Sets and Functions: A Foundation for Digital Libraries, through Streams, Spaces, Structures, and Scenarios (S^4)

Edward A. Fox, Neill A. Kipp, and Paul Mather
Virginia Polytechnic Institute and State University

Digital Libraries in particular, and Information Systems in general, engender no consensus regarding theoretical foundations. In the trivial case, all information systems can be modeled mathematically as sets of information paired with the functions that operate on that information. Using the abstractions Streams, Spaces, Structures, and Scenarios (S^4), we bridge the stark gap between the mathematical definition of a digital library and its pragmatic implementation by presenting a framework rich enough to encompass the wide variety of hypermedia information systems. We show how these abstractions may be implemented in HyTime and give examples related to the National Digital Library of Theses and Dissertations. We conclude by outlining an approach to open Digital Libraries based on HyTime constructs.

Categories and Subject Descriptors: H.3 [**Information Systems**]: Information Storage and Retrieval; I.7.2 [**Computing Methodologies**]: Text Processing—*Document Preparation*

General Terms: Theory, Design, Languages

Additional Key Words and Phrases: Digital libraries, Open Systems, International Standards

1. MOTIVATION

In *Libraries of the Future*, written 32 years ago, J.C.R. Licklider, Director of MIT's Project MAC (sometimes explained as "man and computer"), elucidated a research agenda for what we now call the field of digital libraries [Licklider 1965]. As we take up that agenda, and hope for an elegant solution, it is worthwhile to consider the theoretical challenge he provided [Licklider 1965, p. 78]:

> The most promising approach, it seems to us, is to accept the notion that, for many years at least, we shall not achieve a complete integration of knowledge, that we shall have to content ourselves with diverse partial models of the universe. It may not be elegant to base some of the models in geometry, some in logic, and others in natural language, but that may be the most practicable solution.

Perhaps 32 years counts as 'many years'? Perhaps there is a simple underlying foundation we can adopt, from which the more familiar concepts of Streams, Spaces, Structures, and Scenarios (S^4) can be derived—and we can model digital libraries (DLs) elegantly with those concepts? Perhaps HyTime (ISO/IEC 10744) [International Organization for Standardization 1992b] provides a rich enough descriptive capability to express our vision of DLs?

Our premise is that the minimalist foundation for DLs, and indeed for any information system, is **sets** and **functions**. Licklider dealt with information, storage, organization and retrieval using sets, subsets, spaces, functions, relations, predicate calculus, and higher-order languages [Licklider 1965, Ch. 3]. Operational

Name: Edward A. Fox, Neill A. Kipp, and Paul Mather
Address: e-mail: {fox,nkipp,paul}@cs.vt.edu

systems do the same today. We can improve the situation, plus handle multimedia and hypertext, by using sets and functions to define formally Streams, Spaces, Structures, and Scenarios to provide an easier to understand ("higher level") description of DLs. Software engineers and HCI researchers particularly like **scenarios**. Modern DL researchers have used scenarios to explain DL operations from a variety of perspectives [Gladney et al. 1994; Lynch and Garcia-Molina 1995]; Licklider provided an extensive one [Licklider 1965, p. 45–58].

Licklider discussed 'knowledge workers' who operate in the real world and can contribute directly to DLs, by starting with 'primary nature,' acquiring knowledge, and adding that to the library [Licklider 1965, p. 22]. Clearly their efforts must deal with 4D and other 'spaces,' 'streams' of knowledge and library information, and 'functions' such as transforms between spaces, indexing, and acquisition. Streams are a key representation scheme for multimedia, and 'structures' are crucial for organizing knowledge, so both concepts can be used directly in applications [Licklider 1965, p. 26].

We claim that we can use the concepts of Streams, Spaces, Structures, and Scenarios to describe key aspects of DLs (and related fields). In the sections that follow we validate our claim, directly, and by example, as we elaborate a framework for next-generation, open, integrated DLs.

1.1 NDLTD: A Digital Library with Users

DLs are nothing without users. The National Digital Library of Theses and Dissertations (NDLTD) [Fox et al. 1996] is an effort to collect and integrate Electronic Theses and Dissertations (ETDs); the students that produce them; the faculty committees that approve them; the graduate schools and university libraries that accept and catalog them; the librarians that maintain them; the scholars and browsers that search, study and annotate ETDs; and the personal/shared workspaces of all of these users. Only users will create the NDLTD, fill it with new titles, strew it with hyperlinks, and manage its contents through time. Only users will meet in the virtual stacks to talk about interesting additions, patterns of access, distribution of ETDs, and the library itself.

Throughout this paper, we use the NDLTD as our example DL, and we see how sets, functions, streams, spaces, structures, and scenarios will denote its formal and pragmatic design. Finally, we see how HyTime will be used to encode a DL as rich as the NDLTD for open international use.

2. MINIMIZE AND RATIONALIZE THE DOMAIN

Our premise is that a DL is a **set** acted upon by **functions**.

2.1 Sets and Functions

Sets are collections of unique items. Items in a set are inert; items do not enter or leave a set until some external force (a function) acts to insert or remove them. Without functions, therefore, sets cannot *do* anything.

Functions are the operations that may create and destroy sets or insert and remove items from sets. Functions may apply operations on the items in a set. Relationships between sets may be defined as functions whose range is the set $\{true, false\}$; either the relationship is present, or it is not. Functions may generate items, sets of items, or sets of sets. But clearly, without information to manipulate or generate, functions are nothing.

