[DigLib] Stanford Digital Library Project
Working Paper

Title: Conceptual models for comparison of digital library systems and approaches
Author: Terry Winograd

Abstract: This document is a working paper that grew out of the discussions at the Digital Libraries joint projects meeting in Washington on Nov. 8-9, 1994. It is intended as a first rough cut at a conceptual framework for understanding the significant differences among systems and ideas, so that we can better decide where to work for interoperability and where to take complementary approaches. Contents:

Conceptual models for comparison of digital library systems and approaches

This document is a working paper that grew out of the discussions at the Digital Libraries joint projects meeting in Washington on Nov. 8-9, 1994. It is intended as a first rough cut at a conceptual framework for understanding the significant differences among systems and ideas, so that we can better decide where to work for interoperability and where to take complementary approaches. This is a working draft and open to discussion and debate at all levels. I had thought about trying to do it more thoroughly before distribution, but decided not to for two reasons. First, time is important -- decisions need to be made in various aspects of the digital library project that may be determined by some of these issues. Second, by leaving it as rough as it is (especially the second half) it accurately conveys the fact that it is not a polished reference source, but a set of issues to provoke relevant discussions. The more detailed working out will come on an item-by-item basis when we have actual models to implement and want to anticipate where they will and will not interoperate.

Basic Approach

There are many different levels to be considered in achieving interoperability. Some have to do with platforms, some with specific formats and encodings, others with communications connectivity. This document deals with the structure of the underlying conceptual models on which the systems are based. Integration requires alignment of models, which often is much harder than more straightforward formatting or porting issues. It is not a substitute for those other discussions, but is complementary to them.

The distinctions for talking about models that are described here reflect decisions that must be made (explicitly or implicitly) in each design. In many cases they may not show up directly in the "user model" that end-users of the system need to understand. Often it is possible to get away with fewer distinctions for simple cases. The goal here is to fully clarify and distinguish the full range of possibilities for our understanding and for consistency. By laying bare the bones here, we are concerned with the "builder models" that distinguish different architectures and designs.

Having laid out the dimensions we need to then understand the operational consequences of model choices. In some cases a choice may determine many elements of an implementation. In others, an implementation can be neutral, allowing choice among several models when it is applied in practice. This document is a first step and does not get to the point of addressing the operational questions. Comments on those (as well as any other aspects) are welcome.

Dimensions of the model

The structure of different model dimensions here is a first cut at identifying the important differences. It is based on a to-level distinction between Data models and Action models.

For each dimension we will give the basic distinctions, some examples of where the different model choices are currently used, and some examples of operational consequences of the choice. The following summary is explained in detail on the following pages.

I. Data models

A. Ontology

* World, Identity, Segmentation, Ownership, Validation

B. Representation

* Embodiment, Basic representation, Attribute, Attribute definition, Type and Inheritance, Storage, Naming, Linking

C. Data management

* Distribution, Consistency, Access control, Records management

II. Activity models

A. Basic action models

* Action definition, Automation, Economic

B. Provision models

* Material provision, Change, Update, Indexing, Access management

C. Presentation models

* Presentation control, Client capability, Application, Timing

D. Interaction models

* Initiative, Turn-taking, Client/server state, Dialog context

E. Query models

* Query language, Query translation, Search activity, Query result

---

The Models

I. Data models

Data models are divided into Ontology (the kinds of objects in the world that are modeled in the system), Representation (the way that data represents the objects), and Records (how the representation records are stored and managed in the system).

A. Ontology

1. World model

The collection of "items" in a digital library (DLitems) includes but may not be limited to current connotations of "document". One or more of the following may be allowed or disallowed as "first class" items for storage, access, etc. in the library.

a) Information items

- Base items
  These are the items of end-interest to users, and generally exist independently of the library system. Many system properties derive from the decision as to the range of item types to be included (text, images, structured data records, video, audio, mixed documents, , ...)

- Library items
  These are the information structures that the library adds in order to provide access and management of the base items. It includes things such as collection listings, indices, catalogs and databases about information objects. There is no firm dividing line. An annotated bibliography provided by a professional society may be a thought of as a base item or a library item. The point in distinguishing is to recogognize the different treatment these get in current systems.

- Meta-item
  These are the information structures that the digital library system uses in its representation of other items. The include the meta-descriptions and other data structures used in representing the digital library items. They can in turn be treated as information objects in the same space of objects, or can be kept separate.

- Computational item

  - computer, program, process, schema, method,...

b) Real-world item

- person

- physical information object (book, X-ray, reel of film,...)

- general (e.g., museum objects cataloged in a library)

c) Social-world item

- organization
- project
- group (e.g., for access rights)
- ...

