Metadata Architecture for Digital Library Integration

Su-Hsien Huang1, Hao-Ren Ke2, and Wei-Pang Yang1,3

1 Institute of Computer Science and Engineering, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu, Taiwan 30050, R.O.C.

2 Library and Institute of Information Management, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu, Taiwan 30050, R.O.C.

3 Department of Information Management, National Dong Hwa University 1, Sec. 2, Da Hsueh Rd., Shou-Feng, Hualien, Taiwan 97401, R.O.C.

Abstract: Although the rapid development in digital library has accelerated the use of metadata, conventional metadata is insufficient to be integrated across heterogeneous digital libraries. In this paper, three-layer metadata architecture is proposed to integrate digital libraries by activating the metadata structure. Metadata extraction layer obtains data automatically from distributed digital libraries. Semantic query layer searches transparently to solve semantic heterogeneity. Data clustering layer integrates search result with each sub-structure to improve accuracy. Adapting the three-layer architecture is beneficial for semantic search and alleviates the administrative effort to construct digital library services.

Keywords: Metadata, digital libraries, architecture, semantic inference, structure clustering

1 Introduction

Having received considerable attention in recent years, a digital library (DL) represents an Internet-based architecture that can access all kinds of information from anywhere. Research in DL integration relies on novel architecture to solve the technical, semantic and legal aspects of interoperability. From the perspective of autonomy, the two sides in the spectrum of DL interoperability are centralized and decentralized. The centralized DL architecture, like Stanford Infobus architecture [3], Z39.50, Dublin Core (http://purl.oclc.org/dc/), is simple and reconciles DLs by a concordant middleware [4][7][11]. This architecture leaves interoperability machinery outside independent DLs, and requires large amount mediation to transform different DL information into canonical model. On the other side, the decentralized architecture interacts DLs with a language that communicates the semantics, structure and operations of all materials among different DLs without prearrangement and mediators, like University Michigan Digital Libraries [5], the Agent Communication Language (ACL), Knowledge Interchange Format (KIF) and Knowledge Query and Manipulation Language (KQML). This architecture concentrates in the autonomy of DL but no centralized facilities to ensure the scalability. As a result of the both mentioned architectures, there comes into one common factor – metadata to interoperate heterogeneity. Metadata are the basic units to translate different material and used as a communication basis among DLs. In our point of view, metadata have great power to integrate DLs. The four trusts for metadata in DL integration are: 1. the rich expression power to solve the semantic inconsistency 2. the easy transformation to interoperate diverse formats. 3. the structure information to derive relationships 4. the economic aspect to integrate. Consequently, the rich expression power and easy exchange motivate us to provide metadata architecture for DL integration. However, conventional metadata architecture leads the DL integration into critical challenges, the lack of the structure consideration, and the distribution of metadata. To overcome these challenges, three-layer metadata architecture is proposed: metadata extraction layer, semantic query layer and data clustering layer. Metadata extraction layer adapts a rich data model to reveal metadata structure. The structured metadata are beneficial for data extraction from distributed DLs. Moreover, the second semantic query layer exploits structure information to derive relationships among metadata which can solve the semantic ambiguity in DL search. Finally, the data clustering layer integrates search result by calculating each sub-structure respectively to enhance the clustering accuracy.

The rest of this paper is organized as follows.
Chapter 2 is the related work. Chapter 3 proposes the metadata model. Chapter 4 represents three-layer metadata architecture. Chapter 5 draws the conclusion.

2 Related Work

Many researches have addressed the feasibility of using metadata to describe resources in DLs, using approaches including the Warwick Framework, Dublin Core (http://purl.oclc.org/dc/), Resource Description Framework (RDF, http://www.w3.org/TR/PR-rdf-schema/). The Warwick framework is a conceptual model of metadata that aggregates metadata packages into containers and then relates these packages to each other [10]. This framework separates the management and responsibility of specific metadata and allows access to various different sets of metadata. The container technology has influenced subsequent developments, including Dublin Core and RDF. Dublin Core, which defines 15 basic metadata elements, focuses mainly on resources for Internet-based applications. RDF provides a standard means of representing metadata by XML, employing statements to describe properties of and relationships among items on the Web. Many studies have applied RDF to resource discovery. For example, the Multimedia Description Framework (MDF) extends RDF to describe multimedia contents [7].

