Learning Object Metadata Interchange Mechanism

A thesis presented in partial fulfillment of

the requirements for the degree of

Master of Information Science

at Massey University, Palmerston North, New Zealand.

Yuejun Zhang

2005

Acknowledgements

I would like to thank all my friends who provided various helps in completion of this thesis.

Special thanks are given to Associate Professor Kinshuk, my supervisor, for all his help throughout this project. Without his patient guidance and valuable advice, it would be impossible to complete this thesis.

Abstract

In spite of the current lack of conceptual clarity in the multiple definitions and uses, the term *learning objects* is still frequently used in content creation and aggregation in the online-learning field. In the mean time, considerable efforts have been initiated in the past few years for the standardization of metadata elements for consistent description of learning objects, so that learning objects can be identified, searched and retrieved effectively and efficiently across multiple contexts. However, there are currently a large number of standardization bodies and an even much larger number of ongoing standard initiatives in the learning field, and different learning objects repositories are likely to apply different metadata schemas to meet the specific needs of their intended communities. An interchange mechanism for the conversion between various metadata schemas, therefore, becomes necessary for intensive interoperability.

In this thesis, we first make a brief introduction to the concept *learning objects*, then the term *metadata*, followed by a description of the functional requirements of learning objects, the purposes of metadata, and the importance of metadata for learning objects. After that, this thesis investigates metadata schemas in various fields in general, focused on several mainstream metadata specifications developed for learning objects in particular. The differences among these metadata schemas for learning objects are analyzed and a mapping between their elements is identified. On the basis of literature review, a framework for interchange of metadata schemas is proposed and a prototype to demonstrate the functionalities of the framework is developed. For the high scalability and the high accuracy of the developed system, a so-called LOM-intermediated approach is suggested, and a so-called dynamic-database methodology is adopted. The LOM-intermediated approach significantly simplifies the metadata mapping issues by undertaking the schema-schema mapping in a way of schema-LOM-schema mapping, while the dynamic-database methodology effectively prevents any data-loss resulting as a by-product from the use of LOM-intermediated approach. The prototype currently generates and outputs XML metadata in IMS, EdNA, Dublin Core and LOM. It is a web-based three-tier architecture, using Java technologies for implementation, MySQL as the database server and JDBC for database access.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 2 LEARNING OBJECTS AND METADATA

CHAPTER 3 METADATA SCHEMAS

CHAPTER 4 METADATA INTERCHANGE AND PROTOTYPE DESIGN

CHAPTER 5 PROTOTYPE IMPLEMENTATION AND EVALUATION

CHAPTER 6 CONCLUSION AND FUTURE WORK

LIST OF TABLES

CHAPTER 2 LEARNING OBJECTS AND METADATA

CHAPTER 3 METADATA SCHEMAS

CHAPTER 4 METADATA INTERCHANGE AND PROTOTYPE DESIGN

CHAPTER 5 PROTOTYPE IMPLEMENTATION AND EVALUATION

Chapter 1 Background and Project Outline

1.1 Background

Learning objects, together with the standardized metadata for describing them, present an attractive prospect that independent and self-standing learning resources, once created, can be located, retrieved, and used anywhere at any time to construct individualized learning content. This way, learners benefit because highly adaptive and just-in-time learning becomes possible; courseware providers benefit as well because the reusability and interoperability offered by the learning object approach result in tremendous cost-down. Although researchers are still grappling with the problem of a definition commonly accepted for the term "learning object", basically, learning objects are conceptually small content components that can be used and reused multiple times in different contexts, and can be assembled together in many ways to construct various courses, lessons or programs. Learning objects cannot exist alone, however, only when combined with metadata can learning objects be effectively and efficiently located, retrieved and eventually reused. Literally meant "data about data", metadata is a set of pre-defined elements or attributes along with their values that describe various features or characteristics of a learning object. Metadata consistency relies on standardization, and the metadata element set as well as its use is defined or controlled by a variety of metadata schemas developed by various organizations or initiatives.

Over the past few years, there has been a worldwide interest in the concept of reusable learning resources (Mohan, 2004), and a large number of international efforts have been initiated on the definition of educational metadata specifications for the commonly agreed description of learning resources. As a result, more and more learning object repositories that store learning objects and their metadata in an organized way have been built by various communities. While on the other hand, it is difficult for a single metadata specification to satisfy the functional requirements of all applications, as each community has different specific needs. This situation leads to the fact that hundreds or even thousands of metadata schemas, including various application profiles, developed by a variety of organizations and initiatives, exist today. At the same time, different learning object repositories are likely to employ a different metadata schema, or the same schema but in different version or in different natural language, to describe their learning

resources. Consequently, the issue of metadata interoperability arises. An attractive illusion is that all these metadata schemas can communicate with each other so that an extensive interoperability across the global Internet could be possible. To reach this global interoperability, two possible ways may be considered (Sampson, 2004): either the enforcement of the use of an internationally recognized standard or a conversion among varying metadata schemas. The heavy cost occurring in the implementation of an international standard makes the first option nearly impossible, so we believe that an interchange mechanism would be a promising approach to tackle the problem of metadata interoperability.

Unfortunately, a flexible, general-purpose means that facilitates conversions among multiple metadata schemas is currently unavailable, although mapping tools that can convert from one schema to a limited range of other schemas do exist. The goal of this project is to propose a framework for the interchange among multiple metadata schemas, based on the investigation of various metadata schemas. Further more, a prototype system is implemented to demonstrate the functionalities of the proposed framework.

1.2 Project Outline

1.2.1 Scalable, General-purpose System

Unlike the currently available metadata conversion tools that mostly translate a certain schema to a limited number of other schemas, we intend to propose a general-purpose framework for metadata interchange. The system built under this framework is powerful enough to accommodate a large number of metadata schemas and be able to convert from any schema to any other schemas as long as these schemas are accommodated in the system. In addition, the system is scalable enough to be easily extended so that a new schema can be added into the system without the needs to make many changes to the current system. To build such a metadata interchange system, the main barrier is the huge number of mappings between all the schemas that are to be accommodated in the system; this number would increase exponentially as the number of schemas grows. To add into the system a new schema, mappings from the new schema to all the existing schemas, as well as from all the existing schemas to the new schema, must be identified and coded into the system. So scalability becomes a big issue. A so-called LOM-intermediated

approach accompanied by the dynamic-database methodology is proposed and adopted to tackle this issue.

1.2.2 LOM-intermediated Approach

Instead of mapping the source schema directly to the target schema, conceptually source schema is mapped to the IEEE Standard for Learning Object Metadata (LOM) at the first step, and then to the target schema from the LOM model. In this process, the LOM data model acts as an intermediary between the source schema and the target schema. By using the LOM-intermediated approach, for every schema accommodated in the interchange system, we only need to identify the mapping between this schema and the LOM standard, while mappings between this schema and all other schemas are no longer necessary. This results in a vastly reduced total number of mappings needed for the system. Every time when a new schema is to be added to the system, only the mapping to LOM, rather than mappings to all the existing schemas, should be identified and coded into the application, thus system scalability is greatly enhanced. In addition, mapping each schema to LOM is much easier and less error-prone than mapping each schema to all other schemas, and LOM being the only accredited learning object metadata standard at the moment, it is reasonable to assume that each schema provides as a part of itself the mapping to LOM, which would be accurate and widely accepted. So the LOM-intermediated approach also greatly improves the situation where an accurate, commonly agreed mapping is often difficult because of the intrinsic issues on metadata crosswalks (see Chapter 4 section 4.2.1).

1.2.3 Dynamic-database Methodology

In practice, we implement the LOM-intermediated approach by building a LOM-based database. This means that the database tables and data fields within each table are designed around the LOM data model. An input record, no matter where it was from and what metadata schema it originally was conformed to, is transformed to a data record fit to the LOM-based database, and then stored in the database. However, data loss might happen during this process of converting the source schema to the intermediary LOM, although LOM is intended to cover most aspects of learning objects. This data loss is a by-product of the use of the LOM-intermediated approach thus needs to be completely avoided. To solve this problem, a methodology called dynamic-database is employed in the project. For those few elements that are specified in a schema but are not covered by

LOM, data tables extra to the basic LOM-based database are dynamically created by the program at the first time when such a schema is encountered during the execution of the system. Once the extra tables have been created for a schema, they may be used even in the future for retaining the extra elements data in all records in that schema. In the developed prototype, for example, the Education Network Australia (EdNA) Metadata Standard contains four elements that are unavailable in LOM, so the first time when the system deals with an EdNA record, the table Edna will be automatically created into the database.

1.2.4 System Functionalities

As an implementation example of the proposed framework, the developed prototype currently accommodates LOM, the Instructional Management System (IMS) metadata, the Dublin Core Metadata Element Set (DCMES), and EdNA. It can either convert a database record into any of the four metadata-schemas, or transform a record in any one of the four schemas into all the other three schemas. The output metadata from the system is in eXtensible Markup Language (XML) format, as it is believed that XML is becoming the de facto standard for information delivery on the Internet.

Functionally the prototype system consists of four parts: a metadata editor, a data storage and management device, a metadata interchange mechanism, and an output unit. The metadata editor accepts three types of data input – the ordinary manual input by keyboard type-in, the XML metadata input by uploading the source XML document to the system, and the Web resource input by providing the system the Universal Resource Locator (URL) of the Web resource, from which the system can automatically harvest as much metadata as possible. The data storage and management facilitates metadata storage and user information management, using the MySQL database server. Basically, it provides a LOM-based database to store all the metadata records that have been input into the system and have been converted to the LOM format. The metadata interchange mechanism, the core of the system, is responsible for both the conversion from the source input to LOM, and the conversion from LOM to the desired schema. The output unit simply enables the presentation of the output XML files.

1.2.5 System Architecture and Technologies

The system is a Web based application, having a typical three-tiered architecture. The client tier provides the interface for the interaction between the user and the system. The web tier manages all the business logics in the system, carrying out metadata mapping and conversion. The database tier provides a MySQL database sever to store and manage the metadata records.

Technologically, the system is built around two fundamental technologies: XML for interoperable, platform-independent data, and Java for portable, platform-independent software environment. XML and the Java technology are ideally matched with each other and greatly enhanced by each other in the implementation of the system. Especially, the new Java technology, Java 2 Platform, Enterprise Edition (J2EE), defines the standard to ease the development, deployment and management of Java-based Web applications. J2EE includes Application Programming Interface (API) packages, such as the Java DataBase Connectivity (JDBC) API and the Java API for XML Processing (JAXP). J2EE also includes components technologies such as Servlets, Java Server Pages (JSPs) and Enterprise Java Beans. These J2EE packages and components technologies have been used in the implementation of our prototype. In detail, we use the JDBC API as the access interface to the MySQL database, the JAXP API as the means to parse, validate and transform XML documents, Servlets and JSPs as the server side programming language to generate dynamic content sent to the client, and JavaBeans as an enhancement for JSPs and Servlets to encapsulate more complex control logic. These Java components collaborate with each other to undertake various tasks required for the system functionalities.

1.3 Outline of This Thesis

In chapter 2, we introduce learning objects and metadata from a variety of perspectives, and discuss the relationships between a learning object and the metadata associated with it. We come into the conclusion that the success of the learning object approach relies on interoperable metadata, and point out the importance of metadata standardization. This is naturally followed by an investigation of metadata standardization in chapter 3. In chapter 3, we make a survey on metadata schemas across multiple industries but with the focus on the learning sector. We conclude that hundreds or thousands of metadata schemas exist today, and different metadata schemas might be used in different systems to meet the

specific needs, therefore the communication between heterogeneous systems becomes an issue. This situation results in the necessity of metadata interchange for an intensive interoperability. In chapter 4, we discuss the fundamentals of metadata interchange – crosswalks and the critical issues. Then we have a brief overview on the current status about metadata conversion and mapping tools. We point out that an effective means for the transformation between multiple metadata schemas is still unavailable, and therefore propose a framework under which a general-purpose, scalable metadata interchange system is possible. We further address the design issues for such a system in order to implement a prototype in this project. Following the discussion of prototype design, chapter 5 discusses the implementation aspects of the prototype. First the used technologies are described; this is followed by a presentation of the Java components implemented in the application and correlations among these components. We then introduce some of the components in more detail, and discuss the deployment of the application in the Tomcat server. Then, we evaluate the developed system by showing the test results in this chapter. Finally, in chapter 6, we conclude the work undertaken in this project, and point out the possible future developments.

Chapter 2 Learning Objects and Metadata

2.1 Learning Objects

Originated in the object-oriented paradigm of computer science, the term "learning objects" has become the Holy Grail of content creation and aggregation in the computer-mediated learning field (Polsani, 2003). Although having a number of synonyms, multiple definitions and diverse uses among different individuals and organizations, learning objects are conceptually based on the fundamental idea of small instructional components that once created can be reused a number of times in different learning contexts. This concept of learning objects can be explained by a LEGO metaphor (Hodgins, 2000b), or alternatively and probably more accurately, by Wiley's atom metaphor (Wiley, 2000).

Learning objects envision such a world where distributed network of learning systems will be able to deliver exactly the right content, at the right time, in the right amount of detail, via the right device, to the right place, in the right way to meet the specific needs of individuals. In this learning world, content developers will create learning objects that are tagged with standardized metadata to enable easy search, retrieval and reuse. These learning objects will be stored in a distributed network of industry-standard compliant repositories that are accessible through all kinds of media devices. Publishers, teachers and other courseware providers will flexibly use industry-standard tools to assemble learning objects into courses, lessons and any other collections of knowledge, which in turn can be made available to their intended audiences through the distributed repositories. Users will subsequently use the same industry standard to access these learning materials in a fully individualized way. Once a critical mass of reusable learning objects is reached, a worldwide learning economy will emerge.

2.1.1 Definition of Learning Objects

2.1.1.1 Multiple Definitions

Then, what exactly are learning objects? This is one of the debates among researchers involved in the learning community. In fact, although learning objects are conceptually appealing, what constitutes a learning object in practice has been unclear, even standardization bodies such as IEEE, IMS and the Sharable Content Object Reference Model (SCORM) remain vague on the inside constitution of a learning object (Vossen & Jaeschke, 2003). Historically, the term "learning objects" was popularized by W. Hodgins

in 1994 when he named a CEdMa working group LALO, "Learning Architectures and Learning Objects". It was then adopted by many organizations including the IEEE Learning Technology Standards Committee (LTSC). These organizations together with other individuals, however, offer the same term with different definitions.

According to the IEEE draft standard for LOM (IEEE LTSC, 2002), a learning object is defined as "any entity -digital or non-digital- that may be used for learning, education or training."

Wiley (2000) argues that the IEEE LTSC definition is too broad to be useful, and suggests a more refined definition as "any digital resource that can be reused to support learning." This definition explicitly excludes non-digital and non-reusable objects from the definition offered by IEEE LTSC working group, but includes anything that can be delivered across the network on demand, be it large or small.

Shepherd (2000) gives another definition: "A learning object is a small, reusable digital component that can be selectively applied - alone or in combination - by computer software, learning facilitators or learners themselves, to meet individual needs for learning or performance support." Similar to Wiley's definition, Shepherd emphasizes digital objects and reusability.

Many other definitions of the term "learning objects" still can be found in literatures (Polsani, 2003; Wiley, 2000; Friesen, 2003; Wagner, 2002). On the basis of a wide survey and examination of current definitions for learning objects, McGreal (2004) points out that there are four general types of meaning that can be discerned in these definitions, from very generic to very specific, from 1) anything and everything, through 2) anything digital, to 3) only objects that have an ostensible learning purpose, to 4) those that support learning only in a particular or specific context. This assortment of what can be considered to be a learning object can be graphically presented in figure 2.1.

McGreal (2004) himself argues for a definition to be developed with focus on digital objects that have a stated educational purpose and are marked for specific educational purposes. Therefore he defines learning objects as "any reusable digital resource that is encapsulated in a lesson or assemblage of lessons grouped in units, modules, courses, and

even programs." While a lesson can be defined as "a piece of instruction, normally including a learning purpose or purposes."

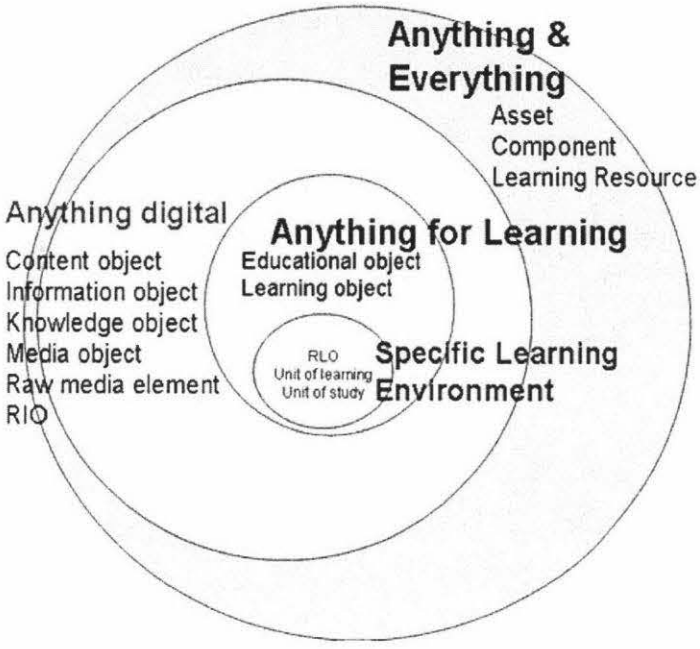


Figure 2.1 Terminology for learning objects (McGreal, 2004)

In addition to the proliferation of definitions, there exist quite a number of synonyms of the term, which imply the same general intention to take an object-oriented approach to computer-supported learning. Examples of these synonyms include "knowledge objects", "components of instruction", "pedagogical documents", "educational software components", "online learning materials", or simply "resources" (Wiley, 2000) and so on.

2.1.1.2 Common Concepts on Learning Objects

However defined and whatever called, Learning objects are actually intended to be an application of object-orientated thinking to the world of learning. Similar to LEGO bricks, learning objects are small reusable components - video demonstrations, tutorials, procedures, stories, assessments, simulations, case studies - that can be combined in nearly infinite ways to construct collections able to be called lessons, modules, courses, or even curricula. Just as Wiley (2000) pointed out, "the main idea of learning objects is to break educational content down into small chunks that can be reused in various learning environments, in the spirit of object-oriented programming." These chunks can be independently created and maintained, and pulled apart and stuck together. Or even further, "we need to stop thinking of learning objects as chunks of instructional content

and to start thinking of them as small, self-reliant computer programs" (Downes, 2002) so that learning objects can be built smart enough to perform basic self-analytical tasks, scan the web for web services and learn about their new environment. Wagner (2002) and Robson (2001) summarized from the prevailing views the main attributes and traits of learning objects.

1. They are the smallest elements of stand-alone information required for an individual to achieve an enabling performance objective or outcome.
2. They are stored and accessed using metadata attributes and tags.
3. They are assembled and contextualized using metafiles that situate meaning and application and facilitate meaningful assembly.
4. Each learning object must be able to communicate with learning systems using a standardized method that does not depend on the system.
5. What happens within a learning object is the learning object's business.
6. How a learner moves between learning objects is controlled by the learning system.

2.1.2 Capabilities of Learning Objects

Although a commonly acceptable definition of learning objects has not emerged and might never emerge, there is a broad understanding among the community members about its capabilities – or functional requirements or design goals as called by some researchers. These capabilities are well addressed in literature (Polsani, 2003; Learning Objects Network, 2001; Metros & Bennett, 2002), and summarized here in Table 2.1.

Among these capabilities, reusability and interoperability are especially imperative to the sustainability of learning objects. Without them, any one with a significant investment in learning content will eventually find them locked into that particular content. Without them, every time a learning content needs to be updated, far more of the material must be recreated than is necessary or desirable. Without them, the process of developing high-quality learning content is prone to unnecessary duplication of efforts.

Table 2.1 Learning objects functional requirements

Goal	Description
Reusability	The flexibility to reuse content objects for multiple purposes, in different applications, in different products, via varying access devices, for numerous markets.
Interoperability	The ability to use content developed by one organization on a given platform with one set of tools at a completely different organization on a different platform with another set of tools.
Durability	The ability to transcend technology (platforms, tools, access devices, products, vendors) changes without requiring redesign or redevelopment.
Accessibility	The ability to search, identify, access, and retrieve content that is widely distributed.

2.1.3 Pros and Cons of Learning Object Approach

Undoubtedly, associating these capabilities to learning content makes content design and creation more expensive both in terms of time and money. So, why would developers wish to add complexities to their work by including object capabilities in their design? This is because the learning objects approach allows the learning content to gain a "value-add" that in most cases may pay off many times over, similarly both in terms of time and money. Learning objects make it no longer necessary to have thousands of the same educational material online, and permit lessons to be generated and customized for specific groups or even individuals.

In fact, by building learning content as reusable learning objects, developers of learning content, learning administrators and learners themselves all benefit (Shepherd, 2000). For developers, the reusability of learning objects and availability of various authoring tools save them large amount of time, effort and money. For administrators, learning objects approach allows for course customization and efficient course construction as well as convenient system administration. For learners, they will be able to access a highly personalized learning on an anywhere, just-in-time and just-enough basis.

Robson (2001) summarizes the Pros and Cons of learning object approach from somewhat different perspective, with respect to production cost, flexibility, pedagogy, end user cost and industry support. Table 2.2 summarizes Robson's findings.

Table 2.2 Pros and cons of learning object approach (Robson, 2001)

	PROS	CONS
Production Costs	By properly breaking content into learning objects, different parts can be maintained and updated separately. If a suitable learning object can be found, a new one does not need to be created. These are costs savers.	Changing to a learning object approach from a "self-contained system" approach involves retooling and retraining costs.
Flexibility	As more and more standards-based learning objects become available, increased choice will translate into more flexibility for designers.	Using standards-based learning objects restricts the scope of learner information that is accessible by content if total interoperability is maintained.
Pedagogy	Learning objects fit nicely into many ISD theories. Instructional templates can be created with slots for specific types of learning objects. Learning objects may encourage designers to operate in more disciplined ways with a positive effect.	Restrictions on learner information available could restrict pedagogical approaches. Approaches using lengthy discursive material may not benefit from the use of learning objects.
End User Cost	The learning object approach prevents consumers from being locked in to specific systems. As standards take hold, the market for content will take on more of the properties of a typical consumer market with lower costs and increased choice.	The cost of converting existing content to a learning object approach may be significant
Industry Support	All leading system vendors and content producers are supporting SCORM and other standards that are based on or that complement a learning object approach.	Realistically, it is twelve to eighteen months between the time the vendor community adopts an approach and the time products that implement the approach are available.

2.1.4 Types of Learning Objects

Now that there is not a unique definition commonly accepted for the term "learning objects", a look at what types of learning objects we could have probably helps in understanding the concept behind this term. However, different criteria by which to classify the learning objects might lead to distinct taxonomies. Shepherd (2000) groups learning objects into three categories according to the variety of purposes learning objects can serve, arguing that learning objects may either provide a fully self-contained mini-tutorial, or comprise the elements in a more extended learning cycle – overviews, case

studies, simulations, assessments and so on. Table 2.3 shows these three categories identified by Shepherd.

Table 2.3 Types of learning objects (Shepherd, 2000)

Integrated	Informational	Practice
Mini-tutorials	Overviews / summaries	Problems / case studies
Mini case studies, simulations, etc. with supportive information	Descriptions / definitions	Games / simulations
	Demonstrations / models	Drill-and-practice exercises
	Worked examples	Review exercises
	Cases / stories	Tests / assessments
	Papers / articles	
	Decision aids	

Wiley (2000) creates another taxonomy, which categorizes learning objects into five types, in terms of the difference in the manner in which learning objects exhibit their characteristics. The five types include "Fundamental", "Combined-closed", "Combined-open", "Generative-presentation" and "Generative-instructional". The characteristics used by Wiley as criteria for his categorization are all critical attributes of learning objects and are stable across environmentally disparate instances. They include "Number of elements combined", "Type of objects contained", "Reusable component objects", "Common function", "Extra-object dependence", "Type of logic contained in object", "Potential for inter-contextual reuse" and "Potential for intra-contextual reuse". Table 2.4 presents the taxonomy, while a more in-depth discussion of each of the characteristics of learning objects and a discussion of the learning object types themselves can be referenced in the work of Wiley (2000).

Table 2.4 Types of learning objects (Wiley, 2000)

		Fundamental Learning Object	Combined-closed Learning Object	Combined-open Learning Object	Generative-presentation Learning Object	Generative-instructional Learning Object
Learning Object Characteristic	*Number of elements combined*	One	Few	Many	Few - Many	Few - Many
	Type of objects contained	Single	Single, Combined-closed	All	Single, Combined-closed	Single, Combined-closed, Generative-presentation
	Reusable component objects	(Not applicable)	No	Yes	Yes / No	Yes / No
	Common function	Exhibit, display	Pre-designed instruction or practice	Pre-designed instruction and / or practice	Exhibit, display	Computer-generated instruction and / or practice
	Extra-object dependence	No	No	Yes	Yes / No	Yes
	Type of logic contained in object	(Not applicable)	None, or answer sheet-based item scoring	None, or domain-specific instructional and assessment strategies	Domain-specific presentation strategies	Domain-independent presentation, instructional, and assessment strategies
	Potential for inter-contextual reuse	High	Medium	Low	High	High
	Potential for intra-contextual reuse	Low	Low	Medium	High	High

2.1.5 Other Issues about Learning Objects

As a booming research topic in the area of computer-mediated learning, the idea of learning objects has brought about several discussion issues among the members of the industry. One of these is the obvious struggle and seeming futility in trying to nail down a precise definition for learning objects (Hodgins, 2000b), as was addressed formerly. What exactly are learning objects? There are nearly as many answers as the number of people being asked with this question. In fact, the definition itself is not so crucial. Some researchers argue that rather than giving a definition it is more important and productive to try to find a model for understanding learning objects, while others echo this viewpoint by suggesting that what is needed is not a definition but a principled base model (Boyle, 2003).

Closely related to the issue on learning objects definition is another issue about granularity. How big should a learning object be? The answer to this question is crucial to the success in the reusability of learning objects. Unfortunately, no unique answer exists, and IEEE LTSC's definition leaves room for an entire curriculum to be viewed as a learning object (Wiley, 2000). At one extreme, learning objects can be regarded at a micro level as media assets such as images and paragraphs of text. At the other extreme, learning objects can also be regarded as a full-contained piece of instruction including information, mechanisms for practice and a means of assessment (Shepherd, 2000). One of the most absurd statements about learning objects is "as small as a grain of sand, as large as an ocean". Wisconsin Online Resource Center determines learning object size on the basis of learning time ranging from 2 minutes to 15 minutes, but this determination is regarded quite subjective and arbitrary by Polsani (2003), according to whom, a learning object should ideally include only one or few related ideas. If a learning object consists of more than one idea, there should be only one main idea and others should be derived from the main idea or be dependent on it. Wiley (2000) points out that the decision regarding learning object granularity can be viewed as a trade-off between the benefits of reuse and the expense of cataloging from an "efficiency" point of view, or viewed as a problem of "scope". He identifies a direct inverse relationship between size and reusability, and argues for small content objects as they enable maximum reusability. Similarly, Duval and Hodgins (2003) identify another trade-off between reusability and added value of learning objects in consideration of granularity. They argue that the smaller the object, the

less the added value while the greater the reusability. Hodgins also suggests that there is no set absolute size to a learning object, since the size of the object will be relative to the needs of learners and the requirements of given learning tasks (Wagner, 2002).

Yet another issue is whether it is necessary to build pedagogical process into learning objects (Boyle, 2003). This issue is crucial to clarifying the concept of learning objects, and to a very large degree dependent on the two issues described above. One position is that learning objects should be as small as possible, as small content objects with little or no intrinsic pedagogical process support the widest possible reuse. The opposite, however, believes that there will be no learning without pedagogical process built in content objects. The latter's argument is that this leads to "information objects" not "learning objects".

2.2 Learning Object Metadata

In the above introduction to learning objects, another vital aspect of learning objects, metadata, was deliberately left unmentioned. This is because metadata is so important to learning objects, as important as is the object content itself, that we intend to start the discussions about it with an entirely new, special section.

In fact, as well as the object content, metadata is also one of the two requisite components of a learning object. To achieve the functional requirements asserted in section 2.1.2, each learning object must be tagged with metadata. However potentially powerful it is, a so-called learning object would be useless without consistently tagged metadata to describe it, identify it, locate it, retrieve it and ultimately use and reuse it. On the other hand, the value of a learning object goes up as its associated metadata increases in richness and completeness. A widely disseminated graphic presentation of the anatomy of learning objects, as shown in figure 2.2, illustrates the role metadata play in the idea of learning objects.

In addition, learning objects cannot exist alone. Learning objects, metadata and a learning content management system (LCMS) are three interdependent components. Only through coordinated repository architectures and metadata technologies can the vision of reusable learning objects be made accessible. The relationship between the three components is illustrated in figure 2.3.

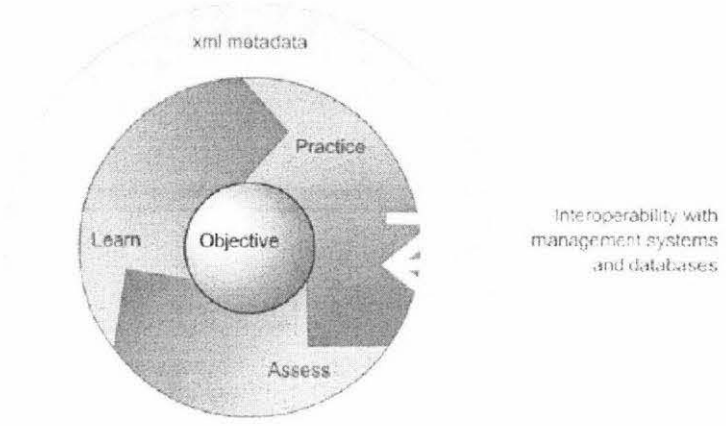


Figure 2.2 Anatomy of a learning object (Wagner, 2002)

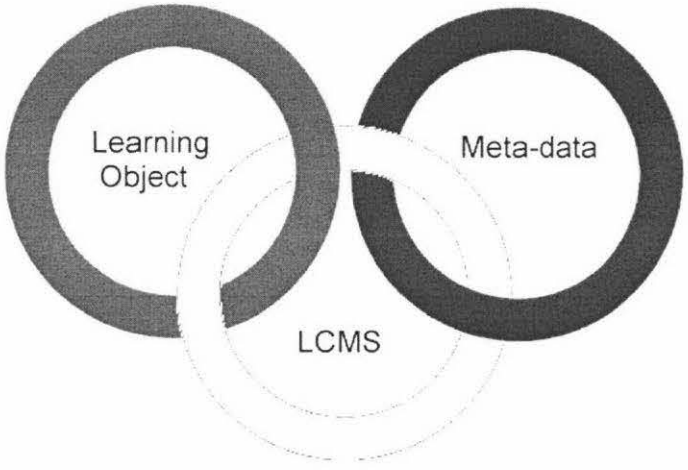


Figure 2.3 The relationship between learning objects, metadata and LCMS (Mortimer, 2002)

Figure 2.4 demonstrates the scenario of information retrieval in a typical leaning object repository, which is composed of not only learning objects distributed across the internet, but also metadata in a metadata repository for these learning objects. Here learning objects refer to digital educational resource; metadata refer to their systematic description that facilitates searching and administration; while repository represents online, searchable collections of educational resources. A little more detailed introduction to learning object repository will be given in a later section. Unlike a common search engine on the web, in this scenario the starting point for information retrieval is the standard-based learning objects metadata, so that a precise search result is guaranteed. It is metadata that makes it possible to select and integrate relevant learning experiences from a relatively small library of learning objects.

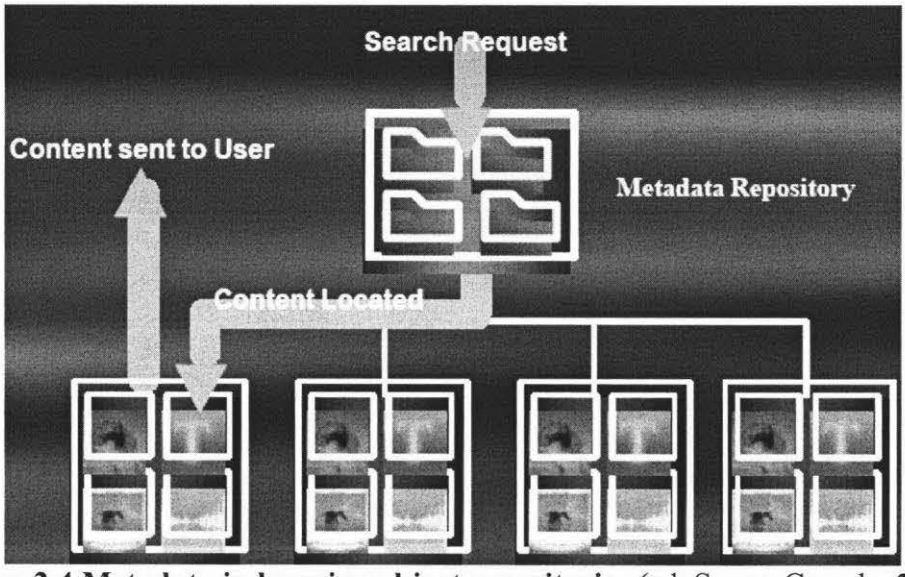


Figure 2.4 Metadata in learning object repositories (eduSorrce Canada, **2003)**

2.2.1 What Is Metadata?

Metadata, according to the IEEE draft standard for LOM (IEEE LTSC, 2002), "is information about object, be it physical or digital". Literally, metadata has the meaning of "data about data", by which, however, the information about non-data object is implicitly excluded from the definition offered by IEEE. For example, a can of sliced pineapple on the supermarket shelf is not a data object, so the data printed on the can label is merely data, strictly not metadata.

Quite similar to the case of learning object definition, metadata is also defined in different ways by people in different industries, with different focuses and from different perspectives. Metadata is the descriptive language for a data resource, sharing many similar characteristics to the cataloging that takes place in libraries, museums and archives. Metadata also differs from catalog data in that the location information is held within the metadata record in such a way that direct data delivery from appropriate application software is possible (Heery, 1996). In learning object paradigm, metadata potentially includes information about the title, author, version number, creation date, technical requirements and educational context and intent. Considering the much wider range of situations in which the definition should be applied while making sensible distinctions from the too broad set of information about objects, we follow the definition offered by Haynes (2004):

Metadata is data that describes the content, format or attributes of a data record or information resource. It can be used to describe highly structured resources or unstructured information such as text documents. Metadata can be applied to description of: electronic resources; digital data (including digital images); and to printed documents such as books, journals and reports. Metadata can be imbedded within the information resource (as is often the case with web resources) or it can be held separately in a database.

A metadata record consists of a number of pre-defined elements representing specific attributes of the resource, and each element may have one or more values. For the purpose of consistency, these pre-defined elements are decided in an element set by various metadata schemas. Examples of metadata schemas are Dublin Core in web community and the IMS Metadata Specification for educational purpose. Each metadata schema usually has these characteristics: a limited number of elements, the name of each element and the meaning of each element. We will discuss metadata schemas in detail in next chapter.

2.2.1.1 A Metadata Example

Figure 2.5 shows a piece of general information about a document titled as "Metadata Watch Report" (SCHEMAS Project, 2002). Such text information is usually on one of the first several pages within a document, and might be so familiar to most of us that we often take it for granted. However, it is this paragraph of text that describes various attributes and characteristics of the document, so the information shown here is indeed functioning as metadata for the described document.

As will be discussed later in the thesis, metadata can be presented in various formats such as XML, Hyper Text Markup Language (HTML) and so on, and conformed to variety of currently available metadata schemas like IMS Specification and EdNA Metadata Standard. Figure 2.6 illustrates the same metadata information in the Resource Description Framework (RDF) format, compliant to the Dublin Core Metadata Initiative (DCMI) standard.

