The explanation was accepted by the participants. Another question which drove part of the discussion was “what do you mean by ‘good’?” and “by good you mean only from the user perspective’?’ This shifted the presentation to the connections between the proposed quality indicators and the several phases of the information cycle.

The discussion now shifted to each proposed quality indicator. Participants seemed to agree with most of the dimensions and definitions. First concerns were raised regarding the “timeliness” indicator for digital objects. One participant argued that in some instances it is impossible to find the age of every object. It was proposed to have a category for “ageless” objects – those never reviewed, reviewed in 5 years, etc. The issue of the age of digital objects which are surrogates of real world artifacts also was discussed. In the context of collection-related indicators, a participant suggested that some of these quality measures could be used for collection management mainly as selection criteria to “get rid of things” in her own words. It was consensus that practical aspects of dealing with storage space and limited resources are rarely discussed in the field.

The strongest reactions were generated by the issues of catalog and collection completeness. It was thought that in some cases, like for example, catalogs based on Dublin Core (15 attributes) this indicator made sense, but in the case of MARC (hundreds of attributes) it would not. It was felt that the context in which those measures would be more applicable should be more elaborated and that these indicators in some cases had more theoretical value than practical application. In the particular case study presented (i.e., OCLC NDLTD Union Catalog), it was conjectured if the low values for completeness of one catalog – MIT – were due to the fact that objects in the respective collection were scanned, not born digital. Finally since we had many indicators for each concept, one argued if we really did need several indicators to get something meaningful.

8.7.2 Post-Presentation Discussion

The post-presentation discussion started with one of the moderators encouraging the participants to think out aloud about how they felt about the whole thing. One of the participants started by expressing his view that the two pieces of the presentation were a bit disconnected, i.e., he felt that the quality model was not very much associated with 5S. He suggested that if the discussion was viewed like a novel, he would think that the “5S” character had disappeared when quality was considered. As a direct consequence of this criticism, we added Table 8.2. The same participant expressed his view that we were proposing two kinds of indicators: 1) some measures against a perfect thing that can not always be known or defined; and others which are highly specific, arguably useful. In the real world, something in between would be the ideal. But the criticism was not restricted to our study, but also applies to most similar studies and other types of quality measures he knows and applies in the library in his work. It was suggested that expressing the contexts where these indicators would be more or less useful would help in their adoption.

Another important question was raised by another participant. In her own words: “What do I do with these measures?”. It was agreed that the goal is to promote improvements and make things better. It also was suggested that these could be used somehow in training of digital librarians and DL administrators.

One of the moderators shifted the discussion to the broader questions of “Which ideas apply to LIS? Which to DL? Which to both?” The first reaction was that DL is only a part of the library; some things can not be digitized. Another participant said that it helped to hear about our minimalist focus – it would help to be more acceptable to LIS. The same participant expressed the personal opinion that 5S would not be much use to him but it might be very helpful to IT people involved with (digital) library issues; impact on LIS was uncertain. Another suggested a study to correlate user satisfaction with the quality indicators. For the explanation of the indicators themselves it was suggested that the application of all of them in a running example would be really helpful. Regarding the proposed quality toolkit, one participant felt more work on services, including infra-structure services and preservation, was necessary [20]. We needed to go beyond the minimal DL for this work to be practical and useful.

In the end, everybody felt the work was very interesting and useful, with potential to help the field of **digital** libraries. In the words of one of the participants who was a little bit familiar with the 5S framework: “you have come a long way with 5S and that is extremely impressive, but more needs to be done before this thing gets widely accepted and practically used”.

8.8 Related Work

The Informetrics and Bibliometrics subfields of Library Science utilize quantitative analysis and statistics to describe patterns of publication within a given field or body of literature. More specifically *evaluative link/citation analysis* is a subfield of Bibliometrics which uses link/citation accounts as indicators or measurements of the level of quality, importance, influence, or performance of individual documents, people, journals, groups, domains, or nations [30]. In computer science, much of the related work has considered the issue of data quality within the database community (e.g., [166, 219, 142, 159]). A comprehensive survey of data quality research [221] has determined that most of the work in data quality has been done in: 1) analysis and design of the data quality aspects of data products; 2) design of data manufacturing systems that incorporate data quality aspects; and 3) definition of data quality dimensions and the measurement of their values. The work proposed here touches several of the aforementioned aspects but is for the digital libraries field, for which this a type of study, mainly one that is based on a formal theoretical foundation, is necessary but has been missing. Indeed, the data/information quality field has mostly defined quality measures/indicators such as completeness in general, nonquantifed terms, such as “the extent to which data is not missing and is of sufficient breadth and depth for the task at hand” [159]. Dhyani et al. [56] present an extensive survey of Web metrics. While many of the suggested Web metrics can be used or adapted to the DL context (and have been), we have seen that DLs differ from the Web in many ways and therefore a specific study of DL quality dimensions and metrics is necessary.

DL quality and evaluation is a very underrepresented research area in the digital library literature. Saracevic [183] was one of the first to consider the problem. He argues that any evaluation has to consider a number of issues such as the context of evaluation, the criteria, the measures/indicators, and the methodology. However, in his analysis, it is concluded that there are no clear agreements regarding the elements of criteria, measures/indicators, and methodologies for DL evaluation. In an attempt to fill some gaps in this area, Fuhr et al. [75] propose a description scheme for DLs based on four dimensions: data/collection, system/technology, users, and usage. We see the work proposed in this dissertation regarding DL quality issues and evaluation as a next, complementary step in this area, one that is based on a sound, formal theory for DLs.

Finally, works in a recent workshop on “Evaluation of Digital Libraries” have touched some of the issues discussed here. Quoting one of the papers in the proceedings of that workshop [5]: “thus it could be worth discussing whether the 5S is an appropriate model for facing this kind of (evaluation) issues and whether it could further a better understanding in this research field”. That is exactly what we have tried to do in this chapter.

8.9 Discussion

It is almost impossible to argue for the completeness of the set of quality indicators here presented, defined, and formalized. While the same argument of being literature-motivated helps, it does not close the matter.

[20] Again, the taxonomy of DL services was not presented to this group.

For example, it is very easy to see that in this work we took a very **system-oriented** view of the quality problem, and, as a self-criticism, we partially neglected its **usage** dimension (as represented by the ‘uses’, ‘participates_in’, and ‘recipient’ relationships of the ontology) where the log analysis could be further benef-ical. Measures/indicators such as popularity of scenarios and digital objects, correctness of scenario models, usability of services, educational potential of resources [203], etc. could be defined under this dimension. However, as is exemplified by the complexities of Tables 3.1 thorugh 3.4, there are many ways that users can interact with services and with the static components of the DL through those services. We leave this for future work.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

Motivated by the almost forty-year-old unanswered challenge from Licklider to construct a unified Computer Science (CS)/ Library and Information Science (LIS) theory, we have, in this dissertation, developed and presented the first comprehensive formal framework for digital libraries – the 5S framework of Streams, Structures, Spaces, Scenarios, and Societies. We show that formal definitions allow the 5S framework to be precisely described and make it possible to clearly and formally define a minimal digital library. Ontological relationships complement the initial conceptualization to compose our theory/framework for DLs. Using that framework we demonstrate its utility: to discuss the terminology found in the digital library literature, to describe a representative digital library and the Open Archives Initiative, to formally define a set of DL constructs and settings in the context of the NDLTD Union Archive, to formally characterize the most typical DL services, and, from that characterization, arrange a taxonomy of services which helps to reason about issues of composability in DLs.

Moreover, we demonstrated how to move from theory to practice by applying the framework to the problems of modeling, generating, and evaluating (by logging and assessing the quality of) digital libraries. Each of these applications is materialized in a number of forms, including: declarative specification languages, visualization and generation tools, standardized XML-based log formats, and formal DL quality models. Throughout this dissertation, we have explained these applications and evaluated them (e.g., through usability studies, focus groups, prototyping, etc).

In summary, this dissertation has made theoretical and practical contributions to the digital library field, which in our opinion are timely and have the potential to lead to significantly positive impact on the future of the field. Yet, much still needs to, and can be done. This is the subject of the next section.

9.2 Future Work

Each of the facets of research in this dissertation could be further studied and developed through future work. We will cover some possibilities in the next sections, following the organization of the dissertation itself.

9.2.1 Theory

As correctly argued in [97], an external review of the 5S TOIS paper, the most effective test for 5S is to examine its ability to describe other DL systems. Accordingly, we intend to use 5S to formally specify and compare some of the most used (and heterogeneous) existing DL systems. Initially, we plan to perform such a task for the three following systems: Fedora [198], SODA [146], and Greenstone, for the exact reasons just mentioned. The reduction of the different characteristics of these systems to our formal framework will allow formal comparisons, emphasize strengths and weaknesses of the systems, and ultimately help to point out where more work is necessary in the DL field as a whole.

