THE USE OF KNOWLEGDE BASES AS REPOSITORIES IN LEARNING ENVIRONMENTS

by

CLEMENTINE T. MUPFUMIRA

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Department of Computer Science
Faculty of Science
University of Zimbabwe

February 2006

Abstract

This dissertation proposes a method of management of Learning Objects Repositories (LORs). It looks at both the storage and search of learning objects (LOs) in the LORs. The focus is on how LORs can be structured in order to promote easier and more efficient search and thus furnishing users with relevant results. The model proposed defines a “neighbourhood”, a concept achieved by grouping LOs by purpose (genre-classification) using metadata elements from IEEE’s LOM. The elements chosen, which fall under the Educational and General groups, are Learning Resource Type and Keyword. This can then be put into a Knowledge base to enable users to get additional results based on searches done by previous users who requested similar objects. This involves the exploitation of similarities between users (including the user conducting the search) to suggest additional LOs that could be relevant. Two algorithms have been developed that use this model, one for searching and the other for insertion.

Acknowledgements

The successful completion of this dissertation would not have been possible without the assistance of Dr Gilford Hapanyengwi, who not only served as my supervisor, but also encouraged and challenged me in pursuit of my goal. He patiently guided me throughout my research and never accepted less than my best effort. I thank him

I would also like to thank the Department of Computer Science at the University of Zimbabwe, for their material and intellectual support.

Lastly, but definitely not least, I would like to extend my heartfelt appreciation to my family and friends whose love and support saw me through the ups and downs of this worthwhile journey

CONTENTS

TABLES

FIGURES

Chapter 1

Introduction

1.1 Problem statement

Learning is no longer restricted to a classroom with charts on the wall, or notes on a black (or white) board, and a pile of textbooks. Students and their teachers are now able to maintain a high standard of education at the click of a button. A lot of content, be it academic or otherwise can be found electronically these days. The sources of this information can be, but is not limited to, CD-ROMs and the Internet. As the Internet expands, CD-ROMs become cheaper to create, and more and more information is being digitalized there is need for ways in which those involved in electronic learning (e-learning) can get the best out of their experience with the electronic world. Databases of all kinds have been set up to try and put together information that would be relevant to a particular community or context and enable faster more relevant searches, with minimal reference to the larger pool of information (the Internet).

Many such databases, called learning object repositories (LORs), have been set up by various communities to enable their members to access and share information with one another. Examples or such communities are Ariadne, Merlot, SMETE and CanCore. This information can be in any form that can be supported electronically. Although the systems in place are working well to meet community needs, work still needs to be done to ensure that systems are less complicated and offer a relevant result to the user that would satisfy their need

The main thrust of this thesis is to look at ways that the e-learning experience can be improved by giving the user the information that they asked for, that is, the right object(s).

1.2 Aims and objectives of the research

The purpose of this research is to find and explore ways the correct information can be delivered to a user. It also looks at how Knowledge Bases can be used as a repository in

Learning Environments. The main focus is on structure of the repository and its content, and the main goal is to come up with solution which offers efficient storage and search (retrieval) of learning objects.

The thesis is organized in the following way:

Chapter 1 (Introduction):

This chapter introduces the research topic and sets the direction in which the thesis will proceed.

Chapter 2 (Literature Review):

A review of all the information that is relevant to the research.

Chapter 3 (Methodology):

In this chapter the proposed solution is presented. It is explained and analysis of graphs and algorithms is made.

Chapter 4 (Conclusion and Recommendations)

This chapter gives the conclusions drawn from the research and its results as well as recommendations on future work that can be done.

Chapter 2

Literature review

2.1 Introduction

The advent of the web has given instructors and learners a convenient way to view and exchange learning materials, in the form of digital learning objects (LOs), especially for long distance learning. These objects can be stored in repositories, built on database technology. These repositories offer mechanisms that encourage the discovery, reuse and exchange of LOs. In this chapter I will review relevant literature on LOs, their repositories and Knowledge Bases.

2.2 Knowledge-based systems

A Knowledge-Based System (KBS) is a program for extending and/or querying a knowledge base. The related term, expert system (ES), is normally used to refer to a highly domain-specific type of KBS used for a specialized purpose such as medical diagnosis, for instance MYCIN.[6]

Knowledge base systems have two modules; the knowledge base and the inference engine**.** The advantage of such a split is that it makes it easier to add new knowledge either during program development or during the program's lifetime. As knowledge is represented explicitly in the knowledge base, not implicitly in the program structure, it can be altered with relative ease. Some of the elements of knowledge-based systems include:

- techniques for acquiring knowledge
- ways of representing knowledge internally. Computers are good at representing numbers, words, even maps, but knowledge is potentially more difficult
- search procedures for working with the internally stored knowledge
- interference mechanism for deducing to problems from stored knowledge [21]

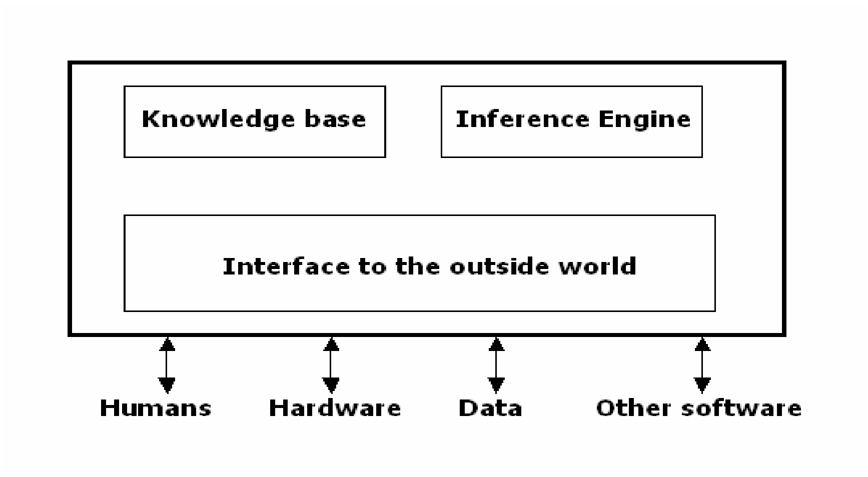

Figure 1: Knowledge base system

2.2.1 Knowledge Base

A knowledge base is a structured repository of domain specific electronic information. It can be thought of as an interactive encyclopedia except that it is designed to help with decision-making and problem-solving in a specialized field.[52]

A knowledge base may also include unstructured or unformalized information expressed in natural language or procedural code. By itself it does not actually do anything. Inference engines provide a means of accessing the base. ES tools provide one or more knowledge representation schemes for expressing knowledge about the application domain. Some tools use both frames (objects) and IF-THEN rules. [46]

When there is a view of constructing an ES the expertise have to first be acquired from the human experts. It is hard to extract human expertise. There are two approaches to knowledge acquisition:

- by asking experts to break down their knowledge into individual facts, rules, etc
- by deducing rules from the behavior of experts. [53]

With reference to Figure 2: Architecture of an Expert System, the knowledge base holds the expertise that the system can deploy. For many expert systems knowledge representation is through the use of rules (could also be frames, semantic nets etc). As it is being used, some facts, representing observations about the domain, are added to the working memory. The inference engine permits new inferences to be made from the knowledge in the knowledge base. These new facts represent conclusions about the state of the domain given the observations. For a rule-based expert system, the inference engine would be a mechanism for carrying out forward and/or backward chaining [64].

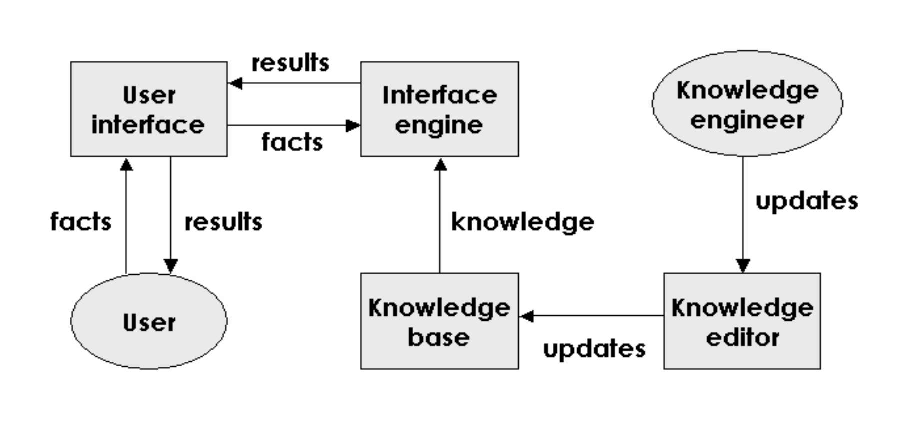

Figure 2: Architecture of an Expert System

Reasoning in an ES is performed using inference techniques and control strategies. Inferencing techniques guide the system as it combines knowledge contained in its knowledge base with problem facts contained in its working memory [13].

An inference engine is the component of an ES, such as a cluster of connected processors and associated software, which applies principles of reasoning to draw conclusions from the information stored in a knowledge base [63]. Inference engine types depend on the nature and complexity of the knowledge with which they are dealing. Two types of inference engine can be distinguished: data driven and event driven.

Data driven: A KBS in this mode takes available information (the “given” facts) and generates as many derived facts as it can. This approach could be used for interpreting problems where we want to know whatever the system can tell us about some data.

Goal-driven: This approach is appropriate when more tightly focused solutions are required [21]

2.2.4 Knowledge Representation

Knowledge is an abstract term that attempts to capture the understanding that individuals have on a given subject, subject area or domain. Knowledge representation (KR) is the method used to encode knowledge in an ES's knowledge base. Understanding how the knowledge in a knowledge base is represented is necessary as this helps to see how the knowledge can be stored and retrieved in the system.

There are many ways of representing knowledge in KBS. Some of the forms of representation include Natural Language Representation, Rules, and Predicate Calculus. Representing knowledge using rules as opposed to "conventional" programming techniques has an advantage in the maintainability of the KB. Programs written as a set of rules may also be comprehensible to the non-programmer [21].

Information about a particular object or idea can be grouped and represented as a frame. Each frame has a number of slots, which may be filled with; for example, attribute values (height, length, age, colour, etc), links to other frames, rules and instructions. The knowledge base can also be based on a semantic net of interrelated concepts.

2.2.4.1 Types of Knowledge

There is no single, ideal knowledge representation structure. Choice of technique has to be based on its suitability for a given application. The types of knowledge are:

DECLARATIVE KNOWLEDGE: Describes what is known about a problem, including simple statements that are either true or false, as well as a list of statements that describe some object or concept more fully

META-KNOWLEDGE (Knowledge about knowledge): It is used to select other knowledge that is best suited for solving a problem as well as to enhance the efficiency of problem solving by directing their reasoning into most promising areas.

HEURISTIC KNOWLEDGE: Describes rule-of-thumb which guides the reasoning process. It is often called *shallow knowledge.* It is empirical and represents the knowledge collected by an expert through the experience of solving past problems. Fundamental (deep) knowledge is often compiled into simple heuristics to help in problem solving. It includes fundamental laws, functional relationships, and so on.

STRUCTURAL KNOWLEDGE: Describes knowledge structure and the expert's overall mental model of the problem. This model is of concepts, sub-concepts, and "objects".

Each of the numerous knowledge representation techniques puts emphasis on particular information about a problem while ignoring other information. Choosing the right representation for a given application produces a structure that supports effective problem solving [13]. Five of these techniques are: Object-attribute-value triplets, Rules, Semantic networks, Frames, and Logic.

2.2.4.1.1 Object-attribute-value triplets

In ESs, facts are used to help describe parts of frames, semantic networks or rule, relationships between more complex knowledge structures and to control these structures during problem solving. It is often referred to as a proposition. The Boolean value of the statement is affirmed into working memory and used in the processing of other knowledge. A fact can also be used to affirm a certain property of some object, for example "the car's colour is blue" assigns the value "blue" to the car's colour. This type of fact is an Object-Attribute-Value (O-A-V) triplet, a more complex type of proposition.

O-A-V divides a statement into three parts, namely:

1. ***object:*** represented can be physical as well as abstract, for example garage
2. ***attribute:*** property or feature of object. It is an important problem consideration
3. ***attribute value:*** specifies the attributes assignment. Can be Boolean, numeric, or string.

As an example, "car" is the object, "colour" is the attribute, and "blue" is the value. Multiple attributes can be defined for the object, with corresponding values [13]. Frames and semantic networks use multiple attributes to describe objects, as described later.

2.2.4.1.2 Logic

Logic is the oldest form of KR in a computer. Propositional logic and predicate calculus are techniques, which use symbols to represent knowledge and operators applied to the symbols to produce logical reasoning. Predicate calculus could be thought of as the assembly language of KR.

Propositional logic

Represents and reasons with propositions/statements that are either true or false. It assigns a symbolic variable to a proposition such as *A = The car will start*. Operators like AND, OR, NOT, IMPLIES, EQUIVALENCE allow for reasoning with various rule structures.

Predicate calculus (predicate logic)

This extension of propositional logic provides a finer representation of the knowledge e.g. instead of representing an entire proposition with a single symbol, such as *A = ball's colour is red*, the relationship of the knowledge can be described in the form of colour*(ball,red)*. Symbols are used to represent knowledge and these may represent either constants, predicates, variables or functions. Operations on these symbols can be done using propositional logic operators

Constants

These are used to name specific objects or properties about the problem

Predicates

In predicate calculus, a fact or proposition is divided into two parts: a predicate and an argument. Arguments represent object(s) of the proposition and the predicate an assertion about the object e.g. to represent the proposition "*John likes Mary*", you would write: *likes(john,mary).* The first word of the expression (e.g. like), is a predicate denoting some relationship between the arguments within the parentheses. Predicates always appear with first character in lower case.

Variable

These are for the representation of general classes of objects or properties. Symbols have first letter uppercase

likes(X,Y)   X = john,bob   Y = mary, judy

Variables can be used as arguments in a predicate expression or a function

<u>Functions</u>

Symbols can be used to represent a function. A function denotes a mapping from the entities of a set to a unique element of another set, e.g.

father(jack) = bob

Functions can also be used within predicates

friends(father(jack),mother(judy)) = friends (bob,kathy)

<u>Operations</u>

Predicate calculus uses some operators found in propositional logic, e.g.

Proposition: John likes Mary likes(john,mary)

Proposition: Bob likes Mary likes(bob,mary)

To account for the obvious jealousy that might occur

likes(X,Y) AND likes(Z,Y) IMPLIES NOT likes(X,Z)

or likes(X,Y) ^ likes(Z,Y) -> ¬ likes(X,Z) => ¬likes(john,bob)

Predicate calculus operators provide reasoning ability to intelligent systems. Reasoning requires the ability to infer conclusions from available facts. Process of reasoning in ESs is called inferencing [13].

2.2.4.1.3 Semantic networks

This method provides a graphical view of a problem's important objects, properties and relationships. The node represents an object and an arc the relationships between objects. Expansion is by the addition of nodes and links to related node already in the network in one of three ways:

1. similar objects
2. more specific object
3. more general object

Some nodes, when added to a semantic network, automatically inherit information from the network, through IS-A links. If a general object is added, then other nodes inherit its properties. Inheritance is a powerful feature in a semantic network, but it can cause problems if a frame has a property unique to itself. Exception handling, achieved by explicitly encoding the value into the frame, can be used to avoid these problems. [13]

2.2.4.1.4 Frames

Frames are data structures for representing conventional knowledge of some concept or object that can be applied to a specific situation for study. It consists of slots that contain data, information, or procedures/algorithms to carry out some task [48]. They are a natural extension of the semantic network schema. A schema sometimes referred to as a frame, is a unit that contains typical knowledge about some concept, and includes both declarative and procedural knowledge.