Title	Metadata Watch Report #8 and Standards Framework Report #4
Creator	Makx Dekkers
Contributor	Laurie Causton, Michael Day, Erik Duval, Annemieke de Jong
Subject-Keywords	Deliverable D29/D35; WP2/WP3; Metadata Watch Report #8; Standards Framework Report #4
Description	This document comprises Metadata Watch Report #8 and Standards Framework Report #4 and domain reports for the Audiovisual, Cultural Heritage, Educational and Publishing domains
Publisher	PricewaterhouseCoopers
Date	31 January 2002
Type	Text Manuscript
Format	application/MSWord 2000
Identifier-	
Document Number	SCHEMAS-PwC-WP2/WP3-D29/D35-Final-20020131
Language	English
Rights	European Commission: External distribution via SCHEMAS Web site

Figure 2.5 An example of document metadata (SCHEMAS Project, 2002)

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="en"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:smes="http://www.schemas-forum.org/registry/schemas/SCHEMAS/1.0/smes#">
<rdf:RDF rdf:resource = " ">
<dc:title> Metadata Watch Report #8 and Standards Framework Report #4
</dc:title>
<dc:creator> Makx Dekkers </dc:creator>
<dc:contributor> Laurie Causton </dc:contributor>
<dc:contributor> Michael Day </dc:contributor>
<dc:contributor> Erik Duval </dc:contributor>
<dc:contributor> Annemieke de Jong </dc:contributor>
<dc:subject> Deliverable D29/D35 </dc:subject>
<dc:subject> WP2/WP3 </dc:subject>
<dc:subject> Metadata Watch Report #8 </dc:subject>
<dc:subject> Standards Framework Report #4 </dc:subject>
<dc:subject> Metadata activity reports </dc:subject>
<dc:description> This report comprises Metadata Watch Report #5 and Standards
Framework Report #4 and domain reports for the Audiovisual, Cultural Heritage,
Educational and Publishing domains </dc:description>
<dc:publisher> PricewaterhouseCoopers </dc:publisher>
<dc:date> 2002-01-31 </dc:date>
<dc:type> Text </dc:type>
<dc:format> application/MSWord </dc:format>
<dc:identifier> SCHEMAS-PwC-WP2/WP3-D29/D35-Final-20020131 </dc:identifier>
<dc:language> en </dc:language>
<dc:rights> European Commission; External distribution via SCHEMAS Web site
</dc:rights>
</rdf:RDF>
```

Figure 2.6 The example metadata in Dublin Core (SCHEMAS Project, 2002)

2.2.2 Problems of Normal Search Technologies

We all are familiar with search engines and frequently use them to find needed information across multiple websites. They do help. Then why do we still need metadata approach for information discovery and data retrieval? This is because the problems with the use of current search engines are also obvious to us: "high recall" and "low precision". "High recall", also known as information overload, means the experience of receiving thousands of hits after submitting a search query, while "low precision" refers to not being able to locate the most useful information. These problems are actually intrinsic to current search engines, as search engine suppliers never regard high hit rate as a problem. On the contrary, they evaluate their products on the basis of web coverage, rather than the precision of search results. Many researchers such as Gill (2000) have addressed in detail the problems that traditional search technology presents to users. These problems are summarized as follows.

1. Relevant information can be missed because sites contain types of resource in addition to HTML text while multimedia or interactive content is not directly searchable;
2. Information on the web is increasingly being generated dynamically from databases in response to user input. This information is beyond the indexing reach of the web crawlers;
3. The search engines frequently do not harvest every page on a site, but often only the top two or three hierarchical levels, thus missing significant documents which, on larger and more complex sites, may be located in lower levels of the hierarchy;
4. The web crawling components of the search engines are fully automated, which means that the indexed web resources are selected by software algorithms rather than people, and are therefore variable in both quality and depth of indexing;
5. Search engines, especially the more comprehensive ones, may index sites on an infrequent basis and may therefore not contain the most current data;
6. Irrelevant information can be retrieved because the search engine has no means (or very few means) of distinguishing between important and incidental words in the document text. As a result, tens or hundreds of thousands of

"matching documents" are retrieved in response to almost any search string; and

7. As the volume of information on the web continues to increase exponentially, the amount of network bandwidth required by the crawlers in order to maintain current and comprehensive indices could eventually reach unacceptable level.

Metadata seems to be an effective solution to these problems.

2.2.3 Advantages and Disadvantages of Metadata

Metadata is not new. Long before the emergence of the learning object model in the learning community, metadata had been widely popularized and utilized in many other information arenas such as library science, publishing industry and web community. In fact, metadata and metadata standards lie behind almost all technological systems involving display and exchange of information (Brody, 2003). One important reason for the popularity of metadata is simply that metadata is becoming the foundation of all information retrieval. To be faced with a large collection of information while unable to find the needed information we know exists somewhere within the collection is a problem as old as the existence of the information collection. As the ocean of information rises and leaves what we need ever more deeply buried in what we do not need, this problem becomes serious. While traditional search engines and indexes are inadequate for locating information due to the lack of the detail and precision to deal with large number of resources and discipline-dependent terminology, metadata is an effective, if not the unique, solution to this problem. In learning paradigm, a universal metadata system could form the backbone for an enormous, searchable repository of learning objects – a repository like the World Wide Web, but one on which finding the particular object you want would be far easier and more efficient than conducting a normal web search on the Internet (Gordon, 2002). We have conceptually illustrated such a learning objects repository in figure 2.4.

Other than assistance in information retrieval, metadata can do much more. According to EdNA Metadata Standard website, "The creation of metadata to describe resources assists in resource discovery and resource management. Consistent cataloguing of online resources means maximized opportunities for searchers to find the most relevant and

comprehensive set of resources for their purposes. Metadata can also be used to organize, store and retrieve items for information management purposes." Hayness (2004) summarizes the main benefits of metadata as follows:

1. Metadata enables resource description and enhances retrieval performance.
2. Metadata provides a way of managing information resources.
3. Metadata helps document ownership and determine authenticity of data.
4. Metadata is the key to reusability and interoperability.

An obvious disadvantage of metadata approach to information discovery is the needs for human effort and the cost to index and describe resources. In addition to the hardware investment, metadata input is a tedious and time-consuming task, and this has been proved to be a main obstacle to the successful use of metadata approach. In some resource discovery systems, metadata-based retrieval is combined with content-based discovery to form a hybrid approach, which gains higher effectiveness and efficiency than either of the single approaches (Sumner, Bhushan, & Ahmad, 2003).

2.2.4 Categorization of Metadata

Metadata can be categorized in many ways. Pöyry, Pelto-Aho and Puustjärvi (2002) classify metadata according to its usage purpose into three categories: structural, control, and descriptive metadata. Structural metadata is used to describe the structural characteristics of an object, such as the format of the object, but it does not itself contain any information on the content of the object. Control metadata is created and used for controlling the flow of content in the information system in question. Descriptive metadata can be further divided into two sub-categories: contextual metadata and content-based semantic metadata. By contextual metadata we mean the conditions and the environment in which the object is created, e.g. the equipment needed to produce the actual object. Semantic metadata refers to the semantic characteristics of the object, i.e. explains the meaning of the object. Using semantic metadata requires commonly agreed semantic interpretations among all the users of metadata. Semantic metadata is very much domain specific, which means that the nature of semantic metadata is highly dependent on the concepts and semantic structures of the specific field.

According to the Advanced Distributed Learning (ADL) initiative's SCORM overview (ADL, 2004), metadata are grouped into three types in terms of the nature of the learning resource to be described. They are Asset Metadata, Content Organization Metadata and Sharable Content Object (SCO) Metadata. Asset Metadata can be applied to "raw media" Assets, providing descriptive information about the Assets independent of any usage or potential usage within courseware content. This metadata is used to facilitate reuse and discoverability, principally during content creation, of such Assets within, for example, a content repository. Content Organization metadata describes the content organization. The purpose of applying Content Organization Metadata is to make the content organization accessible within, for example, a content repository and to provide descriptive information about the content organization. Sharable Content Object Metadata is applied to SCOs to provide descriptive information about the content represented in the SCOs. This meta-data is used to facilitate reuse and discoverability of such content within, for example, a content repository.

From the viewpoint of the nature of the metadata itself, metadata can be either objective or subjective. Objective metadata are factual data, such as creator, contributor, publisher, date, type, format and many other physical attributes. Subject metadata provide opinions and evaluations about the described data resource. A peer review about an online paper, or the opinions supplied by readers at the Amazon.com, are good examples of subjective metadata. According to Hodgins (2000b), and Duval and Hodgins (2003), subjective metadata are more varied but also more valuable, compared to their objective counterparts. Especially as personalization is being more emphasized in learning than ever before, subjective metadata become increasingly important for learning objects.

There still exist other criteria to break down metadata into distinct categories. For example, Gilliland-Swetland (2000) states metadata can be categorized as administrative, descriptive, preservation and technical metadata, in accordance with its functionality. Lightle (2004) has a similar viewpoint and points out that administrative metadata is used to support resource management, descriptive metadata to facilitate resource discovery and identification, structural metadata to describe how the components of complex learning objects are bound together, while preservation metadata describing the hardware requirements, operating system and rendering software of the object can be used in the migrating and archiving of the object.

2.2.5 Who Creates Metadata?

Metadata can be created either manually or automatically. Especially, some of those objective attributes such as author name and create date may be generated by the authoring system when the learning object itself is created. An example of automatic metadata generation is the case when using Microsoft Word, author name and create date can be automatically imbedded in the file properties. In most cases and for most metadata attributes, however, human intervention is need. Who should be responsible to metadata creation?

There are two camps in the metadata world regarding this issue: "internal reference" camp and "external reference" camp (McGreal & Roberts, 2001). The "internal reference" camp believes that the authors of learning objects should input their own metadata. The argument is that the number of learning objects is growing so rapidly that there will not be sufficient number of professionals available for metadata creation. The "external reference" camp asserts that only librarians or information specialists can input the metadata. The reason is that only professionals can ensure the integrity of the input data.

Because of the rapid growth in the number of electronic resources and the accompanying tendency for authors to publish their own material on the web, the prospect of authors adding their own metadata now becomes attractive.

2.3 Learning Objects Repositories

2.3.1 Introduction to Learning Objects Repository

As stated above, learning objects are relatively small content components that can be used and reused for learning purposes in different contexts. Metadata associated to these learning objects facilitate the accessibility and reusability. Building learning object repositories, however, is the crucial first step in delivering online education, because whole courses and curriculums can be built and reused again and again from well-organized repositories.

Learning object repositories are organized collections of content stored on servers and delivered through networks. As illustrated in figure 2.4, a learning object repository stores

both learning objects and their metadata, either by storing them physically together or by storing them separately, while presenting to outside world a combined repository.

A learning object repository allows registered or unregistered users to search and retrieve learning objects, based on criteria that are relative to a certain metadata schema, which can either be an accredited standard, or an open specification, or an application profile developed by a specific community to meet its particular purposes. A learning object repository typically supports both simple and advanced queries. In a simple query, keywords input by users are matched against the text in metadata elements. In an advanced query, users are allowed to specify values for specific metadata elements, and sometimes a combination of search criteria is used. In addition, a learning object repository supports browsing through information materials by subject or discipline, allowing end users to descend in a tree of disciplines and sub-disciplines to get an impression of the objects in different domains.

Some issues must be considered and decided when designing and implementing a learning object repository. These issues are addressed by Neven and Duval (2002):

1. Metadata schema to be used. The organization can either adopt LOM or develop its own application profile by which to gather a more specialized collection of learning items.
2. Whether to manage Intellectual Property Rights.
3. Functionalities to be provided. This includes whether to implement simple or advanced search or both, whether to implement browse function, whether to support peer-reviewing, and whether to provide a personal workspace where the user can fill with objects from the repository.
4. System architecture. The organization must decide on issues such as whether to use client/server or peer-to-peer architecture and relative topics.
5. Database. The organization must decide whether to implement relational database or something else like XML database.
6. Document storage. The organization must decide whether to use a document repository or link to documents spreading over the Internet.
7. Automatic generation of metadata (most likely some fields).

8. The ability to interconnect with other repositories for importing and exporting leaning objects. This introduces the issue of interoperability, especially because systems usually use different metadata schemas, most likely their own application profile.

2.3.2 Examples of Learning Object Repositories

As the international standardization in learning area has been reaching a great progress, the number of learning object repositories is also rapidly increased. Therefore it is quite impossible to exhaustedly list all learning object repositories currently available online. Neven and Duval (2002) carried out a survey on LOM-based learning object repositories. The survey result is shown in table 2.5. McGee (2004) and University of Wisconsin website (2003) provide a more comprehensive and updated list of learning object repositories.

2.4 Summary

In this chapter we introduced learning objects and the associated metadata. Learning objects are conceptually small content components that can be used and reused multiple times in different contexts, and can be assembled together in many ways to construct various courses, lessons or programs. In the pursuit of reusability of learning objects, a consistent description of content is necessary so that learning objects can be located, identified, retrieved and eventually reused from anywhere, at anytime. Metadata is the description language for learning objects. Only combined with metadata, can learning object approach succeed. Literally meant "data about data", metadata is a set of pre-defined elements or attributes along with their values that describe various features or characteristics of a learning object.

Metadata consistency relies on standardization, and the metadata element set as well as its use is defined or controlled by a variety of metadata schemas developed by various organizations or initiatives. In next chapter, we will discuss metadata standardization, followed by a survey on metadata schemas, with focus on those for learning objects. We have seen in this chapter that currently there are hundreds of learning objects repositories accessible online to users, and each learning objects repository tends to develop its own application profile for object description and indexing in order to meet the specific needs of its community. This raises the issue of interoperability. How can we search, retrieve

and reuse learning resources across the whole Internet from different repositories that use different metadata schemas? The solution to this problem is intuitively an interchange mechanism between metadata schemas. Recognizing this, we will make a comparison between several metadata schemas and a mapping between their elements set, on the basis of the survey about metadata schemas.

Table 2.5 Examples of learning objects repositories with features and characteristics (Neven & Duval, 2002)

	ARIADNE	SMETE	Learning Matrix	iLumina	MERLOT	HEAL	CAREO	Learn-Alberta	EdnA	Lydia
Organ-ization	Foundation	Federation (Berkeley)	ENC	Project	Cooperation	US Nat. Science Foundation	Universities	Alberta Learning	non-profit	private org.
Meta-data scheme	IEEE LOM profile	IEEE LOM profile	IEEE LOM profile	IEEE LOM profile	IEEE LOM profile	IEEE LOM profile (CanCore)	IEEE LOM profile (CanCore)	IEEE LOM profile (CanCore)	Dublin Core profile	IEEE LOM profile (SCORM)
Subject domain	All	Science, mathematics, engineering and technology	Science, mathematics, engineering and technology	Science, mathematics, engineering and technology.	All	Health science	All	Kinder-garten to grade 12 (K-12) education	Education	All
# LO's	2498	1645	170	880	7408	N/A	1576	?	15782	48
IPR mgmt	Free and restricted	Free	Free	Free	Free	Free	Free	Free	Free	Free and restricted
Simple/ advanced search/ Browsing	Simple/ Advanced	Simple/ Advanced/ Browse by discipline	Simple/ Advanced/ Browse by discipline or resource type	Simple/ Advanced/ Browse by metadata fields	Simple/ Advanced/ Browse by discipline	Search and browse	Simple/ Advanced/ Browse by discipline	N/A	Simple/ Advanced/ Browse by discipline	Simple/ Advanced
Peer review	Metadata validation	No	Yes	Plan to create peer review, rating and recommendation	Review of content quality, effectiveness, ease of use	No	No	Yes	No	No
Personal Features	Metadata templates	Workspace, recommen-dations, communities	No	Plan to create workspaces	No	No	Workspace, download history	No	No	Transac-tion basket, purchase history
Distribution	Hierarch. knowledge pool syst.	Central server	Central server	Central server	Central server	Central server	Central server	Portal to reposito-ries	Central server	Global svr/ enterprise systems
Replication/ federated search	Replication of metadata, and free LO's	Federation	Federation	N/A	N/A	N/A	N/A	N/A	N/A	Replication
Metadata store	Oracle database for metadata	?	?	?	RDBMS, with export to XML	SQL Server 2000	?	?	?	?
LO store	Document Repository	Links	Links	Links	Links	Links	Document Repository + links	Document Repository	Links	Document Repository
Connection with other LOR's	Planned	API for federated search under developmen t	Federated search	No	Import of LOM metadata records planned	Under investigation	Collects objects from other LOR's	Portal which links to different databases	Open Archive Initiative under develop-ment	API
Other	Soon: Automatic metadata generation	N/A	N/A	Collections of related LO's	Discipline specific websites	N/A	Newest and most popular LO's	N/A	N/A	VPR's can share content

Chapter 3 Metadata Schemas

3.1 Metadata Standardization

Hodgins (2000a) pointed out, "revolutionary changes do not take off without adoption of common standards." While learning object approach is theoretically sound, the premise of its success is the establishment and implementation of consistent industry standards. Otherwise, learning object will become useless as soon as it leaves the database from which it came. The learning object is in many ways the crowning achievement of the standards initiative. Interoperable learning objects never occur without industry-wide standards. Metadata standardization, especially, plays a pivotal role in ensuring interoperability, portability and reusability of learning objects.

Standards, in general, are defined by the International Standard Organization (ISO) as "documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose". A learning object metadata standard defines the minimal set of properties required to allow these objects to be managed, located, and evaluated. The advantage of using a metadata standard is that data created by one will interoperate with those created by others as long as the same standard is used.

In the metadata standardization paradigm, the terms "metadata schema", "metadata standard", "metadata specification" or "application profile" are popularly employed and sometimes interchangeably used in literatures. But strictly they are different from each other. According to Friesen (2002a, 2002c), Duval, Hodgins, Sutton, and Weibel (2002), they can be defined as follows.

Standards are definitions or formats that have been approved by a recognized standards organization, or are accepted as de facto standards by the industry. Standards serve a regulatory function and have been created for programming languages, operating systems, data formats, communications protocols, and electrical interfaces. Publicly accredited Metadata standards are IEEE LTSC LOM and the ISO / International Electrotechnical Commission Joint Technical Committee 1 Sub Committee 36 (ISO/IEC JTC1 SC36).

Specifications are less evolved than standards and attempt to capture a rough consensus in the user or implementer community. Specifications enable people to get on with the job of system and content development. It can take a long time before specifications are finally approved as standards. Well known metadata specifications include Dublin Core, IMS, the Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE) metadata, and ADL SCORM.

Application Profile is a simplified and interpreted version of a standard or specification that is created to serve the needs of a particular community of users or implementers. Application profiles can combine elements from more than one specification or standard into a single profile, but should not modify these in such a way that would impact interoperability negatively. An application profile possesses four principal characteristics (Hunter & Lagoze, 2001): it may draw on one or more existing namespaces; does not introduce new metadata elements; can specify permitted schemes and values; and can refine standard metadata elements. Examples of application profiles include the Canadian Core (CanCore) - a subset of LOM, and EdNA Metadata Standard - an extension of Dublin Core.

The fundamental techniques (Duval & Hodgins, 2003) for the development of application profiles include giving elements a mandatory status, restricting the value space of data elements, imposing relationships between elements, excluding some elements, and identifying taxonomies and classifications in case of LOM.

Also, the different roles specifications, application profiles (implementations or reference models) and standards play in the standardization process can be illustrated as in figure 3.1, which graphically presents the development of LOM.

All these three are totally called ***Schemas*** in this thesis for convenience. In other words, what is called metadata schema throughout this thesis may either imply a metadata specification, a metadata standard, or a metadata application profile specifically developed for a particular community of interest.

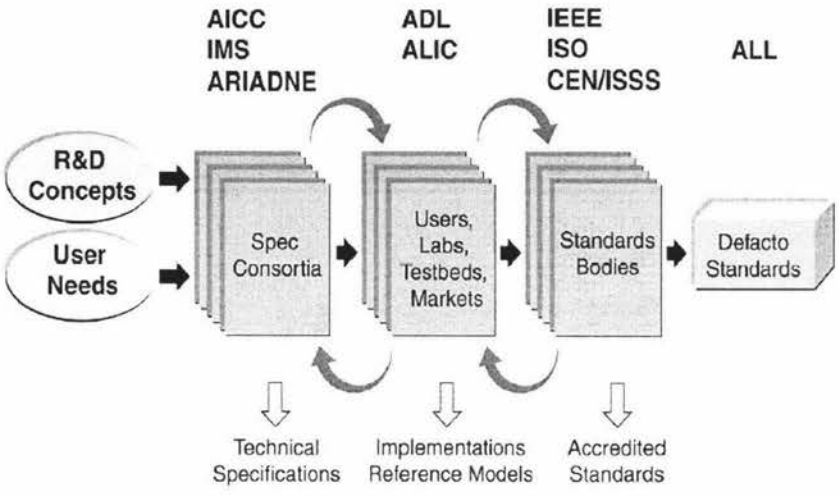


Figure 3.1 Specifications, application profiles and standards (Chen & Chen, 2002)

3.2 Metadata Schemas Overview

There are many metadata standards and specifications in various industries and even more application profiles for the specific needs of different user communities. No single metadata standard can be expected to accommodate the needs of all communities. Currently, an increasing number of software and system suppliers are working to metadata standards or creating their own proprietary standards for metadata. The growth of e-commence depends on metadata standards for exchange of data between applications. The e-government initiatives around the world are underpinned by the concept of interoperability and this in turn often depends on metadata standards. Portal software uses metadata standards to provide access to the information content of websites. Now the learning community is seeing a great effort on learning object metadata standardization in a pursuit of reusability and interoperability, which we think may result in an anywhere, anytime based, personalized learning.

In this section, a commonly accepted metadata schema will be briefly introduced for each of such areas as web community, e-government, libraries, archives, publishing industry, and multimedia, in order to present a total view of the current metadata standardization activities around the world. With respect to the educational sector, we will make a more in-depth survey on several metadata schemas for learning objects in next section. Haynes (2004) has done a very good job in his book on summary of metadata standardizations

across multiple industries. We follow the basic idea in this book to have an overview on metadata standardization activities.

3.2.1 Web Community and Dublin Core

3.2.1.1 Introduction to Dublin Core Metadata Initiative

The Online Computer Library Center (OCLC) and the National Center for Supercomputing Applications (NCSA) convened the invitational Metadata Workshop (Haynes, 2004) on March 1-3, 1995, in Dublin, Ohio to address the issue on describing and categorizing web content. This led to the formation of the Dublin Core Metadata Initiative (DCMI), which, according to the DCMI website, is "an organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems". It is a collaborative effort with participants from around the world and with a range of backgrounds, and relies on the co-operative efforts of individuals and organizations.

3.2.1.2 Dublin Core Metadata Element Set

Dublin Core is currently based on a set of 15 data elements. Table 3.1 shows the data elements, which are classified into three groups that roughly indicate the class or scope of information stored in them.

Dublin Core can be expressed in mark-up languages such as HTML, XML and RDF. Figure 2.6 in chapter 2 shows an example of Dublin Core metadata record coded in RDF format.

3.2.1.3 Dublin Core Metadata Principles

Adherence to the principles revealed at the Metadata Workshop increases the likelihood that the core element set will be kept as small as possible, the semantics of the elements will be clear enough to be understood by a wide range of customers without the need for training, and the element set will be flexible enough for the description of resources in a wide range of subject areas. These principles are addressed by Weibel (1995) as follows.

Table 3.1 The Dublin Core Metadata Element Set

Element		Element description
Content & about the resource	Title	The name given to the resource, usually by the creator or publisher. Can be the same as the title of the resource, or may be more descriptive.
	Subject	The topic of the resource. Typically, it will be expressed as keywords or phrases that describe the subject or content of the resource. Controlled vocabularies and formal classification schemes are encouraged.
	Description	A textual description of the content of the resource, including abstracts in the case of document-like objects or content descriptions in the case of visual resources.
	Source	The work, either print or electronic, from which this object is derived, if applicable. Source is not applicable if the present resource is in its original form.
	Language	The language of the intellectual content of the resource.
	Relation	Relationship to other resources, e.g. images in a document, chapters in a book, items in a collection.
	Coverage	Spatial locations and temporal duration characteristic of the resource.
Intellectual Property	Creator	Person or organization primarily responsible for creating the intellectual content of the resource, e.g. authors in the case of written documents, artists, photographers, etc. in the case of visual resources.
	Publisher	The entity (e.g. agency including unit/branch/section) responsible for making the resource available in its present form, such as a publishing house, a university department, or a corporate entity.
	Contributor	Person or organization not specified in a Creator element who has made significant intellectual contributions to the resource but whose contribution is secondary to any person or organization specified in a Creator element, e.g. editor, transcriber, illustrator.
	Rights	A rights management statement, an identifier that links to a rights management statement.
Electronic or Physical manifestation	Date	A date associated with the creation or availability of the resource.
	Type	The category of the resource, such as home page, novel, poem, working paper, technical report, essay, dictionary.
	Format	The data format of the resource, used to identify the software and possibly hardware that might be needed to display or operate the resource, e.g. postscript, HTML, text, the Joint Photographic Experts Group (JPEG) format, XML.
	Identifier	A string or number used to uniquely identify the resource. Examples for networked resources include URLs, Purls and the Uniform Resource Names (URNs). The International Standard Book Number (ISBN) or other formal names can be used.

1. Intrinsicality: Dublin Core concentrates on describing intrinsic properties of objects, in a way the importance of extrinsic data is by no means demeaned.
2. Extensibility: Dublin Core can be extended to meet the demands of more specialized communities.
3. Syntax Independence: syntactic bindings are avoided because Dublin Core is intended to be used in the widest range of applications.
4. Optionality: all elements are optional while allowing each application to define mandatory ones.
5. Repeatability: all elements are repeatable.
6. Modifiability: The elements may be modified in limited and well-defined ways through the use of specific qualifiers, such as the name of the thesaurus used in the subject element.

3.2.1.4 Current Status of Dublin Core

Dublin Core has received widespread acceptance amongst the resource discovery community and has become the de facto Internet metadata standard. It is also commonly applied in many other sectors because of the advantages provided by the design principles.

The main focus of Dublin Core is on improving resource discovery and cross-domain retrieval on the Internet. However, the development of interoperable standards and domain-specific applications that are compatible with Dublin Core is also underway. This is done by enabling the development of extensions and application profiles to meet the needs of different communities. These communities mainly include education sector, governmental context and library industry nowadays. In the education sector, particularly, a lot of initiatives and projects base their metadata schemas on Dublin Core. Moreover, a working group on education within the Dublin Core Metadata Initiative is working to extend the Dublin Core metadata to better describe educational resources (refer to the DCMI website).

3.2.2 Metadata Standardization in Government Sector

Government activity on the Internet from mid 1990s has led to the recognition that metadata is needed for the public to find information on the government websites. Unsurprisingly, almost all metadata schemas used for e-government happen to be application profiles adapted from the Dublin Core metadata model due to the adequacy of Dublin Core for this area.

The US Federal Government developed the ***Government Information Locator Service*** **(GILS)** (refer to the GILS-government website), which is intended to help people access information across different federal agencies. It is a decentralized service based on several US federal GILS sites with records mounted 32 different federal agencies. It is based on the Global Information Locator Service (refer to the GILS-global website).

The ***Australian Government Locator Service*** **(AGLS)** is a metadata schema for government on the Internet (refer to Australian Government Locator Service Metadata Guidelines web page). It consists of 19 data elements with refinements to some of the data elements. It is an extension of Dublin Core by adding the following four elements to the Simple Dublin Core element set.

1. Availability: primarily intended for use with non-electronic resources and indicates where the resource can be obtained.
2. Function: relates to the business function of the organization. This is the highest level of a functional analysis of the activities that a government organization undertakes.
3. Audience: the target audience for the resource. This may be defined in terms of educational level, age range, industry sector or other classification of individuals.
4. Mandate: an indicator of the legal mandate that requires the resource or service to be provided to the public. This field may refer to specific legislation or provide a Uniform Resource Identifier (URI) to the legislation.

The ***e-Government Metadata Standard*** **(eGMS)** (refer to the UK GovTalk website), based on Dublin Core, is developed for the UK government. At the writing of this thesis (October 2004), eGMS is in its version 3.0, consisting of 25 data elements. It uses similar

syntax to Dublin Core and can be expressed in XML. Many of its data elements can be directly mapped on to metadata elements from other schemas such as Dublin Core, GILS and AGLS. The eGMS is dynamic and development is continuing as new demands and applications become apparent.

3.2.3 Libraries and MARC

Library science has a long history to use metadata to describe intellectual works like books and other information records. MAchine-Readable **C**ataloguing (MARC) is a format standard developed in the library community for the storage and exchange of bibliographic records and related information in machine-readable form. MARC was established in the late 1960s to automate the production of catalog cards. It was used in early computer based library catalogues and has evolved into a series of national standards such as UKMARC managed by the British Library National Bibliographic Service (NBS), USMARK managed by the Library of Congress and MARC/CAN managed by the National Library of Canada. In 1999 USMARC and MARC/CAN were brought together into a single standard, MARC21. MARC21 has now been widely adopted by many cataloging authorities including the British Library. The MARC standard forms the basis for shared library cataloguing and for exchange of data between different library management systems. All MARC Standards conform to ISO 2709:1996 Information and documentation - Format for Information Exchange. The MARC Standards website provides detailed information.

3.2.4 Archives and ISAD

General International Standard Archival Description, ISAD(G), was developed by the International Council on Archives as a result of international collaboration. "This standard provides general guidance for the preparation of archival descriptions. It is to be used in conjunction with existing national standards or as the basis for the development of national standards." (International Council on Archives, 2000) The purpose of use is to identify and explain the context and content of archival material in order to promote its accessibility. In its version 2, there are 26 elements grouped in 7 areas:

1. Identity Statement Area;
2. Context Area;
3. Content and Structure Area;

4. Conditions of Access and Use Area;
5. Allied Materials Area;
6. Notes Area; and
7. Description Control Area.

Only a few among these 26 elements are considered essential for international exchange of descriptive information:

a. reference code;
b. title;
c. creator;
d. date(s);
e. extent of the unit of description; and
f. level of description.

The archival descriptions are based on a model of description with fonds at the highest level, then series and sub-series, then records and finally items.

3.2.5 Publishing Industry and ONIX

ONIX stands for ONline Information eXchange. ONIX is a standard format that publishers and vendors can use to distribute electronic information about books to wholesale, e-tail and retail booksellers, other publishers, and any other supply chain partner involved in the production, review or sale of books. ONIX is a well-coordinated attempt to use modern XML technology to facilitate the exchange of book metadata between companies involved in the book supply chains.

ONIX is the result of international collaboration coordinated by EDItEUR which represents 90 organizations around the world. ONIX Release 2.1 (refer to EDItEUR website) defines over 250 data elements that are arranged in 25 product groups. These groups are shown in table 3.2.

Table 3.2 ONIX product groups (Haynes, 2004)

Product group	Name
Group PR1	Record reference number, type and source
Group PR2	Product numbers
Group PR3	Product form
Group PR4	E-publication detail
Group PR5	Series
Group PR6	Set
Group PR7	Title
Group PR8	Authorship
Group PR9	Conference
Group PR10	Edition
Group PR11	Language
Group PR12	Extents and other content
Group PR13	Subject
Group PR14	Audience
Group PR15	Description and other supporting text
Group PR16	Links to image/audio/video files
Group PR17	Prizes
Group PR18	Content items
Group PR19	Publisher
Group PR20	Publishing dates
Group PR21	Territorial rights and other sales restrictions
Group PR22	Dimensions
Group PR23	Related products
Group PR24	Supplier, availability and prices
Group PR25	Sales promotion information

ONIX is designed around a minimal set of metadata elements so that data is provided in a consistent format that can be exploited by Internet booksellers, who in turn can provide information on price, content, etc. to their clients. ONIX provides following benefits (Haynes, 2004):

1. Allow publishers to provide single, consistent data set to all customers;
2. Provide a guideline for all content that can be used for online selling;
3. ONIX-compliant data supplied directly to retailers provides them with a data feed that can be taken as the authoritative source for all fields;
4. Opportunity for publisher to own how their titles appear on online sellers' sites;
5. Accurate data about publications because it is originated by the publisher; and
6. Savings in data entry by the retailer.

3.2.6 Multimedia Metadata Standards

MPEG-7, formally named "Multimedia Content Description Interface", is an ISO/IEC standard (ISO/IEC TR 15938-8:2002) developed by the Moving Picture Experts Group (MPEG, refer to the MPEG website) for describing the multimedia content data that supports some degree of interpretation of the information's meaning, which can be passed onto, or accessed by, a device or a computer code. MPEG-7 is not aimed at any one application in particular; rather, the elements that MPEG-7 standardizes support as broad a range of applications as possible. ISO/IEC JTC1/SC29/WG11 (2003) provides details on MPEG-7.

