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

Marcos André Gonçalves, Edward A. Fox
Layne T. Watson, and Neill A. Kipp
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA
{*mgoncalv,fox,ltw,kipp*}*@cs.vt.edu*

Abstract

Digital libraries (DLs) are complex information systems and therefore demand formal foundations lest development efforts diverge and interoperability suffers. In this paper, 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 on those sets 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 between and 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". The applicability, versatility and unifying power of the 5S model are demonstrated through its use in three distinct applications: building and interpretation of a DL taxonomy, informal and formal analysis of case studies of digital libraries (NDLTD and OAI), and utilization as a formal basis for a DL description language.

Keywords: digital libraries, theory, foundations, taxonomy, definitions, applications.

1 Motivation

Digital libraries are 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 [73, 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 [72]. Lesk analyzed the relative weights of the words *digital* and *library* in recent efforts in the field, and concluded that those efforts are dissociated from an understanding of users' needs and their use of the resources being provided [71]. Borgman explicitly explored the competing visions of the digital library field, both from the research and from the practitioner communities, and showed the difficulty that this conflict imposes on activities like defining terms, characterizing terminologies, and establishing contexts [15]. 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 [64]. 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" [95]. Underlying all of these is the consensus agreement that digital libraries are fundamentally complex.

Such complexity most probably 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 [35]. 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 risking the fluidity of information assets in the future.

The broad and deep requirements of digital libraries demand new models and theories in order to understand better the complex interactions among their several components [44]. As evidence of this claim, the summary report of the Joint NSF-European Union (EU) Working Groups on Future Directions of Digital Libraries Research recommended that "new models and theories be developed in order to understand the complex interactions between the various components in a globally distributed digital library" [104]. 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 model or theory for digital libraries.

Formal models 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 models that have steered them well: databases [21, 119, 11, 19, 2, 56], information retrieval [60, 101, 96, 118, 133, 116, 8], and hypertext and multimedia [75, 28]. A formal model 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 models 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. Their lack leads to diverging efforts and has made interoperability one of the most important problems faced by the field.

In this paper we introduce five formalisms—streams, structures, spaces, scenarios, and societies (5S)— 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 model of digital libraries.

This paper is organized as follows. Section 2 presents an overview of the 5S model, including definitions, examples, and discussions of three applications including: a) construction and interpretation of a DL taxonomy; b) informal analysis of case studies of digital libraries; and 3) utilization of 5S as a basis for a DL description language. Section 2 is purposely informal and introduces most of key concepts in an intuitive manner without complete precision; that is the role of the following section. Accordingly, section 3 proceeds to formally define key information constructs that were introduced in the previous section. Section 4 then builds on this framework to formally describe several DL higher level constructs and settings. Section 5 discusses related work and Section 6 concludes the paper.

2 5S Overview: Informal Definitions, Applications

2.1 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 has a temporal nature and 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 represents 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 [107]. 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 [76]. The synchronization of streams can be specified with Petri Nets [87] 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.

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

2.2 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 [41]. 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 [40, 22, 48]. Relational and object-oriented databases impose strict structures (called *schemas*) on data, typically using tables or graphs as units of structuring [11]. Indexing in information retrieval systems by a manual process serves to categorize and support future requests, generating an organizational structure for the document space.

With the increase of 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 [1]. Query languages and algorithms can extract structure from these data [66, 2, 83, 78, 89, 49, 124]. Although most of those efforts have a "data-centric" view of semi-structured data, works with a more "document-centric view" have emerged [7, 39]. In general, human and natural language processing routines can expend considerable effort to unlock the interwoven structures found in texts at syntactic, semantic, pragmatic, and discourse levels.

2.3 Spaces

A space is a set of objects together with operations on those objects that obey certain constraints. The combination of operations on objects with the set of objects is what distinguishes spaces from streams and structures. Since this 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 [46]. In the context of digital libraries, Licklider discusses spaces for information [73, 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 [101]. “Feature spaces” are sometimes used with image as well as document collections and are suitable for clustering or probabilistic retrieval [97]. 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 [108, 136]). 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 [10, 84]. Also spaces for computer-supported cooperative work provide a context for virtual meetings and collaborations [24, 92].

Again, spaces are distinguished by the operations on their elements. Digital libraries can use many types of spaces for indexing, visualizing, and other services that they perform. The most prominent of these for digital libraries are measurable spaces, measure spaces, probability spaces, vector spaces, and topological spaces. Section 3.2 defines formally these concepts of space.

2.4 Scenarios

One important type of scenario is a story that describes possible ways to use a system to accomplish some function that the 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 [65]; provide guidelines to build a cost-effective prototype [113]; or help to validate, infer and support requirements specifications and provide acceptance criteria for testing [58, 114, 70]. 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 operations 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 [88]. 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 [129]. 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 [72]. Scenarios also help us consider the complexities of current publishing methods, as well as how they may be reshaped in the era of digital libraries, considering publishing paths, associated participants, and publication functions [128].

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 paper; the reader is referred to [130] 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. From this it is easy to see how dataflow and workflow in digital libraries and elsewhere can be modeled using scenarios.

2.5 Societies

A society is a set of entities and the relationships between them. The entities include humans surrogates 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 have created 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 human surrogates.

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 [12, 33, 16], 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 IR 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. [95]. 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 [86].

Economics, a critical societal concern, is also key for digital libraries [61]. 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 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 [74].

2.6 Applications of 5S

In this section, we illustrate the expressiveness and unifying power of 5S as a theory for digital libraries through three different example applications. In the first, we build a taxonomy of DL concepts derived from the literature and characterize the result in the light of the theory. The second application uses 5S as an analytical tool to understand and dissect a DL instance and a DL protocol for interoperability. Third, we present a brief description of a declarative language based on 5S for the specification and automatic generation of DL applications.

2.6.1 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 [9]. 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 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 our 5S theory. This interpretation aims toward a more informal conceptual understanding of the 'Ss' and corresponding DL components to understand the resulting agglomerations of common concepts in the taxonomy.

In the process of building such a taxonomy, we have considered the principles of taxonomies in social sciences, notably cluster analysis, and the faceted classification schemes [120]. The presentation of the taxonomy also was influenced by the work of Saracevic and Kantor [103] in their taxonomy of *value* in libraries and information services. In particular we were guided by the idea that writing about a subject unequivocally reveal the appropriate facets for that subject [34], and that those facets are enough to describe the phenomenon [94]. 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. In particular, we explored relevant contributions from the following literature sources:

- ACM DL conferences (1995-2000),
- ACM Transactions on Information System,
- Communications of the ACM (particularly 4/95, 4/98, 5/2001),
- D-Lib Magazine,
- European Conference on Digital Libraries (1997-2000),
- IEEE Computer DL Issue (4/97),
- IEEE-CS International Conference - Advances in Digital Libraries (1996-2000),
- Independent (Texas) DL Conferences 94, 95,
- International Journal on Digital Libraries (Springer),
- Journal of the American Society for Information Science (and Technology),
- Web in general.

As a starting point, we used an initial set of terms and phrases listed alphabetically in [36]. To this list we added other terms from the 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. The resulting taxonomy is shown in Figure 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, for example. Second, the higher-level groupings are significantly abstract so that they may be applied to many fields, with possible additions or changes probably 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. In particular, we discuss the key parts of Figure 1 informally in terms of the five "S"s and their combinations.

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 whose design 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

[2] Digital communities are formed by actors who interact with a DL possibly through a 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*
  - *Disseminating*
    - Collaborating
    - Filtering
    - Providing access
    - Publicizing
      - Marketing/Advertising
    - Translating
    - Using Email/Listservs
  - *Evaluating/Assessing*
    - Analyzing logs
    - Certifying
    - Doing action research
    - Doing ethnographic study
    - Doing laboratory study
    - Doing sociological study
    - Leading focus groups
    - Measuring
    - Surveying
    - Reviewing (peer)
  - *Modeling*
    - Extracting
    - Linking
  - *Organizing*
    - Analyzing
    - Annotating
      - Narrating
      - Rating
    - Classifying
    - Clustering
    - Indexing
  - *Personalizing*
    - Customizing
    - Recommending
  - *Preserving/Archiving*
    - Conserving
    - Converting
    - Copying/Replicating
    - Emulating
    - Renewing
    - Translating
  - *Requesting*
  - *Selecting*
    - Browsing
    - 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
    - Business
      - Copyright law
      - Economics
      - Management
      - Marketing
      - Publishing
    - Computer science
      - Databases
      - HCI
      - Hypertext
      - Information retrieval
      - Multimedia
      - Software engineering
    - Education
    - Engineering
    - Ethnography
    - Humanities
      - Museum science
    - Information science
    - Library science
    - Philosophy
    - Sociology
  - *Purposes*
    - Commercial service
    - Education support
    - National library
    - Public services
    - Training
  - *Scope*
    - Company
    - Department
    - Discipline
    - Group
    - National
    - Personal
    - State
    - Worldwide

Figure 1: Taxonomy of Digital Libraries Terms

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 [105, 63]).