The class frame represents general characteristics of some set of common objects. The properties and values form a list of O-A-V type statements that provide a good representation of the object. Property values are of three general types: Boolean, string, or numeric. They can be of two types:

- Static: describes an object feature whose value doesn't change.
- Dynamic: feature value is likely to change during the operation of the system.

An instance frame, used in describing specific instances of a class frame, inherits descriptive information and behavior from its class frame. When an instance is created, it is first asserted that the frame is an instance of some class. This is achieved by including, within a class frame, a procedure (method) that defines some action the frame performs.

Complex frame structures can be created apart from having only a single class and its associated instances. This structure organizes concepts at different levels of abstraction, as can be seen in the example shown in Figure 3.

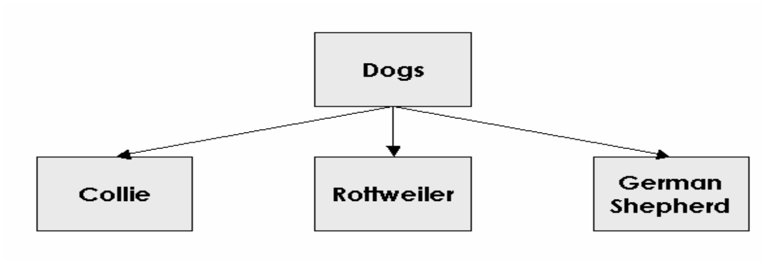

Figure 3: Dog Hierarchical structure

2.2.4.1.5 Rules

Rules are a form of procedural knowledge, which associates given information to some action that may be the assertion of new information or some procedure to perform. Simply, a rule describes how to solve a problem and is useless by itself. The structure of a rule logically connects one or more premises contained in the **IF** part, to one or more conclusions in the **THEN** part.

IF <premise> **THEN** <conclusion>

There can be multiple premises jointed with **AND** (conjunctions), **OR** (disjunction), **IF** <premise1>**AND** < premise2> **THEN** <conclusion>

IF <premise3> **OR** < premise4> **THEN** <conclusion>

or a combination

IF (<premise5> **OR** <premise6>) **AND** < premise7> **THEN** <conclusion>

ELSE statement can also be used.

IF <premise> **THEN** <conclusion1> **ELSE** <conclusion2>

Domain knowledge is captured in a set of rules and entered into the systems knowledge base. The system uses the rules along with information added to working memory. This can cause other rules to fire ("execute"). The inference engine module processes the rules in a rule base.

<u>Types of Rules</u>

RELATIONAL

IF The battery is dead
THEN The car will not start

RECOMMEDATION

IF The car will not start
THEN Take a cab

DIRECTIVE

IF The car will not start
AND The fuel system is ok
THEN Check out electrical system

STRATEGY

IF The car will not start
THEN First check out the fuel system then check out the electrical system

HEURISTIC

IF The car will not start
AND The car is a 1963 ford
THEN Check the float

Rules can be categorized according to the nature of the problem solving strategy or paradigm. The following list shows typical rules found in some of the common paradigms:

Interpretation problem:	**IF**	Voltage of register R1 is greater than 2.0 volts
	AND	The collector voltage Q1 is less than 1.0 volts
	THEN	The pre-amp section is in the normal range
Diagnostic problem:	**IF**	The strain of the organism is grampus
	AND	The morphology of the organism is coccus
	AND	The growth of the organism is chains
	THEN	There is evidence that the organism is streptococcus
Design problem:	**IF**	Current task is assigning a power supply
	AND	Position of power supply in the cabinet is known
	AND	There is space available in cabinet for power supply
	THEN	Put the power supply in the cabinet.

Variable Rules

There may be need to perform the same operation on a set of similar objects. Writing a rule for every object would be inefficient and would make system maintenance difficult. Pattern-matching rules offer efficient ways to process information. One rule can be written instead of many. This eases system coding and permits direct capture of general directions obtained from an expert. Frame-based systems employ these rules extensively

Uncertain rules

An expert will often provide a rule that establishes an inexact association between the premise and conclusion, for example

IF Inflation is high
THEN Almost certainly interest rates are high

To capture the confidence held on the association, certainty factors (CF) are used. For instance: **IF** Inflation is high
THEN Interest rates are high CF=0.8

CF value range is from 0 to 1, where 0 represents complete uncertainty, and 1 is complete certainty

Meta-rules

These rules describe how other rules are to be used. Meta-knowledge is often represented in a meta-rule to direct consultation e.g.

IF The car will not start **AND** The electrical system is operating normally

THEN Use rules concerning the full system

This diagnostic meta-rule directs the system to check the full system if no problems have been found in the electrical system. Meta-rules can be used to enhance system efficiency by directing the reasoning into the most promising areas.

Rule sets

Through experience, an expert forms several sets of rules to apply to a given problem. One set of rules may be applicable to a given problem while being useless for another problem. These rules can be organized into rule sets in a hierarchical fashion. Figure 4 shows and example of this.

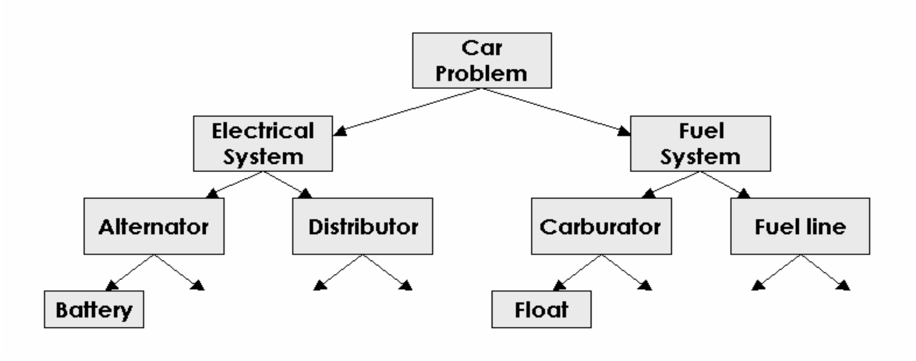

Figure 4: Rule Sets for Car Problems

Each block in Figure 4 represents a set of rules related to the label. Upper level blocks contain a set of abstract rules for general problem solving in order to direct the system's reasoning towards a more detailed investigation. This type of modular structure proposes a top-down approach to problem solving, where certain rule sets are used only when appropriate. The main advantage of this design is that it represents a natural approach to solving complex problems – divide and conquer. It also eases the system's development and maintenance. [13]

Blackboards

A Blackboard is a design in which several ESs share information contained in a common source (the knowledge source). It addresses how modules communicate information with one another, from the viewpoint of the cooperating ES. Distributed problem solving is accomplished in ESs through communicating rule sets.

2.3 Learning objects

The IEEE working group that defined the IEEE Learning Object Metadata standard (section 2.4.2.2) defined LOs as being an entity, digital or non-digital, that may be used for learning, education or training [3]. Digital LOs may be lesson content stored as text, audio, etc [18].

Another definition has LOs as educational resources that can be employed in technology-supported learning [25]. This simple statement means that the learning objects are inherently of electronic/digital form.

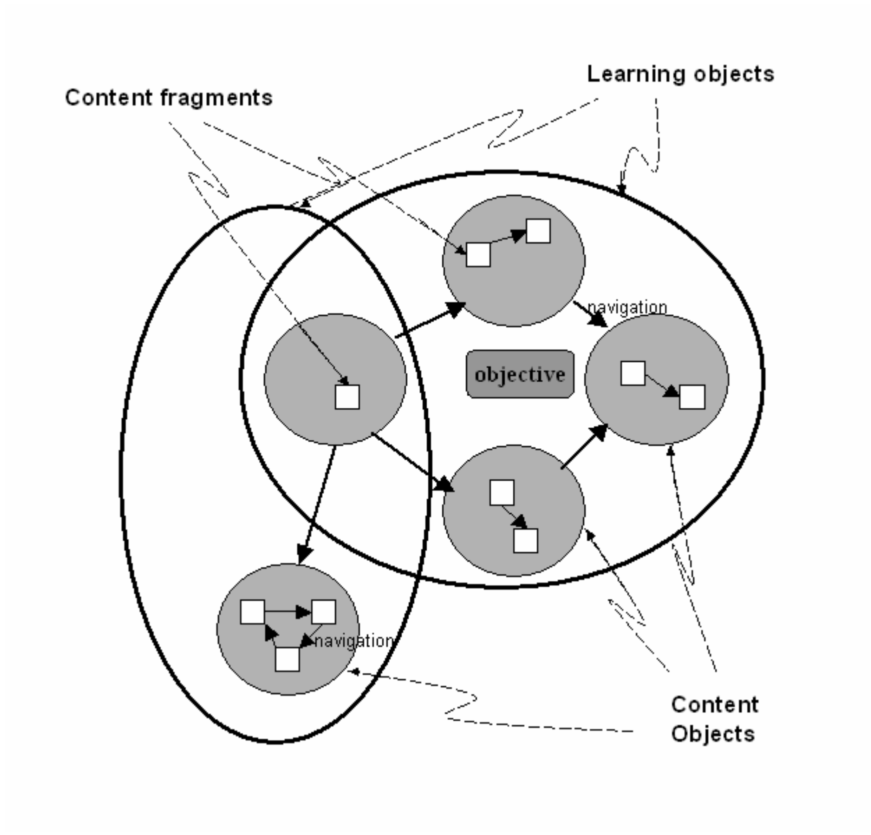

Figure 5: General Learning Object Content Model

There are number of learning object content models that have been developed. Some of these models where analyzed in order to address questions about the repurposing of learning objects in a different context [28]. Content models were mapped to a general model that the authors in [28] had come up with. The models compared were: Learnativity Content Model, Microsoft model, ADL academic co-lab model, SCORM content aggregation model, CISCO RLO/RIO model and NETg learning object model. The general model they produced, shown in Figure 5, is the one they compared these models with. It roughly outlines learning objects and their components. They found that all models could be mapped to the general model and hence a learning object developed using one model could be used in another model if the LO is in a subset of both models.

2.4 Metadata

Metadata, which is essential for addressing LOs, provides the basic framework to catalogue the intended purpose of the LO together with a description of the characteristics, and features of the LOs [33].

2.4.1 Application Profiles

Different needs are catered for by different LORs. Because of this specific metadata elements may be selected, together with the value sets they derive their values from. This selection is done by the designer [20]. This specification of metadata elements and value sets is called an "application profile"(AP). APs are used to adapt metadata specifications to the requirements of the local community[15]. Examples of APs are SCORM and ARIADNE [27]. XML or RDF conceptually map data elements and value sets of a profile into elements and value set of a standard schema. This enables the representation of the syntax and semantics of metadata elements (and their value sets) in an AP [16].

Application profiles (APs) can be classified according to how metadata elements and value sets have been selected:

Complete-Set: The whole set of elements and value sets of a particular standard is present in the AP. Local extensions for elements and vocabularies may be customize elements to a particular purpose (e.g. CELEBRATE (Context eLearning with Broadband Technologies))

Subset: Elements and value sets in the AP are a subset of an existing standard. Local extensions can also be made (e.g. ARIADNE).

Multi-Source: Elements and value sets in the AP are selected from more than one standard. A variety of extensions may be made (e.g. MILO application profile of LOM and Dublin Core).

Ad-Hoc: elements and value sets do not belong to any standard. Only the local community uses them. [27]

Value sets for metadata elements in APs can be defined in different ways in order to tailor these value sets to the requirements of the local community:

Complete-Set: value set of the element in profile is the complete set from the standard

Sub-Set: value set of the element is a subset of the value set the element in the standard

Super-Set: value set of the element extends the value set of the element in the standard
Multi-Source: value set of the element is selected from value sets of multiple standards
Ad-Hoc: value set of the element does not belong to any standard value set. [27]

A community can define the elements as mandatory, recommended, conditional or optional. The type of AP determines the level of interoperability achievable between the local community and other communities of practice. [27]

2.4.2 Metadata standards

Metadata includes a listing of commonly defined fields for each LO which conform to a set of rules. The rules provide a means of creating, handling and storing data and electronically transferring information using common standards that enable interoperability. There are a number of standards that are in use today.

2.3.2.1 Dublin Core Metadata Element Set

Dublin Core Metadata Element Set, from the Dublin Core Metadata Initiative, is commonly known as Simple Dublin Core, and is standardizes as ANSI/NISO Z39.85 – 2001 [3]. The element set seems to be by far the most widely accepted and used set of metadata standards for core categories applicable to any Internet based content [36]. It provides a simpler more loosely defined set of elements with some overlap with LOM. It is useful for sharing metadata across a wide range of dissimilar services.

Almost all existing learning object metadata standards use the Dublin Core as a basis and then extend it with more specialized elements The fifteen Dublin Core elements are; *Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights* [36] (see Appendix 2). Work is being done to produce a set of terms, which will allow the Dublin Core Element Set to be used with greater semantic precision. This set of terms is called the Qualified Dublin Core [3].

2.4.2.2 IEEE Learning Object Metadata

[3] IEEE 1484.12.1 – 2002, also known as IEEE Learning Object Metadata (LOM), is a popular metadata model for the description of learning objects. The standard builds on the Dublin Core and is based on recommendations from IMS and the ARIADNE project. It is the

first part of a multipart standard, and describes the LOM data model, which specifies which aspects of a LO should be described and what vocabularies may be used for these descriptions, as well as how the model can be amended by additions or constraints.

Some of the main things LOM is designed to help achieve are:

- Creation of descriptions of learning resources that are well structured to help facilitate discovery, location, evaluation and acquisition.
- Sharing of descriptions of learning resources between resource discovery systems leading to a reduction in the cost of providing services.
- Tailoring of the resource descriptions to suit the specialized needs of a community, including suitable controlled vocabularies for classification, reducing or adding the number of elements that are described.
- Creators and publishers may use the LOM along with other specifications to "tag" learning resources with a description that can be associated with the resource providing information in a standard format.

The LOM comprises a hierarchy of elements shown in Appendix . There are 9 categories, each of which contains sub-elements, which may be simple elements that hold data, or may themselves be aggregate elements containing further sub-elements. The semantics of an element are determined by its context, i.e. they are affected by the parent or container element in the hierarchy and by other elements in the same container. Some elements may be repeated either individually or as a group The Dublin Core elements fit into several of these categories. None of the elements of the LOM are mandatory meaning a LOM instance with no values for any of the elements still conforms to the standard. [36]

The creation of application profiles allows communities of users to specify which elements and vocabularies they will use. Elements from LOM may be dropped and others from other metadata schemas can be brought in. Vocabularies can be supplemented with values that are appropriate for a particular community. LOM records can be moved between systems using a variety of protocols, for instance Open Archives Initiative Protocol for Initiative Protocol for Metadata Harvesting (OAI-PMH). Tools like Reload and Curriculum Online Tagging Tool allow the creation and/or storage of LOM or IMS LRM 1.2.1 instances.

2.4.2.3 Other standards

ARIADNE & IMS:

The Ariadne project, with IMS, produced a set of recommendations for educational metadata that helped form the basis of the IEEE LOM. Both organizations promote the use of their own metadata standards conforming to the LOM standards. They took a subset of the LOM and augmented them with extra elements of their own. Ariadne specifies a minimal set of mandatory elements for any learning resource along with some other optional ones. This is to address the conflict that exists between the following two principles they believe LORs should adhere to

1. metadata creation by LO authors or indexers should be as easy as possible and
2. search for useful LOs should be as easy as possible.