The overarching MPEG-21 aims at defining the technology needed to support users to exchange, access, consume, trade and otherwise manipulate digital items in an efficient, transparent and interoperable way. It identifies and defines the mechanisms and elements needed to support the multimedia delivery chain as well as the relationships between and the operations supported by them. Within the parts of MPEG-21, these elements are elaborated by defining the syntax and semantics of their characteristics, such as interfaces to the elements. MPEG-21has been coded as an international standard ISO/IEC TR21000. It covers the generation, use, manipulation, management and delivery of multimedia content across different networks and devices. ISO/IEC JTC1/SC29/WG11 (2002) provides details on MPEG-21.

The JPEG2000 (ISO/IEC 15444-1:2000), developed by the Joint Photographic Experts Group for digital images, defines JPX file format (ISO/IEC 15444-2:2002) with metadata elements associated with it. The metadata elements can be available either as an XML Schema Definition (XSD) or as a Document Type Definition (DTD), and cover details of how the image file was created, who was responsible for it and the content of the image. JPEG2000 Metadata website provides details on JPEG2000 metadata.

3.3 Metadata Schemas for Learning Objects

3.3.1 General Introduction

There should be nearly as many metadata schemas as there are initiatives and projects if we take into account various application profiles. So any attempt to make an exhausted survey on these schemas would be impossible because of the huge number (probably

thousands) of initiatives and projects around the world. What we can do is to make the survey on several best-known schemas in this area, and this would be sufficient for the purpose of our thesis.

The principal metadata schemas for learning objects should be Dublin Core, IMS, ARIADNE, ADL SCORM, the European Committee for Standardization/Information Society Standardization System (CEN/ISSS) metadata, ISO IECJTC1 SC36 and IEEE LOM. Among which, the two most broadly accepted metadata schemas are IEEE LOM and Dublin Core. There seems to be a consensus that these two are the most important in the description of learning resources (Fischer, 2001). In fact, there was an agreement on the international metadata specifications under the IEEE umbrella, and the ISO is in this agreement and incorporates the IEEE metadata specification into a formal international standard (McGreal & Roberts, 2001). Nowadays, various metadata application profiles developed by different communities are almost all based on either LOM or Dublin Core. As described above, Dublin Core was originally designed for web resource discovery and retrieval, but now has also been broadly adapted for description of learning resources due to its simplicity and other advantages it provides. LOM, at the same time, is the only officially approved standard for learning objects, and has been widely adopted not only in the education community but also in other industries. Therefore, we will concentrate on the "LOM family", on one hand, by discussing the IEEE LOM standard followed by an introduction to the two important relevant metadata initiatives e.g. IMS and SCORM, and on the "Dublin Core family", on the other hand, by presenting the EdNA Metadata Standard, an application profile based on an extension of Dublin Core.

3.3.2 LOM

3.3.2.1 Development of LOM

The IEEE Learning Technology Standards Committee has been developing the Learning Object Metadata (LOM) standard since 1997, based on original work in IMS and ARIADNE. In fact, both IMS initiative and ARIADNE project took part in the metadata standardization activity in the same year, 1997. Then they began collaboration under the auspices of the IEEE LTSC Committee on effort to reach, as quickly as possible, an educational metadata set that would be widely acceptable (refer to the IMS Global Learning Consortium website and the ARIADNE Educational Metadata Recommendation website). In 1998, IMS and ARIADNE submitted a joint proposal and specification to

IEEE, which formed the basis for the current IEEE Learning Object Metadata (LOM) base document. Since then the development of LOM has been handed over to the LTSC, where, through multiple drafts and revisions, LOM was gradually developed into an official IEEE standard. Now, the IEEE LTSC has initiated the move of this work to the full ISO standards by establishing ISO Joint Technical Committee 1 (JTC1) Sub Committee 36 (SC36) on Learning Technology. The next step envisioned for LOM is formal standardization by ISO through JTC1 SC36. While this was taking place, implementers and others required stable, publicly available versions of the specification - both of the abstract data model, and of its bindings in XML and other encodings. This need was addressed by the IMS initiative, which developed these documents, and in effect "hosted" an open and stable version of the specification for the general community (Friesen, 2002b). LOM development process can be specified from figure 3.2.

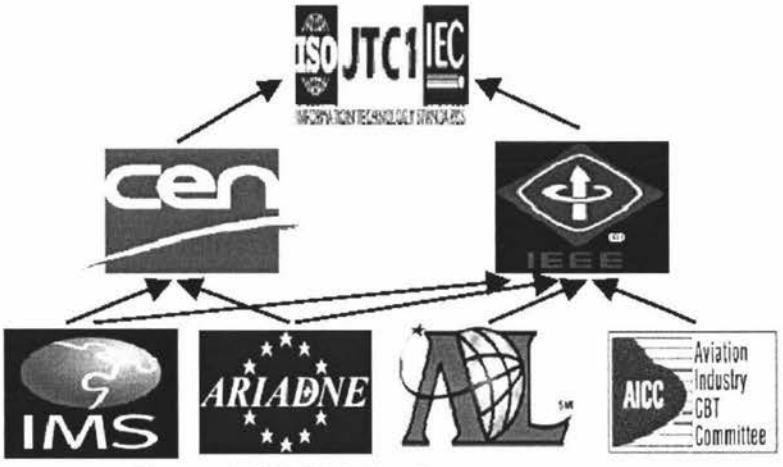


Figure 3.2 LOM development process:
IMS + ARIADNE ►IEEE ►ISO JTC1 (Duval, 2004)

3.3.2.2 LOM Overview

Referred to as IEEE LTSC LOM P1484.12, LOM is currently becoming an approved standard called 1484.12.1 IEEE Standard for Learning Object Metadata (IEEE Std 1484.12.1, 2002) which was released in September 2002. According to this release, "The purpose of this multi-part Standard is to facilitate search, evaluation, acquisition, and use of learning objects, for instance by learners or instructors or automated software processes. This multi-part Standard also facilitates the sharing and exchange of learning objects, by enabling the development of catalogs and inventories while taking into account the diversity of cultural and lingual contexts in which the learning objects and their metadata are reused."

According to Suthers, Johnson, and Tillinghast (2001), the LOM standard is meant to provide a semantic model for describing the learning objects themselves, rather than dealing with matters of how these learning objects may be used to support learning; it specifies the legal value and semantics of each metadata element, its dependency on others, and how metadata elements are composed into a larger structure. The LOM information structures are intended to support information exchange, and are neither specifications of an implementation nor specifications of a user interface. LOM allows for linguistic diversity of both learning objects and the metadata instances that describe them. LOM supports a complete separation of semantic model and its bindings. It is agnostic concerning bindings or implementations of metadata in representations and notations (Suthers, Johnson, & Tillinghast, 2001). LOM accommodates extension mechanism for localization. New data elements can be introduced anywhere in the LOM hierarchy. A good example is the development of metadata schemas by extending the current LOM for the description of dynamic multimedia learning contents and adaptive learning resources (Conlan, Hockemeyer, Lefrere, Wade, & Albert, 2001; Saddik, Ghavam, Fischer, & Steinmetz, 2000). All LOM elements are optional, which allows practitioners to specify mandatory or core elements of their own to meet the specific needs. In order for consistent description and semantic interoperability, all elements need to be described with vocabularies or recommended lists of appropriate terms (Van Assche, Campbell, Rifon, & Willem, 2003), and LOM provides pre-defined vocabularies for some elements. Most of the elements in a LOM instance may have multiple values, while one learning object may has more than one LOM instance associated with it.

Originally, LOM is built on the basis of work done by Dublin Core. It is compatible with Dublin Core and can be mapped to the Dublin Core element set. Such mapping can be found in the 1484.12.1 IEEE Standard for Learning Object Metadata (IEEE Std 1484.12.1, 2002). However, compared to Dublin Core, Farance (2003) identifies some deficiencies in LOM, which have led to resistance to significant adoption of LOM. These deficiencies include: most of LOM has been done better by Dublin Core, the non-Dublin Core portion of LOM is done poorly, the approach towards LOM value domains is fundamentally flawed, and so on.

In spite of these deficiencies, the LOM standard is still being used or referenced in many international repository efforts like MERLOT and ARIADNE, as well as in the U.S.

Department of Defense SCORM initiative. Refer to figure 2.7 in chapter 2 for a more comprehensive but by no means exhausted list of repositories implementing LOM.

3.3.2.3 LOM Element Set

The LOM data model is a hierarchy of data elements, including aggregate data elements (container elements) and simple data elements (leaf nodes in the hierarchy). Only simple data elements may have values of a certain data type, while aggregate elements just function as containers (parents) of other data elements. The hierarchical element set can be graphically illustrated as shown in figure 3.3.

This element set includes generic informational items such as title, author, description and keywords, technical aspects such as file size and type, and also includes educational and interpretive aspects like typical learning time and educational context. As we could see from figure 3.3, all elements in the element set are grouped into 9 categories:

a) The *general* category groups the general information that describes the learning object as a whole.
b) The *lifecycle* category groups the features related to the history and current state of this learning object and those who have affected this learning object during its evolution.
c) The *meta-metadata* category groups information about the metadata instance itself (rather than the learning object that the metadata instance describes).
d) The *technical* category groups the technical requirements and technical characteristics of the learning object.
e) The *educational* category groups the educational and pedagogic characteristics of the learning object.
f) The *rights* category groups the intellectual property rights and conditions of use for the learning object.
g) The *relation* category groups features that define the relationship between the learning object and other related learning objects.
h) The *annotation* category provides comments on the educational use of the learning object and provides information on when and who created the comments.
i) The *classification* category describes where this learning resource falls within a particular classification system.

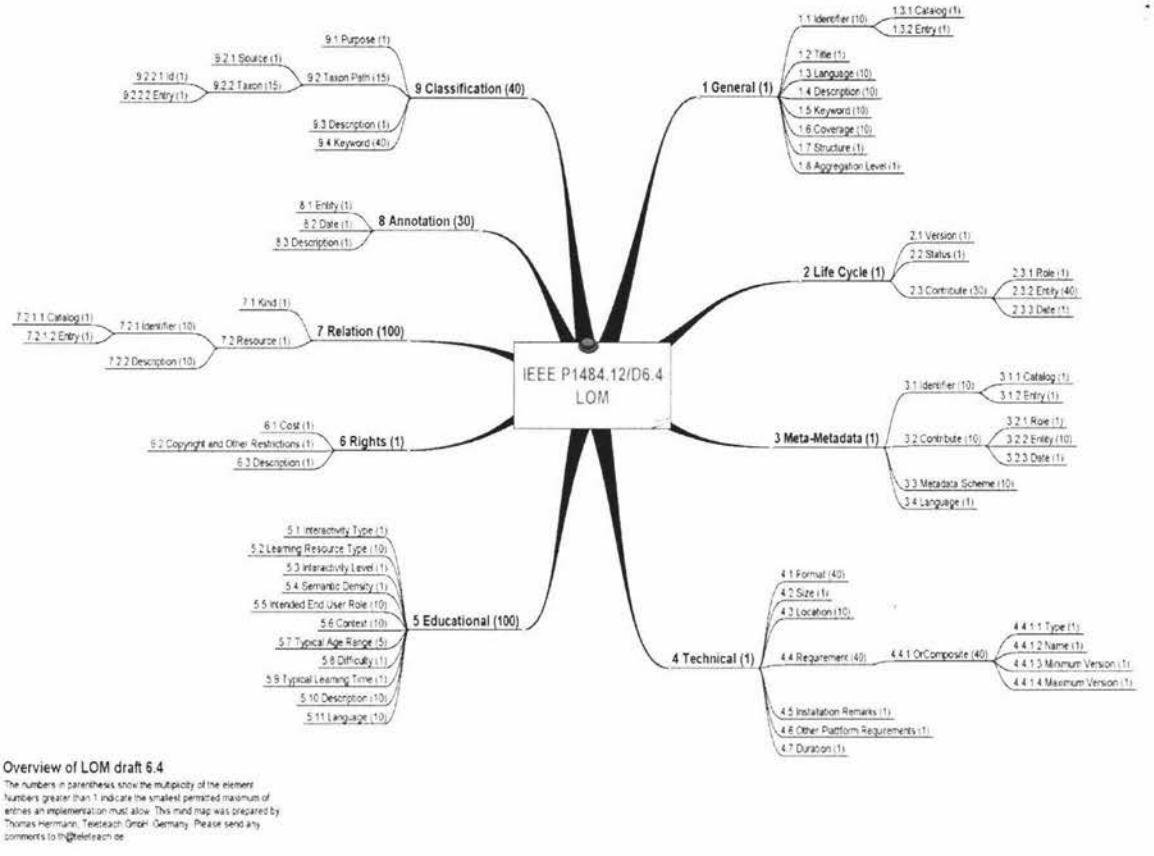


Figure 3.3 LOM element set (Hodgins & Duval, 2004)

Unlike Dublin Core, LOM focuses on the description of modular and reusable learning objects to facilitate their use by educators, authors, learners, and managers (IEEE LTSC, 2002). In further contradistinction to Dublin Core, LOM undertakes this task through a so-called "structuralist" rather than a "minimalist" approach to metadata (Friesen, 2004). Instead of presenting a simple data model that defines a minimal set of data elements, LOM identifies over 70 elements, covering a wide variety of characteristics attributable to learning objects. All LOM elements are placed in interrelationships that are both hierarchical and iterative.

3.3.3 IMS and ADL SCORM

Regarding the LOM family, we discuss the IMS metadata and the SCORM metadata in this section, not only because they both are widely accepted but also they have a very close relationship with the IEEE LOM. This relationship can be seen from figure 3.1 at the beginning of this chapter. As described above, IMS metadata was originally the base for LOM, and has been aligning itself with LOM since LOM was handed over to IEEE and became an officially accredited standard. SCORM metadata, on the other hand, is an

application of LOM to various SCORM Content Model components such as Asset, Sharable Content Object (SCO) and Content Aggregation. The SCORM application or implementation of LOM in turn provides an evaluation of the LOM standard.

Kotze (2004) made a good comparison between data elements of several metadata schemas in a survey on LOM and its application profiles. Figure 3.4 shows the result; from which we may have a total understanding of the element set of IEEE LOM, IMS metadata specification and SCORM metadata. In general, IMS metadata specification is identical to LOM, while the SCORM metadata application profiles directly reference LOM but with some elements specified as mandatory.

3.3.3.1 IMS Metadata Specification

The Instructional Management System body was established by EduCom (now EduCause) in 1994. The mandate of IMS is to serve as a catalyst for the development of instructional software, the creation of an online management infrastructure for learning, the facilitation of collaborative activities and certification. Although the IMS metadata specification represents an important activity and contributes significantly to the IEEE LOM, IMS is not just metadata schema. It is involved in the development of many other specifications for the learning community. These learning specifications include: content packaging, digital repository interoperability and learning design. Regarding metadata, another very important contribution from IMS is the XML binding scheme and the metadata implementation guide. The IMS Learning Resource XML Binding specification provides a sample XML representation and XML control files (DTD, XSD) to assist developers with their metadata implementations. The IMS Learning Resource Meta-Data Best Practice and Implementation Guide provides general guidance about how an application may use LOM meta-data elements.

	IEEE LOM	Dublin Core	IMS	SCORM Content Aggregation	SCORM Sharable Content Object	SCORM Asset	CanCore	UK LOM Core	SA LOM Core
1	General		O	M	M	M	O	M	M
1.1	Identifier		O	M	M	M	O	M	M
1.1.1	Catalog		O	O	O	O	O	M	M
1.1.2	Entry	O	O	O	M	M	O	M	M
1.2	Title	O	O	M	M	M	O	M	M
1.3	Language	O	O	O	O	O	O	M	M
1.4	Description	O	O	M	M	M	O	M	M
1.5	Keyword	O	O	M	M	O		O	M
1.6	Coverage	O	O	O	O	O	O	O	O
1.7	Structure		O	O	O	O		O	O
1.8	Aggregation Level		O	O	O	O	O	O	O
2	Life Cycle		O	O	M	O	O	M	M
2.1	Version		O	O	M	O	O	O	O
2.2	Status		O	O	M	O		O	O
2.3	Contribute		O	O	O	O	O	M	M
2.3.1	Role	O	O	O	O	O	O	M	M
2.3.2	Entity	O	O	O	O	O	O	M	M
2.3.3	Date	O	O	O	O	O	O	M	M
3	Meta-metadata		O	O	M	M	O	M	M
3.1	Identifier		O	O	M	M	O	M	M
3.1.1	Catalog		O	O	O	O	O	M	M
3.1.2	Entry		O	O	M	M	O	M	M
3.2	Contribute		O	O	O	O	O	M	M
3.2.1	Role		O	O	O	O	O	M	M
3.2.2	Entity		O	O	O	O	O	M	M
3.2.3	Date		O	O	O	O	O	M	M
3.3	Metadata Schema		O	O	M	M	O	M	M
3.4	Language		O	O	O	O	O	M	M
4	Technical		O	O	M	M	O	M	M
4.1	Format	O	O	O	M	M	O	O	O
4.2	Size		O	O	O	O	O	O	O
4.3	Location		O	O	O	O	O	M	M
4.4	Requirement		O	O	O	O		O	
4.4.1	orComposite		O	O	O	O		O	
4.4.1.1	Type		O	O	O	O		O	
4.4.1.2	Name		O	O	O	O		O	
4.4.1.3	Minimum Version		O	O	O	O		O	
4.4.1.4	Maximum Version		O	O	O	O		O	
4.5	Installation Remarks		O	O	O	O		O	
4.6	Other Platform Requirements		O	O	O	O	O	O	O
4.7	Duration		O	O	O	O	O	O	O
5	Educational		O	O	O	O	O	O	O
5.1	Interactivity Type		O	O	O	O		O	
5.2	Learning Resource Type	O	O	O	O	O	O	O	O
5.3	Interactivity Level		O	O	O	O	O	O	O
5.4	Semantic Density		O	O	O	O		O	O
5.5	Intended End User Role		O	O	O	O	O	O	O
5.6	Context		O	O	O	O	O	O	O
5.7	Typical Age Range		O	O	O	O	O	O	O
5.8	Difficulty		O	O	O	O		O	O
5.9	Typical Learning Time		O	O	O	O	O	O	O
5.1	Description		O	O	O	O		O	O
5.11	Language		O	O	O	O	O	O	O
6	Rights		O	O	M	M	O	M	M
6.1	Cost		O	O	M	M	O	O	O
6.2	Copyright and Other Restrictions		O	O	M	M	O	M	M
6.3	Description	O	O	O	O	O	O	M	M
7	Relation		O	O	O	O	O	O	O
7.1	Kind		O	O	O	O	O	O	O
7.2	Resource	O	O	O	O	O	O	O	O
7.2.1	Identifier		O	O	O	O	O	O	O
7.2.1.1	Catalog		O	O	O	O	O	O	O
7.2.1.2	Entry		O	O	O	O	O	O	O
7.2.2	Description	O	O	O	O	O		O	O
8	Annotation		O	O	O	O	O	O	O
8.1	Entity		O	O	O	O	O	O	O
8.2	Date		O	O	O	O	O	O	O
8.3	Description		O	O	O	O	O	O	O
9	Classification		O	O	O	O	O	O	O
9.1	Purpose		O	O	O	O	O	O	O
9.2	Taxon Path		O	O	O	O	O	O	O
9.2.1	Source		O	O	O	O	O	O	O
9.2.2	Taxon		O	O	O	O	O	O	O
9.2.2.1	Identifier		O	O	O	O	O	O	O
9.2.2.2	Entry		O	O	O	O	O	O	O
9.3	Description		O	O	O	O		O	O
9.4	Keyword		O	O	O	O	O	O	O

Figure 3.4 Element by element comparison for metadata schemas (Kotze, 2004) M-mandatory, O-optional

It is necessary to talk a little more about the relationship between IMS and LOM, because some statements about the two schemas in literatures are quite confusing, and also because we can hardly differentiate them if comparing IMS and LOM on an element-by-element basis just as done in figure 3.4. Also, terms like "IMS LOM" frequently appear in literatures such as the work by Lightle (2004). CanCore Guideleines Version 2.0. (2002) states "The Learning Object Metadata standard (IEEE 1484.12.1-2002 or LOM) is also known as IMS Learning Resource Meta-data." And Friesen & McGreal (2002) points out "LOM is almost identical to the IMS metadata specification". Originally, IMS metadata was by no means equivalent to LOM. IMS is adding to the LOM standard an XML binding scheme and the implementation guidance for the IMS community, because the LOM standard itself had neither included any information on how to represent metadata in a machine-readable format for exchanging metadata, nor addressed the issue about how an application uses LOM metadata elements before its final release in September 2002. It was in May 2002 when a working group in IEEE LTSC was formed for building standard for XML binding of LOM, and it was not until February 13, 2003 had the first draft of IEEE 1484.12.3 Standard for XML Binding for Learning Object Metadata Data Model been released. Now, according to the IMS Global Learning Consortium website, both the IMS Learning Resource Meta-data Information Model 1.2.1 Final Specification and the 1.2.2 Public Draft are replaced by the IEEE Std 1484.12.1 – 2002, IEEE Standard for Learning Object Metadata (LOM).

It should be noted that previously IMS identified mandatory elements by dividing the LOM element set into core and Standard Extension Library (SEL) elements. The core element set was called "IMS core", which was hoped to simplify metadata implementation. At that time, the IMS metadata was obviously different from the LOM standard. But since its version 1.2 released on May 17, 2001, IMS has dropped any distinction between "core" and "SEL" set of LOM elements.

3.3.3.2 ADL SCORM

Supported by the US Department of Defense and the US government, ADL is the home of the SCORM model, a Web-oriented data model for content aggregation focusing on the structure and run-time environment for learning objects. ADL SCORM combines and interprets a number of interrelated specifications built upon the work of the Aviation Industry CBT Committee (AICC), IMS and IEEE to create a unified content model. For

example, it is clearly shown in figure 3.5 that SCORM integrates Content Structure from AICC, Content Packaging from IMS and Meta-data from IEEE.

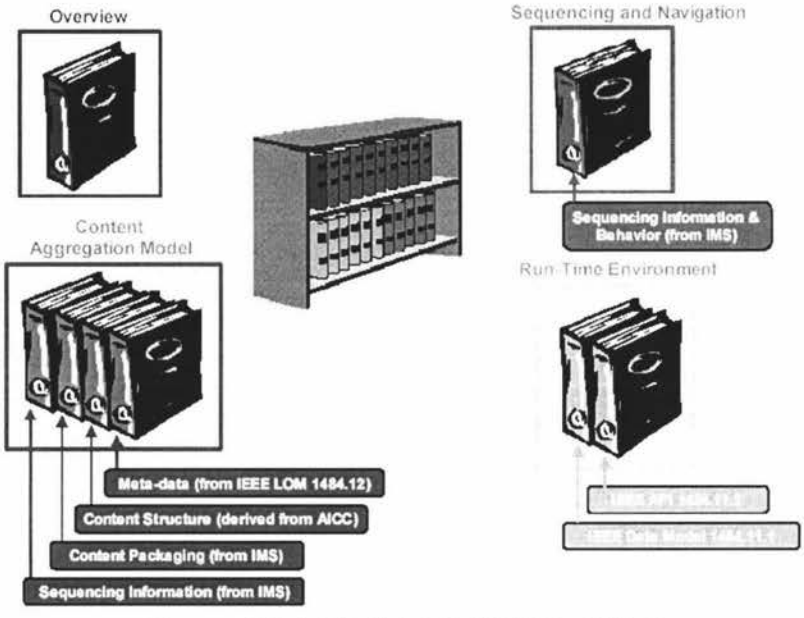


Figure 3.5 ADL SCORM (ADL, 2004)

The SCORM model specifies the behavior and aggregation of modular, interactive learning components, and heavily makes use of XML. Similar to IMS, SCORM is not simply about metadata. It also combines metadata with a number of other specifications that deal with a variety of aspects of learning content and management.

The SCORM's metadata model provides means for describing learning content from its most basic form – atomic resources such as text files, videos and presentations, to complex learning materials like lessons or entire courses (Simoes, Luis, & Horta, 2004), from category "Asset", through category "SCO" to category "Content aggregation". As illustrated in figure 3.4, for each category the metadata application profile is different in the mandatory element set. The SCORM Meta-data Application Profiles defined in the SCORM Content Aggregation Model (CAM) Version 1.3 directly reference both the IEEE 1484.12.1-2002 Learning Object Metadata Standard and the IEEE 1484.12.3 Draft Standard for Extensible Markup Language (XML) Binding for Learning Object Metadata Data Model. SCORM CAM model also provides additional specific guidance for using metadata in the SCORM context, although SCORM fully adheres to the IEEE standard.

3.3.4 EdNA Metadata Standard

3.3.4.1 Brief Introduction to EdNA Online

EdNA Online is a service that aims to support and promote the benefits of the Internet for learning, education and training in Australia (refer to the EdNA Online website). It is a cross-sector nationwide education and training project, managed by ***education.au limited***, a national agency collaboratively funded by the state, territory and ministries of education.

EdNA Online supports the education and training community by providing a range of tools and services, and a metadata repository - a database of metadata records describing evaluated online education and training-related resources. These metadata records are created and maintained by information officers allocated to each of the education sectors. Users can browse or search these records using a category structure organized by sectors (Millea, 2003). The so-called EdNA Metadata Standard is the schema for coding these records.

3.3.4.2 EdNA Metadata Standard

It should be acknowledged that the EdNA Metadata Standard is actually not a standard if measured by the definition given in section 3.1. It is an application profile of Dublin Core, through extending Dublin Core to meet the specific needs of the EdNA Online community. Because of this, the EdNA Metadata Standard is interoperable with many other Dublin Core based schemas such as AGLS (discussed in section 3.2.2) and the Learning Federation Metadata Application Profile, which was developed to provide a framework for describing learning objects in the school education sector. In addition, the EdNA Metadata Standard is extensible itself so that organizations are able to add their own metadata elements and qualifiers if needed.

First published as version 1.0 in August 1998, currently the EdNA Metadata Standard is at its version 1.1, which was approved by the **education.au** Board and ratified by the Australian Information and Communications Technologies in Education Committee (AICTEC) in December 2000. It comprises a set of guiding principles and a set of elements. These elements include the 15 Dublin Core elements (some with EdNA qualifiers) as well as the specific EdNA elements. Some of the extended EdNA elements are added specifically for educational purposes while others are specifically for the

administration of EdNA Online. The whole element set for EdNA Metadata Standard is shown in table 3.3.

Table 3.3 EdNA Metadata Standard element set

Dublin Core	DC.Identifier	DC.Creator	DC.Coverage
	DC.Title	DC.Date	DC.Rights
	DC.Description	DC.Type	DC.Contributor
	DC.Subject	DC.Format	DC.Source
	DC.Publisher	DC.Language	DC.Relation

	Element	**Element Description**
EdNA	EDNA.Audience	A category of user for whom the resource is intended.
	EDNA.Approver	Email of a person or organization approving the item for inclusion in EdNA Online.
	EDNA.CategoryCode	A numerical code derived from the database tables that support the EdNA Online Browse Categories.
	EDNA.Entered	Data item was entered as an entry in the online item database (used for management purposes).
	EDNA.Indexing	To what extent should EdNA Online indexing (spidering) software follow links from this page.
	EDNA.Review	A third party review of the resource.
	EDNA.Reviewer	Name of person and/or organization or authority affiliated with the review.
	EDNA.Version	Version of the EDNA Metadata Standard applied.

The EdNA Metatadata Standard V1.1 is organized into eight categories and has four levels: mandatory, highly desirable, desirable and optional. Each category has different data elements on these four levels. The "Minimum Metadata Sets by Sector" on the EdNA Metadata Standard website provides the details about these categories and levels.

3.4 Summary

In this chapter, we made a brief introduction to metadata standardization and an explanation to several terminologies such as "standard", "specification" and "application profile". After that, we had an overview on various metadata schemas across multiple industries including the web community, the government sector, the library science, the archiving society, the publishing industry and the multimedia area. On the basis of this

overview, we concentrated our discussion on metadata schemas in learning, focusing on the officially approved learning object metadata standard – IEEE LTSC LOM, the IMS metadata specification and the SCORM metadata application profiles. We addressed the close relationships among these three learning object metadata schemas. We identified that most metadata schemas used in the learning community are either based on Dublin Core or the LOM model. Regarding the "Dublin Core family", we had a brief view on the EdNA Metadata Standard, an application of Dublin Core to the EdNA Online community.

Although we only discussed a limited number of metadata schemas for learning objects, there might be in fact hundreds or even thousands of them around the world if we take all those application profiles into account. An attractive illusion is that all these metadata schemas can communicate with each other so that an extensive interoperability across the global Internet could be possible. Pursuit of global interoperability raises the issue of metadata interchange, to which we try to make a step forward in this thesis. In next chapter, we will address topics such as metadata mapping and metadata crosswalk, and then propose a framework for the interchange mechanism between different metadata schemas.

Chapter 4 Metadata Interchange and Prototype Design

4.1 The Issue of Metadata Interchange

While the learning object model presents the illusion of reuse of educational resources, and standardized metadata facilitate the interoperability, portability and reusability of learning objects, a critical issue is raised in the use of metadata, not exclusively for the learning community but for the whole context of information storage and discovery. The advent of the Internet and the rapid growth in online resources has increased the demand for the ability to search information across multiple metadata repositories. At the same time, as discussed in previous chapters, different metadata repositories are very likely to employ different metadata schema for their specific needs. To solve this problem, Woodley (2000) concluded that there are two ways for information societies to follow: either to convert their metadata to a more readily accessible format, or to provide a single interface to search multiple heterogeneous databases simultaneously. No matter which method to follow, he argues, the first step requires the work, referred to as metadata mapping or metadata crosswalk, which is important for supporting the demand of cross-domain searching.

Metadata interchange is needed not only for cross-domain information search, but also for a wide range of other purposes. An institution may want to transfer metadata from one schema to another, to upgrade to a new system because the legacy system has become obsolete, or to exchange data with another organization using a different metadata schema, or to use more than one schema to meet the various needs for documentation, management, security, and access. In addition, to reach the broadest community of users, metadata must be made available in accordance with a number of related metadata schemas (Margaret, 1998). However, as the number, size and complexity of metadata schemas continue to grow, supplying the metadata in multiple schemas becomes more and more repetitious, time-consuming and even impossible. In order to minimize the time and effort needed to create and maintain the metadata, while maximizing its usefulness to the widest community of users, there is an amounting need to maintain the metadata in one schema while allowing for accessibility via alternate schemas. Obviously, a well-established metadata interchange mechanism is needed in this process.

By metadata interchange, we simply mean a mechanism for changing a metadata instance automatically from one format to another, in terms of either metadata schemas or coding schemes, or both. For example, the transformation of metadata from a database record into an IMS-compliant, XML-formatted document is a kind of metadata interchange. We have carried out such a transformation in the developed prototype.

4.2 Crosswalks

4.2.1 Definition and Issues

Crosswalks or metadata mappings are fundamental to successful metadata interchange, and are essential for information resources to be accessible through a variety of portals and search interfaces. A crosswalk or a mapping is a formal identification of equivalent or nearly equivalent metadata elements or groups of metadata elements within different metadata schemas, carried out in order to facilitate semantic interoperability (Woodley, 2000).

A meaningful mapping requires a clear and precise definition of the elements in each schema, as well as in-depth knowledge and specialized expertise about the associated metadata schemas. Therefore, crosswalk is usually a difficult and error-prone task. Further more, as metadata schemas are independently developed by various organizations to meet different needs, and because they often use different terminologies, structures, and value spaces, inherently some issues arise during mapping one schema to another. Margaret (1998) distilled the key issues involved in metadata mappings, concluding that these issues include "one to many mapping", "many to one mapping", "extra elements in source", "unresolved mandatory elements in target", and so on. Similar conclusions are also reached by Woodley (2000), who further points out that there is rarely a one-to-one relationship between all of the elements in two metadata schemas, and in only a few cases does the mapping work equally well in both directions.

A perfect, thorough solution to these mapping issues should rely on the "formal metadata specification" and "formal crosswalk specification", as proposed by Margaret (1998). The former means the standardization of the development of metadata schemas. The latter means the establishment of a standard method to formalize the specification of crosswalks. The standardization of schema development could be reached by formalizing metadata

schemas through the use of a canonical representation or a metadata specification language (MSL), as well as by using a minimum set of data types. However, the expected schema formalization and crosswalk formalization have not occurred. As the result, the mapping between two metadata schemas is largely dependent upon the individual who has made the mapping. Different individuals would map the same two schemas in different ways. In most cases, no commonly accepted mappings are available for a schema to other schemas. However, some schemas provide as part of the schema itself the mapping to a certain metadata specification or standard. For example, the ARIADNE metadata offers a mapping of itself to the LOM standard and the Dublin Core metadata element set. For the proposed framework in this thesis to be powerful enough to accommodate multiple metadata schemas and scalable enough to allow for any new schemas without the need to change the existing system too much, a sophisticated approach is essential for resolving the metadata mapping issues. This approach employs the LOM standard as an intermediary, and we call this approach LOM-intermediated approach. By this approach, every schema accommodated in the system is mapped to the LOM standard in the first step, and then translated to the desired schema in the second step. The rational to this approach, as well as the relative issues, will be discussed in detail in later sections.

4.2.2 Examples of Crosswalks

Up till now, mappings between various metadata schemas have been identified by a range of organizations and individual experts. A typical example is the mapping of the Dublin Core Metadata Element Set to the LOM metadata model, documented by the IMS Initiative as a part of the IMS specification (IMS Guide, 2001), as well as by the IEEE LTSC as a part of the LOM standard (IEEE Std 1484.12.1, 2002). Another crosswalk example is the mapping between the ARIADNE metadata elements, the LOM standard, and the Dublin Core data set, developed by the ARIADNE Initiative itself. Appendices A-1 and A-2 show the mapping table of these two example crosswalks respectively. More examples of crosswalks can be found on the Internet, as many websites provide links to current work on this topic. For example, The MIT Libraries website (http://libraries.mit.edu/guides/subjects/metadata/mappings.html) provides links to crosswalks, such as ADL to the Federal Geographic Data Committee (FGDC) metadata to MARC, Dublin Core to the Encoded Archival Description (EAD), ONIX to MARC 21, and more.

4.2.3 Our Mapping Work

A crosswalk between EdNA, the Gateway to Educational Materials (GEM) Metadata Standard, and LOM is developed for the implementation of the prototype for metadata interchange. Appendix A-3 gives the details of the crosswalk. We briefly describe the EdNA-LOM crosswalk as an example of the mapping works. As discussed before, the EdNA metadata element set includes the publicly accepted Dublin Core data elements and the specific EdNA elements that are added only for the local needs of EdNA Online. What we really need to do is to map those specific EdNA elements to their equivalence in the LOM element set, because the mapping of Dublin Core to LOM has been well identified by the IMS Initiative and the Metadata Working Group of the IEEE LTSC, as discussed before and shown in Appendix A-1. As the result, the work done by us is shown in table 4.1, which presents the mapping between the specific EdNA elements, instead of the whole EdNA element set, and the LOM equivalence.

Table 4.1 EdNA-LOM Mapping

EdNA	LOM
EDNA.Audience	Education.IntendedEndUserRole Education.TypicalAgeRange
EDNA.Approver	Null
EDNA.CategoryCode	Null
EDNA.Entered	Null
EDNA.Indexing	Null
EDNA.Review	Annotation.Description
EDNA.Reviewer	Annotation.Person
EDNA.Version	MetaMetadata.MetadataScheme

Some of the crosswalk issues mentioned above are encountered. It should be acknowledged that four EdNA metadata elements have no counterparts in the LOM metadata standard, as these elements are specifically designed for the purpose of EdNA online resource management only. On the other hand, because LOM identifies much more elements than EdNA does, it is not surprising that many LOM elements have no equivalence in the EdNA metadata element set. Consequently, information loss is inevitable when changing the LOM metadata to the EdNA schema or vice versa. For a

general-purpose metadata framework like the one proposed in this thesis, the following guidelines are helpful in assuring a better solution.

1. Provide a means to allow for human interventions during the metadata interchange process so as to avoid any unnecessary loss of data, though a fully automatic mapping is desired.
2. Clearly inform users of the source of the crosswalks and warn them of the accuracy risks resulted from the automatic mapping.
3. Always retain the data provided by the user even if it does not map to intermediate steps, so that it can be offered for manual mapping or mapping back to original format.

4.3 Current Work on Metadata Interchange

One of the main disadvantages of metadata use is the continuously increasing number of metadata schemas available in the world. To overcome this disadvantage, researches have been launched on the technologies and tools that enable automatic mapping or interchange between metadata schemas. However, according to the author's investigation of the state-of-the-art, mature or commercially valuable metadata conversion tools are still in scarcity.

The technology of metadata registries is thought of as one of the promising approaches to this issue (Duval & Hodgins, 2003). The underlying basic idea is that a registry is undertaken to include information about the description of data elements in relevant metadata schemas, as well as the reference from elements in one schema to elements in other schemas, so that software agents can automatically transform queries on elements in one schema to queries on elements in other schemas, or transform metadata instance in one schema to that in other schemas. However, Duval and Hodgins (2003) pointed out that this technology is still at a very early stage and more effort is needed for its further development.

With respect to the development of metadata conversion tools, several prototypes have been proposed and implemented for the experimentation of metadata transformations. Yin, Xu, & Saddik (2003) proposed a web based mapping tool, which allows a conversion between Dublin Core, LOM and ARIADNE in a transparent way. The tool stores

metadata in a relational database and generates XML files as output. The prototype for this tool is implemented in a typical three-tiered architecture (figure 4.1): the client interface tier which facilitates user interaction with the system, the business logic tier which carries out metadata mapping and conversion, and the information system tier which provides database management. The system is implemented using Java technology.

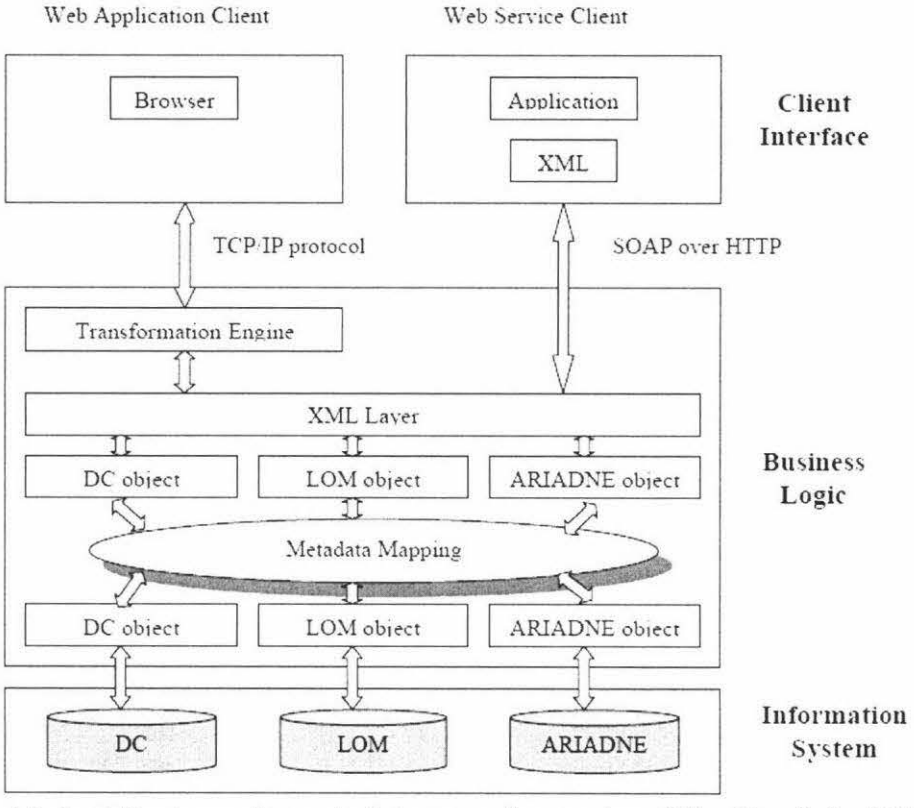


Figure 4.1 Architecture of a metadata mapping system (Yin, Xu, & Saddik, 2003)

Godby, Smith & Childress (2003) proposed a prototype for a web service that translates between pairs of metadata schemas. Despite the trend toward encoding in XML and the eXtensible Stylesheet Language Transformations (XSLT), they presented arguments for a design that features a more distinct separation of syntax from semantics. The result is a system that automates routine processes, has a well-defined place for human input, and achieves a clean separation of the document data model, the document translations and the machinery of the application. Figure 4.2 shows the high-level design of the service. It has three parts: a translator enclosed in a web service wrapper, a record submission client, and a crosswalk registry client. The submitted record from the submission client is translated according to lists of crosswalks, created by human experts.

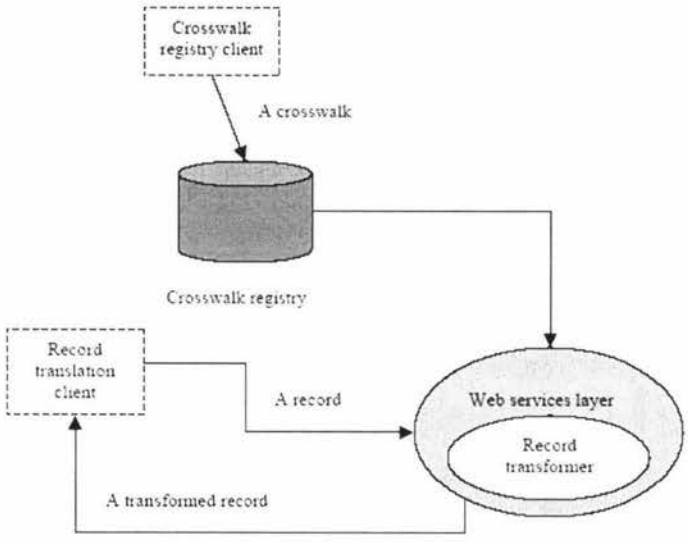


Figure 4.2 A metadata schema translation service (Godby, Smith & Childress, 2003)

In addition to the above two, some other researches on metadata conversion are published in the literature. Lightle, Ridgway and Smith (2003) developed for digital collections the crosswalks between metadata based on MARC, LOM and Dublin Core, with the ability to preserve as much of the data as possible as the data are crosswalked from one schema to another. Also, XML records conformed to a variety of metadata schemas can be generated so that the metadata can be searched and displayed correctly in catalog records generated through different digital library interfaces. The Learning Matrix's Cataloging Tool developed by Greer (2002) is a web-based solution for creating metadata while uploading resources to a digital library. Its database has been designed around LOM to facilitate the exporting of XML records for data sharing, and records can be exported either in LOM standard or in Dublin Core format. Karampiperis and Sampson (2003) presented a prototype implementation of an educational metadata management system (EM2) for the mapping between different metadata schemas. The results showed that the mapping mechanism handles real case mapping problems in a precise way, except that human interferences are needed. Santacruz-Valencia, Aedo, and Delgado-Kloos (2003) proposed a learning object framework, which allows translations between metadata schemas. The translation is governed by a configurable set of transformation rules expressed with XSLT.

There are also some metadata tools in practical use, which have the functionality of converting metadata records from one schema to some other schemas. DC-dot (http://www.ukoln.ac.uk/metadata/dcdot/) is an online metadata generator developed by

the UK Office for Library and Information Networking (UKOLN), which can generate Dublin Core metadata in HTML, RDF, XML, or the eXtensible HyperText Markup Language (XHTML), and transform the generated Dublin Core metadata instance to other schemas such as IMS and LOM according to user requirements. The data input could be made either through filling out a form by the user from scratch, or on the basis of an HTML document from which the values for some metadata fields can be automatically extracted. In the later case, the user needs to provide the URL for the HTML document, and may manually feed values for those metadata fields whose value cannot be obtained from the HTML document.

The MARC-LOM converter (http://marc-lom.athabascau.ca/marc/index.html) takes a MARC record file from the user's local computer normally accessed through Z39.50 - an information retrieval protocol, converts its format into the IEEE LOM schema, then displays a copy on the user's screen and saves a copy to a database.