Another obvious extension regarding theory is to complete a formal definition for all services in Tables 3.1–3.4 in a way similar to that done for searching, browsing, and indexing in Chapter 2. The idea is to further specify additional events and properties for those services. Such definitions will help us to continue exploring issues of reusability/extensibility of services at a finer level of granularity.

One other possible extension is to load a knowledge base system with the axioms of the ontology. This can serve to:

- find any inconsistencies that may exist in the ontology level;
- latter find any inconsistencies within specific DL models (by translating 5SL models to rules in the knowledge base system)

9.2.2 Application/Tools

Language

5SL has already been shown to be useful to describe and generate prototype systems (e.g., for CITIDEL, NDLTD, BDBComp [55]). However the versions of the language currently used in 5SGraph and 5SGen have some incompatibilities. These versions need to be made uniform. Other extensions also can make the language more useful and powerful. For example, METS (Metadata Encoding and Transmission Standard (METS)) [129], a new standard proposed by the Library of Congress to encode descriptive, administrative, and structural metadata regarding objects within a digital library, expressed using the XML Schema language, is a perfect choice to be used in the structural model. Also the scenario and societal models could be made less complex by increasing the level of abstraction and removing unnecessary details. Finally, we want to make use of some features of the spatial model in the generation process, mainly regarding aspects of the retrieval model and the user interface.

Regarding meta-models, a line of investigation that is already being pursued is the development of new meta-models for "application-oriented" digital libraries. The one developed in this dissertation was based on a minimal set of DL concepts. Currently other meta-models for archaelogical digital libraries [63] are being developed and others for "educational" digital libraries are being considered [66].

Visualization

For the current version of 5SGraph, there are a number of possible extensions. First, we need to integrate 5SGraph with other modeling tools. Many conceptual components in the 5S framework and language are based on existing models, which have their own modeling and editing tools. One obvious example is for the modeling of scenarios as sequence diagrams. 5SGraph should be able to associate conceptual components of its meta-model with other existing tools. The result of modeling with those tools should be integrated into 5SGraph and included in the final 5SL file.

Second, in this same direction, 5SGraph needs to be better integrated with 5SGen. As discussed above, the two current implementations of the tools use different, almost incompatible versions of 5SL. One idea would to go one step further and have a better integrated "environment" based on a "Wizard-of-Oz" philosophy. This environment, working as a "meta-tool", would control the whole process of modeling and generating DLs, by invoking the appropriate tools at the appropriate phases in the DL development cycle, thus guaranteeing integration and correctness of the modeling/generation process. A new improved 5SL version would help the communication among the tools in the environment.

A third category of extensions concerns the visualization functionalities. 5SGraph currently displays the model tree in a truncated way, which simplifies the layout problem and helps users focus on the present context. However, a complete view of the entire tree also may be helpful and useful because sometimes the user would like to see an overview of the entire tree. In addition, a print capability could be added to provide documentation of a completed model. Further, 5SGraph presently supports two cardinality indicators. More indicators can be added to enrich the syntax/semantics.

Fourth, we have seen how 5SGraph has helped users to better understand 5S and consequently better appreciate digital libraries. A natural future application could be to employ the tool as an educational resource for teaching about digital libraries and to further evaluate learnability.

Fifth, we need more tests, mainly including more librarians as our participants. We had only one librarian in our study of 5SGraph, who had the slowest speed in completing the experiment, but showed the best improvement in the understanding of the 5S theory. Better evaluation tests also should be added. These should only give participants some general descriptions of the requirements. Participants then could be asked to create a model from scratch, without step-by-step instructions. 5SGraph also could be tested *in situ* so as to allow digital library designers to benefit from its capabilities.

Sixth, once new meta-models are developed for new areas of application, we should be able to load them in the tool and test them with target audiences.

Generation

5SGen is based on two principles: integration of multiple (related) scenarios for (semi-) automatic generation of service implementations, and reuse of software components, more specifically ODL components. Several extensions of the tool are possible, including:

- Use of Web Services

  The ODL components use XOAI, a locally developed extension of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), for component inter-communication. While OAI-PMH is standard, XOAI is a non-official extension, which complicates interoperability with other systems. One idea is to extend the tool to support Web services, an emergent trend to help promote systematic and extensible service-to-service interactions using Web standards such as SOAP, UDDI, and WSDL as basis for interoperability. Besides enforcing standardization and promoting interoperability, such extensions offer the possibility of using Web Services developed by others to construct our new prototypes.

- Incorporation of Native XML repositories as the basis for storage

  The ODL components used relational databases (e.g., MySQL) as a basis for the storage of data and indexes for services such as searching and browsing. One hypothesis that could be investigated is whether XML native repositories offer a more scalable and efficient alternative for building DL components, since XML is the 'de facto' language of representation in digital libraries. More specifically, we intend to investigate the use of FEDORA, system that already offers support for Web Services at the repository level, and XIR-QL [73] which offers a mixed IR/DB functionalities for XML documents

- Improvement of the Scenario Integration Algorithms

  The current 5SGen's algorithms, despite being innovative, produce a unique object responsible for managing all the integrated and generated services, which makes the tool suffer from scalability problems. We intend to investigate a more scalable architecture with multiple (personal) interaction managers [157]. Such an architecture also will serve as a basis for future investigation of personalization issues in DLs.

Logging

Future work will proceed on several fronts regarding our logging efforts. We will be using our log format to allow evaluations of several of our projects, collections, and systems, including those in the context of the Networked Digital Library of Theses and Dissertations (NDLTD, www.ndltd.org) and the Computing and Information Technology Interactive Digital Educational Library (CITIDEL, www.citidel.org). Since CITIDEL is a part of the National Science (technology, engineering, and mathematics) education Digital Library (NSDL, www.nsdl.org), we will advocate use of the log format and tools throughout NSDL. We will test the log tool with other DL systems. A major concern of any comprehensive log format such as ours should be user privacy. We should allow users to choose the level of detail they want the system to log about their activities. Ideally, user information should be logged and maintained at the client side [36] so that users can use that information as they desire, for example, to provide portions of the data to personalization tools in order to get personalization services.

The current XML format can be very verbose. We intend to investigate efficient compression techniques to allow scalable analysis of our DL logs. Also, we will consider the application and possible extensions of our XML format and tools to support new actions. These extensions should follow the guidance of Tables 3.1–3.4. Finally, our log proposal needs to be discussed and related to standards and framework activities like OAIS [175].

9.2.3 Quality

Several extensions are possible in the quality portion of this dissertation. First and more obvious is the definition and formalization of new quality measures. The measures that were proposed here are, in a sense, too system-oriented. We could envision the definition of measures more focused on the actual *usage* of the DL. Here the role of the XML Log format would be essential. Second, we could materialize the proposed quality model in a new Quality tool – 5SQual.

Besides implementing all the proposed measures 5SQual would have few interesting features such as:

- extensive use of the XML Log standard

  The tool would take log entries in our XML log format, as well as other kinds of information such as XML schemas, consistency rules, etc., and produce statistics on the quality of the several DL parts. A specific sub-module would provide visual aids for DL managers to help mapping between their internal log format and our proposed standard.

- Visualization

  Visualization facilities to produce graphs and charts as the ones shown in Chapter 8, would also be incorporated into the tool

- Decomposition according to the life cycle

  As was discussed in Chapter 8, quality indicators should be measured/assessed at different phases of the information life cycle. Accordingly 5SQual should be decomposed into different modules to be used in each of these different phases.

9.2.4 Others

Personalization

We have already experimented using a combination of 5SL specifications and PIPE, a personalization engine, to customize some aspects of DLs. One possible interesting application is to build a theory of personalizable DLs using 5S. For example, DLs could be personalized over all 5 S dimensions. More concrete examples include:

- a DL should present/transmit only streams supported by the user's enviroment in terms of network speed, hardware, etc.;
- the DL content could be re-organized in terms of personal organizational structures representing users' interests;
- retrieval spaces could be specialized to support personal filters or geographical preferences (e.g., only retrieve information within my home town) or user interfaces could be customized for specific preferences;
- non-useful scenarios could be removed to reduce the complexity of interaction or could be added to allow more personal interactions;
- particular communities of the DL society should be only be able to use services and view content to which they have the appropriate rights.

We think that formalizing DL personalization according to 5S and developing these notions into tools, languages, methods, etc. would be a very interesting future work.

Bibliography

[1] S. Abiteboul, B. Amann, S. Cluet, A. Eyal, L. Mignet, and T. Milo. Active views for electronic commerce. In *Proc. of the 21th Int. Conf. on Very Large Databases, Edinburgh, Scotland, 7–10 September, 1999*, pages 138–149, 1999.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web - From Relations to Semi-structured Data and XML*. Morgan Kaufmann Publishers, San Francisco, 1999.