The minimal set should allow for relatively good search capabilities without being too much of a burden to create whereas IMS specifications follow the LOM and do not specify that any field elements must exist. [36]

2.5 Learning Object Repositories

LOs are stored in Learning Object Repositories (LORs) to be used for different kinds of educational purposes and by different end users. Understanding the end user behaviour helps in the improvement of repositories and the associated tools, which means that users are served in a more efficient way [28]. The purpose of these digital repositories is not just safe storage and delivery but also reuse and sharing.

The underlying motivations for establishing repositories differentiate them from other collections. Repositories form an intersection of interest for different communities of practice: digital libraries, research, learning, e-science, publishing, records management, and preservation. The key services that repositories might provide range over several functional areas:

- enhanced access to resources
- new modes of publication and peer review
- corporate information management (records management and content management systems)

- data sharing (re-use of research data, re-use of learning objects)
- preservation of digital resources. [19]

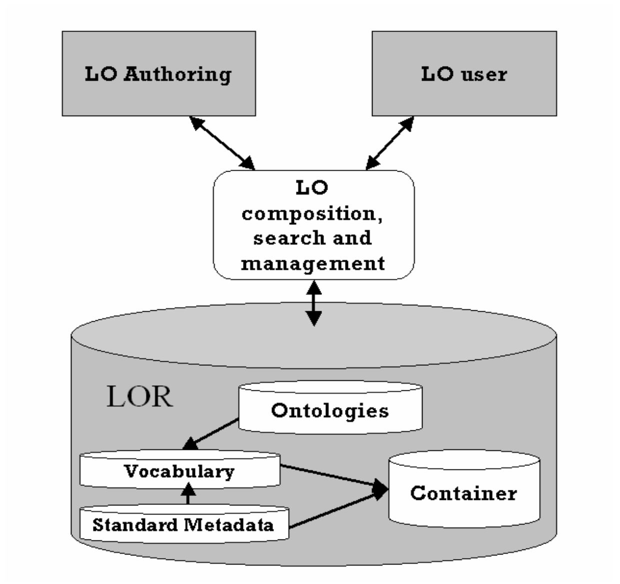

Figure 6: LOR internal architecture

Figure 6 shows the internal structure of a LOR. This structure typically includes metadata, educational content or service. The expressive power of a learning object management system would be given not only by its vocabulary, but also by its description and the abstraction levels that its definition makes possible. A structured container allows use of LOs. The objects should be stored in such a way as to provide a set of common services. Facilities are needed for searching, downloading, saving, indicating relationships between objects, inclusion of events associated with objects, storage of different "versions of the same object", and the provision of some degree of interoperability. [38]

An ontology, which can be thought of as a knowledge representation [62], is typically a hierarchical data structure containing all the relevant entities and their relationships and rules within that domain [57]. It is a framework of the semantic web that permits intelligent navigation [60]. A vocabulary, for describing objects and the relations between them in a formal way, controls it. It has a grammar for the use of the vocabulary terms for the expression of something meaningful within a specified domain [54].

There are two major types of LORs

- those containing both the LOs and the metadata: the repository is used to locate and deliver the learning objects
- those with metadata only: the LOS are located at a remote location. The repository is used as a tool to locate LOs

Most LORs are stand-alone. This means that they act like portals because they contain a web-based interface, a search mechanism, and a category listing [11]. Metadata can be stored using different means, such as file-based repositories, relational databases, XML repositories, or RDF tool kits, which use different query languages constituting a heterogeneous environment [35].

2.5.1 Examples of LORs

Ariadne KPS: The Ariadne project started in 1996. It is an integrated e-learning environment, incorporating a Knowledge Pool (which is the LOR), a web-based course environment and a number of content authoring tools [29]. The Knowledge Pool System (KPS) is a distributed repository (or tree-shaped network of replicating nodes) which encourages the sharing and reuse of LOs. An indexation and query tool uses a set of metadata elements to describe and enable search functionality on learning objects [28]. Data elements are grouped into six categories: General, Semantics, Pedagogical, Technical, Indexation, and Annotation. [26]

Merlot:(Multimedia Educational Resources for Learning and Online Teaching) Centralized repository containing metadata only and pointing to objects located at remote locations. It is stand-alone, acting like a portal for LOs. Merlot is a free and open resource designed primarily for faculty and students of higher education. [49]

CAREO (Campus Alberta Repository of Educational Objects)**:** This is a project supported by Alberta Learning and CANARIE (Canadian Network for the Advancement of Research in Industry and Education). Its primary goal is the creation of searchable, Web-based collection of multidisciplinary teaching materials for educators. This repository is an ongoing research prototype [42]. It is a centralized, stand-alone repository containing metadata and providing access to LOs located on remote web servers.

National SMETE Distributed Library: In the development for the SMETE (Science, Mathematics, Engineering, and Technology Education;), NSDL is intended as a "federation" of LORs, with each library using different document formats, different systems of classification, and different database and repository management schemes. NSDL is intended to join these libraries using a common search engine called Emerge and a method for sharing resources called LOVE (Learning Object Virtual Exchange) [11]

2.6 SCORM

Sharable Content Object Reference Model (SCORM) was the Advanced Distributed Learning (ADL) Initiative's response to specifying learning content and its labeling, storage, and representation in distributed learning. SCORM was first proposed in 2000, with contributions from IMS Global Consortium, AICC (Aviation Industry CBT (Computer-based Training) Committee), Ariadne, and the IEEE LTSC (Learning Technology Standards Committee) [40]. SCORM specifies how to develop and deploy content objects that can be shared and contextualized to suit the needs of the learner [32]. Its objects can be anything ranging from an entire course to a lesson within a course, to a module in a course [9]. SCORM endeavors to produce learning content that is reusable, interoperable, accessible, durable and adaptable: [1]

- *Reusable*: the instructional content can be flexibly incorporated, (i.e. course materials in multiple instructions, applications and contexts);
- *Interoperable*: course materials developed in one location can be with one set of tools or platform can be used in another in another location with a different set of tools or platform;
- *Accessible*: instructional components can be located and accessed from one remote location and delivered in many other locations;
- *Durable*: technology changes can be withstood without costly reconfiguration, redesign or rerecording;
- *Adaptable*: instruction can be tailored to individual and organizational needs.

A central design characteristic of the SCORM specification is the division of responsibilities between the LMS and SCORM compliant content. The data model is defined for implementation on the LMS for the support of various forms of data storage as well as defining a standard communication mechanism between the LMS and the content. This makes content reuse with different LMSs possible without any re-work. Most of the major LMS vendors support SCORM, meaning that SCORM compliant content is fully deliverable on different LMSs without modification. There are a number of tools, e.g. Reload editor, that are readily available, for the preparation of SCORM compliant content. [22]

The latest specification, SCORM 2004 (also known as SCORM 1.3) [31] is made up of three parts: Content Aggregation Model (CAM), Run-Time Environment (RTE), and Sequencing and Navigation (SN).

CAM: This data model is used to describe any content object put together with information from Ariadne (metadata), IMS (content packaging) and IEEE's standard for metadata data model [40]. The SCO, typically (but not necessarily) a Web page, is the smallest grain size of a reusable learning object within SCORM. It can communicate with the LMS to perform data exchange or tracking [22]. Figure 7 shows how they would be typically arranged[40]. CAM is made up of 3 parts:

- **Content Model**: (defines the hierarchy of LOs (levels))
    - *Assets*: any other piece of data that can be delivered to a web client;
    - *Sharable Content Objects (SCOs)*: collection of one or more assets;
    - *Content organization*: Map (content structure) that can be used to combine learning objects into a cohesive unit of instruction (e.g. course chapter, module, etc)
- **Metadata**: describes the Content Model and provides an efficient mechanism for content search;
- **Content Packaging**: definition that allows the content model and structure to be packed into a standard exchangeable file (Figure 8) which allows LO exchange in a standard form across different platforms for presentation. According to IMS, a content package includes two parts:
    - XML document which describes the organization of a course object, and
    - a set of physical files that contain learning objects. [47]

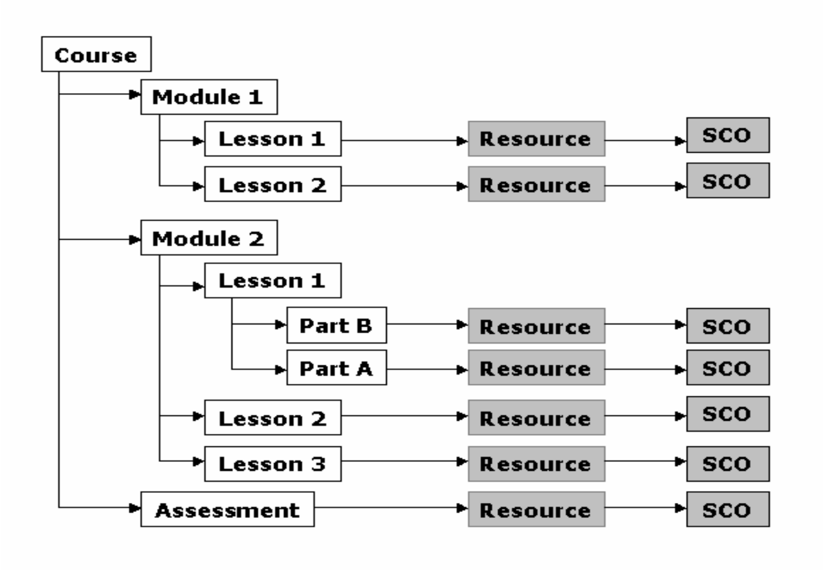

Figure 7: Typical arrangement of content

Usually, each physical content file is associated with an XML file, which has the metadata. The content package will also include a few control programs, which maintain the communication between a learning object (e.g., SCO) and the SCORM RTE [40].

RTE: Provides a standard communications protocol for launching an SCO and its communication with an LMS.

The SCORM RTE reads the XML file when the associated LO starts. The LO viewing procedure would be:

1. LMS starts the launch process
2. Control is passed to the LO (i.e. SCO)
3. SCO issues initialization followed by a series of invocations to the API (handled through client-side API adapter)
4. At the end, SCO calls the termination API function so that communication with the LMS is terminated.

While the LO is being presented, the associated multimedia data are retrieved and presented by the Web browser on the client [40].

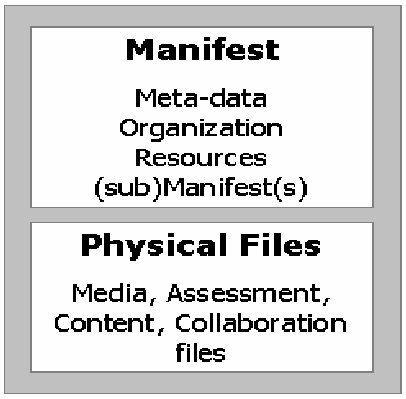

Figure 8: The SCORM Content Packaging (adapted from SCORM 2004 specification)

SN: It deals with the design of sophisticated instructional strategies [31]. Definitions found here are for tracking and controlling of interactions between users (i.e. students) and the LMS [40]. It is made up of two models: sequencing and navigation

- Sequencing Model: This behaviour model is used to describe how an LMS processes rules to sequence delivery of SCOs [31].
- Navigation Model: This is a behaviour description for a collection of user interactions and events. It is a model for the dynamic presentation of learning content based on learner needs [1].

2.6.1 Learning Management Systems (LMSs)

LMS refers to a suite of functionalities designed to deliver, track, report on and manage learning content, learner progress and learner interactions. The term can be applied to very simple course management systems, or highly complex enterprise-wide distributed environments [1]. A highly generalized model showing potential components or services of an LMS is shown in Figure 9.

Some of the LMSs available today

- *CLI Virtuoso* by the Cisco Learning Institute. It is designed to provide a complete, scalable, Web-based, e-learning platform with focus on collaboration, interactivity and personalized feedback for organization-wide e-learning.
- *In.Form@* ,by the DIDAGROUP, is the first Italian AICC and SCORM 1.2 LMS certified platform.

- *Digitalbrain plc* from Digitalbrain plc provides a powerful virtual learning platform, based on IMS content packaging and IMS metadata.
- *Learn eXact*, developed by Giunti Interactive Labs, is based on SCORM 1.3, supports authoring, packaging, and delivery of SCORM-compliant learning experiences on multidevices.[40]

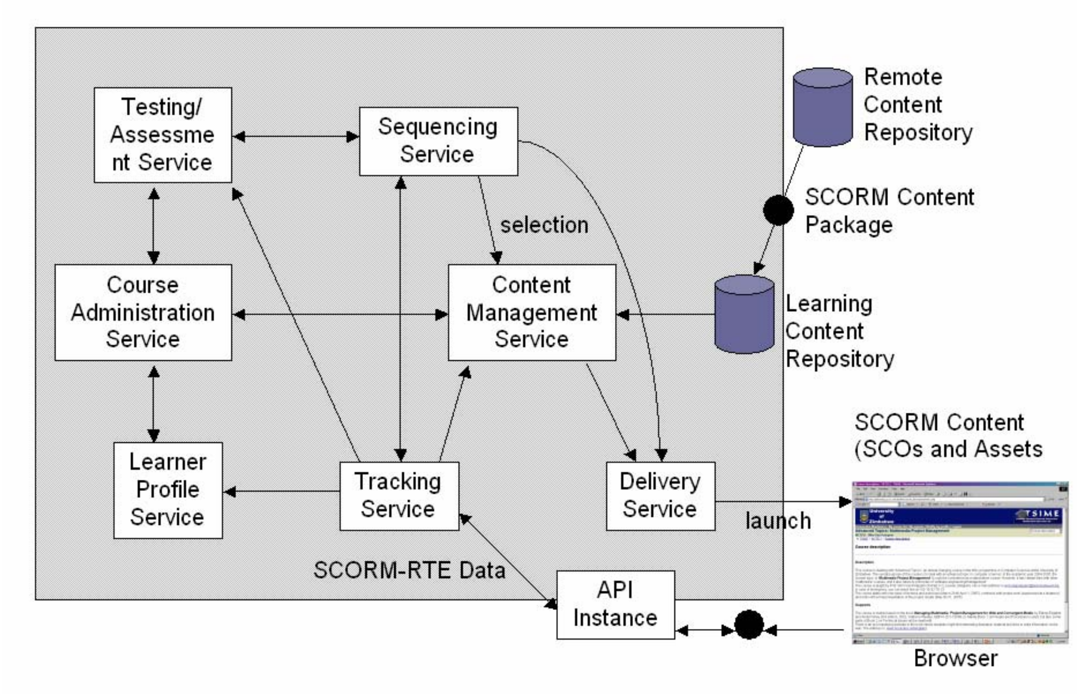
Testing/ Assessment Service
Sequencing Service
Remote Content Repository
SCORM Content Package
selection
Course Administration Service
Content Management Service
Learning Content Repository
SCORM Content (SCOs and Assets
Learner Profile Service
Tracking Service
Delivery Service
launch
SCORM-RTE Data
API Instance
Browser

Figure 9: Highly generalized model of LMS

With SCORM learning content is built from content objects that are relatively small and reusable instructional units. These objects, alone, do not have a particular context meaning that they do not determine by themselves how to sequence or navigate through an aggregation representing a unit of instruction. Having them do so would require content objects to contain information about other content objects within a content organization. This would limit their use to a specific context thereby inhibiting their reusability. Instead, the rules that control sequencing and navigation are defined within the aggregation and interpreted by the LMS. The LMS itself merely processes the externally defined rules and has no actual knowledge about the content's organization [1].