The d2m (http://www.bibsys.no/meta/d2m/) is another online tool for converting Dublin Core to various MARC formats such as USMARC, DANMARC and FINMAEC. The Dublin Core input can be made either by providing the URL of the source metadata record or by pasting the meta-tagged HTML code onto the text-area as input. The MARC output can be presented to users in several coding schemes including plain text, HTML or a format defined by ISO 2709.

As a conclusion, however, either the prototypes proposed by researchers or the mapping tools developed by organizations are based on schema-schema crosswalks. For each schema-schema pair, a mapping between the two must be identified and a description of the relationships between their elements must be coded into the translation system. When the number of schemas we want to process in such a system becomes large, the workload of the metadata crosswalks becomes incredible. It is very difficult, even impossible, to build such a system scalable enough to accommodate as many metadata schemas as desired. In fact, it can be concluded that currently a flexible, general-purpose means that facilitates conversions among multiple metadata schemas is still unavailable, although mapping tools that can convert from one schema to a limited range of other schemas do exist. In this thesis, we will propose a general-purpose framework enabling metadata interchange among multiple schemas, and implement a prototype for this framework, in order to make things one-step forward.

4.4 A Metadata Interchange Framework and Prototype Design

4.4.1 Purpose and General Requirements

In this section, a general-purpose framework will be proposed for interchange among various metadata schemas. The principal purposes behind this framework are to maximize the interoperability of metadata schemas and to broaden the reuse of metadata records that have already been created. It is expected that a data record created in one schema can communicate with a system in which different metadata schemas are used. It is also expected that when a new record is to be created, existing metadata records (usually for other content objects) can be used to avoid any repetitious work often seen in metadata creation. Therefore, the fundamental requirements for this framework are considered as follows.

1. Intended user role: the system developed under this framework is intended to be used by cataloguing professionals, repository administrators, learning resource providers and educators.
2. Metadata schemas it can accommodate: the interchange framework should accommodate various metadata schemas to fit for a wide range of applications. In addition, the system should be scalable enough to be easily extended so that a new schema can be added in the system without much effort. In the prototype developed in this thesis, the conversion between IMS and EdNA is first implemented as an example to demonstrate the interchange mechanism. We then added to the system Dublin Core and LOM to show scalability, by presenting the ability to transform from any schema to any others within the total four schemas.
3. Output format: as the result of the interchange mechanism, the transformed metadata record should be output to users in a format that is platform independent, easy for humans to understand and for machine to process. XML perfectly meets these requirements thus is considered as the encoding scheme.
4. Input ways: in addition to the direct input of the source metadata records that are to be transformed to other metadata schemas, alternative ways should be provided for users to choose for the purpose of flexible and efficient data input. For example, the system might be expected to create as much metadata as possible automatically from existing learning content in a Learning Management System (LMS) such as WebCT.

4.4.2 The Proposed Framework

Taking into account the requirements discussed above, a framework for interchange of metadata schemas is proposed, where users can input metadata in three ways: by manually typing element values from scratch, or by importing element values from a uploaded metadata instance in XML format, or by automatically extracting element values from an online resource (usually in the form of a HTML web page). The system then stores the input data in a relational database, and is able to convert the stored data to various schema formats as desired by users. The transformed data result is output to users in the form of a XML document. This framework can be graphically illustrated as figure 4.3.

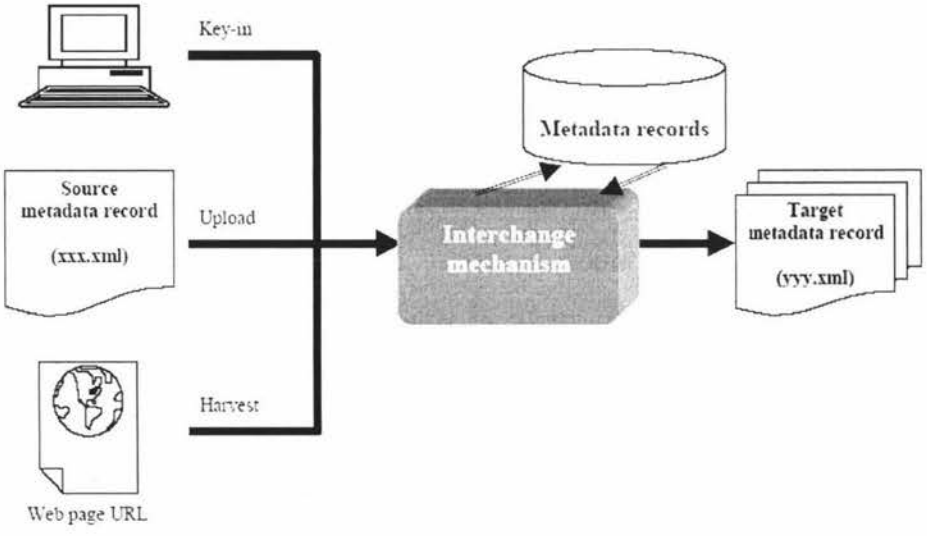


Figure 4.3 The proposed framework for metadata interchange

The main considerations lying behind the design of this framework include the scalability of the system, the interoperability of the data and the efficiency of the data creation.

4.4.2.1 System Scalability and the LOM-intermediated Approach

As a general-purpose metadata interchange system, it is required to be able to convert between each other among multiple metadata schemas. It is also required to be highly scalable or extensible to accommodate a new schema into the system without the need of significant changes to the current system. A crucial issue arising here is how to identify all the mappings between any two schemas. As the number of metadata schemas to be accommodated in the system is growing, the number of mappings for each schema-schema pair will be increased exponentially. In addition, when a new schema is added to

the system, mappings between this new schema and all the existing schemas must be identified and added to the system as well. The situation is especially worsened due to the fact that we never know what will come as the next new schema. As the result, scalability becomes difficult, if not impossible.

Our solution is the use of LOM as an intermediary. Instead of the mapping from the source schema directly to the target schema (we call direct mapping for convenience), the source schema is mapped to LOM firstly, and then mapped to the target schema from LOM. This way, LOM acts like an intermediary or a bridge between the source schema and the target schema. This approach is what we previously called the LOM-intermediated approach. By this approach, the amount of total mapping work will be greatly reduced. The effect of the LOM-intermediated approach compared against the direct mapping is shown in figure 4.4.

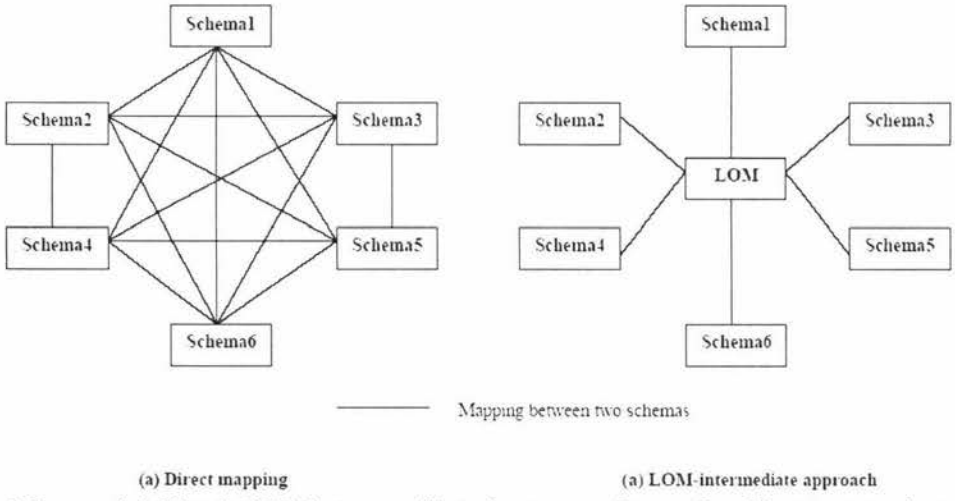


Figure 4.4 The LOM-intermediated approach vs. the direct mapping

As an example, consider the case that there are six schemas (as shown in figure 4.4) currently manipulated in the system and we want to accommodate a new schema into the system. The new schema should be made interchangeable with all the six existing schemas. If the traditional direct mapping is used, we need to identify and add to the system at least 6 mappings for that new schema to map with the six existing schemas. While if the LOM-intermediated approach is used, we only identify and add to the system the mapping between the new schema and LOM.

In conclusion, the LOM-intermediated approach gets rid of the heavy burden of identifying mappings for each schema-schema pair. Instead, each schema only needs to

be mapped to LOM. As mentioned before, although a thorough, perfect resolution for the metadata mapping issues has yet to be developed, the LOM-intermediated approach greatly improves the situation by vastly simplifying the mapping works and increasing the mapping accuracy. Mapping each schema to LOM is much easier and less error-prone than mapping each schema to all other schemas. In addition, as a first step to crosswalk formalization, we may hope each schema provide the mapping to LOM as part of the schema itself (actually some does, examples include ARIADNE, Dublin Core, etc.). This mapping should be able to be widely accepted. In the case the metadata schema itself does not offer a mapping to LOM, the system may have to provide one instead. This mapping provided by the system should be transparent to users and informative of any inaccuracy risks.

The reason for using LOM rather than any other schemas as the intermediary is obvious. The LOM standard specifies a large element set to try to cover all aspects attributable to learning objects. Therefore most other learning object metadata schemas should be (and actually are) a subset of LOM. In addition, LOM is the only officially approved metadata standard in the learning sector, thus many learning object metadata schemas try to make themselves LOM-compliant. Therefore, we can easily come into the conclusion that using the LOM standard as the intermediary may to the largest degree reduce the mapping work, maintain the mapping accuracy, and simplify the system design and implementation.

4.4.2.2 The LOM-based Database as the Implementation of the Approach

Regarding the implementation of the LOM-intermediated approach, the actual intermediary is the relational database that is designed and built around the LOM data model. As the database, together with all the tables and the data fields within it, is constructed on the basis of the LOM template, the database can in fact be thought of as a functional equivalence to the LOM standard. In other words, the LOM-based database, instead of the abstract LOM standard, functions like the intermediary or the bridge between any source data and the corresponding target data. Rather than directly converting the source schema to the target schema, data from the source schema are in the first place transformed into field values and stored in the LOM-based database, therefore can not only be translated to the target schema but also be translated to theoretically any other schemas or used for other purposes in the future. By doing so, the very complicated

schema-schema mapping is replaced by the much simpler schema-database mapping, and the total number of needed mapping tables will be significantly reduced, exactly the same as the situation presented above when we discussed the LOM-intermediated approach. The larger the number of metadata schemas to be accommodated, the more obvious the benefit of using the LOM-based database is.

4.4.2.3 Extra Data Loss and the Dynamic Database Structure

A cost to pay for the high scalability and the easier accommodation of multiple schemas by using the LOM-intermediated approach is the possibility of extra information loss, in addition to the usual information loss resulted from direct mappings as discussed before. The resource data is at first mapped to the intermediate schema –LOM in our case, while data loss might happen during this mapping process. Take the EdNA metadata for example. When we convert an EdNA metadata record into a certain schema, we first convert it into LOM according to the LOM-intermediated approach, rather than directly into the target schema. We know from table 4.1 that four EdNA elements have no equivalences in LOM, consequently, the information contained in these four elements will inevitably be lost in the process of the conversion from EdNA to LOM. While on the other hand, these four elements might have had equivalences in the target schema.

Although the LOM metadata model is developed to try to cover most aspects of learning objects, there still exist a few elements, such as the four in EdNA, which are not available in LOM but are identified by other schemas. To completely avoid this extra data loss, a dynamic database structure is developed to maintain those elements extra to LOM but available to a certain source schema. The idea is, the basic structure of the dynamic database is LOM-based, as described above, but when the input record (usually an uploaded XML file) is in a schema that has extra elements, the extra elements will be stored in extra table(s) dynamically and specifically created by the system for that schema. Once the dynamic table(s) has been created, they can be used for all later input in that schema. When intending to transform and output an XML record in that schema, the system retrieves data not only from the basic database but also from the specific extra tables as well.

4.4.2.4 A Further Option for Improving Mapping Accuracies

Although a fully automatic mapping is desired, the system allows for human interventions to the metadata mapping process in order to achieve higher mapping accuracy. When an XML record in a certain schema is input to the system either to merely save the data or to transform to another schema at the same time, the system presents the user for confirmation before any further processing the data values that were extracted from the XML document and have already been mapped to the LOM element set. If the user thinks that there exist any incorrect or inaccurate mappings done automatically by the system, these mapping results can be manually edited as desired before the data is actually saved to the data base or transformed to another target schema.

4.4.2.5 Interoperability of the Data

For the output data from the system to be interoperable to the widest range of systems, XML is used for exchange of metadata instances. We believe that XML is becoming the standard for information deliveries, and in particular, XML is at present the mostly, if not the exclusively, used technology for exchange of LOM metadata (Duval, 2004). An official XML binding of LOM is under development in the IEEE LTSC metadata working group on the basis of the work done by the IMS initiative. The Dublin Core Initiative has also provided guidelines for implementing the Dublin Core metadata in XML (DC XML Guidelines, 2003). The reasons for the wide use of XML in the metadata community come from several aspects: its platform independence which facilitates interoperability that is required by metadata exchange, its appropriateness for expressing hierarchical data structure that most metadata schemas employ, and its ability to allow the exchange of structured data in an accessible yet easy-to-read format. For our framework to be adaptable to as many systems as possible, it should export metadata record in XML format and should be able to take previously generated XML files as input as well.

4.4.2.6 Efficiency of the Data Input

Multiple ways are integrated for input of source metadata, in order to release the user from the heavy burden of metadata creation. Prompting the user to fill out and submit a form is the most ordinary approach to data input and almost all metadata editors provide users this approach. Some metadata editors or management systems, for example, the one developed by Shen, Shi, & Xu (2002), can receive metadata input by uploading XML files, while some others can automatically harvest metadata values from web documents

if provided with the web URL, like the DC-dot metadata editor described in section 4.3. Our system, however, offers an integration of many of these approaches in order to meet the user's demand of maximizing metadata reuse and minimizing the efforts and cost required for metadata creations.

4.4.3 Prototype Functionalities

Based on the proposed metadata interchange framework, a prototype is developed and implemented. Functionally, the prototype consists of four components: a metadata editor, a data storage and management device, a metadata interchange mechanism, and an output unit. The functional composition is shown in figure 4.5.

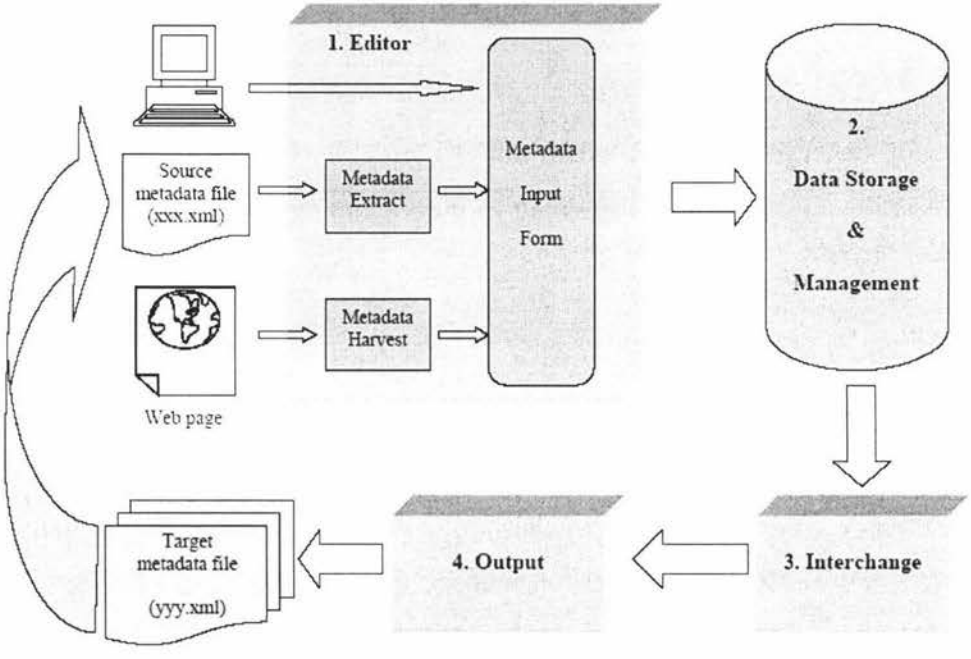


Figure 4.5 Functional compositions of the prototype

The metadata editor is used for data input, as described above, in three ways. In the case of pure keyboard input, the editor simply presents users a blank input form for filling out; in the case of uploading an XML file as input, the editor first extracts metadata values from corresponding XML elements, then presents users the input form on which some fields have been pre-filled with the extracted values as defaults; in the case of providing a web source as input, similarly, the editor harvests as much metadata as possible from the HTML document in the first place, then presents users the input form with default settings coming from the harvested data. In all three cases, necessary syntactic checks are undertaken against the user input when the user intends to submit the form to the system for processing.

In the case of keyboard input from scratch, the editor currently provides three different input forms to type in data: one for the Dublin Core metadata, one for the EdNA metadata, and one for LOM, IMS and any other metadata schemas. This is because the system explicitly covers these four schemas at this stage. LOM and IMS identify the same element set, so they may share the same input form without any problems. Regarding other unknown schemas, as the database is LOM-based and we believe that most of them would be covered by LOM, so by default the input form for LOM is adopted for the input of these schemas. Dublin Core identifies much less elements and has a much simpler data structure than LOM does, so the system provides it a specific data input form in a consideration of user convenience and input efficiency, even though the Dublin Core element set is completely covered by the LOM data model. As for the EdNA metadata, a special input form is essential as it has four special elements that are not available in the LOM data model. As described before, these four extra EdNA elements will be stored in the table that is dynamically created by the system for EdNA only.

The data storage and management device is a relational database management system. It is used for the manipulation of metadata records and user information. In addition to the effective access to the stored metadata, it also facilitates user management such as user registration and user login. The database is composed of a set of tables, about which we will describe more details later. As described above, the whole database system includes two parts: the basic LOM-based structure and the dynamic tables generated during system executions.

The metadata interchange mechanism is the core of the prototype, which transforms a database record into a metadata instance that is conformed to the metadata schema desired by the user. For each target schema, therefore, conceptually exists a database-schema mapping table to define the relationships between the database fields and the metadata schema elements. As examples, the transformations to IMS, EdNA, Dublin Core, and LOM are implemented in the current system; therefore the mappings from the database fields to the four metadata element set are being identified. In addition, the transformation results are required to be encoded in XML format, so reasonably the transformation is implemented by using the World Wide Web Consortium (W3C) Document Object Model (DOM) and the JAXP API. The generated XML file is validated against an XML Schema

file that is predefined through the XML binding of the desired metadata schema. With respect to the underlying technologies, details will be discussed in the next chapter.

The output unit enables the system either to render the resulted XML metadata on the user's screen, or to save it onto the user's local computer, or to export it to other systems. In our implementation, the XML file is simply sent back to the user, who decides how to process it further, no matter whether to save it on the local disk, or to deliver it to somewhere else.

4.4.4 Prototype Architecture

The prototype is a web-based application based on a three-tier model, using Java technologies for implementation, MySQL as the database server, JDBC for database access, and XML for information exchange. Figure 4.6 shows the architecture.

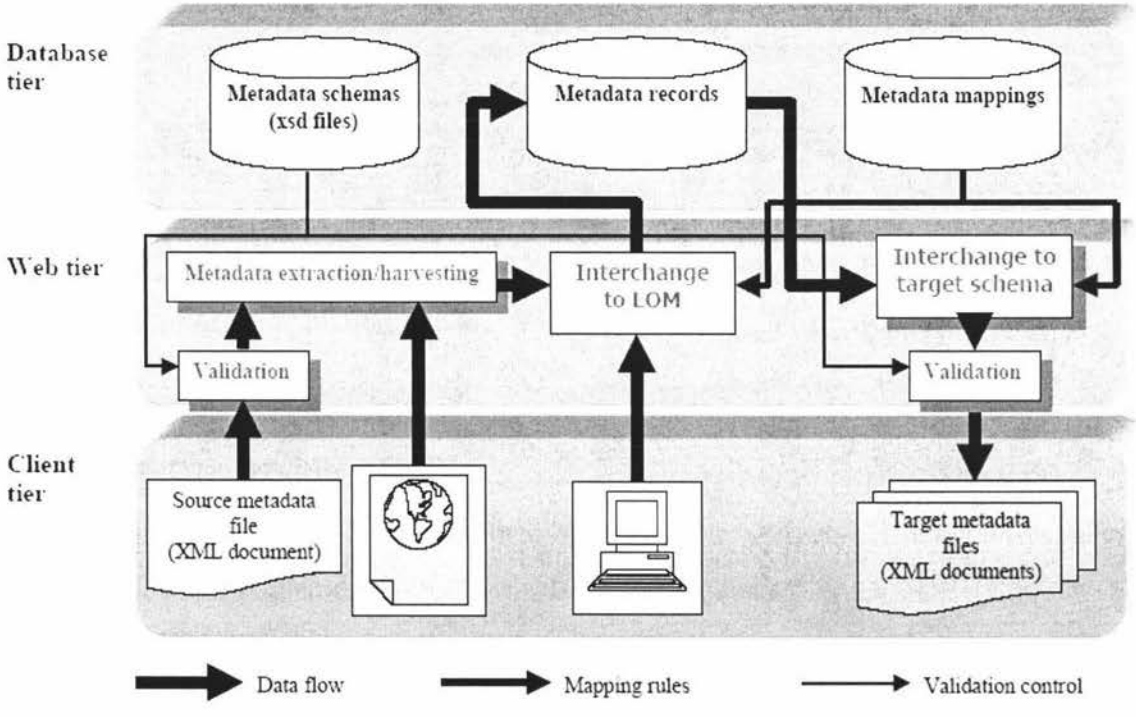


Figure 4.6 Architecture of the prototype

The client tier is a web client (also called thin client), which consists of two parts: dynamic web pages and a web browser. The dynamic pages contain several types of markup languages such as XML and HTML, which are generated from the web tier. The web browser is used to send HyperText Transfer Protocol (HTTP) requests to and receive HTTP responses from the server, and to render the pages coming from the server. Functionally, the client tier provides the interfaces for various interactions between the

user and the system, such as user login, metadata input (in three ways) and XML file display.

The web tier manages all the business logics in the system. It facilitates user logins, validates the data input made by both keyboard typing and XML file uploading, extracts metadata from an uploaded XML file, and harvests metadata from a HTML page. Most importantly, the web tier provides the access to the database server and is responsible for the conversion of metadata from database records to XML instances that are conformed to various schemas as desired. Java technologies are intensively used in the web tier. Servlets, JSPs and Java Beans interact with each other in this tier to undertake different tasks.

In the database tier, a relational database management system, the MySQL databases sever, is used to store and manage the metadata records. As mentioned before, the data in this database is accessed by the Java components in the web tier using the JDBC API, which provides a uniform and vendor-neutral access to relational databases. In addition to the MySQL server for the storage of the metadata records, the database tier conceptually contains the XML schema files against which either the uploaded or the generated XML documents are validated, as well as the mapping rules that determine the relationships between the database fields and the metadata schema elements. Physically, however, the MySQL database, the XML schema files and the above-mentioned mapping rules are kept separately.

4.4.5 Overall Processing Flow

When started, the system presents the user a form for login, or for registration if the user is new to the system. Once the user successfully logs into the system, the system displays to the user the main menu on which a list of titles for all the metadata records owned by the user is shown. From this main menu, the user can select an existing record either for editing (modify it or remove it from the database), or more importantly, for transformation to a user-desired metadata schema in the form of XML format. Also from this main menu, the user can choose any of the three input ways to create a new metadata record and then store the new record into the database. No matter what record-processing procedure the user has chosen from the main menu, the system goes back to the main menu after the chosen procedure is completed. Figure 4.7 shows the overall flowchart (or

screen I/O interfaces) of these processing procedures. It should be noted that all backward links to the main menu from the record-processing procedures are omitted in figure 4.7. Only the forward links are explicitly illustrated.

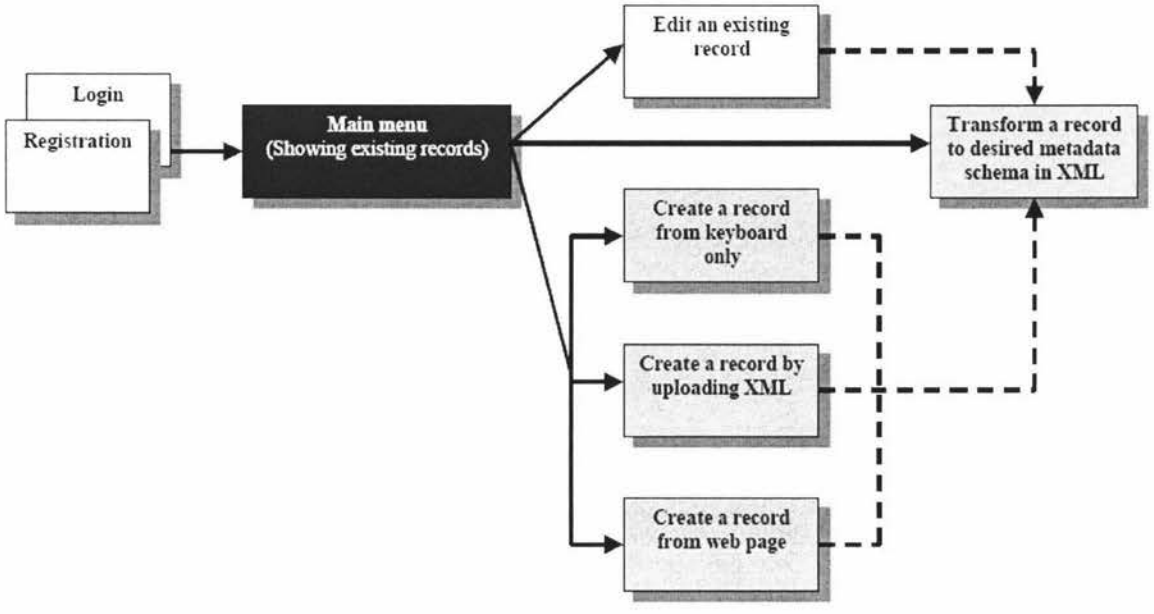


Figure 4.7 The overall flowchart of the processing procedures

The dashed lines in figure 4.7 indicate optional links to the procedure "Transform a record to desired metadata schema in XML". For example, a user can go to the "Edit an existing record" procedure from the main menu. After the user has finished the edit of the selected record, the user can choose either to just save the edit and then directly go back to the main menu or to transform the edited record to the desired metadata schema while at the same time saving the edit. The same option is also available when a new record is created, in any of the three input ways. Because of this option, the user can transform in a straightforward way an XML metadata record in one schema to another XML record in a different metadata schema.

In the procedure "Create a record from keyboard only", the user is allowed to choose one of the three input forms for LOM/IMS/others, Dublin Core and EdNA respectively. These three input forms have been discussed in previous sections. In the procedure "Create a record by uploading XML", the system basically presents the data extracted from the uploaded XML document in a LOM-based edit form for user confirmation and modification before the data is saved to the database or transformed to the target schema. However, if the uploaded XML record is in a schema that contains elements unavailable

in LOM, these extra elements are also displayed on the confirmation form with extra text fields. Similarly, these extra elements will be stored in tables dynamically created for that schema. The same is true for the procedure "Edit an existing record". In this case, an existing record is principally presented in a LOM-based edit form for the user to operate on the record. If the record was originally input as an EdNA record (under the current situation), the edit form will offer extra text fields for the four extra EdNA elements to be displayed and edited. The simplest case is the "Create a record from web page" procedure. The unvaried LOM-based edit form, on which some input fields are set with default values harvested from the Web page, is presented to the user for confirmation and modification.

4.4.6 The Database Design

4.4.6.1 Basic Tables and the Data Fields

As discussed above, the basic database is built around LOM, so in principle all the leaf nodes in the LOM tree are selected as data fields for the database. The container elements in the LOM set are not maintained in the database, because they have no values. To keep the LOM structure, these data elements are allocated in 9 tables, corresponding to the 9 categories in the LOM model. Each metadata record in the database is indexed and identified by a unique ID number that is invisible to the user and is generated automatically by the server machine. In our system, all but the Title element in each metadata record are optional; correspondingly, all but the Title data field in the database can be null values. In addition, each record is associated with the owner (maybe the author or the creator) information by including a data field that stores the email address of the owner. We assume that the email address can be used to uniquely identify a personal individual. We also maintain information in the database about what metadata schema a record originally was in. As discussed in the previous section, if the record was created by keyboard input from empty, the schema for the record was explicitly determined by the user and different input form was used for different schema; if the record was created based on an XML file uploaded by the user, the schema was determined automatically by the system according to the content of the XML file; if the record was created based on data harvested from a Web resource, the schema for the record would be invariably set as "LOM". The purpose of keeping schema information about a record in the database is to decide whether the record has extra elements, accordingly, whether extra text fields are needed when editing this record.

Totally there are 11 tables in the basic database. All the tables and the data fields are defined in table 4.2

Table 4.2 Tables and data fields in the basic database

	Data field	Data type	Null	Default	Extras
Users	Email	Varchar(50)	No		Unique, primary key
	Password	Varchar(8)	No		
	FirstName	Varchar(20)	Yes		
	LastName	Varchar(20)	Yes		
	Address	Varchar(50)	Yes		
	Country	Varchar(20)	Yes		
	Telephone	Varchar(20)	Yes		
MetaIndex	Id	Int(6)	No		Auto-increment, primary key
	Title	Varchar(100)	No		
	UserEmail	Varchar(50)	No		
	Schema	Varchar(20)	No	lom	
General	Id	Int(6)	No		Foreign key
	Identifier	Varchar(255)	Yes		
	Catalog	Varchar(255)	Yes		
	Entry	Varchar(255)	Yes		
	Language	Varchar(50)	Yes		
	Description	Varchar(255)	Yes		
	Keyword	Varchar(255)	Yes		
	Coverage	Varchar(255)	Yes		
	Structure	Varchar(50)	Yes		
	AggregationLevel	Varchar(50)	Yes		
LifeCycle	Id	Int(6)	No		Foreign key
	Version	Varchar(50)	Yes		
	Status	Varchar(50)	Yes		
	AuthorEntity	Varchar(255)	Yes		
	CreateDate	Varchar(50)	Yes		
	PublisherEntity	Varchar(255)	Yes		
	PublishDate	Varchar(50)	Yes		
	ContributeRole	Varchar(50)	Yes		
	ContributeEntity	Varchar(255)	Yes		
	ContributeDate	Varchar(50)	Yes		
Meta Metadata	Id	Int(6)	No		Foreign key
	Identifier	Varchar(255)	Yes		
	Catalog	Varchar(255)	Yes		

	Entry	Varchar(255)	Yes		
	CreatorEntity	Varchar(255)	Yes		
	CreateDate	Varchar(50)	Yes		
	ValidatorEntity	Varchar(255)	Yes		
	ValidateDate	Varchar(50)	Yes		
	MetadataScheme	Varchar(50)	Yes		
	Language	Varchar(50)	Yes		
Technical	Id	Int(6)	No		Foreign key
	Format	Varchar(50)	Yes		
	Size	Varchar(50)	Yes		
	Location	Varchar(255)	Yes		
	OsName	Varchar(50)	Yes		
	OsMinVersion	Varchar(50)	Yes		
	OsMaxVersion	Varchar(50)	Yes		
	BrName	Varchar(50)	Yes		
	BrMinVersion	Varchar(50)	Yes		
	BrMaxVersion	Varchar(50)	Yes		
	InstallRemarks	Varchar(255)	Yes		
	OtherPlatformReq	Varchar(255)	Yes		
	Duration	Varchar(50)	Yes		
Educational	Id	Int(6)	No		Foreign key
	InteractivityType	Varchar(50)	Yes		
	LearningResourceType	Varchar(50)	Yes		
	InteractivityLevel	Varchar(50)	Yes		
	SemanticDensity	Varchar(50)	Yes		
	IntendedEndUserRole	Varchar(50)	Yes		
	Context	Varchar(50)	Yes		
	TypicalAgeRange	Varchar(255)	Yes		
	Difficulty	Varchar(50)	Yes		
	TypicalLearningTime	Varchar(50)	Yes		
	Description	Varchar(255)	Yes		
	Language	Varchar(50)	Yes		
Rights	Id	Int(6)	No		Foreign key
	Cost	Varchar(50)	Yes		
	CopyrightAndOtherRestrictions	Varchar(50)	Yes		
	Description	Varchar(255)	Yes		
Relation	Id	Int(60)	No		Foreign key
	Kind	Varchar(50)	Yes		
	ResourceIdentifier	Varchar(255)	Yes		

	ResourceDescription	Varchar(255)	Yes		
	ResourceCatalog	Varchar(255)	Yes		
	ResourceEntry	Varchar(255)	Yes		
Annotation	Id	Int(6)	No		Foreign key
	Person	Varchar(255)	Yes		
	Date	Varchar(50)	Yes		
	Description	Varchar(255)	Yes		
Classification	Id	Int(6)	No		Foreign key
	Purpose	Varchar(50)	Yes		
	TaxonSource	Varchar(255)	Yes		
	TaxonId	Varchar(255)	Yes		
	TaxonEntry	Varchar(255)	Yes		
	Description	Varchar(255)	Yes		
	Keyword	Varchar(255)	Yes		

4.4.6.2 Dynamic Tables and the Data Fields

The tables dynamically created during the execution of the system are completely schema-specific, so it is impossible for them to be pre-defined in the database system. Rather, they can only be defined and generated by the programming code. Currently, the system only needs to create a dynamic table for the EdNA schema, as only the EdNA schema identifies four elements that cannot be covered by the LOM standard. This dynamic table is named Edna in the system, and table 4.3 shows its details.

Table 4.3 Dynamic table and the data fields for EdNA

	Data field	Data type	Null	Default	Extras
Edna	Id	Int(6)	No		Foreign key
	Approver	Varchar(50)	Yes		
	CategoryCode	Varchar(255)	Yes		
	Entered	Varchar(255)	Yes		
	Indexing	Varchar(255)	Yes		

4.4.6.3 Correlations among Data Tables

The correlations among these data tables (including the basic tables and the dynamic tables) is illustrated in figure 4.8.

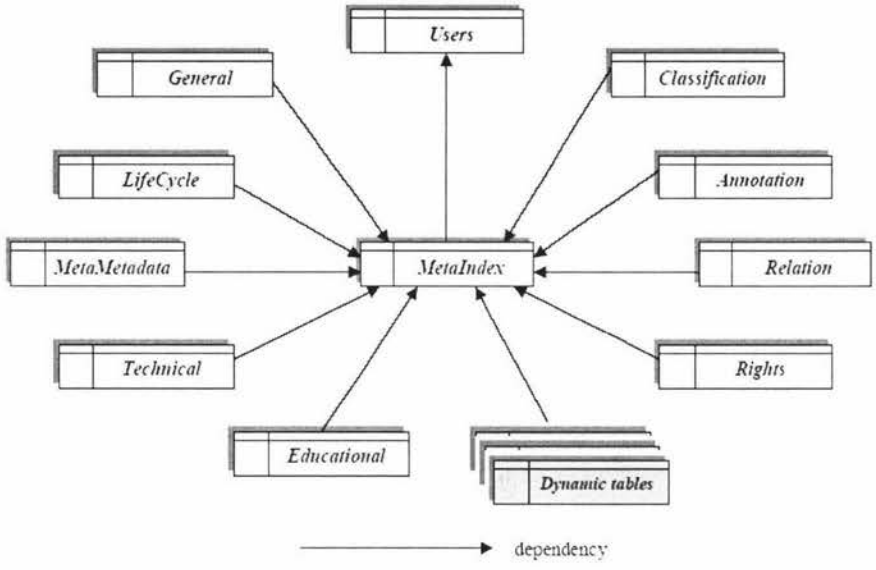


Figure 4.8 Correlations between the data tables

4.4.7 Mappings between the Database and Metadata Schemas

As we discussed before, the LOM-based database is actually acting as the intermediary between the source schema and the target schema, on behalf of the conceptual LOM data model. To input an XML record in certain schema and to output an XML record in another schema, the mapping between the database and the schemas accommodated in the system needs to be identified. The prototype implemented in this thesis accommodates LOM, IMS, Dublin Core, and EdNA at the moment, so we need to map the data fields in the database to the elements in LOM, IMS, Dublin Core, and EdNA respectively. As an example, table 4.4 illustrates the mapping between the database and the EdNA Metadata standard. This table clearly shows the four EdNA elements mapping to the dynamic table Edna.

The mapping for LOM is very simple, as the database itself has been built around the LOM element set. And so is for IMS, as IMS has the same data elements and data structure as LOM does. Since Dublin Core is completely a subset of the EdNA element set, from table 4.4 we can easily find the mapping for Dublin Core as well.

Table 4.4 Mapping between the database and the EdNA Metadata Standard

EdNA element	Database field	
	Table name	Field name
DC. Identifier	General	Identifier Entry
	Technical	Location
DC. Title	MetaIndex	Title
DC. Description	General	Description
DC. Subject	General	Keyword
DC. Publisher	LifeCycle	PublisherEntity
DC. Creator	LifeCycle	AuthorEntity
DC. Date	LifeCycle	PublishDate
DC. Type	Educational	LearningResourceType
DC. Format	Technical	Format
DC. Language	General	Language
DC. Coverage	General	Coverage
DC. Rights	Rights	Description
DC. Relation	Relation	ResourceDescription
DC. Contributor	LifeCycle	ContributeEntity
DC. Source	Relation	ResourceIdentifier (when Kind = "IsBasedOn")
EDNA. Audience	Educational	IntendedEndUserRole TypicalAgeRange
EDNA. Approver	Edna	Approver
EDNA. CategoryCode	Edna	CategoryCode
EDNA. Entered	Edna	Entered
EDNA. Indexing	Edna	Indexing
EDNA. Review	Annotation	Description
EDNA. Reviewer	Annotation	Person
EDNA. Version	MetaMetadata	MetadataScheme
	Or string value "EdNA Metadata Standard Version 1.1"	

4.4.8 XML Bindings and XML Schema Files

4.4.8.1 Introduction

Many metadata schemas, such as LOM and Dublin Core, are dependent on data bindings for the interoperable representation, communication and transport of metadata records. "Binding" refers to the expression of the data model via a formal language or syntax for the purpose of effective data exchange and processing. In most cases such as

the case of LOM and Dublin Core, the general standards used for creating these bindings include RDF and XML.

In our system, XML is used as the binding scheme for all metadata schemas that are manipulated in the system, as mentioned previously. While in XML, both DTD and XML Schemas can be used as the binding languages, which define what elements are allowed in an XML file and in what combinations.

Rather than DTDs, XML Schemas are used in the developed system for the semantic validation of the XML documents. The generation of the XML documents for output, however, is carried out by implementing the W3C DOM specification via the JAXP interface that is bundled with the Java Development Kit (JDK). We use XML Schema as the validation language because it is far more powerful than DTD in that XML Schemas are extensible, support explicitly structural cardinality, data types, namespaces, inheritance and presentation rules (IMS XML Binding, 2001). XML DTDs, on the other hand, cannot be seriously considered as a solution to expressing application profiles, as they do not explicitly support namespaces (Hunter & Lagoze, 2001). This point of view was echoed by DC XML guidelines (2003), which recommends that implementators should base their XML applications of Dublin Core on XML Schemas, not on XML DTDs.

4.4.8.2 XML Binding for IMS Specification

The IMS consortium provides the IMS Learning Resource Meta-data XML Binding Specification (Version 1.2.1) and the XML Schemas as an implementation of the binding specification. These XML Schema files are available for downloading from the IMS Global Learning Consortium website. Among these XML Schema files, the main XML Schema file is the one named imsmd_rootv1p2p2.xsd (the latest version at the writing of this thesis). In the developed prototype, the IMS Learning Resource Meta-data XML Binding Specification is used as guidelines when encoding IMS metadata records in XML format, while the XML Schema files provided by the IMS consortium are employed to validate the uploaded IMS-compliant XML data when the system parses the XML input, as well as the generated IMS-compliant XML data before the system outputs them to users.

Figure 4.9 graphically shows the overall structure of the XML Schema file imsmd_rootv1p2p2.xsd, documented by the XML application tool XMLSPY. The IMS Global Learning Consortium website provides the full content of all the XML Schema files necessary to validate an IMS-compliant metadata record.

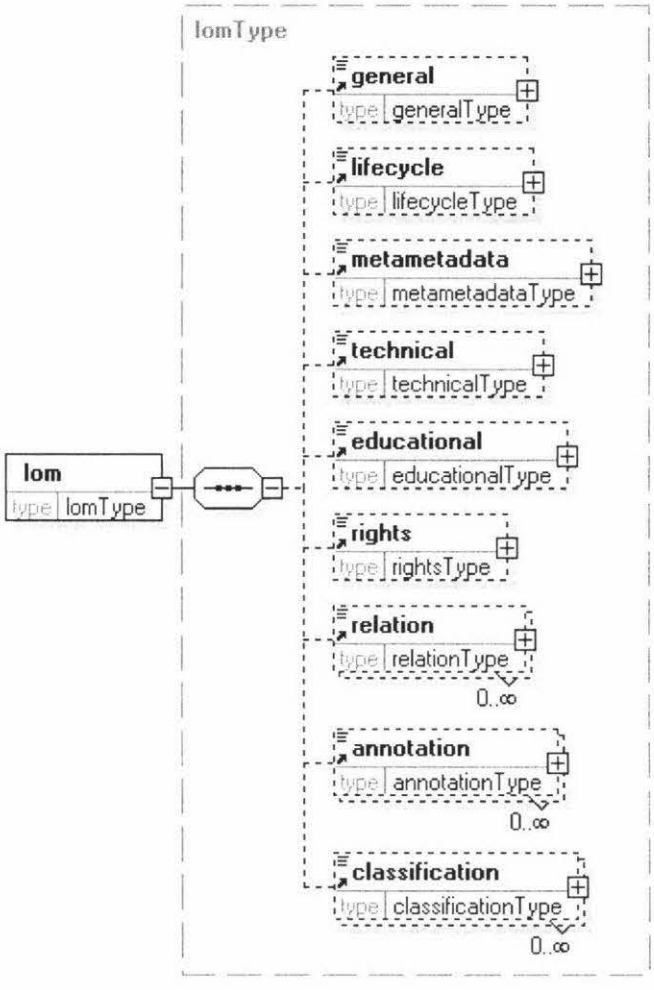


Figure 4.9 The XML Schema file imsmd_rootv1p2p2.xsd

4.4.8.3 XML Binding for LOM Metadata

An official XML binding of LOM is under development in the IEEE LTSC metadata working group on the basis of the work done by the IMS initiative. At the time of writing of this thesis, The IEEE XML Binding P1484.12.3/D5 (2004) is the latest version of the draft standard for XML binding for LOM. We follow this draft standard to encode a LOM-compliant metadata record in XML format, and validate the resulted XML document against the XML Schema file lom.xsd provided by the Resource Discovery Network (RDN) in the UK. The overall structure of lom.xsd in a graphic view is shown in figure 4.10, which is also documented by XMLSPY. The full content of this XML

Schema file can be downloaded from the RDN's Web site at http://www.rdn.ac.uk/oai/lom/20040413/.

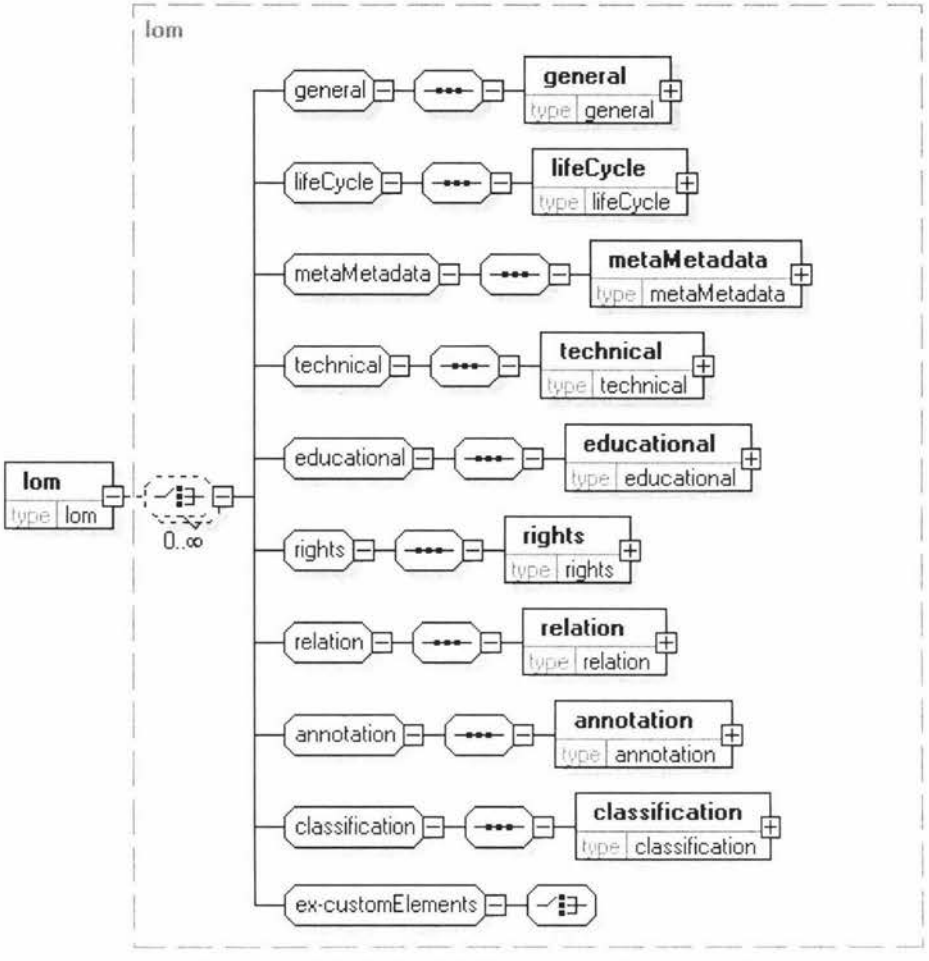


Figure 4.10 The XML Schema file lom.xsd

4.4.8.4 XML Binding for Dublin Core Metadata

Unlike LOM and IMS that formally define an XML binding scheme as part of the specifications, Dublin Core is usually encoded in RDF format so far and has no official binding scheme available for the XML applications. However, there exist guidelines (DC XML Guidelines, 2003) and recommendations (DC XML Recommendations, 2002; DC XML Schema Notes, 2003) suggested by different organizations for implementing Dublin Core in XML. Also, the DCMI Website provides examples of XML Schema files to define the legal structure of a DC-compliant XML record. We follow the DC XML Guidelines (2003) to encode the Dublin Core metadata in XML, and develop our own XML Schema files in order to validate the XML encoding. The main schema file is named dc.xsd, whose overall structure is graphically shown in figure 4.11, similarly

documented by XMLSPY. A summary of all the files in the schema dc.xsd is attached in Appendix B-1.

4.4.8.5 XML Binding for EdNA Metadata

For the EdNA Metadata Standard, currently there are neither XML binding schemes suggested by any parties for XML encoding, nor XML Schema files available for validating or defining an EdNA XML instance. However, as EdNA metadata is inherited from the Dublin Core metadata standard by simply adding eight specific elements to the Dublin Core data set, the guidelines, recommendations and the XML Schema files for Dublin Core provide a very good reference for us to implement EdNA in XML. In this project, we basically follow the DC XML Guidelines (2003) to encode EdNA metadata in XML, taking into consideration the eight extended elements. As a result, an XML-formatted EdNA metadata record will be like the one shown in figure 4.12. Also, XML Schema files, among which the main is named edna.xsd, are developed for the validation of the XML-formatted EdNA records. The overall structure of edna.xsd is graphically illustrated in figure 4.13, also from the documentation generated by XMLSPY. A summary of all the files in the schema edna.xsd is attached in Appendix B-2.

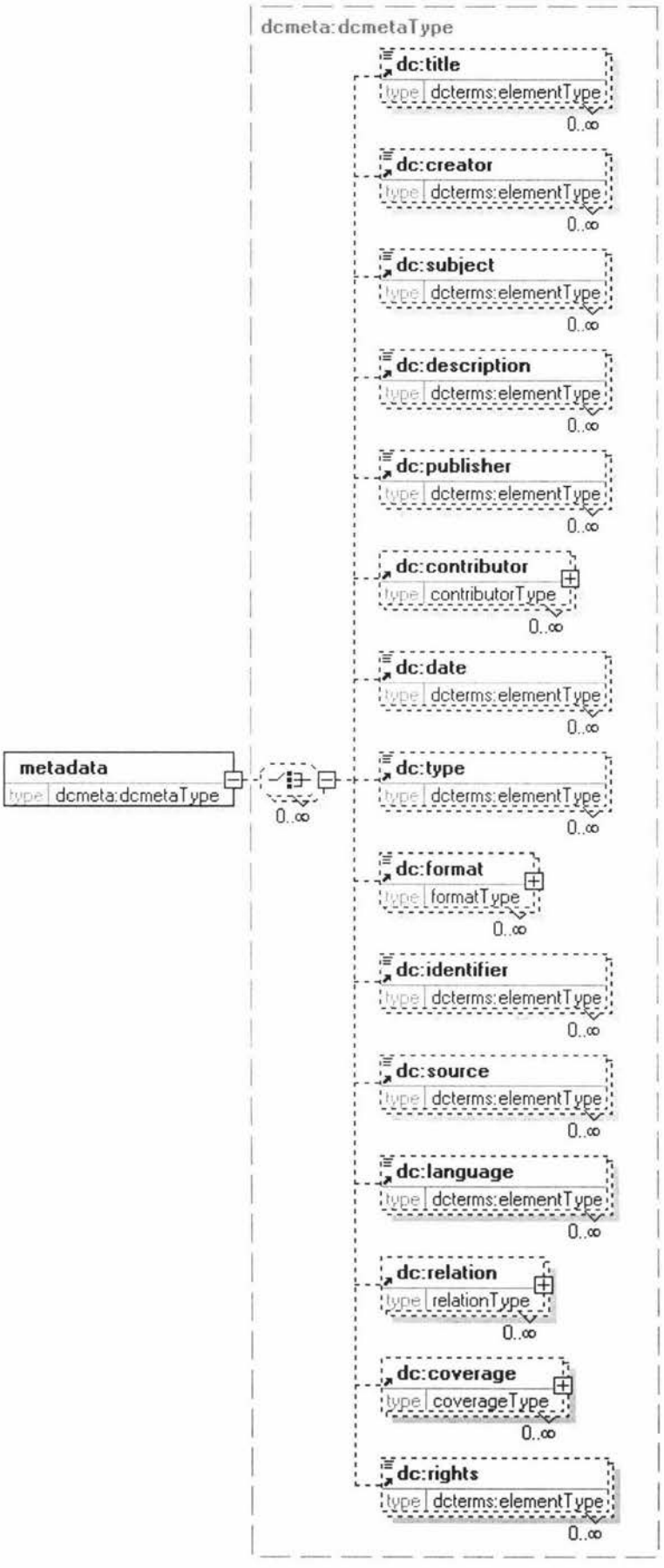


Figure 4.11 The XML Schema file dc.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--File name: ednacreator.xml-->
<!--Generated by Metadata Interchange in Metawork-->
<metadata xmlns="http://localhost:8080/metawork/xsd/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/" xmlns:edna="http://www.edna.edu.au/edna/go/pid/333"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://localhost:8080/metawork/xsd/edna/
http://localhost:8080/metawork/xsd/edna/edna.xsd">
  <dc:identifier xml:lang="en_US">identifier</dc:identifier>
  <dc:title xml:lang="en_US">A complete EdNA test record</dc:title>
  <dc:description xml:lang="en_US">description</dc:description>
  <dc:subject xml:lang="en_US">subject</dc:subject>
  <dc:publisher xml:lang="en_US">publisher</dc:publisher>
  <dc:creator xml:lang="en_US">creator</dc:creator>
  <dc:contributor>
    <dcterms:GraphicalDesigner xml:lang="en_US"
      xsi:type="dcterms:LOMv1.0">contributor</dcterms:GraphicalDesigner>
  </dc:contributor>
  <dc:date>
    <dcterms:issued xml:lang="en_US" xsi:type="dcterms:W3C-DTF">date</dcterms:issued>
  </dc:date>
  <dc:type xml:lang="en_US" xsi:type="dcterms:LOMv1.0">Exercise</dc:type>
  <dc:format xml:lang="en_US" xsi:type="dcterms:MIME">application/excel</dc:format>
  <dc:language xml:lang="en_US" xsi:type="dcterms:RFC1766">en</dc:language>
  <dc:coverage>
    <dcterms:spatial xml:lang="en_US">coverage</dcterms:spatial>
  </dc:coverage>
  <dc:rights xml:lang="en_US">rights</dc:rights>
  <dc:relation>
    <dcterms:IsBasedOn xml:lang="en_US">source</dcterms:IsBasedOn>
  </dc:relation>
  <dc:source xml:lang="en_US">source</dc:source>
  <edna:audience xml:lang="en_US" xsi:type="edna:LOMv1.0">Audience</edna:audience>
  <edna:approver xml:lang="en_US">Approver</edna:approver>
  <edna:categoryCode xml:lang="en_US">category code</edna:categoryCode>
  <edna:entered xml:lang="en_US">entered</edna:entered>
  <edna:indexing xml:lang="en_US">indexing</edna:indexing>
  <edna:review xml:lang="en_US">review</edna:review>
  <edna:reviewer xml:lang="en_US">reviewer</edna:reviewer>
  <edna:version xml:lang="en_US">version</edna:version>
</metadata>
```