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel Query Language for Semistructured Data. *Int. Journal on Digital Libraries*, 1(1):5–19, April 1997.

[4] ACM-DL. The ACM Digital Library. http://www.acm.org/dl, 2004.

[5] M. Agosti, G. M. D. Nunzio, and N. Ferro. Evaluation of a digital library system. In *Proc. of the DELOS WP7 Workshop on the Evaluation of Digital Libraries*, pages 73–76, Padova, Italy, October 4-5, 2004.

[6] C. R. Anderson, A. Y. Levy, and D. S. Weld. Declarative web site management with Tiramisu. In *WebDB (Informal Proceedings)*, pages 19–24, 1999.

[7] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, N. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web service description for the Semantic Web. *Lecture Notes in Computer Science*, 2342:348–363, 2002.

[8] W. Arms. *Digital Libraries*. MIT Press, Cambridge, Massachussetts, 2000.

[9] A. Atkins, E. Fox, R. France, and H. Suleman. Interoperability metadata standard for electronic theses and dissertations. http://www.ndltd.org/standards/metadata/current.html, 2001.

[10] D. L. Atkins, T. Ball, G. Bruns, and K. Cox. Mawl: A domain-specific language for form-based services. *IEEE Transactions on Software Engineering*, 25(3):334–346, May/June 1999.

[11] E. Babbie. *The Practice of Social Research*. Wadsworth Publishing Company, Belmont, California, 6th edition, 1990.

[12] R. Baeza-Yates and G. Navarro. XQL and Proximal Nodes. In *Proc. of the ACM SIGIR 2000 – Workshop on XML and Information Retrieval*, Athens, Greece, 2000.

[13] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, May 1999.

[14] D. Bainbridge, J. Thompson, and I. H. Witten. Assembling and enriching digital library collections. In *JCDL'03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 323–334, Houston, Texas, 2003.

[15] D. Bainbrigde, K. J. Don, G. R. Buchanan, I. h. Witten, S. Jones, and M. J. nad Malcolm I. barr. Dynamic digital library construction and configuration. In *Proc. 8th European Conf. Research and Advanced Technology for Digital Libraries, ECDL*, number 3232 in LNCS, pages 1–13, Bath, UK, Sept. 2004. Springer-Verlag.

[16] J. Barclay. Assessing the benefits of learning logs. *Education and Training*, 38(2):30–38, 1996.

[17] L. Baresi, F. Garzotto, and P. Paolini. Extending UML for modeling web applications. In R. H. Sprague, Jr., editor, *Proc. 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*. IEEE Computer Society, 2001.

[18] D. Batory, C. Johnson, B. MacDonald, and H. Heeder. Achieving Extensibility through Product-Lines and Domain-Specific Languages: A Case Study. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(2):191–214, 2002.

[19] K. D. Bayley. *Typologies and Taxonomies – An Introduction to Classification Techniques*. SAGE Publications, Thousand Oaks, California, 1994.

[20] M. Bayraktar, C. Zhang, B. Vadapalli, N. A. Kipp, and E. A. Fox. A web art gallery. In *DL'98: Proc. of the 3rd ACM Int. Conf. on Digital Libraries*, pages 277–278, Pittsburgh, PA, 1998.

[21] C. Beeri. A formal approach to object-oriented databases. *IEEE DKE*, 5:353–382, December 1990.

[22] N. J. Belkin, R. N. Oddy, and H. M. Brooks. ASK for information retrieval. *Journal of Documentation*, 33(2):61–71, June 1982.

[23] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5), May 2001.

[24] D. Bolchini and P. Paolini. Goal-oriented requirements specification for digital libraries. *Lecture Notes in Computer Science*, 2458:107–117, 2002.

[25] G. Booch. UML in action. *Communications of the ACM*, 42(10):26–28, 1999.

[26] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, Massachusetts, USA, 1st edition, 1999.

[27] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web Services Architecture. http://www.w3.org/TR/ws-arch/, October, 2004.

[28] C. L. Borgman. Social aspects of digital libraries. In *DL'96: Proceedings of the 1st ACM International Conference on Digital Libraries*, D-Lib Working Session 2A, pages 170–171, 1996.

[29] C. L. Borgman. What are digital libraries? competing visions. *Information Processing and Management*, 35(3):227–243, January 1999.

[30] C. L. Borgman and J. Furner. Scholarly communication and bibliometrics. In *Annual Review of Information Science and Technology – Volume 36, 2002*, pages 3–72. ASIST, Medford, NJ, USA, 2002.

[31] C. L. Borgman, S. G. Hirsh, and J. Hiller. Rethinking online monitoring methods for information retrieval systems: From search product to search process. *Journal of the American Society of Information Science*, 47(7):568–583, 1996.

[32] P. Borlund and P. Ingwersen. The development of a method for the evaluation of interactive information retrieval systems. *Journal of Documentation*, 53(3):225–250, June 1997.

[33] M. D. Boualem Benatallah, Quan Z. Sheng. The self-serv environment for web services composition. *IEEE Internet Computing*, 7(1):40–48, 2003.

[34] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, 1996.

[35] J. M. Carrol. *Scenario-Based Design:Envisioning work and technology in system design*. John Wiley, New York, 1995.

[36] L. N. Cassel and U. Wolz. Client Side Personalization. In *Proc. of the Joint DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, pages 8–12, Dublin, Ireland, June 18-20, 2001.

[37] D. Castelli, C. Meghini, and P. Pagano. Foundations of a multidimensional query language for digital libraries. *Lecture Notes in Computer Science*, 2458:251–265, 2002.

[38] D. Castelli and P. Pagano. A system for building expandable digital libraries. In *JCDL'03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 335–345, Houston, Texas, 2003.

[39] R. G. G. Cattell, T. Atwood, J. Dubl, G. Ferran, M. Loomis, and D. Wade. *The Object Database Standard: ODMG*. Morgan Kaufmann Publishers, Los Altos, CA, USA, 1994.

[40] S. Ceri, P. Fraternali, and S. Paraboschi. Data-driven, one-to-one web site generation for data-intensive applications. In *Proc. of the 25th Int. Conf. on Very Large Data Bases (VLDB '99)*, pages 615–626, San Francisco, Sept. 1999.

[41] CITIDEL. Computing and Information Technology Interactive Digital Educational Library. http://www.citidel.org, 2004.

[42] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath, 1999.

[43] C. L. Clarke, G. V. Cormack, and F. J. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38:43–56, 1995.

[44] E. F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[45] J. H. Coombs, A. H. Renear, and S. J. DeRose. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947, 1988.

[46] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, Cambridge, Massachusetts, 1990.

[47] A. Crabtree, M. B. Twidale, J. O'Brien, and D. M. Nichols. Talking in the library: implications for the design of digital libraries. In *Proc. of the 2nd ACM Int. Conf. on Digital Libraries*, pages 221–229, New York, July 1997.

[48] A. Crespo and H. Garcia-Molina. Archival storage for digital libraries. In *DL'98: Proceedings of the 3rd ACM International Conference on Digital Libraries*, pages 69–78, 1998.

[49] F. Crestani, M. Lalmas, C. J. van Rijsbergen, and I. Campbell. “Is this document relevant? ... probably”: A survey of probabilistic models in information retrieval. *ACM Computing Surveys*, 30(4):528–552, Dec. 1998.

[50] F. Curbera and et al. Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI. *IEEE Distributed Systems Online*, 3(4), 2002.

[51] R. da S. Torres, A. X. Falcão, and L. da F. Costa. A Graph-based Approach for Multiscale Shape Analysis. *Pattern Recognition*, 37(6):1163–1174, June 2004.

[52] L. Davis and M. Dawe. Collaborative design with use case scenarios. In *JCDL’01: Proc. of the 1st Joint Conf. on Digital Libraries*, pages 146–147, Roanoke, Virginia, 2001.

[53] M. D. Davis, R. Sigal, and E. J. Weyuker. *Computation, Complexity, and Languages (second edition)*. Academic Press, 1994.

[54] M. C. F. de Oliveira, T. Turine, and P. C. Masiero. A statechart-based model for hypermedia applications. *ACM Transactions on Information Systems*, 19(1):28–52, 2001.

[55] D. P. V. del Pozo, L. V. e Silva, A. H. F. Laender, and M. A. Gonçalves. Modelagem de bibliotecas digitais usando a abordagem 5s: Um estudo de caso. In *Anais do XIX Simpsio Brasileiro de Bancos de Dados*, page (to be published, Brasilia, DF, Brazil, 2004. SBC.

[56] Dhyani, Ng, and Bhowmick. A survey of web metrics. *ACM Computing Surveys*, 34(4):469–503, 2002.

[57] A. Doan, J. Madhavan, R. Dhamankar, et al. Learning to match ontologies on the Semantic Web. *VLDB Journal: Very Large Data Bases*, 12(4):303–319, Nov. 2003.

