Creating Ontological Metadata for Digital Library Content and Services

PETER C. WEINSTEIN AND WILLIAM P. BIRMINGHAM

University of Michigan Digital Library

Artificial Intelligence Laboratory

University of Michigan

1101 Beal Ave., Ann Arbor, MI 48109-2110

{peterw, wpb}@eecs.umich.edu

Abstract

We use formal ontologies to represent knowledge about digital-library content and services. Formal ontologies define concepts with logic in a frame-inheritance structure. The expressiveness and precision of these structures supports computational reasoning that can be used in important ways. This paper focuses on the creation of ontological metadata.

We create ontological content metadata by generating it from MARC (MAchine Readable Cataloging) data. MARC contains much information that is hard to exploit computationally. In particular, relationships between works are implicit in shared values and natural-language notes. The conversion process involves specifying an ontological model, mapping MARC to the ontology, and reasoning about the data to create explicit links between works.

Service metadata will be supplied by providers who wish to participate fully in a digital library that is implemented as a decentralized multi-agent system. Agents advertise by describing their services in terms of ontologically defined concepts. We reason about these descriptions to organize them into subsumption taxonomies. Agents can then find the best available services to meet their needs by describing their needs, without requiring *a priori* knowledge of other agents. This infrastructure has demonstrated its usefulness in a multi-agent system organized as a computational economy.

1 Introduction

A fundamental issue in building libraries is how to organize large amounts of information so that users can find what they need, when they need it. To this end, librarians have developed sophisticated classification schemes and cataloging rules for creating metadata. Metadata describes works contained in libraries. Metadata enhances works' usefulness by providing a basis for search, and more generally, by identifying their intellectual and historical contexts.

Digital libraries will make greater demands on metadata than do traditional libraries. The quantity of information will be even vaster, and access will be provided with a large variety of services that help users define and satisfy their needs. Most of these services must be routinely provided without human assistance. Thus, it is essential that metadata used in digital libraries be amenable to computation. Digital libraries should be capable of reasoning about their contents to reformulate queries, customize services to the task and user, deduce new relations between works, and so forth. In short, the metadata must support inference.

We use metadata based on formal ontologies to support sound computational reasoning. Formal ontologies use logic to define concepts in relation to other concepts. We describe digital-library works with instances of ontology concepts. We can reason about relations among attributes defined in the ontology, and about relations among the works themselves. We define services as ontology concepts, and can reason about these definitions. For example, we identify when one service is a specialization of another even when this relationship is not asserted explicitly in the definition. This paper focuses on the creation of ontological metadata for digital-library content and services.

Formal ontology is an appropriate technique for modeling complex domains. Concept definitions can form webs of relations, rather than being limited to trees. Concepts can have many descriptive dimensions (attributes), may be partially described at any level of granularity (with any combination of dimensions), and may be viewed from many perspectives (accessed by different sequences of attribute values). For example, we can represent a song as being simultaneously music, linguistic expression, and possibly fiction, without needing to prioritize these characteristics with respect to each other. A user might search for an audio tool to hear some group of songs, or for songs that can be played with a certain audio tool. In an ontology, retrieval supports access from either perspective and at any level of granularity. In comparison, declarative formalisms with less expressiveness than ontologies, such as relational databases, force commitments to particular combinations and orderings of dimensions.

Our ontological model for content is centered around a hierarchy of five concepts that loosely describes the creation of work, and thus, how works are derived from other works. A CONCEPTION is an abstract work, an EXPRESSION adds description of the content, a MANIFESTATION adds publishing format, a MATERIALIZATION adds production format, and an INSTANCE has an address for a particular copy. In many respects this model articulates librarians' traditional world-view: it borrows most heavily from the proposal by the International Federation of Library Associations (IFLA 1996).

To make the creation of ontological content metadata practical, we generate it from MARC (Network Development and MARC Standards Office 1994-), thus leveraging the tremendous investment in that format. When two MARC records share data—for example, they might have the same uniform title and publisher—we may infer either that one work is derived from the other, or that both are derived from a common ancestor. MARC also includes many natural-language notes with information about derivation. Computers can make inferences based on shared data and process natural language to interpret notes, but preferable not to do this while users are waiting. Generating ontological metadata from MARC is thus a form of preprocessing, in which relationships implicit in MARC are converted into explicit, labeled relations amenable to manipulation by computers.

We are not currently trying to develop specific ontological models of digital-library services; we believe is premature to do so. Rather, we provide strong incentives for providers to describe their services ontologically as they become available. Our system, the University of Michigan Digital Library (UMDL)(Birmingham, Drabenstott et al. 1994), has a decentralized, agent-based architecture (Durfee, Mullen et al. 1998). A defining characteristic of this architecture is that agents form teams with other agents to solve problems. Agents choose to team with other agents—at least the first time they work together—on the basis of other agents' ontological definitions of their services.

Figure 1 provides a high-level overview of agents cooperating to answer a user's content query. The user agent asks a mediator agent to recommend one or more collection agents appropriate to the query. The mediator agent communicates with other mediator agents to help it make its recommendation. Each agent provides a specific service. For example, one agent might provide a thesaurus, another might know about topic hierarchies, and another might keep track of individual agent locations. The user agent then directly contacts the recommended collection agents to execute the query.

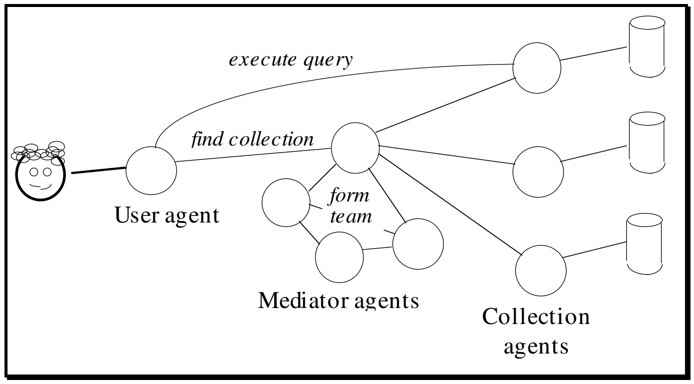

Figure 1: Forming an agent team to satisfy a query

The difference between service metadata for users and agents is mostly in the degree of detail. In Figure 1, the user is typically not aware of the need for a thesaurus. On another occasion, the same user might want to use a thesaurus directly. (Agent services, however, may be hidden from users.)

If we compare our approaches to the creation of content metadata and service metadata, we might ask whether we could create content metadata in a manner analogous to our approach for service metadata. Why not build infrastructure that encourages information providers to supply their own metadata? Potentially, other kinds of users—researchers, students, and so on—could also contribute to metadata knowledge bases as they use them. Computational reasoning would then be used to facilitate, edit, filter, manage, and apply user contributions. The role of the professional cataloging community would change correspondingly, to become increasingly focused on quality control and less on creating metadata. Indeed, we suspect that the ever-increasing flood of new information will eventually force changes in this direction. There are, however, institutional as well as technical obstacles to creating metadata in this way. For now, we consider this kind of knowledge sharing "future research".

The remainder of this paper is structured as follows. Section 2 defines "formal ontology", and describes its representation with description logic and the kinds of reasoning that are then available.

Section 3 reports on our creation of ontological metadata for content. Section 3.1 presents the UMDL ontology of digital-library content. Section 3.2 describes the generation of a knowledge base of metadata from a sample of MARC records.

Section 4 describes the infrastructure that incorporates the creation of ontological service metadata into the growth of the system. Section 4.1 explains "runtime service classification", the process by which agents advertise and find services using ontological metadata. Section 4.2 illustrates service classification in the context of the UMDL's experimental system, a society of agents organized as a computational economy.

Section 5 discusses the advantages and disadvantages of aspects of our approach compared to related work. Section 6 concludes the paper.

2 Formal Ontologies

In artificial intelligence, an "ontology" is a set of vocabulary definitions that expresses a community's consensus knowledge about a domain. This knowledge is meant to be stable over time, and reused to solve problems. Note that this concrete and utilitarian approach is quite different from "ontology" in philosophy, which concerns the abstract nature of reality apart from human endeavor.

Formal ontologies define vocabulary with logic. The syntax and semantics of the logic depends on the representation language (see Genesereth and Fikes (Genesereth and Fikes 1992) for KIF, and Borgida and Patel-Schneider (Borgida and Patel-Schneider 1994) for description logics). We focus on description logics, which are roughly equivalent to first-order logic in expressiveness, but have some limited second-order capabilities. They are different from first-order logic primarily in their focus on inheritance relations. Ontological concepts are typically organized in a subsumption taxonomy. Concepts are defined as specializations of their parents by appending additional relations.

Semantically, instances denote objects, concepts denote sets of objects, and relations denote sets of tuples of objects. A concept (C) subsumes another (D) if every instance of the latter is always also the former: $C \dashv D$ if and only if $D^M \subseteq C^M$, for all logical models, M, that map from symbols to a universe of objects where D^M and C^M are the extensions (denoted sets of objects) of those concepts. The meaning of a concept is appropriately construed as a function that maps from logical models to the concept's extension (Guarino 1997), Ω_C: $M \rightarrow C^M$. Thus, ontologies can be understood as nets of constraints that restrict the set of possible models.

Concept definitions may be construed, more or less explicitly, as frames. Definitions include a name, a set of relations to other concepts, and a natural-language description, which serves strictly as documentation. In slightly different contexts, relations may alternatively be called roles, slots, or dimensions. Subsumption may be called a "kind-of" relation. The "is-a" relation indicates membership of an instance in a concept. Objects that are instances of the concept inherit its relations. Objects that instantiate associated concepts linked by relations are called "role fillers", or just "fillers". Restrictions can be placed on filler values, such as numerical restrictions, or constraints in relation to other fillers. An instantiation of a concept may be partial, with some roles filled, and others waiting for fillers.