Sharing metadata architecture in distributed DLs drives the use of metadata. INRIA [1], 5SL [6] and Open Digital Library [13] apply metadata to describe, derive and federate services in distributed DL. To efficiently manage metadata in distribution environment, a well-designed metadata architecture should support not only the effective manipulation of native metadata but also the autonomous management of distributed metadata [2][12]. The metadata architecture proposed by Stanford University, called Infobus, is one realization of mediation architectures [3]. Infobus uses five service layers to enable uniform access to distributed heterogeneous information resources and services. In Infobus, Attribute Model Proxies describe metadata as first class objects, and Metadata Repository deposits and indexes metadata can be provided. These two kinds of metadata help Stanford DLs to seamlessly communicate metadata. Metadata in DL interactive architecture also attempts to facilitate metadata communication. The University of Michigan proposed an agent-based architecture, called University Michigan Digital Libraries (UMDL) [5]. UMDL refines the basic agent architecture to satisfy the needs for an open information economy. UMDL expresses agents using ontological semantics and employs metadata to represent information. UMDL communicates agents with each other by Knowledge Query and Manipulation Language (http://www.cs.umbc.edu/kqml/).

3 Metadata Model

![Data Model Diagram](image)

Fig. 1 Data model

3.1 Metadata representation

The data model of a metadata is constructed by three types of Resources - Entity, Attribute and Value, which is based on Warwick framework to further extend its functions[10]. For example in Fig. 1, a Person (Entity) has a Name (Attribute) called John (Value).

Definition 1 (Resource): A Resource $R=\{Id, Name, Type, Doc, Ver, MO, Def, Lang, RA, Obl\}$ is the fundamental unit of metadata with ten basic properties listed in Table 1. The ten properties are optional. Only required properties are contained in the Resource.

Definition 2 (Schema): A Schema is a triple set $\{Ent, Incs, Atts\}$ to represent metadata structure:

- $Ent \subseteq Resource$, $Ent$ depicts name and information of a Schema.
- $Incs \subseteq (Atts_a, Atts_c)$, each pair $<Atts_a, Atts_c>$ indicates that $Atts_c$ is the subtype of $Atts_a$. $Incs$ is assumed to be acyclic.
- $Atts \subseteq Resources$ is the set of Attribute. In $Atts$, the Type attribute represents the name of $Atts$.

Definition 3 (Metadata): A Metadata is a triple set $\{Ent, Sche, Values\}$ to represent how to encapsulate data by Schema where,
Ent ⊆ Resource, Ent describes information of a Metadata. The value of Type attribute in Ent is the name of the Metadata.

Sche ⊆ Schema, Sche represents the Schema associate with the Ent.

Values ⊆ (Attribute, Resource), each pair (A, V) represents that V is the value of Attribute A. V is allowed to contain composite Values corresponding to Sche.

<table>
<thead>
<tr>
<th>ID</th>
<th>the unique identifier of a Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of a Resource and can be hierarchical ( e.g.: Name1,Name2,...,Name_n )</td>
</tr>
<tr>
<td>Type</td>
<td>The schema type of a Resource</td>
</tr>
<tr>
<td>Doc</td>
<td>The description of a Resource</td>
</tr>
<tr>
<td>Ver</td>
<td>The version of a Resource</td>
</tr>
<tr>
<td>MO</td>
<td>The maximal occurrence of a Resource to represent the repeatability</td>
</tr>
<tr>
<td>Def</td>
<td>The definition of a Resource which contains a list of ontology terms that clearly represents concept and the essential nature of a Resource</td>
</tr>
<tr>
<td>Lang</td>
<td>The language of a Resource</td>
</tr>
<tr>
<td>RA</td>
<td>The registry authority of a Resource</td>
</tr>
<tr>
<td>Obl</td>
<td>the obligation of a Resource to indicate if the Resource is required to be presented</td>
</tr>
</tbody>
</table>

Table 1. Ten properties of Resource

3.2 Structure representation

Given a Schema S, the structure of S can be represented by the following two constructors:

- **Tuple constructor (TC).** A tuple constructor TC of a Schema S is an ordered list constructed by the union of S.Incs with the same precedent value in S.Incs.Attrs. For example, the set of S.Incs \((\tau, c_1), (\tau, c_2), \ldots, (\tau, c_n)\) are represented as \(TC_1 = \{c_1, c_2, \ldots, c_n\}\). The subscript \(\tau\) is the name of TC.