[58] A. Dong and A. M. Agogino. Design principles for the information architecture of a SMET education digital library. In *Proc. of the 1st Joint Conf. on Digital Libraries*, pages 314–321, Roanoke, Virginia, June 24-28, 2001.

[59] D. Ellis. The physical and cognitive paradigms in information retrieval research. *Journal of Documentation*, 48:45–64, 1992.

[60] W. Fan, M. D. Gordon, and P. Pathak. Personalization of search engine services for effective retrieval and knowledge management. In *The Proceedings of the International Conference on Information Systems 2000*, pages 20–34, 2000.

[61] M.-C. Fauvet, M. Dumas, F. Rabhi, and B. Benatallah. Patterns for e-service composition. In J. Noble, editor, *Pattern Languages of Programs 2002. Revised papers from the Third Asia-Pacific Conference on Pattern Languages of Programs, (KoalaPLoP 2002)*, volume 13 of *Conferences in Research and Practice in Information Technology*, page 37, Melbourne, Australia, 2003. ACS.

[62] M. Fernández, D. Florescu, A. Levy, and D. Suciu. Declarative specification of Web sites with Strudel. *VLDB Journal: Very Large Data Bases*, 9(1):38–55, 2000.

[63] J. W. Flanagan, E. A. Fox, and W. Fan. Managing complex information applications: An archaeology digital library. http://feathers.dlib.vt.edu/, 2003.

[64] D. J. Foskett. A note on the concept of relevance. *Information Storage and Retrieval*, 8(2):77–78, 1972.

[65] D. J. Foskett. Thesaurus. In *Encyclopedia of Library and Information Science - Volume 30*, pages 416–462. Marcel Dekker, New York, 1980.

[66] E. Fox, J. Carroll, P. fan, L. Cassel, M. Zubair, K. Maly, G. McMillan, N. Ramakrishnan, and H. Halbert. Science of digital libraries(sciDL). Technical report, Virginia Tech, Feb. 01 2003.

[67] E. A. Fox, R. M. Akscyn, R. K. Furuta, and J. J. Leggett. Digital libraries. *Communications of the ACM*, 38(4):22–28, 1995.

[68] E. A. Fox, J. L. Eaton, G. McMillan, N. A. Kipp, P. Mather, T. McGonigle, W. Schweiker, and B. DeVane. Networked digital library of theses and dissertations: An international effort unlocking university resources. *D-Lib Magazine*, 3(9), September 1997.

[69] E. A. Fox, M. A. Gonçalves, G. McMillan, J. Eaton, A. Atkins, and N. Kipp. The Networked Digital Library of Theses and Dissertations: Changes in the University Community. *Journal of Computing in Higher Education*, 13(2):3–24, Spring 2002.

[70] E. A. Fox and G. Marchionini. Toward a worldwide digital library. *Communications of the ACM*, 41(4):22–28, 1998.

[71] P. Fraternali and P. Paolini. Model-driven development of Web applications: the AutoWeb system. *ACM Transactions on Information Systems*, 18(4):323–382, 2000.

[72] J. Frew, M. Freeston, N. Freitas, and L. Hill. The Alexandria digital library architecture. *Lecture Notes in Computer Science*, 1513:61–73, 1998.

[73] N. Fuhr, , and K. Grobjohann. XIRQL - an XML query language based on information retrieval concepts. *ACM Transactions on Information Systems*, 22(4):313 – 356, 2004.

[74] N. Fuhr. XIRQL - An Extension of XQL for Information Retrieval. In *Proc. of the ACM SIGIR 2000 – Workshop on XML and Information Retrieval*, Athens, Greece, 2000.

[75] N. Fuhr, P. Hansen, M. Mabe, A. Micsik, and I. Sølvberg. Digital libraries: A generic classification and evaluation scheme. *Lecture Notes in Computer Science*, 2163:187–199, 2001.

[76] N. Fuhr, C.-P. Klas, S. Schaefer, and P. Mutschke. Daffodil: An integrated desktop for supporting high-level search activities in federated digital libraries. *Lecture Notes in Computer Science*, 2458:597–612, 2002.

[77] R. Furuta. Defining and using structure in digital documents. In *Proc. of the 1st Annual Conf. on the Theory and Practice of Digital Libraries*, College Station, Texas, 1994.

[78] R. Furuta, V. Quint, and J. Andre. Interactively editing structured documents. *Electronic Publishing—Origination, Dissemination, and Design*, 1(1):19–44, Apr. 1989.

[79] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison Wesley Professional Computing Series. Addison Wesley, 1995. http://www.aw.com.

[80] D. A. Garza-Salazar. Phronesis. http://copernico.mty.itesm.mx/ tempo/Proyectos/, 2001.

[81] H. Gladney, E. A. Fox, Z. Ahmed, R. Ashany, N. J. Belkin, and M. Zemankova. Digital library: Gross Structure and Requirements: Report from a March 1994 Workshop. In *Proc. 1st Annual Conf. on the Theory and Practice of Digital Libraries*, pages 101–107, Texas, 1994.

[82] H. M. Gladney and A. Cantu. Authorization management for digital libraries. *Communications of the ACM*, 44(5):63–65, 2001.

[83] R. Godement. *Algebra*. Kershaw Publ. Co. Ltd, London, 1969.

[84] D. Goh and J. Leggett. Patron-augmented digital libraries. In *DL'00: Proc. of the 5th ACM Int. Conf. on Digital Libraries*, pages 153–163, San Antonio, TX, USA, 2000.

[85] C. F. Goldfarb and P. Prescod. *The XML Handbook*. Prentice-Hall PTR, Upper Saddle River, NJ 07458, USA, 1998.

[86] M. A. Gonçalves and E. A. Fox. 5SL – A language for declarative specification and generation of digital libraries. In *Proc. of the 2nd Joint Conf. on Digital Libraries (JCDL'2002)*, pages 263–272, Portland, Oregon, July 14-18, 2002.

[87] M. A. Gonçalves, E. A. Fox, A. Krowne, P. Calado, A. H. F. Laender, A. S. da Silva, and B. Ribeiro-Neto. The Effectiveness of Automatically Structured Queries in Digital Libraries. In *Proc. of the 4th Joint Conf. on Digital Libraries (JCDL'2004)*, pages 98–107, Tucson, Arizona, June 7-11, 2004.

[88] M. A. Gonçalves, E. A. Fox, and L. T. Watson. Towards a digital library theory: A formal digital library ontology. In *Proceedings of the ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, Sheffield, England, 2004.

[89] M. A. Gonçalves, E. A. Fox, L. T. Watson, and N. Kipp. Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries. Technical Report 03-04, Virginia Tech, 2003.

[90] M. A. Gonçalves, E. A. Fox, L. T. Watson, and N. Kipp. Streams, structures, spaces, scenarios, societies (5S): A formal model for digital libraries. *ACM Transactions on Information Systems*, 22(2):270–312, 2004.

[91] M. A. Gonçalves, E. A. Fox, B. Zhang, and L. T. Watson. Towards a quality model for digital libraries. In *Proceedings of the DELOS Workshop on the Evaluation of Digital Libraries*, pages 39–40, University of Padua. Italy, 2004.

[92] M. A. Gonçalves, R. K. France, and E. A. Fox. MARIAN: Flexible Interoperability for Federated Digital Libraries. In *Proc. of the 5th European Conf. on Research and Advanced Technology for Digital Libraries*, pages 173–186, Darmsdadt, Germany, 2001. Springer.

[93] M. A. Gonçalves, M. Luo, R. Shen, M. F. Ali, and E. A. Fox. An XML log standard and tool for digital library logging analysis. *Lecture Notes in Computer Science*, 2458:129–143, 2002.

[94] M. A. Gonçalves, P. Mather, J. Wang, Y. Zhou, M. Luo, R. Richardson, R. Shen, L. Xu, and E. A. Fox. Java MARIAN: From an OPAC to a modern digital library system. In *Proc. of SPIRE'02*, pages 194–209, Lisbon, Portugal, September 11-13 2002.

[95] M. A. Gonçalves, A. A. Zafer, N. Ramakrishnan, and E. A. Fox. Modeling and Building Personalized Digital Libraries with PIPE and 5SL. In *Proc. of the Joint DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, pages 67–72, Dublin, Ireland, June 18-20, 2001.

[96] M. A. Gonçcalves, G. Panchanathan, U. Ravindranathan, A. Krowne, F. Fox, F. Jagodzinski, and L. Cassel. The XML log standard for digital libraries: analysis, evolution, and deployment. In *JCDL'03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 312–314, 2003.

[97] M. M. Gonzalez. Review for Streams, structures, spaces, scenarios, societies (5s):a formal model for digital libraries by Gonalves M., Fox E., Watson L., Kipp N., July, 2004.