Components: What constitutes DLs? Digital libraries can contain repositories of knowledge, information, data, metadata, relationships, logs, annotations, user profiles, and documents, all which can be interpreted as distinct forms of digital objects, according to their particular structures, metadata, and streams. 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. Those 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 [45], 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.

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 [102]. 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 the societies which the DL must serve and determine a specific structure of libraries that gives particular scenarios for those users.

2.6.2 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 section are threefold: 1) to show the use of 5S as an analytical tool helpful to better comprehend specific DL phenomena; 2) to present the complex interplays that occur among 5S components and DL concepts in real DL application; and 3) to illustrate the possibility of using 5S as a instrument for requirements analysis in DL development.

2.6.2.1 Networked Digital Library of Theses and Dissertations (NDLTD)

The Networked Digital Library of Theses and Dissertations (NDLTD) [90, 80, 37] 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 there also are 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. 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.

NDLTD Society 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 potentially 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 (see next paragraph).

The 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 because of 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.

NDLTD Scenarios/Services 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 (including abstracts) about their ETDs. Staff in the graduate school and library also use other parts of the workflow software as they check, approve, archive, and catalog 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 digital library. 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 the MARIAN [38, 52, 51] and the Virtua [59] digital library 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) [77]. There is also experimental software, extending the SIFT package [134] from Stanford University and a prototype in the MARIAN system, to provide filtering and 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 [131, 132], Phronesis [43], and Emerge [42] may allow us to support other universities that choose to use those packages to provide access services for their local ETDs.

NDLTD 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 like searching relates to the conceptual space used in indexing. This will become more disciplined, as members use both some version of MARC, Dublin Core, or the new developed ETD-MS thesis and dissertations metadata standard [5], 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) [55]. 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 [57]. In addition, we have investigated how the library metaphor applies to using our collection in our 10x10x10' CAVE (virtual reality environment) [84].

NDLTD 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, which must be rendered as streams. These present challenges regarding quality of service if played back in real time, or alternative 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 [52, 53, 51].

NDLTD 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 Germany. Moreover, NDLTD has developed and is promoting the Interoperability Metadata Standard for Electronic Theses and Dissertations (ETD-MS) as a standard descriptive metadata set for describing electronic theses and dissertations [5]. Structures in the form of *semantic networks* are used inside MARIAN to represent ETD collections and metadata and are explored through the services provided.

2.6.2.2 Open Archives Initiative The Open Archives Initiative (OAI) [68, 29] 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) [85], 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.

Open Archives Initiative Society The main communities designed for the OAI society are electronic, namely active agents called harvesters and repositories, which interact through the Open Archives Initiative Protocol for Metadata Harvesting (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 e-print 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 and those involved with administrative aspects of repositories/archives.

Open 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 Open Archives Scenarios) 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.

Open Archives Structures Major structures of OAI are involved with *records*, *sets*, and *metadata formats*. OAI records can be considered containers [67], 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.

Open Archives 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/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 a **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.

Open Archives 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.

2.6.3 Declarative Generation of DLs

As a third application of the 5S framework, we have designed 5SL, a domain-specific, declarative language for conceptual modeling and generation of digital library applications [50]. 5SL is an XML serialization of 5S and has a formal semantics, which can be understood in terms of a translation of the language constructs and primitives into the 5S formalisms. Its formal basis provides an unambiguous and precise DL specification tool, which can facilitate prototyping, allow proofs of assertions, and aid validation of implementations.

In 5SL, the specification of a digital library encompasses five complementary dimensions, including: the kinds of multimedia information the DL supports (Stream Model); how that information is structured and organized (Structural Model); different logical and presentational properties and operations of DL components (Spatial Model); the behavior of the DL (Scenario Model); and the different societies of actors and managers of services that act together to carry out the DL behavior (Societal Model).

To improve acceptability and interoperability, 5SL makes extensible use of existing standard specification sublanguages for representing DL concepts, when it turns out to be possible. That possibility is defined by the ability to formally map those standards and sublanguages to 5S formal specifications. Moreover, the need for the integration of multiple languages is a key aspect of the domain-specific language approach [6]. 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 especially well. 5SL utilizes an XML syntax, whose abundance of supporting software tools facilitates the construction of DL generators. 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 [122] and/or RDF Schema [121] are the primary tools for describing structures. User Interface Markup Language (UIML) [3] and MathML [123] are used to represent some aspects of spaces. And finally, an adapted and extended version of UXF [115], an XML serialization of UML [14], is used with the Societal and Scenario Models.

The general process of automatic creation of DLs for a particular application is shown in Figure

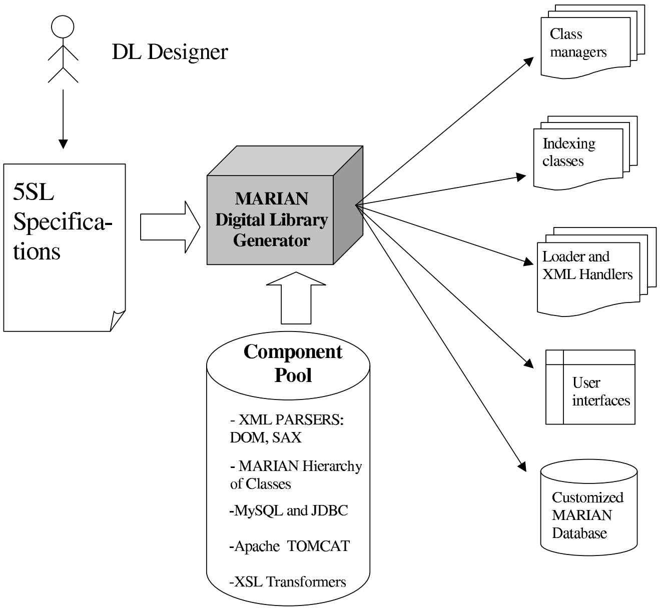


Figure 2: DL Generation Process with 5SL

2. Initially a DL designer is responsible for formalizing a conceptual description of the digital library using the language concepts. This phase is normally preceded by a 5S analysis of the DL requirements and characteristics as in the previous subsection. Declarative specifications in 5SL are then fed into a DL generator, to produce tailored DLs, suitable for specific platforms and requirements. These are built upon a collection of stock parts and configurable components that provide the infrastructure for the new DL. This infrastructure includes the (e.g., MARIAN [54, 30]) classes of objects and relationships that make up the DL, and processing tools to create/load the actual library collection from raw documents, as well as services for searching, browsing, and collection maintenance.

5SL is in its infancy but we already have used it to build pilot systems and prototypes. In one of these, we built a 5SL generator for the MARIAN digital library system [51] and used 5SL to design a union archive for a federation of ETD sites in NDLTD. In this union archive, metadata in ETD-MS format is periodically harvested from ETD sites using the Open Archives Initiative Protocol for Metadata Harvesting. The MARIAN system works as a portal for accessing the collection. In this particular application, the component pool includes XML parsers, an OAI harvester and the MARIAN digital library API. MARIAN is built around a semantic network model, involving labeled digraphs or structures in 5S terminology, improved with weights and a hierarchy of classes. Any collection of nodes or links in a network can be weighted to represent how well they suit some description or fulfill some role.

The MARIAN DL generator, which is based on a DOM [117] XML parser, automatically generates four kinds of output for the 5SL model of the NDLTD union archive (Figure 2):

1. Class managers and indexing classes for the NDLTD application

   This includes class managers to represent the MARIAN semantic network view of the ETD-MS descriptive metadata standard. Class managers define the logical schema of the DL application, which in MARIAN corresponds to a set of Java classes that represent digital objects, their component parts, and linking information. Class managers also store and maintain instance objects of their class, and function as the search engines for the system. Indexing classes also are generated and are represented as sets of bipartite weighted semantic networks involving document parts and document features.