2. Identity model

What constitutes a distinct item in the information space?

a) bitwise (e.g., same checksum)

Digital items are distinct iff they have different bits. This is an extremely narrow but easily manageable sense of identity

b) multi-dimensional

Something can be the same object but have "variants" along one or more dimensions:

- Encoding (ASCII vs. other character codes; uuencoded, etc.)
- Format (Postscript, RTF, PDF,...)
- Resolution (spatial or in color dimensions)
- Version (over time)
- Language (e.g. English or French)
- Formatting (e.g., HTML form and corresponding pure ASCII)
- Contents intent (e.g., "today's weather forecast for Chicago" can be a single object whose contents change daily)
- Location (same file stored on two servers)
- ...etc.

c) identity of object vs. identity of record

In dealing with objects that aren't purely information, there can be a sense of identity that is based on the system contents (e.g., two different and distinguishable records for a person) and one that is based on the thing referred to.

3. Segmentation model
Composite objects have a structure of parts. In talking about hierarchies, we take sequences as just the one-level special case.

a) Predefined hierarchy (e.g., Head and Body for HTML)

b) Tagged hierarchy (E.g., SGML. Interpretation of the tags requires a definition, such as a DTD)

c) Labeled hierarchy (e.g., the sections and subsections of a reference manual, where there is a label for each)
This includes the special case of numbered pages. Note that there can be more than one segmentation of a DLitem -- a book may be segmented by sections and also by pages.

d) Derivable hierarchy (e.g., the sequence implicit in taking the frames as the parts of a video, or the paragraphs or words as parts of a text)

e) Open segments
These are parts whose boundaries can be set in a specific use, such as "clip" of video, a stretch of words in a document, a geometrically defined region of a map,... They can are treated as independent DLitems in many cases, such as sending a URL with selection information encoded at its end.

f) Granularity model
The grain size of the objects that will be treated as distinct DLitems (E.g., words in text, regions of the screen in video, whole file systems,.journals, book series, dictionary entries,...) This also includes composites. For example what is the unit that is a"document" that has been spread onto a number of interlinked web pages?

4. Ownership model
There are many systems that record some kind of "owner" of information objects, with different meanings.

a) Right to assign access permissions

b) Party responsible for the contents (in a social and legal sense)

c) Person to contact with regard to breakdowns concerning the information (the meaning on most web pages)

d) Recipient of payments for use

5. Validation model

In most systems today, validation is an all-or-none matter. If something is in a "library" it has whatever degree of authority and reliability is associated with the material as a whole. In a distributed library there will be a finer-grained association of these properties with particular DLitems.

a) Declaring authority (who said so -- e.g., on a list of prices)

b) Third-party certification (e.g., digital notary, seals of approval, ...)

B. Representation

Every information system must map the objects of interest onto some kinds of data structures, whether they be files, objects, relational databases, etc. We will not deal here with the specifics but the general nature of the mapping.

1. Embodiment model

The model of the relationship between the digital representations and the objects they represent. There is a big difference here between the tradition of bibliographic systems (the digital representation is a partial description of the "real" thing) and on-line information systems (e.g. WWW) in which what you get is all there is. As with many of the distinctions here, many systems provide some mixture.

a) "real" online object

There is a "real" information object in digital form, and all other characterizations are derived from it. This is the way we often think of files, on-line documents, etc.

b) "true" record

The representation is not a full description of the object, but only captures certain relevant characteristics. This is typically the case for bibliographic systems and is inevitably the case for things such as people, organizations, etc. where the "real thing" isn't in the computer.

c) multiple partial records
There may be no one privileged representation, but multiple records, each of which may express some attributes of the object.

2. Basic representation model
There are two basic forms for representation in common use: objects, and structured byte streams (which may be texts, or may be streams representing other media such as video).

a) object/attribute
An item is represented by giving a set of attribute-value pairs. (see below for the nature of the attributes)

b) object/method
An item is represented by an object which can respond to some standard set of methods indicating operations. Often these include the explicit setting and reading of attributes.

c) structured byte sequence
The most developed format of this kind is SGML, but many information objects are structured sequences based on some specialized or ad hoc conventions. Often the conventions correspond in part to attribute/value pairs (as in the <title> tag for HTML)

3. Attribute model
Since all of the representations end up with attribute value mappings of some kind we can look separately at the nature of the attributes. It is possible to intermix these in various ways.

a) data type attributes [string, number, date,...]
These are what have traditionally been in relational databases, standards for headers, etc.

b) content-encoding attributes
In putting DLitems into an attribute-only model, the part normally thought of as the "content" can be thought of as just one attribute. This will be of whatever type is suitable for the content type. One question is whether there should be a privileged content-encoding attribute for every item.

c) Pointer attributes [inter-item relationships]
Some attributes, though they may be represented using byte sequences such as numbers or alphanumeric strings are treated specially as the names of other DLitems. This includes the use of object pointers as values in objects, and of links in hypertext.