2.2 Sets and Functions in the DL

Publication routines that lead to DLs are but one example of how functions—meant formally as a key concept in our theoretical framework, but analogous to transforms, processes, filters, programs, and procedures which will be used almost interchangeably where the subtle variations are clear—fit into the world of DLs.

In information retrieval, indexing is a key function. Matching is another, but has even more general applicability in DLs, where one might hum a melody to find the right tune in a collection of MIDI files [Ghias et al. 1995] or try to detect plagiarism [Shivakumar and Garcia-Molina 1995]. Matching may combine various types of information [Fox 1983] or integrate results from various sources [Belkin et al. 1995].

Functions that transform information into information are at the heart of coding, decoding, compressing, decompressing, importing, exporting, rendering, displaying, and various types of information analysis (of text, speech, image, video). In Project Athena, there are 11 editors to handle transforms of various media forms and actions, as well as the high-level application constructor, MuseBuilder [Hodges and Sasnett 1990, p. 166].

Functions that compute and generate information have particular importance in DLs. Active documents [Zellweger 1992], which combine processes with more traditional document components, are likely to become more common. As interfaces become more supportive of human tasks, information visualizations and rich interactions supported by extensive processing [Rao et al. 1995] will become the norm.

Users affect the DL by means of functions that act upon the information in the library in some known fashion. These functions may be persistent, opportunistic, cooperative, and goal-based. Several functions may be involved in satisfying a user's information need within a DL; these will typically utilize the products of other functions inhabiting the DL. For example, the function that adds ETDs to the NDLTD will check the authenticity of a new submission with the authentication function, store the new ETD with the archiving function, meanwhile passing the ETD's metadata to library cataloging functions.

A navigation function could read the user's profile and allow collection browsing according to a preferred view. The navigation function could solicit the services of a recommender function on the user's behalf to guide browsing. It might utilize the user's past browsing history, or those of other users whose preferences are judged similar, in forming the structure the user sees of the library. The navigation function also might call upon the services of a query function which, in turn, would utilize the previous output of an indexer.

Each of these functions might come in several 'flavors,' chosen by the user, as appropriate to the task, economic need, or resource availability. Service functions will plan the efficient execution of a given user information need by invoking the relevant function. Distributed objects might provide an enabling technology to realize such process and brokerage services (as is done by specialized agents in the University of Michigan DL [Birmingham 1995]). The Knowledge Query and Manipulation Language (KQML) [Mayfield et al. 1996] may additionally be integrated to effect cooperation between functions.

2.3 Basis in OO world, but not too OO

Indeed, with its {*set, functions*} tuple definition, the DL fits the definition of an object-oriented 'object.' Furthermore, because we define all digital objects in the collection to conform to some structure, and have classes conform to one or more metastructures, and so on, that an object-oriented implementation of the DL is a natural step is true at an abstract level. In the DL, on the other hand, certain functionality must protect the data, particularly the functions that implement access restrictions based on intellectual property rights, authenticity validation, and other security issues. This dualism is preserved by keeping the {*set, functions*} paradigm intact.

Functions project a semantic interpretation of the information according to a context which may be provided by the user or the DL, or a combination. The inputs and outputs of each function are information, the specific nature of which is local to the form of the function. For example, an indexer function for the NDLTD might take a stream of incoming ETDs as its input and, along with extant library items, update its index. In addition, certain functions may not have permission to access certain data; an access manager would intercede.

2.4 Further Reductions

Indeed, a DL could be simply {*information*}, with the semantic difference being provided through the information's metadata. Kolmogorov complexity describes the close parallels between information and functions, and that all functions are inherently information [Li and Vitányi 1994]. It is feasible that the information content of a DL includes functions as items in the collection, e.g., a random number generator that samples radioactive decay could be an *item* in the DL collection.

Alternatively, a DL could be simply {*functions*}, with the semantic difference being that information is useless unless it is active, i.e., unless some function delivers it. Every piece of information could be seen as the result of calling a function—even the psychological paradigm of thinking is entirely functional. We feel, in summary, that the distinction between information and function has value and should be preserved.

Table 1. BNF for DL

digital library	:=	information, functions
information	:=	item+
item	:=	unique identifier, metadata, digital object
metadata	:=	title, author, date, access rights, encoding, . . .
digital object	:=	document | log | relationship+ | . . . | item+
functions	:=	indexer | browser | querier | item manager | recommender | rights checker | authenticator | agent | . . .

2.5 BNF for DL

Below is a mathematical formalism for our denotational definition of a DL. We represent this with a grammar, expressed in BNF, and shown in Table 1.

—The **digital library** is information and functions that process that information.

—The **information** in the DL is a set of items.

—The **items** are unique digital objects, each with semantic interpretation seen in terms of its constituent digital object and that object's associated metadata.

—The **metadata** may include the title, author, date, any access rights the information may have, and hints on how to interpret the data's encoding. We encourage adoption of the Dublin Core where suitable, and extensions where their correctness and interpretation are assured [Weibel 1995]. Clues in the metadata will tell the functions about the digital object, as will attributes inherent in the object. Furthermore, clues in the underlying representation of the digital object will give information to functions as well (e.g., a filename suffix of '.jpg' tells us the file may contain a JPEG-encoded image).