Figure 4.12 An example of EdNA record in XML

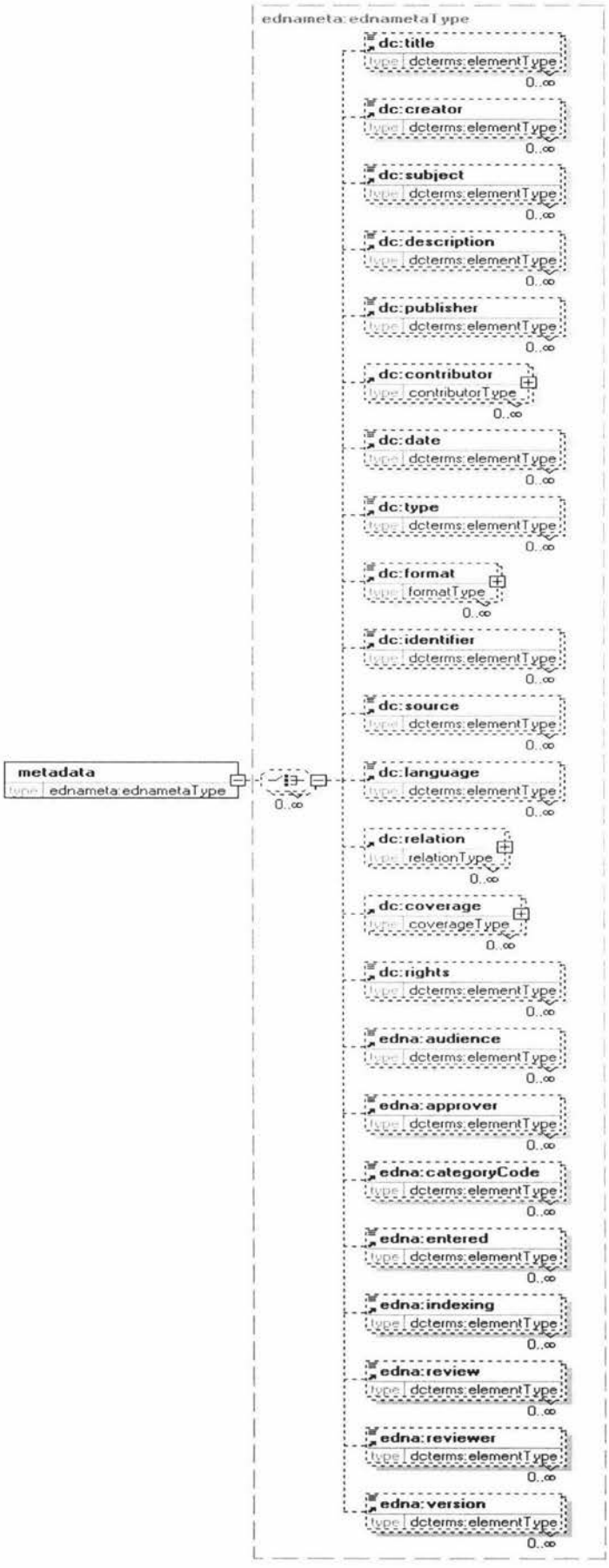


Figure 4.13 The XML Schema file edna.xsd

4.5 Summary

In this chapter, we addressed the issue of metadata interchange on the basis of the survey and discussion in the previous chapter about various metadata schemas. We came into the conclusion that a well-established metadata interchange mechanism is needed not only for cross-repository data search but also for a variety of other purposes. After that, we discussed crosswalks – the element mappings between different metadata schemas, and pointed out that crosswalks are fundamental to successful metadata interchange on one hand, and inherently difficult and error-prone on the other hand. We then had a brief overview on some metadata crosswalks developed by organizations, and presented the crosswalks developed by ourselves for mappings between IMS, EdNA, and GEM.

After investigating several metadata transformation prototypes proposed by researchers and mapping tools developed by organizations, we concluded that currently a flexible, general-purpose means that facilitates conversions among multiple metadata schemas is unavailable, although mapping tools that can convert from one schema to a limited range of other schemas do exist. Therefore, we proposed a general-purpose framework that enables interchanges among a wide variety of metadata schemas, and is highly scalable to be easily extended to accommodate a new schema into the system. This is because we adopt the LOM-intermediated approach that greatly improves the situation of metadata mappings, and the dynamic database methodology that effectively prevents any extra data loss from happening during the mapping to the intermediate schema –LOM. On the basis of the framework, we suggested a prototype that currently carries out the transformation among IMS, LOM, Dublin Core, and EdNA. We believe that XML would be the standard encoding scheme for metadata exchange, so both the source schema (input data) and the target schema (output data) are in XML format. In addition, the system accepts keyboard input for user convenience, and is able to harvest as much metadata as possible from an HTML resource in order to minimize the workload in metadata creation.

A relational database is employed in the system for data storage and user information management. Functionally, the prototype consists of four components: a metadata editor, a data storage and management device, a metadata interchange mechanism, and an output unit. The metadata editor facilitates all three types of data input. The MySQL database is built around the LOM model. The metadata interchange mechanism, the system core, is

implemented using Java technologies. The output unit is responsible for the presentation of the output XML files. With respect to the architecture, the system is a web-based, three-tiered application. Regarding the user/system interactions, the user needs login first. After successful log in, the user is presented with the main menu, where the user can choose the desired processing procedure.

The database design has also been described in this chapter. Based on the LOM template, database tables together with data fields within each table are defined. To generate XML instances in various schemas from this database, the mappings from the database to the desired metadata schemas are identified. Finally, we discussed the XML binding schemes for encoding the metadata in XML format. In particular, we addressed the issue of XML Schemas, against which the XML metadata is validated.

So far we have had the framework for the metadata interchange mechanism, and all the design issues about the prototype development have been addressed. In next chapter, discussion will be focused on the implementation aspects. We will present the used technologies in some more details. We will also discuss the various Java components implemented in the system, the collaborations among theses components, and the deployment of the system. Finally, we will show examples of the implementation results, as an evaluation of the developed prototype system.

Chapter 5 Prototype Implementation and Evaluation

5.1 Implementation Technologies

5.1.1 Introduction

The prototype system is built as a web-based three-tiered architecture and implemented using technologies such as XML for data exchange, Java for logic control, MySQL for database construction, JAXP for XML processing, and JDBC for database access. Among these technologies, XML and Java are the fundamentals. XML, on one hand, provides interoperable, platform-independent data; Java, on the other hand, provides portable, platform-independent software environment. XML and the Java technologies are chosen because they are ideally matched with each other for developing web-based applications. Java technology greatly enhances the benefits offered by XML because of its "Write Once, Run Anywhere" philosophy. In addition, the new J2EE specification defines the standard to ease the development, deployment and management of Java-based applications in the widely accepted Internet environment. The J2EE technologies include API packages such as JDBC and JAXP, as well as components technologies such as Servlets, JSPs and Enterprise Java Beans. We use these J2EE technologies in our system. JDBC is used for access to the MySQL database; JAXP is used as an API to parse, validate and transform XML documents; Servlets and JSPs are used as the server side programming languages to generate and send dynamic responses to a Web browser client; and JavaBeans are used to encapsulate more complex business logic than could be supported in JSP or servlet alone. We use the Tomcat server from the Apache group as the servlet container to implement the servlets and JSPs.

5.1.2 Java Technology

According to SUN Microsystems (http://java.sun.com/index.jsp), Java technology is a portfolio of products that are based on the power of networks and the idea that the same software should run on many different kinds of systems and devices. As the Internet continues to grow, Java has become the most suitable technology for building the new generation of network applications, because Java provides a secure, operating system-neutral and platform-independent environment for quick creation and deployment of Web services. Any computer with a Java Virtual Machine (JVM) can run the same Java byte-code without the necessity to revise or even recompile it. The same is true of the Tomcat

web server, also written in Java. In addition, Java's byte-code gives us mobile code that can be squirted across the network and reconstituted as objects within a remote JVM, perhaps on a completely different operation system (Hunt and Loftus, 2003).

J2EE goes even further by providing a sophisticated set of distribution APIs, with vendors and open source organizations providing implementations. J2EE provides a component-based approach to design, development, assembly, and deployment of enterprise applications. The J2EE platform offers a multi-tiered distributed application model, reusable components, a unified security model, flexible transaction control, and web services support through integrated data interchange on XML-based open standards and protocols (Armstrong, Ball, Bodoff, Carson, Evans, Green, et al., 2004). The multi-tier architecture is illustrated in figure 5.1.

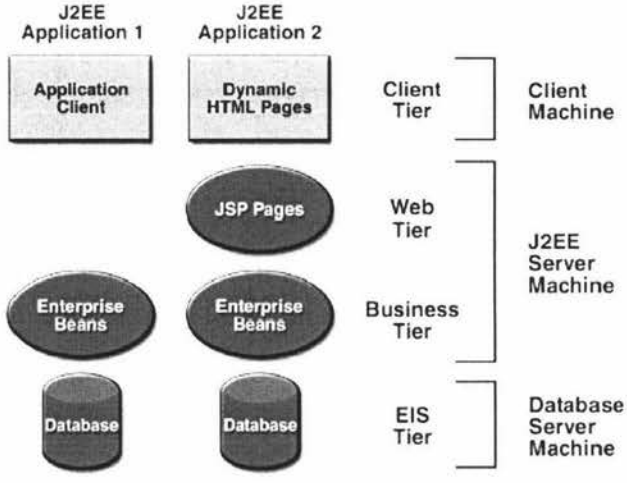


Figure 5.1 J2EE Multi-tier architecture
(Armstrong, Ball, Bodoff, Carson, Evans, Green, et al., 2004)

A J2EE component is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files, and communicates with other components. J2EE defines three types of components: client components including application clients and applets, web components including Java Servlets and JSPs, and business components including Enterprise JavaBeans.

J2EE is a superset of J2SE (Java 2 Platform, Standard Edition). In addition to the standard packages of J2SE, J2EE defines several other packages of its own. The commonly used J2EE packages or APIs are listed in table 5.1.

Table 5.1 Core J2EE packages (Taylor, 2003)

API	Description
JDBC	Provides connectivity to relational databases.
JNDI	Provides Java access to naming services (LDAP, Windows Registry).
XML Processing (JAXP)	Provides for manipulation of XML documents.
Web Services (JAXM)	Provides the ability to send and receive XML messages using protocols such as SOAP.
JSSE	Allows secure SSL communications both as a server and a client.
JCE	Allows common encryption techniques to be used with Java applications.
RMI	Allows Java objects to be invoked remotely using the Remote Method Invocation (RMI) protocol.
Servlets	Provides a web tier component using the HTTP protocol.
JMS	The Java Messaging Service. Provides access to message queues and topics both as a client and as a server.
JSPs	Provides a scripting language for insertion into HTML documents.
Java-IDL	Allows Java applications to interact with CORBA servers.
EJBs	Allows the use of business tier components operating within an abstract container that provides a number of services.
JavaMail	Allows access to email servers using common protocols such as POP3 and IMAP.

The JAXP API and supporting tools enable a simple and quick development of XML-based applications. Through JAXP, XML documents can be easily parsed and processed with Java programming language and deployed in various platforms.

5.1.3 XML Technology

XML has a close relationship to metadata. XML was even once thought of as the metadata language in early years (Maruyama, Tamura, Uramoto, Murata, Clark, Nakamura, et al., 2002). Nowadays, XML metadata has become the de-facto standard for indexing, defining and searching learning objects. The reason for the use of XML in our system as an approach to metadata delivery has been described roughly in previous

chapters and in the introduction section of this chapter. However, it is necessary to discuss a little more about the XML technology itself.

5.1.3.1 Brief Introduction to XML

Developed by the W3C XML Working Group, XML is a text-based markup language that is fast becoming the standard for data interchange on the Web. As with HTML, data is identified with tags, known as markup. But unlike HTML, XML tags identify the content rather than specify the representation about data. In the same way that field names are defined for a data structure, any XML tags are free for use as long as they make sense for a given application. Of course, for multiple applications to use the same XML data, they must agree on the tag names they intend to use. As XML elements describe the data contained in those elements, XML-processing programs can search, sort, manipulate and render XML documents using technologies such as the Extensible Stylesheet Language (XSL).

5.1.3.2 Well-formedness, Validity and XML Parsers

One big difference between XML and HTML is that an XML document is always constrained to be *well formed*. Well-formedness requires that:

(1) Each element must have a start-tag and an end-tag.
(2) Empty element must terminate properly.
(3) XML is case sensitive, so start-tags and end-tags must match.
(4) In XML, attribute values must always be enclosed in either single (') or double (") quotes.

A well-formed document is a *syntactically* correct XML document. An XML document can also optionally reference a document that *semantically* defines that XML document's structure. This document is either a DTD or an XML Schema. When an XML document references a DTD or an XML Schema, some parsers can read the DTD/Schema and check if the XML document follows the structure defined by the DTD/Schema. These parsers are called validating parsers, and the XML document is called valid if it conforms to the DTD/Schema. Parsers that cannot check for document conformity against DTDs/Schemas are called nonvalidating parsers. By definition, a valid XML document is also a well-formed document. Obviously, the data model identified by a metadata schema can be

presented as a DTD or an XML Schema. Any XML document that is valid against this DTD or XML Schema can be thought of as compliant with the metadata schema.

An XML parser is a software program required for processing an XML document. Most parsers are available at no charge and for a variety of programming languages. Examples include Xerces from the Apache XML Project, Crimson that Sun bundles with the JDK 1.4, and much more (Harold, 2002). Parsers support the DOM API or the Simple API for XML (SAX) or both.

5.1.3.3 DOM and SAX

DOM and SAX are different APIs, mainly in that DOM models the XML document while SAX models the parser, in that DOM lends itself to monolithic applications where one program does everything while SAX lends itself to assembly-line like automation where different stations perform small operations on just the parts they have at hand right at the moment, and in that DOM works in a pull model while SAX works in a push model (Harold, 2002). By DOM's pull model, the client program extracts the data it wants from a document by invoking various methods on the document; by SAX's push model, the parser tells the client program what data it reads when it reads the document, and whether or not it is ready for that data. In detail, DOM-based parsers build tree structures containing the entire XML document in memory, so client programs can manipulate the XML data by traversing the DOM tree. SAX-based parsers process XML documents and generate events when the parser encounters tags, text, comments, etc. These events contain data from the XML document so client programs "listen" for these events to obtain data. Comparing these two approaches, DOM allows for interactive applications since the entire object model is present in memory, where it can be accessed and manipulated by the client program. SAX, on the other hand, is more efficient in execution and consumes far less memory capacities, because reading the entire XML structure and holding the object tree in memory as done by DOM are very CPU- and memory-intensive. Which to use in an application is to a large degree dependent upon programmer preference. The only rule that really mandates one or the other is the document size vs. available memory. If the document is too big for the available memory, SAX (or another streaming API) becomes the only choice. We use the DOM approach not only because the DOM's tree structure is more familiar to the author, but also due to the fact that the application requires high interactions for which DOM is much more appropriate than

SAX is. In addition, memory requirement is not a crucial point in the developed system, as a metadata record has only a very limited size.

5.1.4 The JAXP API

5.1.4.1 Brief Introduction to JAXP

We use XML technology for metadata exchange and Java technology for XML data manipulation, so naturally the Java API for XML Processing (JAXP) is the choice for the implementations.

JAXP, defined by Sun and bundled with Java 2 SDK 1.4 (Hunt & Loftus, 2003), is for processing XML data in applications written in the Java programming language. JAXP incorporates both the de facto standard SAX API and the W3C DOM API so that the client program can choose either to process the XML data as a stream of events or to build an object representation of it. JAXP also supports XSLT standard, giving the client control over the presentation of the data and enabling the client to convert the data to other XML documents or to other formats. In addition, JAXP provides namespace support.

JAXP allows for the use of any XML-compliant parser from within the application. It does this with the pluggability layer, which enables a compliant DOM or SAX parser to be plugged in without any visible affect on the JAXP interface. Therefore, the JAXP API is just a front end to those parsers, not a replacement for them. By implementing the JAXP API, the programmer does not need the knowledge of the specific implementation of the parser any more when using a DOM or SAX parser. By default, Sun has bundled the Crimson XML parser into the standard Java class library. We use the default distribution in our applications.

5.1.4.2 Java Packages for XML Processing

As discussed above, JAXP is just an Java interface to XML parsers thus cannot be independent of other APIs such as the DOM API, the SAX API, or the XSLT API. The packages that define these APIs are summarized in table 5.2, according to Armstrong, Ball, Bodoff, Carson, Evans, Green, et al. (2004). The two APIs important for our application are defined in the javax.xml.parsers package and the javax.xml.transform package. These two packages contain three vendor-neutral factory classes:

SAXParserFactory, DocumentBuilderFactory, and TransformerFactory. The three factory classes offer the client program a SAXParser, a DocumentBuilder, and an XSLT transformer, respectively. The DocumentBuilder, subsequently, creates a DOM-compliant Document object. We use DOM API, so the process of how to instantiate the DocumentBuilderFactory class and eventually create a Document object for the XML data to be processed will be discussed later.

Table 5.2 Java packages for XML processing

Package	Description
javax.xml.parsers	The JAXP APIs, which provide a common interface for different vendor's DOM and SAX parsers.
org.w3c.dom	Defines the Document class (a DOM) as well as classes for all the components of a DOM.
org.xml.sax	Defines the basic SAX APIs.
javax.xml.transform	Defines the XSLT APIs that allows XML to be transformed into other forms.

5.1.4.3 Implementing DOM via JAXP

To implement a DOM parser with the JAXP API, a DocumentBuilder instance must be obtained by using the javax.xml.parsers.DocumentBuilderfactory class, and then that instance is used to create a Document object that conforms to the DOM specification. This process can be graphically illustrated in figure 5.2. A fragment of Java code for parsing a user-uploaded XML document by implementing DOM via JAXP is shown in figure 5.3. Figure 5.4 shows a similar case but with a little difference to figure 5.3, in that a new XML document is to be created from the database rather than an existing XML document is parsed. It is clear from figure 5.3 and 5.4 that three basic steps are necessary to build the tree-structured Document object in the memory:

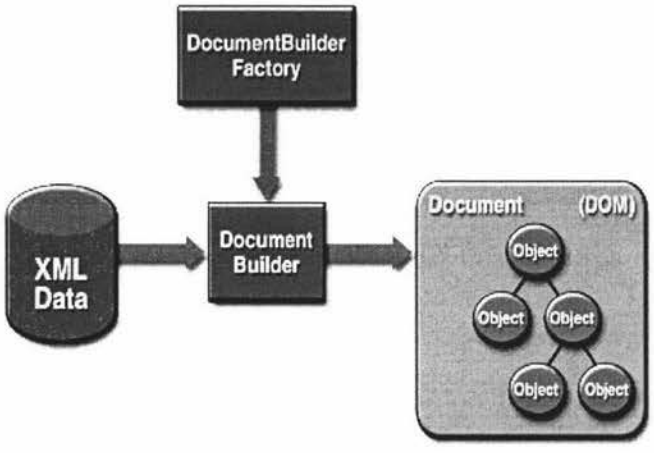


Figure 5.2 DOM Implementation via JAXP
(Armstrong, Ball, Bodoff, Carson, Evans, Green, et al., 2004)

```
package beans;

import org.w3c.dom.*; //import W3 definition for DOM and DOM exceptions
import javax.xml.parsers.*; //import JAXP APIs
   ...

public class FileRecordBean {
   ...
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    dbf.setValidating(true);
    dbf.setNamespaceAware(true);
    DocumentBuilder db = dbf.newDocumentBuilder();
    db.setErrorHandler(errorHandler);
    doc = db.parse(FILE_NAME);
   ...
```

Figure 5.3 Example Java code for parsing XML using DOM via JAXP API

```
package servlets;

import org.w3c.dom.*;
import javax.xml.parsers.*;
   ...

public class EdNACreator extends HttpServlet {
   ...
   DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
   DocumentBuilder db = dbf.newDocumentBuilder();
   Document doc = db.newDocument();
   ...
```

Figure 5.4 Example Java code for creating XML using DOM via JAXP API

1. Obtain a DocumentBuilderFactory object

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

2. Create a DocumentBuilder object

```
DocumentBuilder db = dbf.newDocumentBuilder();
```

3. Create a Document object into memory by parsing an XML file or from empty

```
Document doc = db.parse(FILE_NAME); // by parsing an existing XML file, or
Document doc = db.newDocument(); // from empty
```

Now that the tree-structured doc object has been built, the client program can invoke various methods in the Document and Node class to traverse the tree, to add a node, to delete a node, or to modify a node. After finishing the desired manipulations to the doc object, this object can be serialized to an XML file for storage or to an output stream for presentation as XML format.

5.1.5 The JDBC API

5.1.5.1 JDBC Structure

The Java Database Connectivity (JDBC) API provides Java applications a vendor-neutral, platform-independent access to relational databases using SQL-based queries. Much like the JAXP API to XML parsers, JDBC API provides a common front end to different database drivers but dos not itself provide an actual connection to a database. Instead, JDBC provides a uniform way for a Java program to connect with a variety of databases in a general manner without having to deal with the specifics of those database systems. The overall structure of JDBC is shown in figure 5.5.

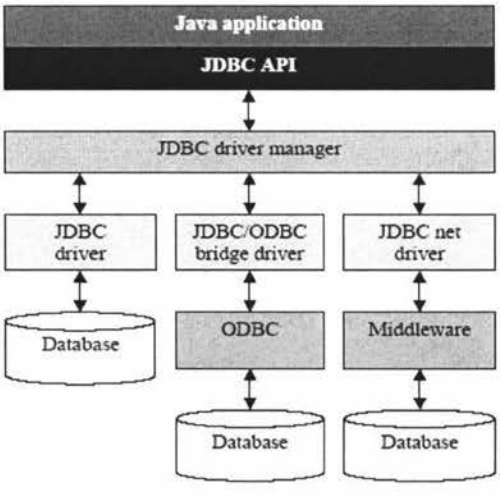


Figure 5.5 The JDBC library structure (Harms, 2001)

Via the JDBC API, the application makes a connection with the database, retrieves and updates data, executes commands on the data source, and finally closes the connection. At the other end, the database drivers connect either directly to a specific database, or to another protocol such as the Open DataBase Connectivity (ODBC) or middleware product. The JDBC API and the driver manager are part of the Java JDK, while the JDBC drivers come from database vendors or third parties.

5.1.5.2 JDBC Interfaces/Classes

The java.sql package is the JDBC core package, in which various interfaces/classes are defined. The JDBC database driver is created by implementing these interfaces/classes by the database vendor or third parties. The commonly used JDBC interfaces/classes in most applications as well as in our implementations for database connections and data retrievals are: the DriverManager class, the Connection class, the Statement class, the PreparedStatement class, and the Resultset class. Theses five JDBC classes are briefly summarized in table 5.3.

Table 5.3 Core JDBC classes

Interface/Class	Description
DriverManager	Loads the JDBC driver and manages the database connection. Controls access to the database driver.
Connection	Connects to the database and allows the creation of Statement or PreparedStatement object.
Statement	Allows the execution of SQL statements and returns results, either integer values indicating the number of rows updated or ResultSet objects representing the rows returned by the execution of a SQL select statement.
PreparedStatement	Similar to the Statement class in that it allows the execution of SQL statements. But unlike the Statement class, it should be pre-compiled and stored in the database and thus be able to be executed multiple times in an efficient way.
ResultSet	Allows access to individual rows in the set of rows returned by a SQL select statement query.

```
package beans;

import java.sql.*;
   ...

public class LoginBean {
   ...
  try {
   Class.forName("com.mysql.jdbc.Driver").newInstance();
   con = DriverManager.getConnection(
      "jdbc:mysql://ispost.massey.ac.nz/MetaDB");
  } catch (Exception ex) {
   System.out.println("Exception occurs when connecting to database: " +
                ex.getMessage());
  }
   ...
  try {
   stmt = con.prepareStatement(
      "SELECT Password FROM Users WHERE Email LIKE ?");
   stmt.setString(1, email);
   rs = stmt.executeQuery();
   if (rs.next()) {
     String pw = rs.getString(1);
     if (!pw.equals(password)) {
       errMsg = "Incorrect password! Check your password carefully and try again.";
     }
   }else {
     errMsg = "This email has not been registered yet! " +
        "Check your email and try again, or register as a new user email.";
   }
   rs.close();
   stmt.close();
  }catch (Exception ex) {
   System.out.println("Exception occurs: " + ex.getMessage());
  }
   ...
  try {
   con.close();
  }catch (Exception ex) {
   System.out.println("Exception occurs when closing database connection: " +
                ex.getMessage());
   }
  ...
```

Figure 5.6 Example Java code for implementing JDBC to access the database

5.1.5.3 Using JDBC

The basic steps for a simple data retrieval operation with JDBC are as follows.

1. Load the vendor specific driver

   ```
   Class.forName("com.mysql.jdbc.Driver").newInstance();
   ```

2. Create the database Connection object using the DriverManager

```
Connection con = DriverManager.getConnection(
        "jdbc:mysql://ispost.massey.ac.nz/MetaDB");
```

3. Create a Statement or PreparedStatement object using the Connection object

```
PreparedStatement stmt = con.prepareStatement(        // This is a PreparedStatement
        "SELECT FirstName,LastName FROM Users WHERE Email LIKE ?");
stmt.setString(1, userEmail);
```

4. Execute the Statement or PreparedStatement object to return a ResultSet object

```
ResultSet rs = stmt.executeQuery();
```

5. Iterate through the ResultSet object to access the query results

```
while (rs.next()) {
...
}
```

6. Close the ResultSet, Statement object and the database Connection object

```
rs.close();
stmt.close();
con.close();
```

A fragment of Java code implemented in our applications is shown in figure 5.6 as an example for using JDBC to access databases.

5.1.6 Java Servlets, JSPs, and JavaBeans

5.1.6.1 Introduction

As discussed in section 5.1.2, Java Servlets, JSPs, and JavaBeans are all J2EE components, hosted on the J2EE server machine. Java Servlets and JSPs are provided by J2EE for creating dynamic web page content, usually in HTML format. They together form the Web tier of the J2EE architecture. Java Servlets enhance the functionality of World Wide Web servers and are the most common form of servlet today. JSPs are a powerful and convenient way to implement the request/response Web model without getting into the lower-level details of servlets. Historically, Java Servlets came first and JSPs are built on top of servlet technologies. When a JSP-enabled server receives the first request for a JSP, the JSP container translates that JSP into a Java Servlet that handles the current request and future request to the JSP. In fact, Java Servlets and JSPs can be used interchangeably in an application. When most of the content sent to the client is generated dynamically by Java code, and only a small portion of the content is static text and

markup, Java Servlets are used; otherwise JSPs are the choice. In some cases, servlets do not produce content at all. Rather, they perform a task on behalf of the client, and then invoke other servlets or JSPs to provide a response. Generally, servlets are the right solution for database-intensive applications, in which minimal client-side support is required. The server that executes a servlet (or a JSP) is referred to as the servlet container.

5.1.6.2 Java Servlets

Servlets are Java classes developed by implementing the Serlvet interface. All methods in the Serlvet interface are invoked automatically by the servlet container. Among these methods, the key one is the service method, which receives both a ServletRequest object and a ServletResponse object. These objects provide access to input and output streams that allow the servlet to read data from the client and send data to the client. A servlet's life cycle begins when the servlet container loads the servlet into the memory, in response to the first request to that servlet. First of all, the servlet container invokes the servlet's init method. Then the servlet can respond to its first request. The service method handles all requests, and the destroy method is called to release servlet resources when the servlet container terminates the servlet. The servlet packages define two abstract classes to implement the Servlet interface – GenericServlet and Httpservlet. Web-based servlets like the ones implemented in our system extend class Httpservlet, which overrides the service method to distinguish between the typical requests (also known as request methods) received from a client web browser. The most common HTTP requests from a client are the get method and the post method, to which the class HttpServlet defines methods doGet and doPost to respond respectively. Methods doGet and doPost are invoked from the service method that is called when a request arrives at the server. In addition, Methods doGet and doPost receive as arguments an HttpServletRequest object and an HttpServletResponse object that enable interactions between the client and the server. Finally, a response is sent to the client through a PrintWriter object returned by the getWriter method of the HttpServletResponse object. Figure 5.7 shows code fragment from the servlet class IMSCreator, which produces and sends to the client browser an IMS metadata record in XML format.

```
package servlets;

import javax.servlet.*;
import javax.servlet.http.*;
   ...

public class IMSCreator extends HttpServlet {
   ...
 public void doPost(HttpServletRequest request, HttpServletResponse response)
              throws ServletException, IOException {
   this.request = request;
    this.response = response;
    ...
     DOMSource source = new DOMSource(doc);
     PrintWriter pw = response.getWriter();
     StreamResult result = new StreamResult(pw);
      transformer.transform(source,result);  //put the doc obj to output stream to send to the client
  ...
```

Figure 5.7 Example Java code from the servlet class IMSCreator

5.1.6.3 JSPs

JSPs are an extension of servlet technology. Unlike Java Servlets, which both include the HTML to be rendered by the client browser and contain the logic that produce that HTML, JSPs attempt to separate the presentational aspects from the logical aspects, so that people even unfamiliar with Java programming language can develop web pages with enhanced dynamic content and powerful processing capabilities. In detail, JSPs enable web application programmers to create dynamic content by reusing pre-defined components and by interacting with components using server-side scripting. Regarding the file format, a JSP page is essentially an HTML file that uses a .jsp extension and provides some additional JSP tags. Therefore, JSPs focus on the HTML tags while the Java code called from them deals with the business logic.

There are four key components to JSPs: directives, actions, scriptlets, and tag libraries. Directives specify global information that is not associated with a particular JSP request. Actions encapsulate functionality in pre-defined tags that programmers can imbed in a JSP. Scriptlets enable programmers to insert Java code that interact with components in a JSP to perform request processing. Tag libraries are part of the tag extension mechanism that enables programmers to create new tags that encapsulate complex Java functionality. The request/response mechanism and life cycle of a JSP are the same as those of a servlet, because JSPs are based on the servlet technology.

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<%@ page contentType="text/html; charset=GBK" %>
<%@ page errorPage="NewRecord_error.jsp" %>
<%@ page import = "beans.RecordBean" %>
<%@ page import = "beans.OpRecordBean" %>
<jsp:useBean id="recordBeanId" scope="page" class="beans.RecordBean" />
<jsp:useBean id="opRecordBeanId" scope="page" class="beans.OpRecordBean" />

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
    <title>View/Edit Metadata Record</title>
  ...
<body>
<jsp:setProperty name="recordBeanId" property="*" />
<jsp:setProperty name="opRecordBeanId" property="*" />
  ...
<%
if (request.getParameter("Delete") == null) { // ***** Save button pressed *****
 if (!recordBeanId.checkTitle()) {
%>
<p align="center"><b><font face="Tahoma" size="6" color="#008000">View/Edit Metadata
Record</font></b></p>
  ...
<%
        }
}
}
else { // ***** Delete button pressed *****

        opRecordBeanId.deleteRecord(recordId);
%>
        <jsp:forward page ="RecordsList.jsp">
            <jsp:param name = "email"
                        value = "<%= userEmail %>" />
        </jsp:forward>
<%
}
%>
</body>

</html>
```

Figure 5.8 Example code from the JSP page EditRecord.jsp

Figure 5.8 shows code fragments from a JSP page EditRecord.jsp written in our applications. The page directive defines information that is globally available in a JSP. Directives are delimited by <%@ and %>. The page directive's errorPage attribute indicates where all uncaught exceptions are forwarded for processing. The page directive's import attribute enables programmers to specify Java classes and packages that

are used in the context of a JSP. Action <jsp:useBean> enables a JSP to manipulate a Java object. This action can be used to create a Java object for use in the JSP or to locate an existing object. Action <jsp:setProperty> has the ability to match request parameters to properties of the same name in a bean by specifying "*" for attribute property. Correspondingly, there is another action <jsp:getProperty> that obtains the value of JavaBean's property. Action <jsp:getProperty> has two attributes – name and property – that specify the bean object to manipulate and the property to get. Action <jsp:forward> enables a JSP to forward the processing of a request to a different resource. Processing of the request by the original JSP terminates as soon as the request is forwarded. Action <jsp:forward> is usually accompanied by another action <jsp:param>, which specifies name/value pairs of information that are passed to the <jsp:forward> action. Every <jsp:param> action has two required attributes – name and value.

5.1.6.4 JavaBeans

Java code can be embedded in a JSP by using Java scriptlets, which become part of the servlet when the servlet container compiles the JSP. As powerful and useful as scriptlets are, the scriptlet approach still has some disadvantages. The more Java code is embedded in the JSPs, the more often the Java developer has to wrestle the page designer for access to the page (Harms, 2001). In addition, it would be more difficult to read the Java code when it is being buried in HTML. The JavaBean is the solution we used to overcome these disadvantages. The underlying basic idea is: since Java scriplet code executes on the server, it is possible to put this code into an independent class and then call that class when necessary. By doing so, the scriplet code can be separated from the page development, and what is more, the Java code can be reused. A JavaBean is nothing more than an ordinary Java class, but to be used as a bean component, it has to conform to the standards established for JavaBeans, Java's component architecture. Figure 5.9 shows the simplest JavaBean class implemented in our system. For the purpose of servlet and JSP development, the following points of JavaBean specification are important (Harms, 2001).

1. JavaBeans should not have any public properties. All properties should be marked private and be only visible to this class.
2. Any of the JavaBean's properties that do need to be exposed should have getter and setter methods, as necessary.
3. All JavaBeans must have a no-argument constructor method.