4. Attribute definition model
For a particular type of object, there can be a specified set of possible attributes.

a) schema-free / self-describing (property list)
In a schema-free model, attribute names can be assigned at will, and it is up to conventional agreement among the applications to interpret them appropriately. This the convention, for example for the experimental (X-...) headers in email, for Lisp property lists, etc.

b) universal schema
All items of a type can be encoded with a fixed schema which is "wired in" to the standards. The MARC bibliographic format and TEI headers are examples for specific subsets of DLitems.

c) Published standards
These can include widely-used standards, such as one of the bibliographic schemas (e.g.,. IAFA, Bibref,...) or image description standards (
TIFF...), as well as others we invent. Publication can included real-time publication on the net (by having a known server).

d) Schema carried with object
e.g., a DTD sent with an SGML file based on it.

e) Extensible
There can be a mixture of standardized attributes plus additional extensibility, as in GAIA, Internet email headers, Harvest...

5. Type and inheritance model

a) hardwired set of types

b) type hierarchy

- no inheritance (flat type system)

- single inheritance (most working object systems)

- multiple inheritance

c) cross-identity inheritance
e.g., inheriting attributes from the source file to the corresponding object file even though they have different identities

6. Storage model

a) file-system-centric
This identifies DLitems directly with corresponding file system objects (files and directories). It covers most current networked models, such as WWW basic model (extended in an ad hoc way by specialized URLs). It is easy to build, not very general (for many purposes we want a single file to hold many DLitems while one DLitem may be spread into multiple files)

b) document-centric (e.g. SGML documents with internally encoded meta-information in the form of tags and headers)
This doesn't handle second-party information independent of the original information, since the meta-information is included in the document.

c) database-centric (relational or object database)

d) descriptor-centric
There is an independent abstraction of "descriptors" which can be stored in more than one way (e.g., embedded in files and also stored in databases)

7. Naming model

a) locality of name spaces

- Global name space

- Hierarchically scoped declarable naming contexts (as in programming languages, and with hierarchical global schemes such as DNS)

- Name space associated with "chunk" (e.g., server designation in the standard for URLs)

b) stability of names

- Name uniquely determines content

- Content of name changes over time - e.g., URLs and Unix file names are time-dependent names.

- Special names to reflect "current" or "starting" time

c) canonicality

- Distinct canonical names imply distinct objects. Given an object it may or may not be possible to derive its canonical name.

d) meta-information encoded as part of name
For example, time/host stamps encode source/origin/history information, other naming schemes encode type information.

e) correlation with other name spaces [URL, URN, etc.]

8. Linking model
These have been laid out at length in the literature on hypertext.

a) object-object links

b) segment or point links

c) two way vs. one-way links

d) enforced link coherence vs. open linking

C. Data management
Whatever the form of representation structures, there will be mechanisms for managing them -- storing, copying, retrieving, etc.

1. Distribution model

a) Single privileged source (most older systems)

This can include caching to reduce load.

b) Coordinated replication (e.g., Lotus Notes, DNS, News groups)

c) Multiple partial sources (e.g., resource discovery systems)
The data may be partitioned into known places or there may be overlapping and/or redundant sources. One common division is to have metadata stored in one place (e.g., a database) and the "content" in another (e.g., conventional file system). Another is to have multiple databases, with items partitioned by producer, topic, etc.

2. Consistency model

a) Absolute
Inconsistent data should appear only in the case of a system error. This is a simplifying ideal but one that is hard to meet in a system that integrates information from existing (and unreliable) sources, such as library catalogs.

b) Dynamic (partial) resolution
Inconsistency is expected and there are heuristics or adjudication mechanisms for making decisions when it is detected (e.g., give a person the choice of which to use, or give "unknown" as a result).

c) Qualified information (Meta-marked)
Inconsistency is expected as part of system operation, and data is tagged with meta-information about its source that helps resolve inconsistencies. This is the general case for what we do in the real world when we get conflicting information.

3. Access control model
In this section we are looking at the way that access rights are allocated, not the mechanisms for establishing access.

a) Granularity of access control

- For materials: server, database, directory, file, record,...