[98] H. Greisdorf. Relevance thresholds: a multi-stage predictive model of how users evaluate information. *Information Processing and Management*, 39(3):403–423, 2003.

[99] H. Han, C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. A. Fox. Automatic document metadata extraction using support vector machines. In *JCDL'03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 37–48, Houston, Texas, 2003.

[100] J. V. Hansen. Audit considerations in distributed processing systems. *Communications of the ACM*, 26(8):562–569, 1983.

[101] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[102] L. S. Heath, D. Hix, L. T. Nowell, W. C. Wake, G. A. Averboch, E. Labow, S. A. Guyer, D. J. Brueni, R. K. France, K. Dalal, and E. A. Fox. ENVISION: A user-centered database of computer science literature. *Communications of the ACM*, 38(4):52–53, 1995.

[103] P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, and C. Chen. Formal approach to scenario analysis. *IEEE Software*, 11(2):33–41, Mar. 1994.

[104] J. Hunter and S. Choudhury. A semi-automated digital preservation system based on semantic web services. In *Proceedings of the Fourth ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 269–278, Tucson, Arizona, 2004.

[105] P. Ingwersen, K. van Rijsbergen, and N. Belkin. Context in Information Retrieval. http://ir.dcs.gla.ac.uk/context/IRinContext_WorkshopNotes_SIGIR2004.pdf, 2004.

[106] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden Web: Hierarchical database sampling and selection. In P. A. Bernstein et al., editors, *VLDP 2002: proceedings of the Twenty-Eighth International Conference on Very Large Data Bases, Hong Kong SAR, China, 20–23 August 2002*, pages 394–405, Los Altos, CA 94022, USA, 2002. Morgan Kaufmann Publishers.

[107] S. C. Jane Hunter. Implementing Preservation Strategies for Complex Multimedia Objects. In *Proc. 7th European Conf. Research and Advanced Technology for Digital Libraries, ECDL 2003*, pages 473–486, Trodheim, Norway, August 17-22, 2003.

[108] K. S. Jones and P. Willett, editors. *Readings in Information Retrieval*. Multimedia Information and Systems. Morgan Kaufmann Publishers, 1997.

[109] B. Kahin and H. R. Varian. *Internet Publishing and Beyond: The Economics of Digital Information and Intellectual Property*. MIT Press, Cambridge, Massachusetts, 2000.

[110] L. A. Kalinichenko, D. O. Briukhov, N. A. Skvortsov, and V. N. Zakharov. Infrastructure of the subject mediating environment aiming at semantic interoperability of heterogeneous digital library collections. In *Proc. of the 2nd Russian Scientific Conf. on Digital Libraries: Advanced Methods and Technologies*, 2000.

[111] N. K. Kaske. Research methodologies and transaction log analysis: Issues, questions and a proposed model. *Library Hi Tech*, 42(11):79–86, 1993.

[112] H. Kautz, B. Selman, and M. Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[113] R. Kelapure. Scenario-Based Generation of Digital Library Services. Master's thesis, Virginia Tech – Departement of Computer Science, July 21 2003.

[114] R. Kelapure, M. A. Gonçalves, and E. A. Fox. Scenario-based generation of digital library services. In *Proc. 7th European Conf. Research and Advanced Technology for Digital Libraries, ECDL*, number 2769 in LNCS, Trondheim, Norway, Aug. 2003. Springer.

[115] D. A. Kemp. Relevance, pertinence, and information system development. *Information Storage and Retrieval*, 10(2):37–47, 1974.

[116] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14(1):10–25, Jan. 1963.

[117] T. Kochtanek and K. K. Hein. Delphi study of digital libraries. *Information Processing and Management*, 35(3):245–254, 1999.

[118] M. Kying. Creating contexts for design. In *Scenario-Based Design: Envisioning Work and Technology in System Development*. John Wiley & Sons, New York, NY, USA, 1995.

[119] A. H. F. Laender, B. A. Ribeiro-Neto, and A. S. da Silva. DEByE - data extraction by example. *Data and Knowledge Engineering*, 40(2):121–154, 2002.

[120] C. Lagoze. The Warwick framework: A container architecture for diverse sets of metadata. *D-Lib Magazine*, 2(7), July 15 1996.

[121] C. Lagoze and H. V. de Sompel. The Open Archives Initiative. In *Proc. of the 1st Joint Conf. on Digital Libraries (JCDL'2001)*, pages 54–62, Roanoke, Virginia, June 24-28, 2001.

[122] C. Lagoze, D. Fielding, and S. Payette. Making global digital libraries work: Collection services, connectivity regions, and collection views. In I. Witten, R. Akscyn, and F. M. Shipman, editors, *Proc. of the 3rd ACM Conf. on Digital Libraries (DL-98)*, pages 134–143, New York, June 23–26 , 1998. ACM.

[123] A. V. Lamsweerde and L. Willemet. Inferring declarative requirements specifications from operational scenarios. *IEEE Trans. on Soft. Engineering*, 24(12):1089–1114, December 1998.

[124] R. Larsen. Knowledge Lost in Information – Report of the NSF Workshop on Research Directions for Digital Libraries. http://www.sis.pitt.edu/ dlwkshop/report.pdf, 2004.

[125] S. Lawrence, C. L. Giles, and K. D. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[126] M. Lesk. Expanding digital library research: Media, genre, place and subjects. In *Proc. of the Int. Symposium on Digital Libraries 1999: ISDL'99*, Tsukuba, Ibaraki, Japan, September 28-29, 1999.

[127] D. M. Levy. Heroic measures: reflections on the possibility and purpose of digital preservation. In *DL'98: Proceedings of the 3rd ACM International Conference on Digital Libraries*, pages 152–161, Pittsburgh, PA, 1998.

[128] D. M. Levy and C. C. Marshall. Going digital: a look at assumptions underlying digital libraries. *Communications of the ACM*, 38(8):77–84, 1995.

[129] LibraryOfCongress. METS - Metadata Encoding and Transmission Standard. http://www.loc.gov/standards/mets/, 2003.

[130] J. C. R. Licklider. *Libraries of the Future*. MIT Press, Cambridge, Massachusetts, 1965.

[131] R. A. Lorie. Long Term Preservation of Digital Information. In *Proc. of the 1st Joint Conf. on Digital Libraries (JCDL'2001)*, pages 346–352, Roanoke, Virginia, June 24-28, 2001.

[132] D. Lucarella and A. Zanzi. A visual retrieval environment for hypermedia information systems. *ACM Transactions on Information Systems*, 14(1):3–29, Jan. 1996.

[133] C. Lynch. Personalization and recommender systems in the larger context: New directions and research questions (keynote speech). In *Proc. of the DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, pages 84–88, Dublin, Ireland, June 18-20, 2001.

[134] W. E. Mackay and M. Beaudouin-Lafon. DIVA exploratory data analysis with multimedia streams. In *Proc. of CHI-98*, pages 416–423, Los Angeles, CA, USA, Apr. 18-23, 1998.

[135] G. Marchionini. Advanced Interface Designs for the BLS Website: Final Report to the Bureau of Labor Statistics. http://ils.unc.edu/m̃arch/blsreport98/final_report.html, 1998.

[136] T. Miller. Annotation system for a collection of ETDs. http://www.ndltd.org/ndltd-sc/990416/annsystem.pdf, 1999.

[137] B. J. Mirza, B. J. Keller, and N. Ramakrishnan. Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems*, 20(2):131–160, 2003.

[138] S. Mizzaro. A cognitive analisys of information retrieval,. In *Information Science: Integration in Perspective – Proceedings of CoLIS2*, pages 233–250, Copenhagen, Denmark, 1996.

[139] S. Mizzaro. How many relevances in information retrieval? *Interacting With Computers*, 10(3):305–322, 1998.

[140] B. Mobasher, R. Cooley, and S. Srivastava. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8):142–151, Aug. 2000.

[141] E. L. Morgan. MyLibrary. http://dewey.library.nd.edu/mylibrary/, 2004.

[142] A. Motro and I. Rakov. Estimating the quality of databases. In T. Andreasen, H. Christiansen, and H. L. Larsen, editors, *Proceedings of the 3rd International Conference on Flexible Query Answering Systems (FQAS-98)*, volume 1495 of *LNAI*, pages 298–307, Berlin, May 13–15 1998. Springer.

[143] G. Navarro and R. Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems*, 15(4):400–435, 1997.

[144] NDLTD. Networked Digital Library of Theses and Dissertations. http://www.ndltd.org, 2004.

[145] M. L. Nelson and K. Maly. Buckets: Smart objects for digital libraries. *Communications of the ACM*, 44(5):60–61, 2001.