- **Set constructor (SC).** A set constructor SC of a Schema S is an occurrence type with the same S.Attrs.MO larger than one. For example, the set of S.Attrs, \(c_1.MO=i, c_2.MO=i \ldots c_n.MO=i\), are represented as \(<c_1, c_2, \ldots, c_n>\_i\). The subscript \(\tau\) represents the name of SC and i represents the maximum occurrence.

For instance, the structure expressions of schemas in two examples of Fig. 1 are:

\{Name\}_Person and

\{Title, <First_Name, Last_Name>\}_Author \_Book

respectively.

3.3 Metadata translation

Given two Schemas \(S_{Source}=(∪ TC_i∪ SC_i)\) and \(S_{Dest}=(∪ TC_i∪ SC_i)\), a Translation Service TS contain a set of translation rules that \(S_{Dest}. TC_1 = op (S_{Source})\). \(op\) represent metadata operations that can manipulate the source metadata into the destination. The \(op\) is supported depending on different schema types. For example, the numeric type attributes can support math operations, like add, subtract, multiple, divide... etc.; the string type attributes can support concatenate and subtract operations to translate source schema into destination. As in Fig. 1, the translation of book authors into person type with comma between different authors can be expressed as:

\(TC_{Person\_Name} = SC_{Author}((SC_{Author\_First\_Name}+\_"\_+SC_{Author\_Last\_Name}\_\"))\).

4 Metadata Architecture

The main activity of DL integration is to search information from DL services, like Webpac, and finally combine the result. The metadata architecture proposed herein facilitates DL search from the perspective of metadata. The architecture contains three layers, metadata extraction layer, semantic query layer and data clustering layer. The metadata extraction layer structuralizes output of each DL service into metadata. The semantic query layer derives relationships between DL services and content to enhance the query semantics. The data clustering layer integrates the result by considering the structure of metadata.

4.1 Metadata extraction

The first step to integrate DLs is to collect output from distributed DL affiliation. However, the easiest way is to collect information from the public Web services (like WebPacs) because there is unnecessary to have prior arrangement with DLs. Interestingly, most of the DL services output result with regular format in every search; that means to analyze one output in advance can obtain the others by following the identical structure. Therefore, the purpose of metadata extraction layer is to automatic extract metadata form output of each DL search services according to the identical structure from commonly structured documents, and to represent this structure information by metadata. By means of this method, DLs access is transparent through HTML protocol and metadata extraction is easy and automatic.

4.1.1 Structure hierarchy
For example, two content C₁, C₂ are "Identical" if they have the same semantics and format, i.e., Identical(C₁, C₂) if C₁.Sche = C₂.Sche and C₁.Semantics = C₂.Semantics. A WebPAC service S can Produce content C with Dublin Core format, i.e., Produce(S, C = "Dublin Core") if C.Sche = S.Outputs and C.Sche = "Dublin Core". To refer the formal definition of each relationship can be found in [9].

### 4.2.2 Manipulation operations

<table>
<thead>
<tr>
<th>^operations</th>
<th>( ^\Sigma ) operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>^( \Sigma ) operations</td>
<td>verify if the given content and service with the specified relationship and return TRUE or FALSE.</td>
</tr>
</tbody>
</table>
| ^\( \Pi \) operations     | derive the content or services conforming to the specified relationship and

A piece of Content, C, is a quadruple \( \{Id, Sche, Presentations, Semantics\} \):
- \( Id \) is the identifier of C.
- \( Sche \) is the schema of C in Definition 3.
- \( Presentations \subseteq Names \) is the set of the presentation interfaces of content C.
- \( Semantics \subseteq Names \) is the set of semantics of content.