Course materials may contain pedagogical properties, which may not be found by ordinary search engines. One of the remaining challenges for the SCORM specification is the design of a reasonable set of metadata, which can be used to precisely and easily describe course materials [40].

2.6.2 CORDRA

Although SCORM provides the means for content to be discovered and accessed at a later time; it is silent about discovery and access. This is where CORDRA (Content Object Repository Discovery and Registration/Resolution Architecture) comes in. CORDRA is an open, standards-based model for how to design and implement software systems for the purposes of discovery, sharing and reuse of learning content through the establishment of interoperable federations of learning content repositories. It provides a model of how to create local networks and a global learning content infrastructure.[32] Implementations of the model will be open and flexible and will coexist and interoperate with existing systems [5].

The CORDRA core seeks to achieve a state in which published content can be widely available, persistent outside a course and discoverable. The overall CORDRA model is a collection of local content repositories with a master catalog (content registry), repository registry (participation repositories) and a system registry (system models and data), and some common services and applications.

The key CORDRA operations are:

- Register a content object in the content catalog for later search and retrieval
- Search the *content catalog* and return content objects, using their IDs and metadata
- Register a content repository in the *repository registry* by specifying descriptive data and rules
- Query the *repository registry* for the operational, policy and business rules [5]

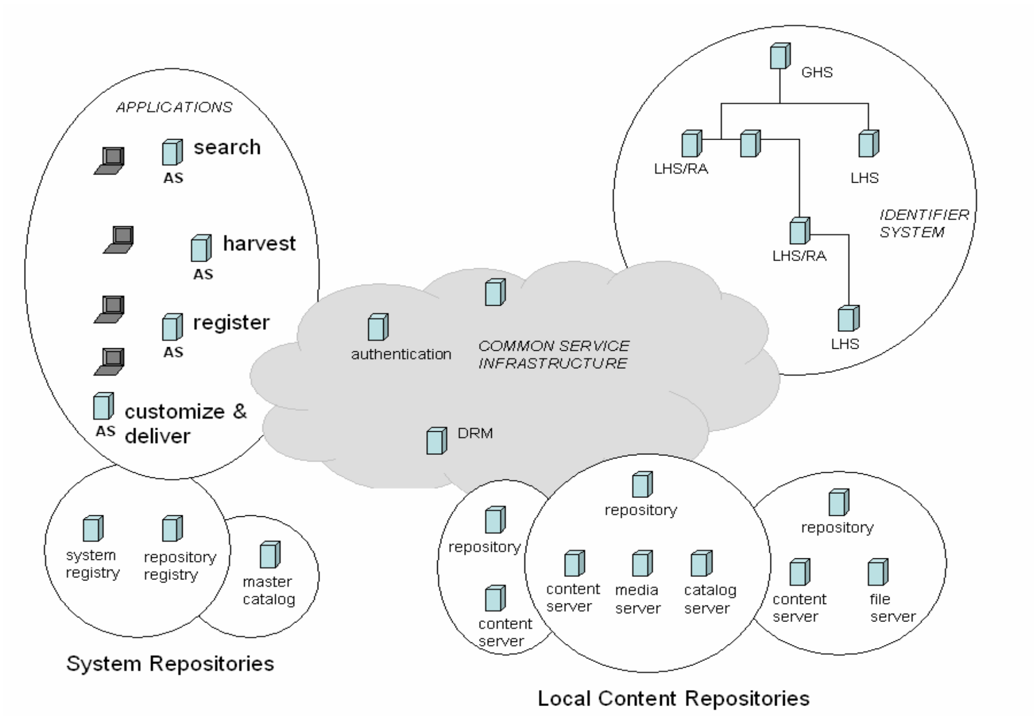

Figure 10: The CORDRA model

2.7 Classification

There are criteria by which objects can be classified: subject-based and genre-based classification.

2.7.1 Subject-based classification

Subject-based classification is any form of content classification that groups objects by the subjects they are about. Metadata properties or fields that directly describe what the objects are about by listing discrete subjects use a subject-based classification [17].

Creation of an effective subject index is not easy. Users need to have the right word to match the topic they are interested in for them to be able to successfully locate the appropriate documents. A problem arises (vocabulary problem) because different words can be used to refer to the same topic. A range of users will need to be accommodated as a result, as it is unlikely that every user will pick the same word to refer to a particular topic. There are several strategies that could be used to come up with key words. Some of these strategies are:

designer-based subject index, author defined keywords, card sort, automated keyword generation based on frequency, and optimized keyword. [2]

a. Designer-Based Subject Index: This method allows the subject matter expert to choose the subject categories. It assumes that the expert is familiar with the topic area. The usability of the index is likely to vary depending on the individual subject matter expert.

b. Author Defined Keywords: In this case the author provides a set of keywords or phrases for a document. The keywords can be consolidated into a master list and used as a basis of a subject–based index.

c. Card Sort: This is a strategy commonly used by human factors specialists for structuring information. This technique is employed, to create subject-based categorization, by giving 5-10 participants index cards. Each card contains an abstract and the participants are asked to put them many or few categories as they would like, giving each category a name. Each of the category names can be used as a topic in the subject-based index. A process which can take days

d. Automated Keyword Creation Based On Frequency: Words that occur with a high frequency are used for keywords, which are then used as the basis for topic headings. A problem with this technique is that the software often produces words that are used frequently but did not summarize the main points of the document

e. Optimized Keyword: This strategy requires 4-6 participants. Each of these participants is asked to provide keywords or key phrases for an abstract. The top keywords, for an abstract across participants, would then be considered the optimized keywords. People use a few words very frequently even thought their vocabulary is wide. Since this method requires different users to provide words for the abstract, it simulates different approaches to the categorization of individual abstracts.

Each of these strategies results in a different set of keywords or phrases for each document. The keywords or phrases can be used as the basis of a subject index. It has been found that the optimized keyword method produces the highest success rate, and card slot the lowest[2]. It could be argued that including more terms in the index could increase the success rate. As much as it is important to include enough keywords to accommodate many users, it is inefficient and not practical to provide a large selection for each individual document. An optimal subject-based index maximizes the hit rate while minimizing the number of unnecessary keywords. The researchers in [2] also found that using the authors' keywords produced the most hits as opposed to designer-based index created by the subject matter expert.[2]

2.7.2 Genre classification

Genre is a French word meaning type or kind [42]. Other definitions have it as " a term designating a specific kind of material distinguished by the style, technique, or format of its content" [61]. According to [34] genres are socially recognized regularities of form and purpose in documents. An example of where the concept of the genre is used is in music (gospel, country, jazz, soul, etc). One can think of the genre of a document as being tied to its purpose. If the purpose of a document is known, it could be useful for information retrieval. When a genre is put in the context of a community different from the one that uses it, that genre may not be recognized, and its purpose may be unknown [23].

Research has been carried out on Web access where it is believed, by some researches, that the recognition of the genre of a Web document can improve the quality of web searches. In their studies of people searching the Web, the authors in [34] observed that document genre was one of the clues in assessing document relevance, value, quality and usefulness. Also, the quality of the user interface can also affect the user's experience when accessing information. They suggest that the interface should enable users to

1. limit searches to specific genres
2. visualize the hierarchy of genres discovered in the search, and
3. provide user feedback on relevance of the specified genre[34]

Search engines in the ranking process can also use genres. Documents from one set of genres that are known to be more useful than those from other genres for a given search query can be ranked higher up list [23].

The process of assigning genre to documents can be done automatically by running documents through a program that checks for certain characteristics that it uses to determine the classification. A document may exhibit characteristics that qualify it to belong to more than one genre. Work done so far is on automatic categorization of text documents on the web [39].

In the traditional approach to the automation of text classification the system learns the statistical distribution of words in order to differentiate between the relevant categories. This works well when the subject matter between varies significantly between the categories. The technique, however, does not do an equally good job of determining genres of text, where more or less the same words can appear, but instead language styles vary [39]. This automation is more difficult as compared to subject automation for subject classification

Genres can be identified in a top-down or bottom-up fashion. In the top-down approach the names of the genres are gathered and attempts are made to classify documents into these genres. The problem with this approach is that genres created by one community may not be applicable or understandable to another community. With the bottom-up approach information is gathered on the characteristics that make up a genre. It is these characteristics that are then examined to establish how they can be used to in order to define genres. The advantage of this approach is that new emerging and dynamic genres can be recognized. Unlike top-down that has been the method of choice for some time now, the bottom up approach has only been carried out with small-scale studies.

The difference between LO genre and web (text) genre is the nature of the "thing" that is being classified. With web classification we are dealing with the content of the page itself, whereas with LOs we are operating on a metadata level, hence genre (or purpose) can be viewed as being influenced by the person who authored the object and possible users who have studied and successfully applied the object in other settings. Therefore an object can be having more than one purpose.

2.8 Conclusion

LOs have come a long way in terms of their development. The use of metadata and metadata standards has opened the door for LO sharing and reuse. Research is still being carried out to find out ways in which the actual search of these repositories can become a more fulfilling experience for the user. People ask for what they expect they can get that will most closely match what they really want, and thus their requests are often presented in a compromised form. Thus we see that topic alone is not enough to define an information problem because different users require different solutions to seemingly similar information problems. In [7] it is suggested that incorporating non-topical characteristics of the document that signal their purpose can enhance document representations.

Chapter 3

Methodology

3.1 Introduction

Computerized information-access systems know what documents say, but they do not know what those documents mean or the purpose for which they might be of use [7]. Most search tools were based on an electronic form that enabled end users to compose Boolean combinations of each search criteria. This approach had serious usability problems and failed to take advantage of the structured nature of LOM. What was needed was more research on how to provide flexible, efficient and effective access to a large repository of LOs so that the end user could quickly zoom in on those LOs that are relevant [14]. Merlot, ARIADNE and other such repositories have made strides at achieving this goal with substantial success.

This research seeks to find ways in which users could be presented with results that gives them the information that they asked for, that is to say the right object(s), ordered appropriately. Search results can be broad and the order in which they appear may put the most relevant information right at the bottom of the list, which can be very inconvenient when the results spans several pages. We want to look at LOs and their metadata and find ways in which the search space can be drastically reduced and a richer result set obtained at the end of the day.

3.2 Genre-based search - LO classification

By identifying the genre of a document it is possible to get information about the documents purpose and how it fits in with the user's situation. The topic alone is not enough. Search results containing genre information would be easier to understand [7]. A set of documents belonging to the same class (because they share a common topic) often serve different purposes hence they fall into different genres. Classification by genre will give a set of results that are different from those obtained from a subject-based classification [24]. In as much as genre gives us the advantage of better, more useful results, subject cannot be ignored completely. There are objects that will have the same keyword(s) and purpose but belonging to different subjects. For example, searching on the keyword “2D”would yield results containing objects from Computer Graphics and Programming. If the repository contains a large number of objects that correspond to this keyword and are for the correct purpose (LRT), the user will have to sift through all of these to get what they want.

A look at IEEE’s LOM metadata standard reveals that element 5.2 (Educational element Learning Resource Type (LRT)) would provide the genre information that is required. Element 1.5 (General element Keyword) can be used for the description of the LO. As most repositories have the actual LOs stored in separate locations from the metadata there would be a need for a link to these objects, and the Uniform Resource Identifier (URI) provides this link.

By ordering these metadata elements, the subject and the URI in a hierarchy (not the same as the LOM hierarchy) the LOs can be put into “neighborhoods”. These neighborhoods will have relevant objects included in the search space and hence present the user with the right information that suits their purpose. This approach helps to reduce the amount of extra work a user has to do to isolate the objects that are relevant to them. The search can also offer the user other objects that might be of use or of interest to them. Users can be offered additional objects gathered using previous search statistics. This can be achieved by compiling results based on what their colleagues, who have carried out the same, or a similar search, have also retrieved.

It is important to note that some LOs can belong to more than one genre and at the same time a LO can have more than one keyword. Figure 11 shows the elements arranged hierarchically. Each subject forms the root of a sub-hierarchy (level 1) that contains LRTs (level 2), keywords (level 3) and URIs (level 4). What this means is that each object introduced to the repository must have at least one subject it belongs to. These sub-hierarchies all have a common root (level 0). This hierarchy is in the form of a directed graph.

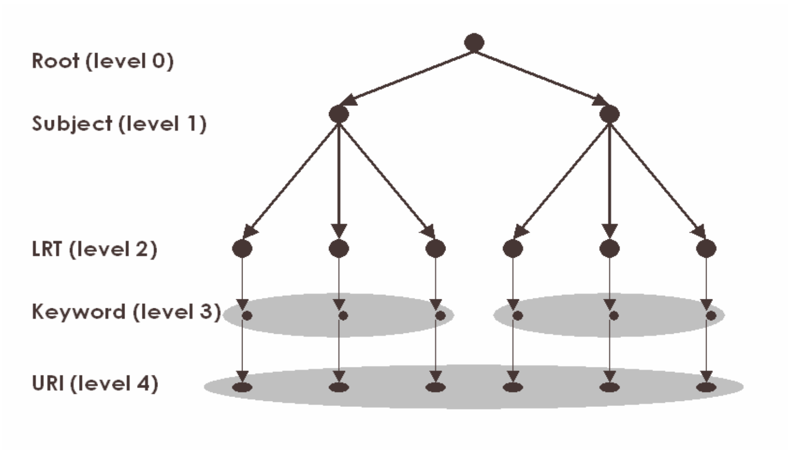

Figure 11: The Metadata "Hierarchy"

3.3 Graph analysis

The "flavor" of the graph as represented in Figure 11 is:

1. Directed: Each edge arc is an ordered pair of vertices.
2. Acyclic: There are no paths which start and end on the same node (i.e. no cycles)

Therefore the graph is a Directed Acyclic Graph (DAG).

The DAG is made up of subject sub-hierarchies that have a common root. There are as many subject-trees as there are subjects, and each tree contains a set of third level nodes (Learning Resource Types (LRTs)) that are duplicated across the subjects, fourth level nodes (Keyword) and the fifth level (URI).

A keyword can be associated with as few as one to as many as all the LRTs for the particular subject it belongs to. A keyword is not unique to a subject, but can exist in many subjects. Each subject has its own set of keywords; it is not a common pool. The Uniform Resource Identifiers (URIs) are found in an identifier pool that is common to all subject-trees. Nodes in the keyword level point to these URIs. Having the URIs being sourced from a single pool by the entire graph encourages the sharing and reusability of LOs.

Let ***s*** be the number of subjects, ***l*** the number of LRTs, ***k*** the number of keywords, and ***u*** the number of URIs.

Below are the degrees for the tree as in Figure 11 to Figure 13 without any cross-connections in levels 2, 3 and 4. The degree is the number of edges from a node in a tree

<u>Degree at level 0</u>:

The number of subjects represented gives the degree at this level therefore the order is $\boldsymbol{s}$

<u>Degree at level 1</u>:

The number of LRTs represented gives the degree at this level therefore the degree is $\boldsymbol{p}$.

The total number of edges at this level is $\boldsymbol{sp}$ (i.e. the number of LRTs specified in the vocabulary for the element multiplied by the number of subjects).