[146] M. L. Nelson, K. Maly, M. Zubair, and S. N. T. Shen. SODA: Smart objects, dumb archives. In *Proc. 3rd European Conf. Research and Advanced Technology for Digital Libraries, ECDL*, number 1696 in LNCS, Paris, France, Sept. 1999. Springer-Verlag.

[147] S. Nestorov, S. Abiteboul, and R. Motwani. Inferring structure in semistructured data. *SIGMOD Record*, 26(4):39–43, 1997.

[148] F. D. Neves and E. A. Fox. A study of user behavior in an immersive virtual environment for digital libraries. In *Proc. of the 5th ACM Conf. on Digital Libraries: ACM DL'00*, pages 103–112, San Antonio, Texas, 2000.

[149] NSDL. National science digital library. http://www.nsdl.org, 2004.

[150] OAI. Open Archives Initiative protocol for metadata harvesting - v.2.0. http://www.openarchives.org/OAI/openarchivesprotocol.html, 2001.

[151] D. Oard, C. Peters, M. Ruiz, R. Frederking, J. Klavans, and P. Sheridan. Multilingual Information Discovery and AccesS (MIDAS): A joint ACM DL'99 / ACM SIGIR'99 workshop. *D-Lib Magazine*, 5(10), Oct. 15 1999.

[152] A. Oberweis and P. Sander. Information system behavior specification by high-level Petri nets. *ACM Transactions on Information Systems*, 14(4):380–420, Oct. 1996.

[153] R. Ogawa, H. Harada, and A. Kaneko. Scenario-based hypermedia: A model and a system. In *Proc. of the ECHT'90 European Conf. on Hypertext*, pages 38–51, 1990.

[154] OMG. OMG-XML Metadata Interchange (XMI) Specification, v1.2. http://cgi.omg.org/docs/formal/02-01-01.pdf, 2002.

[155] S. Payette and T. Staples. The mellon fedora project. *Lecture Notes in Computer Science*, 2458:406–421, 2002.

[156] S. Perugini, M. A. Gonçalves, and E. . A. Fox. Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems*, 23(2):107–143, 2004.

[157] S. Perugini, K. McDevitt, R. Richardson, M. Perez-Quinones, R. Shen, N. Ramakrishnan, C. Williams, and E. A. Fox. Enhancing Usability in CITIDEL: Multimodal, Multilingual, and Interactive Visualization Interfaces. In *Proc. of the 4th Joint Conf. on Digital Libraries (JCDL'2004)*, pages 315–324, Tucson, Arizona, June 7-11, 2004.

[158] T. A. Peters. The history and development of transaction log analysis. *Library Hi Tech*, 42(11):41–66, 1993.

[159] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, Apr. 2002.

[160] J. Powell and E. A. Fox. Multilingual federated searching across heterogeneous collections. *D-Lib Magazine*, 5(8), 1998.

[161] R. Prince, J. Su, H. Tang, and Y. Zhao. The design of an interactive online help desk in the Alexandria Digital Library. In *Proc. of the Int. Joint Conf. on Work Activities and Collaboration: WACC '99*, pages 217–226, San Francisco, CA, 1999.

[162] N. Ramakrishnan. PIPE: Web Personalization By Partial Evaluation. *IEEE Internet Computing*, 4(6):21–31, 2000.

[163] S. R. Ranganathan. *A Descriptive Account of Colon Classification*. Bangalore: Sarada Ranganathan Endowment for Library Science, 1965.

[164] U. Ravindranathan, R. Shen, M. A. Gonçalves, W. Fan, E. A. Fox, and F. Flanagan. Prototyping digital libraries handling heterogeneous data sources - the ETANA-DL case study. In *Proc. 8th European Conf. Research and Advanced Technology for Digital Libraries, ECDL*, number 3232 in LNCS, pages 186–197, Bath, UK, Sept. 2004. Springer-Verlag.

[165] R. Reddy and I. Wladawsky-Berger. Digital Libraries: Universal Access to Human Knowledge - A Report to the President. President's Information Technology Advisory Committee (PITAC), Panel on Digital Libraries. http://www.itrd.gov/pubs/pitac/pitac-dl-9feb01.pdf, 2001.

[166] T. C. Redman. *Data Quality – Management and Technology*. Bantam Books, New York, 1992.

[167] W. Reisig. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1985.

[168] D. Riecken. Introduction: personalized views of personalization. *Communications of the ACM*, 43(8):26–28, Aug. 2000.

[169] S. E. Robertson. The probability ranking principle in IR. *Documentation J.*, 33:294–304, 1977.

[170] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

[171] M. B. Rosson. Integrating development of task and object models. *Communications of the ACM*, 42(1):49–56, 1999.

[172] M. B. Rosson and J. M. Carroll. Object-oriented design from user scenarios. In *Proc. of ACM CHI 96 Conf. on Human Factors in Computing Systems*, pages 342–343, 1996.

[173] J. Rothenberg. *Using Emulation to Preserve Digital Documents*. Koninklijke Bibliotheek, The Netherlands, Aug. 21 2000.

[174] M. Rowan, P. Gregor, D. Sloan, and P. Booth. Evaluating web resources for disability access. In *Fourth Annual ACM Conference on Assistive Technologies*, pages 80–84, Arlington, Virginia, 2000. ACM.

[175] P. Rdig, U. M. Borghoff, J. Scheffczyk, and L. Schmitz. Preservation of digital publications: an oais extension and implementation. In *Proceedings of the 1st ACM Symposium on Document Engineering*, pages 131–139, 2003.

[176] A. B. S. Ceri, P. Fraternali. Web modeling language (WebML): a modeling language for designing web sites. In *Proc. of the 9th Int. World Wide Web Conference*, Amsterdam, 2000.

[177] G. Salton and M. E. Lesk. The SMART automatic document retrieval system—an illustration. *Communications of the ACM*, 8(6):391–398, 1965.

[178] G. Salton and M. J. McGill. *The SMART and SIRE Experimental Retrieval Systems*. McGraw-Hill, New York, 1983.

[179] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[180] B. Sandore. Applying the results of transaction log analysis. *Library Hi Tech*, 42(11):87–97, 1993.

[181] S. Sanett. The Cost to Preserve Authentic Electronic Records in Perpetuity: Comparing Costs across Cost Models and Cost Frameworks. *RLG DigiNews*, 7(4), August 2003. http://www.rlg.org/preserv/diginews/v7_n4_feature2.html.

[182] T. Saracevic. Relevance: a review and a framework for thinking on the notion in information science. *Journal of the American Society for Information Science*, 26:321–343, 1975.

[183] T. Saracevic. Digital library evaluation: Toward evolution of concepts. *Library Trends*, 49(2):350–369, 2000.

[184] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, USA, 1996.

[185] P. Schauble and A. F. Smeaton. Summary report of the series of joint NSF-EU working groups on future directions for digital library research: An international research agenda for digital libraries. http://www.ercim.org/publication/ws-proceedings, October, 1998.

[186] S. Schönberger, R. K. Keller, and I. Khriss. Algorithmic support for model transformation in object-oriented software development. *Concurrency and Computation: Practice and Experience*, 13(5):351–383, Apr. 2001.

[187] D. Schwabe, G. Rossi, and S. D. J. Barbosa. Systematic hypermedia application design with OOHDM. In *Proc. of the 7th ACM Conf. on Hypertext*, pages 116–128, 1996.

[188] M. F. Schwartz and D. C. M. Wood. Discovering shared interests using graph analysis. *Communications of the ACM*, 36(8):78, 1993.

[189] D. E. Shackelford, J. B. Smith, and F. D. Smith. The architecture and implementation of a distributed hypermedia storage system. In *Proc. of the 5th Conf. on Hypertext*, pages 1–13, Seattle, Washington, Nov. 1993.

[190] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, Oct. 1948.

[191] E. Silva de Moura, G. Navarro, N. Ziviani, and R. Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems*, 18(2):113–139, Apr. 2000.

[192] M. Singhal and N. Shivaratri. *Advanced Concepts in Operating Systems: Distributed, Database, and Multiprocessor Operating Systems*. McGraw-Hill, New York, 1994.

[193] H. G. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, July-Aug. 1973.

[194] A. G. Smith. Web links as analogues of citations. *Information Research*, 9(4), 2004. Available at http://InformationR.net/ir/9-4/paper188.html.

[195] H. V. D. Sompel. Roadblocks. http://www.sis.pitt.edu/ dlwkshop/paper_sompel.html, 2003.

[196] M. Spiliopoulou. Web usage mining for Web site evaluation. *Communications of the ACM*, 43(8):127–134, Aug. 2000.

[197] J. Spivey. *Introducing Z: A Specification Language and its Formal Semantics*. Cambridge University Press, 1988.

[198] T. Staples, R. Wayland, and S. Payette. The fedora project - an open-source digital object repository management system. *D-Lib Magazine*, 9(4), Apr. 2003.