Another way to understand the nature of formal ontologies is to compare them to less formal structures. Consider the following spectrum of structures used to define vocabulary. On the informal end of the spectrum are library-classification schemes, such as

Dewey and Library of Congress. These structures are trees of unadorned symbols, where the relation between symbols, often called "is-a", is actually an imprecise commingling of subsumption and instantiation.

Moving to the middle of the spectrum, we find "informal ontologies", such as WordNet (Miller 1990). These structures have other kinds of relations besides "is-a", although "is-a" links are still the most important. Furthermore, informal ontologies are not trees, but directed graphs: concepts can have multiple parents.

On the formal end of the spectrum are ontologies such as those in the Ontolingua library at Stanford (Farquhar, Fikes et al. 1996). Here, the variety and precision of relations is elaborated to the point where the entire meaning of central concepts is captured by their relations to other concepts. In informal ontologies, the authoritative meaning of terms is in their natural-language descriptions. In formal ontologies, this descriptive text is documentation only. By analogy, imagine studying a C program. You read the comments first to learn how the program works, but keep in mind that the documentation may be wrong—it is the code that executes.

By itself, an ontology is a static structure without behavior. The representation system determines how the ontology can be used. We use description logic to represent our ontologies: specifically, Loom (MacGregor 1991), a description logic system in the KL-ONE family (Woods and Schmolze 1992). Description logic originally developed out of semantic networks, starting in the 1970's with a classic paper by Woods (Woods 1975) that drove home the need to remove ambiguity by formal specification of the semantics of concepts and links. Description-logic systems are susceptible to high computational costs. There are many options, however, with complex tradeoffs between the ability to reason in various ways at various times, and the efficiency and scalability of the system. We discuss this issue further in Section 5.

Description logics can do "automatic classification", which algorithmically places new concept definitions into their proper location in the ontology. The judgment of whether one concept subsumes another is based on the structure and content of the concept definitions. For example, in Figure 2 two services are defined, both a-kind-of RECOMMEND-DLCOLLECTION. The first line of each service indicates inheritance from the "recommend-dlcollection" concept, and the remaining lines further restrict the values of its slots. An initial colon (:) identifies a Loom keyword. Other symbols are from the ontologies. QUERY-PLANNING-FOR-SCHOOLS is found to subsume HIGH-SCHOOL-INFO-FINDER because every role in the former corresponds to a role in the definition of the latter, with a filler value that subsumes the corresponding filler value. Therefore, automatic classification depends on a precise meaning for every relation in the ontology. (See Woods (Woods 1991) for a thorough discussion of automatic classification.

This technology should not be confused with "automatic classification" based on keywords, or other non-logical approaches).

comparisons

<u>QUERY-PLANNING-FOR-SCHOOLS</u>
(:and recommend-dlcollection
    (:all recommend-dlcollection.has.audience school)

<u>HIGH-SCHOOL-INFO-FINDER</u>
(:and recommend-dlcollection
    (:all recommend-dlcollection.has.audience high-school)
    (:all recommend-dlcollection.has.topic science)))

Figure 2: Intensional check for subsumption

Description logic is also amenable to inference by *modus ponens* , which is a deductive process that allows entailed facts to be found.

3 Ontological Metadata for Content

This section describes the UMDL ontology for digital-library content, and the process of converting a sample of MARC records to a knowledge base of ontological metadata. The ontology itself is a set of concept definitions, the product of group discussions and hand-coding, first in stylized English and then in Loom.[1] The result of converting the MARC data is a Loom knowledge base: a set of interrelated instances of the ontology concepts.

3.1 The UMDL Content Ontology

Here we present the central concepts of the ontology, focusing on our model of bibliographic relations (relationships between works). At the time of writing, the ontology includes 183 concepts, not including concepts built into Loom. Of these, however, all but 16 central concepts are primitive. In other words, most of the concept definitions do not have relations sufficient to justify typing an instance as a member of the concept unless it is explicitly asserted to be a member.

A word of caution: few of our concept definitions have come easily. We spent hours considering alternative ways to conceptualize familiar concepts such as SONG, or DOCUMENT, and substantially more effort extracting order from slippery areas such as genre (we use upper-case to reference ontology concept definitions, as opposed to their meaning in the world). From this experience, we learned two important lessons. First, we need to find ways to make ontological definition easier. This problem is discussed further in Weinstein and Alloway (Weinstein and Alloway 1997). Second, we believe that any attempt to impose a single conceptualization on all digital libraries is bound to fail. We do

[1] http://www.umich.edu/~peterw/Ontology/ontology.html.

not seek one classification system to be used by everybody, but rather, a system that may be useful to many, and an approach (ontological modeling) that may be useful to many more. In particular, research to support semantic heterogeneity (see Section 5.3) may increase the feasibility of tolerating local variation in language without sacrificing all ability to compute using that language.

3.1.1 The Work Hierarchy

The backbone of the digital-library content ontology is a hierarchy of concepts that loosely models the creation of work (Table 1). This hierarchy revises, extends, and clarifies the proposal by IFLA (IFLA 1996). The creative process starts with an idea. As long as it is abstract, we call the work a CONCEPTION. After the content of the work has been specified, it is an EXPRESSION. When the EXPRESSION is published, it becomes a MANIFESTATION. The physical embodiment of the MANIFESTATION is a MATERIALIZATION; if digital, the MATERIALIZATION is a DIGITIZATION. A particular copy of a MATERIALIZATION is an INSTANCE (not to be confused with a description-logic “instance” of a concept). We believe that our work hierarchy is consistent with informal definitions of work by Yee (Yee 1994) and others in the library community. Note that “work” itself is not a defined concept, but an ambiguous natural-language word that is very useful for conversation about fully or partially instantiated objects in the work hierarchy.

Table 1: The work hierarchy

Concept	Definition	Example
CONCEPTION	A concept, plan, or design for work.	An idea for a story.
EXPRESSION	A conception with specified content.	A manuscript for a novel.
MANIFESTATION	An expression packaged in a publishing format.	A published edition of the novel.
MATERIALIZATION, DIGITIZATION	A manifestation embodied in a physical format.	The novel coded in SGML.
INSTANCE	A copy of a digitization.	A particular file on a disk.

More precisely, each level of the work hierarchy is defined by the attributes associated with it. Figure 3 provides an overview. Any unlabelled links indicate “kind-of” relations. A CONCEPTION is identified by its UNIFORM-TITLE, CREATOR, and SEQUENTIAL-NUMBER for numbered musical works. An EXPRESSION has a “genre”. This is the most complex part of the ontology; we use a natural-language term to refer to it, and discuss it separately below. A MANIFESTATION has a PUBLISHING-FORMAT, which is divided into RECORDED for BOOKs, JOURNALs, etc., and LIVE for performances, ongoing scientific observations, and so on. A MATERIALIZATION has a DIGITAL-FORMAT, which includes TEXT, IMAGE, VIDEO, and other kinds of formats.

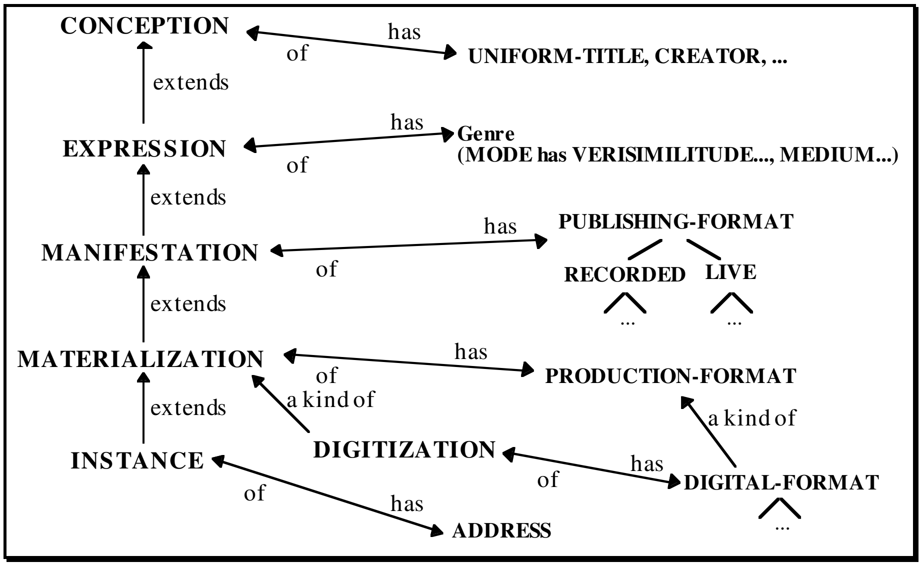

Figure 3: Attributes associated with the work hierarchy

The relationship between the levels of the work hierarchy is somewhat trickier than is immediately apparent. We want the lower levels to inherit from the higher. So, an INSTANCE is a kind of MATERIALIZATION, MANIFESTATION, and so on. Thus, an INSTANCE has a UNIFORM-TITLE that can be retrieved directly without navigating all the way up the work hierarchy. We also, however, want sibling instances to share data at all levels of the hierarchy. In object-oriented systems, instantiating an object allocates separate memory for the members of every inherited class. In description logic, fortunately, we can have this cake and eat it too: each concept in the work hierarchy is defined as both being a "kind-of" and "extending" the concept above. The inherited value is then constrained to be the "same-as" the value associated by the "extends" relation. Currently, Loom does not properly enforce these "same-as" constraints, but we achieve equivalent results for our limited application by procedurally enforcing these constraints.

Genre has been especially difficult to define, and this area of the ontology is provisional. We use "genre" as the Library of Congress does (Network Development and MARC Standards Office 1994-), with a much more general meaning than in English. Like "work", no single definition can capture all of the meaning required; rather, "genre" refers to MODE and all of its subsidiary concepts. MODE has multiple dimensions. VERISIMILITUDE divides FICTION from NON-FICTION, which are mutually exclusive. MEDIUM can be SOUND, SYMBOLIC, or VISUAL presentation: these are not mutually exclusive. Thus, an instance of SONG can be FICTION, SOUND, and SYMBOLIC all at the same time.