2. Collection Loader

   The loader is an automatically generated SAX event handler that checks incoming XML documents against specifications, extracts structuring and indexing information from valid documents, including controlled authorities like person's names and subject headings, and invokes corresponding class manager loading methods which materialize and incrementally update structures and indexes, and manage underlying databases.

3. User interfaces

   For user interfaces there are HTML web query forms for structured searches based on document and metadata structures, and classes for flat representations of document/metadata with methods for presenting different views of them using generated XSL stylesheets.

4. Tailored Databases

   Finally, a set of customized tables are created which tailor the MARIAN general database schema for the specific structures of the DL applications at hand.

MARIAN architecture and features as well as a complete specification and generation of a digital library with 5SL [50] are out of the scope of this paper. Figure 3, however helps to give an overview idea of the generation process, by showing a portion of the 5SL description for ETD-MS, the ETD metadata standard. In this particular case, we use an XML Schema for describing and generating the MARIAN semantic network representation of the descriptive metadata.

```
<xsd:element name="thesis">
  <xsd:complexType>
    …
      <xsd:element name="identifier" type="xsd:string"/>
      <!—thesis author -->
      <xsd:element ref="creator" maxOccurs="unbounded"/>
      <!-- abstract -->
      <xsd:element ref="description"/>
      <!—Subject Heading -->
      <xsd:element ref="subject" minOccurs="0"
maxOccurs="unbounded"/>
      <element name="degree">
        <xsd:complexType>
          <element ref="name"/>
           <element ref="level"/>
           <element ref="discipline"/>
           <element ref="grantor"/>
        </xsd:complexType>
      </element>
…
  </xsd:complexType>
</xsd:element>
<xsd:element name="“person" type="mapi:ControlledText"/>
<xsd:element name="“description" type="mapi:EnglishText"/>
<xsd:element name="subject" type="mapi:ControlledText"/>
…
```

Figure 3: Portion of an XML schema defining the structure of ETD-MS, the electronic thesis and dissertation descriptive metadata standard

One important feature of the MARIAN generator is its use of XML namespaces and the MARIAN API. The MARIAN hierarchy of class managers (or API) defines a set of basic types for semantic networks (e.g., nodes, unweighted links), information retrieval (e.g., weighted links, weighted sets), and digital library systems (e.g., controlled strings like personal names and subject headings, English and non-English terms, phrases, etc.). 5SL descriptions use namespaces to import MARIAN types to specify properties of the many different parts of documents and metadata records. These properties include specific matching methods, as well as methods for management of indexes, databases, and sets of instances of the particular class/type. These features tremendously facilitate the process of DL construction and maintenance.

To be more specific, in the example above, XML references are mapped to MARIAN unweighted link class managers (e.g., hasCreator, hasDescription), XML complex types to general MARIAN network nodes, typed elements to sink nodes that inherit behavior (including loading and match) from the corresponding MARIAN class/type. Weighted links classes (e.g., occursInCreator, occursInDescription) for indexing element contents in actual XML documents also are generated. Therefore, in MARIAN, structure, content, and behavior are all represented by the use of weighted semantic networks in conjunction with a hierarchy of classes and a powerful API.

By using these techniques, we already have automatically created several digital library applications. Most developed are those for the NDLTD union archive and the National Library of Medicine DIRLINE collection. We have plans to host collections and searching and browsing services for the PhysDoc collection in Germany, and the Virginia Tech Library catalog, among others. Work on these applications successfully demonstrated the feasibility of the 5SL generation process in conjunction with MARIAN and its component pool. However the current 5SL design has its limitations. First, XML is very verbose and 5SL design of complex digital libraries can be very cumbersome. We are working on a user interface for graphical manipulation of 5S constructs that will automatically generate 5SL code. Second, the current XML schema implementation is awkward for representing structural metadata other than containment relationships. We expect that the use of RDF in specific cases or the explicit introduction of XML elements for links or associations such as those in the topic map markup language [112] will help with the problem. Finally, searching and browsing services were taken for granted due to the powerful MARIAN digital library API. More work is necessary to investigate the process of generating DL prototypes and implementations for different and more complex scenarios/services using different component pools.

Nevertheless, it seems clear that 5S has intuitive appeal and practical application. In the next sections, we continue to explore 5S further with a more formal treatment.

3 The 5S Formal Framework

In this section, we proceed to precisely and unambiguously formalize most of the informal digital library concepts introduced in previous sections. Figure 4 shows a map of most important concepts and formal definitions. Each concept is associated with the corresponding section number of its formal definition; arrows mean that a concept is formally defined in terms of previously defined concepts that point to it [3].

3.1 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 later definitions

[3] The notion of a tuple (def. 4) is used in almost all other definitions, so, for simplicity, we are not showing arrows coming out of that particular concept in the figure. Other popular definitions are treated likewise.

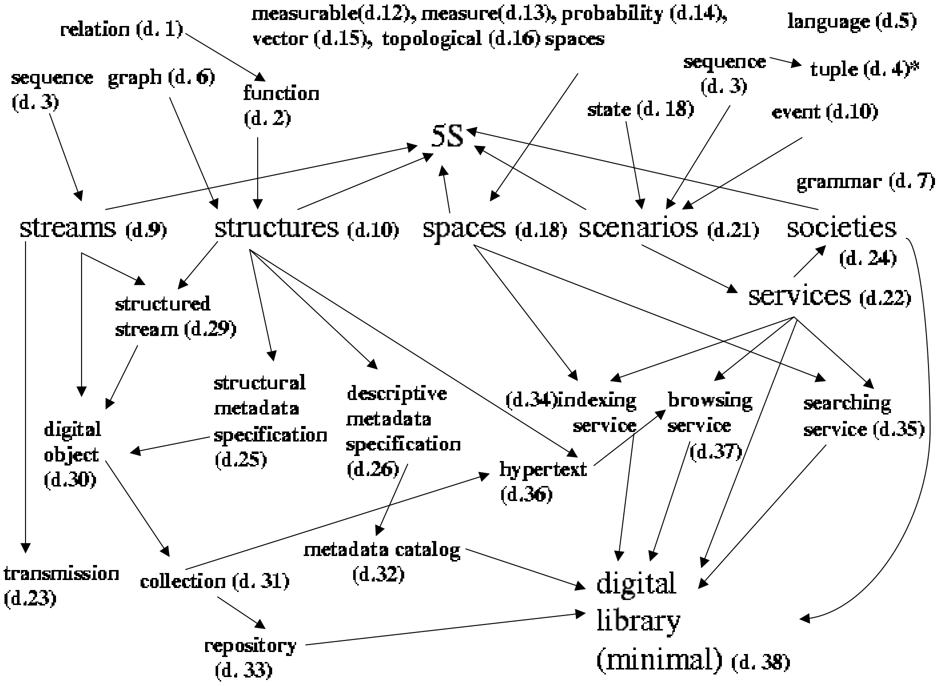


Figure 4: 5S map of formal definitions

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 [23]. Readers familiar with these concepts can skip this section or simply refer to it as needed when some of the concepts are used in higher level 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.

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 $A_1 \times A_2 \times ... \times A_n$.*

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.*

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.*

Definition 4 *A* ***tuple*** *is a finite sequence that is often denoted by listing the range values of the function as $\langle f(1), f(2), ..., f(n)\rangle$.*

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^*.*

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 $E' \subseteq E$.

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).

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 being **accepted**.

3.2 5S Formalisms

Definition 9 *A **stream** is a sequence whose codomain is a nonempty set.*

Definition 10 *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 11 *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 12 *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 $A_i \in \mathbb{B}$ ($i = 1, 2, 3, \ldots$), 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 $\mathbb{B}$.

Definition 13 *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.

Definition 14 *A **measure space** $(X, \mathbb{B}, \mu)$ is a measurable space $(X, \mathbb{B})$, with measure μ defined on $\mathbb{B}$.*

Definition 15 *A **probability space** is a measure space $(X, \mathbb{B}, \mu)$, such that measure $\mu(X) = 1$.*

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 [25].