```
package beans;

public class RecordIndexBean {
  private int id;
  private String userEmail,title;

  //Setter method
  public void setId(int id) {
    this.id = id;
  }

  public void setTitle(String title) {
    this.title = title;
  }

  public void setUserEmail(String userEmail) {
    this.userEmail = userEmail;
  }

  //Getter method
  public int getId() {
    return id;
  }

  public String getTitle() {
    if (title != null)
      return title;
    else
      return "";
  }

  public String getUserEmail() {
    if (userEmail != null)
      return userEmail;
    else
      return "";
  }
}
```

Figure 5.9 A JavaBean example – RecordIndexBean

5.2 Main Java Components in the Application

The Java components implemented in the developed system are devided into three categories: Java Servlets, JSPs and JavaBeans. These Java components collaboratively interact with each other to manage the logic controls and generate the responses to the user, in order to facilitate the functionalities that are required by the system design. The main Java Servlets, JSPs, and JavaBeans are summarized in table 5.4. For the convenience of system management, the Java Servlets are put in the servlets package, and the JavaBeans are put in the beans package.

Table 5.4 The main Java components implemented in the system

Type	Name	Description
Java Servlets	IMSCreator	Retrieves a record from the database and transforms the record to IMS schema in XML format, if given the record ID.
	EdNACreator	Retrieves a record from the database and transforms the record to EdNA schema in XML format, if given the record ID.
	DCCreator	Retrieves a record from the database and transforms the record to Dublin Core schema in XML format, if given the record ID.
	LOMCreator	Retrieves a record from the database and transforms the record to LOM schema in XML format, if given the record ID.
JSPs	Login	Facilitates user login by presenting the login form.
	Register	Facilitates user register by presenting the register form.
	RecordsList	Presents the user the main menu with a list of titles for all existing records, allows the user to choose a desired processing procedure.
	RecordProcessing	Forwards the processing of the user request to a different Java component according to whether the user chooses to edit a record or transform a record.
	EditRecord	Facilitates the edit of an existing record by presenting the edit form. It is possible to transform the edited record to the desired metadata schema after saving the edit.
	NewRecord_DC	Facilitates the creation of a new record in Dublin Core from empty through keyboard input, by presenting a blank DC-based input form. It is possible to transform the created record to the desired metadata schema after saving the new record.
	NewRecord_EdNA	Facilitates the creation of a new record in EdNA from empty through keyboard input, by presenting a blank EdNA-based input form. It is possible to transform the created record to the desired metadata schema after saving the new record.
	NewRecord_LOM	Facilitates the creation of a new record in LOM and other schemas from empty through keyboard input, by presenting a blank LOM-based input form. It is possible to transform the created record to the desired metadata schema after saving the new record.
	NewRecordXml	Facilitates the creation of a new record from another metadata record in XML format, by uploading the XML file plus necessary manual editing. It is possible to transform the created record to the desired metadata schema after saving the new record.

	NewRecordWeb	Facilitates the creation of a new record from a web resource, by providing the URL for the web resource plus necessary manual editing. It is possible to transform the created record to the desired metadata schema after saving the new record.
JavaBeans	DatabaseBean	Gets a connection to the database.
	RecordIndexBean	Defines a record index, including the three elements (id, userEmail, title) and the setters/getters.
	RecordBean	Defines a class representing the basic record in the database.
	OpRecordBean	Connects to the database to manipulate (retrieve, delete, save, update) a record, given the record ID.
	EdnaRecordBean	Defines a class representing the four extra EdNA elements.
	OpEdnaRecordBean	Connects to the database to manipulate (retrieve, delete, save, update) the four extra EdNA elements, given the record ID. Creates the Edna table in the database dynamically.
	LoginBean	Connects to the database and checks the legitimacy of the user input for login. Returns error message when incorrect otherwise returns blank error message and the user name.
	RegisterBean	Connects to the database and checks legitimacy of the user input for registering a new user. Returns error message when incorrect otherwise returns blank error message and saves the new user.
	RecordsListBean	Connects to the database to retrieve all the records owned by the user, given the user email. The retrieved records are returned in a list.
	FileUploadBean	From an input stream, extracts the value when a field is encountered; or saves as a temporary XML file on the server when an uploaded file is encountered.
	FileRecordBean	Creates and returns a RecordBean object with data retrieved from the temporary XML file produced by the FileUploadBean.
	WebRecordBean	Creates and returns a RecordBean object with data retrieved from a web page, given the web URL.

The collaborations between these Java components are shown in figure 5.10. Similar to figure 4.7 in chapter 4, the backward links such as the link from the RecordProcessing.jsp to the RecordsList.jsp (the main menu) are omitted for better readability. Also, two servlets DCCreator and LOMCreator are not shown in this figure due to the space limitation. For similar reasons, the collaborations between the JavaBeans

themselves are not presented in figure 5.10. Instead, the collaborations between the JavaBeans are illustrated separately in figure 5.11.

One of the advantages offered by the J2EE specification is the maximal reuse of Java components and the intensive sharing of programming code. In figure 5.10, we can clearly see how the JavaBean components such as RecordBean and OpRecordBean are efficiently used/reused by different other components – JSPs and Java Servlets. Of course, JavaBean components can also be used/reused by other JavaBeans. Figure 5.11 clearly illustrates the use/reuse of DatabaseBean by other bean components.

From figure 5.10 and 5.11, we can also see the flexibility and the scalability of the system. Each time when we want to extend the system to accommodate another metadata schema, we just need to write one more Java Servlet class corresponding to this schema and to request (or forward request to) this servlet from related JSPs, without the need to make changes to the whole system. In fact, after the system accommodating only IMS and EdNA was completed, we added to the system another two schemas: Dublin Core and LOM. We made this extension just by adding another two servlets: the DCCreator and the LOMCreator, correspondingly. We could simply do so because neither Dublin Core nor LOM contains extra elements, however, if the newly added schema contains extra elements that are not available in the LOM data model, extra database table(s) needs to be dynamically created by the program to accommodate the extra elements, as discussed before. Consequently, JavaBean components, such as the EdnaRecordBean and the OpEdnaRecordBean in the case of EdNA, must also be added to the system to manipulate the extra elements. Fortunately, these new components are not tightly coupled with the existing components in the system. Rather, they are relatively independent of the existing system. When a new schema is to be accommodated, we basically add new components to the system, rather than largely modify the existing components.

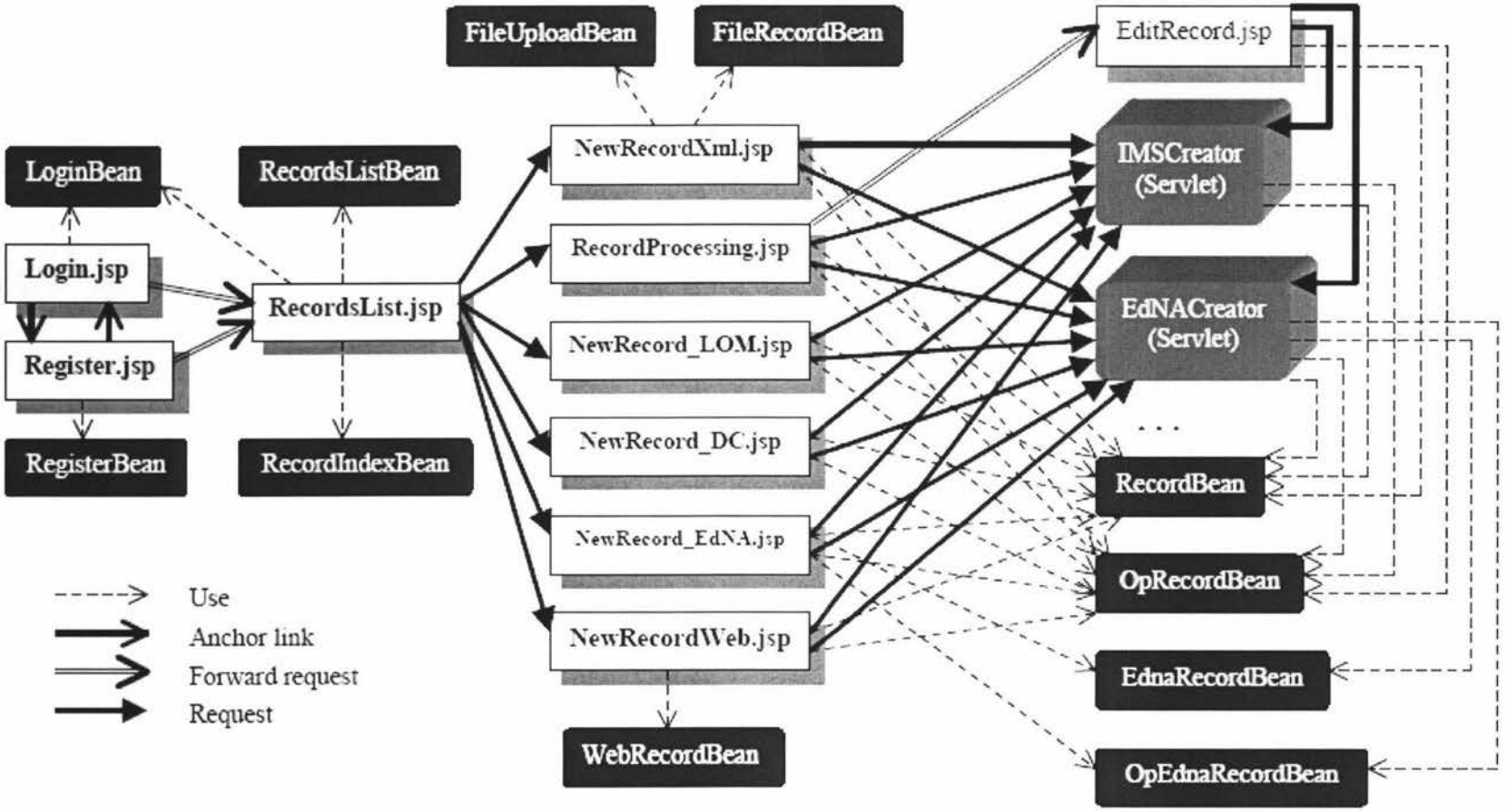


Figure 5.10 Collaborations between the main Java components

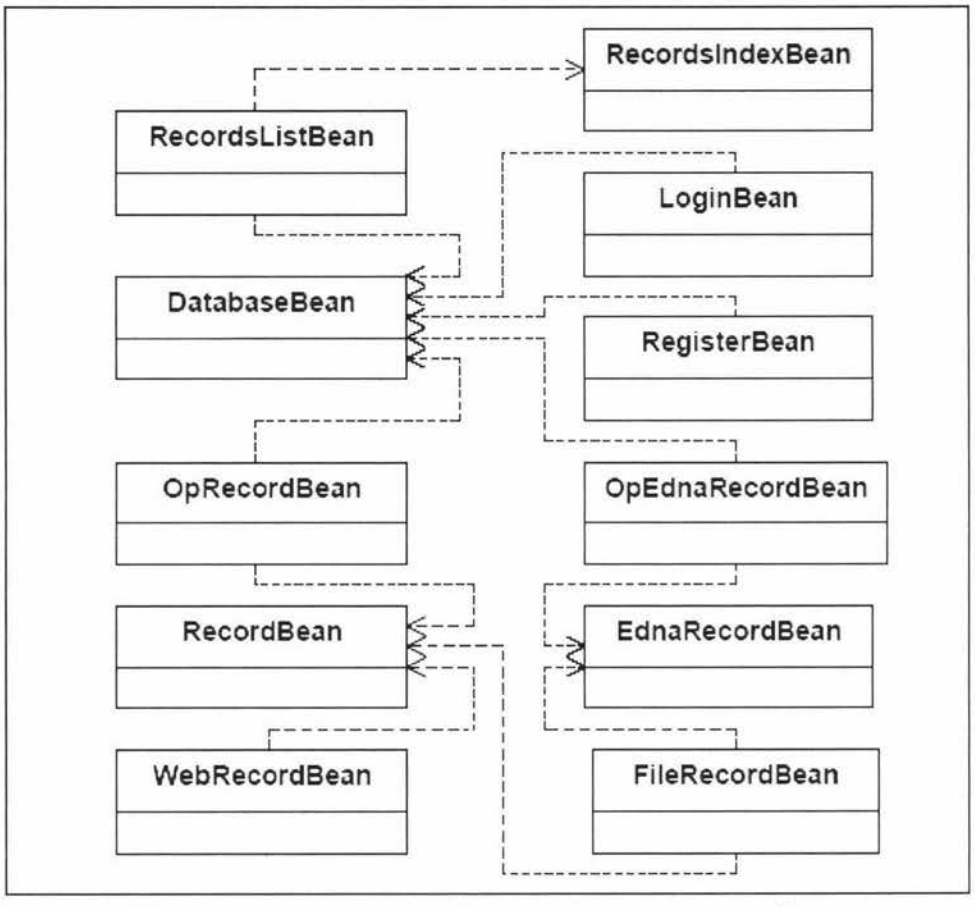


Figure 5.11 Class diagram for the Java bean classes in the beans package

5.3 Description of Java Components

In section 5.2, we had a total view of the Java components and described the interactions among them. To discuss each component in detail is neither possible nor necessary, but some details about the several important or typical Java components will be given here, in order that we may have a more concrete understanding about what properties and methods they contain, what components and other Java library classes they interact with, and how. Although we believe all the Java components summarized in section 5.2 are indispensable to the system, the followings would need some further discussions.

5.3.1 The RecordBean and EdnaRecordBean Classes

The JavaBean class RecordBean defines an object representing the data record manipulated in the system. As shown in figure 5.10, almost every operation of the system, no matter whether it is the creation of a new record, the edit of an existing record, or the transformation of a record to the desired metadata schema, must instantiate a RecordBean object at the first place and then access the RecordBean objet for various

data manipulations. The Uniform Modeling Language (UML) view of the RecordBean class is shown in figure 5.12, from which the class properties, the methods, and the correlations with other Java components are clearly illustrated. The correlations in figure 5.12 indicate that the RecordBean is heavily used by the servlets and the JSPs (the same as shown in figure 5.10), as well as by other JavaBeans (the same as shown in figure 5.11) in the system.

Another JavaBean class, EdnaRecordBean, is very similar to RecordBean in terms of structure, but much simpler than RecordBean from the viewpoint of content. This is because RecordBean basically represents the whole LOM data set, while EdnaRecordBean only deals with the four specific EdNA elements that are not covered by LOM. Due to this similarity, we do not intend to introduce the EdnaRecordBean class in detail.

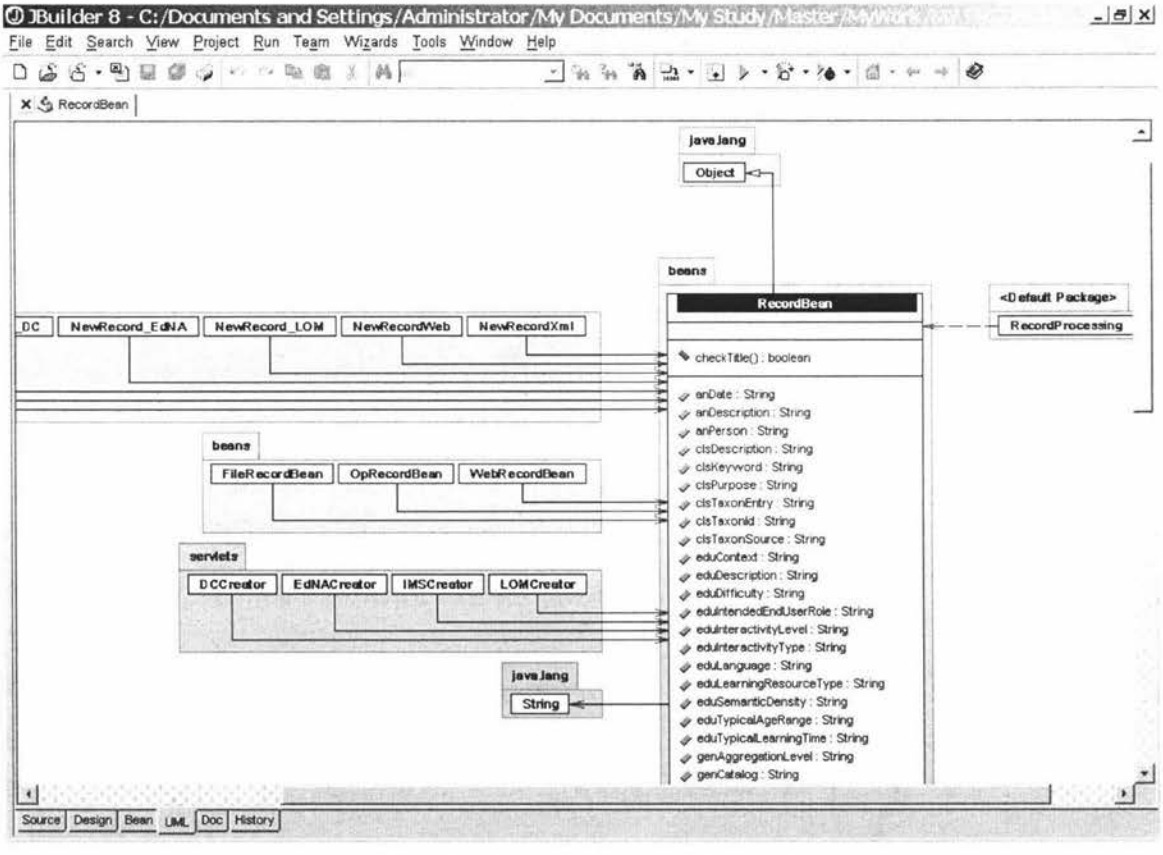


Figure 5.12 The UML view of the RecordBean class

5.3.2 The OpRecordBean and OpEdnaRecordBean Classes

It may be concluded from figure 5.10 that wherever the RecordBean class is used, the use of OpRecordBean class accompanies. In fact, it is the OpRecordBean class that defines various methods to operate on a RecordBean object. It is also the OpRecordBean class that actually accesses the database and manipulates a database record on behalf of various servlets and JSPs. Figure 5.13 shows the UML view of the OpRecordBean class. We can see from figure 5.13 the interactions with the classes defined in the Java library java.sql, which is the JDBC package that allows for the access to the database via the JDBC APIs. OpRecordBean defines a method to retrieve a data record from the database, and wrap the record in a RecordBean object in order to manipulate the record. Oppositely, OpRecordBean also defines a method to serialize a RecordBean object into the database. In addition, OpRecordBean defines a method to remove a data record from the database. These three public methods are named getSavedRecord, saveRecord, and deleteRecord, respectively.

Same as the similarity of EdnaRecordBean to RecordBean, the bean class OpEdnaRecordBean is also similar to OpRecordBean in structure, and different from the OpRecordBean class in complexity. It should be acknowledged, however, that the OpEdnaRecordBean class executes a private method that has no equivalence in the OpRecordBean class. This method is called createTable(), which dynamically create the database table Edna for the storage of the four elements specific to EdNA, if this table has not already existed in the database system.

5.3.3 The IMSCreator, EdNACreator, DCCreator, and LOMCreator Classes

The Java Servlets are believed important just because it is them that ultimately transform a database record to the desired XML document and present the XML document to the user. Currently there are four servlets – the IMSCreator, the EdNACreator, the DCCreator, and the LOMCreator in the application to generate IMS metadata, EdNA metadata, Dublin Core metadata, and LOM metadata respectively. As they are quite similar in structures, we just take the IMSCreator as an example. The UML view of the IMSCreator class is shown in figure 5.14.

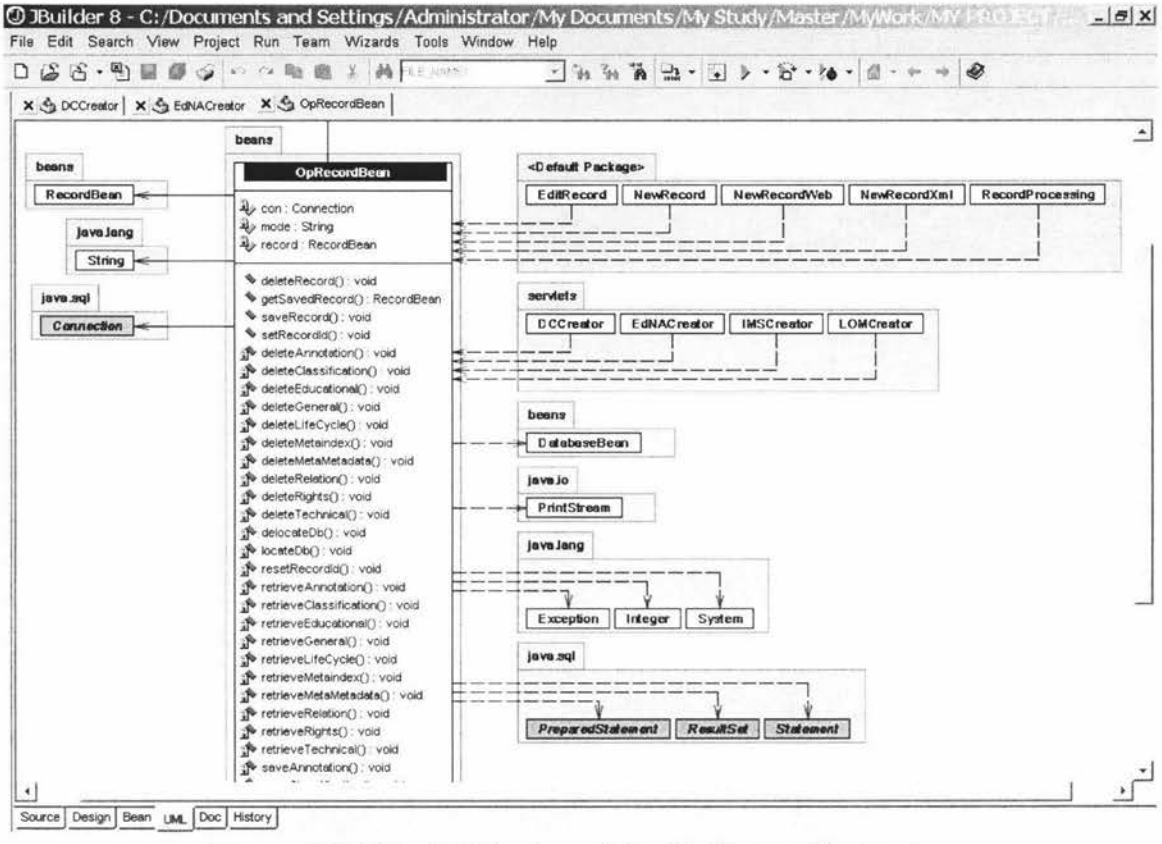


Figure 5.13 The UML view of the OpRecordBean **class**

Logically, the process of the generation of the IMS metadata in XML format can be divided into several steps. First of all, the IMSCreator makes use of the OpRecordBean object to access the database, retrieving the target data record and wrapping it into a RecoraBean object. Then the IMSCreator manipulates the RecordBean object to obtain the values for the data elements. After that, the IMSCreator constructs an XML DOM object in the memory from the element values by implementing the JAXP APIs. Of course this DOM object is constructed strictly according to the IMS XML Binding (2001) specification. Finally, the generated DOM tree is validated against the XML schemas provided by the IMS consortium, and if valid is transformed to an output stream to send to the client browser. The transformation from the DOM tree to the output stream is carried out by using the TransformerFactory class and the Transformer class in the javax.xml.transform package. Figure 5.14 clearly shows the use of the relative Java packages and classes for XML processing.

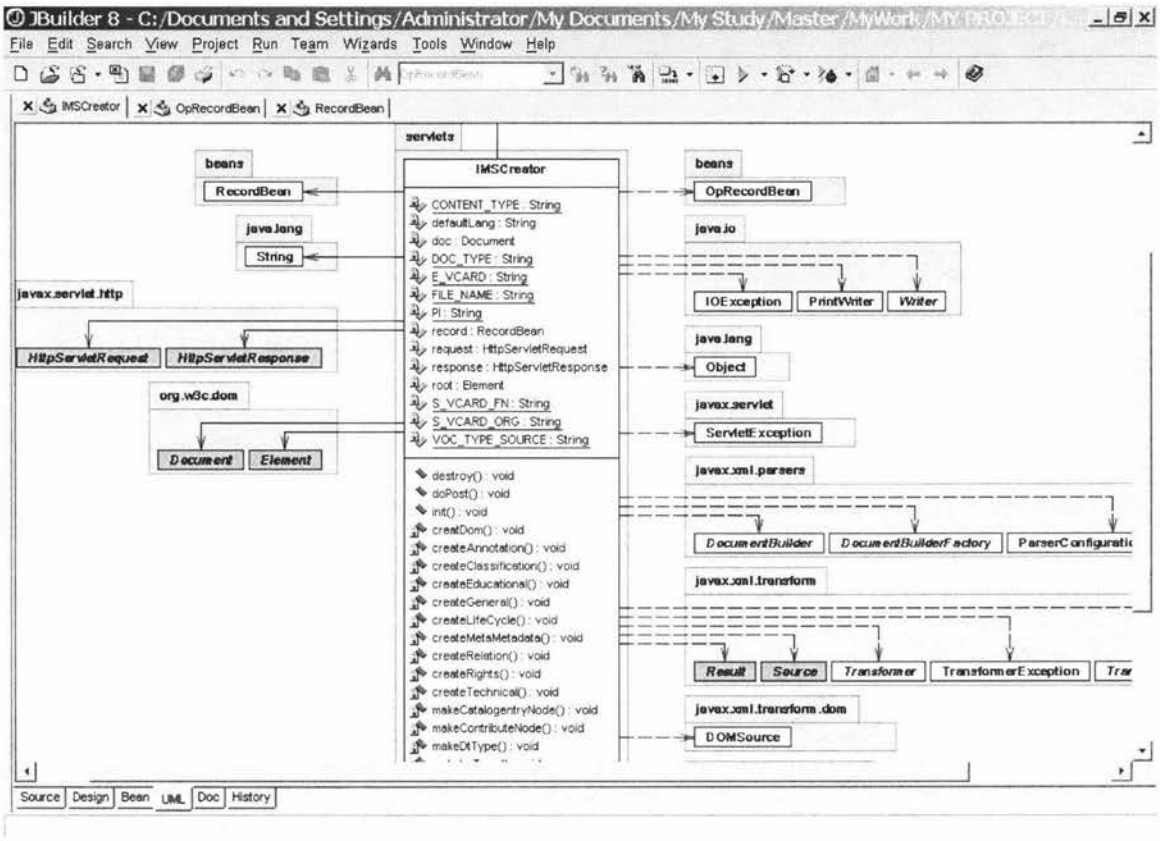


Figure 5.14 The UML view of the IMSCreator class

5.3.4 The FileUploadBean Class

FileUploadBean uploads an XML file from the client machine, and then save the uploaded file in a temporary folder on the server machine. This file uploading is realized by using a ServletInputStream object, which provides an input stream for reading binary data from a client request and includes an efficient readLine method for reading data one line at a time. With the HTTP POST protocol, the ServletInputStream object is used to read data sent from the client. The ServletInputStream object is retrieved via the HttpServletRequest.getInputStream() method. ServletInputStream is an abstract class defined in the javax.servlet package and is implemented by the servlet container. Figure 5.15 shows the UML view of the FileUploadBean class.

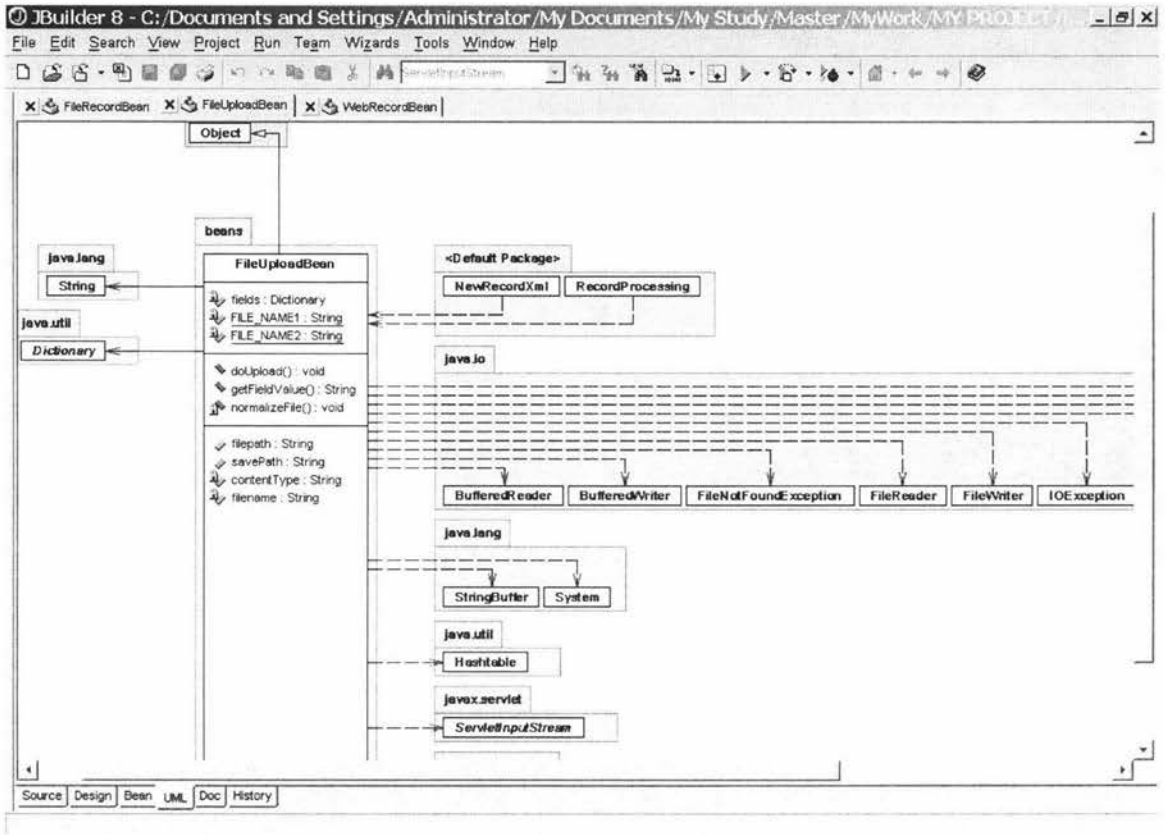


Figure 5.15 The UML view of the FileUploadBean class

5.3.5 The FileRecordBean Class

Exactly opposite to the servlet classes IMSCreator, EdNACreator, DCCreator, or LOMCreator that all change a RecordBean object to an XML file conformed to the corresponding metadata schema, the FileRecordBean class produces a RecordBean object for further processing from an XML file that is uploaded from the client by the FileUploadBean class. Similar to the servlet classes such as IMSCreator, JAXP APIs are implemented in the FileRecordBean class to process the uploaded XML file. A DOM object (or DOM tree) is built in the memory by parsing the uploaded XML file using the DocumentBuilder class offered by the DocumentBuilderFactory class. Then FileRecordBean traverses the tree structure to access all the nodes in the tree, so that the tag values in the uploaded XML file can be retrieved. Finally, FileRecordBean constructs and returns a RecordBean object, within which the values retrieved from the uploaded XML file were wrapped. Figure 5.16 shows the UML view of the FileRecordBean class.

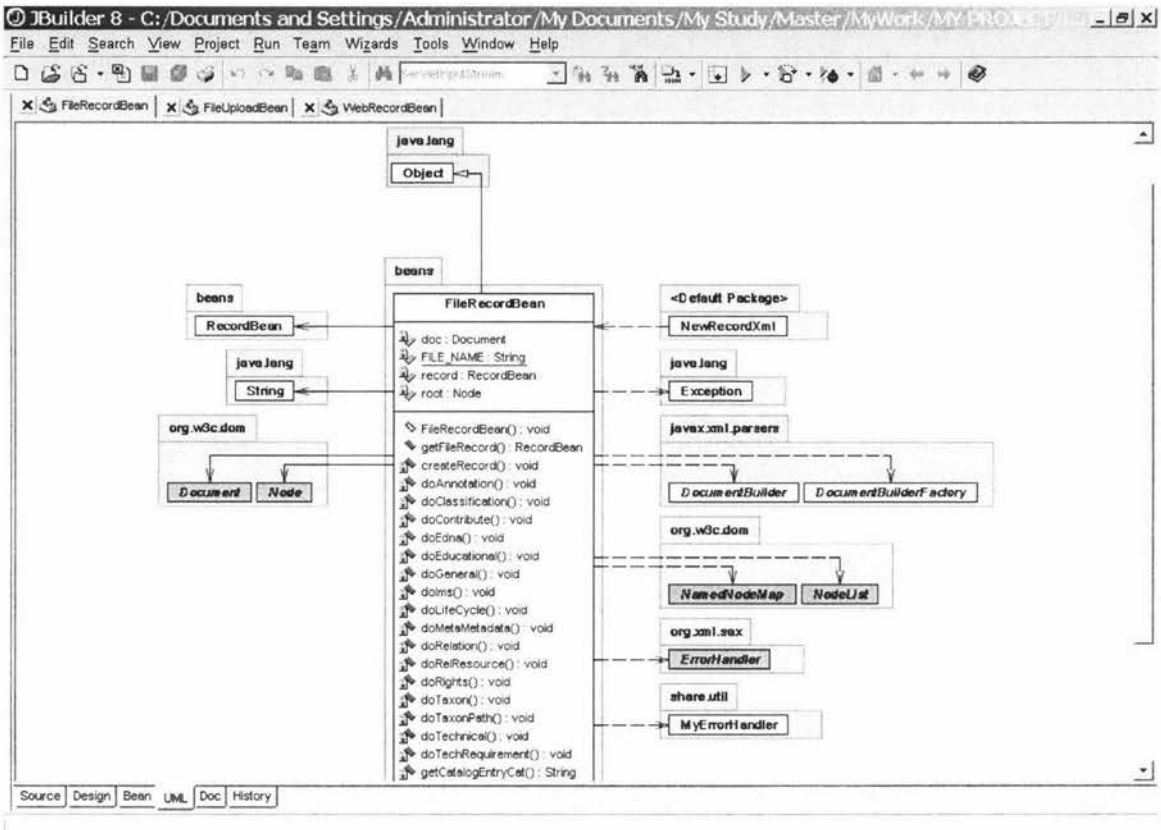


Figure 5.16 The UML view of the FileRecordBean class

5.3.6 The WebRecordBean Class

The WebRecordBean class tries to harvest as much metadata as possible automatically from a Web resource. In our application, a web resource means a HTML page with a title and/or several meta-tags. The purpose of the WebRecordBean class is to retrieve the title and the values for these meta-tags from the HTML page, and then create a RecordBean object using the retrieved values. Some of the targeted HTML elements and the corresponding database fields are listed in table 5.5.

We use the java.net package for implementing the networking applications. Two important classes defined in this package are URL and URLConnection. Class URL represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. The abstract class URLConnection is the super class of all classes that represent a communications link between the application and a URL. Instances of this class can be used both to read from and to write to the resource referenced by the URL. A URL object is created by passing the constructor the Web URL, which in our case is provided by the user. Invoking the URL object's openConnetion() method returns a URLConnection

object that represents a connection to the remote HTML page referred to by the URL. The WebRecordBean then reads this HTML page into a string buffer using an InputStream object, which is obtained via the getInputStream() method of the URLConnection object. The WebRecordBean finally parses the string to extract the desired metadata, and wraps the metadata into a RecordBean object. Figure 5.17 shows the UML view of the WebRecordBean class.

Table 5.5 Extracted HTML data and the corresponding database fields

HTML element	Database field	
	Table name	Field name
Title	MetaIndex	Title
Page URL	General	Identifier
	General	Entry
	Technical	Location
String "URL"	General	Catalog
Content length	Technical	Size
<meta name="author" ...>	LifeCycle	AuthorEntity
<meta name="publisher" ...>	LifeCycle	PublisherEntity
<meta name="keyword" ...>	General	Keyword
<meta name="description" ...>	General	Description

5.4 The Web Application Deployment

5.4.1 The Tomcat Server

We use Tomcat as the servlet container to implement the Java Servlets and JSPs. Tomcat is a free, open-source functional implementation of Java Servlet and JavaServer Pages technologies. Tomcat is developed under the Jakarta project at the Apache Software Foundation, and is officially integrated into the J2EE reference implementation from Sun Microsystems. The most current version at the time of writing of this thesis is Tomcat 5.5, which supports the Servlet 2.4 and JSP 2.0 specifications.

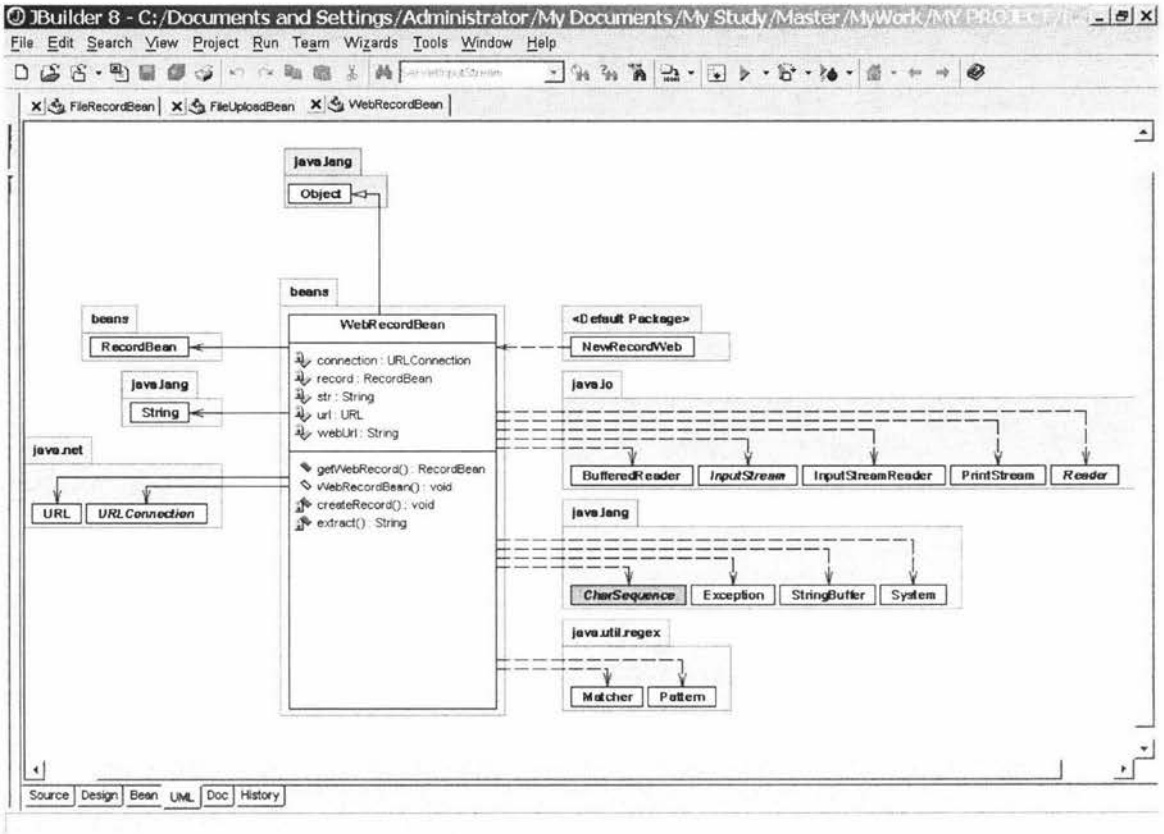


Figure 5.17 The UML view of the WebRecordBean class

Tomcat includes a Web server, so it can be used as a standalone server for JSPs and servlets. It can also be configured as the handler for JSP and servlet requests received by typical Web servers such as the Apache Web server or the Microsoft's Internet Information Server (IIS). In a typical Web server environment, all requests for Web pages and servlets are first received by the typical Web server, which then decides, usually based on the file name or directory of the request, whether to serve up a static page by itself or to pass control over to a servlet container – the Tomcat server in our case. The Tomcat server responds to an HTTP request for a servlet, creates/invokes that servlet as necessary, and handles any error conditions. The information flow between the client browser, the typical Web server, and the Tomcat server is shown in figure 5.18.

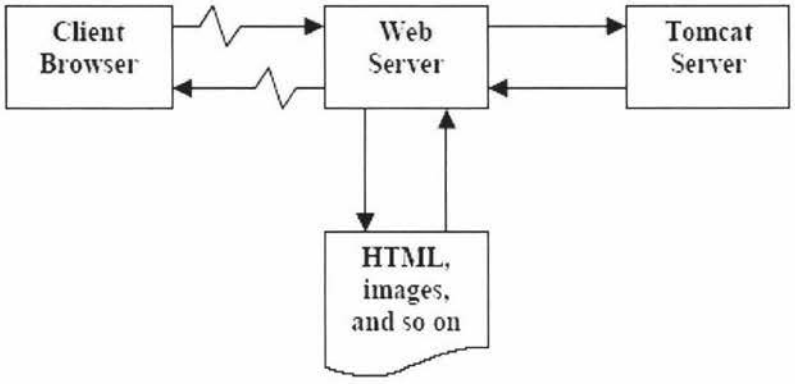


Figure 5.18 Information flow between the client, the Web server, and the Tomcat server

5.4.2 Directory Structure of the Application

The most important folder in Tomcat is webapps, which is directly under the root of the Tomcat installation directory (also called "$CATALINA_HOME"). The webapps folder holds all Web applications. Our application, named metawork, is therefore on the following directory path.

$CATALINA_HOME/webapps/metawork/

A Web application is a set of folders and files with a certain structure. The folder structure of our application metawork is illustrated in figure 5.19.

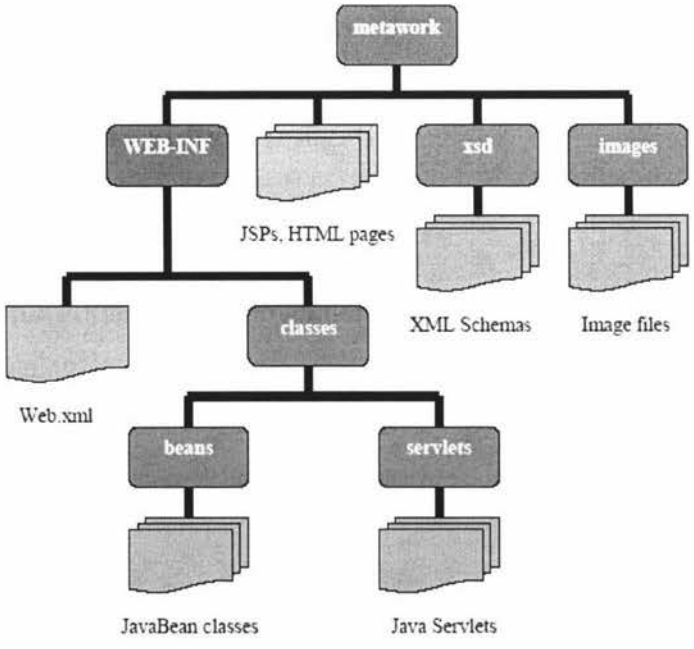


Figure 5.19 Structure of the Web application for our project

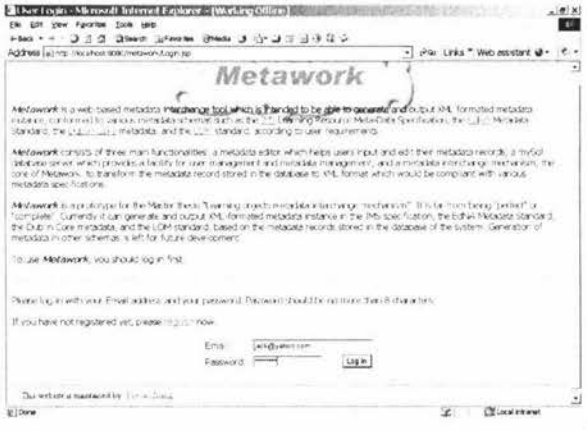
5.5 System Evaluation

Finally, we tested the system with various input data. In this section, we show some of the testing input to the system and the corresponding output from the system, in order to demonstrate the test results.

5.5.1 Screen Interfaces

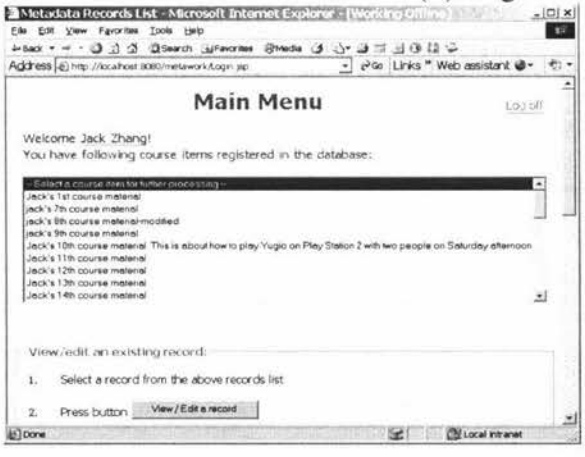
5.5.1.1 User Login and the Main Menu

When the system is started, it prompts the user to login to the system, or alternatively, to register to the system if the user is new. After the user successfully logs in or registers to the system, the system presents the user the main menu, where the user can make various choices such as inputting a new record, editing an existing record, and converting a record to the desired schema. Figure 5.20 shows the screen shots of the login form, the registration form, and the main menu.



(i) Login form

(ii) Registration form



(iii) Main menu

Figure 5.20 User login, registration and the main menu

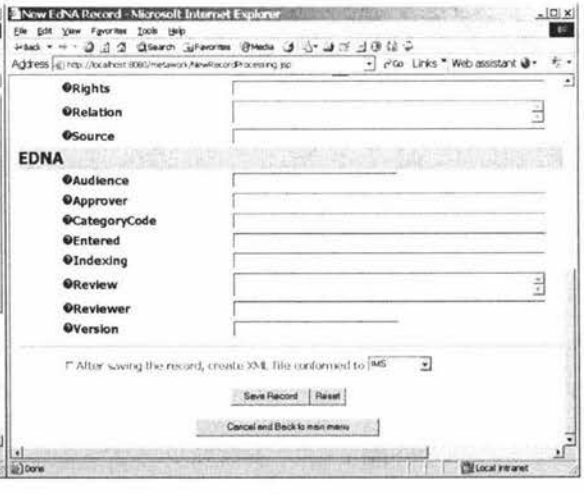
5.5.1.2 Data Input by Keyboard

On the main menu, firstly the user may choose to input a new record from scratch by using keyboard. On the input form, after all the data values have been typed up, the user may choose just to save the record, or convert it to the desired metadata schema in XML while saving the record. There are three types of input forms: the DC-input form for Dublin Core input, the EdNA-input form for EdNA metadata input, and the LOM-based form for all other schemas under the current situation. The user can explicitly select from the main menu what form to use according to the data requirement. The three types of input form are shown in figure 5.21. The only difference between the EdNA-input form and the DC-input form is that, the former has eight input fields more than the latter does.

(i) Select input type from main menu

(ii) LOM-based form

(iii) DC-input form

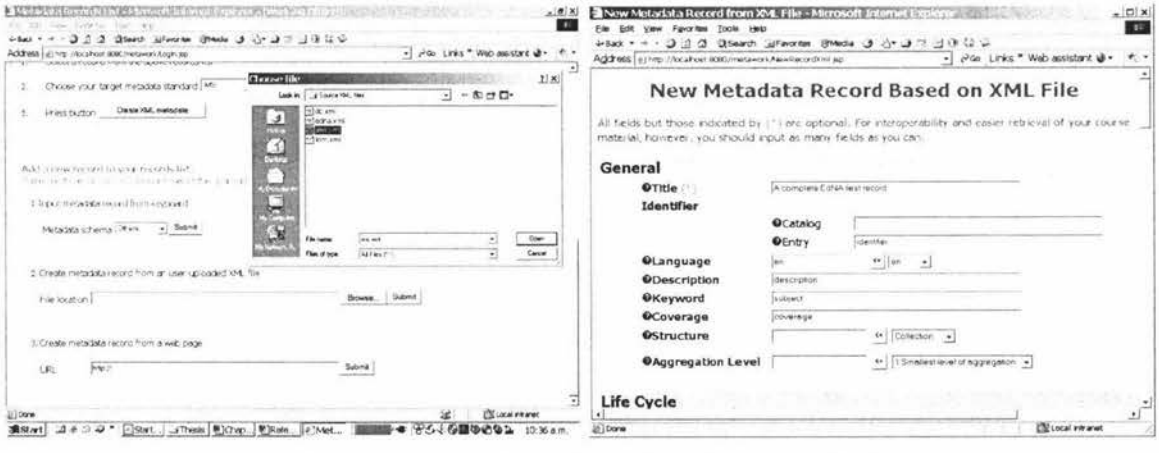


(iv) EdNA-input form

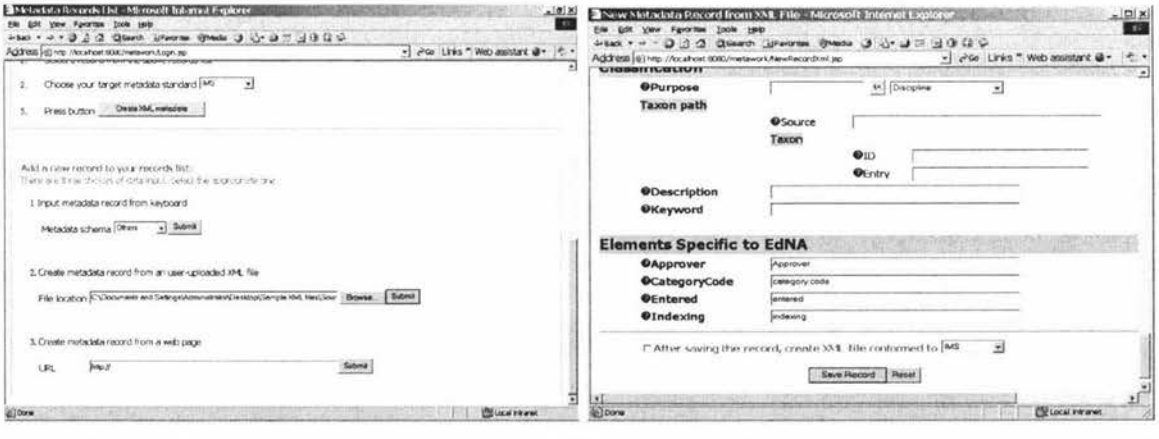
Figure 5.21 Record input form for various metadata schemas

5.5.1.3 Input by Uploading XML

When uploading an XML record, the user may input the location of the source XML file either by directly typing the location or by using the "Browse..." button from the main menu. After the source XML file is submitted to the system, the system parses it to extract the data values. The extracted data is presented to the user on an edit form for confirmation and modifications, before the data is saved to the database or converted to another XML file in a different schema. The edit form is basically LOM-based; if the source XML file is in a schema that holds elements unavailable in LOM, however, the edit form provides extra text fields to accommodate these extra elements. In figure 5.22, (i) shows the main menu and the edit form when uploading an IMS-compliant XML file (located at C:\Documents and Settings\Administrator\Desktop\Sample XML files\Source XML files\ims.xml on the client machine); (ii) shows the uploading of an EdNA-compliant XML file (located at C:\Documents and Settings\Administrator\Desktop\Sample XML files\Source XML files\edna.xml on the client machine), which has extra elements.



(i) uploading XML file in IMS: LOM-based edit form

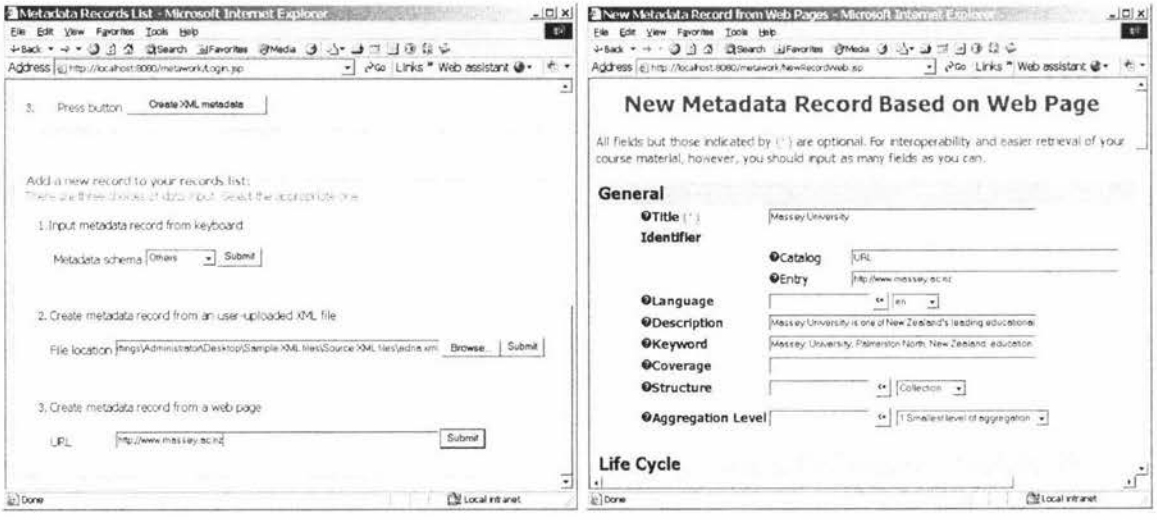


(ii) uploading XML file in EdNA: extra text fields on edit form

Figure 5.22 The Main menu and the edit form when uploading XML

5.5.1.4 Input by Harvesting Web Pages

In this case, the Web URL is provided by the user on the Main menu. After data has been harvested from the web page, the system presents the data on an edit form to the user, so that the user can edit the data as desired before the data is further processed. The edit form in this case is always LOM-based, and the record is saved as LOM-compliant. Figure 5.23 shows the URL input on the main menu and the harvested data on the edit form, when a Web resource (the Massey University homepage in this example) is provided to the system for automatic creation of the metadata for this Web resource.