Figure 4 illustrates the structure of genre. (The concepts at the top of the boxes "have" the underlined concepts, the concepts in brackets are "kinds-of" the underlined concepts, and unlabelled links also indicate inheritance). The full structure is not a tree, but a web of connections. For example, the concept SONG is defined in stylized natural language as:

SONG. MUSIC with lyrics, thus a LINGUISTIC-STYLE and a TONGUE.

Stating that SONG is MUSIC indicates a kind-of inheritance relation, while the references to other concepts identify roles.

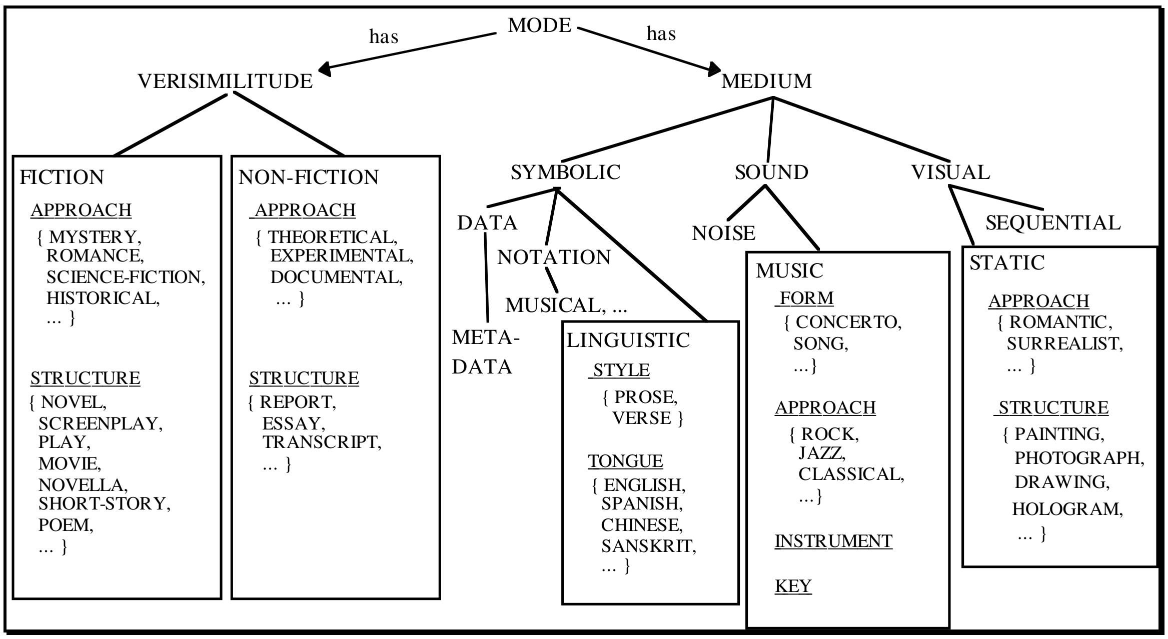

Figure 4: The structure of genre

In addition, every level of the work hierarchy is associated with CREATORs constrained to be of certain types. For example, on the EXPRESSION level there are EDITORs, TRANSLATORs, and so on.

3.1.2 Relations among Works

Families of related works have a simple and elegant structure that is amenable to efficient maintenance and manipulation by computers. Namely, these structures are trees, as illustrated in Figure 5. CONCEPTIONs can have multiple EXPRESSIONs, but an EXPRESSION can have only one CONCEPTION; and so on for each level down to INSTANCE. Full metadata for a particular copy of a work includes a single (description logic) instance at each level.

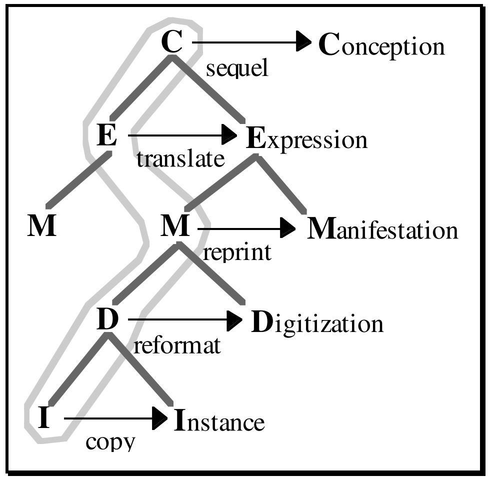

Figure 5: A work with five derivation relations

Each kind of relation between works occurs at a particular level of the work hierarchy. For example, sequels are between CONCEPTIONs; translation is from one EXPRESSION to another; reprinting is at the MANIFESTATION level, reformatting involves DIGITIZATIONs, and copying is between INSTANCEs. Similar constraints apply to relations that add, continue, or describe other works rather than derive from them. In general, we call relationships describing the creation of work *ontogenic* relations.

Services that operate on higher levels of the work hierarchy involve rights that are relatively profound. For example, borrowing a book is a service at the INSTANCE level. A digital library might reformat a document from Microsoft Word to ClarisWorks: this would be an operation at the DIGITIZATION level. Customized news service is at the MANIFESTATION level. Collaborative editing is at the EXPRESSION level. Licensing the rights to use fictional characters, for example, would be at the CONCEPTION level.

Whole-Part relations are different from ontogenic ones, and we have not yet implemented these. Figure 6 illustrates our current proposal using a hypothetical scenario involving a textbook for teaching C++. The second edition adds a diskette containing sample programs. CONTAINS relations are established at each level of the work hierarchy, starting at the first level for which the contained work is considered independent relative to the original work. Links at each level are required to avoid ambiguity about the contents of the original work if, subsequently, new works are created that are derived from the contained work.

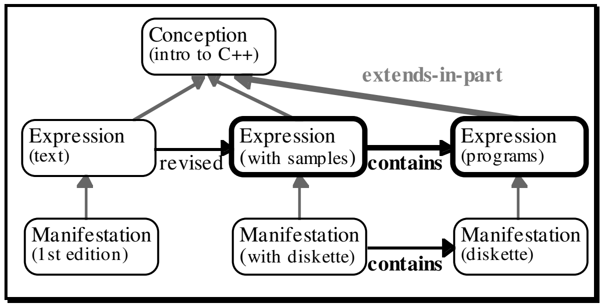

Figure 6: Adding a Part to a Whole

The ontological model changes the nature of the venerable debate about when a work requires its own "main entry" (Yee 1994). Here, a subjective decision is apparently required to determine the level of the work hierarchy where the new work deserves an independent instance that is not shared with the original work. For example, in Figure 6, if the programs already existed as examples in the original text, then they would share the EXPRESSION with the whole work, but would have a separate MANIFESTATION. Instead, the figure is drawn with a separate EXPRESSION, consistent with a judgment that the sample programs were first created for the second edition. The ontology refines the issue by distinguishing among five levels of work description, rather than two. The ontology also changes the practical context of the decision; to some extent, the choice is determined by the attributes associated with each level. For example, if the successor work does not share its genre with the predecessor, then it cannot share its EXPRESSION either.

3.1.3 Potential Benefits

Ontology-based metadata can provide the following important benefits:

- <u>Very precise queries</u>. For example, consider an amateur musician who needs to learn a part. She can pose a query equivalent to: "Find me a score for Beethoven's Fifth Piano Concerto, arranged for cello, with a recorded performance that is also available". A similar query could be posed without specifying the particular concerto. The traditional distinction between access and descriptive metadata is blurred, because all ontology-based data can be used for both purposes. Attributes designated for controlled vocabulary, however, will continue to be particularly useful for access.
- <u>Explicit paths from vague to precise queries</u>. A vague query identifies values desired for one or two attributes. In ontologies, attributes are defined by their relations to other attributes, providing numerous intuitively natural sequences for prompting users to articulate their requirements. Frequently, query refinement will occur by starting at the top of the work hierarchy and progressing downwards. Queries can also be refined by requesting increasingly specific versions of known attributes, or by adding values for attributes associated with the known values.

- Integrating catalogs. The same kind of reasoning used to merge work descriptions when generating our knowledge base can be applied to integrate multiple catalogs (see Section 3.2).
- Per-service calculation of license needs. License requirements are a function of the kind of service provided, the work involved, and the licensing agreement that covers the work. The ontology's model of relations between works can provide both a language for expressing license agreements, and a means for computing fees associated with services at runtime. See Section 4.2 for preliminary work in this direction.

3.2 A Knowledge Base of Metadata

This section describes the process of converting MARC data to a knowledge base of ontological metadata, and a user interface for browsing the knowledge-base.[2]

Originally, we requested 500 US MARC records from the University of Michigan library for works either by or about Beethoven. The idea was to provide data with many works related to others in the sample, and also a variety of publishing formats and genres. The actual sample contains 493 records, out of a total of almost 5000 in the library related to Beethoven.

3.2.1 Generation from MARC

Generating the knowledge base involves mapping data from MARC to the ontology, and reasoning about the data to identify relations among works. The process is almost fully automatic, and is certainly fully automatable.

We generate the knowledge base in four steps:

1. Convert binary MARC data to text tagged with ontology concepts.
2. Extract coded attributes and values from natural-language comments in the tagged text.
3. Convert the tagged text to Loom assertions. At this stage, every value is treated as a distinct object.
4. Reason about the data to establish explicit relations between works.

We convert MARC fields and values to tagged text with a set of four control files that describe the MARC format and its relation to the UMDL ontology. The primary control file maps each selected MARC field and code to one or more ontology concepts. Some mappings are conditioned on the MARC value to permit a single field to be mapped to multiple ontological concepts. The second control file establishes priorities for situations where multiple MARC fields map to the same ontological concept. The third control file

[2] This interface is available at http://www.umich.edu/~peterw/Ontology/Beethoven/demo.html.

identifies the location of codes within the MARC fields, contingent on MARC record type and bibliographic level. The fourth control file includes information for each code, including its length and the location of the code-value table.