[199] H. Suleman. *Open Digital Libraries*. PhD thesis, Virginia Tech Department of Computer Science, 2003.

[200] H. Suleman, A. Atkins, M. A. Gonçalves, R. K. France, E. A. Fox, V. Chachra, M. Crowder, and J. Young. Networked digital library of theses and dissertations: Bridging the gaps for global access - part 1: Mission and progress. *D-Lib Magazine*, 7(9), 2001. Available at http://www.dlib.org/dlib/september01/suleman/09suleman-pt1.html.

[201] H. Suleman, A. Atkins, M. A. Gonçalves, R. K. France, E. A. Fox, V. Chachra, M. Crowder, and J. Young. Networked digital library of theses and dissertations: Bridging the gaps for global access - part 2: Services and research. *D-Lib Magazine*, 7(9), 2001. Available at http://www.dlib.org/dlib/september01/suleman/09suleman-pt1.html.

[202] T. Sullivan and R. Matson. Barriers to use: Usability and content accessibility on the web's most popular sites. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, Easy Access and The Web, pages 139–144, 2000.

[203] T. Sumner, M. Khoo, M. Recker, and M. Marlino. Understanding educator perceptions of ”quality” in digital libraries. In *JCDL'03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 269–279, Houston, Texas, 2003.

[204] A. Sutcliffe. A technique combination approach to requirements engineering. In *Proc. of the 3rd Int. Symp. on Requirements Engineering*, pages 65–77, Annapolis, 1997. IEEE.

[205] A. G. Sutcliffe, N. A. M. Maiden, S. Minocha, and D. Manuel. Supporting scenario-based requirements engineering. *IEEE Trans. on Soft. Engineering*, 24(12):1072–1088, 1998.

[206] J. Suzuki and Y. Yamamoto. Making UML models interoperable with UXF. *Lecture Notes in Computer Science*, 1618:78–87, 1999.

[207] J. Tague, A. Salminen, and C. McClellan. Complete formal model for information systems. In *Proc. of the 14th annual int. ACM/SIGIR conf. on research and development in information retrieval*, pages 14–20, Chicago, IL, USA, October 13-16, 1991.

[208] J. Tanaka and S. Vasilache. Synthesizing statecharts from multiple interrelated scenarios, Oct. 01 2001. http://citeseer.ist.psu.edu/561606.html.

[209] R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan, and M. Smith. DSpace: An institutional digital repository system. In *Proc. of the 3rd Joint Conference on Digital Libraries*, pages 87–97, Houston, Texas, 2003.

[210] R. S. Taylor. Question-negotiation and information seeking in libraries. *College and Research Libraries*, 29:178–194, 1968.

[211] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Trans. on Inf. Sys.*, 9(3):187, 1991.

[212] J. D. Ullman. *Principles of Database and Knowledge-Base Systems. Volume I: Classical Database Systems*. Computer Science Press, Rockville, Maryland, 1988.

[213] B. C. Vickery. Faceted classification schemes. In *Rutgers Series for the Intellectual Organization of Information – Volume 5*. Rutgers University Press, New Brunswick, NJ, USA, 1965.

[214] E. M. Voorhees. Evaluation by highly relevant documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82, 2001.

[215] VTLS. VTLS. http://www.vtls.com, 2001.

[216] W3C. *Resource Description Framework (RDF) Model and Syntax Specification*, 1998. http://www.w3.org/TR/WD-rdf-syntax/.

[217] W3C. Resource Description Framework (RDF) Schema Specification 1.0. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, 2000.

[218] W3C. XML Schema Part 0: Primer, W3C Working Draft. http://www.w3.org/TR/xmlschema-0, 2000.

[219] Y. Wand and R. Y. Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95, Nov. 1996.

[220] B. Wang. A hybrid system approach for supporting digital libraries. *Int. Journal on Digital Libraries*, 2(2-3):91–110, 1999.

[221] R. Y. Wang, V. C. Storey, and F. Firth. A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):623–640, Aug. 1995.

[222] W. Wang and R. Rada. Structured hypertext with domain semantics. *ACM Transactions on Information Systems*, 16(4):372–412, October 1998.

[223] A. Waugh, R. Wilkinson, B. Hills, and J. Dell'oro. Preserving digital information forever. In *DL'00: Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 175–184, San Antonio, Texas, 2000.

[224] P. Weinstein and G. Alloway. Seed ontologies: Growing digital libraries as distributed, intelligent systems. In *DL'97: Proceedings of the 2nd ACM International Conference on Digital Libraries*, Agents, pages 83–91, 1997.

[225] G. Wiederhold. Digital libraries, value, and productivity. *Communications of the ACM*, 38(4):85–96, 1995.

[226] R. Wilkison and M. Fuller. Integration of information retrieval and hypertext via structure. In *Information Retrieval and Hypertext*, pages 257–271. Kluwer Academic Publishers, 1996.

[227] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction.* Foundations of Computing series. MIT Press, Cambridge, MA, USA, Feb. 1993.

[228] I. Witten and D. Bainbridge. *How to Build a Digital Library*. Elsevier, New York, 2002.

[229] I. H. Witten, D. Bainbridge, and S. Boddie. Greenstone: Open-source DL software. *Communications of the ACM*, 44(5):47, 2001.

[230] I. H. Witten, R. J. McNab, S. J. Boddie, and D. Bainbridge. Greenstone: A comprehensive open-source digital library software system. In *Proc. of the 5th ACM Int. Conf. on Digital Libraries*, pages 113–121, San Antonio, TX, June 2-7, 2000.

[231] T. W. Yan and H. Garcia-Molina. The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4):529–565, 1999.

[232] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Text Categorisation, pages 13–22, 1994.

[233] X. Yu, R. Yang, and M. Luo. Generalized search platform for digital library. class project report.

[234] B. Zhang, M. A. Gonçalves, and E. A. Fox. An OAI-Based Filtering Service for CITIDEL from NDLTD. In *Proc. of the 6th International Conference on Asian Digital Libraries, ICADL 2003*, pages 590–601, Kuala Lumpur, Malaysia, December 8-12, 2003.

[235] Q. Zhu, M. A. Gonçalves, and F. Fox. 5SGraph demo: a graphical modeling tool for digital libraries. In *JCDL'03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, Demonstrations, page 385, 2003.

[236] Q. Zhu, M. A. Gonçalves, R. Shen, L. Cassell, and E. A. Fox. Visual semantic modeling of digital libraries. In *Proc. 7th European Conf. Research and Advanced Technology for Digital Libraries, ECDL*, number 2769 in LNCS, Trondheim, Norway, Aug. 2003. Springer.

[237] N. Ziviani, E. S. de Moura, G. Navarro, and R. Baeza-Yates. Compression: A key for next-generation text retrieval systems. *IEEE Computer*, 33(11):37–44, Nov. 2000.

[238] M. Zubair, K. Maly, I. Ameerally, and M. Nelson. Dynamic construction of federated digital libraries. In *Proc. of the 9th Int. World Wide Web Conf.*, Amsterdam, May 15-19, 2000.

APPENDIX

Mathematical Preliminaries

Here, we briefly review the mathematical foundations necessary for the development of the following discussion. Since the goal is complete precision, all terms used in the other definitions must be carefully and unambiguously defined. Authors' definitions of terms even as basic as "function" often disagree, so (for completeness) we begin at the most fundamental level, with set notations, relations, functions, sequences, tuples, strings, graphs, and grammars [46]. Readers familiar with these concepts can skip this section or simply refer to it as needed when some of the concepts are used in other definitions.

Formally, *set* and $\in$ ("element of") are taken as undefined terms in the axioms of set theory. We remark that a set cannot contain itself and the "set of all sets" does not exist. That x is an element of set S is denoted $x \in S$. There is an "empty" set ($\emptyset$). The notation $S = \{x|P(x)\}$ defines a set S of precisely those objects x for which the logical proposition $P(x)$ is true. Standard operations between sets A and B include union: $A \cup B = \{x|x \in A \text{ or } x \in B\}$; intersection: $A \cap B = \{x|x \in A \text{ and } x \in B\}$; and Cartesian product: $A \times B = \{(a, b)|a \in A \text{ and } b \in B\}$ where (a, b) is called an *ordered pair*. A is called a *subset* of B, denoted by $A \subset B$, if $x \in A$ implies $x \in B$. The set of all subsets of set S (including $\emptyset$) exists, is called the *power set* of S, and is denoted 2^S.