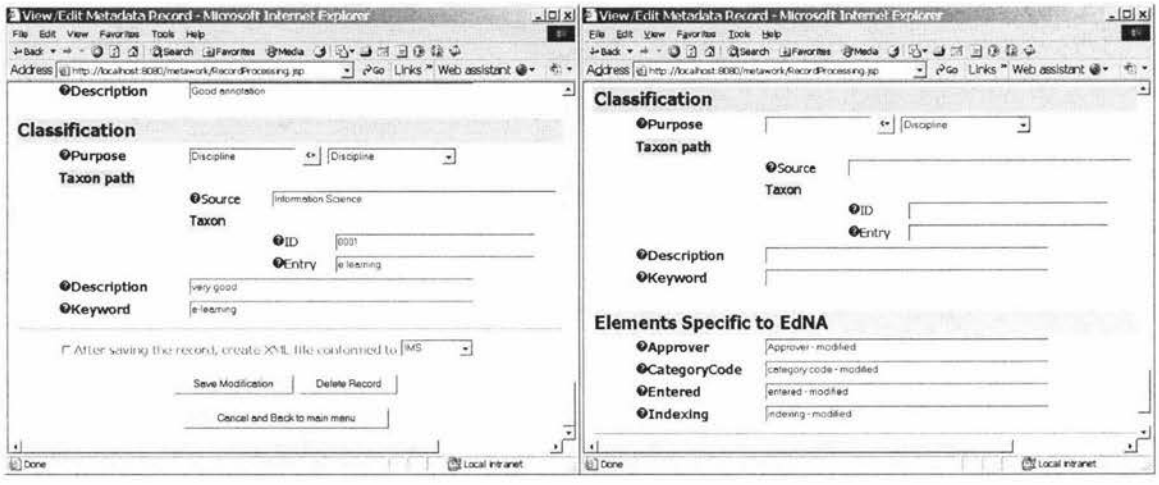


(a) input the URL (b) harvested data displayed on the edit form

Figure 5.23 Create metadata from Web source

5.5.1.5 Edit Existing Records

Unlike the keyboard input of a new record, which is done with different input forms for different metadata schemas, editing an existing record is carried out basically on the same LOM-based edit form. This is because the database itself is LOM-based, and all records, no matter what schema they initially were, once saved to the database, are converted to the intermediate schema – LOM. Only for those records that contain elements that are not covered by the LOM model, does the edit form provide extra text fields for editing these extra data elements. In our system, the records originally in the EdNA schema contain the four extra elements unavailable in LOM, so for these records only, the edit form is different in that it has four text fields more than it ordinarily does. Figure 5.24 shows the difference.



(a) edit form for a LOM record (b) edit form for an EdNA record

Figure 5.24 Edit an existing record

5.5.2 Examples of Interchange

The system can generate an XML record in one of the four metadata-schemas (IMS, EdNA, Dublin Core, and LOM) either from the main menu by selecting an existing record, or directly from any of the input forms or edit forms by translating the current record manipulated on theses forms. We tested the system with various data and in various ways; the results show that the system functions correctly. It is impossible, however, to present all the test results here. The following two examples demonstrate the most two typical interchange processes and the results.

5.5.2.1 Transform a Database Record to XML Record in Various Schemas

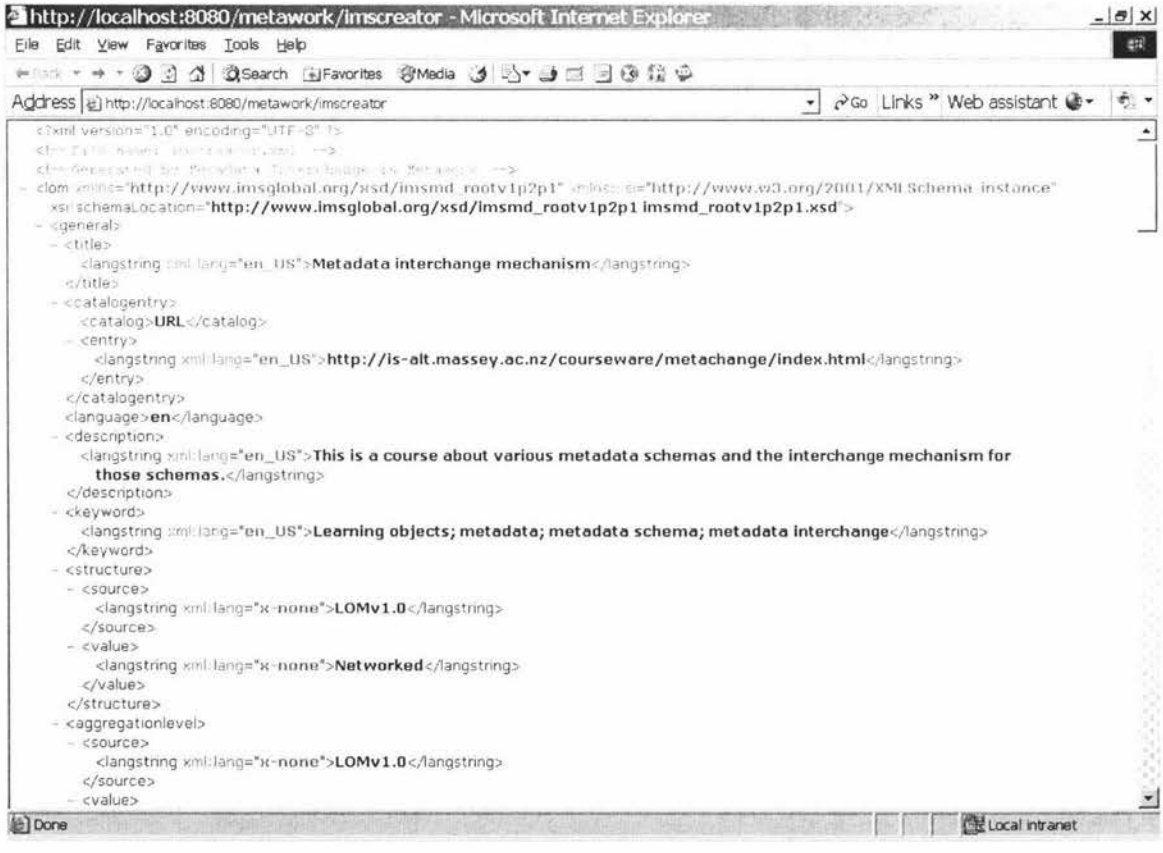
The example data we used for this test is shown in table 5.6. We used the LOM-based input form (figure 5.21 (ii)) to input and save the record into the system. Then we select the record from the main menu, and transform it to XML format in IMS, EdNA, Dublin Core, and LOM orderly. The XML results are shown in figure 5.25, rendered by Microsoft Internet Explorer. Comparing the content of these four XML documents and the source data (table 5.6), it is clear that the system functions just as what we hoped.

Table 5.6 An example metadata record

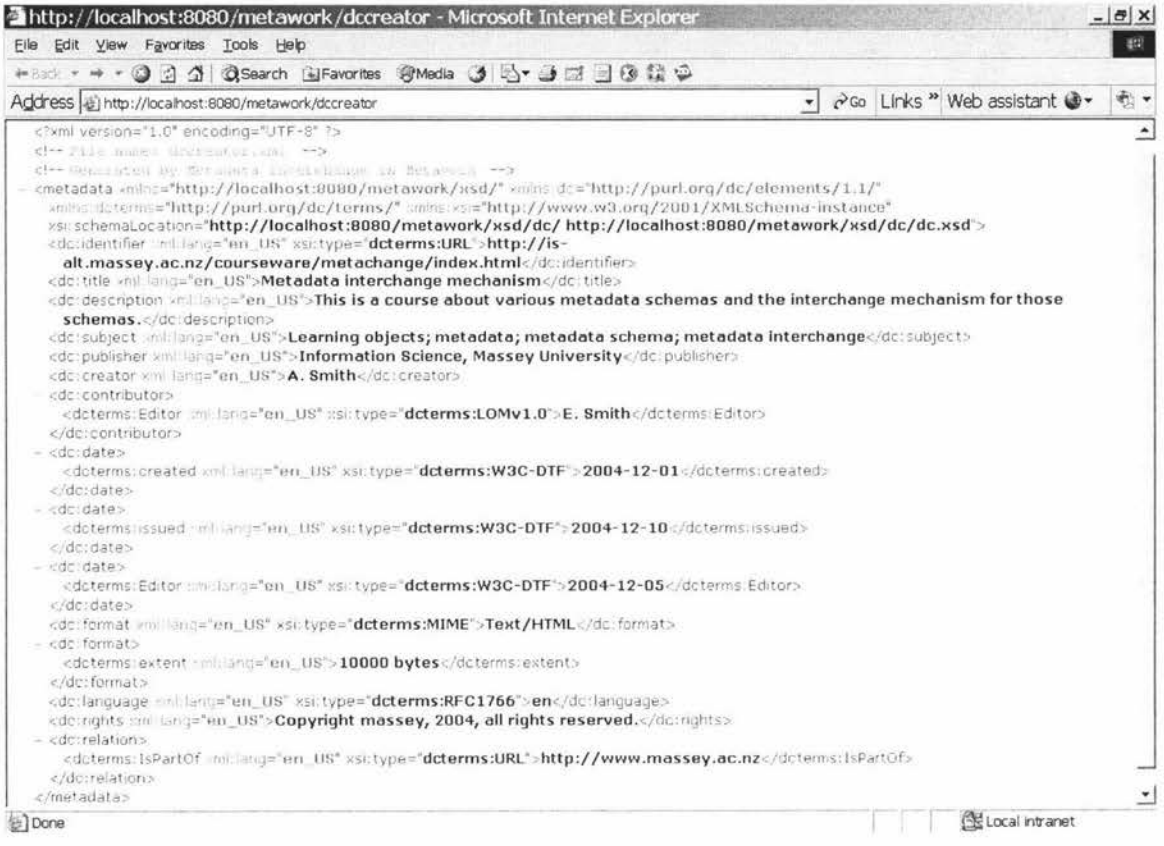
Attribute Name	Attribute Value
Title	Metadata interchange mechanism
Identifier	http://is-alt.massey.ac.nz/courseware/metachange/index.html
Language	English
Description	This is a course about various metadata schemas and the interchange mechanism for those schemas.
Keyword	Learning objects; metadata; metadata schema; metadata interchange
Structure	Networked
Aggregation level	Collection of atoms
Version	2.0
Status	Final
Author	A. Smith
Date created	2004-12-01
Publisher	Information Science, Massey University
Date published	2004-12-10
Editor	E. Smith
Date edited	2004-12-05
Metadata identifier	http://is-alt.massey.ac.nz/courseware/metachange/meta.html
Metadata creator	M. C. Smith
Date of metadata creation	2004-12-20
Metadata validator	M. V. Smith
Date of metadata validation	2004-12-25
Metadata scheme	LOM1.0
Metadata language	American English
Format	Text/HTML
Size	10000 bytes
Operating system	Win2000
Browser	Any
Intended end user role	Learner
Typical age range	18-30
Typical learning time	1 hour
Copyright and other restrictions	Yes
Copyright description	Copyright Massey, 2004, all rights reserved.
Relation kind	Is part of
Relation source identifier	http://www.massey.ac.nz
Relation source description	Massey University home page
Annotation content	A good course material for learning metadata standards and their interchange mechanism.
Person of annotation	A. N. Smith
Date of annotation	2004-12-31


http://localhost:8080/metawork/lomcreator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/metawork/lomcreator
Done
Local intranet

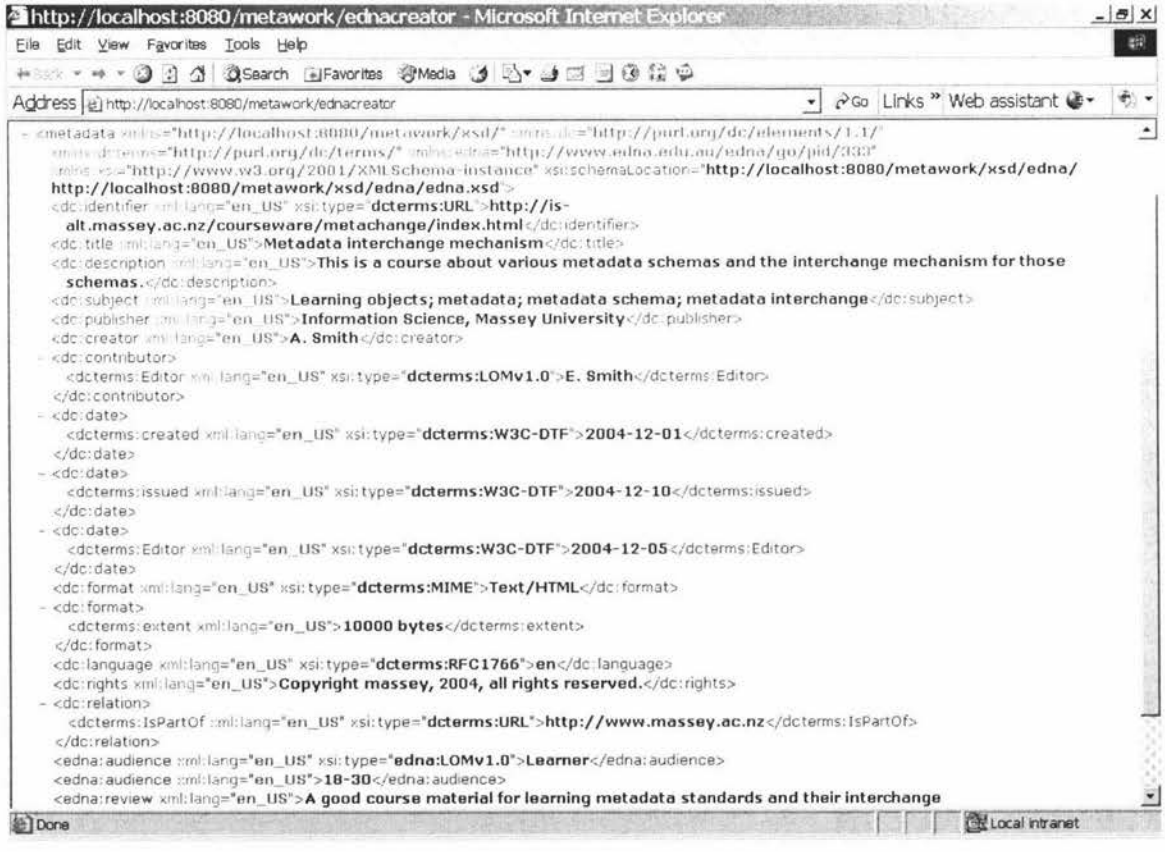
(i) LOM


http://localhost:8080/metawork/imscreator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/metawork/imscreator
Done
Local intranet

(ii) IMS



(iii) Dublin Core

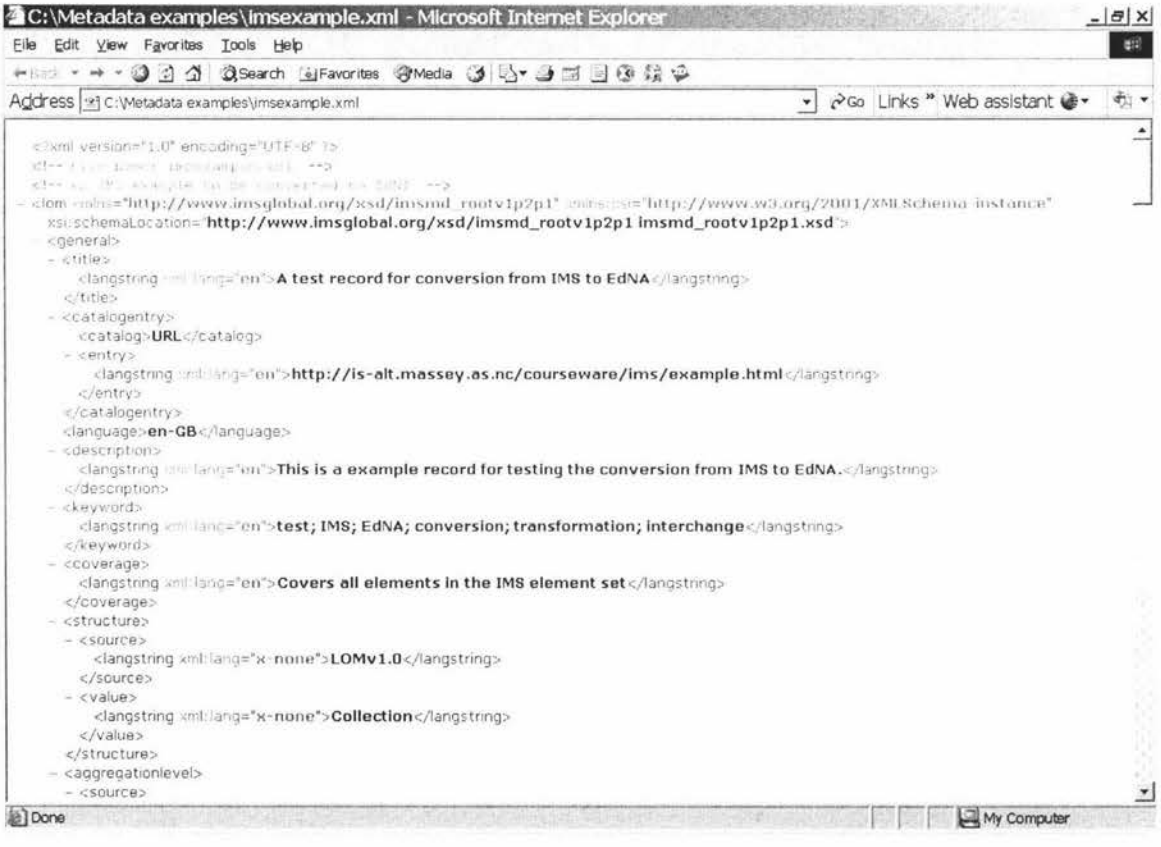


(iv) EdNA

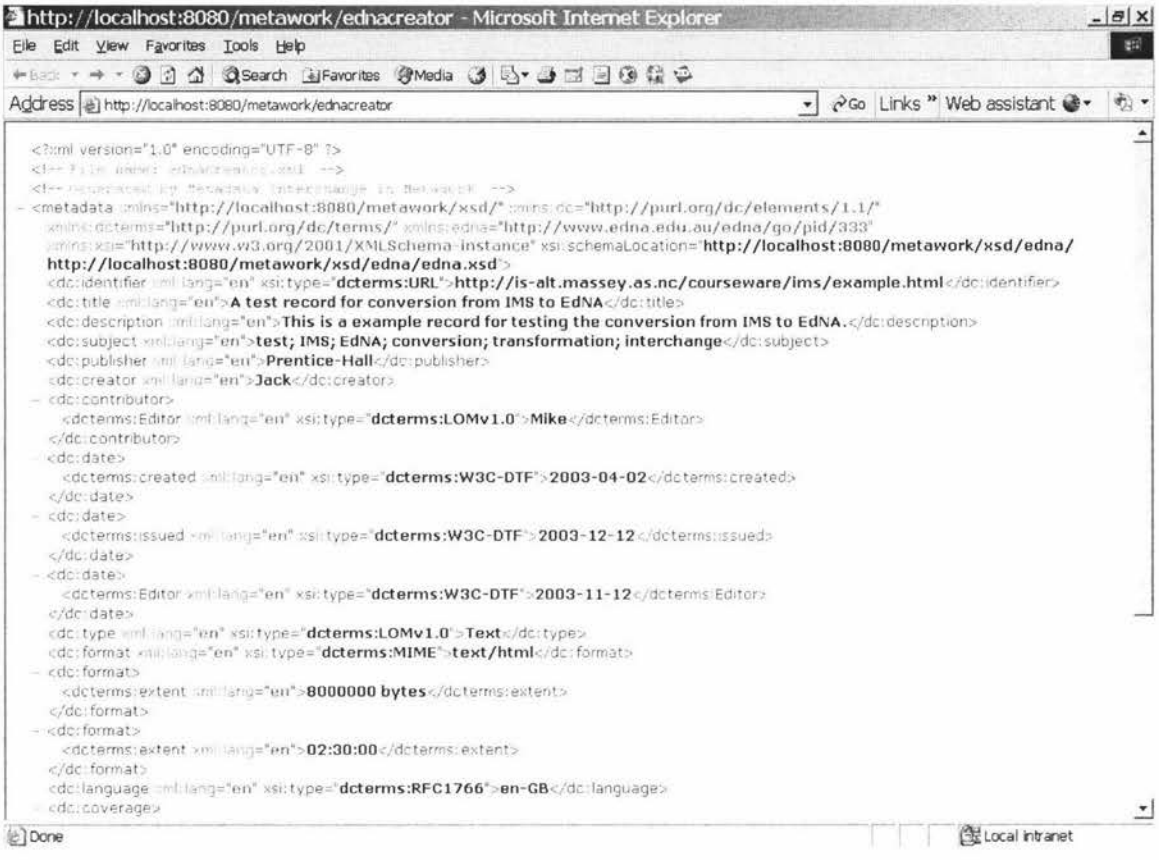
Figure 5.25 XML format of the metadata record in various schemas

5.5.2.2 Transform an XML Record From One Schema to Another Schema

In this case, an XML record in a certain schema (IMS, EdNA, Dublin Core, or LOM) is uploaded from the client machine by the user to the system. The system will be able to generate and output this metadata record also in XML format but in another schema targeted by the user. The following example shows the transformation from the IMS Metadata Specification to the EdNA Metadata Standard. The IMS-compliant XML file (the source file) to be uploaded to the system is named imsexample.xml and located in the directory c:\Metadata examples on the client machine. The uploading is carried out from the main menu, as shown in figure 5.22. Figure 5.26 shows both the source XML file and the targeted EdNA-complint XML file, rendered by the Microsoft Internet Explorer. Comparing theses two XML files, we may conclude that the metadata interchange mechanism in the system works correctly, as expected.



(a) source XML file (IMS) located on client machine



(b) target XML file (EdNA) output by the prototype system

Figure 5.26 Transform a record from one schema to another schema

5.6 Summary

This chapter discussed the implementation aspects of the prototype system. Firstly, the main technologies involved in the system were introduced, and how these technologies were implemented in the system was described. Around the core technologies – Java and XML, we discussed the JDBC APIs as the access interface to the MySQL database, the JAXP APIs as the means to parse, validate and transform XML documents via Java, Servlets and JSPs as the server side programming language to generate dynamic responses to the client, and JavaBeans as an enhancement for JSPs and servlets to encapsulate more complex control logic. Secondly, the main Java components, including JSPs, servlets, and JavaBeans, were summarized with a description of the role they play in the system. We discussed how theses components collaborate with each other to undertake various tasks required by the system design. From the collaboration diagram for all the components and the class diagram for the JavaBeans, we could see how Java components is used and reused by other components; how a general-purpose metadata interchange system becomes possible due to the use of the LOM-intermediated approach.

We also pointed out from the implementation point of view that the system is highly scalable and extensible, so that a new metadata schema can be accommodated into it without the effort to reconstruct the system, similarly, due to the LOM-intermediated approach. Thirdly, the most important Java components (servlets and JavaBeans) were described in somewhat detail. We presented the UML view of these components, although it is impossible to copy the source code in this chapter. We discussed how the components undertake the tasks, how they use other Java resources, how they are used by other components, and what properties and methods they have. After that, the Tomcat server for implementing the servlets and JSPs were briefly introduced, and the deployment structure of this Web application in the Tomcat server was illustrated. Finally, we evaluated the developed system with various input data and in various ways. In this chapter, we presented the input/output screen shots under different situations, to demonstrate that the system functions as expected. Most importantly, we presented the XML metadata in different schemas, generated by the system with different types of source data input. The result shows that the encoding of the XML files is strictly conformed to the XML binding scheme specified for that metadata schema, and the semantic structure of the XML files is legal, when validated against the XML Schemas defined for that metadata schema.

In next chapter, we will come into the conclusion about the project and point out some future works for improving the developed system.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This thesis presented a framework for metadata interchange mechanism, along with the description of the design and implementation of the prototype system. The positive evaluation results show that the system is fully functional and highly scalable. The LOM-intermediated approach proposed by the thesis was successful in developing such a general-purpose, extensible system for conversions among multiple metadata schemas, and the dynamic-database methodology adopted in the thesis was effective for preventing any extra data-loss resulting from the use of the LOM-intermediated approach. Basically, the LOM-intermediated approach allows a schema-schema mapping to be undertaken in a schema-LOM-schema way, with LOM acting as an intermediary, while the dynamic-database methodology allows necessary database tables to be dynamically created for some schemas to retain the data that are identified in these schemas but are not covered by the LOM data model. In fact, the LOM-intermediated approach, together with the dynamic-database methodology, are the keys in this project to the development of the learning object metadata interchange system that requires high scalability and high accuracy.

The wide accommodation that enables the manipulation of large number of metadata schemas and the high scalability that facilitates the easy extension to incorporate a new schema into the system are the two main advantages over the current metadata mapping tools, which mostly can only convert one schema to a limited number (often 3 – 4) of other schemas, and can hardly be extended. Taking into consideration the continually increasing number of metadata schemas and the amounting need for the interoperability across heterogeneous systems that use different schemas, we believe that the wide accommodation and the high scalability are essential for such a metadata interchange system. Therefore, we believe that the proposed metadata interchange framework, together with the LOM-intermediated approach and the dynamic-database methodology that were implemented to reach the goals of wide accommodation and high scalability, are the main contributions made by this thesis to the metadata research area. Further more, the development of the prototype shows that the framework itself, the LOM-

intermediated approach and the dynamic-database methodology are also practical and quite simple to implement.

6.2 Future Work

Some future work can be taken into consideration to further develop the metadata interchange framework proposed in this thesis. Especially, the implementation of the prototype, as a demonstration of the functionalities of the framework, is far from being perfect. The scope of future development is discussed below.

Firstly, more comprehensive considerations might be taken on the interactions with the client systems. Currently, the system is used as a mere piece of software independent of any client systems. It simply receives input and instructions from a human user, and then responses the user with an output, an XML document conformed to the target metadata schema. In this process, the user needs to explicitly tell the system what the target schema is, while the system does not take any care about who the user is, and how the user will use the XML document generated by the system. However, we may imagine that the client of the developed system is either a learning object repository that use a certain schema or a personal individual who belongs to a specific community in which a certain schema is applied. In both cases, the system may identify the client that made the request, and automatically response with the desired metadata schema. To do so, the system may keep a profile for each client, and for different clients, the system outputs the metadata in different schemas. Such client adaptability is illustrated in figure 6.1.

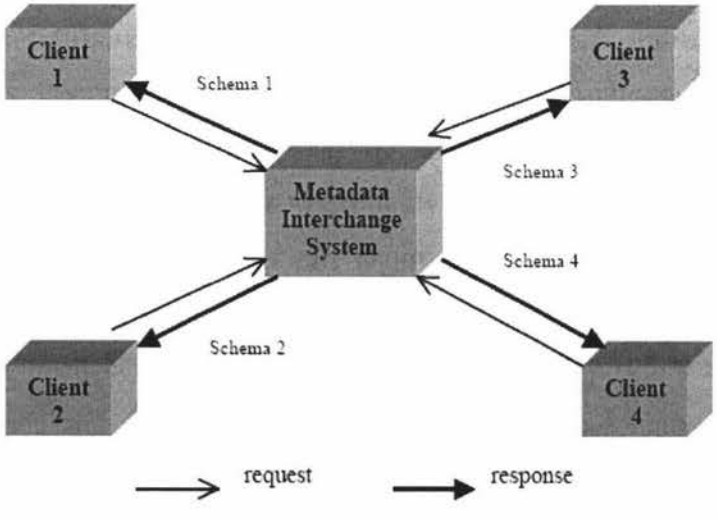


Figure 6.1 Response with different schemas automatically for different clients

Secondly, collaborations with related standardization bodies or standard initiatives might be needed. The collaboration includes two aspects: the mapping of the metadata schema to LOM, and the XML binding scheme of the schema. For those schemas such as Dublin Core from which mapping to the LOM standard has already been made clear and widely accepted, the collaboration on mapping issues with the standard organization (in this case the Dublin Core Metadata Initiative) is no longer necessary; similarly, for those schemas such as IMS whose XML binding has already been well identified as a part of the specification itself, the collaboration with the standard organization (in this case the IMS consortium) on how to encode the schema into XML format can be omitted. However, few schemas have specified both the mapping to LOM and the XML binding scheme, therefore in most cases it might be better to collaborate with the standard organizations to identify a publicly accepted mapping to LOM and to establish a widely used guideline for XML encoding.

Take the EdNA Metadata Standard that is being manipulated in our prototype as an example. For EdNA, there is neither a widely accepted LOM-mapping nor a commonly known XML encoding scheme identified by the EdNA Online or any other parties, so currently either the mapping between EdNA and LOM or the XML encoding of EdNA in the project are largely dependent of the author's understanding of the EdNA data set and even the author's personal preference. While on the other hand, all users and client systems of the developed metadata interchange system are required to conform with the LOM-mapping as well as the XML encoding both identified by the author, in order for the seamless communication across systems and a common understanding of the data. In this case, it is obvious that collaboration with the EdNA Online about the LOM-mapping and the XML binding is needed for higher accuracy, wider acceptability and better interoperability.

The reasons for the collaboration with the standard organization, rather than with any other parties, are simple. First, the standard organization has the best expertise about the metadata schema and the best understanding of the specific needs of the community, so the highest accuracy is guaranteed. Second, any documentation, directly or indirectly issued as a part of the schema from the standard organization, has the widest acceptability, so that the seamless communications between the developed system and the client system are possible. The collaboration may be in several ways. One possible way is that we

document the LOM-mapping and the XML encoding that are used in the developed system, and then submit the documentation to the standard organization. The standard organization may make the documentation available to public as drafts or recommendations, or eventually as a part of the schema itself (probably after several turns of discussions and revisions).

Thirdly, the data input form (metadata editor) of the prototype system also leaves an avenue for improvement. Although the current metadata editor sufficiently enables the input and edit of the data that are used to demonstrate the functionalities of the prototype, a more mature, fully functional metadata editor that is able to deal with repeatable data elements in a flexible way might be desired. With such a flexible editor, any repeatable element, no matter whether it is a container element or a leaf element, can be repeatedly assigned with as many values as desired by the user. This functionality can be realized by cloning and inserting the corresponding text field for that element. Figure 6.2 illustrates the demo of such a flexible metadata editor, which was revised from the Reg-Metadata editor (DSTC Pty Ltd, 1999), designed for the creation of qualified Dublin Core metadata.

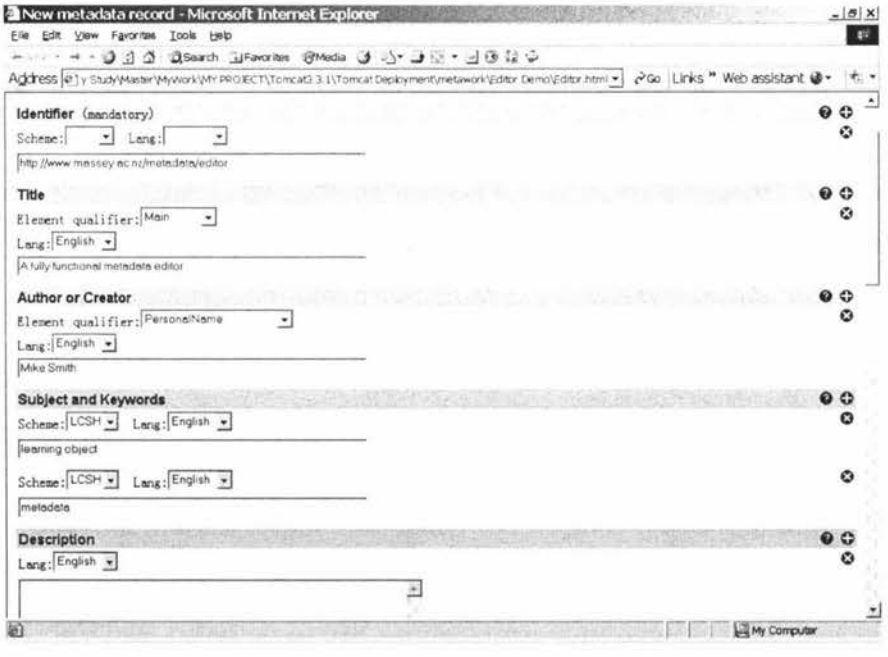


Figure 6.2 Demo of a flexible DC editor (DSTC Pty Ltd, 1999)

The user can repeatedly input values for an element by clicking the "+" icon on the right of the element name bar, so that a clone of the text field for that element may be inserted right afterward. For example, the text field for the "Subject and Keywords" element has

been cloned and inserted once in figure 6.2, and now this element has two values: "learning object" and "metadata". On the contrary, the user can also delete a repeated text field by clicking the "x" icon at the up-right corner of that text field.

References

ADL. (2004). SCORM ® 2004 overview. Retrieved March 9, 2004, from http://www.adlnet.org/

Armstrong, E., Ball, J., Bodoff, S., Carson, D. B., Evans, I., Green, D., et al. (2004). *The J2EE 1.4 tutorial*. Santa Clara, California: Sun Microsystems.

Boyle, T. (2003). A report to the CETIS educational content SIG. *Learning Objects 2003 Symposium: Lessons Learned, Questions Asked, June 24, 2003, Honolulu, Hawaii*. Retrieved July 9, 2004, from http://www.cetis.ac.uk/groups/20010809144711/FR20030728173304

Brody, R. (2003). Information ethics in the design and use of metadata. *IEEE Technology and Society Magazine, 22(2), Summer 2003,* pp. 34-39.

CanCore Guideleines Version 2.0. (2002). CanCore guidelines version 2.0: introduction. Retrieved May 10, 2004, from http://www.cancore.ca/guidelines/CanCore_Guidelines_Introduction_2.0.pdf

Chen, Y. A., & Chen, S. S. (2002). Metadata for e-learning objects. *Presentation at PN 2002 Annual Conference and Joint Meeting, Osaka, Japan, September 21, 2002*. Retrieved March 5, 2004 from http://pnclink.org/annual/annual2002/pdf/0921/11/e211105.pdf

Conlan, O., Hockemeyer, C., Lefrere, P. Wade, V., & Albert, D. (2001). Extending educational metadata schemas to describe adaptive learning resources. *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia* (pp. 161-162), New York: ACM Press.

DC XML Guidelines. (2003). Guidelines for implementing Dublin Core in XML. Retrieved May 10,2004, from http://dublincore.org/documents/2003/04/02/dc-xml-guidelines/

DC XML Recommendations. (2002). Recommendations for XML Schema for qualified Dublin Core. Retrieved May 10, 2004, from http://www.ukoln.ac.uk/metadata/dcmi/xmlschema

DC XML Schema Notes. (2003). Notes on the W3C XML Schemas for qualified Dublin Core. Retrieved May 10, 2004, from http://dublincore.org/schemas/xmls/qdc/2003/04/02/notes/

Downes, S. (2002). Smart learning objects. *Stephen Downes – Online Articles*. Retrieved October 8, 2004, from http://education.qld.gov.au/staff/learning/courses/sdownesapril.html

DSTC Pty Ltd. (1999). Reg-Metadata editor. Retrieved May 12, 2004, from http://www.metadata.net/reg/

Duval, E. (2004). Learning technology standardization: making sense of it all. *Computer*

Science and Information Systems, 1(1).

Duval, E., & Hodgins, W. (2003). A LOM research agenda. *Proceedings of the 12th International World Wide Web Conference.* Retrieved July 7, 2004, from http://www2003.org/cdrom/papers/alternate/P659/p659-duval.html.html

Duval, E., Hodgins, W., Sutton, S., & Weibel, S.L. (2002). Metadata Principles and Practicalities, *D-Lib Magazine, April 2002, 8(4).*

eduSource Canada. (2003). Learning objects & learning object repositories. Retrieved October 15, 2004, from http://www.col.org/onlineseminar/learning/1_Edusource.pdf

Farance, F. (2003). IEEE LOM standard not yet ready for "prime time". In Kinshuk (Ed.), *IEEE Computer Society Learning Technology Task Force (LTTF), 5(1), January 2003.*

Fischer, S. (2001). Course and exercise sequencing using metadata in adaptive hypermedia learning systems. *ACM Journal of Educational Resources in Computing, 1(1), Spring 2001.*

Friesen, N. (2002a). Building educational metadata application profiles. *Pro. Int. conf. on Dublin Core and Metadata for e-Communities 2002*, pp. 63-69.

Friesen, N. (2002b). *E-learning standardization: an overview.* Retrieved May 10, 2004, from http://www.cancore.ca/documents.html

Friesen, N. (2002c). International e-learning specifications. *The International Review of Research in Open and Distance Learning: Online Version, 3(2).* Retrieved October 15, 2004, from http://www.irrodl.org/content/v3.2/tech11.html

Friesen, N, (2003). Three objections to learning objects. In McGreal, R. (ed.), *Online Education Using Learning Objects* (pp. 59-70). London: RoutledgeFalmer.

Friesen, N, (2004). Semantic & syntactic interoperability for learning object metadata. In Hillman, D. (ed.), *Metadata in Practice.* Chicago: ALA Editions.

Friesen, N., & McGreal, R. (2002). International e-learning specifications. *International Review of Research in Open and Distance Learning, October 2002*, Retrieved October 18, 2004, from http://www.irrodl.org/content/v3.2/tech11.html

Gill, T. (2000). Metadata and the World Wide Web. *Getty website, Introduction to Metadata.* Retrieved July 9, 2004, from http://www.getty.edu/research/conducting_research/standards/intrometadata/2_articles/gill/index.html

Gilliland-Swetland, A. J. (2000). Setting the stage. *Getty website, Introduction to metadata.* Retrieved July 9, 2004, from http://www.getty.edu/research/conducting_research/standards/intrometadata/2_articles/index.html

Godby, C. J., Smith, D., & Childress, E. (2003). Two paths to interoperable metadata. *The 2003 Dublin Core Conference, DC-2003: Supporting Communities of Discourse and Practice – Metadata Research & Applications, September 28 – October 2, in Seattle, Washington, USA*. Retrieved June 22, 2004, from http://www.oclc.org/research/publications/archive/2003/godby-dc2003.pdf

Gordon, J. (2002). Where oh where is plug & play? *Learning & Training Innovations Magazine: Online Version*. Retrieved July 11, 2004, from http://www.ltimagazine.com/ltimagazine/article/articleDetail.jsp?id=41961

Greer, L. R. (2002). The learning matrix: cataloging resources with rich metadata, *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, July 2002*, pp. 375.

Harms, D. (2001). *JSP, Servlets, and MySQL*. New York: M&T Books.

Harold, E. R. (2002). *Processing XML with Java: a guide to SAX, DOM, JDOM, JAXP, and TrAX*. Boston: Addison-Wesley.

Haynes, D. (2004). *Metadata for information management and retrieval*. London: Facet Publishing.

Heery, R. (1996). Review of metadata formats. *Program, 30(4), October 1996*, pp. 345-373.

Hodgins, W. (2000a). *Everything you ever wanted to know about learning standards but were afraid to ask*, Retrieved March 1, 2004, from http://www.linezine.com/2.1/features/wheyewtkls.htm

Hodgins, W. (2000b). The future of learning objects. In D. A. Wiley (Ed.), *The Instructional Use of Learning Objects: Online Version*. Retrieved March 5, 2004, from http://reusability.org/read/chapters/hodgins.doc

Hodgins, W., & Duval, E. (2004). Learning object metadata (LOM). *Presentation on CEN/ISSS + LTSC Meetings, Madrid Spain, January 2004*.

Hunt, J., & Loftus, C. (2003). *Guide to J2EE, Enterprise Java*. London: Springer.

Hunter, J., & Lagoze, C. (2001). Combining RDF and XML schemas to enhance interoperability between metadata application profiles. *Proceedings of the 10th International Conference on World Wide Web, April 2001* (pp. 457-466), New York: ACM Press.

IEEE LTSC. (2002). IEEE 1484.12.1-2002 draft standard for learning object metadata. Retrieved March 7, 2004, from http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