—The **digital object** could be a report, dissertation, annotation, user profile, graphic, animation, video, etc. A digital object could be any single collectible unit, or it could be a group of items, therefore making the grammar recursive.

—The **functions** listed in the grammar only hint at the complete list. The indexer function could index text, video, audio, or other media. The browser function may browse anything from ASCII to virtual sensoria. Other functions may include billing subsystems, document storage subsystems, network directory servers, security servers, query splitters and recombiners, search engines, filter services, link managers, previewers and thumbnailers, renderers, data analyzers, navigators, authoring servers, editing servers, input/output servers, indexing tools, and agents [Gladney et al. 1994].

—The **relationship** tells what the relationship is, why it exists, and what the roles of each of the objects may be. It also tells what implicit functions may be applied. There may be one, many, or arbitrary numbers of objects of the relationship, but a relationship must have at least one. The functions tell what explicit operations occur as the relationship endures through time and via observation. For example, a simple 'bibliographic reference' is the name of one authorable relationship (see Figure 1). Traditionally this relationship has a direction and two objects, from point of reference in the referring text to the target of reference that occurs in a bibliography or bibliographic database. One may build complex webs of reference using this simple construct. Furthermore, this relationship may have attributes, such as who made the reference and why.

3. STREAMS, SPACES, STRUCTURES, AND SCENARIOS

Because the above formalism is very broad and does not use the terminology of system designers, ontological bridges must be built before practical applications may be implemented. In information retrieval systems, data structures like B-trees, feature spaces, hash tables, sets, strings, tries, and vector spaces are common [Frakes and Baeza-Yates 1992, Ch. 2]. For hypertext, the focus is on graphs—which also can provide a knowledge representation foundation for artificial intelligence. Logics can be used in AI, in information retrieval, and in logic databases—though relations are more commonly chosen as the basis for database management systems. Among the most comprehensive models in the information area are those attempting to integrate information retrieval and hypertext [Chiaramella and Kheirbek 1996; Lucarella and Zanzi 1996;

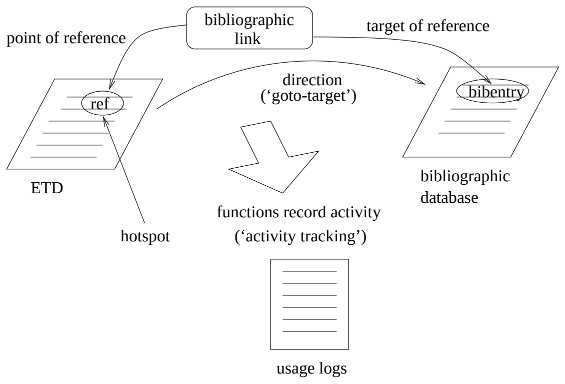

Fig. 1. Functional linking

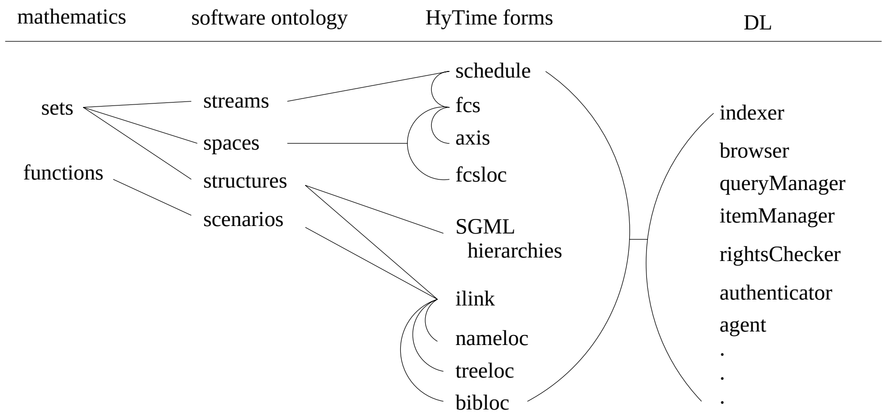

Fig. 2. Concept map for digital libraries

Fox et al. 1991]. While each of these papers relates to prototype systems, and provides interesting insights, they begin with powerful constructs like conceptual graphs or object-oriented databases, and fail to address the breadth of requirements for DLs [Gladney et al. 1994; Licklider 1965, p. 36-39].

In the multimedia information system arena today, the problem is more clearly visible. Books in that area typically devote a chapter to each sub-area, with different formalisms, approaches, and technical concerns [Khoshafian and Baker 1996]. At best, a common framework at the level of object-oriented methods is recommended [Gibbs and Tsichritzis 1995]. Yet, with decades of experience, Englebart reminds us that open systems and knowledge domain interoperability is the real goal [Engelbart 1991]. Since we have such difficulty with interoperability of complex representations, it is worth considering interoperability founded at lower levels (see first two columns of Figure 2).

3.1 Streams

Multimedia researchers will agree that streams are of fundamental importance in their work. Through the general process of capture, and the specific one of digitization, a digital information stream can be produced to correspond to real-world video or audio. Thus, from the analog world, where acoustic or EMF waves are continuously generated, we transform to the digital domain. To simplify our description, we assume that streams are sets of finite sequences from some alphabet.