Definition 16 *A **vector space** is a set V (whose elements are called **vectors**) together with a field of "scalars"* [4] *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).*

Vector spaces are the basis for a widely used information retrieval model, the Vector Space Model (VSM) [100]. 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* $s : D \times D \to \mathbb{R}$ between documents.

Definition 17 *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 $\mathcal{T}$.*

[4] In this context, the filed of real numbers.

$\mathcal{T}$ 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.

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 18 *A* ***space*** *is a measurable space, measure space, probability space, vector space, topological, or a metric space.*

Definition 19 *A* ***system state*** *(from now on, just state) is a function $s : L \to V$, from labels L to values V. A* ***state set*** *S consists of a set of state functions $s : L \to V$.*

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 paper. 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 20 *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 [23]. Rather, it is more like the events in networked operating systems theory [109], transitions in finite state machines [27], those modeled by the Unified Modeling Language (UML) [14], or transitions between places in Petri Nets [87].

The condition is used to describe circumstances under which a state transition can take place. An action models a reference to an operator, 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 21 *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 must be a pseudodigraph [5], since loops (s_i, s_i) are possible as events.)

Definition 22 *A* ***service, activity, task, procedure,*** *or* ***operation*** *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" [58] and a "use case" in the UML [14]. In this framework, a simple transmission service of streams can be formally specified as:

Definition 23 *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 [99]. 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 [130] and object-oriented abstractions [98] offer a programming language perspective for the question of formal scenario implementation.

Definition 24 *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, activities, components, hardware, software, data);*

[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

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. Definition 22) that describes the interactions or communications among individuals.

The second part of the definition emphasizes the collaborative nature of societies such 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). (cf. Section 4 for more formal examples of Societies.)

3.3 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 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 is a managed *collection* of information with associated *services* involving communities where information is stored in digital formats and accessible over a network [4]. 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. Metadata have been informally defined as data about other data. 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 digital library. 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 the 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.*

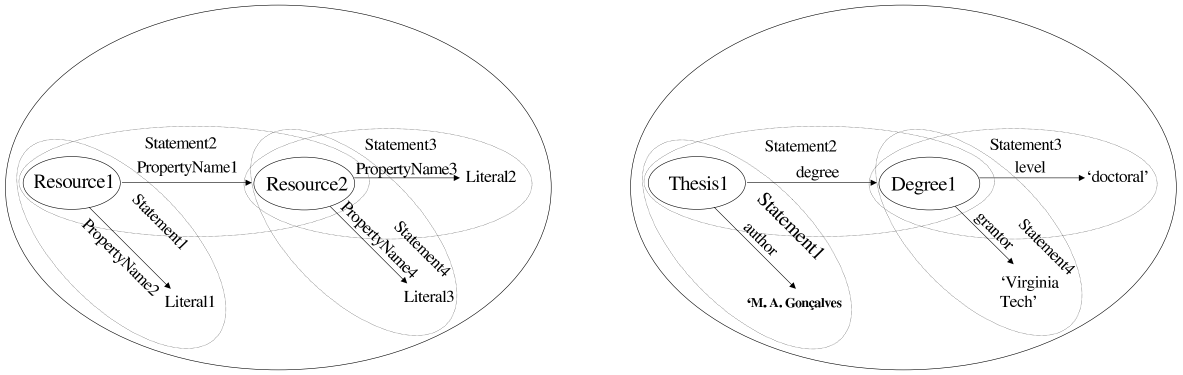
Definition 25 *A **Structural metadata** specification is a structure.*

This simple definition emphasizes the role of structural metadata as a representation or abstraction of relationships between digital objects and their component parts (cf. Definition 30). The graph-based representation of this type of metadata can be explicitly expressed, as in the case of markup [22], or implicitly computed [79, 20].

The next definition, for **descriptive metadata specifications**, is inspired by new developments in the metadata area, mainly those related to the *Semantic Web* [13] and the Resource Description Framework (RDF) [111, 127], and emphasizes the semantic relationships implied by the labeling function in a structure. Figure 5a 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 5b 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 26 *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. *The $\mathcal{F} : (V \cup E) \rightarrow (\mathcal{R} \cup \mathcal{L} \cup \mathcal{P})$ 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.*



(a) (b)

Figure 5: Overview of the descriptive metadata model with example (adapted from [17])

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 as a literal).*

Definition 27 *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 resources 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 [18]. 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 28 *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* $\mathcal{R}_{MS} \subseteq \mathcal{R}_{MF}$, $\mathcal{L}_{MS} \subseteq \mathcal{L}_{MF}$,

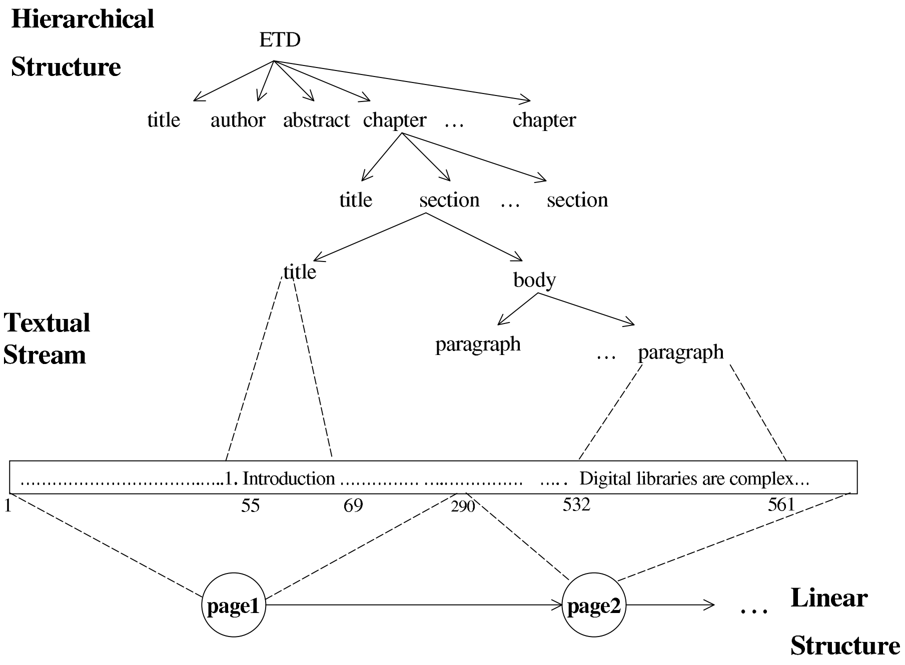


Figure 6: A StructuredStream for a Electronic Dissertation (adapted from [79])

$\mathcal{P}_{MS} \subseteq \mathcal{P}_{MF}$, 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 29 *Given a structure $(G, L, \mathcal{F})$, $G = (V, E)$ and a stream S, a* ***StructuredStream*** *is a function $V \to (\mathbb{N} \times \mathbb{N})$ 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 6. 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 [79]. Finally, it should be noted that this definition works also for multimedia streams like audio, video, and images.

Definition 30 *A* ***digital object*** *is a tuple $do = (h, SM, ST, StructuredStreams)$, where*

1. *$h \in H$, where H is a set of universally unique handles (labels);*

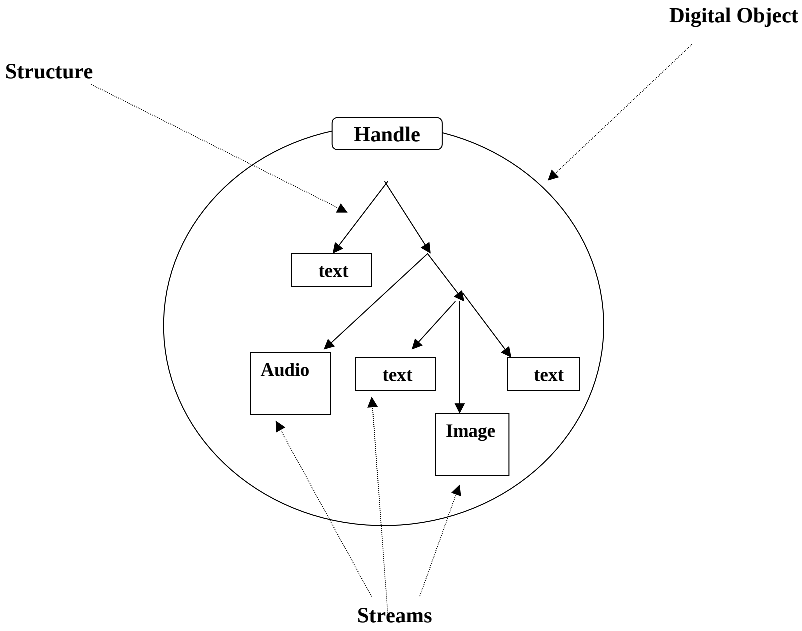


Figure 7: A simple digital object

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 7 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 a digital object (e.g., [69, 81, 82]) where operations, like different disseminations or exporting of subparts, are performed by external entities, like the user interface or the repository (cf. Definition 30). While there is no explicit restriction regarding this, the definition does conform to our minimalist

approach.