<u>Degree at level 2</u>:

As the LRT is made up of a predetermined list of possible values (vocabulary) it is guaranteed that there are a constant number of LRT nodes for each subject. The degree at this level is $\boldsymbol{k}$. The number of edges to expect for one subject ($\boldsymbol{s}_i$) is $\boldsymbol{pk}_i$. For the entire level, which takes into account all of the subjects with their associated keywords, the number of edges becomes:

$$\boldsymbol{pk}_1 + \boldsymbol{pk}_2 + \boldsymbol{pk}_3 + \ldots + \boldsymbol{pk}_n$$

This comes down to $\boldsymbol{pn}$ where $\boldsymbol{n}$ is the number of keywords

<u>Degree at level 3</u>:

Each keyword $\boldsymbol{k_i}$ belonging to subject $\boldsymbol{s_x}$ has edges $\boldsymbol{u}$ URIs in the common pool. Level 3 is a common pool, meaning that there are no URIs particular to a subject. The degree is $\boldsymbol{uk}_i$ for a keyword $\boldsymbol{k}$ belongs to a subject $\boldsymbol{s}_i$. The number of edges for a subject would be $\boldsymbol{uk}_1 + \boldsymbol{uk}_2 + \ldots + \boldsymbol{uk}_x$ where x is the number of keywords for that subject. The number of edges for the entire level would then be:

$$\boldsymbol{u} \sum_{1}^{s} (\boldsymbol{k}_i) + \boldsymbol{u} \sum_{1}^{s} (\boldsymbol{k}_{i+1}) + \ldots + \boldsymbol{u} \sum_{1}^{s} (\boldsymbol{k}_x)$$

<u>Degree at level 4</u>

The URIs have no nodes leading out of them so the degree is 0 (zero).

From Figure 12 it can be seen that the addition of a node at either level 3 or 4 means that the number of edges from an LRT increases by one.

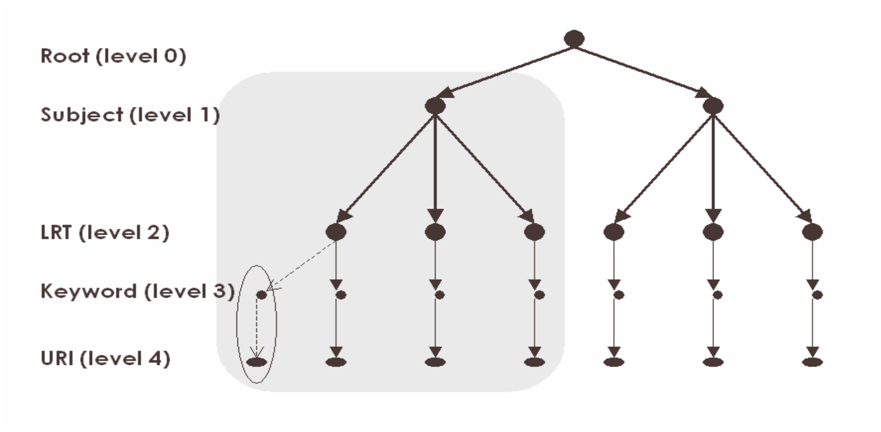

Figure 12: Adding a node to levels 3 and 4

The addition of $\boldsymbol{x}$ subjects (Figure 13) increases the degree of the root by the $\boldsymbol{x}$ and the degree becomes $\boldsymbol{s+x}$. The number of edges at level 1, in total become $\boldsymbol{sp+p}$.

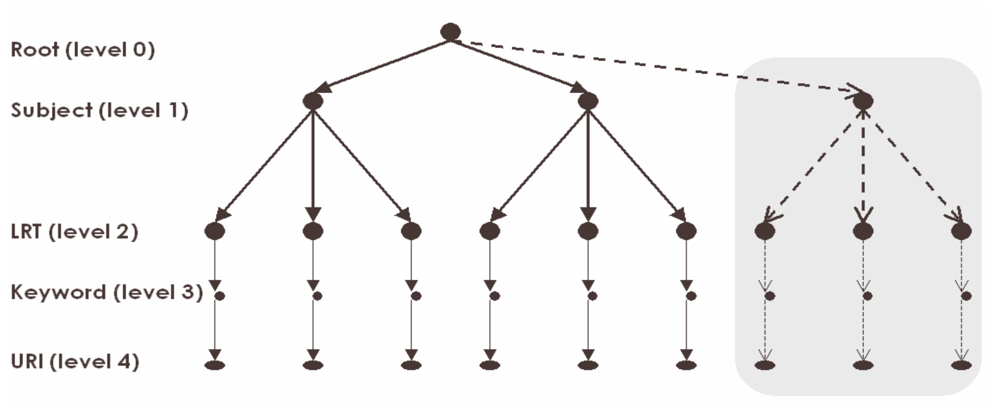

Figure 13: Adding an entire subject

By allowing cross connection as shown in Figure 14 the degrees of a node at levels 2 and 3 increase by the number of added edges (see "Degree at level 2" and " Degree at level 3" above).

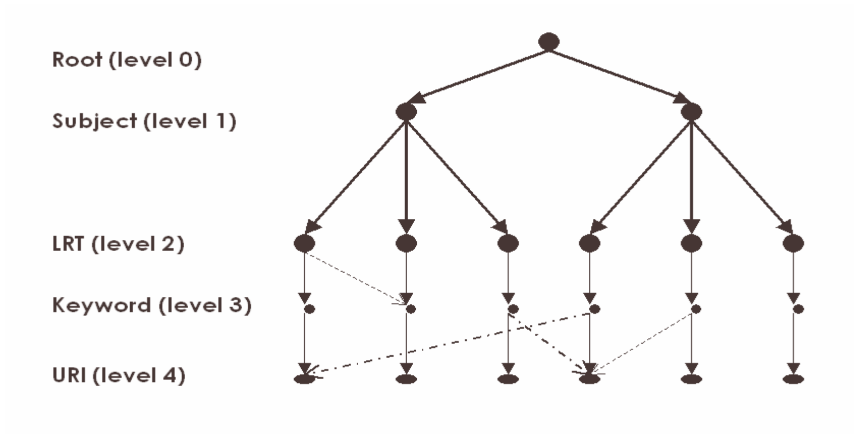

Figure 14: Cross-connection of nodes within a subject and across subjects

In Figure 14 it is possible to get from the root to a URI using more than one path. As the URIs belong to a common pool a keyword in one subject can point to a URI pointed to by another keyword from a different subject.

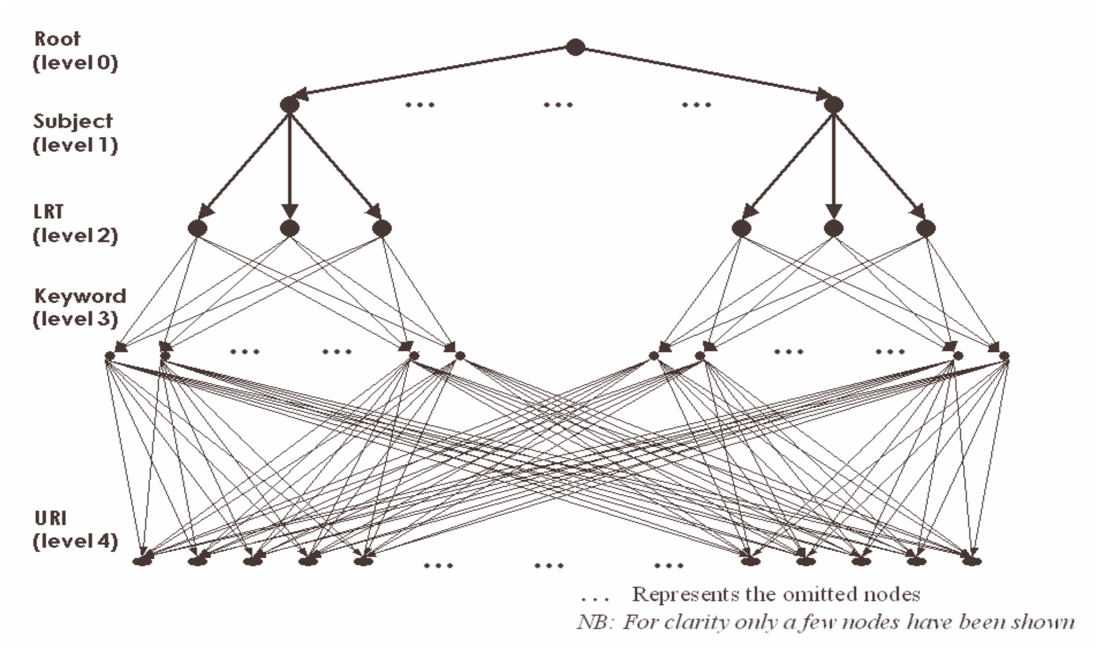
Root (level 0)
Subject (level 1)
LRT (level 2)
Keyword (level 3)
URI (level 4)
... Represents the omitted nodes
NB: For clarity only a few nodes have been shown

Figure 15: Worst case DAG

The graph in Figure 15 presents the worst-case scenario that a repository can approach highly unlikely in the real world. In such a situation the genre classification would become meaning less as one can get to any URI using any purpose through any subject.

3.3.1 Graph representation

The pictorial representation is useful especially for the examination of the graph structure as a whole, however it is impractical to store it as it is on a computer. There are several possible ways of representing graphs. These are:

- Store the set of vertices and the list of edges, used especially when the graph is "sparse" (a lot of edges but not very many edges).
- Take each vertex in turn and list those vertices, which are adjacent to it. By joining each vertex to its neighbor, the graph can be reconstructed easily
- Draw up a table indicating which pairs of vertices are adjacent or a table indicating which vertices is incident with each edge. [10]

The last method is particularly useful in a number of practical applications. It uses a matrix (incident or adjacency) to represent the graph.

<u>Incidence Matrix</u>: This type of matrix involves the incidence of vertices and edges.

> *Let* G *be a graph without loops,* n *vertices labeled* v1*,* v2*,* v3*, ...,* vn *and* m *edges labeled* e1*,* e2*,* e3*, ...,* em*. The incidence matrix* A(G) *is the* n x m *matrix in which the entry in row* i *and column* j *is 1 if the vertex* vi *is incident with* ej*, and 0 otherwise.*

The letter A is used to denote the incidence matrix rather than the adjacency matrix [10].

Selecting the right data structure to represent graphs can have an enormous impact on the performance of an algorithm. Your two basic choices are adjacency matrices and adjacency lists [43]

<u>Adjacency Matrix</u>: This involves the adjacency of the vertices.

> *Let* G *be a graph without loops, with* n *vertices labeled* 1*,* 2*,* 3*, ...,*n*. the adjacency matrix* M(G) *is the* n x n *matrix in which the entry in row* i *and column* j *is the number of edges joining the vertices* i *and* j.

Every entry on the main (top-left to bottom-right) diagonal is 0 since there are no loops and the graph is symmetrical about this diagonal. [10]

Adjacency matrices are the simplest ways to represent graphs, however, they use O(n2) spce no matter how many edges are in the graph, which can prove fatal for large graphs. [43]

Adjacency list:

It consists of an array of n-element pointers, where each element *i* points to a linked list of edges incident on vertex *i*. To find if an edge (*i*,*j*) is in the graph, the *i*th list is searched for j. this takes O(*di*) where di is the degree of the ith vertex. For a complete, or almost complete graph, *di* = Θ(*n*). Testing for the existence of an edge can be expensive relative to adjacency matrices. *di* can, however, be much less than n in a sparse graph, which is useually the case in real life graphs. The key is in processing the edges systematically (breadth- or depth- first search) [43]. Implementation with adjacency list would be the better option though a major drawback is in having to deal with linked lists structures.

Two variations of this structure are:

Adjacency Lists in List: sparse graphs can be represented more efficiently using linked lists to store th neighbors adjacent to each vertex.

Adjacency Lists inMatrices: Adjacency can be embedded in matrices thus getting rid of pointers. A list can be represented in an array. The elements are counted and packed into the first elements of the array. Elements can now be visited successively from the first to the last by incrementing an index loop.[59]

	Adjacency matrix	Adjacency list
Faster to test if (*x*,*y*) is in graph?	Yes	-
Faster to find a vertex's degree?	-	Yes
Less memory on small graphs	-	Yes (*(m+n)* vs. *(n^2)*)
Less memory on big graphs?	Yes (by small margin)	-
Edge insertion or deletion?	Yes (*O(1)* vs *O(d)*)	-
Faster graph traversal	-	Yes (*Θ(m+n)* vs. *Θ(n^2)*)
Better for most problems?	-	Yes

Table 1:Adjacency Matrix vs. Adjacency List

Table 3 shows some of the results used to come up with the graphs in Figure 17. These results were obtained by counting the number of edges that would have to be traversed in a search of DAGs like those shown in Figure 16. The DAGs are for a repository containing LOs for 3 subjects and the LRT vocabulary having 3 values. Each LRT is connected to 1, 6, 10 or 20 keywords. These are arbitrarily selected values. The search method applied is depth-first, going from left to right. It does not stop until all relevant level 3 nodes (keywords) have been visited. At each successful keyword, all the associated URIs are "collected" and put out to the user in the list of results.

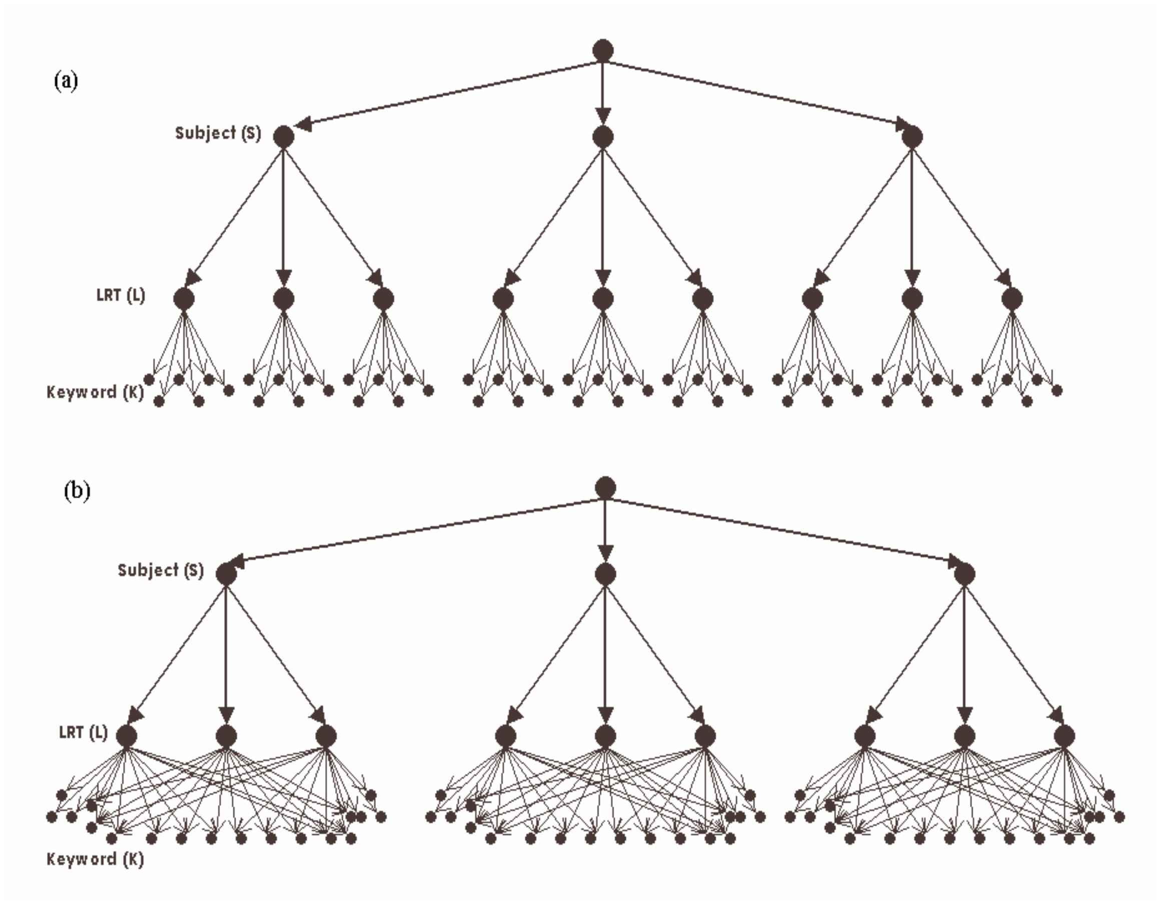

Figure 16: (a) an example of the first case scenario with no interconnections within a subject; (b) an example of the second case scenario with interconnections.