Appendix Definition 1 *A binary* ***relation*** *R on sets A and B is a subset of $A \times B$. We sometimes write $(a, b) \in R$ as aRb. An n-ary relation R on sets $A_1, A_2, ..., A_n$ is a subset of the Cartesian product $A_1 \times A_2 \times ... \times A_n$.*

Appendix Definition 2 *Given two sets A and B, a* ***function*** *f is a binary relation on $A \times B$ such that for each $a \in A$ there exists $b \in B$ such that $(a, b) \in f$, and if $(a, b) \in f$ and $(a, c) \in f$ then $b = c$. The set A is called the domain of f and the set B is called the codomain of f. This is shown as $f : A \rightarrow B$. We write $b = f(a)$ as a common notation for $(a, b) \in f$. The set $\{f(a)|a \in A\}$ is called the range of f.*

Appendix Definition 3 *A* ***sequence*** *is a function f whose domain is the set of natural numbers or some initial subset $\{1, 2, ..., n\}$ of the natural numbers and whose codomain is any set.*

Appendix Definition 4 *A* ***tuple*** *is a finite sequence that is often denoted by listing the range values of the function as $\langle f(1), f(2), ..., f(n)\rangle$.*

Appendix Definition 5 *A* ***string*** *is a finite sequence of characters or symbols drawn from a finite set with at least two elements, called an* ***alphabet****. A string is often denoted by concatenating range values without punctuation. Let Σ be an alphabet. Σ^* denotes the set of all strings from Σ, including the empty string (an empty sequence ϵ). A* ***language*** *is a subset of Σ^*.*

Appendix Definition 6 *A* ***graph*** *G is a pair (V, E), where V is a nonempty set (whose elements are called* ***vertices****) and E is a set of two-item sets of vertices, $\{u, v\}$, $u, v \in V$, called* ***edges****. A* ***directed graph*** *(or* *digraph) G is a pair (V, E), where V is a nonempty set of vertices (or nodes) and E is a set of edges (or arcs) where each edge is an ordered pair of distinct vertices (v_i, v_j), with $v_i, v_j \in V$ and $v_i \neq v_j$. The edge (v_i, v_j) is said to be* ***incident*** *on vertices v_i and v_j, in which case v_i is* ***adjacent to*** *v_j, and v_j is* ***adjacent from*** *v_i.*

Several additional concepts are associated with graphs. A **walk** in graph G is a sequence of not-necessarily distinct vertices such that for every adjacent pair $v_i, v_{i+1}, 1 \leq i < n$, in the sequence, $(v_i, v_{i+1}) \in E$. We call v_1 the origin of the walk and v_n the terminus. The **length** of the walk is the number of edges that it contains. If the edges of the walk are distinct, the walk is a **trail**. If the vertices are distinct, the walk is a **path**. A walk is **closed** if $v_1 = v_n$ and the walk has positive length. A **cycle** is a closed walk where the origin and non-terminal vertices are distinct. A graph is **acyclic** if it has no cycles. A graph is **connected** if there is a path from any vertex to any other vertex in the graph. A **tree** is a connected, acyclic graph. A **directed tree** or (DAG) is a connected, directed graph where one vertex - called the root - is adjacent from no vertices and all other vertices are adjacent from exactly one vertex. A graph $G' = (V', E')$ is a **subgraph** of $G = (V, E)$, if $V' \subseteq V$ and $E' \subseteq E$.

Appendix Definition 7 *A* ***context-free grammar*** *is a quadruple (V, Σ, R, s_0) where V is a finite set of symbols called non-terminals, Σ is an alphabet of terminal symbols, R is a finite set of rules and s_0 is a distinguished element of V called the* ***start*** *symbol.*

A **rule**, also called a production, is an element of the set $V \times (V \cup \Sigma)^*$. Each production is of the form $A \rightarrow \alpha$ where A is a non-terminal and α is a string of symbols (terminals and/or non-terminals).

Appendix Definition 8 *A* ***deterministic finite automaton*** *is a 5-tuple $(Q, q_0, A, \Sigma, \delta)$ where Q is a finite set of symbols called states, $q_0 \in Q$ is the* ***start*** *automaton state, $A \subseteq Q$ is a distinguished set of accepting states, Σ is an alphabet (defining what set of input strings the automaton operates on), and δ is a function from $Q \times \Sigma$ into Q, called the transition function of the automaton.*

The finite automaton begins in state q_0 and reads characters of an input string one at a time. If after reading the string the automaton is in a state $q \in A$ the string is **accepted**.

Appendix Definition 9 *Let X be a set. A σ-**algebra** is a collection $\mathbb{B}$ of subsets of X that satisfies the following conditions:*

1. *every union of a countable collection of sets in $\mathbb{B}$ is again in $\mathbb{B}$, i.e., if $A_i \in \mathbb{B}$ ($i = 1, 2, 3, \dots$), then $\bigcup_{i=1}^{\infty} A_i \in \mathbb{B}$;*
2. *if $A \in \mathbb{B}$, then $\tilde{A} \in \mathbb{B}$, where $\tilde{A}$ is the complement of A with respect to X.*

One consequence of the definition of σ-algebra is that the intersection of a countable collection of sets in $\mathbb{B}$ is again in $\mathbb{B}$.

Appendix Definition 10 *A* ***measurable space*** *is a tuple $(X, \mathbb{B})$ consisting of a set X and a σ-algebra $\mathbb{B}$ of subsets of X.*

A subset A of X is called *measurable* (or *measurable with respect to* $\mathbb{B}$) if $A \in \mathbb{B}$. A *measure* μ on measurable space $(X, \mathbb{B})$ is a nonnegative real-valued function defined for all sets of $\mathbb{B}$ such that the following conditions are satisfied:

1. $\mu(\emptyset) = 0$ where $\emptyset$ is the empty set, and

2. $\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \Sigma_{i=1}^{\infty}\mu(A_i)$ for any sequence A_i of pairwise disjoint measurable sets.

Appendix Definition 11 *A* ***measure space*** $(X, \mathbb{B}, \mu)$ *is a measurable space* $(X, \mathbb{B})$*, with measure* μ *defined on* $\mathbb{B}$*.*

Appendix Definition 12 *A* ***probability space*** *is a measure space* $(X, \mathbb{B}, \mu)$*, such that measure* $\mu(X) = 1$*.*

Appendix Definition 13 *A* ***vector space*** *is a set* V *(whose elements are called* ***vectors****) together with a field of "scalars"* [1] *with an addition operation* $+ : V \times V \to V$ *and a multiplication operation* $* : S \times V \to V$ *such that if* x, y, z *are in* V *and* α *and* β *are in* S *then:*

1. *there is a unique vector* $0 \in V$ *such that* $x + 0 = x$ *for all* $x \in V$ *(additive identity);*
2. *for each vector* $x \in V$ *there exists a vector* $-x \in V$ *such that* $x + (-x) = 0$ *(additive inverse);*
3. $(x + y) + z = x + (y + z)$ *(associativity of* $+$*);*
4. $x + y = y + x$ *(commutativity of* $+$*);*
5. $1 * x = x$ *(identity);*
6. $(\alpha * \beta) * x = \alpha * (\beta * x)$ *(associativity of* $*$*);*
7. $(\alpha + \beta) * x = \alpha * x + \beta * x$ *(distributivity of* $*$ *over* $+$*, right); and*
8. $\alpha * (x + y) = \alpha * x + \alpha * y$ *(distributivity of* $*$ *over* $+$*, left).*

Appendix Definition 14 *A* ***topological space*** *is a pair* $(X, \mathcal{T})$ *consisting of a set* X *and a family* $\mathcal{T} \subset 2^X$ *of subsets of X such that:*

1. $\emptyset$ *(the empty set)* $\in \mathcal{T}$ *and* $X \in \mathcal{T}$*;*
2. *for any collection of sets in* $\mathcal{T}$*,* $\{A_i \in \mathcal{T} | i \in I\}$*,* $\cup_{i \in I} A_i$ *is also in* $\mathcal{T}$*, and if the index set* I *is finite,* $\cap_{i \in I} A_i$ *is in* $\mathcal{T}$*.*

$\mathcal{T}$ is said to be a topology for X, and elements of $\mathcal{T}$ are called **open** sets. The complement of an open set is called a **closed** set.

[1] In this context, the field of real numbers.

Vita

Marcos André Gonçalves concluded his doctoral degree in Computer Science at Virginia Polytechnic Institute and State University in 2004. He earned a Master degree from State University of Campinas (UNICAMP) in 1997 and a Bachelor degree from the Federal University of Ceará (UFC) in 1995, both in Computer Science. He has published 5 book chapters, 10 journal/magazine papers, and more than 30 conference/workshop papers. In these papers, he has collaborated with more than 70 different researchers. He received 5 awards including the Lewis Trustee Award from Laspau for promoting collaborative research between the U.S. and Latin America (Brazil) and the ACM/IEEE 2004 Joint Conference on Digital Library's Best Student Paper Award. His research interests include Digital Libraries and Information Retrieval. His next activities will include collaborating with his advisor on a book (tentative title: "Foundations for Information Systems: Digital Libraries and the 5S Framework") based on some of the ideas explored in his dissertation.