Images can be represented as arrays, and mapped to sequences of pixel descriptions in row or column major order, for example. Text, as handled in a word processor, is a sequence of characters (string), and documents or books or collections of them can be similarly represented by following some ordering (e.g., by Library of Congress call number, then by page, then by line). While the PAT model [Frakes and Baeza-Yates 1992, Ch. 5] calls for semi-infinite strings, very long finite sequences really suffice.

Streams often are labeled by media type or semantic type. Thus, video-on-demand is representative of one type of application, while another is exemplified by filtering and routing of streams of news, messages, and publications. Networked information research considers logs of packets or their headers, and HCI considers streams of keystrokes or other human/system actions. Failure analysis of retrieval systems sometimes deals with streams of queries. Decades ago, Shannon focused on information flow as bit sequences over channels, while recently Devlin began to develop a theory of information by considering context and situation (which are determined by proximity in a stream) as well as strings (especially logical expressions) [Devlin 1991].

3.2 Spaces

Licklider (at MIT) briefly discussed spaces for information [Licklider 1965, p. 62] at the same time that Salton and Lesk (at Harvard) were formulating the theory of vector spaces and first implementing it in the SMART system. Those spaces can be 'cleaned up' mathematically [Raghavan and Wong 1986], or reduced algorithmically into spaces with fewer dimensions, as in Koll's early Weird system or with LSI [Deerwester et al. 1990]. They can be generalized into feature spaces, suitable for clustering or probabilistic retrieval.

Humans, working with knowledge and concepts, have evolved various structures, including concept spaces. Thus, a simple controlled vocabulary (or an enumeration of word senses) can specify a concept space analogous to a vector space. Various spaces (or subspaces as in [Fox 1983]) can handle metadata like author and date, or relationships like citation-based links.

Multimedia systems must represent real as well as synthetic spaces: 1D, 2D, 3D, or 4D. Many of the synthetic spaces represented in virtual reality systems are analogs to real spaces, or to information spaces of various types.

Consider this scenario: the NDLTD is a virtual space, modeling the usual university library, where the items in the library's collection are arranged by call number (fielded unique identifiers, where fields in the identifier correspond to subjects, perhaps authors' names and years, and finally serial numbers within these categories). In this 4-space, users appear as trails through the stacks, traveling to item after item and lingering over those they find intriguing. Users may pop in and out of the stacks as they search the collection via queries, and have the option of viewing other characteristics of the library, particularly the other users that are present. Perhaps they may strike up a whispered conversation, share recommendations, or even arrange a meeting in a library cooperative-work space.

Librarians (agents or human-representations) are also users: they are present as items in the DL data set, and they are present in both 4-space and its transform into the virtual environment.

3.3 Structures

Structures mark and measure streams. Thus, we describe information structures by imposing a grammar onto a stream. Exploiting the well-known dualism of formal languages and automata theory, we see that information structures can be described by machines that recognize and generate them. Therefore, we see that functions (machines) can generate information structures from streams.

Structures are critical to DLs [Furuta 1994]. Traditional structures exist, such as those based on layout, like pages of books, as well as logical structures, often based on hierarchies [Wilkinson and Fuller 1996, p. 258]. Thus, articles are structural parts of journals, proceedings, or encyclopedia. Similarly, poems have stanzas, symphonies have movements, and a researcher working with an information system works during sessions that contain queries. Our framework allows an arbitrary (finite) number of structures to be associated with a stream. Maybe this will help reduce use of the misnomer 'unstructured information'; clearly it enables handling of structured documents [Andre et al. 1989] as commonly studied in the context of electronic publishing.

As a result of the dualism between grammar and machine, we see that structures can be static or dynamic. Thus, one method of video segmentation may lead to different structuring of a real-time video stream than would another method (function). Indeed, video can be viewed either as a stream or as a structure that can generate a stream [Ghandeharizadeh 1996], and the most powerful compression methods are model-based (i.e., expressing a large stream in a small structure interpretable according to some model). CGI scripts and database systems can generate varying numbers of WWW pages in response to a particular request, or a pre-stored page can be delivered.

In our framework, one can have multiple structures, each of which describes some aspects of a stream, and these may even vary from person to person, working with the stream. Structures can be composed, such as when one low level structure is the basis for a number of higher level structures. This approach solves some basic problems faced in information retrieval. In their attempt to model the typical results of electronic publishing, the vector space and probabilistic models consider the notion of 'document.' However, it is unclear what a 'document' should be in an encyclopedia, or inside a book, or inside a transcription of the human dialog captured over the lifetime of a project, or from the multichannel data of a spacecraft over years of a mission to the outer planets? With multiple structures, multiple interpretations of 'document,' or retrieval into whatever structure most closely surrounds a match, are supported.

A stream or space may contain a structure, or be organized by the application of an external one. Thus, MARC records as well as SGML or HTML documents have internal markup that describes logical structuring. Multiple structures can be overlaid using the HyTime hyperlinking facility, and structures can be manipulated using primitives in the Document Style Semantics and Specification Language (DSSSL). Multiple spaces can be managed by the HyTime finite coordinate space (fcs) facility. Other streams, like those coming from scanners or cameras, may break down into images (in raster order, after being split into frames) according to simple algorithmic (simple language) processing. Finding the structure may be arbitrarily difficult, such as analyzing a color image (stream or 2D-space) to extract the person, cat, and house depicted.