Figure 16(a) show a case in which each LRT has its own set of keywords associated with it, and there are no "inter-connections" between LRTs within a subject. The second case is a half way case in which there are an average number of inter-connections (i.e. an LRT is connected to only half the keywords of its fellow LRTs (Figure 16(b))). The last case could be seen as a sort of worst-case scenario in which each LRT is also connected to all the LRTs of its fellow LRTs up to level 3 (Figure 15).

SLK:	Subject, Learning Resource Type and Keyword
LK:	Learning Resource Type and Keyword
SL:	Subject and Learning Resource Type
SK:	Subject and Keyword
L:	Learning Resource Type
S:	Subject
K:	Keyword

Table 2: Element combinations

The symbols in Table 2 show the conditions used to test how the number of edges ($|e|$) behaves as the number of keywords ($|k|$) increases. *e(SLK)* stands for the number of edges that are attained for a certain number of keywords where the user has inputted *S* (Subject), *L* (LRT) and (*K*eyword). *e(SK)* would be results for when the input was the subject and keyword only. There is an eighth option of not specifying anything in the fields, which gives a list of all the URIs that are in the repository.

(a) no crossing

k	1	6	10	20
e (SLK)	3	8	12	22
e (SL)	3	8	12	22
e (SK)	7	22	34	65
e (S)	7	22	34	65
e (LK)	9	24	36	69
e (L)	9	24	36	69
e (K)	21	66	102	195

(b) "half" crossing

k	1	6	10	20
e (SLK)	4	14	22	43
e (SL)	4	14	22	43
e (SK)	10	40	64	125
e (S)	10	40	64	125
e (LK)	12	42	66	129
e (L)	12	42	66	129
e (K)	30	120	192	375

(c) "full" crossing

k	1	6	10	20
e (SLK)	5	20	32	63
e (SL)	5	20	32	63
e (SK)	13	58	94	185
e (S)	13	58	94	185
e (LK)	15	60	96	189
e (L)	15	60	96	189
e (K)	39	174	282	555

Table 3: Tree traversal results

Table 3 shows the results of traversing DAGs having the structure shown in the examples DAGs in Figure 16. From the graphs in Figure 17, it is apparent that knowing the subject and LRT will really help in narrowing down the search and reducing the total number of edges traversed. The lines representing the combinations *(SL)* and *(SLK)* are the most gentle in all 3 cases, and corresponding rows in the results table carry exactly the same information. This means that it is the most effective combination. In the presence or absence of a keyword the resultant number of edges traversed does not change. This is because the search has to gather all URIs that represent all the LOs that would satisfy the users query. On the other hand, giving the keyword only is the least favourable condition as it leads to the traversal of all the nodes in the graph.

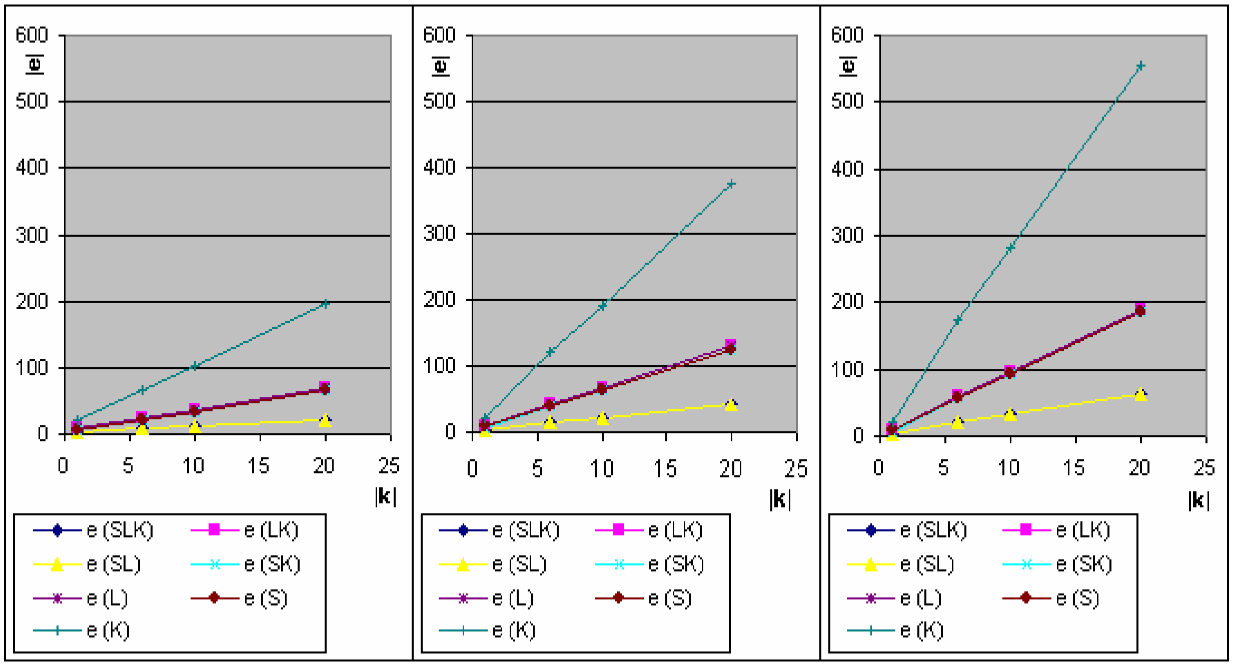

Figure 17: Graphical representation of results in Table 3

3.4 Searching the graph

Having established the type and structure of graph that the LOR can be in, there is need for an effective mechanism for searching through it.

3.4.1 Search Mechanisms

Applications go through a search space in order to find one or more solutions that are optimal or satisfactory. A search tree is made up of alternatives that are examined in a search process and, unless the search is exhaustive, the search tree is a sub-set of the search space. The term “search tree” comes from the categorization of solutions such that similar solutions are together in the same branch of the tree. [21] Three search types that can be considered when designing the structure of the tree are:

- Depth first search (DFS)
- Breadth first search (BFS)
- Best first search (BeFS)

DFS and BFS are two alternatives strategies for systematic search that are examples of *blind search*.

These search strategies are powerful methods for graph exploration. They all start at a node v of a directed graph and visit all nodes that can be reached form v.

Depth first search: All the edges out of the most recently visited node. It proceeds down a tree as far as possible in some predefined order (e.g. left to right), until it comes to a “dead end” reaches the goal. If a dead end is reached then the system will back up and try the next leftmost branch, i.e. it backtracks. (Figure 18) [13]

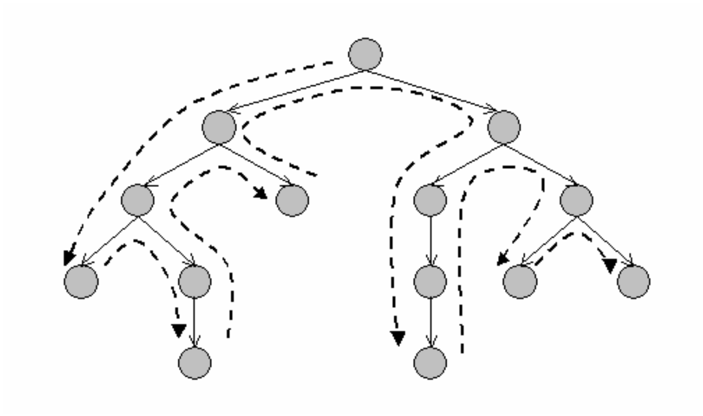

Figure 18: Depth first search

The basic algorithm is implemented using a stack and is initialized with the start node and terminates when the goal is found. When a state is expanded, each resulting state is pushed onto the stack. When a node fails the test he search resumes at the previous node, popping the failed node form the stack.

DFS algorithm

1. Let Q = queue containing start state

2. If head of Q is a goal state STOP
3. Generate successors of head and add them at the head of Q.
   Remove head
4. Go to 2 [51]

This search technique cannot recover from bad choices, which may lead down a large, possibly infinite, number of non-goal states. The search space could contain cycles that may lead to its repeated traversal forever. [41]

Breadth First Search: In this search technique the solutions are sought along all the nodes on the same level before considering those on the next level (Figure 19). The process is repeated at each level until a solution is found. It does not use heuristics and gives the shortest path solution. Each node is a generalization of the nodes on its sub-branches therefore if a node fails; all subcategories are assumed to fail as well and are eliminated from the search. Extremes are examined last. This is typically implemented in a queue.[50]

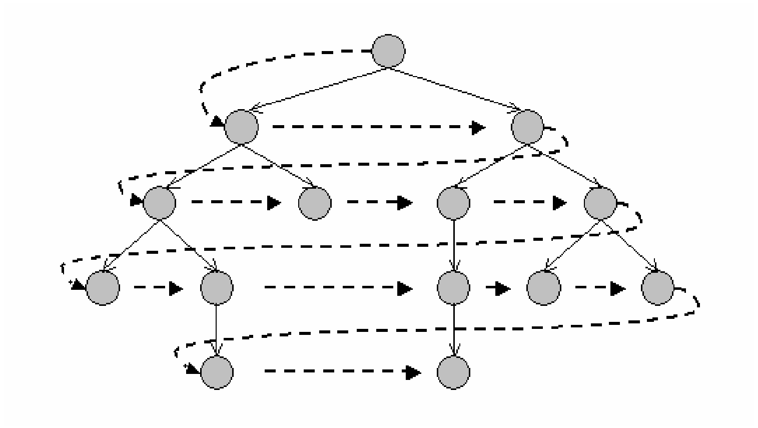
Figure 19: Breadth first search

BFS algorithm

1. Let Q = queue containing start state
2. If head of Q is a goal state STOP
3. Generate successors of head and add them at the tail of Q.
   Remove head
4. Go to 2 [51]

Best first search: A search can be made more efficient either by eliminating unfeasible categories (“pruning the tree”) or by ensuring that the most likely alternatives are tested before less likely ones. Heuristics can be applied to search process.

> *“A heuristic is a rule of thumb, strategy, trick, simplification, or any other kind of device which drastically limits search for solutions in large search spaces. Heuristics do not guarantee optimal solution; in fact they do not guarantee any solution at all; all that can be said for a useful heuristic is that it offers solutions which are good enough most of the time.”* [4]

BeFS uses heuristics to guide the search towards the solution nodes of the problem space. It is a combination of depth first and breadth first searches. BeFS allows for switching between paths thus gaining the benefit of both approaches. At each step the most promising node is chosen. If one of the nodes chosen generates nodes that are less promising, it is possible to choose another at the same level (switching from depth first to breadth first [44]

BeFS algorithm

(OPEN – list of nodes to be searched. CLOSED – list of nodes that have been searched)

```
1. OPEN = [start]; CLOSED = [];
2. If OPEN = [], exit with FAIL;
3. Choose best node form OPEN, call it X;
4. If X is goal node, exit with a traced solution path;
5. Move X from OPEN to CLOSED;
6. Generate successors of X (record a their parent)
     For each successor S of X
     a. Evaluate f(S);
     b. If S is new (i.e. not on OPEN, CLOSED), add it to OPEN
     c. If S is not new, compare f(S) on this path to the previous one.
        If previous us better or same, keep previous and discard S
        Otherwise replace previous by S.
                if S is in CLOSED, move it back to OPEN.
7. Go to 2. [51]
```

This method might not find a solution. It is guided to the part of the search space that seems promising. While DFS and BFS are exhaustive, these two are guaranteed to find solutions, if one exists. BeFS might never reach one. [13]

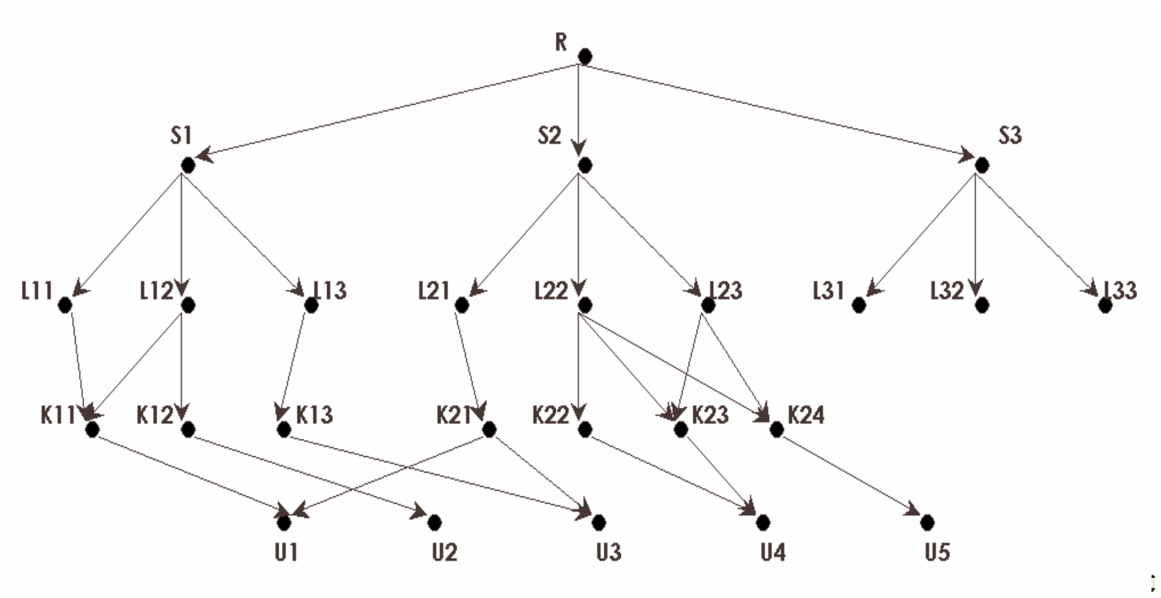

Figure 20: Example DAG

The DAG modeled in this paper uses input form the user to make “decisions” on the path to take as it proceeds down the tree. BeFS uses information it would have gotten form the user to decide which node to go to next and because of this property becomes search technique of choice. Figure 20 gives an example of a repository that has 3 subjects, and a LRT vocabulary with 3 possible values. There are only 5 LOs referenced.

The graph in Figure 20 can be translated into a table as shown in Table 4. Each node in Table 4, appearing in the leftmost column (Node) has, or can have, a degree given it, hence the nodes listed under “Adjacent nodes” are the ones that are adjacent to it. The nodes representing the URIs do not appear in the “Node” column because they have no degree associated with them hence they will not point to anything.

The adjacency table in Table 4 can be implemented as shown in Figure 21. The nodes are kept in arrays, with levels 1, 2 and 3 each having an array for the nodes in the level. Each node points to a list (linked list) the addresses of the nodes that are adjacent to it that are in the next level array. L31, L32 and L33 do not have nodes incident on them yet, but they are incident to S3.

Node	***Adjacent nodes***
R	S1, S2, S3
S1	L11, L12, L13
S2	L21, L22, L23
S3	L31, L32, L33
L11	K11
L12	K11, K12
L13	K13
L21	K21
L22	K22, K23, K24
L23	K23, K24
L31	-
L32	-
L33	-
K11	U1
K12	U2
K13	U3
K21	U1
K22	U4
K23	U4
K24	U5

Table 4: Adjacency table for example in Figure 20

This choice of representation is in a bid to allow for easier addition of nodes to the graph at different levels as well as for new adjacencies to be recorded. It would hopefully when deleting a node, which is unlikely possibility.