For the Beethoven knowledge base, the extraction of coded attributes and values from the MARC notes was done by a human, but in a rapid, mechanical way that could be achieved computationally with a lexicon and some straightforward natural-language processing. For example, the 245c value "compiled and edited by H. C. Robbins Landon" translates into "RE H. C. Robbins Landon" ("RE" is a keystroke-saving code for "role, editor"). We spent an average of only forty seconds per record on this transcription; with less than 500 records, it was simply easier to do this by hand than to write a program to do it.

The conversion of the tagged text to Loom assertions (TELL statements) is trivial, since we postpone all reasoning until the data is in Loom. Every MARC record is assigned an ITEM-NUMBER associated with a MATERIALIZATION (we are dealing only with metadata and so do not include the INSTANCE level). The fifth control file provides the paths that link each of the destination ontology concepts to MATERIALIZATION. For each step along a path, a separate Loom instance is created, with the MARC value (if there is one) linked to the instance of the destination concept. Thus, after execution of the TELL statements, the knowledge base contains metadata where every value assertion is separate. For example, every ITEM-NUMBER has its own MATERIALIZATION, MANIFESTATION, EXPRESSION and CONCEPTION. Furthermore, single MATERIALIZATIONs are frequently linked to multiple instances of intermediate concepts, such as MUSIC, with one instance for each value related to the concept.

Stage 4 is by far the most interesting. Here, we reason about relations implicit in the MARC data, to make them explicit. Our treatment, however, can only be considered a crude first pass. There is no end to the effort that might be invested in developing increasingly refined rules to glean additional information. See Tillet (Tillet 1992) for an interesting review of how relationships among works are represented in past and current catalogs.

Frequently, partial or imprecise data suggests relationships that cannot be confirmed. In these cases we assert relations that are considered "tentative". For example, the knowledge base includes two works that are missing UNIFORM-TITLEs; one has the MANIFESTATION-TITLE "Goethe et Beethoven", while the other is "Goethe und Beethoven". We consider the CONCEPTIONs of these works to be "tentatively-the-same".

By definition, "certain" relations that are not true are errors; false "tentative" relations are merely false. Instances are merged only if their common identity is certain. Otherwise, a "tentatively-the-same" relation is asserted to link them.

We are liberal with "tentative" relations, but conservative with "certain" relations. Thus, many tentative relations are incorrect. The result of any deduction that uses a tentative relation is also tentative.

Our processing includes the following steps:

a) Merge multiple instances of intermediate concepts with unification. The unification algorithm uses cardinality constraints that are part of ontology definitions. If both an instance's asserted type is subsumed by another's, and multiple values are permitted, then the instances are merged.
b) Identify and merge shared CONCEPTIONs. CONCEPTIONs with the same UNIFORM-TITLE, CREATOR, and SEQUENTIAL-NUMBER for numbered musical works are considered to be the same. When the UNIFORM-TITLE is missing, we use the MANIFESTATION's TITLE instead. Differences in punctuation or case are ignored.
c) Identify and merge EXPRESSIONs that have the same CONCEPTION, if all genre and associated CREATOR values match, or if a manifestation-level certain derivation is recognized.
d) Identify and merge MANIFESTATIONs that have the same EXPRESSION, if the PUBLISHER and PUBLICATION-DATE match, and other PUBLISHING-FORMAT values of one instance are a subset of those of the other instance. Also merge MANIFESTATIONs if a materialization-level certain derivation is recognized.
e) Identify CONCEPTIONs that are "tentatively-the-same" using the same rule described in (b) above, but ignoring missing CREATOR names, and using a permissive matching algorithm for comparing titles.
f) Identify EXPRESSIONs that are tentatively-the-same as in (c), with either matching genre or CREATORs, but not both.
g) Identify MANIFESTATIONs that are tentatively-the-same as in (d), but rather than a shared EXPRESSION, require only EXPRESSIONs that are tentatively-the-same, or a shared CONCEPTION.
h) Establish relations to preceding works based on MARC notes describing derivation relations. If no predecessor is found in the work-family, generate metadata that describes what is known about the predecessor (including all levels of the work hierarchy above the level of the derivation, and information from the MARC note).

   There are nineteen types of relations transcribed from the MARC data, mapped to eight relations in the ontology. Values transcribed from the notes of the successor work are compared to ordinary values describing candidate predecessor works. For each of the source relations, a subset of the following attributes are compared: author, editor, language, title, publisher, city, date, publisher-item-number. "Points" are awarded for a match for each value, depending on the source relation. A match to a candidate predecessor is considered certain or tentative if points are accumulated above respective thresholds.

i) Establish certain and tentative reproduction relations on the MATERIALIZATION level, triggered by a FACSIMILE production format or an "original date" in the MARC publication-date code.

We generated the Beethoven knowledge base on a Sun SparcStation 20. Stages 1 and 3, written in C++, completed in a few seconds. Stage 4 is written in Lisp using Loom, and required several hours. The binary MARC data required 0.5 MB of disk. The Lisp image of the resulting knowledge base requires 35 MB (this is not a space-efficient format), but loads for execution in about half a minute.

Table 2 summarizes the transformation of the knowledge base as the reasoning proceeds. After initial loading of the data there is one instance on each level of the work hierarchy for each MARC record. The "Merged" column shows the results of steps (a) through (d), above. Steps (e) through (g) do not affect the number of work instances. The "New" column shows the number of predecessors generated to represent works outside of the collection in steps (h) and (i). The "All" column includes empty instances at the lower levels, generated to complete the work family of new predecessor instances created at higher levels.

Table 2: Number of work-hierarchy instances

	Raw	Merged	New	All
Conceptions	493	286	1	287
Expressions	493	473	128	602
Manifestations	493	490	26	645
Materializations	493	493	58	706

These results reflect the nature of MARC data, our sample, and our reasoning process. Perhaps the single most salient feature is the relative success of merging works at the CONCEPTION level compared to other levels. This is primarily due to authority-controlled titles—the UNIFORM-TITLE—and creator names. The UNIFORM-TITLE is designed to link bibliographic records. Not surprisingly, our output reflects this. 295 of the original 493 CONCEPTIONs included a UNIFORM-TITLE; these were merged to yield only 88 CONCEPTIONs. Of the 198 MARC records that do not have a UNIFORM-TITLE, we found 60 that are tentatively-the-same as at least one other CONCEPTION.

Our sample of MARC records extracted from the university library is not actually random, in the sense that it seems to either include, or not include, all of the records for a given UNIFORM-TITLE. Thus, we believe that had we included all records related to Beethoven we would have obtained a comparable rate of merging of CONCEPTIONs. We surmise, however, that this rate would be smaller if all records in the library were included (we picked the Beethoven domain because we expected to find many relations among works).

The potential for merging on levels below CONCEPTION depends on the relative percent of the total work universe contained in the sample. For example, the most convincing evidence for merging EXPRESSIONs is to find a certain manifestation-level derivation between two works: for example, that a work was reprinted from a version (that shares the same CONCEPTION) published in some given year. We were able to merge only three pairs of EXPRESSIONs on this basis. Of course, a library is less likely to purchase a given work if it already owns a closely related version.

The ontological model of genre associated with EXPRESSION is the most complex part of our ontology, and rules for reasoning about merging EXPRESSIONs should be correspondingly complex. The rules ((c) and (f) above) that we applied in this effort were simplified to the greatest degree of all our rules, compared to what they should be. Therefore, we were able to confirm relatively few certain equivalences on this level, and have a relatively large number of tentative matches (we consider 214 EXPRESSIONs to be tentatively-the-same as at least one other EXPRESSION).

MARC contains more notes about ontogenic relations at the EXPRESSION level than at any other. An ontogenic relation on the EXPRESSION level identifies sharing at the CONCEPTION level; again, MARC illuminates most strongly the distinction of greatest traditional interest, whether the item in hand is a new “work”. We were able to establish 23 cases where an EXPRESSION in our sample had been derived in some way from another EXPRESSION in the sample. In 128 cases the original EXPRESSION was not in our sample, and so we generated new EXPRESSIONs including whatever is known about the predecessors.

Table 3 lists the frequency of the different types of ontogenic relations identified. All “certain” relations are also considered “tentative” relations: the semantics is that we know, with at least that level of certainty, that the relation holds. A single predecessor might have relations to several successors. The converse can also hold, but is less frequent. In our results the fanout is almost exclusively tentative, but this may be because of our inability to merge new work instances generated when the predecessor was not found in the sample.

Table 3: Types of ontogenic relations identified in the Beethoven sample

Level	Relation	Certain Predecessors	Certain Successors	Tentative Predecessors	Tentative Successors
Conception	Sequel	1	1	1	1
Expression	Critiques	2	2	2	2
	Revised	43	43	43	48
	Supplements	7	7	7	11
	Translated	78	80	78	118
Manifestation	Reprinted	16	16	16	19
	Republished	12	12	12	14
Materialization	Reproduced	59	58	59	58
	Total	**218**	**219**	**218**	**271**

The new, partial work instances that we generate when identifying ontogenic relations may be of some interest to users in and of themselves. More generally, they will be merged with full descriptions of the original works if they are found in other catalogs that are integrated into the knowledge base.

3.2.2 Browsing the Knowledge base

We developed a Java interface to help library researchers develop a feeling for the structure of ontology-based metadata. The initial window (Figure 7) lets the researcher select a work by identifying a type restriction from a list of genre and publishing format attributes: autobiographies, concertos, sound recordings, and so on. The researcher can also specify a keyword to be included in one of several fields, including creator, title, topic, and others such as editor.