- For accessors: host machine, IP address, group membership, individual, capability,...

b) Repertoire of rights (view, destroy, modify, modify additively, execute, copy, ...)

c) Nature of access control

- Authentication of appropriate rights

- Quid-pro-quo (e.g., charging for access)

d) Access characteristics

- prices, copyright restrictions,...

4. Records management model
Records management is another tradition, which has dealt with official records in organizations. But many of its issues apply to the extended library.

a) Persistence Guarantees

- distributed / open - nothing can be assumed

- archive of immutable selected items

- strategies and markers for selective archiving and retention schedules (e.g., time-to-live attributes)

b) Recording history of items

- provenance, original order, change history, access history, audit trail,...

II. Activity models

A. Basic action models

1. Action definition model

a) Standard actions, defined by applications and tools [Web]]

b) Specialized action by object type [Gaia]
This may include restriction of actions depending on object attributes.

c) Specialized action suites as "plug-ins" to applications [Viola]

2. Automation model

a) Human action primary (e.g., traditional library systems)

b) Support machine "users" (e.g., Z39.50 in place of traditional teletype interface to database)
This is not incompatible with a human-based interfaces. For example Dienst has two interfaces, one for each.

c) Provide "agents" in servers (e.g., notification agents)

d) Incremental automation with workflow
e.g., when a source document is updated, people who have derived things from the older version can be notified that updates are needed.

3. Economic model

a) resource measurement in use (e.g., search given a fixed amount of resources)

b) charging (various schemes from fee-per-byte to subscription, etc.)

B. Provision models

1. Material provision model

a) Centralized (most current library systems)

b) Distributed specialized (HTTP servers)

c) General use (read/write access to file systems, post to newsgroups)

2. Change model

a) Incremental change of elements of a DLitem (edit in place, or append)

b) checkout/edit/replace cycle

3. Update model
As base DLitems are entered or changed, how do corresponding changes get reflected in library DLitems such as indices, catalogs, summaries, etc.?

a) Always get information direct from base sources (update at query time)

b) Delayed distribution (e.g., re-index overnight)

c) Recompute when used if invalidated (e.g., lazy compilation)

d) Manual update (some user needs to invoke the operation)

4. Indexing model

a) Indexing is an add-on [Web]

b) Automatic indexing [WAIS]

c) Explicit provider control of indexing [Lotus Notes]

d) Cooperative indexing (between provider and other services) [Harvest]

5. Access management model
This includes the actions for establishing access control over individual DLitems or groups of items (e.g. by associating passwords with files or directory subtrees), for establishing individual identities, group identities, group membership, etc.

C. Presentation models

1. Presentation control models

a) Detailed control by provider (Acrobat/PDF)

b) Multiple views by provider (Lotus Notes)

c) Control by client (HTML)

2. Client capability model

a) Client has known characteristics (most in-house research systems that have specialized capabilities)

b) Broad base of client characteristics with full implementation for each (e.g., Acrobat/PDF)

c) Broad base of client characteristics with partial results (e.g., if you can't display images, use Lynx)

3. Application model

a) multiple applications for media types (current Mosaic)

b) Integrated single "viewer" application

4. Timing model

a) No control

b) Ordering of elements (e.g., inline images in HTML)

c) Controlled synchronization (e.g., Hytime, ScriptX,..)

d) View program runs interactively in the client (e.g., Viola)

D. Interaction models

1. Initiative model

a) Pull (send a query, get an answer)
This includes browsing, in which the "query" is the selection of some item that was represented in the result of the previous action, and is effectively linked to serve as a subsequent request.

b) Push (server gets new info and sends on basis of previous standing request)

2. Turn-taking model

a) Complete an action once initiated

b) interruptability, feedback in process, progress monitoring,....

3. Client/server state model

a) Stateless (original intention of Gopher, HTTP)

b) Server maintains state (most operating systems with logins, Z39.50, etc.)

c) Client maintains state (X-Windows, some Web browsers,...)

d) Mixture

4. Dialog context

a) history of interactions
This can be short term (session, result set) or long term (persistent history records).

b) user/host/network identity

E. Query models

1. Query language model

a) defined structure-based (e.g., SQL)

b) text-based (WAIS)

c) text-based with field structure (Notes)

2. Query translation model

a) canonical query language

b) pairwise translation

3. Search activity model

a) parallel searching of multiple sources

b) incremental searching (some results before completion)

c) resource-controlled searching

d) internal and external indices (w.r.t. the specific library system)

4. Query result model

a) one-shot results

b) incremental result sets

c) queries, results,etc., as full-fledged DLitems

Change history:
Converted (some parts) to HTML Jule 27, 1995
Original version Novemberr 21, 1994 by Terry Winograd