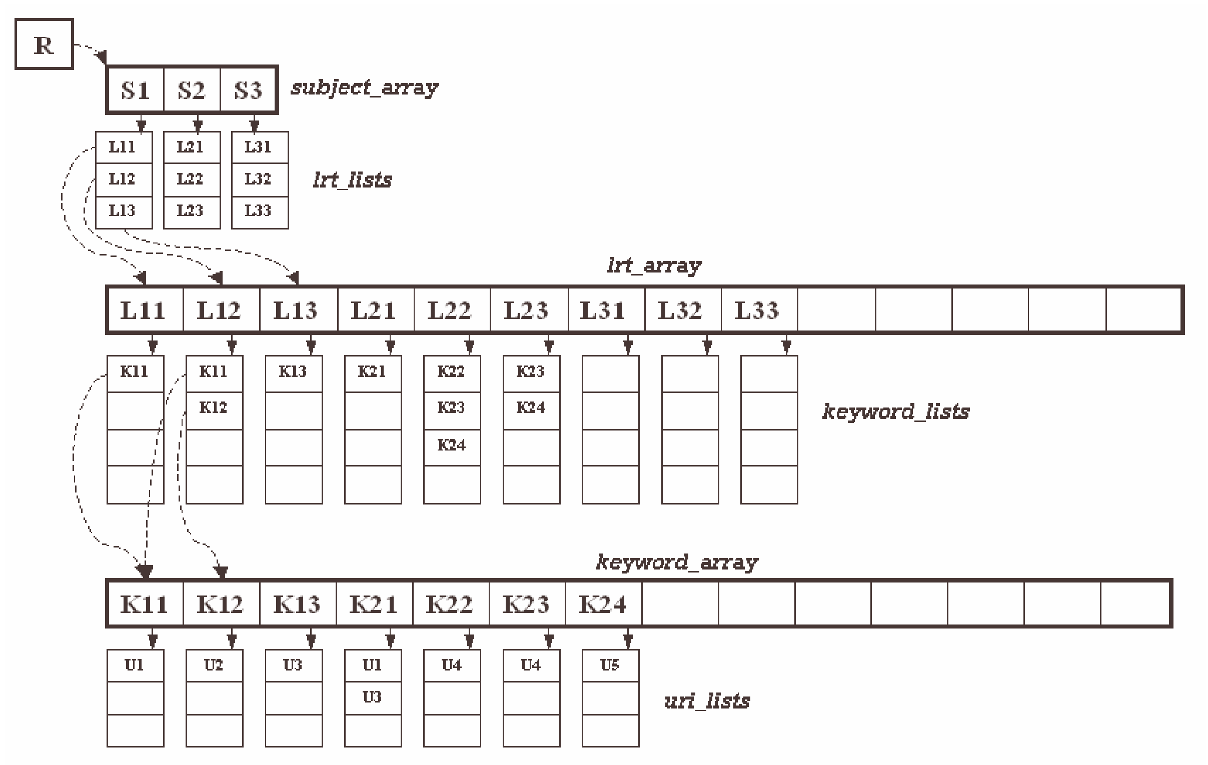

Figure 21: Adjacency lists and their respective pointers

The user's input of subject, LRT and or keywords (Table 2) are used isolate the appropriate lists and hence establish is such a search can be fulfilled and to actually go ahead and bring back results. For instance if the input is (S1, L12 and K12), the list pointed to by R is first searched for S1. If S1 is found then it's associated list is searched for the node L12. Upon confirmation that L12 is adjacent, L12's list is searched for K12. When this succeeds, the contents (URIs) in the list corresponding to K12 are then copied to a separate result list for presentation to the user. If a node does not exist in the adjacency list that has been searched, a "failure" report is sent back to the user. If no search elements were inputted a similar failure report would be sent requesting that some search criteria be given.

If input were made of the other 6 possible combinations, then for the level that is "missing" a search item, all elements in the list where it would be expected to be found (the neighborhood) would then be placed on a stack and an extensive search of all adjacencies done. Given the (SL) combination of (S2, L22), for example, when the search gets to L22, the elements of the attached list can be pushed onto a stack and the corresponding keyword lists would be searched and a URI list compiled.

3.4.2 The search algorithm

The purpose of the algorithm below is to find the right subject tree. From here the correct purpose is selected and then the related keywords are compared with the search keyword inputted. When the matching keyword is/are identified the respective URIs are then retrieved and presented to the user, in an appropriate rank order.

The search algorithm has been structured into cases based on Table 2. Each search query that is invoked comes will contain search information in any of these combinations. The arrays that are specified and their corresponding list (Figure 21) are: subject_array (lrt_list), lrt_array (keyword_list), and keyword_array (uri_list).

***Algorithm 1*:**

Input: subject S, learning resource type L, keyword K.
Output: result_list (containing the URIs of the appropriate LOs)

```
    CASE 1: SLK
    CASE 2: SL
    CASE 3: SK
    CASE 4: LK
    CASE 5: S
    CASE 6: L
    CASE 7: K
    CASE 8: none given
    boolean match = FALSE;

1)  If CASE 1
    a)  Search subject_array for S
    b)  Get the lrt_list and search for L
    c)  Go to the L in lrt_array and get the keyword_list
```

```
        i) Push keyword_list onto stack
        ii) For each keyword
            (1) keyword_match (K)
            (2) If match = = TRUE, copy uri_list to result_list and pop keyword,
            (3) Else pop keyword

2) If CASE 2
    a) Search subject_array for S
    b) Get the lrt_list and search for L
    c) Get keyword_list for L
        i) For each keyword copy uri_list to result list

3) If CASE 3
    a) Search subject_array for S
    b) Push lrt_list onto stack.
    c) Go to the L in lrt_array and get the keyword_list
        i) Push keyword_list onto stack
        ii) For each keyword
            (1) keyword_match (K)
            (2) If match = = TRUE, copy uri_list to result_list and pop keyword,
            (3) Else pop keyword

4) If CASE 4
    a) Push subject_array onto stack
    b) For each subject
    c) Get the lrt_list and search for L
    d) Go to the L in lrt_array and get the keyword_list
        i) Push keyword_list onto stack
        ii) For each keyword
            (1) keyword_match (K)
            (2) If match = = TRUE, copy uri_list to result_list and pop keyword,
            (3) Else pop keyword
        iii) Pop subject
```

```
5) If CASE 5
    a) Search subject_array for S
    b) Push lrt_list onto stack
    c) For each lrt push keyword list onto stack
        i) For each keyword, copy uri_list to result list and pop keyword

6) If CASE 6
    a) Push subject_array onto stack
    b) For each subject
        i) Search lrt_list for L
            (1) Push keyword_list onto stack
            (2) For each keyword copy uri_list into result_list and pop
        ii) Pop subject

7) If CASE 7
    a) Push subject_array onto stack
    b) For each subject
        i) Push lrt_list onto stack
        ii) For each lrt
            (1) Push keyword_list onto stack
                (a) keyword_match (K)
                (b) If match = = true, copy uri_list to result_list and pop keyword,
                (c) Else pop keyword
        iii) Pop lrt
    c) Pop subject

8) If CASE 8:
    a) Push subject_array onto stack
    b) For each subject
        i) Push lrt_list onto stack
        ii) For each lrt
            (1) Push keyword_list onto stack
        (2) For each keyword
            (a) Copy uri_list to result_list
            (b) Pop keyword
        (3) Pop lrt
    iii) Pop subject

9) keyword_match (K)          /* string/pattern matching function*/
    a) for i<length.keyword_list;
        i) if K = = keyword_list[i]
            (1) match = TRUE
        ii) else
            (1) match = FALSE
        iii) i = i + 1;

10) If result list is EMPTY
            Print "No results found"
    Else
            Print out result list
```

3.4.3 Time complexity

The time complexity for the algorithm is taken from the case that gives a worst scenario. Case 8 in which the user does not specify anything and the system returns everything in the repository gives this.

***CASE 8**:*

```
a) Push subject_array onto stack
b) For each subject
    i) Push lrt_list onto stack
    ii) For each lrt
        (1) Push keyword_list onto stack
        (2) For each keyword
```

```
                (a) Copy uri_list to result_list
                (b) Pop keyword
            (3) Pop lrt
        iii) Pop subject
```

The case contains 3 nested for loops; one for the subject, one for the LRT and one for the keyword. Let ***s*** be the number of subjects, ***l*** the number or LRTs and k the number of keywords. The subject for loop is executed ***s*** times hence the LRT for loop is executed ***sl*** times. The number of times a uri_list is written to the result_list is:

$k_1 + k_2 + k_3 + \ldots + k_n$ where ***n*** is the number of keywords in total

The number of times it writes the uri_lists to the result list is ***n***

The complexity of this algorithm is given by

$s * l * n$

So the order of this algorithm is **O(sln)**

3.4.4 Insertion of a URI

For the repository to grow, LOs need to be added to the repository which means that URI entries need to be made into the appropriate uri_lists. The algorithm below is for this insertion and uses the metadata information for LRT and keyword, together with subject information that can be gotten from the depositor or derived form other metadata elements in the standard.

Input: subject S, learning resource type L and keyword K

```
1. if S is in subject_array
        a. get the lrt_list
        b. go to x
2. else
        a. add S to subject_array
        b. add LRTs to lrt_array and create the lrt_list
        c. go to x
3. find L in the lrt_list
```

```
    a. get keyword_list
    b. if K is in keyword_list
        i. get uri_list and append the URI
    c. else
        i. add K to list end,
        ii. append K to keyword_array
```

Time complexity

Taking ***s*** to be the number of subjects, ***l*** the number of LRTs and ***k*** the number of keywords, the time complexity for the insertion algorithm is found, from the IF-statements, to be $s + l + k_S$. So the order of the insertion algorithm is $\mathbf{O(s + l + k_s)}$

3.5 A knowledge based repository

By thinking of LOs and their metadata as pieces of knowledge it might be possible to translate their storage and retrieval in a knowledge base. There is need for the recognition of new contexts in which the LOs can be used in addition to the ones they were originally designed for. Some of the steps that will need to be taken before successful implementation of a KB repository include:

- Identify the boundaries of the environment from which the content will be acquired
- Identify the basic units of the content to be stored
- Identify the metadata fields that will be used for identifying the object.
- Determine, and use, the language of the user [12]
- Build a knowledge base for the learning objects, which is store a reference number or identifier for the objects and have the actual object in a database.

The system layouts shown in Figure 1 and Figure 6 suggest that the structures are comparable. The KB can be thought of to correspond to the Container. The screening process that the KBS conducts using the specifications provided by the user will help reduce the effort a person will exert in putting together a lesson or module.

A learning object does not only have its own characteristics and learning value, but its relationship with other learning objects offers additional learning experiences. The advantage of using a KBS is in that the system will be able to, through heuristics, provide the user with additional objects that have:

i) Been used frequently with a selected LO. This could be through observing or "remembering" what previous users who have accessed that LO have also accessed.

ii) Related LOs that seem to have a common application area with the chosen one.

As the graph is traversed the KBS can identify the popular nodes, or paths, by giving them a weighting and so that whenever a similar search is conducted it will recognize the kind of request and so be able to present the user with additional information. This weighting becomes very useful in applying heuristics to a search. A user can actually go ahead and request URIs for similar searches done before with the system not needing to go through the whole graph.

Chapter 4.

Conclusion and Recommendations

4.1 Conclusions

Learning Objects are the most meaningful and effective way of creating content for e-learning. Despite the fact that there are many attributes in learning object metadata description, they do not fully capture the "behavior" of a learning object. A learning object is created with specific learning objectives in mind, holds specific behavior, and interoperates with surrounding learning objects. Isolating a learning object and reusing it means that either this learning object can remain intact, because it might fit well into the new learning context, or this learning object needs changes.

Genre is an effective way of classifying LOs. A search that uses this kind of classification will perform better and will help in the compilation of more meaning results lists. This can be seen in the results from Table 3. The cases with genre (L) as well as those with subject (S) have a very gentle graph as compared to those without it, meaning that the number of nodes that are visited is much lower than in the other cases. However, the cases that have both the subject and the genre are the lowest showing that the two are very effective in pruning the graph. The results in Table 3 indicate that subject is slightly more effective than genre. This slightness is quite small and might be attributed to subject's positioning in the graph. Despite its being on a lower level genre seems to be an important piece of information that a user can offer the system. The keyword K, in combination with the LRT or subject or both does not make a difference to the results. However, giving just the keyword is least effective. As can be seen from the graphs, the line representing *e(K)*, is the steepest. As the number of keywords increases, the number of edges that need traversal also increase at a faster rate. Other observations from the graphs is that as the number of keywords increases, the number of edges increases, and this increase is amplified as the number of cross-connections is increased.

The algorithms follow the paths that can be defined in the graph for the different cases described in Table 3. The graph structure is such that nodes can be added at any of the 4 levels (excluding the root node). The actual node implementation, which is done with arrays and linked lists (Figure 21) "mimics" the structure of the graph, making it easier to add to and traverse the repository. The degrees of the levels are affected differently by adding a node to its lower level. Level 0 (root) has its degree increased by one for every subject added. This in turn would mean adding ***l*** LRTs, and hence ***l*** edges to level 2 (since LRT is a fixed vocabulary). Adding a keyword, which could mean adding a URI, will directly affect the number of edges in the subject branch to which it belongs but the other subject keyword pools are not affected. Adding a URI will either add a keyword and the respective edge or just add an edge if the keyword already exists. It is difficult to determine how the lower two levels will react in general to the addition of a node to either or both of them.

4.2 Recommendations

This research did not look at how to actually implement the repository in a KBS, therefore the next step would be to put this graph into a knowledge-base system which uses a powerful text processing language like PERL, for the processing of data passed via the Common Gateway Interface (CGI). Also, there is a need to form "synonym" groups within the repository so that objects that are listed under keywords that are related can be included in search results.

There is also need to explore the possibility of linking the KBS to the LMS so that the former can compile a list of extra results based on the searches of users belonging to the same group so that a current user can have an extra list of LOs that might be useful to them. This idea is implemented at *amazon.com*.

References

[1] Advanced Distributed Learning (ADL). (2004) ***Sharable Content Object Reference Model (SCORM®) 2004***. 2nd Edition Overview. Available from: http://cisit.santafe.cc.fl.us/~cat/SCORM_2004_Overview.pdf. Retrieved 10/2005

[2] Ahlstrom, V. A. (2005) **Comparison of Subject-based Classification Strategies for Enhanced Usability**. *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, p1439-1443. Available from http://hf.tc.faa.gov/products/bibliographic/ahlstrom_2005.htm. Retrieved 11/2005

[3] Baker, P. (2005) ***What is IEEE Learning Object Metadata/IMS Learning Resource Metadata?*** Cetis. Available from: www.cetis.ac.uk/lib/media/WhatIsLOMscreen.pdf. Retrieved 10/2005

[4] Barr, A. and Feigenbaum, E. A. (1981) ***The handbook of Artificial Intelligence.*** Volume 1, Los Altos, California: William Kaufmann.