Definition 31 *A* ***collection*** $C = \{do_1, do_2, \ldots, do_k\}$ *is a set of digital objects.*

Definition 32 *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 33 *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:*

1. *get* $: H \to C$ *maps a handle* h *to a digital object get(h).*
2. *store* $: C \times R \to R$ *maps* $(do, \tilde{C})$ *to the augmented collection* $\{do\} \cup \tilde{C}$*.*
3. *del* $: H \times R \to R$ *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 34 *Let* $I : 2^{\mathcal{T}} \to 2^H$ *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 state functions).*

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.

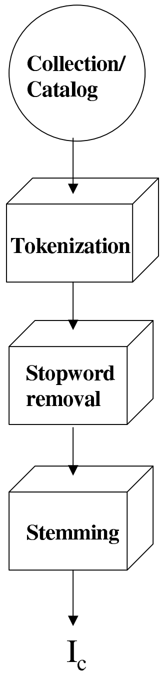


Figure 8: Simple indexing service

A very simple instance of such an indexing service is shown in Figure 8 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 of query terms [8]. Each one of the scenarios can be thought of as doing some transformation (e.g., graph transformation) in the representations of digital objects in order to produce the index function. Note again that we are making use our minimalist approach by not considering complex indexes, for example, defining locations inside streams of a digital object for phrase, proximity, or structural queries.

Definition 35 *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 search 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, $do_k(2)$ denote 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 36 *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:*

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* [126]. 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 9 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 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 [106]. It should be noted that for the sake of brevity we are not describing here links to services, for example, external plugins that can be invoked by browsers or Web forms.

Definition 37 *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 can for example unify the three modes of browsing defined by Baeza-Yates and Ribeiro-Neto [8]: flat browsing, structured guided, and navigational mode. The third one is the most general case and fits exactly our model. 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. It is, for example, the work of many semi-structured

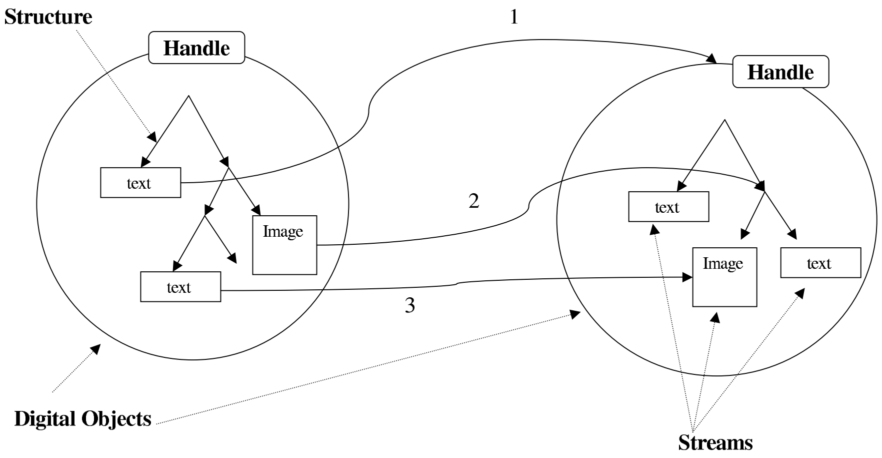


Figure 9: A simple hypertext

wrapper algorithms to 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 [93] 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.

Definition 38 *A **digital library** is a 4-tuple* $(\mathcal{R}, DM, Serv, Soc)$*, where*

- $\mathcal{R}$ *is a repository;*
- $DM = \{DM_{C_1}, DM_{C_2}, ..., DM_{C_K}\}$ *is a set of metadata catalogs for all collections* $\{C_1, C_2, ..., C_K\}$ *in the repository;*

[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).

- *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 should the scenarios in $Serv$ be built from $\mathcal{R}$ and DM and from the relationships among communities inside the society Soc, or what are the consistency rules among digital objects in collections of $\mathcal{R}$ and metadata records in DM?) are not specified here. Those will be a subject of future research.

4 Example: Formal Treatment of Open Archives and the NDLTD Union Archive

4.1 Open Archives Initiative

Definition 39 *Let $dl = (\mathcal{R}, DM, Serv, Soc)$ 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 $OAI\_Harvesting \in dl(4) = Soc$ whose behavior is defined by the Open Archives Protocol for Metadata Harvesting (see below); and*
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 are ruled by the OAI Protocol for Metadata Harvesting and constitute the OAI harvesting service $OAI\_Harvesting$. $OAI\_Harvesting = \{Identify,\ ListMetadataFormats,\ ListSets,\ ListIdentifiers,\ ListRecords,\ GetRecord\} \in Serv$ is a service defined by the Open Archives Initiative Protocol for metadata harvesting and encompasses six scenarios, formally defined below:

1. $Identify$

   Goal: Returns general information about the archive (what in OAI terms corresponds to the repository $\mathcal{R}$ along with a metadata catalog $DM_{C_k} \in DM$ 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 action in *dp* of *dl*, e_2 is the event corresponding to the response from the data provider *dp*, p : specifies the corresponding action that is being invoked, and *identification* is a parameter of the response action. 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 [85]:

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

4. ListRecords

   Goal: Retrieves metadata for multiple records.

   Scenario: $\langle e_1 : p = ListRecords(from, until, set, metadataPrefix, resumptionToken), e_2 : p = response(\{oai-record_1, ..., oai-record_k\}, resumptionToken_i)\rangle$. Each $oai-record_i$ is a 3-tuple ($header_i$, $metadata_i$, $about_i$, $status_i$) where:

   (a) $header_i$ is a 4-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 $ListRecords$ apply 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.

4.2 NDLTD Union Archive

- A digital library federation is a set $DLF = \{dl_1, dl_2, ..., dl_k\}$ of independent and possibly heterogeneous digital libraries (DLs). NDLTD is a digital library federation where each independent DL $dl_k = (ETD\_\mathcal{R}_k, ETD\_DM_k, ETD\_Serv_k, ETD\_Soc_k)$. $ETD\_\mathcal{R}_k$ is a repository having a collection $ETD\_Coll_k = \{etd_{1k}, etd_{2k}, ..., etd_{jk}\}$ composed of a set of digital objects etd_{ik} corresponding to electronic theses and/or dissertations (ETDs). The possible set of streams of an ETD, $etd_{ik}(2)$, 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 $etd_{ik} \in ETD\_Coll_k$ there should be at least one $etd\_dm_{ik} \in ETD\_DM_{k_{ETD\_Coll_k}}$, $ETD\_DM_{k_{ETD\_Coll_k}} \in ETD\_DM_k$, $ETD\_DM_{k_{ETD\_Coll_k}}$ being a metadata catalog for the ETD collection ETD_Coll_k.
- NDLTD promotes ETD-MS as the metadata format for ETD descriptive metadata specifications. For each dl_k in NDLTD, let:
  - $ETD\_IDs_k = \{h_{ik} | h_{ik} = etd_{ik}(1), etd_{ik} \in ETD\_Coll_k\}$, $NDLTD\_ETD\_IDs = \bigcup_{dl_k \in NDLTD} ETD\_IDs_k$ 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 $(r, p, z) \in def_{ETD}$:

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

- Society ETD_Soc_k of dl_k is such that {Patron, Student, ETDReviewer, ETDCataloguer, ETDSearchManager, ETDWorkflowManager,...} $\subset ETD\_Soc_k(1)$ and $\{creates = (Student \times ETDWorkflow, ETDCreation), searches = (Patron \times ETDSearchManager, searching), is\_a = (Student \times Patron, \emptyset)\} \subset ETD\_Soc_k(2)$.