IEEE Std 1484.12.1. (2002). IEEE standard for learning object metadata. *IEEE*

Standards.

IEEE XML Binding P1484.12.3/D5. (2004). Draft Standard for Learning Technology – Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata. Retrieved December 2, 2004, from http://ltsc.ieee.org/wg12/files/IEEE_1484_12_03_d5_changes.pdf

IMS Guide. (2001). IMS learning resource meta-data best practice and implementation guide. Retrieved March 5, 2004, from http://www.imsproject.org/

IMS Metadata. (2001). IMS learning resource meta-data information model. Retrieved March 5, 2004, from http://www.imsproject.org/

IMS XML Binding. (2001). IMS learning resource meta-data XML binding. Retrieved March 5, 2004, from http://www.imsproject.org/

International Council on Archives. (2000). ISAD (G): General international standard archival description, second edition. Retrieved October 24, 2004, from http://www.ica.org/biblio/cds/isad_g_2e.pdf

ISO/IEC JTC1/SC29/WG11. (2002). *MPEG-21 overview*, Retrieved October 23, 2004, from http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm

ISO/IEC JTC1/SC29/WG11. (2003). *MPEG-7 overview*. Retrieved October 23, 2004, from http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

Karampiperis, P., & Sampson, D. (2003). Enhancing educational metadata management systems to support interoperable learning object repositories. *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies,* pp. 214-218.

Kotze, P. (2004). Learning object metadata application profiles. *Proceedings of SAICSIT 2004, pp. 111-113.*

Learning Objects Network. (2001). Capitalizing on the learning object economy: the strategic benefits of standard learning objects. *Learning Objects Network White Paper, July 16, 2001.* Retrieved April 25, 2004, from http://learningobjectsnetwork.com/resources-main.html

Lightle, K. S. (2004). Using metadata standards to support interoperability. In C.M. Gynn, & S.R. Acker (Eds.), *Learning objects: Contexts and connections* (pp. 43-48). Ohio: Ohio State University.

Lightle, K. S., Ridgway, J., & Smith, J. K. (2003). Generation of XML records across multiple metadata standards. *The 2003 Dublin Core Conference, DC-2003: Supporting Communities of Discourse and Practice – Metadata Research & Applications, September 28 – October 2, in Seattle, Washington, USA.* Retrieved July 10, 2004, from http://dc2003.ischool.washington.edu/Archive-03/03lightle.pdf

Margaret S. P. (1998). Issues in crosswalking content metadata standards. *National Information Standards Organization White Papers, 1998*. Retrieved March 23, 2004, from http://www.niso.org/press/whitepapers/crsswalk.html

Maruyama, H., Tamura, K., Uramoto, N., Murata, M., Clark, A., Nakamura, Y., et al. (2002). *XML and Java (2nd edition)*. Boston: Addison-Wesley.

McGee, P. (2004). *Learning Object Repositories*. Retrieved October 15, 2004 from http://elearning.utsa.edu/guides/LO-repositories.htm

McGreal, R. (2004). Learning objects: a practical definition. *International Journal of Instructional Technology & Distance Learning: Online Version, 1(9)*. Retrieved October 8, 2004 from http://www.itdl.org/Journal/Sep_04/article02.htm

McGreal, R., & Roberts, T. (2001). A primer on metadata for learning objects: fostering an interoperable environment. *Learning & training Innovations magazine: Online Version*. Retrieved March 22, 2004 from http://www.ltimagazine.com/ltimagazine/article/articleDetail.jsp?id=2031

Metros, S. E. & Bennett, K. (2002). Learning objects in higher education. *ECAR Research Bulletin, Volume 2002 (19), October 1, 2002*. Retrieved July 7, 2004, from http://www.educause.edu/ir/library/pdf/ERB0219.pdef

Millea, J. (2003). The EdNA metadata standard. *DC-ANZ Metadata Conference 2003*. Retrieved August 15, 2004 from http://www.educationau.edu.au/papers/edna_metadata.pdf

Mohan, P. (2004). Reusable online learning resources: problems, solutions and opportunities. *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT'04)*.

Mortimer, L. (2002). Learning objects of desire: promise and practicality. *ASTD's Source for Learning*. Retrieved June 30, 2004, from http://www.learningcircuits.org/2002/apr2002/mortimer.html

Neven, F., & Duval, E. (2002). Reusable learning objects: a survey of LOM-based repositories. *Proceedings of the 19th ACM International Conference on Multimedia, December 2002*, pp. 291-294.

Polsani, P. R. (2003). Use and abuse of resuable learning objects. *Journal of Digital Information, 3(4), article No. 164, February 19, 2003*.

Pöyry P., Pelto-Aho, K., & Puustjärvi, J. (2002). The role of metadata in the CUBER system. *Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*.

Robson, R. (2001). Learning object tutorial. Retrieved October 8, 2004, form http://www.eduworks.com/LOTT/tutorial/index.html

Saddik, A. E., Ghavam, A., Fischer, S., & Steinmetz, R. (2000). Metadata for smart multimedia learning objects. *Proceedings of the Australasian conference on Computing education* (pp. 87-94), New York: ACM Press.

Sampson, D. (2004). The evaluation of educational metadata: from standards to application profiles. *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT'04).*

Santacruz-Valencia, L. P., Aedo, I., & Delgado-Kloos, C. (2003). Designing le@rning objects with the ELO-tool. *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies.*

SCHEMAS Project. (2002). Eighth metadata watch report. Retrieved October 4, 2004 from http://www.schemas-forum.org/metadata-watch/

Shen, Z., Shi, Y., & Xu, G. (2002). A learning resource metadata management system based on LOM specification. *Proceedings of The 7th International Conference on Computer Supported Cooperative Work in Design*, pp. 452-457.

Shepherd, C. (2000). Objects of interest. *TACTIX.* Retrieved March 2, 2004, from http://www.fastrak-consulting.co.uk/tactix/features/objects/objects.htm

Simoes, D., Luis, R., & Horta, N. (2004). Enhancing the SCORM metadata model, *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, May 2004.*

Sumner, T., Bhushan, S., & Ahmad, F. (2003). A comparison of two educational resource discovery systems. *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries* (pp. 408-408), Washington DC: IEEE Computer Society.

Suthers, D. D., Johnson, S. M. & Tillinghast, B. (2001). Learning object metadata for a database of primary and secondary school resources. Retrieved March 22, 2004, from http://lilt.ics.hawaii.edu/lilt/papers/2001/suthers-et-al-ILE-2001.pdf

Taylor, A. (2003). *J2EE and beyond.* Upper Saddle River, New Jersey: Prentice Hall.

University of Wisconsin website. (2003). Learning objects: collections. Retrieved October 15, 2004, from http://www.uwm.edu/Dept/CIE/AOP/LO_collections.html

Van Assche, F., Campbell, L.M., Rifon, L. A., & Willem, M. (2003). Semantic interoperability: use of vocabularies with learning object metadata, *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies, pp. 511-514.*

Vossen, G., & Jaeschke, P. (2003). Learning objects as a uniform foundation for e-earning platforms. *Proceedings of the Seventh International Database Engineering and Application Symposium.*

Wagner, E. D. (2002). The new frontier of learning object design. *The e Learning*

Developer's Journal, June 18, 2002.

Weibel, S. (1995). Metadata: the foundation of resource description. *D-Lib Magazine: Online Version, July 1995*. Retrieved August 24, 2004, from http://www.dlib.org/dlib/July95/07weibel.html

Wiley, D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D. A. Wiley (Ed.), *The Instructional Use of Learning Objects: Online Version*. Retrieved March 5, 2004, from http://reusability.org/read/chapters/wiley.doc

Woodley, M. (2000). Crosswalks: the path to universal access? *Getty website, Introduction to metadata.* Retrieved July 9, 2004, from http://www.getty.edu/research/conducting_research/standards/intrometadata/2_articles/woodley/index.html

Yin, Z., Xu, Z., & Saddik, A. E. (2003). Study of metadata for advanced multimedia learning objects. *Canadian Conference on Electrical and Computer Engineering, May 4-7,2003. IEEE CCECE 2003, Volume: 2*, pp. 1099 –1102.

Websites Referred to in the Text of the Thesis:

ARIADNE Educational Metadata Recommendation website, http://www.ariadne-eu.org/en/publications/metadata/

Australian Government Locator Service Metadata Guidelines website, http://www.agimo.gov.au/practice/delivery/examples/agls_metadata/guidelines

DCMI website, http://www.dublincore.org/

EDItEUR website, http://www.editeur.org/onix.html

EdNA Metadata Standard website, http://www.edna.edu.au/edna/go/pid/333

GILS-global website, http://www.gils.net/

GILS-government website, http://www.access.gpo.gov/su_docs/gils/

IMS Global Learning Consortium website, http://www.imsproject.org/

JPEG2000 Metadata website, http://www.jpeg.org/jpeg2000/metadata.html

MARC Standards website, http://www.loc.gov/marc/

MPEG website, http://www.chiariglione.org/mpeg/

UK GovTalk website, http://www.govtalk.gov.uk/schemasstandards/metadata.asp

Appendix A Metadata Elements Mapping

A-1 Mapping Between Dublin Core and IEEE LOM

(IEEE Std 1484.12.1 (2002) & IMS Version 1.2.1 Best Practice and Implementation Guide)

Dublin Core data element	LOM data element
DC.Identifier	1.1.2:General.Identifier.Entry
DC.Title	1.2:General.Title
DC.Language	1.3:General.Language
DC.Description	1.4:General.Description
DC.Subject	1.5:General.Keyword or 9:Classification with 9.1:Classification.Purpose equals "Discipline" or "Idea."
DC.Coverage	1.6:General.Coverage
DC.Type	5.2:Educational.LearningResourceType
DC.Date	2.3.3:LifeCycle.Contribute.Date when 2.3.1:LifeCycle.Contribute.Role has a value of "Publisher."
DC.Creator	2.3.2:LifeCycle.Contribute.Entity when 2.3.1:LifeCycle.Contribute.Role has a value of "Author."
DC.OtherContributor	2.3.2:LifeCycle.Contribute.Entity with the type of contribution specified in 2.3.1:LifeCycle.Contribute.Role.
DC.Publisher	2.3.2:LifeCycle.Contribute.Entity when 2.3.1:LifeCycle.Contribute.Role has a value of "Publisher."
DC.Format	4.1:Technical.Format
DC.Rights	6.3:Rights.Description
DC.Relation	7.2.2:Relation.Resource.Description
DC.Source	7.2:Relation.Resource when the value of 7.1:Relation.Kind is "IsBasedOn."
NOTES 1—The Dublin Core Metadata Initiative is also developing data element qualifiers to further refine the semantics of the Dublin Core data elements.[a] A further refinement of the mapping in Table B.1 can be based on these qualifiers. 2—The LOM working group is committed to working with the Dublin Core Metadata Initiative (DCMI) to develop interoperable metadata, as outlined in the Memorandum of Understanding between the IEEE LTSC LOM WG and the DCMI.[b]	
[a]http://dublincore.org/documents/dcmes-qualifiers/	
[b]http://standards.ieee.org/announcements/metaarch.html	

A-2 Mapping between ARIADNE, LOM and Dublin Core

(ARIADNE Initiative)

ARIADNE 3.1	LOM 6.1	Dublin Core
1.0 identifier	1.3.2:General.CatalogEntry.Entry, with 1.3.1:General.CatalogEntry.Catalog= 'Ariadne'	Identifier
1.1 title	1.2:General.Title	Title
1.2 authors	2.3.2:LifeCycle.Contribute.Entity, with 2.3.1:LifeCycle.Contribute.Role='Author'	Creator
1.3 date	2.3.3:LifeCycle.Contribute.Date, with 2.3.1:LifeCycle.Contribute.Role='Author' or 'Publisher'	Date
1.4 language	1.4:General.Language	Language
1.5 institution	2.3.2:LifeCycle.Contribute.Entity with 2.3.1: LifeCycle.Contribute.Role='Publisher'	Publisher
1.6 source	7.2.2:Relation.Resource.Description with 7.1:Relation.Kind='IsBasedOn'	Source
2.1 discipline type	9.2.2.2:Classification.TaxonPath.Taxon[1].Entry, with 9.1:Classification.Purpose='Discipline', and 9.2.1:Classification.TaxonPath.Source='Ariadne'	Subject
2.2 discipline	9.2.2.2:Classification.TaxonPath.Taxon[2].Entry, with 9.1:Classification.Purpose='Discipline' and 9.2.1:Classification.TaxonPath.Source='Ariadne' and 9.2.2.2:Classification.TaxonPath.Taxon[1]=2.1 discipline type	Subject
2.3 subdiscipline	9.2.2.2:Classification.TaxonPath.Taxon[3].Entry, with 9.1:Classification.Purpose='Discipline' and 9.2.1:Classification.TaxonPath.Source='Ariadne' and 9.2.2.2:Classification.TaxonPath.Taxon[1]=2.1 discipline type and 9.2.2.2:Classification.TaxonPath.Taxon[2]=2.2 discipline	Subject
2.4 main concept	9.2.2.2:Classification.TaxonPath.Taxon[4].Entry, with 9.1:Classification.Purpose='Discipline' and 9.2.1:Classification.TaxonPath.Source='Ariadne' and 9.2.2.2:Classification.TaxonPath.Taxon[1]=2.1 discipline type and 9.2.2.2:Classification.TaxonPath.Taxon[2]=2.2 discipline and 9.2.2.2:Classification.TaxonPath.Taxon[3]=2.3 subdiscipline	Subject
2.5 main concept synonyms	Alternative langstring values in 9.2.2.2:Classification.TaxonPath.Taxon[4].Entry, with 9.1:Classification.Purpose='Discipline' and 9.2.1:Classification.TaxonPath.Source='Ariadne' and 9.2.2.2:Classification.TaxonPath.Taxon[1]=2.1 discipline type and 9.2.2.2:Classification.TaxonPath.Taxon[2]=2.2 discipline and 9.2.2.2:Classification.TaxonPath.Taxon[3]=2.3 subdiscipline and 9.2.2.2:Classification.TaxonPath.Taxon[4].Entry=2.4 main concept	Subject
2.6 other concepts	9.2.2.2:Classification.TaxonPath.Taxon[4].Entry, of additional 9.2.2:Classification.TaxonPath.Taxon, with 9.1:Classification.Purpose='Discipline' and 9.2.1:Classification.TaxonPath.Source='Ariadne' and 9.2.2.2:Classification.TaxonPath.Taxon[1]=2.1 discipline type and 9.2.2.2:Classification.TaxonPath.Taxon[2]=2.2 discipline and 9.2.2.2:Classification.TaxonPath.Taxon[3]=2.3 subdiscipline	Subject
3.1 end user type	5.5:Educational.IntendedEndUserRole	
3.2 document type	5.1:Educational.InteractivityType	Type
3.3 document format	5.2:Educational.LearningResourceType	
3.4 didactical context	9:Classification, with	

	9.1:Classification.Purpose='Educational Level', and 9.2.1:Classification.TaxonPath.Source='Ariadne'	
3.4.1 country	9.2.2.2:Classification.TaxonPath.Taxon[1].Entry, with 9.1:Classification.Purpose='Educational Level', and 9.2.1:Classification.TaxonPath.Source='Ariadne'	
3.4.2 context	5.6:Educational.Context	
3.4.3 level	9.2.2.2:Classification.TaxonPath.Taxon[2].Entry, with 9.1:Classification.Purpose='Educational Level', and 9.2.1:Classification.TaxonPath.Source='Ariadne'	
3.5 difficulty level	5.8:Educational.Difficulty	
3.6 interactivity level	5.3:Educational.InteractivityLevel	
3.7 semantic density	5.4:Educational.SemanticDensity	
3.8 pedagogical duration	5.9:Educational.TypicalLearningTime	
3.9 granularity	1.9:General.AggregationLevel	
4.1 document handle	4.3:Technical.Location	
4.2 file media types	4.1:Technical.Format	Format
4.3 package size	4.2:Technical.Size	
4.4 operating system type	4.4.2:Technical.Requirements.Name, with 4.4.1:Technical.Requirements.Type='Operating System'	
4.5 OS version	4.4.3:Technical.Requirements.MinimumVersion, with 4.4.1:Technical.Requirements.Type='Operating System', and 4.4.2:Technical.Requirements.Name=4.4 operating system type	
4.6 other platform requirements	4.6:Technical.OtherPlatformRequirements	
4.7 installation remarks	4.5:Technical.InstallationRemarks	
5.1 access rights	6.1:Rights.Cost	Rights
5.2 restrictions	6.2:Rights.CopyrightAndOtherRestrictions	Rights
5.3 usage remarks	6.3:Rights.Description	
6.1 author	3.3.2:MetaMetaData.Contribute.Entity, with 3.3.1:MetaMetaData.Contribute.Role='Creator'	
6.2 creation date	3.3.3:MetaMetaData.Contribute.Date, with 3.3.1:MetaMetaData.Contribute.Role='Creator', and 3.3.2:MetaMetaData.Contribute.Entity=6.1 author	
6.3 last modified date	Chronologically last 3.3.3:MetaMetaData.Contribute.Date	
6.4 language	3.5:MetaMetaData.Language	
6.5 validator	3.3.2:MetaMetaData.Contribute.Entity, with 3.3.1:MetaMetaData.Contribute.Role='Validator'	
6.6 validation date	3.3.3:MetaMetaData.Contribute.Date, with 3.3.1:MetaMetaData.Contribute.Role='Validator', and 3.3.2:MetaMetaData.Contribute.Entity=6.5 validator	
7.1. annotator	8.1: Annotation.Person	
7.2. creation date	8.2: Annotation.Date	
7.3. content	8.3: Annotation.Description	

A-3 Mapping between LOM, GEM and EdNA

LOM	GEM	EdNA
1 General		
1.1 Identifier	DC. Identifier	DC. Identifier
1.1.1 Catalog		
1.1.2 Entry		
1.2 Title	DC. Title	DC. Title
1.3 Language	DC. Language	DC. Language
1.4 Description	DC. Description	DC. Description
1.5 Keyword	DC. Subject	DC. Subject
1.6 Coverage	DC. Coverage	DC. Coverage
1.7 Structure		
1.8 Aggregation Level		
2 Life Cycle		
2.1 Version		
2.2 Status		
2.3 Contribute	DC. Contributor [when Role is neither "Author" nor "Publisher"] DC. Creator [when Role = "Author"] DC. Publisher [when Role = "Publisher"]	DC. Contributor [when Role is neither "Author" nor "Publisher"] DC. Creator [when Role = "Author"] DC. Publisher [when Role = "Publisher"] EDNA. Reviewer [when Role = "validator"]
2.3.1 Role		
2.3.2 Entity		
2.3.3 Date	DC. Date [when Role = "Publisher"]	DC. Date [when Role = "Publisher"]
3 Meta-Metadata		
3.1 Identifier		
3.1.1 Catalog		
3.1.2 Entry		
3.2 Contribute	GEM. Cataloging [when Role = "Creator"]	
3.2.1 Role		
3.2.2 Entity		
3.2.3 Date		
3.3 Metadata Schema		EDNA. Version
3.4 Language		
4 Technical		
4.1 Format	DC. Format	DC. Format
4.2 size		
4.3 Location	DC. Identifier	DC. Identifier
4.4 requirement		
4.4.1 OrComposite		
4.4.1.1 Type		

4.4.1.2 Name		
4.4.1.3 Minimum Version		
4.4.1.4 Maximum Version		
4.5 Installation Remarks		
4.6 Other Platform Requirements		
4.7 Duration		
5 Educational		
5.1 Interactivity Type		
5.2 Learning Resource Type	DC. Type	DC. Type
5.3 Interactivity Level		
5.4 Semantic Density		
5.5 Intended End User Role	GEM. Audience	EDNA. Audience
5.6 Context	GEM. Grade	
5.7 Typical Age Range	GEM. Grade	EDNA. Audience
5.8 Difficulty		
5.9 Typical Learning Time	GEM. Duration	
5.10 Description		
5.11 Language		
6 Rights		
6.1 Cost		
6.2 Copyright and Other Restrictions		
6.3 Description	DC. Rights	DC. Rights
7 Relation	DC. Relation	DC. Relation
7.1 Kind		
7.2 Resource	DC. Source [when Kind = "IsBasedOn"] GEM. EssentialResources [when Kind = "Requires"]	DC. Source [when Kind = "IsBasedOn"]
7.2.1 Identifier		
7.2.1.1 Catalog		
7.2.1.2 Entry		
7.2.2 Description		
8 Annotation		
8.1 Entity		EDNA. Reviewer
8.2 Date		
8.3 Description		EdNA. Review
9. Classification	DC. Subject [when Purpose = "discipline"]	DC. Subject [when Purpose = "discipline"]
9.1 Purpose		
9.2 Taxon Path		
9.2.1 Source		
9.2.2 Taxon		
9.2.2.1 Id		
9.2.2.2 Entry		
9.3 Description		

	9.4 Keyword	DC. Subject	DC. Subject
		GEM. Quality	
		GEM. Standards	
			EDNA. Approver (solely for EdNA online)
			EDNA. CategoryCode (↑)
			EDNA. Entered (↑)
			EDNA. Indexing (↑)

Appendix B XML Schema Files

B-1 Schema dc.xsd

1

schema location: http://post.massey.ac.nz:8080/metawork/xsd/dc/dc.xsd
targetNamespace: http://post.massey.ac.nz/metawork/dc/

Elements	Complex types
metadata	dcmetaType

2

schema location: http://post.massey.ac.nz:8080/metawork/xsd/dc/qlfdc.xsd
targetNamespace: http://purl.org/dc/elements/1.1/

Elements	Groups	Complex types
contributor	elementsGroup	contributorType
coverage		coverageType
creator		dateType
date		formatType
description		relationType
format		
identifier		
language		
publisher		
relation		
rights		
source		
subject		
title		
type		

3

schema location: http://post.massey.ac.nz:8080/metawork/xsd/dc/xml.xsd
targetNamespace: http://www.w3.org/XML/1998/namespace

Attr. groups
specialAttrs

4

schema location: http://post.massey.ac.nz:8080/metawork/xsd/dc/xsi.xsd
targetNamespace: http://www.w3.org/2001/XMLSchema-instance

5

schema location: http://post.massey.ac.nz:8080/metawork/xsd/dc/dcterms.xsd
targetNamespace: http://purl.org/dc/terms/

Elements	Complex types
ContentProvider	elementType
created	
Editor	
EducationalValidator	
extent	
GraphicalDesigner	
HasFormat	
HasPart	
HasVersion	
Initiator	
InstructionalDesigner	

IsBasedOn
IsBasisFor
IsFormatOf
IsPartOf
IsReferencedBy
IsRequiredBy
issued
IsVersionOf
References
Requires
ScriptWriter
spatial
TechnicalImplementer
TechnicalValidator
Terminator
Unknown
Validator

B-2 Schema edna.xsd

1

schema location: http://post.massey.ac.nz:8080/metawork/xsd/edna/edna.xsd
targetNamespace: http://post.massey.ac.nz/metawork/edna/

Elements	Complex types
metadata	ednametaType

2

schema location: http://post.massey.ac.nz:8080/metawork/xsd/edna/qlfdc.xsd
targetNamespace: http://purl.org/dc/elements/1.1/

Elements	Groups	Complex types
contributor	elementsGroup	contributorType
coverage		coverageType
creator		dateType
date		formatType
description		relationType
format		
identifier		
language		
publisher		
relation		
rights		
source		
subject		
title		
type		

3

schema location: http://post.massey.ac.nz:8080/metawork/xsd/edna/ednaterms.xsd
targetNamespace: http://www.edna.edu.au/edna/go/pid/333

Elements
approver
audience
categoryCode
entered
indexing
review
reviewer
version

4

schema location: http://post.massey.ac.nz:8080/metawork/xsd/edna/xml.xsd
targetNamespace: http://www.w3.org/XML/1998/namespace

Attr. groups
specialAttrs

5

schema location: http://post.massey.ac.nz:8080/metawork/xsd/edna/xsi.xsd
targetNamespace: http://www.w3.org/2001/XMLSchema-instance

6

schema location: http://post.massey.ac.nz:8080/metawork/xsd/edna/dcterms.xsd
targetNamespace: http://purl.org/dc/terms/

Elements
ContentProvider
created
Editor
EducationalValidator
extent
GraphicalDesigner
HasFormat
HasPart
HasVersion
Initiator
InstructionalDesigner
IsBasedOn
IsBasisFor
IsFormatOf
IsPartOf
IsReferencedBy
IsRequiredBy
issued
IsVersionOf
References
Requires
ScriptWriter
spatial
TechnicalImplementer
TechnicalValidator
Terminator
Unknown
Validator

Complex types
elementType

Appendix C Acronyms

ADL	Advanced Distributed Learning initiative
AGLS	Australian Government Locator Service
AICC	Aviation Industry CBT Committee
AICTEC	Australian Information and Communications Technologies in Education Committee
API	Application Programming Interface
ARIADNE	Alliance of Remote Instructional Authoring and Distribution Networks for Europe
CAM	SCORM Content Aggregation Model
CBT	Computer Based Training
DCMES	Dublin Core Metadata Element Set
DCMI	Dublin Core Metadata Initiative
DOM	Document Object Model
DTD	Document Type Definition
EAD	Encoded Archival Description
EdNA	Education Network Australia
eGMS	e-Government Metadata Standard
FGDC	Federal Geographic Data Committee
GEM	Gateway to Educational Materials
GILS	Government Information Locator Service
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
IEEE	Institute of Electrical & Electronics Engineers
IIS	Microsoft's Internet Information Server
IMS	Instructional Management System consortium
ISAD	International Standard Archival Description
ISBN	International Standard Book Number
ISO	International Standard Organization
ISO/IEC JTC1 SC	ISO / International Electrotechnical Commission Joint Technical Committee 1 Sub Committee
J2EE	Java 2 Platform, Enterprise Edition
J2SE	Java 2 Platform, Standard Edition

JAXP	Java API for XML Processing
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JPEG	Joint Photographic Experts Group
JSP	Java Server Page
JVM	Java Virtual Machine
LCMS	Learning Content Management System
LMS	Learning Management System
LOM	IEEE standard for Learning Object Metadata
LTSC	IEEE Learning Technology Standards Committee
MARC	MAchine-Readable **C**ataloguing
MPEG	Moving Picture Experts Group
MSL	Metadata Specification Language
NCSA	National Center for Supercomputing Applications
OCLC	Online Computer Library Center
ODBC	Open DataBase Connectivity
RDF	Resource Description Framework
RDN	Resource Discovery Network
SAX	Simple API for XML
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
UKOLN	UK Office for Library and Information Networking
UML	Uniform Modeling Language
URI	Uniform Resource Identifier
URL	Universal Resource Locator
URN	Uniform Resource Name
SEL	Standard Extension Library
W3C	World Wide Web Consortium
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language
XSLT	XSL Transformatins