[5] Carnegie Mellon: Learning Systems Architecture Lab. (2005) ***CORDRA^{TM} (Content Object Repository Discovery and Registration/Resolution Architecture).*** Available from: http://www.lsal.cmu.edu/lsal/expertise/projects/cordra/. Retrieved 09/2005

[6] Copeland, J. (2000) ***What is Artificial Intelligence?*** Available from: http://www.alanturing.net/turing_archive/pages/Reference%20Articles/what_is_AI . Retrieved 07/2005

[7] Crowston, K. and Kwasnik, B.H. (2003) ***Can Document-Genre Metadata Improve Information Access to Large Digital Collections?*** *Library Trends*. Available from: http://crowston.syr.edu/papers/libtrends03.pdf. Retrieved 11/2005

[8] Crowston, K. and Kwasnik, B.H. (2004) **A Framework for Creating a Facetted Classification for Genres: Addressing Issues of Multidimensionality**. In *Proceedings of the 37th Hawaii International Conference on System Sciences*. Available from: http://csdl.computer.org/comp/proceedings/hicss/2004/2056/04/205640100a.pdf. Retrieved 11/2005

[9] Dodds, P. and Fletcher, J. D. (2003) ***Opportunities for New "Smart" Learning Environments Enabled by Next Generation Web Capabilities***. Available from:

http://www.cs.kuleuven.ac.be/~erikd/PRES/2003/LO2003/index.html. Retrieved 10/2005

[10] Dolan A, and Aldous J. (1995). **Networks and algorithms. An Introductory Approach**.1st Edition. England: John Wiley & Son

[11] Downes, S. (2002). ***Design and Reusability of Learning Objects in an Academic Context: A New Economy of Education?*** Available from: http://www.usdla.org/html/journal/JAN03_Issue/article01.html. Retrieved 11/2005

[12] Downes, S. (2004). ***The Learning Marketplace. Meaning, Metadata and Content Syndication in the Learning Object Economy.*** Available from http://www.downes.ca/files/book3.pdf. Retrieved 07/2005

[13] Durkin, J. (1994). ***Expert Systems Design and Development.*** Englewood Cliffs, NJ: Prentice Hall.

[14] Duval E and Hodgins W. (2003). ***A LOM Research Agenda.*** Available from: http://www2003.org/cdrom/papers/alternate/P659/p659-duval.html.html. on 10/2005

[15] Duval, E. Hodgins, W. Sutton, and S. Weibel, S, L. ***Metadata principles and Practicalities***. *D-Lib Magazine*, Vol. 8, No.3. 2004 Available from: http://www.dlib.org/dlib/april02/weibel/04weibel.html. Retrieved 12/2005

[16] Euzenat, J. **Towards a principled approach to semantic interoperability**. *IJCAI 2001 Workshop on Ontology and Information Sharing*, Seattle. Available from: http://www-agki.tzi.de/buster/IJCAIwp/Finals/euzenat.pdf. Retrieved 01/2006

[17] Garshol L, M. (2004). ***Metadata? Thesauri? Taxonomies? Topic Maps! Making sense of it all***. Available from: http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html. Retrieved 09/2005

[18] Hatala M, Richards G, Eap T, Willms J. (2004) **The Interoperability of Learning Object Repositories and Services: Standards, Implementation and Lessons Learned**. *WWW2004*, May 17-22. New York, NY, USA. Available from: http://www.sfu.ca/~mhatala/pubs/www04-interop-final.pdf. Retrieved on 10/2005

[19] Heery R, and Anderson S, (2005). ***Digital Repositories Review.*** Available from: http://www.jisc.ac.uk/uploaded_documents/rep-review-final-20050220.pdf. Retrieved 07/2005

[20] Heery, R. and Patel, M. (2000*). **Application Profiles: Mixing and Matching Metadata Schemas***. *Ariadne*, Issue 25. Available from: http://www.ariadne.ac.uk/issue25/app-profiles/intro.html. Retrieved 11/2005

[21] Hopgood, A,A. (1993*). **KNOWLEDGE-BASED SYSTEMS for Engineers and Scientists***. 1st Edition. Boca Raton: CRC Press, Inc.

[22] Ip, A and Canale R, (2003). ***Supporting Collaborative Learning Activities with SCORM***. Available from: http://users.tpg.com.au/adslfrcf/scorm/ED031016.pdf. Retrieved 10/2005

[23] Kennedy A. (2004) ***Automatic Genre Classification of Homepages on the Web***. Available from: http://torch.cs.dal.ca/~kenney/genre_thesis.pdf. Retrieved 10/2005

[24] Lee, Y, and Myaeng, S, H. 2004 **Automatic Identification of Text Genres and their Roles in Subject-Based Categorization.** In *Proceedings of the 37th Hawaii International Conference on System Sciences*. Available from: http://ieeexplore.ieee.org/iel5/893428293/01265269.pdf?arnumber=1265269. Retrieved 10/2005

[25] McGreal R. **Learning Objects: A Practical Definition**. 2004 *International Journal of Instructional Technology and Distance Learning*. September, Vol. 1 No. 9, p21-32. Available from http://www.itdl.org/Journal/Sep_04/. Retrieved 08/2005

[26] Najjar, J., Duval, E., Ternier, S. and Neven, F. 2003. **Towards Interoperable Learning Object Repositories: The Ariadne Experience.** *IADIS International Conference on WWW/Internet 2003,* Vol 1,219-226. Availabel from: http://www.cs.kuleuven.ac.be/najjar/paper/WWW2003_najjar.pdf. Retrieved on 11/2005

[27] Najjar, J. Ternier, S. and Duval, E. (2004*). **Interoperability of Learning Object Repositories: Complications and Guidelines***. Available from http://www.cs.kuleuven.ac.be/~najjar/paper/IadiswwwJournal.pdf. Retrieved 12/2005

[28] Najjar, J, Ternier S, and Duval, E. (2004) ***User Behaviour in Learning Objects Repositories: An Empirical Analysis***. Available from http://www.cs.kuleuven.ac.be/~najjar/papers/edmedia2004.pdf. Retrieved 11/2005

[29] Neven f, Duval E, Ternier S, Cardinaels K, Vandepitte P. (2003). ***An Open and Flexible Indexation- and Query Tool for Ariadne.*** Available from: http://www.cs.kuleuven.ac.be/~hmdb/publications/files/pdfversion/41249.pdf Retrieved 12/2005.

[30] Ragett J and Bains W (1992). ***Artificial Intelligence from A to Z***. London: Chapman and Hall.

[31] Rehak, D, (2004). ***SCORM 2004***. Available from: http://lsal.cmu.edu/lsal/expertise/papers/presentations/idea2004/ideascorm_20040816.pdf Retrieved 09/2005

[32] Rehak, D and Blackmon, W. (2005). ***An Introduction to CORDRA.*** Available from: http://cordra.net/cordra/information/presentations/2005/intro/intro20050204.ppt. Retrieved 09/2005

[33] Rehak R and Mason R. ***Keeping the Learning in the Learning Objects.*** Available from: http://www.lsal.cmu.edu/lsal/expertise/papers/chapters/reusing/. Retrieved 09/2005

[34] Roussinov D, Crowston K, Nilan M, Kwasnik B, Cai J and Liu X. ***Genre Based Navigation on the Web***. *Proceedings of the 34th Hawaii International Conference on System Sciences.* IEEE Computer Society, Maui, Hawaii. 2001

[35] Simon B, David M, Van Assche F, Ternier S and Duval E (Editors). (2005). ***Learning Object Repository Interoperability Framework***. Available from: http://nm.wu-wien.ac.at/e-learning/interoperability/LORInter_V1.0beta_2005_04_13.pdf. Retrieved 11/2005

[36] Todorova M and Petrova V. **Learning Objects**. *International Conference on Computer Systems and Technologies – CompSysTech'2003*

[37] Verbert K, and Duval E. **Towards a Global Component Architecture for Learning Objects: A Comparative Analysis of Learning Object Content Models**. Available from: http://www.cs.kuleuven.ac.be/~hmdb/publications/files/pdfversion/41315.pdf. Retrieved on 11/2005

[38] Verdejo, M.F., Barros, B., Mayorga, J.I. and Read, T. 2002. ***Including Collaborative Learning Designs in a Learning Object Repository***. Available from http://sensei.lsi.uned.es/ea2c2/articulos/Verdejo_AIED03.pdf Retrieved 11/2005

[39] Wastholm, P., Kusma, A. and Megyesi, B.B. (2004) ***Using Linguistic Data for Genre Classification***. Availabale from http://www.wastholm.net/files/suc_genres.pdf. Retrieved 10/2005

[40] Won, K. and Timothy K, S. **On Reusability and Interoperability for Distance Learning**. *Journal of Object Technology*, vol. 3, no. 8, September-October 2004, pp. 27-34. Available from: http://www.jot.fm/issues/issue_2004_09/column3. Retrieved 09/2005

[41] ***AIAWiki.*** http://ai.squeakydolphin.com/wiki.php?pagename=AIAWiki.DepthFirstSearch. Retrieved 04/2005

[42] ***Careo homepage.*** http://www.careo.org/. Viewed 12/2005

[43] ***Data Structures for Graphs.*** http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK2/NODE61.HTM. Retrieved 01/2006

[44] ***Dave Marshall's homepage.*** http://www.cs.cf.ac.uk/Dave?AI2node23.html. Retrieved 05/2005

[45] ***Dublin Core Metadata Initiative.*** http://dublincore.org/documents/dces/. Viewed 09/2005

[46] ***Expert systems building tools: Definitions.*** http://www.wtec.org/loyola/kb/c3_s2.htm. Retrieved 07/2005

[47] ***IMS homepage.*** http://www.imsglobal.org/. Viewed 10/2005

[48] ***Intelligent real-time software embedded systems.*** http://www.engr.sjsu.edu/lwesley/teaching/cmpe196T/LectureSlidesPDF/Lec2.pdf. Retrieved 07/2005

[49] ***Merlot homepage.*** http://www.merlot.org/. Viewed 1/2005 – 01/2006

[50] ***NIST.*** http://www.nist.gov/dads/HTML/breadthfirst.html. Retrieved 07/2005

[51] ***Search.*** http://www.vancouver.wsu.edu/fac/rhewett/ai/search.pdf.

[52] ***STI Knowledge.*** http://www.stiknowledge.com/cccpdemo/cccp_demo_r/glossary.html. Retrieved 07/2005

[53] ***Unit74 Knowledge base techniques*** Available from: http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u74.html#OUT74.1. Retrieved 07/2005

[54] http://members.optusnet.com.au/~webindexing/Webbook2Ed/glossary.htm. Retrieved 01/2006

[55] http://burks.brighton.ac.uk/burks/foldoc/85/30.htm. Retrieved 07/2005

[56] http://en.wikipedia.org/

[57] http://en.wikipedia.org/wiki/Ontotlogy_(computer_science). Retrieved 01/2006

[58] http://www.bedfordsstmartins.com/literature/bedlit/glossary-f.htm. Viewed 11/2005

[59] http://www.cs.sunysb.edu/~skiena/392/lectures/week9/. Retrieved 12/2005

[60] http://www.electronicscriptorium.com/glossary.htm. Retrieved 01/2006

[61] http://www.library.wwu.edu/ref/howtoguides/glossay.html.Viewed 11/2005

[62] http://www.noisebetweenstations.com/personal/essays/metadata_glossary/metadata_glossary.html. Retrieved 01/2006

[63] http://www.thinkhdi.com/publications/glossary.asp. Retrieved 07/2005

[64] http://www.sci.brooklyn.cuny.edu/~parsons/courses/716-spring-2004/notes/lect14-4up.pdf. Retrieved 07/2005

Glossary / Acronym and Abbreviation Key

IEEE	(Institute of Electrical and Electronics Engineering). It is a membership organization including engineers, scientists and students in electronics and allied fields which involves setting standards for communications and computers.
RDF	(Resource Description Framework). A recommendation for the creation of meta-data structures defining data on the Web to improve searching and navigation. RDF is implemented in XML (RDF/XML)
URI	(Uniform Resource Identifier). This is a technology for the identification of resources on the Internet or private intranet.
VLE	(Virtual Learning Environment). This is a software system that is designed to facilitate teachers in the management of educational courses for their students. It can track learner progress. It is often used to supplement the face-to-face classroom.
XML	(Extensible Mark-up Language) A metalanguage written in SGML which is an open standard for describing data. It allows one to design a markup language that allows for the easy interchange of documents on the World Wide Web.

Appendices

Appendix A

IEEE Learning Object Metadata (LOM) data elements

1. **General**
   - 1.1. Identifier
     - 1.1.1. Catalog
     - 1.1.2. Entry
   - 1.2. Title
   - 1.3. Language
   - 1.4. Description
   - 1.5. Keyword
   - 1.6. Coverage
   - 1.7. Structure
   - 1.8. Aggregation Level
2. **Life Cycle**
   - 2.1. Version
   - 2.2. Status
   - 2.3. Contribute
     - 2.3.1. Role
     - 2.3.2. Entity
     - 2.3.3. Date
3. **Meta-Metadata**
   - 3.1. Identifier
     - 3.1.1. Catalog
     - 3.1.2. Schema
   - 3.2. Contribute
     - 3.2.1. Role
     - 3.2.2. Entity
     - 3.2.3. Date
   - 3.3. Metadata Schema
   - 3.4. Language
4. **Technical**
   - 4.1. Format
   - 4.2. Size
   - 4.3. Location
   - 4.4. Requirement
     - 4.4.1.OrCompositon
       - 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
   - 5.3. Interactivity Level
   - 5.4. Semantic Density
   - 5.5. Intended End User Role
   - 5.6. Context
   - 5.7. Typical Age Range
   - 5.8. Difficulty
   - 5.9. Typical Learning Time
   - 5.10. Description
   - 5.11. Language
6. **Rights**
   - 6.1. Cost
   - 6.2. Copyright and Other Restrictions
   - 6.3. Description
7. **Relation**
   - 7.1. Kind
   - 7.2. Resource
     - 7.2.1. Identifier
       - 7.2.1.1. Catalog
       - 7.2.1.2. Entry
   - 7.3. Description
8. **Annotation**
   - 8.1. Entry
   - 8.2. Date
   - 8.3. Description
9. **Classification**
   - 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

Appendix B

Dublin Core Metadata Initiative (DCMI) elements

Title:	Resource name
Creator:	The entity primarily responsible for making the content of the resource. Examples include person, organization or service.
Subject:	A topic of the content of the resource. Will typically be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.
Description:	An account of the content of the resource
Publisher:	The entity responsible for making the resource available. Examples include person, organization or service.
Contributor:	The entity responsible for making contributions to the resource's content. Examples include person, organization or service.
Date:	The date of an event in the lifecycle of the resource (typically associated with the creation or availability of the resource
Type:	Nature or genre of the content of the resource. Includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary
Format:	The physical or digital manifestation of the resource. May include the media type or dimensions. Recommended best practice is to select from a controlled vocabulary.
Identifier:	An unambiguous reference to the resource within a given context. Recommended best is to identify the resource by means of a string or number conforming to a formal identification system e.g. Uniform Resource Identifier (URI), Digital Object Identifier (DOI) and International Standard Book Number (ISDN)
Source:	Reference to a resource from which the present resource is derived. Can identify the referenced resource by means of a string or number conforming to a formal identification system.
Language:	Language of the intellectual content of the resource. E.g. "en" or "eng" for English, or "en-GB" for English used in the United Kingdom.
Relation:	Reference to a related resource. Use formal identification system

Coverage:	The extent or scope of the content of the resource. Recommended best practice is to select from a controlled vocabulary and to use, where appropriate, named places or time periods in preference to numeric identifiers such as sets of coordinates or date ranges.
Rights:	Information about rights held in and over the resource. Often encompasses Intellectual Property Rights (IPR), Copyright and various Property Rights.