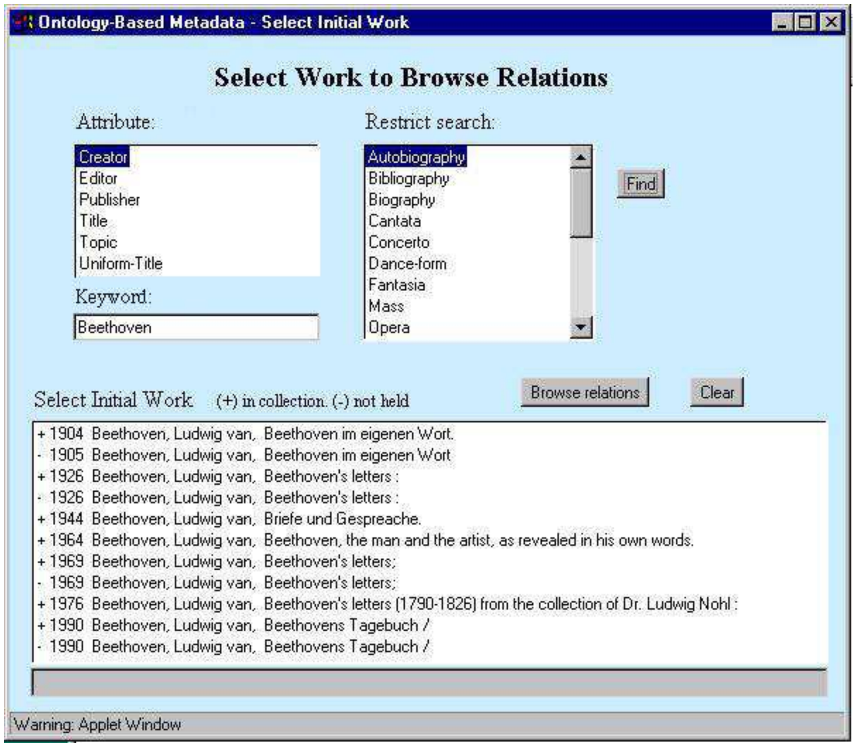

Figure 7: Selecting a work

The second window supports browsing work families by navigating relations that connect the works (Figure 8). All metadata associated with the selected "current" work is displayed in a text area to the left of the window. In the middle are the relations that link that work to others. After picking a relation, and then one of a list of related works, all metadata for the related work is also displayed. The related work can be made the current work, and the process repeats.

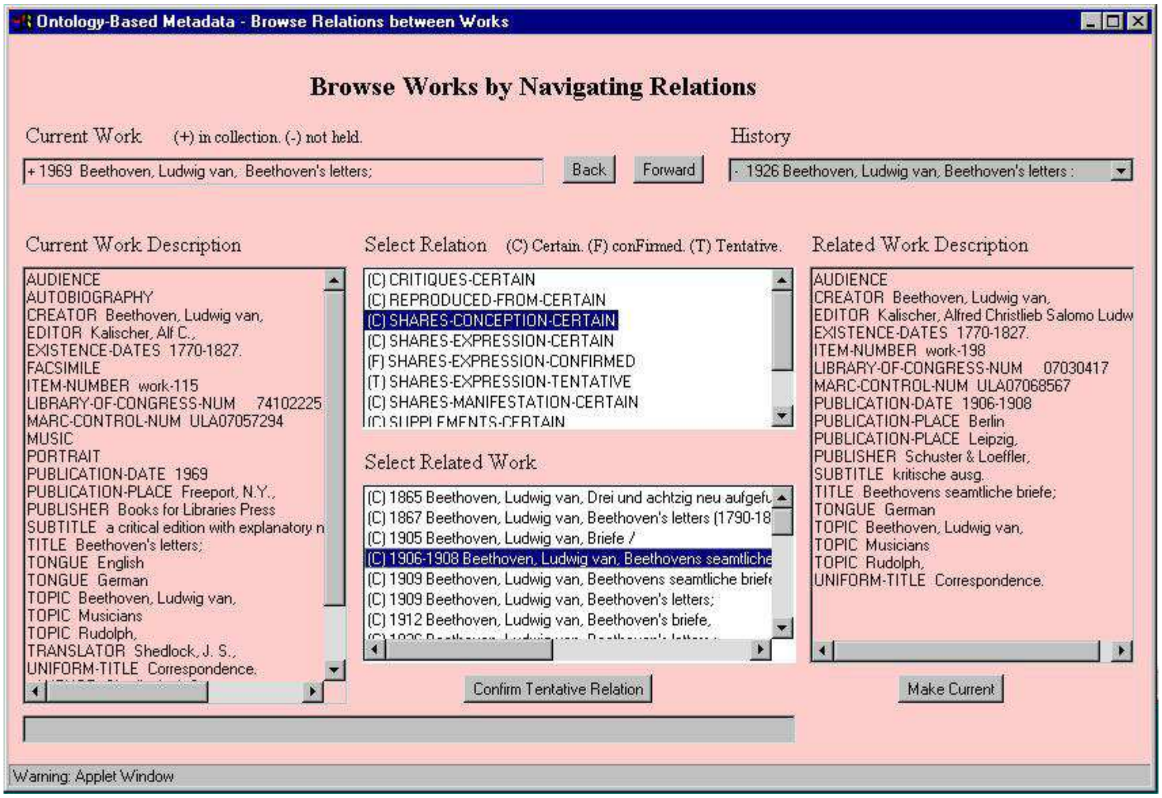

Figure 8: Browsing by navigating relations

Users in various institutional contexts may potentially contribute to the knowledge base while they are using it. To suggest this possibility, our demonstration interface provides a button that lets a user "confirm" a tentative relationship between two displayed works. We consider "confirmed" to be an intermediate level of certainty between "tentative" and "certain".

4 Ontological Metadata for Services

This section describes the role of ontological metadata for services in a decentralized, agent-based digital-library architecture. This system requires (the functional equivalent of) ontological service metadata to meet the design goal of extensibility. Furthermore, agents enhance their competitiveness by using ontological service metadata. Thus, the architecture assures the development of ontological service metadata.

The UMDL Service Classifier Agent (SCA) [3] maintains ontologies of agent services via a process that we call "runtime service classification". Section 4.1 describes how agents advertise and find services using an SCA. Section 4.2 describes the SCA's roles in an experimental multi-agent system organized as a computational economy. This simulation

[3] Developers can communicate directly with an SCA on the web; the URL is http://www.umich.edu/~peterw/Ontology/sca.html.

illustrates how the benefits of runtime service classification on the level of individual agents manifest on the level of the system as a whole. Section 4.3 summarizes these benefits.

Agent communication requires shared language. We assume that agents adhere to linguistic constraints on several levels. On the level of message structure (how to order the bytes to describe the communicated object) we require a protocol equivalent to CORBA (Object Management Group 1995). On the level of agent dialogue (the structure of conversational interaction) something like KQML is required (ARPA Knowledge Sharing Initiative 1993). KQML messages include pairs of attributes and values that describe the purpose and context of the message. The ":content" field contains the substance of the communication; the part that talks about the world. We assume that the syntax of the content value is constrained by a language, such as KIF (Genesereth and Fikes 1992), that can be translated to the language of an SCA. The terminology included in the content value must be either reserved words in the syntax language, or terms from ontologies. The ontologies define linguistic constraints on the level of symbol semantics. To be specific, we use Xerox's ILU (Inter-Language Unification) version of CORBA, KQML, and the SCA uses Loom.

The UMDL service ontologies are divided into nested modules, each of which is an ontology. The most general includes services that we consider part of a "generic" digital library. The second module adds concepts specific to the UMDL implementation, such as "auctions". The third module describes agent services. We call this last ontology "dynamic" because agents define new service concepts at runtime. In contrast, "static" ontologies are either fixed, or are changed slowly over time by human committees.

4.1 Runtime Service Classification

The SCA is implemented using the UMDL agent class (Durfee, Kiskis et al. 1996), which handles communication on the levels of message structure and agent dialogue. For message syntax, we use Loom.

To advertise a service, an agent submits a service description to an SCA, which uses automatic classification to locate the description within the agent-services ontology. The service description can use terms from any of the nested UMDL ontologies. The agent also includes a preferred label in the classification request (the desired name for its service). The SCA responds to classification requests in one of three ways:

- If the service description is classified as a new concept, the preferred label is returned as the recommended service label.
- If the service description is logically equivalent to a service that has already been classified, then the label for the existing service will be returned.

- If the concept is new but the label has already been used for another service description, the SCA automatically generates a new label.

If an agent provides multiple services, it classifies each one separately. It is also possible for several different kinds of agents to provide the same service—the SCA ensures that they all use the same label.

The example in Figure 9 is a description for a service to recommend a collection on the topic “science” for “middle school” audiences. The first line indicates inheritance from the “recommend-dlcollection” concept, and the second and third lines further restrict the values of its slots. An initial colon (:) identifies a Loom keyword. Other symbols are from the ontologies. Service concepts are always treated as "defined" (not primitive) to support recognition based on their descriptions.

```
(:and recommend-dlcollection
        (:all recommend-dlcollection.has.audience middle-school)
        (:all recommend-dlcollection.has.topic science)))
```

Figure 9: Example service description for classification

In KL-ONE systems, there is a deeply rooted bifurcation between concepts, which denote sets of objects, and instances, which denote individual objects. For example, classification operates on concepts, but retrieval is over instances. In Figure 9, therefore, the fillers (“middle-school” and “science”) must be both concepts (for classification of the description) and instances (for retrieval of the advertised description). The SCA automatically asserts that fillers are instances when necessary.