- The NDLTD Union Archive is a tuple $(NDLTD\_Union, UA\_Harvester)$ where $NDLTD\_Union = \bigcup_{dl_k \in NDLTD} ETD\_DM_{k_{ETD\_Coll}}, ETD\_DM_k = dl_k(2), ETD\_DM_{k_{ETD\_Coll}} \in ETD\_DM_k$, 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 $UA\_Harvester$. Conversations between the $UA\_Harvester$ and dp_k are governed by the OAI Protocol for Metadata Harvesting and constitute an OAI harvesting service as defined in the previous section.

5 Related Work

Formal models, 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 [125] 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 formal language Z [110]. Kalinichenko *et al.* [62] presented a canonical model for information systems and a compositional approach that they applied to provide a partial solution for interoperability in DLs. Castelli *et al.* [18] have presented the closest work 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 comprehensive formalization for the digital libraries field.

Formal models precisely and unambiguously define the semantics of specific abstractions of a knowledge field. In the case of computer science (CS), this allows for the exploitation and development of declarative approaches in design and development. Not surprisingly, each of the cited CS sub-fields has proposals to investigate declarative approaches. Accordingly, we have proposed 5SL for declarative specification and generation of digital library applications. Closely related works, which are not supported by a rigorous underlying formal theory, include the Digital Library Definition Language [137], the SearchDL interface [91], and FEDORA's structoid approach [32]. All of these deal with small parts of the whole problem (e.g., federated search, digital objects rendering) and are not as comprehensive as the 5S model in dealing with almost all aspects of DL design and construction.

The flexibility of the 5S theory 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., [26, 31, 47, 72]), to the best of our knowledge there is no other comprehensive DL taxonomy published in the literature, other than that presented in [36], which served as a basis for ours.

6 Conclusions

Motivated by the challenge of Licklider [73] to construct a theory for digital libraries, we have developed 5S. We show that formal definitions allow the 5S framework to be fully described and make it possible to clearly and formally define a minimal digital library. 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 construct 5SL – a declarative specification language from which digital libraries can be generated, and to formally define a set of DL constructs and settings in the context of the NDLTD Union Archive.

Future work with the 5S framework will proceed in several directions. We will use our framework to help guide further development of the OAI and the NDLTD, as well as other digital library applications such as NSDL [135]. We will extend 5SL to be more complete, and to enable generation of personalized digital libraries in connection with PIPE [93, 55]. Further, we will encourage and assist others to adopt and adapt 5S and 5SL.

Finally, we plan to continue our work on the theory of digital libraries. We will explore qualitative aspects of the model and language including consistency, completeness, correctness, and evaluation. We also intend to use 5S to help with formal analyses of interoperability issues in digital libraries. It can serve as a canonical DL model to allow us to go beyond the current shallow descriptions of DL systems for federated search or harvesting purposes. The formal definitions given here can be used to prove helpful lemmas and theorems, and to guide future work in the field.

Acknowledgements

We are indebted to Robert K. France for many constructive discussions. The authors wish to thank the Digital Library Research Laboratory team members that assisted in this process. Anonymous reviewers of earlier drafts helped shape this work significantly. This material is based upon work supported by the National Science Foundation under Grants No. IIS-9986089, IIS-0002935, IIS-0080748, IIS-0086227, DUE-0121679, DUE-0121741, and DUE-0136690. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The first author also is supported by CAPES, process 1702-980.

References

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

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

[3] Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster. UIML: an appliance-independent XML user interface language. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1695–1708, May 1999.

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

[5] Anthony Atkins, Edward Fox, Robert France, and Hussein Suleman. Interoperability metadata standard for electronic theses and dissertations. http://www.ndltd.org/standards/metadata/current.html, 2001.

[6] 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.

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

[8] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley-Longman, May 1999.

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

[10] Murat Bayraktar, Chang Zhang, Bharadwaj Vadapalli, Neill A. Kipp, and Edward A. Fox. A web art gallery. In *DL'98: Proceedings of the 3rd ACM International Conference on Digital Libraries*, pages 277–278, Pittsburgh, PA, 1998.

[11] Catriel Beeri. A formal approach to object-oriented databases. *IEEE Data and Knowledge Engineering*, 5:353–382, December 1990.

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

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

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

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

[16] 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.

[17] K. Selçuk Candan, Huan Liu, and Reshma Suvarna. Resource description framework: metadata and its applications. *ACM SIGKDD Explorations Newsletter*, 3:1, 2001.

[18] Donatetella Castelli, Carlo Meghini, and Pasquale Pagano. Foundations of a multidimensional query language for digital libraries. *Lecture Notes in Computer Science*, 2458:251–265, 2002.

[19] 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.

[20] Charles L. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38:43–56, 1995.

[21] E. F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[22] J. H. Coombs and A. H. Renear S. J. DeRose. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947, November 1988.

[23] Thomas H. Cormen, Charles Eric Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, Cambridge, Massachusetts, 1990.

[24] Andy Crabtree, Michael B. Twidale, Jon O'Brien, and David M. Nichols. Talking in the library: implications for the design of digital libraries. In Robert B. Allen and Edie Rasmussen, editors, *Proceedings of the 2nd ACM International Conference on Digital Libraries*, pages 221–229, New York, July 1997. ACM Press.

[25] 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, December 1998.

[26] Lynne Davis and Melissa Dawe. Collaborative design with use case scenarios. In *JCDL'01: Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, Studying the Users of Digital Libraries: Formative and Summative Evaluations, pages 146–147, 2001.

[27] M. D. Davis, R. Sigal, and E. J. Weyuker. *Computation, Complexity, and Languages (second edition)*. Academic Press, 1994.

[28] Maria Cristina Ferreira de Oliveira, Marcelo Augusto Santos Turine, and Paulo Cesar Masiero. A statechart-based model for hypermedia applications. *ACM Transactions on Information Systems*, 19(1):28–52, 2001.

[29] Herbert Van de Sompel and Carl Lagoze. The Santa Fe Convention of the Open Archives Initiative. *D-Lib Magazine*, 6(2), February 15, 2000.

[30] MARIAN development team at Virginia Tech. Marian digital library system. http://www.dlib.vt.edu/products/marian.html, 2002.

[31] Andy Dong and Alice M. Agogino. Design principles for the information architecture of a SMET education digital library. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 314–321, Roanoke, Virginia, June 24-28, 2001.

[32] Naomi Dushay. Using structural metadata to localize experience of digital content. Technical Report cs.DL/0112017, The Computing Research Repository (CoRR), 2001.

[33] D. Ellis. The physical and cognitive paradigms in information retrieval research. *Journal of Documentation*, 48:45–64, 1992.

[34] D. J. Foskett. Thesaurus. In *Encyclopedia of Library and Information Science - Volume 30*, pages 416–462. Marcel Dekker, New York, 1980.

[35] E. A. Fox and G. Marchionini. Toward a worldwide digital library. *Communications of the ACM*, 41(4):22–28, April 1998.

[36] Edward A. Fox, Robert M. Akscyn, Richard K. Furuta, and John J. Leggett. Digital libraries. *Communications of the ACM*, 22-28(4):1022–1036, 1995.

[37] Edward A. Fox, John L. Eaton, Gail McMillan, Neill A. Kipp, Paul Mather, Tim McGonigle, William Schweiker, and Brian DeVane. Networked digital library of theses and dissertations: An international effort unlocking university resources. *D-Lib Magazine*, 3(9), 1997.

[38] Edward A. Fox, Robert K. France, Eskinder Sahle, Amjad Daoud, and Ben E. Cline. Development of a modern OPAC: From REVTOLC to MARIAN. In *Proceedings of the*

Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Interface Issues, pages 248–259, 1993.

[39] N. Fuhr. XIRQL - An Extension of XQL for Information Retrieval. In *Proceedings of the ACM SIGIR 2000 – Workshop on XML and Information Retrieval*, Athens, Greece, 2000. ACM Press.

[40] R. Furuta, V. Quint, and J. Andre. Interactively editing structured documents. *Electronic Publishing—Origination, Dissemination, and Design*, 1(1):19–44, April 1989.