A Service, S, is a quadruple \( \{Id, Capabilities, Outputs, Inputs\} \):
- \( Id \) is the identifier of S.
- \( Capabilities \subseteq Names \) is the service capabilities performed.
- \( Outputs \subseteq Sche \) are the output schema of service S.
- \( Inputs \subseteq Sche \) are the input schema accepted by S.

**Definition 4 (Content).** Each (semi-) structured document (like HTML, XML... etc.) can be analyzed into a Structure Hierarchy. A structure hierarchy is a tree where each node represents a critical element (a node with hierarchy information). In structure hierarchy, a critical element connects with its child critical element nodes with level property (if two critical elements are hierarchical) and connects with sibling with parallel property (if two critical elements are in the same level). As shown in Fig. 2, a structure hierarchy is illustrated according to a structure table (define level property and parallel property of the tags).

**4.1.2 Common structure**

A common structure is a structure expression (defined in section 2.2) to extract structured document semantics. For example, the structure expression in Fig. 2 can be expressed as:

\[ \{\text{Title}^{1.1}, \text{First Name}^{1.2.2}, \text{Last Name}^{1.2.3}, \text{Author}^{1.4}\} \]

The structured document produced by the same DL service can be analyzed into structure hierarchy first, and obtain the book title from level 1.1.1 to 1.1. The “First_Name” can be iteratively extracted from level 1.2.2 to 1.2.3 and “Last_Name” from level 1.2.3 to 1.3. Each name of set or tuple constructor is given as the metadata semantics. By following the process, DL services output is extracted easily and transparently into metadata. The detail process can be referred in [8].

**4.2 Semantic query**

In the second layer, we encapsulate two integral aspects of digital libraries into metadata, content and service, and derive relationships to obtain more semantics. Notably, the content and service metadata is administrative metadata, which is different from data metadata.

**4.2.1 Relationship between content and service**

**Table 2. Relationships between content and service**

<table>
<thead>
<tr>
<th>Service to Service</th>
<th>Identical Inclusive Homonymous Synonymous Replaceable Translatable Combinable Combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content to Content</td>
<td>Identical, Homonymous Synonymous InheritFrom Translatable</td>
</tr>
<tr>
<td>Service to Content</td>
<td>Produce ManipulatedBy</td>
</tr>
</tbody>
</table>

| Table 3. Manipulation operations |
|---------------------------------|-----------------------------------------------|
| \( ^\Sigma \) operations | \( ^\Pi \) operations |
| \( ^\Pi \) Translatable \( ^\Pi \) Identifiable \( ^\Pi \) Identical \( ^\Pi \) Homonymous \( ^\Pi \) Synonymous |
| \( ^\Pi \) Identical \( ^\Pi \) Inclusive \( ^\Pi \) Homonymous \( ^\Pi \) Synonymous \( ^\Pi \) Replaceable |

- \( ^\Sigma \) operations: verify if the given content and service with the specified relationship and return TRUE or FALSE.
- \( ^\Pi \) operations: derive the content or services conforming to the specified relationship and...
return the result.

For example, \( \text{\texttt{Sched}(Content\ A, Content\ B)} \) return true if for each \( r \in A.\text{Sches}.\text{Incs} \) \( r=(A, B) \). That mean \( B \)'s schema is derived from \( A \). \( \text{IT}\text{InheritFrom}(Content\ A) \) return all content in \( A.\text{Sches}.\text{Incs} \).

### 4.2.3 Query language
**Example:** Find the services with the service capability “Catalog System” and the output format is “Dublin Core”.

**Query Statement:** \( \text{Select} S.\text{Id From Service} S \text{ where} S.\text{Capabilities} = \text{“Catalog System”} \text{ and} S.\text{Output} = \text{“Dublin Core”} \)

**Algorithm:**

\[
\text{Query}(\text{Attributes} \ A, \text{Services} \ S)\{
\begin{align*}
1 & : \text{For each service} s \text{ in database,} \ S.\text{Id} = S.\text{Id,} \text{ and} \\
& \quad \quad S.\text{Capabilities} \prec \text{Capabilities} \ A.\text{Capabilities} = \text{“Catalog System”}\\
2 & : \text{For each} r \in s, \ c \leftarrow c \cup \prod\text{Translatable}(r, A.\text{Sche} = \text{“Dublin Core”})\\
3 & : \text{return} c;
\end{align*}
\]

The \( \prod\text{Translatable} \) operation can be achieved by adapting translation services in section 2.3. This query returns a set of services (not only single service) that can complete the task. For example, a union catalog system and a translation service that converts MARC format data into Dublin Core could be return together even any single part does not thoroughly conform the query criteria. This query suggests user to obtain more possible result and the librarian can reuse existent service or component to achieve the task. In the semantic query layer, an administrator is only to maintain the ontology table and interoperability can be achieved with minimum user-intervene.