Agents can use a variety of strategies to learn about classified services. Often a single query suffices, but a series of queries can also be used to systematically explore the dynamic ontology. Loom's query expression language has full first-order expressiveness. Variables may be chained to traverse ontological relationships, either to restrict matches to services with specified role fillers, or to reveal fillers for services that are otherwise selected. If the query is successful, the SCA returns a list of sets of bindings, where each set includes a value for each variable in the query expression. The example in Figure 10 is a service description in a query that asks for a service to recommend a collection, where the service is suitable for an audience that is a kind of school. Symbols starting with a question mark (?) are variables.

```
(:and   (recommend-dlcollection ?service)
        (recommend-dlcollection.has.audience ?service ?audience)
        (school ?audience))
```

Figure 10: Example service description for a query

Loom's query language can be extended with custom predicate functions that accept or reject every combination of bindings considered by the query. The SCA also provides several "wrapper" functions that are invoked once for the entire query. Whereas predicate functions execute within Loom queries, wrapper functions typically include a Loom query. The example in Figure 11 shows an example of the most-specific-subsuming wrapper function, which ranks all subsuming services in order by the weighted proximity of their role fillers to the ideal. This wrapper is used to search upwards in the concept taxonomy through increasingly general services for the first that is available. The first phrase is a Loom query that identifies candidate services. The next line characterizes the ideal solution. The :priorities list weights the importance of each query dimension. The :attenuation-factors quantify the judgment of proximity, compensating for the density of development of the static ontology in which the role fillers are defined.

```
(:wrapper #'most-specific-subsuming
        '(:and   (recommend-dlcollection ?service)
                 (recommend-dlcollection.has.audience ?service ?audience)
                 (school ?audience)
                 (recommend-dlcollection.has.topic ?service ?topic)
                 (topic ?science))
        '((middle-school audience) (biology ?topic))
        :priorities (2 1)
        :attenuation-factors (0.6 0.8))
```

Figure 11: Wrapper for ‘most specific subsuming’

The SCA can also automatically construct new concepts from existing terms. The “define-inclusive-concept” wrapper function returns a definition for a concept that subsumes each of a list of concept labels. The new definition is for the least-common-subsuming concept subject to a language restriction. Since Loom's concept language includes disjunction (“or”), the unrestricted least-common-subsuming concept would be the (uninteresting) disjunction of the input concepts. Thus, only elements of concept definitions that fit the form specified in Figure 12 are considered. The inclusive concept is defined as having:

1. the intersection of superconcepts,
2. only roles with relations shared by all, and
3. filler value restrictions that are the intersection of role value restriction superconcepts.

```
{ (and   [direct superconcepts]*
         [(ALL relation value-restrictions)]* ) }
```

Figure 12: Syntax of definitions constructed by the ‘define-inclusive-concept’ wrapper

4.2 The SCA in the Service Markets Society

The UMDL’s Service Markets Society (SMS) (Durfee, Mullen et al. 1998) is a computational economy. Agents buy and sell services within markets. Markets are implemented with auctions, which provide a bidding protocol whereby buyers and sellers negotiate over price. This approach provides efficient resource allocation with surprisingly reasonable communication costs (Wellman 1993). The basic mechanism is familiar. As a selling agent's computational load increases, it costs more to produce marginal services. So the agent asks for a higher price, receives less business, and effectively sheds load to other producers. A full discussion of the rationale and implementation of the computational economy is out of the scope of this paper, but Durfee et. al. (Durfee, Mullen et al. 1998) provides a good overview.

The SMS is a simulation, and does not deliver digital-library services to end users. Nor does it identify the quantity of a service. For example, a purchase might be for an hour's worth of query planning, or a bundle of 100 queries.

The current SMS includes a single SCA, and five other kinds of agents. Three of these use the SCA. Several dozen instances of these agents may be active simultaneously. At any time, new agent instances can become active, and active ones can close. All SMS agents use the same SCA, and thus subscribe to the same ontologies.

Services are sold by Query Planning Agents (QPAs). These agents are instantiations of the Task-Planning Agent, which is a generalized, goal pursuing procedural reasoner (Vidal and Durfee 1995). Each QPA is configured upon creation to provide query planning service specialized for one of 49 (hypothetical) possible services. This space of services is the cross product of seven values for each of two attributes. The possible values are arranged in small subsumption hierarchies. The attribute "audience", for example, can be "any-audience", "school", "professional", "middle-school", "high-school", or "government" or "business" as kinds of professional.

Services are purchased by User Interface Agents (UIAs). These agents seek to buy one of the same 49 services potentially provided by QPAs. Since this service may not be provided by any active QPA, UIAs identify the best available service using either of two search strategies: increasing generality (most-specific-subsuming), or increasing specificity (most-general-subsumed). The search criteria and strategy can be changed at any time. UIAs seek to buy services periodically, at a rate subject to interactive modification.

Auction Manager Agents (AMAs) define markets, and maintain an appropriate population of auctions to service each market. In the SMS, a single AMA spawns an auction for each service provided by a QPA. A QPA asks the AMA for an auction to sell its services; if no auction is active selling that service, the AMA spawns a new auction. The AMA uses the SCA to classify the auction service description, and then checks the registry to see if an auction providing that service is active.

Figure 13 illustrates the SCA's interactions with the other agents. Agents are in ovals, ontologies in rectangles, and the arrows connecting agents represent messages, paraphrased in natural language. To communicate with the SCA, agents use terminology from the nested ontologies. The thin line around the SCA is jagged to show that the set of available terms is dynamic; both QPAs and the AMA add concepts to the agent services ontology. Messages use font styles that correspond to the ontology labels (plain, italics, and upper-case), to highlight the source of their terminology.

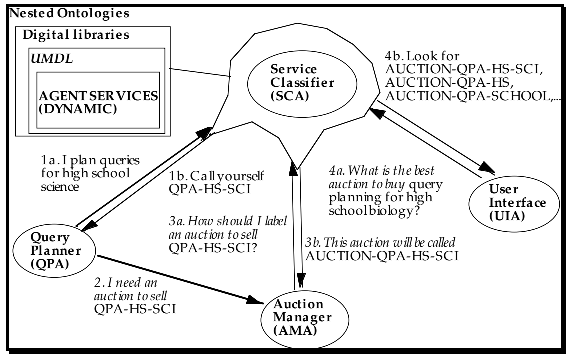

Figure 13: Agent interaction with the SCA in the SMS

Figure 13 shows how agents use the SCA to define new terminology from existing terms at runtime. Auctions sell some service, but they don't need to know anything about that service. The AMA asks the SCA for a service label for an auction, using the QPA’s service label, but it does not know anything about that service. The SCA classifies the auction service using characteristics inferred from the QPA’s service label. Thus, the SCA can respond to the UIA’s request for an auction, which is given in terms from the static ontologies, rather than the label that the AMA used to define the auction service. Concepts in the SCA’s dynamic ontology hide knowledge, just as words chunk meaning in natural language. This appropriate hiding of knowledge reduces overall system complexity, and increases reusability and maintainability.

One SMS scenario illustrates the capacity of the system to utilize new agents without requiring any modification to existing agents. For example, Figure 14 shows a display that reports on the status of a UIA that is requesting query planning appropriate for high school biology, with a search strategy that identifies the least general available service that subsumes the desired service. Prior to the snapshot shown in the figure, this agent was buying from an agent selling query planning for high school science. When the SMS experimenter spawns a new agent whose service subsumes the request more specifically than that of existing agents, the UIA switches without prompting to buy from the new agent, as seen in the figure. Similarly, if the SMS experimenter shuts the new agent down, the UIA automatically reverts to buying high school science query planning, because that is the best service available to meet its needs.

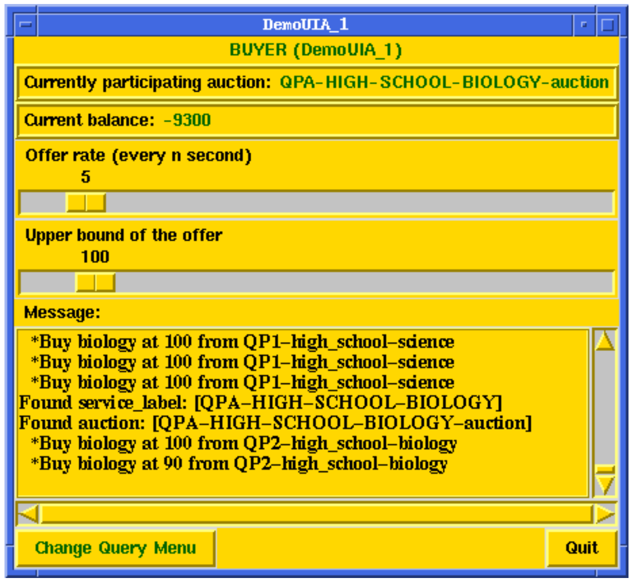

Figure 14: A buying agent after automatically switching to a more satisfactory service

In another SMS scenario, markets include providers with different service descriptions: these markets sell goods that subsume the advertised services of each seller. In this scenario, user agents learn to identify a subset of providers that provide relatively fast results compared to other providers. The user agents then use the SCA to define a new concept that includes the advertised services of the fast providers, but excludes the services of the slower providers. On request from the users, the auction manager creates a new, more specialized market. Sellers that can provide fast service can then participate in the new, relatively specialized market.

4.3 Third-Party Development

Runtime service classification fosters evolution of the society to meet user needs by encouraging third-party development of new agents. The SCA:

- Identifies opportunities. By examining the gap between service requests and recommendations, entrepreneurs can identify niches for services that are desired, but not currently available.
- Reduces agent entry costs. Existing agents may automatically switch to using the new agent, without requiring either modification or notification beyond routine advertising.

These benefits derive from the technology of service classification, and the way in which it is used. Technically, the key is declarative description of services, organized to enable ranking of available services given a target and search strategy. In practice, agents requesting services must state their requests in terms of what they need, not in terms of what they know is currently available. They must also periodically repeat their search, rather than always using agents with whom they have previous experience. The SMS illustrates this behavior.

5 Discussion

In this section we compare our research to other work with similar objectives. We scope the discussion to include efforts to create metadata that is significantly more complex, and thus more powerful in its potential applications, than is currently available. We organize the discussion by considering issues related to representation, modeling, and the degree to which it is necessary to share syntactic and semantic constraints.

5.1 Representation

Selecting a representation language is usually a difficult decision. The fundamental problem is the tradeoff between expressiveness and tractability. Informally, "expressiveness" is the degree of generality and complexity that can be encoded in statements of a language. Tractability is the degree of efficiency possible when interpreting and reasoning with the language. Both expressiveness and tractability have many facets depending on what one is expressing and computing, respectively. The choice of representation is therefore highly sensitive to the definition of system objectives.