Pointers into a stream, space, or structure can mark the extent of some part of a structure, much like tags can delineate extent in a stream. When a stream is read-only, or too large to copy, pointers may be necessary since tags cannot work. The Hyper-G/Hyperwave system can handle 'hot-links' from an anchor whose extent in text or images is specified externally from the stream. Clearly, internal versus external location of structuring are equivalent and interchangeable, though for performance or other reasons (e.g., packaging, security) one may be preferable (e.g., HTML pages can be shipped and carry anchor and link information, while Hyper-G anchor details must be accessed through a distributed object-oriented data to get the same result; on the other hand, Hyper-G allows links to be followed in both directions). Modern views of documents consider their structure in terms of streams, hierarchies, and composites as an extension of the traditional view of internal content structure plus links [DeRose and Durand 1994, p. 21-22].

3.4 Scenarios

Scenarios can be used as part of the process of designing information systems [Carroll 1995]. Humans can quickly grasp the potentials and complexities of DLs through scenarios. Human information needs, and the processes of satisfying them in the context of DLs, are well suited to description with scenarios, including these key types [Wilkinson and Fuller 1996, p. 260]: fact-finding, learning, gathering, and exploring. Additionally, scenarios can aid understanding of how DLs affect organizations and society, and how challenges to support social needs relate to underlying assumptions of DLs [Levy and Marshall 1995]. Scenarios can help us consider the complexities of current publishing methods, as well as how they may be reshaped in the era of DLs, considering publishing paths, associated participants, and publication functions [Wiederhold 1995].

Scenarios tell what happens to the streams, in the spaces, and through the structures. Scenarios let us visualize the spaces, by setting up a stream from views of the structure. To the digital library, scenarios supply the verbs, where streams, spaces, and structures supply the subjects and objects. For this reason, we use scenarios to define the functions of a digital library.

4. MAPPING OF BNF TO SGML/HYTIME DEFINITION

We feel that HyTime offers an excellent computer science formalism upon which to build a DL. SGML, the parent of HyTime, is the language for document structuring, and it even defines a structuring mechanism—Document Type Definition (DTD)—for representing a document in terms of its BNF. SGML, the Standard Generalized Markup Language [International Organization for Standardization 1986a], is useful for adding structure to streams of data. HyTime, the Hypermedia/Time-based Structuring Language, is largely encoded in SGML, and represents relationships between information; it is useful for encoding connection between structures, for binding functions to information, and for declaring metastructures of information that contain streams and stream connections.

HyTime standardizes hundreds of concepts and utilities found in the widest variety of hypermedia encodings. It incorporates the Dexter model [Leggett and Schnase 1994] for static document encoding and the Amsterdam model [Hardman et al. 1994] for linking to and from events in schedules (like frames in a video or animation). Documents encoded in SGML/HyTime are transportable across networks and through time. For these reasons, HyTime documents can survive technological innovation and vendor-specific encodings.

Below we present the mapping from the abstract DL concepts in Table 1 to the applicable HyTime constructs. Additionally, we detail the application of the HyTime architecture to the DL problem and indicate a path for its implementation. (Readers unfamiliar with SGML and HyTime are encouraged to read overviews [Goldfarb 1991; Newcomb et al. 1991].)

4.1 'Digital Library' maps to HyTime Grove

The HyTime 'grove' construct is a set of SGML entities (SGML documents and other data entities named by SGML documents). The grove is defined as a set of entity declarations in the root document of a HyTime system. Groves are useful for collecting heterogeneous document sets into a single, navigable structure. All information and functions will be encoded as data entities and documents within the HyTime grove for the digital library.

4.2 'Information' maps to HyTime Architectural Forms and Documents

Information in a DL is a set of items. Each item is a relationship between a unique identifier, metadata, and a digital object. This connection is neatly encoded using a HyTime **independent link** (**ilink**) architectural form with a unique identifier and two 'linkends,' one to refer to the metadata and one to refer to the associated digital object. Indeed, all the structure in the DL can be represented using HyTime ilinks.

4.3 'Unique Identifier' maps to HyTime Item Identification

For any library to function, all information present must be unique and identifiable. Therefore the HyTime implementation of the DL will employ unique identification of items. The unique identifier will often contain relative naming from an agreed universal starting point, then repeatedly qualify until the named object is reached. The final qualifier of the 'item' is its SGML unique identifier (ID). Uniqueness may be guaranteed trivially, by consulting the index of unique names ($O(\log n)$ in the worst case), and generating new identifiers through a serializing protocol.

Using unique identifiers, automated processes may locate the information in the DL. Furthermore, users may use the names associated with the different information objects in the collection as anchors of their own hyperlinks, e.g., for annotating other works in the NDLTD, and making bibliographic references to other theses and dissertations in the collection.

In short, HyTime conceptually solves the naming problem that plagues digital library design.

4.4 'Substructures of Digital Objects' maps to HyTime Location Addressing

Suppose an ETD author wishes to refer to a particular paragraph in another thesis or dissertation. The target is not an entire item in the collection, instead it is part of one. Further, suppose that it does not have a unique ID. Substructures of items within the DL may be identified using HyTime architectural forms (see Table 2). For example, each of the location address forms may be used successively as a 'location ladder' to qualify real and logical space. Location address ladders may contain queries to extract information from

Table 2. HyTime location address architectural forms

Data Type	HyTime Construct	Abbreviation
Object in SGML hierarchy	tree location address	treeloc
Object positioned on coordinate axes (diorama or animation)	finite coordinate space address	fcsloc
Named external digital object	named location address	nameloc
Named or positioned external non-digital object	bibliographic location address	bibloc

a domain. Also, a location address may refer to an aggregate of disparate, heterogeneous objects. This is useful when one is resolving a query for all the ETDs available from a certain school or within a given major discipline.