[41] Richard Furuta. Defining and using structure in digital documents. In *Proceedings of the First Annual Conference on the Theory and Practice of Digital Libraries*, College Station, Texas, 1994.

[42] Joe Futrelle, Su-Shing Chen, and Kevin C. Chang. NBDL: A CIS Framework for NSDL. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'2001)*, pages 124–125, Roanoke, Virginia, June 24-28, 2001.

[43] David A. Garza-Salazar. Phronesis. http://copernico.mty.itesm.mx/ tempo/Proyectos/, 2001.

[44] H. Gladney, E. A. Fox, Z. Ahmed, R. Ashany, N. J. Belkin, and Maria Zemankova. Digital library: Gross Structure and Requirements: Report from a March 1994 Workshop. In *Proceedings of Digital Libraries '94: the First Annual Conference On the Theory and Practice of Digital Libraries*, pages 101–107, College Station, Texas, 1994.

[45] H. M. Gladney and A. Cantu. Authorization management for digital libraries. *Communications of the ACM*, 44(5):63–65, May 2001.

[46] R. Godement. *Algebra.* Kershaw Publ. Co. Ltd, London, 1969.

[47] Dion Goh and John Leggett. Patron-augmented digital libraries. In *DL'00: Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 153–163, San Antonio, TX, USA, 2000.

[48] Charles F. Goldfarb and Paul Prescod. *The XML Handbook.* Prentice-Hall PTR, Upper Saddle River, NJ 07458, USA, 1998.

[49] R. Goldman and J. Widom. Interactive query and search in semistructured databases. *Lecture Notes in Computer Science*, 1590:52–63, 1999.

[50] Marcos André Gonçalves and Edward A. Fox. 5SL – A language for declarative specification and generation of digital libraries. In *Proceedings of the Second ACM/IEEE-CS*

Joint Conference on Digital Libraries (JCDL'2002), pages 263–272, Portland, Oregon, July 14-18, 2002.

[51] Marcos André Gonçalves, Robert K. France, and Edward A. Fox. MARIAN: Flexible Interoperability for Federated Digital Libraries. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, pages 173–186, Darmsdadt, Germany, 2001. Springer.

[52] Marcos André Gonçalves, Robert K. France, Edward A. Fox, and Tamas E. Doszkocs. MARIAN Searching and Querying across Heterogeneous Federated Digital Libraries. In *First DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, Zurich, Switzerland, December 2000.

[53] Marcos André Gonçalves, Robert K. France, Edward A. Fox, Eberhard R. Hilf, Michael Hohlfeld, Kerstin Zimmermann, and Thomas Severiens. Flexible interoperability in a federated digital library of theses and dissertations. In *Proceedings of the 20th World Conference on Open Learning and Distance Education, The Future of Learning - Learning for the Future: Shaping the Transition, ICDE2001*, April 1-5, 2001.

[54] Marcos André Gonçalves, Paul Mather, Jun Wang, Ye Zhou, Ming Luo, Ryan Richardson, Rao Shen, Liang Xu, , and Edward A. Fox. Java MARIAN: From an OPAC to a modern digital library system. In *Proceedings of the 9th International Symposium on String Processing and Information (SPIRE'2002)*, pages 194–209, Lisbon, Portugal, September 11-13 2002.

[55] Marcos André Gonçalves, Ali A. Zafer, Naren Ramakrishnan, and Edward A. Fox. Modeling and Building Personalized Digital Libraries with PIPE and 5SL. In *Proceedings of the Joint DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, Dublin, Ireland, June 18-20, 2001.

[56] Giovanna Guerrini, Elisa Bertino, Barbara Catania, and Jesus Garcia-Molina. A formal model of views for object-oriented database systems. *Theory and Practice of Object Systems*, 3(3):157–183, 1997.

[57] Lenwood S. Heath, Deborah Hix, Lucy T. Nowell, William C. Wake, Guillermo A. Averboch, Eric Labow, Scott A. Guyer, Dennis J. Brueni, Robert K. France, Kaushai Dalal, and Edward A. Fox. Envision: A user-centered database of computer science literature. *Communications of the ACM*, 38(4):52–53, April 1995.

[58] Pei Hsia, Jayarajan Samuel, Jerry Gao, David Kung, Yasufumi Toyoshima, and Cris Chen. Formal approach to scenario analysis. *IEEE Software*, 11(2):33–41, March 1994.

[59] VTLS Inc. VTLS. http://www.vtls.com, 2001.

[60] Karen Sparck Jones and Peter Willett, editors. *Readings in Information Retrieval.* Multimedia Information and Systems. Morgan Kaufmann Publishers, 1997.

[61] B. Kahin and H. R. Varian. *Internet Publishing and Beyond: The Economics of Digital Information and Intellectual Property.* MIT Press, Cambridge, Massachusetts, 2000.

[62] 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 *Proceedings of the 2nd Russian Scientific Conference on Digital Libraries: Advanced Methods and Technologies*, 2000.

[63] Henry Kautz, Bart Selman, and Mehul Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, March 1997.

[64] T. Kochtanek and K. K. Hein. Delphi study of digital libraries. *Information Processing and Management*, 35(3):245–254, 1999.

[65] 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.

[66] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, and Altigran Soares da Silva. DEByE - data extraction by example. *Data and Knowledge Engineering*, 40(2):121–154, 2002.

[67] Carl Lagoze. The Warwick framework: A container architecture for diverse sets of metadata. *D-Lib Magazine*, 2(7), July 15, 1996.

[68] Carl Lagoze and Herbert Van de Sompel. The Open Archives Initiative. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'2001)*, pages 54–62, Roanoke, Virginia, June 24-28, 2001.

[69] Carl Lagoze, David Fielding, and Sandra Payette. Making global digital libraries work: Collection services, connectivity regions, and collection views. In Ian Witten, Rob Akscyn, and Frank M. Shipman, editors, *Proceedings of the 3rd ACM Conference on Digital Libraries (DL-98)*, pages 134–143, New York, June 23–26 , 1998. ACM.

[70] A. V. Lamsweerde and L. Willemet. Inferring declarative requirements specifications from operational scenarios. *IEEE Transactions on Software Engineering*, 24(12):1089–1114, December 1998.

[71] M. Lesk. Expanding digital library research: Media, genre, place and subjects. In *Proceedings of the International Symposium on Digital Libraries 1999: ISDL'99*, Tsukuba, Ibaraki, Japan, September 28-29, 1991.

[72] D. M. Levy and C. C. Marshall. Going digital: a look at assumptions underlying digital libraries. *Communications of the ACM*, 38(8):77–84, April 1995.

[73] J. C. R. Licklider. *Libraries of the Future.* MIT Press, Cambridge, Massachusetts, 1965.

[74] Raymond A. Lorie. Long Term Preservation of Digital Information. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'2001)*, pages 346–352, Roanoke, Virginia, June 24-28, 2001.

[75] Dario Lucarella and Antonella Zanzi. A visual retrieval environment for hypermedia information systems. *ACM Transactions on Information Systems*, 14(1):3–29, January 1996.

[76] Wendy E. Mackay and Michel Beaudouin-Lafon. DIVA exploratory data analysis with multimedia streams. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI-98) : Making the Impossible Possible*, pages 416–423, New York, April 18-23, 1998. ACM Press.

[77] Todd Miller. Annotation system for a collection of ETDs. http://www.ndltd.org/ndltd-sc/990416/annsystem.pdf, 1999.

[78] Ian Muslea, Steve Minton, and Craig Knoblock. A hierarchical approach to wrapper induction. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 190–197, Seattle, WA, USA, 1999. ACM Press.

[79] Gonzalo Navarro and Ricardo Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems*, 15(4):400–435, 1997.

[80] NDLTD. Networked digital library of theses and dissertations homepage. http://www.ndltd.org, 2001.

[81] Michael L. Nelson, Kurt Maly, Mohammad Zubair, and Stewart N. T. Shen. SODA: Smart objects, dumb archives. In S. Abiteboul and A.-M. Vercoustre, editors, *Proc. 3rd European Conf. Research and Advanced Technology for Digital Libraries, ECDL*, number 1696 in Lecture Notes in Computer Science, LNCS, Paris, France, September 1999. Springer-Verlag.

[82] Micheal L. Nelson and Kurt Maly. Buckets: Smart objects for digital libraries. *Communications of the ACM*, 44(5):60–61, May 2001.

[83] Svetlozar Nestorov, Serge Abiteboul, and Rajeev Motwani. Inferring structure in semistructured data. *SIGMOD Record*, 26(4):39–43, 1997.

[84] Fernando Das Neves and E. A. Fox. A study of user behavior in an immersive virtual environment for digital libraries. In *Proceedings of the Fifth ACM Conference on Digital Libraries (ACM DL'2000)*, pages 103–112, 2000.

[85] OAI. Open Archives Initiative protocol for metadata harvesting - v.2.0. http://www.openarchives.org/OAI/openarchivesprotocol.html, 2001.

[86] Douglas Oard, Carol Peters, Miguel Ruiz, Robert Frederking, Judith Klavans, and Praic Sheridan. Multilingual Information Discovery and AccesS (MIDAS): A joint ACM DL'99 / ACM SIGIR'99 workshop. *D-Lib Magazine*, 5(10), October 15, 1999.

[87] Andreas Oberweis and Peter Sander. Information system behavior specification by high-level Petri nets. *ACM Transactions on Information Systems*, 14(4):380–420, October 1996.

[88] Ryuichi Ogawa, Hiroaki Harada, and Asao Kaneko. Scenario-based hypermedia: A model and a system. In *Proceedings of the ECHT'90 European Conference on Hypertext*, Toolkits for Hypermedia Applications, pages 38–51, 1990.

[89] Yannis Papakonstantinou and Vasilis Vassalos. Query rewriting for semistructured data. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data: SIGMOD '99, Philadelphia, PA, USA, June 1–3, 1999*, volume 28(2), pages 455–466, New York, NY 10036, USA, 1999. ACM Press.

[90] C. Phanouriou, N. A. Kipp, O. Sornil, P. Mather, and E. A. Fox. A Digital Library for Authors: Recent Progress of the Networked Digital Library of Theses and Dissertations. In *Proceedings ACM Digital Libraries '99*, pages 20–27, Berkeley, CA, Aug 11-14, 1999.

[91] J. Powell and E. A. Fox. Multilingual federated searching across heterogeneous collections. *D-Lib Magazine*, 4(8), 1998.

[92] Robert Prince, Jianwen Su, Hong Tang, and Yonggang Zhao. The design of an interactive online help desk in the Alexandria Digital Library. In Dimitrios Georgakopoulos, Wolfgang Prinz, and Alexander L. Wolf, editors, *Proceedings of the International Joint Conference on Work Activities and Collaboration: WACC '99*, pages 217–226, San Francisco, CA, 1999. ACM Press.

[93] N. Ramakrishnan. PIPE: Web Personalization By Partial Evaluation. *IEEE Internet Computing*, 4(6):21–31, 2000.

[94] S. R. Ranganathan. *A Descriptive Account of Colon Classification.* Bangalore: Sarada Ranganathan Endowment for Library Science, 1965.

[95] 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.

[96] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

[97] Stephen E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33:294–304, 1977.

[98] Mary Beth Rosson. Integrating development of task and object models. *Communications of the ACM*, 42(1):49–56, January 1999.

[99] Mary Beth Rosson and John M. Carroll. Object-oriented design from user scenarios. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, pages 342–343, 1996.

[100] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[101] Gerard Salton and Micheal E. Lesk. The SMART automatic document retrieval system—an illustration. *Communications of the ACM*, 8(6):391–398, 1965.

[102] 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.

[103] T. Saracevic and P. B. Kantor. Studying the value of library and information services. Part II. Methodology and Taxonomy. *Journal of the American Society for Information Science*, 48(6):543–563, 1997.

[104] 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.

[105] M. F. Schwartz and D. C. M. Wood. Discovering shared interests using graph analysis. *Communications of the ACM*, 36(8):78, Aug. 1993.

[106] Douglas E. Shackelford, John B. Smith, and F. Donelson Smith. The architecture and implementation of a distributed hypermedia storage system. In *Proceedings of the 5th Conference on Hypertext*, pages 1–13, New York, NY, USA, November 1993. ACM Press.

[107] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[108] Edleno Silva de Moura, Gonzalo Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems*, 18(2):113–139, April 2000.

[109] M. Singhal and N. Shivaratri. *Advanced Concepts in Operating Systems: Distributed, Database, and Multiprocessor Operating Systems.* McGraw-Hill, New York, 1994.

[110] J. Spivey. *Introducing Z: A Specification Language and its Formal Semantics.* Cambridge University Press, 1988.

[111] W3C Standards. *Resource Description Framework (RDF) Model and Syntax Specification*, 1998. Available at http://www.w3.org/TR/WD-rdf-syntax/.

[112] XML topic maps. http://www.topicmaps.org/xtm/#ref_xtm.

[113] A. Sutcliffe. A technique combination approach to requirements engineering. In *Proceedings of the Third International Symposium on Requirements Engineering*, pages 65–77, Annapolis, 1997. IEEE.

[114] A. G. Sutcliffe, N. A. M. Maiden, S. Minocha, and D. Manuel. Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering*, 24(12):1072–1088, December 1998.

[115] J. Suzuki and Y. Yamamoto. Making UML models interoperable with UXF. *Lecture Notes in Computer Science*, 1618:78–87, 1999.

[116] J. Tague, A. Salminen, and C. McClellan. Complete formal model for information systems. In *Proceedings of the fourteenth annual international ACM/SIGIR conference on Research and development in information retrieval*, pages 14–20, Chicago, IL, USA, October 13-16, 1991.

[117] DOM team. Document object model (DOM) homepage. http://www.w3.org/DOM/.

[118] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Trans. on Inf. Sys.*, 9(3):187, 1991.

[119] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems. Volume I: Classical Database Systems.* Computer Science Press, 1988.

[120] 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.

[121] W3C. Resource Description Framework (RDF) Schema Specification 1.0. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, 2000.

[122] W3C. XML Schema Part 0: Primer, W3C Working Draft. http://www.w3.org/TR/xmlschema-0, 2000.

[123] W3C. Mathematical Markup Language (MathML). http://www.w3.org/Math/, 2001.

[124] W3C. XML Query. http://www.w3.org/XML/Query, 2001.

[125] Bing Wang. A hybrid system approach for supporting digital libraries. *International Journal on Digital Libraries*, 2(2-3):91–110, 1999.

[126] W. Wang and R. Rada. Structured hypertext with domain semantics. *ACM Transactions on Information Systems*, 16(4):372–412, October 1998.

[127] Stuart Weibel. The Dublin Core Metadata Initiative. http://purl.oclc.org/dc/, 1998.

[128] Gio Wiederhold. Digital libraries, value, and productivity. *Communications of the ACM*, 38(4):85–96, April 1995.

[129] R. Wilkison and M. Fuller. Integration of information retrieval and hypertext via structure. In *Information Retrieval and Hypertext*, pages 257–271, Boston, USA, 1996. Kluwer Academic Publishers.

[130] Glynn Winskel. *The Formal Semantics of Programming Languages: An Introduction.* Foundations of Computing series. MIT Press, Cambridge, MA, USA, February 1993.

[131] Ian H. Witten, David Bainbridge, and Stefan Boddie. Greenstone: Open-source DL software. *Communications of the ACM*, 44(5):47, 2001.

[132] Ian H. Witten, Rodger J. McNab, Stefan J. Boddie, and David Bainbridge. Greenstone: A comprehensive open-source digital library software system. In *Proceedings of the Fifth ACM International Conference on Digital Libraries (ACM DL'2000)*, pages 113–121, San Antonio, TX, June 2-7, 2000.

[133] S. K. M. Wong and Y. Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68, January 1995.

[134] Tak W. Yan and Hector Garcia-Molina. The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4):529–565, 1999.

[135] Lee L. Zia. The NSF National Science, Mathematics, Enginnering, and Technology Education Digital Library Program. *Communications of the ACM*, 44(5):83, May 2001.

[136] Nivio Ziviani, Edleno Silva de Moura, Gonzalo Navarro, and Ricardo Baeza-Yates. Compression: A key for next-generation text retrieval systems. *IEEE Computer*, 33(11):37–44, November 2000.

[137] M. Zubair, K. Maly, I. Ameerally, and M. Nelson. Dynamic construction of federated digital libraries. In *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, May 15-19, 2000.