It is often possible to translate between different representations as appropriate for different tasks (Gruber 1993). A statement in a relatively expressive language can be translated to a relatively simple language with some loss of information, although it is hard to formulate general rules for appropriate ways to simplify. Translation cannot add information, of course. Therefore, the practical issue is to determine a representation to store knowledge that is expressive in the most important ways for the anticipated applications, that can be feasibly maintained, and that can be translated to other languages as needed.

Ontological models of bibliographic relations have advantages over other proposals for new catalog structures that use less expressive representations. One idea, with many variations, is to extend MARC with fields for explicit links to other works. MARC, however, is highly redundant both within and especially between records (Leazer 1993). Redundant data reduces the efficiency of storage and updating. The most serious threat for digital libraries, however, is the prospect of overwhelming users with numerous versions of very similar works (Vellucci 1997). MARC also has an archaic file organization. It is possible to convert MARC's idiosyncratic record structure, field codes, subfield codes, and value-conditioning "indicators" to more standard formats. The need to do so, however, impedes utilization of advances in mainstream database technology.

Several proposals explore using relational databases to store the new catalogs (Green 1996; IFLA 1996). These systems are fast and reliable for massive quantities of data. Unfortunately, relational technology is best suited for applications such as banking, with large quantities of highly standardized data. The complexity of bibliographic data subverts normalization (Green 1996). Normalization is the process whereby data is divided into a separate table for each domain entity, indexed by a combination of dimensional values that provides a unique key for each record. Bibliographic data requires many descriptive dimensions, partial descriptions at multiple levels of granularity (with any combination of dimensions), and viewing from many perspectives (access by different sequences of dimensional values). Relational design forces a commitment to particular combinations and orderings of dimensions. In complex domains, the result is a proliferation of tables that destroys efficiency and maintainability.

Object-oriented databases are appropriate for complex domains. Heaney (Heaney 1995) has proposed an object-oriented catalog with a model similar to our own. Actually, our ontologies are object-oriented, with a class structure that is declaratively encoded. In typical object-oriented systems, the class structure is encoded procedurally (in a programming language rather than in a data structure). Information that is procedurally encoded is not accessible to reasoning. For example, in standard C++ there is no built-in way to determine the type of a given object. Of course, developers can add methods to return object type—and superclasses, subclasses, and so on. If this is done in a well-principled way, then the developers are heading back down the road to description logic, or other representation systems used to reason with ontologies.

For generating ontological content metadata, description logic is too expressive in some ways, and not enough in other ways. Description logic's greatest asset is its ability to do automatic classification. We did not use automatic classification to generate the knowledge base of metadata, nor is it likely to be important to support end-user queries. Deductive object-oriented databases seek to combine logic with support for object-oriented structures (Kifer 1995). These systems do not do automatic classification, but are good at deducing new individuals and relations and can handle large quantities of data efficiently (Kandzia and Schlepphorst 1997).

On the other hand, a formal way to reason about uncertainty would be very useful for generating ontological content metadata, and uncertainty is not handled well by description logic. For example, the ontology contains many attributes that do not have corresponding MARC data. Many of them could be deduced, however, from other values. To illustrate, given a BIOGRAPHY without a SOUND or VISUAL medium, we might deduce BOOK. This kind of reasoning could be most vigorously pursued with default (non-monotonic) logic.[4] Relatively fine-grained evaluation of the degree of uncertainty associated with a proposition, such as afforded by probability-based approaches, would also be valuable. See Hunter (Hunter 1996) for a good overview of methods for reasoning with uncertainty. Unfortunately, no known approach is entirely satisfactory. For now, some sort of *ad hoc* compromise (such as adding "tentative" relations to description logic) is unavoidable.

Automatic classification is vital for runtime service classification. This application reasons with description logic concepts, whereas content metadata is represented as instances. One idea, therefore, would be to split description logic systems into two parts, a relatively heavy-duty part for concepts, and a leaner part for managing large number of instances. This is not a new idea, however; almost all of the KL-ONE systems do have this hybrid design (Woods and Schmolze 1992).

The poor performance of particular existing description logic systems does not necessarily mean that description logic in general cannot handle large knowledge bases. The various proofs showing that subsumption algorithms, and thus classification, are NP-complete are not quite to the point, since concept definitions generally do not grow arbitrarily large; rather, the biggest problem is that classification of very general concepts can take time proportionate to the size of the concept taxonomy (Woods 1991). Loom, for example, is a very expressive system with a huge array of capabilities, including truth maintenance, default reasoning, a full first-order query language, and disjunction and negation in concept definitions. Loom can be surprisingly fast, but in some situations is very slow. The high-level, general finding of an empirical study of description logic performance showed performance quadratic in the size of the knowledge base (Heinsohn, Kudenko et al. 1994); but, it is not clear what the bottlenecks are, and these systems have come a long way since the time of the study. The new version of Loom is implemented in C++ rather than Lisp, as is the new CLASSIC (Brachman, McGuinness et al. 1991). Also, it is always possible to reimplement a description logic customized to an application's needs. In research now underway, we achieved a speedup of more than two orders of magnitude in this way.

5.2 Model of Work

Other proposals for new catalog structures have work hierarchies with two (Leazer 1993), three (Heaney 1995), or four (IFLA 1996) levels (see Velluci (Vellucci 1997) for an overview). Ours has five. There is no advantage to having fewer levels, however, as long

---

[4] Loom, as the most expressive of the KL-ONE systems, does support default implications, but at the cost of sacrificing other forms of inference (such as the :RELATES concept-forming operator).

as every level adds information to every work. The associated attributes remain, and need to be attached somewhere.

We borrowed most heavily from the IFLA proposal (IFLA 1996); indeed, this can be considered our starting point. IFLA's model also has a central hierarchy, consisting of the concepts WORK, EXPRESSION, MANIFESTATION, and ITEM. We renamed WORK to CONCEPTION because attributes at all levels are part of what we mean by "work". We added MATERIALIZATION because we needed a place to attach digital formats, and renamed ITEM to INSTANCE for clarity, although perhaps COPY would be better, as in Green (Green 1996).

The big advantage of our model compared to IFLA's, we believe, derives from using an ontological rather than a relational approach. IFLA defines the meaning of its terms with pages of often confusing text. We define each level of the work hierarchy by precise association with attributes that are also ontologically defined.

Note that the work hierarchy imposes a dimensional ordering that we could avoid. Figure 15 illustrates an alternative structure for work, in which EXPRESSION, MANIFESTATION and DIGITIZATION are complementary subconcepts of CONCEPTION, and INSTANCE inherits from all of them. In this model, a new EXPRESSION creates a new INSTANCE that shares the old MANIFESTATION and DIGITIZATION.

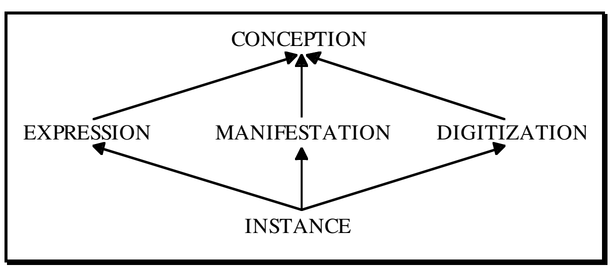

Figure 15: An alternative structure for work

The advantage of this less-structured conceptualization of work is that it reduces redundant metadata in cases where there are minor changes on the EXPRESSION level. We are concerned in particular with the "versioning problem" that derives from the ease of modifying many electronic documents, resulting in a proliferation of very similar works. A new version creates redundant metadata, depending on the type of modification. If a new version requires a new DIGITIZATION, then very little redundancy in metadata occurs in the UMDL work hierarchy. A minor change in the content of the document, however, requires a new EXPRESSION, and all of the lower levels of the work hierarchy are duplicated. With the structure in Figure 15, however, only a new EXPRESSION node is required.

We chose the work hierarchy instead of a less-structured model because we believe it will be more useful for supporting user inquiry. Also, techniques traditionally used for version control—storing changes from one version to another, rather than replicating information that stays the same—can also be applied to metadata, thus reducing the disadvantage of the work hierarchy.

5.3 Syntactic and Semantic Flexibility

One other project has taken an approach very similar to ours for runtime service classification. Agents in the InfoSleuth project at MCC (Bayardo, Bohrer et al. 1996; Nodine and Unruh 1997) advertise by submitting ontology-based descriptions of their services. Broker agents reason to match requests for services to these descriptions. The InfoSleuth ontologies are highly self-descriptive: the descriptions talk about themselves as frames. This enables translation between alternative syntaxes that support different types of reasoning.

Out approach, in comparison, is semantically rich. We organize the space of potential services by maintaining a taxonomy of services, with the ultimate intention of supporting communication without requiring developers to agree on term semantics at design time. InfoSleuth agents communicate about the ontologies and objects that they can manipulate, but there is little attempt to represent the meaning of these objects or the relationships between them.

We call language “semantically heterogeneous” when single terms reference disjoint or overlapping sets of objects, or different terms reference the same or overlapping sets. Neither InfoSleuth nor service classification in the UMDL handle semantic heterogeneity, since the SCA assumes that agents share all ontologies. In InfoSleuth, the syntax can vary, but shared symbols are assumed to share semantics. If a system grows in a decentralized manner, however, semantic heterogeneity is inevitable: the space of problems is infinite, while lexicons are not. In the library world, semantic heterogeneity manifests as incompatible metadata sets. Semantic heterogeneity is a formidable problem. Restrictions on syntax, semantics, and the process of language development, however, should make it easier to create shared understanding with ontologies than in less constrained contexts.

Precise mapping between ontologies almost always requires a manual process that is difficult and time-consuming. Some types of concept mismatches are easily handled, but others are deeply rooted and subtle (Lehmann 1995; Visser, Jones et al. 1997).

Imprecise mapping is potentially more amenable to automation. These methods necessarily start with some sort of overlap between the source and target ontologies. This overlap can be in the form of shared "typical" instances (Lehmann and Cohn 1994), shared concepts (Campbell and Shapiro 1995), or shared parents from which terms in the source and target ontologies inherit.

Currently, we adapt the latter approach by assuming a restricted form of semantic heterogeneity that we call “differentiated ontologies” (Weinstein 1997). Concepts in differentiated ontologies are not shared, but inherit definitional structure from concepts that are shared. Hypothetically, a society of agents starts with all agents subscribing to the same ontologies. Over time, new, increasingly specialized agents join the society, and these agents (or their developers) define new ontologies to describe themselves by adding new relations and concepts to definitions in existing ontologies. The union of differentiated ontologies is approximately a single ontology. (We permit only monotonic growth of this union).

We expect agents to participate in communities that are defined by their use of an SCA associated with some particular differentiated ontology. We match inter-community requests to candidate target expressions by building and evaluating "rough mappings". These are structures that include a set of one-to-one correspondences between concepts and relations in the source and target expressions. The expressions are coded in description logic, and represented as graphs; concepts are nodes and relations are edges (Borgida and Patel-Schneider 1994). In rough mappings, nodes are pairs of matched concepts and edges are matched relations. Rough mappings thus identify syntactic similarity as structural isomorphism. Between any pair of expressions there are many alternative mappings. The largest and most densely linked are ranked as the best. This mapping algorithm extends artificial intelligence research on analogy (e.g., see Gentner (Gentner 1990)).

6 Conclusions

We organize digital-library content and services with formal ontologies. The UMDL content ontology models bibliographic relations between works. Families of related works have a tree structure. Common attributes are generally associated with the upper, shared levels of the hierarchy, therefore greatly reducing redundant data.

It is possible to automatically create ontological content metadata from existing metadata. We generated a knowledge base from a sample of almost 500 MARC records. This process identifies relationships implicit in the data and makes them explicit.

We obtain ontological service metadata from agents seeking to advertise. Service classifier agents use description logic to automatically classify these definitions in subsumption taxonomies, and can then recommend the best available services to agents that make requests in terms of what they need.

The results reported in this paper are small demonstrations of what we intend to be large systems. We expect our ontological model of digital-library content to be useful to others working in this domain, although it is far from complete in detail. Our procedure for converting MARC records to an ontological knowledge base may be useful to other efforts to enrich existing metadata. We anticipate mapping data described by multiple kinds of metadata to families of differentiated ontologies, which will then support loosely integrated query service.

Description logic is the right tool for runtime service classification. Applying it to the SCA was straightforward, and yields some interesting behavior. For example, SMS agents automatically switch to using new agents that more closely fit their needs. The agents can reason in this way because of the expressiveness and precision of ontological metadata.

Acknowledgments

Gene Alloway and Judy Ahronheim have been instrumental to the development of the UMDL ontology for digital-library content. Tracy Mullen, Anisoara Nica, Sunju Park, and Jose Vidal each contributed substantially to the SMS, and Edmund Durfee was our steady guide and critic. We also thank the anonymous referees for their accurate critique of the submission draft of this paper.

This work was supported by the NSF/ARPA/NASA Digital Library Initiative under grant CERA IRI-9411287.

References

ARPA Knowledge Sharing Initiative (1993). Specification of the KQML agent-communication language., ARPA Knowledge Sharing Initiative, External Interfaces Working Group. http://retriever.cs.umbc.edu/kqml/.

Bayardo, R. J. J., W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, D. Woelk (1996). InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. Austin, TX, MCC. http://www.mcc.com/projects/infosleuth/papers/sigmod97_final.ps.

Birmingham, W. P., K. M. Drabenstott, et al. (1994). The University of Michigan Digital Library: This is Not Your Father's Library. Digital Library '94 Proceedings, June 1994, 53-60. http://www.si.umich.edu/UMDL/pubs.html.

Borgida, A. and P. F. Patel-Schneider (1994). A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Language. Journal of Artificial Intelligence Research 1: 277-308.

Brachman, R. J., D. L. McGuinness, et al. (1991). Living with CLASSIC: When and How to Use a KL-ONE-Like Language. Principles of Semantic Networks. J. F. Sowa. San Mateo, California, Morgan Kaufmann: 401-56.

Campbell, A. E. and S. C. Shapiro (1995). Ontologic Mediation: An Overview. IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal.

Durfee, E. H., D. L. Kiskis, et al. (1997). The Agent Architecture of the University of Michigan Digital Library. Special Issue of the IEE British Computer Society Journal of Software Engineering on Developing Agent Based Systems 144(1):61-71.

Durfee, E.H., T. Mullen, S. Park, J. M. Vidal, and P. Weinstein (1998). The dynamics of the UMDL service market society. In Matthias Klusch and Gerhard WeiB, editors, Cooperative Information Agents II, LNAI, pp. 55-78. Springer.

Farquhar, A., R. Fikes, J. Rice (1996). The Ontolingua Server: a Tool for Collaborative Ontology Construction. Palo Alto, California, Computer Science Department, Stanford University.

Genesereth, M. R. and R. Fikes (1992). Knowledge Interchange Format Version 3.0 Reference Manual. Palo Alto, California, Computer Science Department, Stanford University.

Gentner, D. (1990). The mechanisms of analogical learning. Readings in Machine Learning. J. W. Shavlik and T. G. Dietterich, Morgan Kauffman: 601-622.

Green, R. (1996). The Design of a Relational Database for Large-Scale Bibliographic Retrieval. Information Technology and Libraries 15(4): 207-221.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5(2): 199-210.

Guarino, N. (1997). Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology. (Ed.) M. T. Pazienza, Lecture Notes in Computer Science, Vol. 1299, Springer.

Heaney, M. (1995). Object-Oriented Cataloging. Information Technology and Libraries 14(3): 135-153.

Heinsohn, J., D. Kudenko, et al. (1994). An empirical analysis of terminological representation systems. Artificial Intelligence 68(2): 367-397.

Hunter, A. (1996). Uncertainty in Information Systems. London, McGraw-Hill.

International Federation of Library Associations (IFLA) (1996). Study on the Functional Requirements for Bibliographic Records. Frankfurt am Main, Germany, Deutsche Bibliothek.

Kandzia, P.-T. and C. Schlepphorst (1997). DOOD and DL - Do We Need an Integration? Intelligent Access to Heterogeneous Information Workshop at KRDB-97, Athens, Greece.

Kifer, M. (1995). Deductive and Object Data Languages: A Quest for Integration. 4-th Intl. Conf. on Deductive and Object-Oriented Databases, Singapore, Springer.

Leazer, G. H. (1993). A Conceptual Plan for the Description and Control of Bibliographic Works. School of Library Service. New York, Columbia University.

Lehmann, F. (1995). Combining Ontologies, Thesauri, and Standards. IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada.

Lehmann, F. and A. G. Cohn (1994). The EGG/YOLK Reliability Hierarchy: Semantic Data Integration Using Sorts with Prototypes. Third International ACM Conference on Information and Knowledge Management (CIKM-94), New York, ACM Press.

MacGregor, R. M. (1991). The Evolving Technology of Classificiation-based Knowledge Representation Systems. Principles of Semantic Networks. J. F. Sowa. San Mateo, California, Morgan Kauffman: 385-400.

Miller, G. A. (1990). WORDNET: An On-Line Lexical Database. International Journal of Lexicography 3(4): 235-312.

Network Development and MARC Standards Office (1994-). USMARC format for bibliographic data: including guidelines for content designation. Washington D.C., USA, Library of Congress, Cataloging Distribution Service.

Nodine, M. H. and A. Unruh (1997). Facilitating Open Communication in Agent Systems: the InfoSleuth Infrastructure.

Object Management Group (1995). CORBA: The Common Object Request Broker: Architecture and Specification, Release 2.0.

Tillet, B. B. (1992). The History of Linking Devices. Library Resources and Technical Services 36(1): 23-31.

Vellucci, S. L. (1997). Bibliographic Relationships. International Conference on the Principles and Future Development of AACR, Toronto, Canada.

Vidal, J. M. and E. H. Durfee (1995). Task Planning Agents in the UMDL. In Proceedings of the Fourth International Conference on Information and Knowledge Management (CIKM) Workshop on Intelligent Information Agents, 1995. http://jmvidal.ece.sc.edu/papers/index.html.

Visser, P. R. S., D. M. Jones, et al. (1997). An Analysis of Ontology Mismatches; Heterogeneity versus Interoperability. AAAI-97 Spring Symposium on Ontological Engineering, Palo Alto, California.

Weinstein, P. (1997). Agent Communication in Semantically Heterogenous Societies. http://www.umich.edu/~peterw/papers.html.

Weinstein, P. and G. Alloway (1997). Seed Ontologies: growing digital libraries as distributed, intelligent systems. Second ACM International Conference on Digital Libraries, Philadelphia, PA, USA.

Wellman, M. (1993). A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems. Journal of Artificial Intelligence Research 1: 1-23.

Woods, W. A. (1975). What's in a Link: Foundations for Semantic Networks. Readings in Knowledge Representation. R. J. Brachman and H. J. Levesque. Los Altos, California, Morgan Kauffman: 217-242.

Woods, W. A. (1991). Understanding Subsumption and Taxonomy: A Framework for Progress. Principles of Semantic Networks. J. F. Sowa. San Mateo, California, Morgan Kaufmann Publishers: 45-94.

Woods, W. A. and J. G. Schmolze (1992). The KL-ONE Family. Semantic Networks in Artificial Intelligence. (Ed.) F. Lehmann, Pergamon Press.

Yee, M. M. (1994). What is a Work? Part 1: The User and the Objects of the Catalog. Cataloging and Classification Quarterly 19(1): 9-28.