4.5 'Relationships' maps to HyTime Ilinks

A relationship comprises $\{$*name, objects, functions*$\}$. The relationship name corresponds to the 'generic identifier' of the ilink. The name of each object as it relates to the ilink (like point of reference, target document) become HyTime 'anchor roles' of the ilink. The locations of the objects, in the order they appear, become the HyTime 'linkends.' Furthermore, the existence of a relationship implies functionality. The system will present a 'hotspot' from the referring object's location and a 'goto-target' functionality when the user selects the hotspot. Other functions may be called at activation of this relationship, record the traversal for log-keeping, billing, or load evaluation.

To bind function calls to relationships, we use HyTime 'activity tracking.' Bound functions are called for each activation of the ilink. In the NDLTD, authors of referenced works could be notified when their ETD is the target of a bibliographic link. Because access rights are in the metadata record stored with the digital object, and all are hyperlinked by the 'item' ilink, then access rights are always available for each digital object. Functions that provide data to users of the DL will consult the linked access rights to determine permission and to invoke a billing function as necessary.

4.6 'Non-traditional Items' maps to HyTime Elements

DLs contain more that just traditional collection items. The data structures that are generated by indexing functions are items in the DL. The index itself is a ilink that connects a multitude of index entry tuples. Each index entry is also an ilink that connects the term and all its occurrences in the collection. Additionally, the user is an item in the DL. The user's profile will have an SGML encoding and be available as the anchors of any sort of hyperlink that the DL supports. Further, the user will have a position in the stream of time-based usage logs. Because of the consistent, well-understood structure of each possible constituent in the DL, and the standardized encoding thereof, the system is provably reducible to a simple HyTime encoding. We leave for future work the grammatical description of each ilink type and each defined interconnection, as well as the syntactic HyTime and SGML instantiation of each.

4.7 'Streams' maps to HyTime Finite Coordinate Space (fcs)

Using HyTime, one may construct one-dimensional fcs data structures and fill them with events and groups of events. Each event contains an object, be it an item in the collection, a timepiece, calendar, a user, a librarian, or an agent of either.

The flow of the objects through time is measured on the time axis, the primary axis of the finite coordinate space. Objects in a collection flow from shelf to user to shelf, or to multiple users at once. Objects in the DL may undergo revision and versioning, and the fcs is an excellent device to track changes in the object as time passes.

4.8 'Spaces' maps to HyTime fcs

The HyTime fcs encoding scheme builds and stores the virtual spaces required by a fully-functional DL. An fcs may contain an arbitrary number of axes, and therefore allow events to be scheduled with position and extent along each axis, in space and in time. Entire virtual worlds may be created using fcss based on the contents and hyperlinks of a digital library. Users may manipulate objects in time, perhaps leaving annotations for one another in the stacks. HyTime also has facilities to project fcss into other fcss, perhaps even flattening an entire spatial dimension. In this way, all of a user's movements can be observed as a stream through the stacks.

4.9 'Scenarios' map to HyTime fcs and ilink

System administrators for the DL require different views of the DL. They can use fcs models to create a **visualization** of the DL state, that, for example, describes caching and flow of materials from one physical site to another. These fcss could show the physical arrangement of servers and can show the connections between them. As traffic increases, the arcs between the depicted servers could change color or the diameter of the arc could increase. If one site becomes thick with traffic, the administrator could arrange for objects to be redistributed to a site nearer the user's physical position. After viewing such an fcs, users may request that a set of desired digital objects be replicated to decrease latency over long stretches of crowded network.

Visualization is vital for all users of the DL. With such a vast and heterogeneous collection available, users navigating through the collection of ilinks will easily become 'lost in hyperspace.' Visualizing where they are in the hyperlinking structure can help them visualize relevance, clustering, and associated materials, and therefore allow them to find more accurate information faster. All of this is possible and natural with HyTime's ilink and fcs constructs.

4.10 HyTime Engine

The bookkeeping for hyperlinking and scheduling requires a HyTime Engine: a software toolkit that has an extensive API for manipulating HyTime constructs within an (distributed) object database.

Various DL functions, like indexing, user management, etc., are written using the HyTime Engine API calls plus calls to other DL functions. These routines run at various times as they are needed (lazy evaluation) by users or intrinsic DL processes. Each of these applications is intended to be lightweight and singular of purpose so that monolithic construction of the DL can be avoided. The management of the HyTime layer is well-defined so that designers avoid exponential numbers of changes of the application layer.

This framework for specification enables a multitude of software implementations (DL products), e.g.,

—a single-site system with one running DL program (e.g., to serve a private collection to a single user),

—a distributed database system with many servers performing data-lookups for client-side DL functions,

—a distributed DL with peer-level interaction between sites as well as a cross-distributed data layer,

—an object-oriented substrate using any of the popular object databases or distributed object databases,

—bus-based (as at Stanford [Stanford Digital Library Group 1995]), and

—agent-based with users and information robots [Birmingham 1995].

5. CONCLUSION: THE DIGITAL LIBRARY APPLICATION

The separation of definitions (S^4) within the information and functions of a DL simplifies its implementation. Further, the use of HyTime to structure the streams of data and perhaps to impose a spatial ordering on it makes the set of information in the DL manageable. HyTime adds refinement to the ontology required for a robust DL implementation. That leads to using HyTime engines to manage the transport protocol between the various heterogeneous sites in the NDLTD (see Figure 3). Each federated system interacts with the rest of the NDLTD through a conversation between HyTime engines.

We plan to continue work on our framework in the context of NDLTD. Our approach will be to collect requirements from libraries and graduate schools interested in the project, to translate those into S^4 and thence into HyTime, and to specify an API/protocol for NDLTD as an open digital library.

We have provided scenarios for the NDLTD. One can easily see that domain and range of each function is the information within the digital library combined with the information that travels to and from each human user. Having detailed the data model, we leave the exact specification of the set of scenarios, spaces, and functions to the architects and users of a streamed and structured digital library.

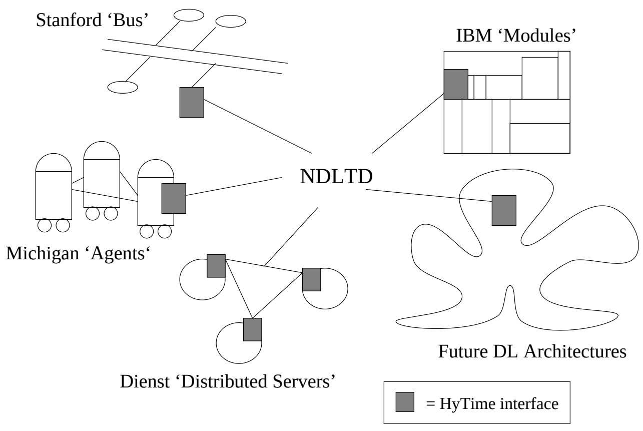

Fig. 3. NDLTD though open DL integration

6. ACKNOWLEDGMENTS

Special thanks go to J.C.R. Licklider and Gerard Salton for training and guidance. Layne Watson guided the early development of the basic formalism. The National Science Foundation, through grants CDA-9312611 and IRI-9116991, as well as SURA and U.S. Dept. of Education, FIPSE Program (for their support of our thesis and dissertation project), provided support.

REFERENCES

Andre, J., Furuta, R., and Quint, V. Eds. 1989. *Structured Documents*. Cambridge University Press, Cambridge.

Belkin, N. J., Kantor, P., Fox, E. A., and Shaw, J. A. 1995. Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management 31*, 3 (May-June), 431–448.

Birmingham, W. P. 1995. An agent-based architecture for digital libraries. *D-Lib Magazine*.

Carroll, J. M. 1995. *Scenario-Based Design: Envisioning work and technology in system design*. John Wiley, New York.

Chiaramella, Y. and Kheirbek, A. 1996. An integrated model for hypermedia and information retrieval. In M. Agosti and A. Smeaton Eds., *Information Retrieval and Hypertext*, pp. 139–178. Boston: Kluwer Academic Publishers.

Deerwester, S., Dumais, S., Furnas, T., Landauer, T., and Harshman, R. 1990. Indexing by latent semantic analysis. *J. of the ASIS 41*, 6, 391–407.

DeRose, S. J. and Durand, D. G. 1994. *Making Hypermedia Work: A User's Guide to HyTime*. Kluwer Academic Publishers, Boston.

Devlin, K. 1991. *Logic and Information*. Cambridge University Press, Cambridge.

Engelbart, D. C. 1991. Knowledge-domain interoperability and an open hyperdocument system. In E. Berk and J. Devlin Eds., *Hypertext/Hypermedia Handbook*, pp. 397–413. New York: McGraw-Hill, Inc.

Fox, E. A. 1983. *Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types*. Ph. D. thesis, Cornell University Dept. of Computer Science. Available from University Microfilms Int.

Fox, E. A., Chen, Q. F., and France, R. K. 1991. Integrating search and retrieval with hypertext. In E. Berk and J. Devlin Eds., *Hypertext/Hypermedia Handbook*, pp. 329–355. New York: McGraw-Hill, Inc.

Fox, E. A., Eaton, J. L., McMillan, G., Kipp, N. A., Weiss, L., Arce, E., and Guyer, S. 1996. National Digital Library of Theses and Dissertations: A scalable and sustainable approach to unlock university resources. *D-Lib Magazine*.

Frakes, W. and Baeza-Yates, R. Eds. 1992. *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, Englewood Cliffs, NJ.

Furuta, R. 1994. Defining and using structure in digital documents. In J. L. Schnase, J. J. Leggett, R. K. Furuta, and T. Metcalfe Eds., *Proceedings of Digital Libraries '94: The First Annual Conference on the Theory and Practice of Digital Libraries* (College Station, TX, June 19-21, 1994), pp. 139–145. Texas A&M University.

Ghandeharizadeh, S. 1996. Stream-based versus structured video objects: Issues, solutions, and challenges. In V. S. Subrahmanian and S. Jajodia Eds., *Multimedia Database Systems: Issues and Research Directions*, pp. 215–236.

Berlin: Springer-Verlag.

Ghias, A., Logan, J., Chamberlin, D., and Smith, B. C. 1995. Multimedia documents with elastic time. In P. Zellweger Ed., *Proceedings ACM Multimedia '95: The Third ACM International Multimedia Conference and Exhibition* (New York, November 5-9, 1995), pp. 143–154. ACM Press.

Gibbs, S. J. and Tsichritzis, D. C. 1995. *Multimedia Programming: Objects, Environments and Frameworks.* ACM Press Books, ACM Press in NY and Addison-Wesley in Reading MA.

Gladney, H. M., Fox, E. A., Ahmed, Z., Ashany, R., Belkin, N. J., and Zemankova, M. 1994. Digital library: Gross structure and requirements: Report from a March 1994 Workshop. In J. L. Schnase, J. J. Leggett, R. K. Furuta, and T. Metcalfe Eds., *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries* (College Station, TX, June 19-21, 1994), pp. 101–107. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atg1.WUSTL.edu/DL94.

Goldfarb, C. F. 1991. *The SGML Handbook.* Oxford University Press.

Hardman, L., Bulterman, D. C. A., and van Rossum, G. 1994. The Amsterdam hypermedia model: Adding time and context to the Dexter model. *Commun. of the ACM 37*, 2 (Feb.), 50–63.

Hodges, M. E. and Sasnett, R. M. 1990. *Multimedia Computing: Case Studies from MIT Project Athena.* Addison-Wesley Publishing Company, Reading, MA.

International Organization for Standardization Ed. 1986a. *ISO 8879: Information Processing — Text and Office Information Systems — Standard Generalized Markup Language.* ISO.

International Organization for Standardization Ed. 1992b. *ISO/IEC 10744: Hypermedia / Time-based Structuring Language: HyTime.* ISO/IEC.

Khoshafian, S. and Baker, A. B. 1996. *MultiMedia and Imaging Databases.* Morgan Kaufmann, San Francisco.

Leggett, J. J. and Schnase, J. L. 1994. Viewing Dexter with open eyes. *Commun. of the ACM 37*, 2, 76–86.

Levy, D. M. and Marshall, C. C. 1995. Going digital: A look at assumptions underlying digital libraries. *Commun. of the ACM 38*, 4 (Apr.), 77–84.

Li, M. and Vitányi, P. M. B. 1994. *An Introduction to Kolmogorov Complexity and its Applications.* Addison-Wesley.

Licklider, J. C. R. 1965. *Libraries of the Future.* The MIT Press, Cambridge, MA.

Lucarella, D. and Zanzi, A. 1996. Information modelling and retrieval in hypermedia systems. In M. Agosti and A. Smeaton Eds., *Information Retrieval and Hypertext*, pp. 121–138. Boston: Kluwer Academic Publishers.

Lynch, C. and Garcia-Molina, H. Eds. 1995. *Interoperability, Scaling, and the Digital Libraries Research Agenda: A Report on the May 18-19, 1995 IITA Digital Libraries Workshop* (Palo Alto, CA, Aug. 22, 1995). Stanford Univ. http://www-diglib.stanford.edu/diglib/pub/reports/iita-dlw/main.html.

Mayfield, J., Labrou, Y., and Finin, T. 1996. Evaluation of KQML as an agent communication language. In M. Wooldridge, J. P. Miller, and M. Tambe Eds., *Intelligent Agents Volume II—Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages*, Lecture Notes in Artificial Intelligence (1996). Springer-Verlag.

Newcomb, S. R., Kipp, N. A., and Newcomb, V. T. 1991. The HyTime hypermedia/time-based document structuring language. *Commun. of the ACM 34*, 11 (Nov.), 67–83.

Raghavan, V. V. and Wong, S. K. M. 1986. A critical analysis of vector space model for information retrieval. *J. of the ASIS 37*, 5 (Sept.), 279–287.

Rao, R., Pedersen, J. O., Hearst, M. A., Mackinlay, J. D., Card, S. K., Masinter, L., Halvorsen, P.-K., and Robertson, G. G. 1995. Rich interaction in the digital library. *Commun. of the ACM 38*, 4 (Apr.), 29–39.

Shivakumar, N. and Garcia-Molina, H. 1995. SCAM: A copy detection mechanism for digital documents. In J. L. Schnase, J. J. Leggett, R. K. Furuta, and T. Metcalfe Eds., *Proceedings of Digital Libraries '95: The Second Annual Conference on the Theory and Practice of Digital Libraries, Austin, TX* (College Station, TX, June 11-13, 1995), pp. 155–163. Texas A&M University.

Stanford Digital Library Group. 1995. The Stanford digital library project. *Commun. of the ACM 38*, 4 (Apr.), 59–60.

Weibel, S. 1995. Metadata: The foundation of resource description. *D-Lib Magazine*.

Wiederhold, G. 1995. Digital libraries, value, and productivity. *Commun. of the ACM 38*, 4 (Apr.), 85–96.

Wilkinson, R. and Fuller, M. 1996. Integration of information retrieval and hypertext via structure. In M. Agosti and A. Smeaton Eds., *Information Retrieval and Hypertext*, pp. 257–271. Boston: Kluwer Academic Publishers.

Zellweger, P. T. 1992. Toward a model for active multimedia documents. In M. M. Blattner and R. B. Dannenberg Eds., *Multimedia Interface Design*, pp. 39–52. Reading, MA: ACM Press and Addison-Wesley Publishing Company.