### 4.3 Data clustering

#### 4.3.1 Self-Organizing Map

The data clustering layer integrates structured data (called metadata), like patent document, by means of clustering each sub-part of data structure to improve the accuracy. In this layer, Self-Organizing Map (SOM) is adapted to cluster document with similar structure. SOM provides unsupervised neural network clustering and maps high-dimension data into a low-dimension map. There are five steps to construct SOM (in Fig. 3):

1. Initialize weight vectors of output map.
2. Present input documents in order.
3. Compute the distance between the input document and all nodes in the map and select the closest node as the winner.
4. Update the weights of the winner node and its neighbors.
5. Repeat step 3-4 to all documents and iterate all inputs until convergence. Label the regions of the final map to represent the clustering result.

#### 4.3.2 Structured SOM

Applying SOM in structured domain requires modification of input vector and algorithm. The main concept is to calculate each sub-structure respectively and concatenate the result to the upper structure. In the structured clustering, the first step is to determine the maximum item number (\( MI \)) of tuple and set constructor in metadata schema. The shortage of nodes less than \( MI \) required additional Null nodes. For the example in Fig. 1, if the \( MI = 3 \), the structure expression can be rewrite as:

\[ \{\text{Title,} <\text{First_Name,} \text{Last_Name,} \text{Null}_1 ; '\text{Author}_1' \text{Null}_3 \}_\text{book} \]

The structure expression is assigned into SOM input vector by Directed Acyclic Graph (DAG) sequence, for example, given a three node input vector, the input vector are rewrite as following:

\[
d_\text{book} = (V_\text{book} (x_\text{Title}, y_\text{Title}, (x_\text{Author}_1, y_\text{Author}_1), (x_\text{Null}_1, y_\text{Null}_1)))
\]

Additionally, the distance calculation is updated as following formula:

\[
d = \sqrt{|V_\text{Node}_1 - V_\text{Node}_2|^2 + |V_\text{Node}_2 - V_\text{Node}_3|^2}
\]

where \( |V_\text{Node}_1 - V_\text{Node}_2| \) represents the distance between these two vectors. The modification of distance formula means the cascaded calculation to all connected nodes. The adapting of Structured SOM is also updated as following:

\[
\begin{align*}
\text{w}_{dx,y}(t+1) &= \text{w}_{dx,y}(t) + \eta(t) (x(t) - w_{dx,y}(t)) \\
\text{w}_{dx',y'}(t+1) &= \text{w}_{dx',y'}(t) + \eta(t) (y(t) - w_{dx',y'}(t))
\end{align*}
\]

The structure clustering is essential especially in the aspect of data with structure, like patent documents.
or dissertations with regular structure. Our experiments show that structured clustering has substantial improvement in accuracy. Data with similar structure can be clustered together. Notably, increasing overhead in time must be paid attention owing to the iterative calculating of each sub-part.

5 Conclusion
In this paper, three-layer metadata architecture is proposed to integrate digital libraries by activating the structure information. Metadata extraction layer provides a transparent way without prearrange with library affiliations. Semantic query layer solves semantic heterogeneity. Data clustering cluster documents improve the clustering accuracy. Adapting this architecture reveals that DL integration from the perspective of metadata is useful and save lots of administrative load when constructing a library service, like union catalog system. This metadata architecture has substantial work to extend in the future. The inductive result of semantic query could be various and parts of them are meaningless. A ranking function is required to reduce redundant and provide practical result. In the data clustering layer, some prediction techniques are required to reduce the computing time and cannot lose too much preciseness.

Acknowledgement
The project is sponsored by National Science Council of the Republic of China, Taiwan, Contract No. NSC90-2213-E-009-082.

References: