P-3575

AN ONTOLOGY APPROACH TO ORGANIZATIONAL MEMORY FOR ENHANCING LEARNING OBJECTS

PROJECT REPORT

Submitted by

T.SATHISHKUMAR

Reg. No: 0920108019

In partial fulfillment for the award of the degree of

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

Department of Computer Science and Engineering

PROJECT WORK

APRIL 2011

This is to certify that the project entitled

AN ONTOLOGY APPROACH TO ORGANIZATIONAL MEMORY FOR ENHANCING LEARNING OBJECTS

is the bonafide record of project work done by

T.SATHISHKUMAR

Register No: 0920108019

of M.E. (Computer Science and Engineering) during the year 2010-2011.

Project Guide

Head of the Department

Submitted for the Project Viva-Voce examination held on 25.04.2011

DECLARATION

I affirm that the project work **titled "AN ONTOLOGY APPROACH TO ORGANIZATIONAL MEMORY FOR ENHANCING LEARNING OBJECTS"** being submitted in partial fulfillment for the award of **M.E** degree is the original work carried out by me. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.

T. Sathishkumar.

SATHISHKUMAR.T

Register No: 0920108019

I certify that the declaration made above by the candidate is true

Rajini

Mrs. S.RAJINI, M.S.,

Associate Professor,

Department of Computer Science and Engineering,
Kumaraguru College of Technology,
(An Autonomous Institution)
Coimbatore-641 049.

GOVERNMENT COLLEGE OF TECHNOLOGY
COIMBATORE - 13
INDIA

IEEE

NCVIT 11

Dr./Prof./Mr./Ms. T. SATHISH KUMAR

of KUMARAGURU COLLEGE OF TECHNOLOGY

has ~~participated~~/presented a paper titled AN ONTOLOGY APPROACH TO MEMORY ORGANISATION FOR ENHANCING LEARNING OBJECTS

in the National Conference on Recent Advances in Computer Vision & Information Technology organized by the Department of Computer Science & Engineering and Information Technology on 7th March, 2011.

ORGANIZING SECRETARY

CHAIRPERSON

PATRON

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project.

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.,** for giving this opportunity to pursue this course.

I would like to thank **Dr.S.Ramachandran, Ph.D.,** Principal for providing the necessary facilities to complete my thesis.

I take this opportunity to thank **Mrs.P.Devaki, Ph.D.,** Head of the Department, Computer Science and Engineering, for her precious suggestions.

I thank all project committee members for their comments and advice during the reviews. Special thanks to **Mrs.V.Vanitha, M.E., (Ph.D),** Associate Professor and **Mr.V.Subramani, M.Tech.,** Associate Professor, Department of Computer science and Engineering, for arranging the brain storming project review sessions.

I register my hearty appreciation to my Guide **Mrs.S.Rajini, M.S.,** Associate Professor, Department of Computer Science and Engineering, who is my thesis advisor. I thank her for the support, encouragement and ideas.

I would like to convey my honest thanks to all **Teaching** and **Non Teaching** staff members of the department for their support. I would like to thank all my classmates for extending technical support whenever I needed it.

I dedicate this project work to my **parents**, without whose love this work wouldn't have been possible.

ABSTRACT

Ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain. Ontology provides a shared vocabulary, which can be used to model a domain - the type of objects and/or concepts that exist, and their properties and relations. The creation of domain ontology is also fundamental to the definition and use of an enterprise architecture framework. This system enables capitalization of existing learning resources by first creating "content metadata" through text mining and natural language processing and second by creating dynamically learning knowledge objects, i.e., active, adaptable, reusable, and independent learning objects. The proposed model also suggests integrating explicitly instructional theories in an on-the-fly composition process of learning objects. Semantic Web technologies are used to satisfy such an objective by creating an ontology-based organizational memory able to act as a knowledge base for multiple training environments.

ஆய்வுச் சுருக்கம்

இன்றைய காலக் கட்டத்தில் இணையதளத்தின் முலம் கற்கும் பொருள்களின் எண்ணிக்கை பெருகி வருவதால், அவற்றை சேமித்தல், தேடல், குறியிடுதல் என்பது எளிதானது அல்ல. பெரும்பாலான முறைகள் தரமான மீத்தரவுகளை குறியிடுதலுக்கு உபயோகித்துப் பொருள்களை கற்கப் பயன்படுகிறது. இருப்பினும் இம்முறை இணையதளத்தில் கற்ப்போருக்கு சரியான உட்ப்பொருள்களை கற்றுக் கொள்ள சாதகமாக அமையவில்லை. எனவே சொற்க்களின் தொடக்கம் பற்றிய ஆய்வு இத்திட்ட வரைவில் அறிமுகப்படுத்தப்ப்டுகிறது.

சொற்க்களின் தொடக்கம் பற்றிய ஆய்வு என்பது களத்தில் உள்ள கருத்துக்களின் தொகுப்பினையும், அவற்றிக்கிடையேயான தொடர்பினையும் வெளிப்படுத்தும் முறையான அமைப்பாகும். இம்முறை களத்தினையும், களத்தின் பண்புகளையும் வரையறுக்க உதவுகிறது. மேலும் இது சொற்ப்பகிர்வினை வழங்கி, களத்தினை வடிவமைக்கிறது. இயக்கநிலை அறிவுப்பொருள்களை உருவாக்கி கற்க்கும் பொருள்களை மேம்படுத்துகிறது. இந்த அறிவுப்பொருள்கள் தனிப்பட்டதாகவும், திருப்பி உபயோகப்படுத்தும் திறனும் உடையது. எனவே சொற்பொருள் சார்ந்த வலையமைப்பு தொழில்நுட்பங்களைப் பயன்படுத்தி, சொற்க்களின் தொடக்கம் பற்றிய ஆய்வின் அடிப்படையில் ஒழுங்குப்பாட்டு நினைவகம் அமைக்கப்பட்டு, கற்றல் பொருள்களின் தன்மை மேம்படுத்தப்படுகிறது.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

OWL	Web Ontology Language
LO	Learning Object
LOR	Learning Objects Repository
LKO	Learning Knowledge Objects
OM	Organizational Memory
LCMS	Learner Centric Management System
LOM	Learning Object Metadata
RDF	Resource Description Framework
OML	Ontology Markup Language
CKML	Conceptual Knowledge Markup Language
SHOE	Simple HTML Ontology Extensions
AI-ED	Artificial Intelligence and Education

CHAPTER 1

INTRODUCTION

1.1 ONTOLOGY

Ontology is a formal, explicit specification of a shared conceptualization. The Reuse of knowledge bases is an important area in knowledge technologies."A Learning Object is an independent and self-standing unit of learning content that is predisposed to reuse in multiple instructional contexts." The semantic web is an extension of current web in which information is given in well defined manner. The semantic web provides a common framework. This framework allows data to be shared and reused across application, enterprise and community boundaries.

The large amount of Learning Objects (LOs) and their continuous growth, the issue of storing, searching, and indexing learning objects is not a trivial one. Most of the approaches suggested the use of standard metadata to index Learning Objects. Even if metadata allow the description of LOs characteristics (language, version, educational purpose, etc.), we believe that this kind of annotation is not sufficient to effectively guide a learner or a teacher towards the right content.

The vision of Learning Object Repositories (LORs) that consider learning objects as static content aggregations seems outdated. This is a particularly acute issue since the corporate world has also adopted learning objects as a mean to ensure the training of its members at a reduced cost. The just-in-time just-enough learning concept requires a very fine grained model of competences and learning materials.

Reusability must be an important parameter in LOs. Polsani [10] stated that, "A Learning Object is an independent and self-standing unit of learning content that is predisposed to reuse in multiple instructional contexts."

Apart from the reusability dimension, this definition implies a number of important factors that should be considered when talking about Learning Objects:

- The independent dimension that implies the autonomy of the LO in terms of data and behaviour, which contrasts with the current vision of LO in the e-Learning community, and
- The pedagogical dimension that should be taken into account when creating a Learning Object but which should not be embedded in the LO itself to allow for multiple instructional adaptations.

The current packaging approach of LO fails to consider these two dimensions. Therefore, new processes must be set up in order to create active Learning Objects, which are able to manage their own conceptual structures and take into account learning instructional theories.

Learning Objects themselves can provide a solid foundation for obtaining such structures. This can be done through a capitalization process of Learning Object content, which consists of disaggregating it into small instructional units and generating valuable structures such as concept maps and domain ontologies.

The generation of such structures necessitates LO content mining. An approach for building Learning Knowledge Objects (LKO), i.e., active, independent, and theory-aware LOs. LKOs are built through a reverse engineering process of existing textual learning resources. This vision is implemented in the Knowledge Puzzle Project through text mining, natural language processing, and semantic annotation leading to an Ontology-based Organizational Memory (OM).

The main objective of the project is to produce a memory with new semantic structures to store fine-grained resources, to propose a new transformation process to feed the memory from LOs or other types of educational resources, to propose a new flexible

composition process of LKOs thanks to the use of the OM, and to propose solutions for LKO deployment in training environments, including standard-based Learning Management Systems (LMS).

The Knowledge Puzzle Project, which enables capitalization of existing Learning Objects with the creation of an Ontology based Organizational Memory. The Organizational Memory concept that represents a new perspective for the e-learning field as it is founded on a rather different idea: Learning Objects as content packages must not exist. Instead, assets, i.e., small fine-grained instructional units, can be exploited by composition mechanisms and learning services in order to aggregate Learning Knowledge Objects to fulfil specific training needs.

Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.

1.1.1 WHY ONTOLOGY

- Share common understanding of the structure
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

In practical terms, developing an ontology includes:

- Defining classes in the ontology,
- Arranging the classes in a taxonomic (subclass–superclass) hierarchy,
- Defining slots and describing allowed values for these slots,
- Filling in the values for slots for instances.

1.1.2 ONTOLOGY USES

Ontologies are used in

- Artificial intelligence
- The Semantic Web
- Systems engineering
- Software engineering,
- Biomedical informatics
- Library science
- Enterprise bookmarking
- Information architecture as a form of knowledge representation

1.1.3 ONTOLOGY COMPONENTS

Common components of ontologies include:

- **Individuals**

  Instances or objects (the basic or "ground level" objects)

- **Classes**

  Sets, collections, concepts, classes in programming, types of objects, or kinds of things.

- **Attributes**

  Aspects, properties, features, characteristics, or parameters that objects (and classes) can have.

- **Relations**

  Ways in which classes and individuals can be related to one another.

- **Function terms**

Complex structures formed from certain relations that can be used in place of an individual term in a statement

- **Restrictions**

Formally stated descriptions of what must be true in order for some assertion to be accepted as input.

- **Rules**

Statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form.

- **Axioms**

Assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In those disciplines, axioms include only statements asserted as *a priori* knowledge. As used here, "axioms" also include the theory derived from axiomatic statements.

- **Events**

The changing of attributes or relations

Ontologies are commonly encoded using ontology languages.

1.2 CONCEPTUALIZATION

Conceptualization is an AI term and consists of a set of objects existing in the target world and the relationships between them. It forms the basis on which many systems performance heavily depend. One of the major difficulties common to most existing systems is the lack of an explicit representation of the conceptualization each systems is based on.

Taking an expert system as an example, it has a knowledge base in which it declaratively represents what it knows about problem solving in the task domain.

The knowledge base can be a source of intelligence used to solve problems. Once a user tries to modify the knowledge base or to reuse some portion of an existing knowledge base developed by other people, however, he/she immediately has serious difficulty in doing so. This is due to the fact that the conceptualization of the knowledge base as well as the underlying assumptions are implicit. Even worse, the same terms may have different meanings to each user. Such information has rarely been represented explicitly, which has been a serious cause of many drawbacks that current knowledge-based systems suffer from. The same applies to IISs.

Few IISs have an explicit representation of their conceptualization. While they know about the understanding state of the learner and have tutoring knowledge, etc., they do not know any concept of which their knowledge is composed. In other words, most of the concepts, as well as their primitive actions are hard-wired.

We could say that such systems are illiterate because they do not understand the basic concepts. Therefore, such systems will never be able to communicate with users (authors) in terms of fundamental knowledge. An authoring system should be aware of this conceptualization of an IIS because it creates an IIS by manipulating concepts according to the design rationale employed.

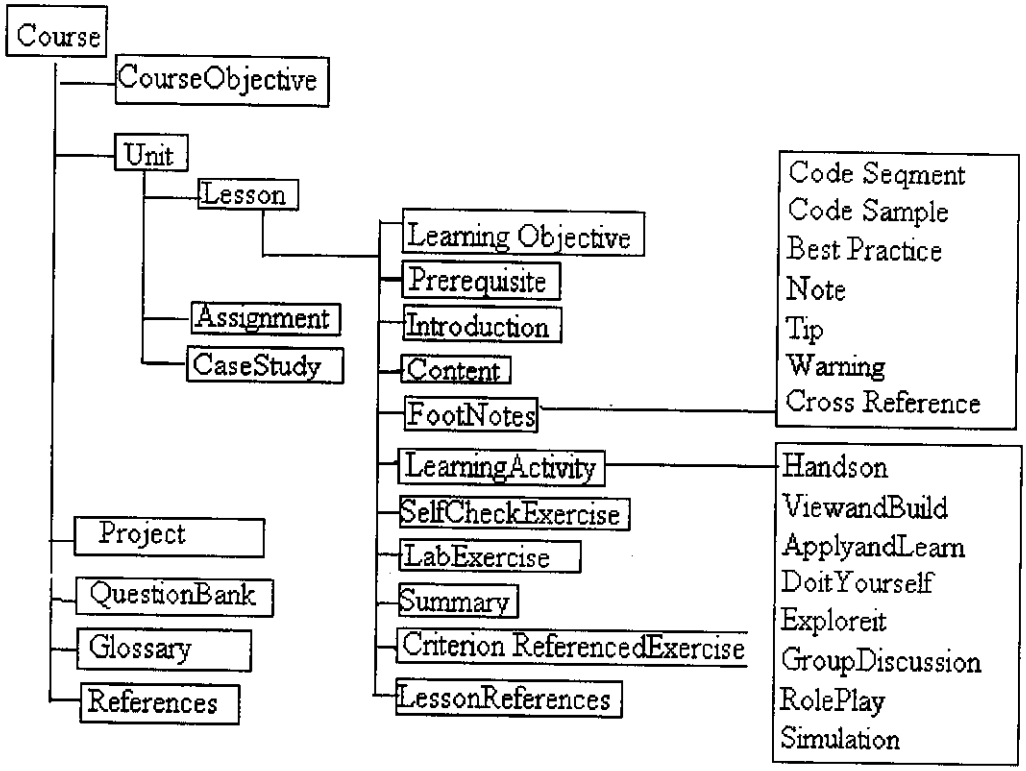


Figure 1.1: Learner-centric LCMS Conceptual Model

1.3 E-LEARNING SYSTEM

In the existing system, e-learning leads to evolutions in the way of designing a course. A course can be seen as an organization in which different actors are involved. These actors produce documents, information and knowledge that they often share. An ontology-based document-driven memory which is particularly adapted to an e-learning situation. The utility of a shared memory is reinforced in this kind of situation, because the interactions do not usually occur in the same place and in the same time.

First we precise our conception of e-learning and we analyze actors needs. Then we present the main features of our learning organizational memory and we focus on the ontologies on which it is based.

We consider two kinds of ontologies:

- The first one is *generic* and concerns the domain of training;
- The second one is related to the *application domain* and is specific to a particular training program.

1.3.1 Difficulties in the e-learning system

- In the existing system we widely make use of Google to search information on the internet. But Google search engine finds hard to search for suitable ontology files.
- Google searches files, based on keywords given by the users. In this case it does not check the real content and structure of the files. Due to this some irrelevant files will be returned to the user with respect to the keyword given.
- Time consumption is more in the existing system.

CHAPTER 2

LITERATURE REVIEW

2.1 ADAPTIVE LEARNING OBJECT TECHNOLOGY

The Web offers the perfect technology and environment for individualized learning because learners can be uniquely identified, content can be specifically personalized, and learner progress can be monitored, supported, and assessed. Technologically and technically, researchers are making progress toward realizing the personalized learning dream with adaptive Learning Object technology [9].

The two important considerations are being ignored or overlooked in accomplishing the personalization dream. One missing consideration concerns a whole-person understanding about key psychological sources that influence how individuals want and intend to learn online. Conventional, primarily cognitive solutions (which focus on how learners' process, build, and store knowledge) offer a restricted view of how people learn and too often lead to unstable or ineffective online learning solutions. A more whole-person perspective includes emotions and intentions as critical factors in the learning process. Also missing is the integration of instructional purpose, values, and strategies into the design, development, and presentation of content (objects). Up to now, developments have focused on technology rather than more important learner-centric issues.

2.2 AUTOMATING METADATA GENERATION

Cardinaels [7] focus on the development of a framework for automatic metadata generation. The first step towards this framework is the definition of an Application Programmer Interface (API), which we call the Simple Indexing Interface (SII). The second step is the definition of a framework for implementation of the SII. It also report on empirical evaluation of the metadata that the SII and supporting framework generated in a real-life context.

The Learning Object Metadata can be derived from two different types of sources. The first source is the learning object itself; the second is the context in which the learning object is used. Metadata derived from the object itself is obtained by content analysis, such as keyword extraction, language classification and so on. The contexts typically are learning management systems in which the learning objects are deployed. A Learning Object context provides with extra information about the learning object to define the metadata. The framework consists of two major groups of classes that generate the metadata, namely Object-based indexers and Context-based indexers. The object-based indexers generate metadata based on the learning object itself, isolated from any other learning object or learning management system. The second class of indexers uses a context to generate metadata. The framework is easily extensible for new learning object types and new contexts. To be complete, the framework also has some Extractors that for example extract the text and properties from a PowerPoint-file, and a Metadata Merger that can combine the results of the different indexers into one set of metadata.

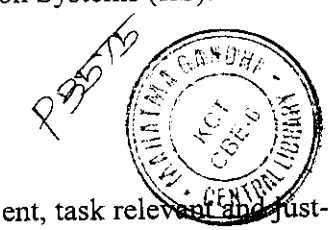
2.3 ONTOLOGY-BASED AUTOMATIC ANNOTATION OF LEARNING CONTENT

Jovanovi [14] approaches to automatic annotation of Learning Objects' (LOs) content units that we tested in TANGRAM, an integrated learning environment for the domain of Intelligent Information Systems. The approach does not primarily focus on automatic annotation of entire LOs, as other relevant solutions do. It provides a solution for automatic metadata generation for LOs'components (i.e., smaller, potentially reusable, content units). Here we mainly report on the content mining algorithms and heuristics applied for determining values of certain metadata elements used to annotate content units.

The main focus is on the following elements: title, description, unique identifier, subject (based on a domain ontology), and pedagogical role (based on an ontology of pedagogical roles). Additionally, as TANGRAM is grounded on an LO content structure ontology that drives the process of an LO decomposition into its constituent content units, each thus generated content unit is implicitly semantically annotated with its role/position in the LO's structure. Employing such semantic annotations, TANGRAM allows assembling content units into new LO's personalized to the user's goals, preferences, and learning styles.

Learning Content represented in the form of reusable Learning Objects (LOs) promised to significantly reduce the time and cost of authoring high-quality learning materials, making them more affordable and readily available. The principal objective is to enable faster, cheaper, and better learning. Current research efforts are almost exclusively oriented toward reusability of Los in their entirety. Annotations of LOs with the standard compliant metadata sets Learning Object Metadata [LOM] aim at enabling search and retrieval of existing LOs stored in LO repositories. Accordingly, metadata is seen as the primary mean for fostering LOs reusability. The process of authoring new learning materials typically proceeds in the following steps: an author first searches both LO repositories and the Web to find potentially useful learning content.

To enable reusability of content units of varying granularity levels, an explicit definition of the LO's structure is needed. Additionally, if the process of reusing content units has to be (semi-)automatic, the definition of the LO's structure must be formally specified and expressed in a machine understandable language. Ontologies and Semantic Web languages provide means to achieve both things. In this paper, the approach to automatic annotation of LOs' components based on a number of ontologies. The approach is tested in TANGRAM — an integrated learning environment for the domain of Intelligent Information Systems (IIS).



2.4 E-LEARNING BASED ON THE SEMANTIC WEB

Berners-Lee [6] has mentioned that the E-Learning is efficient, task relevant and just-in-time learning, grown from the learning requirements of the new, dynamically changing, distributed business world. The term "Semantic Web" encompasses efforts to build a new WWW architecture that enhances content with formal semantics, which enables better possibilities for navigating through the cyberspace and accessing its contents. As such, the Semantic Web represents a promising technology for realizing eLearning requirements. This leads to an approach for implementing the eLearning scenario using Semantic Web technologies. It is primarily based on ontology-based descriptions of content, context and structure of the learning materials and thus provides flexible and personalized access to these learning materials.

ELearning is just-in-time education integrated with high velocity value chains. It is the delivery of individualized, comprehensive, dynamic learning content in real time, aiding the development of communities of knowledge, linking learners and practitioners with experts. The consequences of such organisation on the learning are expensive, slow and too unfocused (problem-independent) learning process. But dynamically changed business environment puts completely different challenges on learning process – fast, just-in-time (cheap) and relevant (problem-dependent) learning, as mentioned in first section. This can be solved with the distributed, student-oriented, personalised, nonlinear/ dynamic learning process – eLearning. *Table 2.1* shows the characteristics of the standard learning and improvements achieved using the eLearning environment. These are also the most important characteristics of eLearning.

Table 2.1 Differences between training and eLearning

Dimension	Training	Elearning
Delivery	Push– Instructor determines agenda	Pull – Student determines agenda
Responsiveness	Anticipatory – Assumes to know the problem	Reactionary – Responds to problem at hand
Access	Linear – Has defined progression of Knowledge	Non-linear – Allows direct access to knowledge in whatever sequence makes sense to the situation at hand
Symmetry	Asymmetric – Training occurs as a separate activity	Symmetric – Learning occurs as an integrated activity
Modality	Discrete – Training takes place in dedicated chunks with defined starts and stops	Continuous – Learning runs in the parallel loops and never stops
Authority	Centralized – Content is selected from a library of materials developed by the educator	Distributed – Content comes from the interaction of the participants and the educators
Personalization	Mass produced – Content must satisfy the needs of many	Personalized – Content is determined by the individual user's needs and aims to satisfy the needs of every use
Adaptivity	Static – Content and organization/taxonomy remains in their original authored form without regard to environmental changes	Dynamic – Content changes constantly through user input, experiences, new practices, business rules and heuristics

The principle behind eLearning is that the tools and knowledge needed to perform work are moved to the workers – wherever and whoever they are. Simply put, eLearning revolves around people. This is in stark contrast to the way learning has typically involved

people flocking around the learning, i.e. a typical scholastic environment. ELearning has its origins in Computer-Based Training (CBT), which was an attempt to automate education, replace a paid instructor, and develop self paced learning. The ability to reduce the cycle time for learning and to adapt "content, size and style" of learning to a user and to the business.

2.5 USING ONTOLOGICAL ENGINEERING TO OVERCOME COMMON AI- ED PROBLEMS

2.5.1 Analysis of current state of the art of AI-ED research

Mizoguchi [17], first enumerate drawbacks of current IISs from AI and ED point of views.

(1) There is a deep conceptual gap between authoring systems and authors.

(2) Authoring tools are neither intelligent nor particularly user-friendly.

(3) Building an IIS requires a lot of work because it is always built from scratch.

(4) Knowledge and components embedded in IISs are rarely sharable or reusable.

(5) It is not easy to make sharable specifications of functionalities of components in IISs.

(6) It is not easy to compare or cross-assess existing systems.

(7) Communication amongst agents and modules in IISs is neither fluent nor principled.

(8) Many of the IISs are ignorant of the research results of IS/LS

(9) The authoring process is not principled.

(10) There is a gap between instructional planning for domain knowledge organization and tutoring strategy for dynamic adaptation of the IIS behavior.

All these issues are content-related ones. In other words, neither inference techniques nor beautiful theoretical formalism can contribute to an improvement of the situation. The authors' claim that what we need to overcome these drawbacks is ontology-based architecture and appropriate ontologies, that is, the introduction of ontological engineering. Now, need to analyze the situation in more detail to see the justifications of how ontological engineering can make a useful contribution.

2.5.2 Intelligence

Adaptively is at the heart of intelligent systems[17]. It comes from the declarative representation of what the system knows about the world it is in. In IIS cases, the world consists of a learner and the system itself. So such a system behaves adaptively with respect to the learner's state of understanding. A learner model, which is a representation of system's knowledge about the learner, serves as the source of intelligence. The system can investigate the learner model to adapt its behavior to the learner. In this sense, the model should be represented declaratively in order for the system to update and interpret. This is one of the major reasons why authoring systems are not intelligent. Intelligence of authoring systems is not an issue specific to authoring system research but a general issue common to most AI-ED research, since it is deeply related to knowledge of how to build IISs. As discussed above, the source of intelligence of the systems is declarative descriptions of what they know. They can dynamically inspect these declarative descriptions in order to adapt their behavior to the circumstances under which they are operating. The intelligent systems know what they know and what they are doing. The next issue is what knowledge authoring systems should possess. Conventional authoring can be viewed as a kind of Knowledge Acquisition (KA) from teachers/ instructors/trainers. This view gives us a few constructive suggestions to improve the performance of authoring systems. The learner could learn from the research of knowledge acquisition in knowledge-based systems community where KA from domain experts has been the bottleneck of expert system building. One of the major causes of the difficulty is due to the heavy dependence on heuristic knowledge, which is hard to acquire, to manipulate and justify. This is why model based approaches are incorporated in modern knowledge-based systems which rely mainly on domain theory and model of the target system instead of heuristics. In this case, analogously, a "Model-based approach" is strongly needed to make IIS building more scientific and principled. Need to depart from the "heuristics based approach" and employ a new technology, that is, ID knowledge to enable the development process to follow instructional design decisions based on principled and theory-based knowledge.

2.5.3 Conceptualization

Conceptualization is an AI term and consists of a set of objects existing in the target world and the relationships between them [17]. It forms the basis on which many systems' performance heavily depends. One of the major difficulties common to most existing systems is the lack of an explicit representation of the conceptualization each systems is based on. Taking an expert system as an example, it has a knowledge base in which it declaratively represents what it knows about problem solving in the task domain. The knowledge base can be a source of intelligence used to solve problems. Once a user tries to modify the knowledge base or to reuse some portion of an existing knowledge base developed by other people, however, he/she immediately has serious difficulty in doing so. This is due to the fact that the conceptualization of the knowledge base as well as the underlying assumptions is implicit. Even worse, the same terms may have different meanings to each user. Such information has rarely been represented explicitly, which has been a serious cause of many drawbacks that current knowledge-based systems suffer from. The same applies to IISs. Few IISs have an explicit representation of their conceptualization. While they know about the understanding state of the learner and have tutoring knowledge, etc., they do not know any concept of which their knowledge is composed. In other words, most of the concepts, as well as their primitive actions are hard-wired. Such systems are illiterate because they do not understand the basic concepts. Therefore, such systems will never be able to communicate with users (authors) in terms of fundamental knowledge. An authoring system should be aware of this conceptualization of an IIS because it creates an IIS by manipulating concepts according to the design rationale employed.

2.5.4 Standardization

Needless to say, industries have attained today's high productivity thanks to the standardization of components, for example, bolts and nuts. It is a pity that we have no such standardized components in the AI-ED community. In order to model target objects, such components would help a lot and facilitate model-based problem solving. Standardization of components does not necessarily imply standardization of knowledge in general. Using standardized basic components, one can easily design one's own knowledge by configuring them, as is supported by the current engineering production. Standardization is mainly for

providing a common vocabulary for understanding what has been done to date with less ambiguity. It never implies any restriction to the exploration of future research activities. However, one may say that AI-ED research would be premature in establishing a standard. If "standardization" is a bit too strong, we could use the term "shared vocabulary". It sounds much more gentle and acceptable. Specification of functional components should be described in terms of common vocabulary. The problem, however, is that the terminology used by teachers, authors, and developers are different from each other. As is discussed above, implemented systems do not understand either of the vocabularies. In short, none of the four participants, three humans and one computer, share a common vocabulary. This has caused a lot of misunderstanding. This is even the case among human participants. Furthermore, when they start discussion on comparison between several IISs of different domains, it is not easy for them to properly perform the comparison because of the different terminologies used in the respective systems. The same applies to communication between component modules in IISs. Because no common functional specifications or vocabulary are available, the components cannot communicate with each other properly. This is one of the main factors which prevent their reuse.

2.5.5 Theory-Awareness

In general, expertise is composed of heuristics and domain theories in general. Once it is extracted and is at hand, heuristic knowledge is easy to implement and resulting systems usually works well. Many knowledge-based systems have been built employing heuristic knowledge. One of the shortcomings of heuristic approach is that, it is not principled and it ignores existing theories [17]. Another way of building a knowledge-based system is to use domain theories which are, in general, objective and convincing. People can easily accept such systems that are based on theories. The issue here is that all of the theories in many of the theory based systems are built-in in the procedures. The developer, not the system, knows the theory. Such a system cannot be said to be "theory-aware". Authoring systems, which are a kind of meta-system in the sense that they generate IISs, need to be "theory-aware" to be intelligent. The rich accumulation of instructional and learning theories should be used to make authoring systems knowledgeable. Declarative representation of those theories enables them to be called "theory aware".

2.6 KNOWLEDGE AND ONTOLOGICAL ENGINEERING

2.6.1 Knowledge Modelling and Ontology

Bourdeau [17] expert system community, the knowledge principle, "The power exists in the knowledge", proposed by Feigenbaum has been accepted and carried out with a deep appreciation, since it is to the point in the sense that the importance of the accumulation of knowledge is larger than that of formal reasoning and logic in making expert systems work. This has been proved by the success of expert system development and a lot of research activities have been carried out under the flag of "knowledge engineering". After an initial success, however, people realized serious difficulties in knowledge base technology, that is, the expense of building each knowledge base and the low reusability of knowledge bases. They tried to deal with heuristic knowledge using simple rule base technology. They did not have any sophisticated methodologies or theories for eliciting knowledge from the knowledge sources or transforming, organizing, and translating the domain knowledge to enable the computer to utilize it. Knowledge modelling, which is a substitute for rule base technology, is a new technology for overcoming such difficulties. It has made it easier to elicit expertise, to organize such expertise into a computational structure, and to build knowledge bases. The idea is to find domain-independent activities which specify the roles that the domain objects play in the problem solving process. The latest knowledge modelling research introduces the idea of ontology. Roughly speaking, ontology consists of task ontology, which specifies the problem solving architecture of knowledge-based systems, and domain ontology, which specifies the domain knowledge. The concept of task ontology has been proposed by one of the authors and it serves as a theory of vocabulary/concepts used as building blocks for the modeling problem solving structure. Task ontology provides us with an effective methodology and vocabulary for both analyzing and synthesizing knowledge-based systems. It is a system of terms/concepts used to describe how human experts perform the task (problem solving) domain-independently. Task ontology could be what we need to make knowledge-based systems aware of what they know about the task they are performing, and of what conceptualization the knowledge in the knowledge base is based upon, etc.

Research on ontology from an engineering point of view is called ontological engineering. Thus, knowledge engineering has developed into ontological engineering. It is true that knowledge is domain-dependent, and hence knowledge engineering, which directly investigates such knowledge, has been suffering from a rather serious difficulty caused by its specificity and diversity. However, ontology research is different. Ontological engineering investigates knowledge in terms of its origins and elements from which knowledge is constructed. The hierarchical nature of concepts and the decomposability of knowledge are exploited to deeply investigate primitives of knowledge as well as background theories of knowledge which enables us to avoid the difficulties that knowledge engineering has been faced with. Here, by a task, we mean a problem solving process like diagnosis, monitoring, scheduling, design, and so on. In our context, instruction is a task, as is supporting the learning process. Task ontology is obtained by analyzing the task structures of real world problem solving. It does not cover the control structure but do components or primitives of unit activities taking place during the performance of the tasks. The ultimate goal of task ontology research includes providing a theory of all the concepts necessary for building a model of the human problem solving processes.

2.6.2 Computational Semantics of an Ontology

This is one of the most crucial points of an ontology. Regardless of the fact that an ontology sometimes looks like just a set of terms, it has richer computational semantics. One of the authors has proposed the following three levels of ontologies.

Level 1: A structured collection of terms. The most fundamental task in ontology development is articulation of the world of interest, that is, elicitation of concepts and identifying the so-called is-a hierarchy among them. These are indispensable things to an ontology. Typical examples of ontologies at this level include topic hierarchies found in internet search engines and tags used for metadata description. Little definition of the concepts is made.

Level 2: In addition to the contents of a level 1 ontology, we can add formal definitions to prevent unexpected interpretation of the concepts and necessary relations and constraints also formally defined as a set of axioms. Relations are much richer than those at the level 1. Definitions are declarative and formal to enable them to be interpretable by computers.

The interpretability of an ontology at this level enables computers to answer questions about the models built based on the ontology. Many of the ontology building efforts aim to build ontologies at this level.

Level 3: The ontology at this level is executable in the sense that models built based on the ontology run using modules provided by some of the abstract codes associated with concepts in the ontology. Thus, it can answer questions about the runtime performance of the models. Typical examples of this type are found in task ontologies. Software components in component ware roughly correspond to an ontology at this level. But, they have nothing corresponding to levels 1 and 2.

2.6.3 Ontology Providing Us with a Solution

A level 1 ontology provides a set of terms which should be shared among people in the community, and hence could be used as well-structured shared vocabulary. These terms enables us to share the specifications of components' functionalities, tutoring strategies and so on and to properly compare different systems. Ontology is also defined as "an explicit specification of a conceptualization" which suggests that it explicitly represents the underlying conceptualization which has been kept implicit in many cases. A Level 2 ontology is composed of a set of terms and relationships with formal definitions in terms of axioms. Such axioms are declarative, and hence such an ontology represents the conceptualization declaratively. A level 2 ontology is the source of intelligence of an ontology-based system. Another role of an ontology is to act as a meta-model. A model is usually built in the computer as an abstraction of the real target. An ontology provides concepts and relationships which are used as the building blocks of the model. Axioms give semantic constraints among concepts. Thus, an ontology specifies the models to build by giving guidelines and constraints which should be satisfied. This is how the function is viewed at the meta model level. Needless to say, this characteristic is what an authoring system really needs. In fact, we can find some research based on the meta-model function of an ontology. A shared ontology is a first step towards standardization. Not only informal definitions of terms/concepts but also intermediate concepts are made explicit by the level 1 ontology. The structuring usually employs is-a and part-of links to relate concepts to each other. The structure obtained in a level 1 ontology itself represents an understanding about the

domain of the developer. It is usually much more informative than definition of a term. An ontology cannot instantly become a standard. An ontology designed gives a test-bed for establishing a standard. On the basis of standardized terms and concepts, knowledge of the domain can be systematized in terms of the concepts and standardized relationships identified in the ontology. This is what we are intending to do in our ambitious plan "building an ontology of Instructional Design" which makes an authoring system "ID-theory-aware".

2.6.4 A Road Map

The following is a road map towards the new direction. Challenges are threefold:

1) The sharing among humans, and through computer technology, of the knowledge we have accumulated thus far.

2) The sharing extended from among humans to among computers,

3) The operationalization of this knowledge to support the building of IISs.

The first challenge is to have computers mediate the sharing of our knowledge. In order to enable computers to mediate humans in knowledge sharing, there has to be more than an information retriever. If the knowledge is derived from different conceptualizations of the world of interest, and unrelated terms are used to represent the knowledge, computers cannot do the mediation. They need at least to share the fundamental conceptualization of the target world with humans, and a common vocabulary for representing the knowledge, in order to do meaningful mediation using the common terms. The level 1 ontology plays a sufficient role in this goal. This would be the first step in our enterprise. It is important to note that "shared vocabulary" does not mean excluding the variety of theories based on different viewpoints, nor does it mean standardization of knowledge. The goal is to share primitive concepts in terms that we can use to describe the knowledge and theories, in full respect of their integrity. We anticipate that most of the Instructional knowledge can be described in terms of a shared vocabulary. One further step towards the level 1 ontology would make a concrete contribution to the goal. The second challenge is to extend this sharing from among humans to among computers. Since a shared vocabulary is not rich in meaning - it consists only of a set of common terms structured in terms of is-a and part-of links resulting in a semantically poor situation, computers cannot go deeper. To enhance the computer's intelligence, the level 2 ontology introduces definitions of each term

and richer relationships than in level 1, using axioms. An axiom relates two concepts semantically, which allows computers to partially understand the rationale of the configuration of the world of interest, in this case learning and instruction. The operationalization of this knowledge leads to the building of IISs. This requires a level 3 ontology to enable computers to run the code corresponding to the activity-related concepts. Knowledge at this level is mainly concerned with task ontology which contains concepts of action of the system in performing a specific task (instruction, learning support). When we have a shared vocabulary, the ID-knowledge server communicates with humans who need help in finding ID knowledge appropriate for their goals; it can give justifications to an ontology in ID-knowledge-aware authoring environments. Thus, such authoring environments can discuss the appropriateness of strategies adopted with authors helped by the ID knowledge server. Future IISs developed in this way would assist the seamless flow of knowledge from the designers to the learners. One step in this direction could be to build upon an existing courseware engineering workbench such as AGD, to isolate one set of ID decisions and re-engineer them based on task ontology; a micro-domain ontology could be built from meta-models related to this set. This preliminary work could illustrate and test the idea of an ID Ontology server.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This project proposes a formal representation of a set of concepts within a domain and the relationships between those concepts. The domain ontology is created with the help of Protégé tool. Creation of domain ontology enables the learning objects to be shared and reused. Hierarchical views for the domains are created. For each concept the learning materials are fed into all domains. The concept map is designed to analyze the related links for a particular domain.

3.2 THE KNOWLEDGE PUZZLE PRODUCTION SUBSYSTEM

The Knowledge Puzzle architecture includes two subsystems: the first one, the production subsystem, which enables the constitution of the OM's elements and the second one their exploitation. The production subsystem is made up of two major tool suites: ONTO-AUTHOR and ONTO-ENGINE. The latter simply includes TEXCOMON and the Protege Ontology Environment [20]. The Protege OWL Java API is employed as the communication language between the tools and the ontologies. The first component, TEXCOMON, exploits pattern matching techniques to learn the domain ontology from the document pool.

3.3 CAPITALIZING LEARNING OBJECTS THROUGH AN INNOVATIVE ONTOLOGY ENGINEERING APPROACH (TEXCOMON)

The most important tool within the ONTO-ENGINE system is TEXCOMON. ONTO-ENGINE is simply a framework that includes both TEXCOMON and Protege, and uses the Protege OWL Java API for communication purposes.

Table 3.1 Lexicon-Syntactic Pattern and Their Semantic Transformation Methods

Input Links(t)	Output links(t)	Method
-	NOMINAL_SUBJECT(nsubj) with u as destination, DIRECT_OBJECT(dobj) with v as destination	Create a new relationship between u and v labelled with t.
-	NOMINAL_SUBJECT(nsubj) with u as destination, COPULA(cop) with v as destination	Create a new relationship between u and t labelled v.
Rcmod (Relative clause modifier)	Passive Nominal Subject(nsubjpass) with u as destination, Preposition(prep_) with v as destination	• Ignore u because it is a relativ pronoun • s=Find subject(t) • newLabel=concatenate t and th preposition prep • Create a new relationshi between s and v labelle newLabel.

TEXCOMON stands for Text Concept Maps-Ontology to indicate the process followed in order to convert texts into domain concept maps, which are, in turn, transformed into OWL ontology. This ontology represents the implicit domain knowledge contained in the learning objects, which is currently not accessible to training environments. The ontology engineering process is as follows: textual learning objects are taken as inputs and an index structure is created. This index structure decomposes each document into paragraphs and sentences using Unstructured Information Management Architecture (UIMA) - based annotators [21]. Other structural annotations can be performed to manually identify figures, tables, etc. A Keyword extraction algorithm [10] is then executed in order to retrieve document keywords and

key sentences. These sentences are parsed through the Stanford Statistical Parser [15] that outputs a typed dependency network [9] for each sentence. In TEXCOMON, the typed dependency network is called a grammatical concept map. Once these grammatical structures are available, they are mined to find instances of lexicon-syntactic patterns that we defined in a linguistic knowledge base (made up of around 20 patterns for the moment).

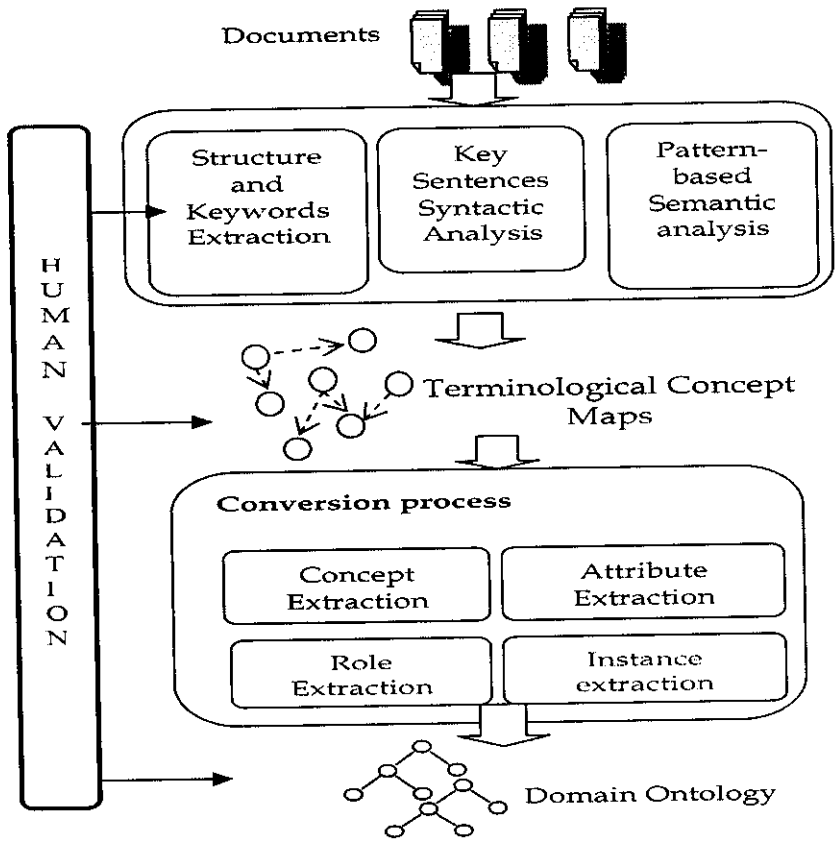


Figure 3.1 The Domain Knowledge Acquisition Process in Texcomon

The lexicon-syntactic pattern allows identifying grammatical sub graphs that can be transformed to obtain semantic representations. ***Table 3.1*** shows examples of lexicon-syntactic patterns and their semantic transformation methods. The table depicts the transformation process from grammatical links to semantic links. Let t, u, and v be the domain terms linked by grammatical links (input and output links).

A pattern is defined as a data structure composed of input and output grammatical links organized around a given term t. These links constitute the syntactic structure that should be fetched in each key sentence. Each detected configuration triggers a method to obtain a sentence concept map. Formalizing such patterns in a linguistic knowledge base allows for progressively identifying grammatical structures and their semantic possible interpretations. However, adding new patterns does not imply changing the underlying framework, thus making it flexible and extensible.

Domain independent linguistic knowledge bases can then be reused in other contexts. All the semantic maps are merged in order to create Domain Concept Maps (DOMCMAP) around concepts. TEXCOMON integrates the different semantic representations (relationships and concepts) around a given domain term. Hence, one concept map can be built from different documents. ***Figure 3.1*** depicts the domain knowledge acquisition process in TEXCOMON.

A Domain Ontology (DOM-ONTO) is created from the available concept maps by detecting ontological concepts and relationships. Since the process of learning a domain ontology follows a set of learning tasks defined in [5], TEXCOMON implements various tasks by determining significant concepts, attributes, relationships (hierarchical and conceptual), and instances. The significance of a concept is manually defined by stating a threshold representing the out-degree of a given concept. This states that a concept is considered as ontological when it has a significant number of relationships with others. The relationships linking significant concepts are also considered as ontological. Other patterns are used to determine hierarchical links, instances, and attributes. TEXCOMON encodes Hearst's patterns [12] for discovering hyponyms and instance relationships. In fact, differentiating subclasses from instances is a difficult problem. In TEXCOMON, when a hyponym link is detected between two domain concepts, then a subclass link is created.

3.4 CAPITALIZING LEARNING OBJECTS THROUGH SEMANTIC ANNOTATION AND EDITION (ONTO_AUTHOR)

Contrary to Polsani's definition of learning object, we believe that it is not the learning object itself that should be reused in multiple instructional contexts, but the instructional resources that compose the learning object. This necessitates that these resources are clearly

identified. ONTO AUTHOR is an authoring environment that uses the ontologies to define and extract the resources layer of the OM.

ONTO_AUTHOR includes a variety of tools. Polsani [19] stated that, "The formal composition of a LO is the arrangement of elements. An element could be text, image, video, animation, glossary, assessment, multimedia, etc. Preferably a LO should be a combination of multiple elements. The multiplicity not only reinforces the concept communicated, but it also opens up multiple avenues to foster a richer understanding of the idea(s) represented, facilitating learning based on learners' choices and learning characteristics."

The Instructional Role Annotator is a semantic annotation tool dedicated to the annotation of instructional role instances in a document (or a learning object). This annotation aims at producing assets that will be used for LKO composition. Assets are very fine-grained knowledge blocks that confer a very high flexibility to the LKO composition process.

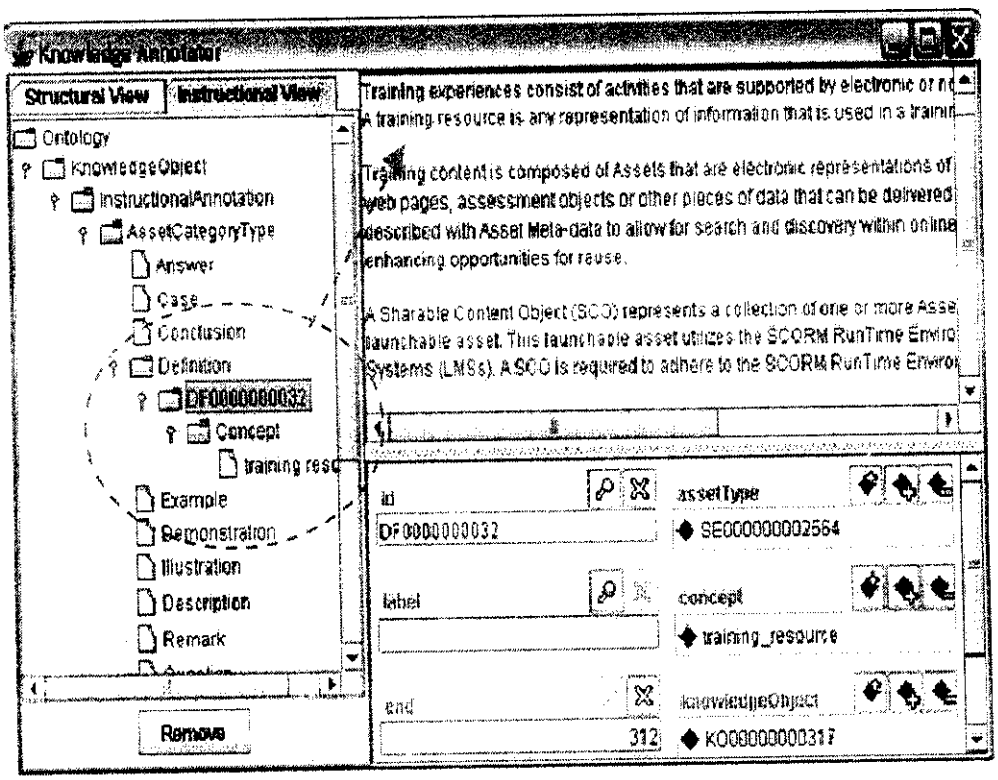


Figure 3.2 The Instructional Role Annotator

They are elements that have a pedagogical function. For example, a text fragment can be a definition of a concept X. This definition can be reused in multiple training contexts that are linked to this concept X. ***Figure 3.2*** shows the Instructional Role Annotator Tool.

The competence editor defines competences as skills that are linked to domain ontology concepts through an OWLObjectProperty "Concept." There is a debate on whether competence ontologies should also model the domain knowledge or if they should only take into account competence levels terminology leaving the domain knowledge in a separate ontology. Keeping both ontologies separate from each other's fosters the reusability of the competence ontology and enables the graceful migration or evolution of the domain knowledge in an independent manner.

The SWRL Rule Editor serves to define the memory rule layer. Protégé´ already provides one editor of this kind [8]. Finally, the theory editor enables the creation a learning theory and linking it to a set of instructional events represented as instances of the class "Instructional Event." For example, Gagne''s theory states that the first instructional event should be to gain learner's attention.

The instructional events of a given theory are then associated to SWRL rules in order to refer to OM's assets. A new theory is created by stating its instructional events and the rules associated to each one of them. For the moment, the tool is quite simple, but further improvements will enable the integration of more complex concepts such as learning conditions and other types of constraints that can occur in the learning process.

3.5 THE KNOWLEDGE PUZZLE EXPLOITATION SUBSYSTEM

The OM is used as a knowledge base that sustains the dynamic aggregation of learning knowledge objects. The OM feeds the Knowledge Puzzle Exploitation process that is composed basically of three layers: the composition layer, the standardization layer, and the deployment layer.

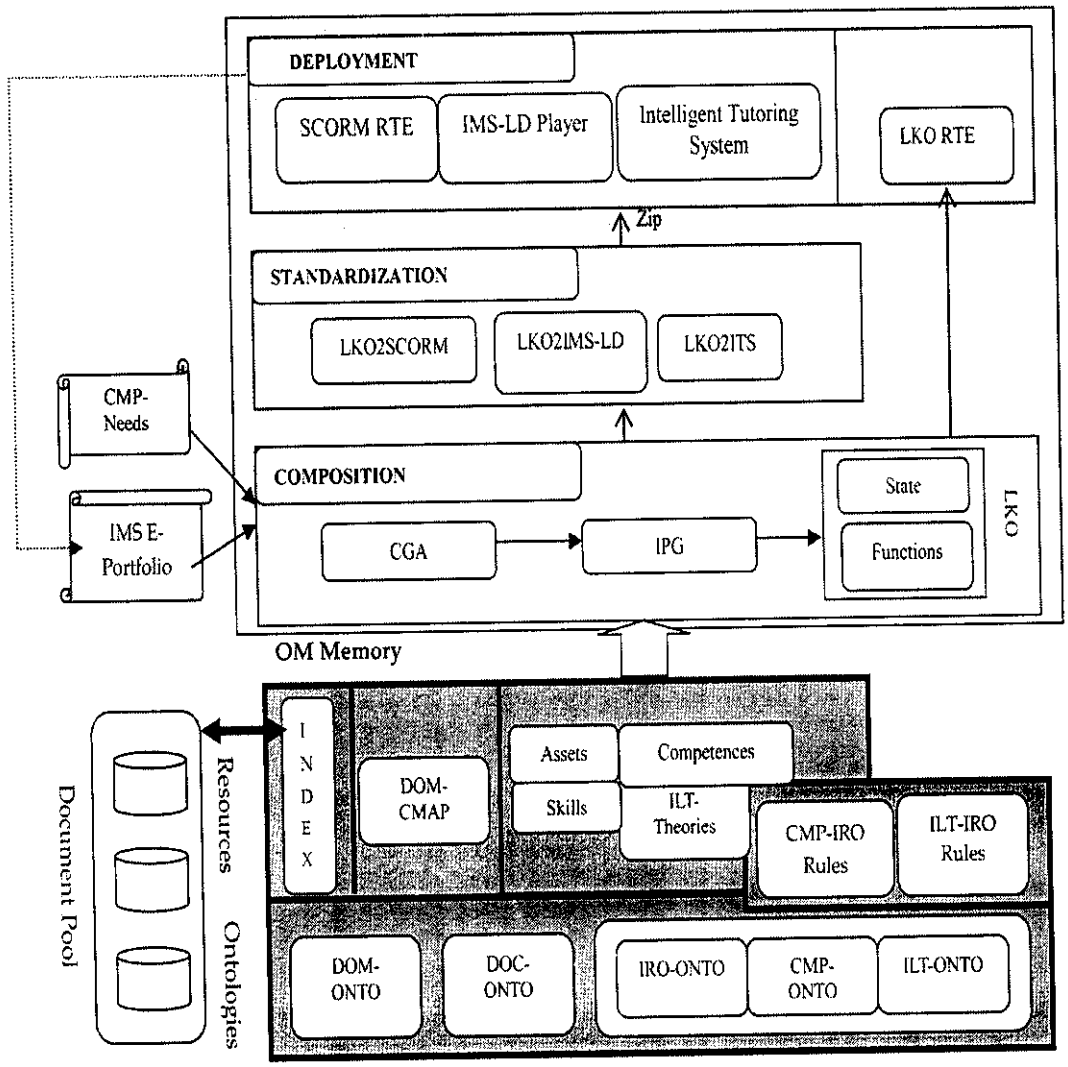


Figure 3.3 The knowledge exploitation component architecture

3.5.1 Composition Layer

This layer enables initiating the LKO's aggregation process starting from competence needs for a specific learner profile. Competence needs are expressed as an OWL file that describes the skills to acquire and the domain concepts that are concerned. The learner profile is expressed as an IMS ePortfolio [13] and converted in OWL format before being used by the composition subsystem. The Learner profile serves to store mastered competencies as well as learner's characteristics. It is an overlay model meaning that the learner knowledge is

progressively built and compared to an expert knowledge until it reaches the expert level. When learning objectives are specified, a Competence Gap Analyzer (CGA) computes an adjusted competence definition tailored to the learner's profile. In fact, for each skill in the targeted competence, the learner profile is fetched in order to find out if this skill is already mastered. If this is the case, then the skill is removed from the competence definition. The CGA then outputs a set of skills that are new to the learner in the form of an Adjusted Competence. The Adjusted Competence is passed to an Instructional Plan Generator (IPG) that is in charge of composing the actual Learning Knowledge Object. The IPG exploits all the OM's layers:

- The resource layer offers the assets that represent the basic knowledge units in an LKO.
- The ontology layer, and more particularly, the Instructional Learning Theory Ontology (ILT-ONTO) describes the instructional events and conditions that will effectively guide the composition. A specific instructional theory must then be chosen (Merrill, [16], Gagne´ [11], Ad Hoc, etc.). This enables a flexible independent way for theory incorporation in Learning Knowledge Object. It also offers an explicit pedagogical framework understandable by humans and software.
- The Rule Layer is used to connect a particular competence and skill with the more appropriate assets able to fulfill it. For example, the skill "define" would require a "Definition." It also connects the Instructional learning events with the appropriate assets. The following rule is an example which links the instructional event "Insure_retention" to the instructional role "Summary":

  InstructionalStep (Insure_retention) -> query: select (Summary).

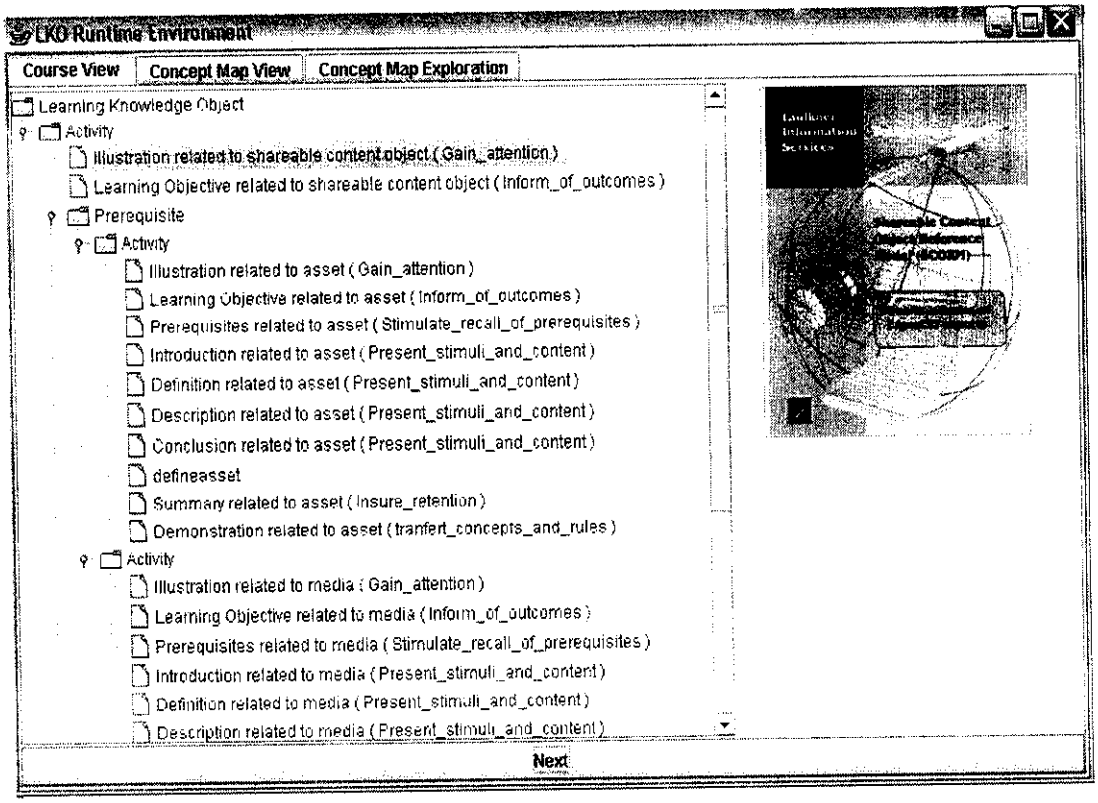


Figure 3.4 A generated learning knowledge object according to Gagne''s theory

The execution of the IPG produces a Learning Knowledge Object that can be seen as an independent object composed of a Data State and a set of functions to manipulate it (an interface). The data state is an OWL data structure formed from the various resources necessary for the LKO: the competence, the skills, the domain ontology around the concepts targeted by the skills as well as the domain concept maps and the learner profile. The LKO functions are a sort of a standard interface that enables any LKO to act as a small "Intelligent Tutoring System."

This standard interface offers the following functions:

- Scenario Control to guide the learner's progression through the LKO content,
- Evaluation of learner's actions and exercises,
- Learner's e-Portfolio update,
- Domain Ontology and Concept Maps Exploration,
- Explanation of different concepts by their context,

and

- Automatic Generation of LKOs related to the concept's context.

The added value of the LKO when compared to existing learning objects relies on its inner characteristics: an LKO acts as an independent small tutoring system, it has a domain model, it is guided by an instructional theory, and it possesses an interface to act on its data and to provide an individualized training. An LKO does not need to be stored as a whole in a learning object repository. Since the aim of the organizational memory is to conserve only sound reusable pedagogical fragments, and not whole learning objects, an LKO can be generated by using these resources and a particular theory. In this point of view, learning objects must not exist as fixed static content packages. Semantic services coupled with instructional roles should be the new paradigms that sustain learning object generation. To summarize,

- An LKO is an independent object: it is implemented as a software package (an applet) that receives, at runtime, an on-the-fly generated OWL file able to fulfill a specific competence for a specific learner.
- An LKO possesses pedagogical knowledge in the form of a scenario plan generated according to an instructional theory. This theory (Gagne', Merrill, etc.) is also chosen right at the time of the aggregation process. The same content can be reused to produce another LKO compliant with another instructional theory. So the reusability dimension is not really related to the LKO itself but to the independent knowledge fragments (the instructional roles) that are stored in the OM. The composition service can then generate the same LKO as often as needed.

In ***Figure 3.4***, can see the generated LKO for this competence according to the Gagne' Theory, can also notice that two prerequisites are added to the LKO related to the skills "define asset" and "define media." Hence, the resulting LKO is composed of three skills, each skill being taught according to the Gagne' Theory. These prerequisites are added to the definition of the competence during the competence gap analysis. Once the adjusted competence is available, and once an instructional theory is chosen, the IPG loads the instructional events related to the theory (here the Gagne' Theory). Since each instructional event is linked to an SWRL rule, the IPG executes these rules in order to gather the different assets that are required to fulfill each instructional event. The rule engine Jess is used to run the SWRL rules. As a result,

a course structure compliant with the chosen theory is generated and linked to adequate instructional roles.

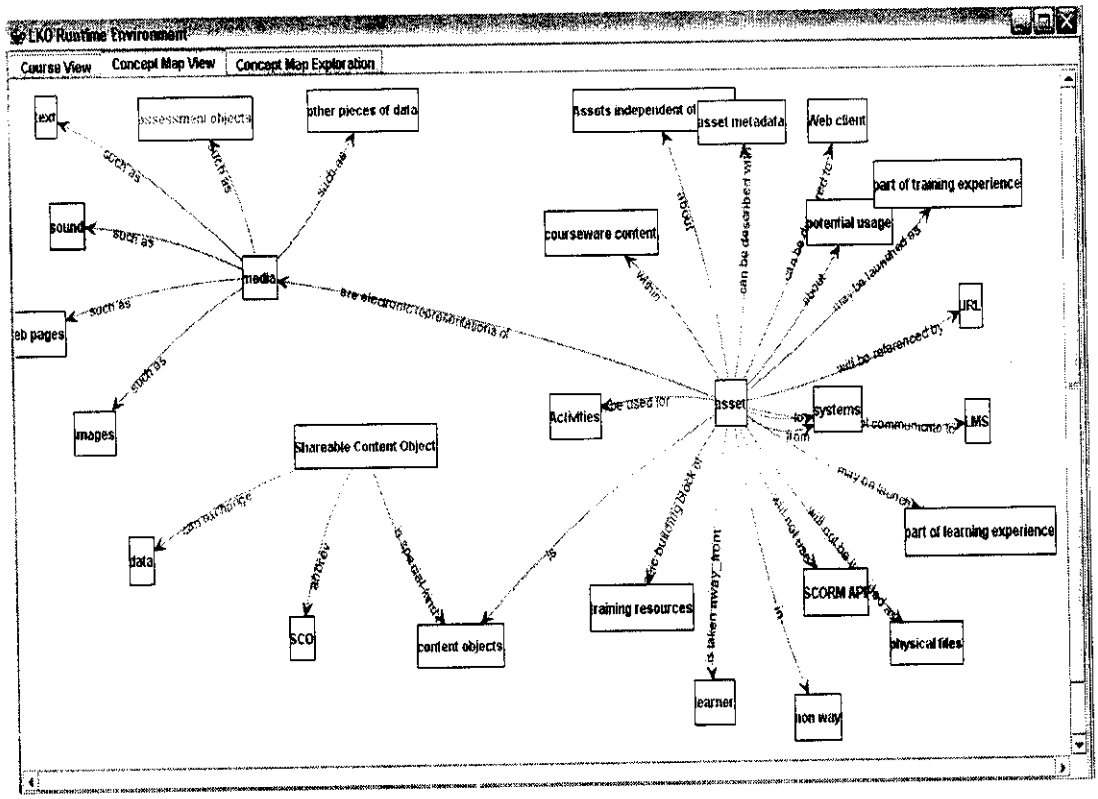


Figure 3.5 The concept Map View in the current LKO

3.5.2 Standardization Layer

The standardization layer serves as an interface to the different standards and usable environments for an LKO. It comprises the following:

- A SCORM LKO Generator (LKO2SCORM) that generates a SCORM-compliant content package. It encapsulates the LKO applet into a standard SCORM template.
- An IMS-LD LKO Generator (LKO2IMS-LD) that generates an IMS-LD conformant content package.
- An Intelligent Tutoring System LKO Generator (LKO2ITS).

The standardization layer exports, when needed, a zip file to any type of training environment.

3.5.3 Deployment Layer

The LKOs are targeted toward any kind of training environment. The Knowledge Puzzle produces LKOs deployable in a SCORM Runtime Environment, in an IM LD player or in any intelligent tutoring system. It also provide an LKO Runtime Environment (LKORTE) for users that do not have access to an ITS or that do not want to comply with a particular e-Learning standard. The LKO-RTE makes it possible to run the LKO as a standalone resource. The user interface gives access to relevant functions that are implemented within the LKO to support a variety of learning services, including possible access and exploration of the concept map around the LKO's concepts (***Figure 3.5***), thus fostering meaningful learning [14].

3.6 MODULE DESCRIPTION

The project steps are following:

1) First, promotes the state-of-the-art on LOs sharing and reusing issues.
2) Second, construct the structure of the OM followed by the explanation of the capitalization process (the process that aims to feed the OM).
3) Third, the OM is exploited for LKOs composition and how resulting LKOs are deployed in standard and nonstandard learning environments.
4) Finally, ILLUSTRATING the value of the approach through a semantic evaluation and some results will be obtained before concluding the project.

CHAPTER 4

IMPLEMENTATION DETAILS

4.1 SYSTEM REQUIREMENTS:

4.1.1 Hardware Requirements:

- PROCESSOR : PENTIUM III 866 MHz
- RAM : 128 MB DD RAM
- MONITOR : 15" COLOR
- HARD DISK : 20 GB
- FLOPPY DRIVE : 1.44 MB
- CDDRIVE : LG 52X
- KEYBOARD : STANDARD 102 KEYS
- MOUSE : 3 BUTTONS

4.1.2 Software Requirements:

- LANGUAGE : JAVA
- FRONT-END TOOL : SWING, PROTEGE TOOL
- OPERATING SYSTEM : WINDOWS-VISTA

4.2 PROTÉGÉ TOOL

Protégé is a tool which allows a user to construct domain ontology, customize data entry forms and enter data. The tool can be easily extended to access other knowledge based embedded applications. For example, Graphical widgets can be added for tables and diagrams. Protégé can also be used by other applications to access the data. Protégé allows a user to simultaneously work on classes and instances. This is provided for by a uniform GUI whose top level is composed of overlapping tabs for compact representation.

'Classes' tab is used to define classes and the class hierarchy, slot and slot-value restrictions, relationships between classes and properties of these relationships. The 'Instances' tab can be used to acquire instances of classes defined in the ontology. The forms are used for acquiring instances based on the type of slots that you have specified. The default form can then be changes by rearranging the fields on the screen, changing the size, label and properties for a slot.

The main assumption of Protégé-2000 is that knowledge-based systems are usually very expensive to build and maintain. For example, the expectation is that knowledge-based system development is a team effort, including both developers and domain experts who may have less familiarity with computer software. Protégé-2000 is designed to guide developers and domain experts through the process of system development. Protégé-2000 is designed to allow developers to reuse domain ontologies and problem-solving methods, thereby shortening the time needed for development and program maintenance. Several applications can use the same domain ontology to solve different problems, and the same problem-solving method can be used with different ontologies.

4.2.1 Basic Features

- Import format
    - XML, RDF(S) and XML Schema
- Export format
    - XML, RDF(S), XML Schema, FLogic, CLIPS and Java HTML
- Graph view
    - Via GraphViz plug-in (browsing of classes and global properties)
    - Via Jambalaya plug-in (nested graph view)
- Consistency check
    - Via plug-ins (PAL and FaCT)
- Limited multi-user support
    - Protégé 2.0 has new multi-user capabilities added to it. It is intended for experienced Protégé users. Multiple users can read the same database and make incremental changes or changes that don't conflict with one another.

However, there's no support for multiple users trying to modify the same elements of a knowledge base or notification of changes made by other users. Concurrent changes to the same section will cause severe problems.

- Web support
    - Via Protégé-OWL plug-in
    - Protégé doesn't provide direct support for accessing knowledge base from the web, but it can easily be done. A number of our users have communicated with Protégé knowledge bases from the Web via servlets. Protégé can be run as an applet. The new release of the Protégé Web Browser is now available. It can be downloaded from the Protégé plug-ins page. The Protégé Web Browser allows users to browse Protégé ontologies and knowledge bases in their web browser, without having to install the Protégé application locally.

4.2.2 Additional Features

- Merging
    - Via Anchor-PROMPT plug-in
- Not support to add a new basic type
- Extensible plug-in architecture
- Ontology storage
    - File and DBMS(JDBC)

4.3 OWL LANGUAGE

OWL (Web Ontology Language) builds on RDF and enables more sophisticated semantics to be defined; it is currently the focus of much work and attention. Its development arose from the need for machine-readable descriptions of the meaning of terminology used in Web pages, to improve the "poorly mapped geography" of the Web so that computational agents can use it more effectively. It has been driven by the following goals:

- To make ontologies publicly available so that their semantics can be shared.
- To provide a consistent approach to ontology revision.

- To enable ontologies which model concepts in different ways to be mapped.
- To detect inconsistencies between ontologies.
- To express different kinds of knowledge, and be able to reason with it.
- To be easy to learn, encouraging adoption.
- To be compatible with other standards.
- To support multilingual ontologies and cross-cultural views.

OWL has recently become a World Wide Web Consortium Recommendation, and these goals are reflective of the current issues and concerns facing ontology developers as a whole. It provides three sub-languages, intended to meet different users' needs:

- OWL Lite can provide a class hierarchy together with some simple constraints.
- OWL DL is based on Description Logic formalisms which allow for powerful expression. The ontology can be reasoned but a "reasoner" is required.
- OWL Full enables maximum expressiveness, without regard for computational limits.

OWL is the product of much prior work, and is based on another ontology language called **DAML+OIL**, developed by a joint EU-USA, DARPA-sponsored ad hoc committee. Built on RDF(S), DAML+OIL is a markup language which allows semantics to be expressed in XML. It, in turn, supersedes **DAML+ONT**, and both are largely based on OIL.

OIL (Ontology Inference Layer) provides semantics for describing term meanings. It comprises four levels, each one more expressive than the last. Programs capable of only processing the core layer are still compatible with ontologies expressed in higher layers.

Other Web ontology languages to note include:

- **SHOE** (Simple HTML Ontology Extensions), an extension of HTML with tags for incorporating semantic knowledge into documents. Based on an ISA hierarchy, it is very simple but less expressive than some other specifications.
- **OML** (Ontology Markup Language) and **CKML** (Conceptual Knowledge Markup Language) adapt SHOE to XML and are based on Conceptual Graphs. OML was later redesigned to be RDF compatible.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 LOADING DATASETS

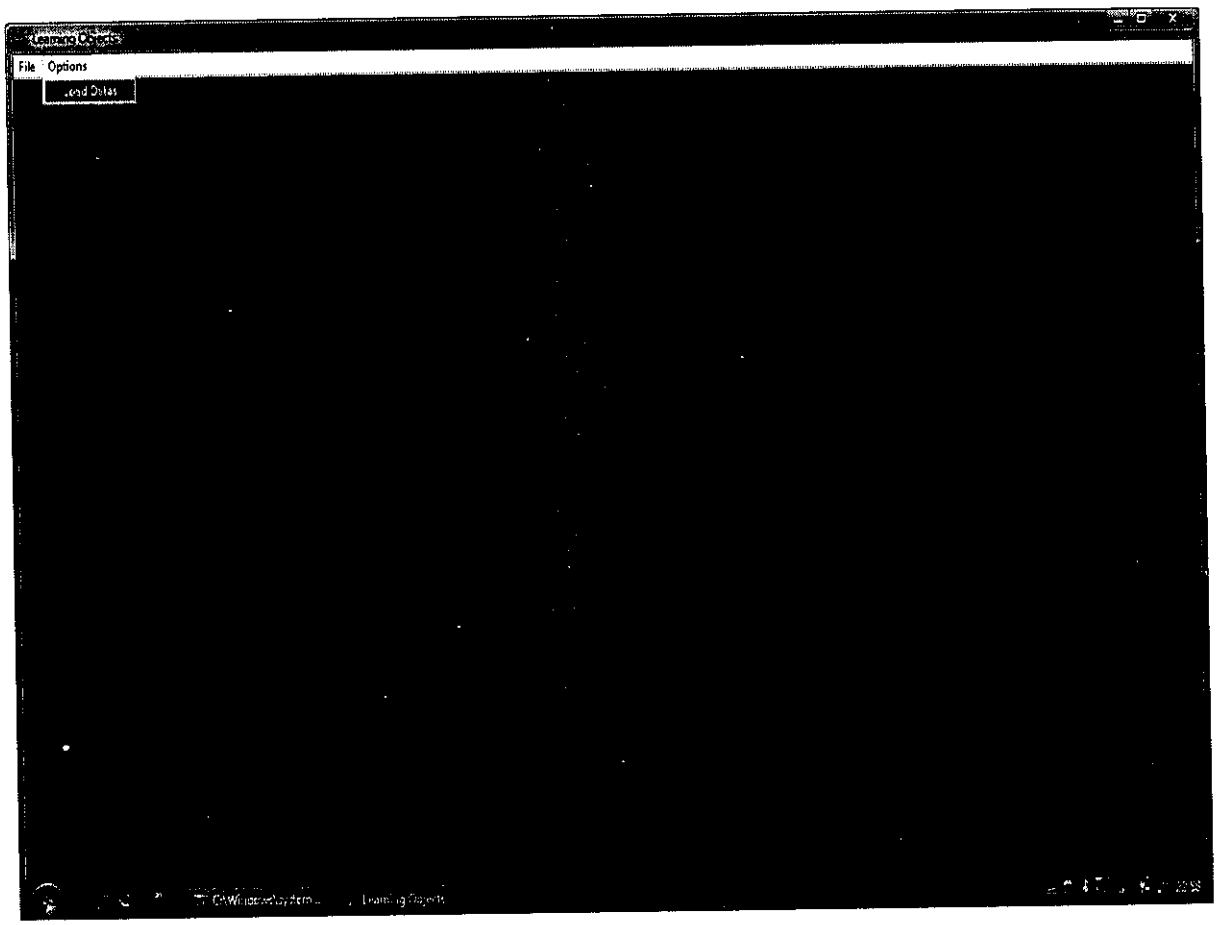


Figure 5.1 Loading Datasets

The collected datasets are loaded into source file

5.2 DATA EXTRACTION

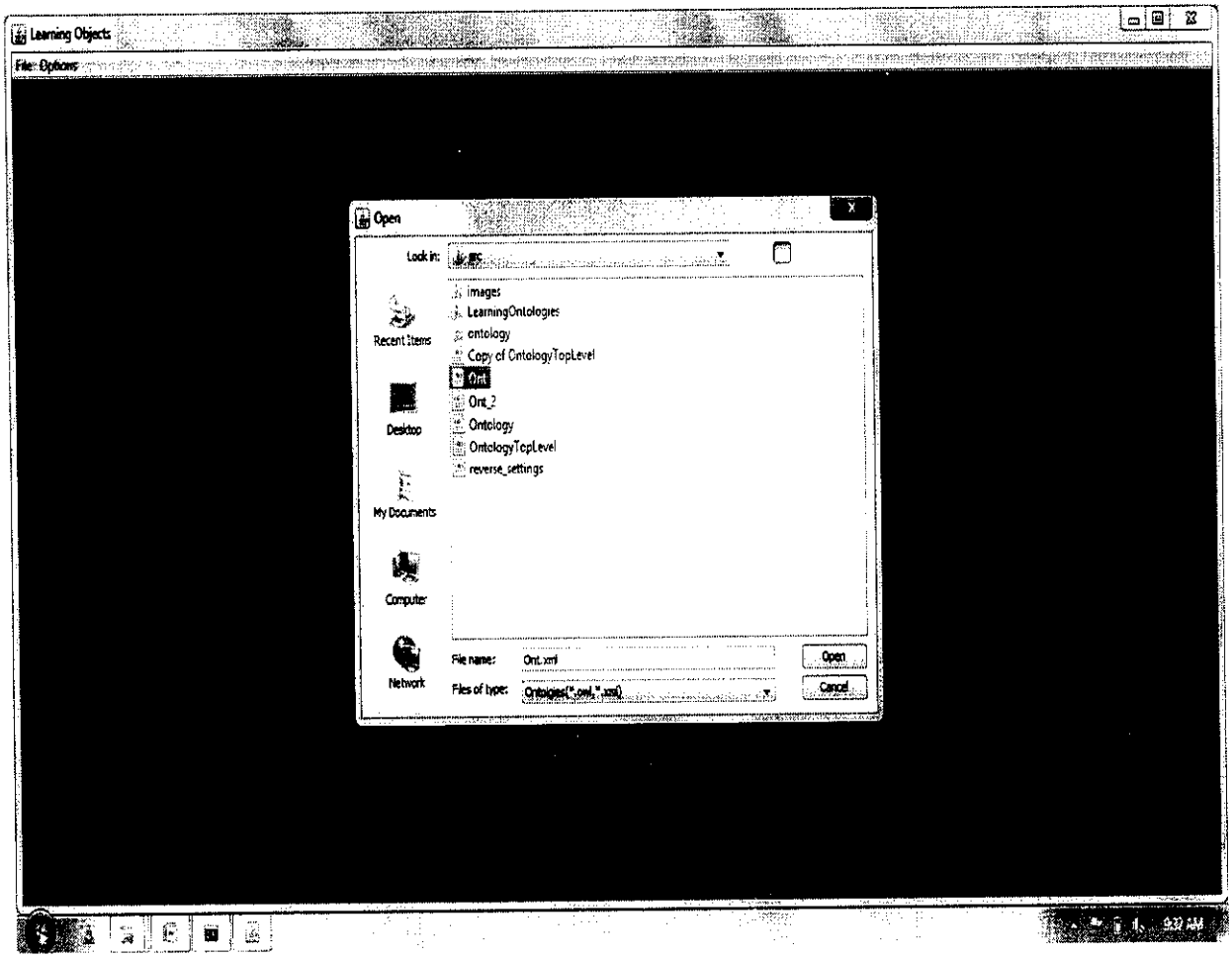


Figure 5.2 Domain Extraction

From the available datasets, a particular domain is selected to view the Hierarchical structure.

5.3 TREE VIEW OF DOMAINS

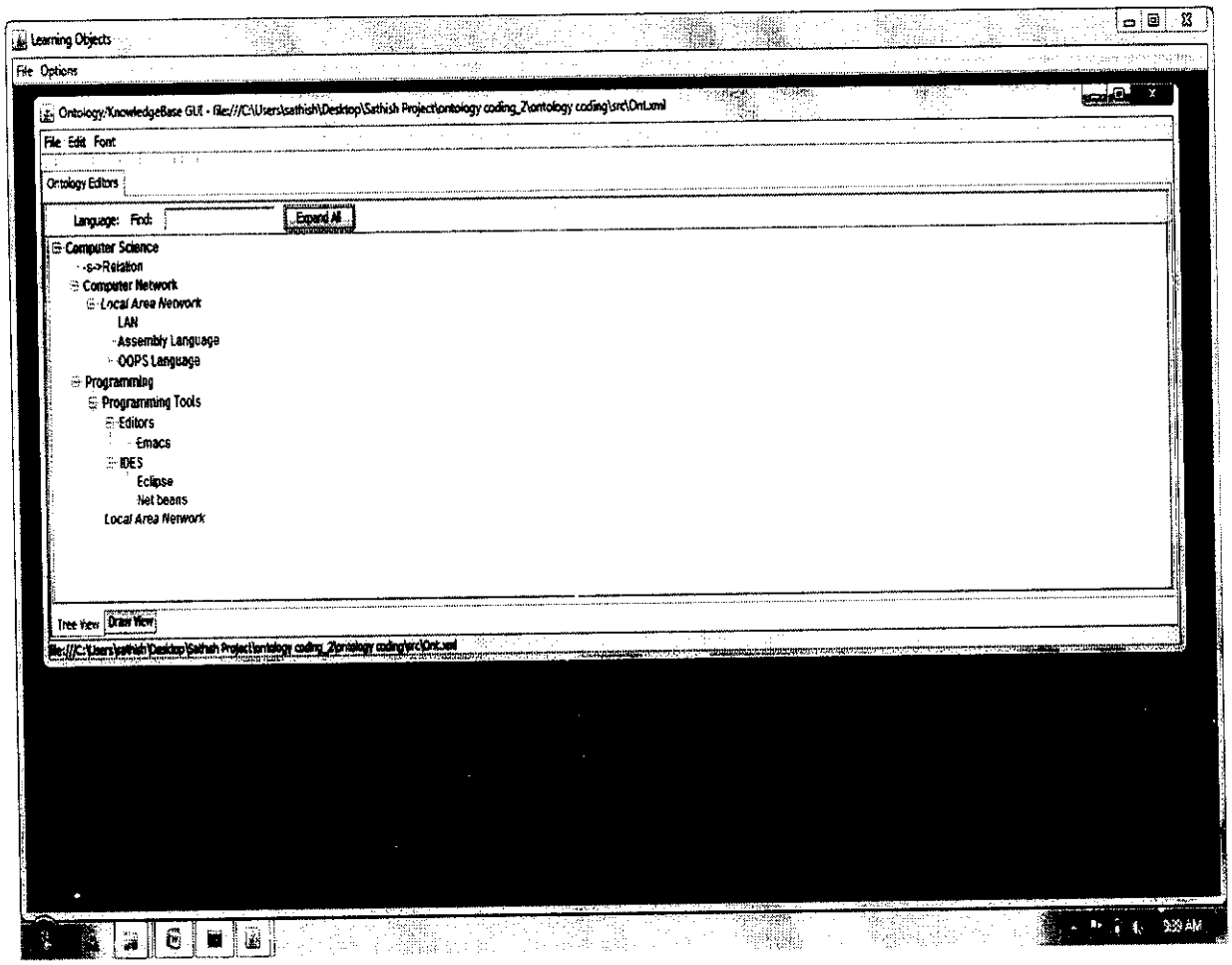


Figure 5.3 Hierarchical View (Or) Tree View of Domains

This figure depicts the hierarchical view of the domain. For instance, here the computer science is the domain chosen by the learner. Here the user can view the subsets of the domain computer science.

5.4 SEARCHING SUB CONCEPT

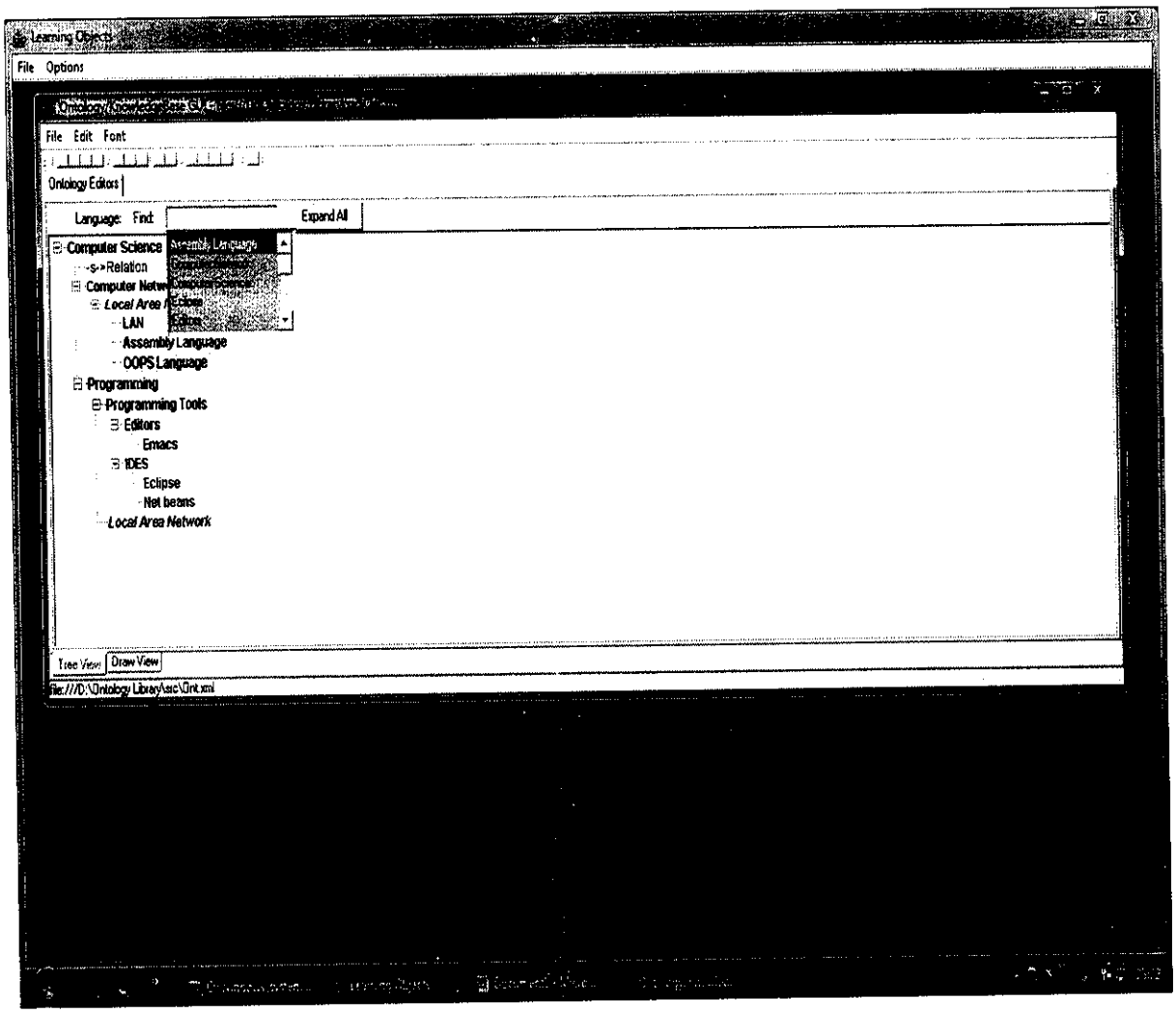


Figure 5.4 Searching Sub Concept under the Domain (Computer Science)

The topic given under the search engine is displayed. This makes the learners to view their sub concepts under the particular domains in easy manner. For instance, here the sub concept assembly language is selected under the domain i.e. computer science.

5.5 VIEWING THE LEARNING OBJECTS

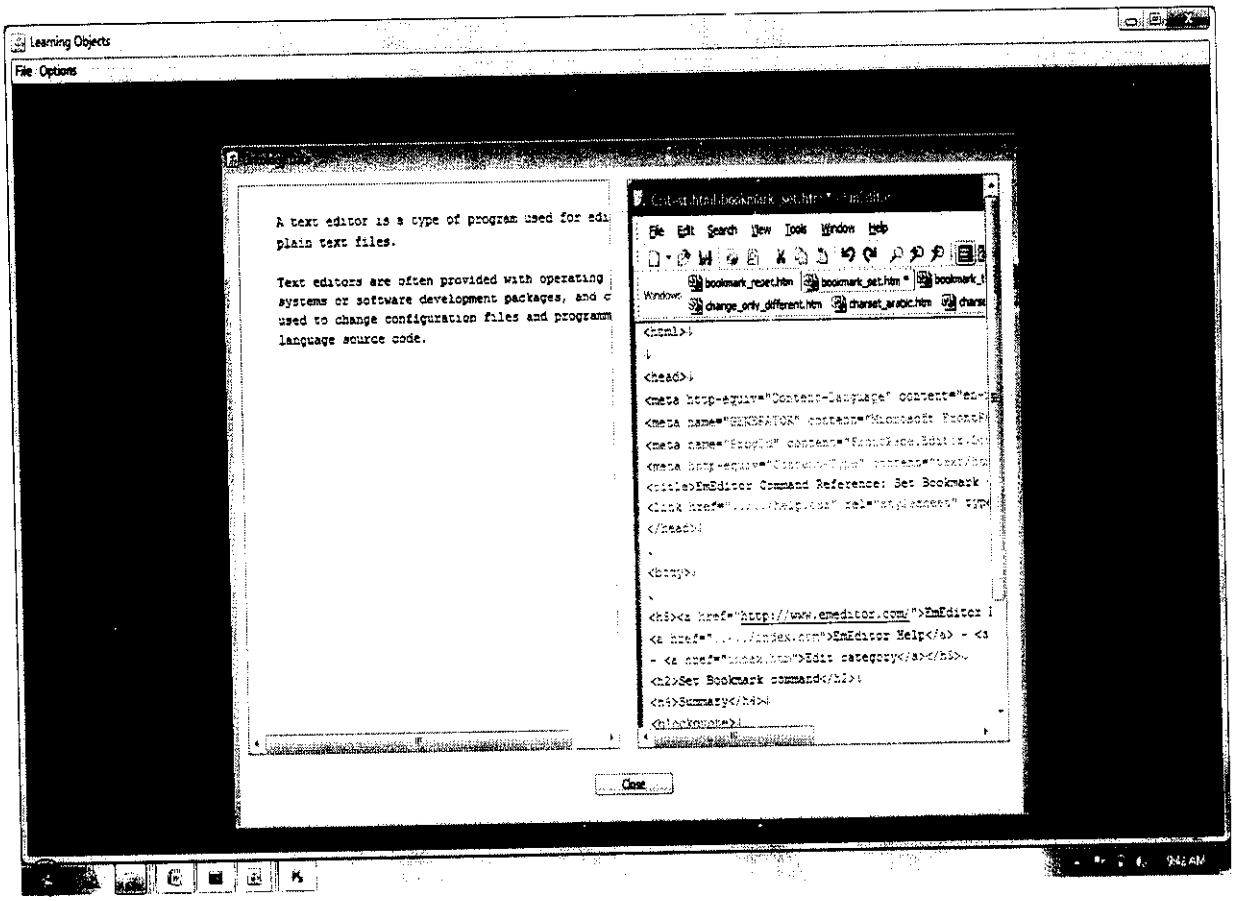


Figure 5.5 Viewing the Learning Objects or Learning Materials

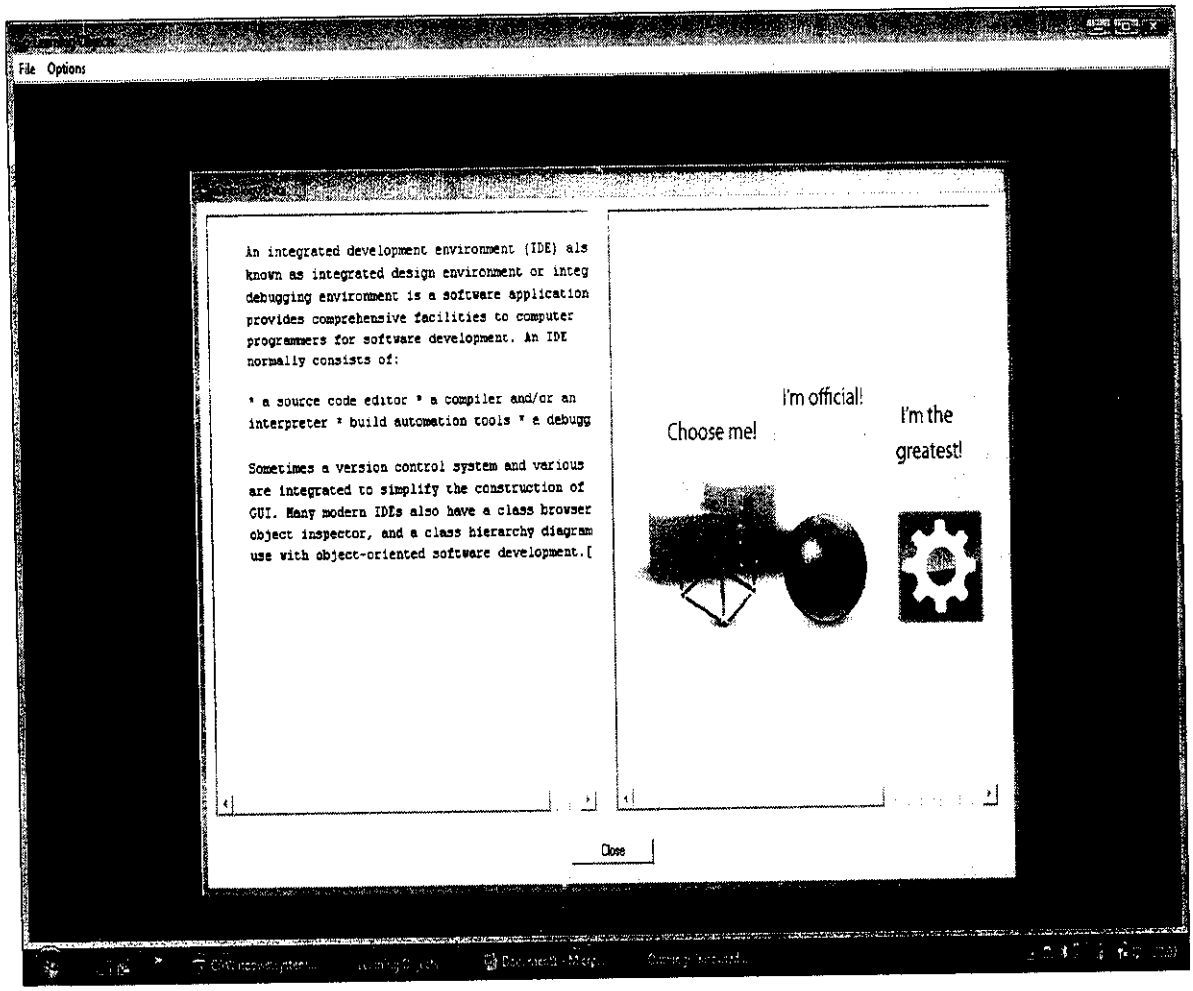


Figure 5.6 Viewing Next Learning Objects or Learning Materials

5.6 CREATION OF CONCEPT MAP

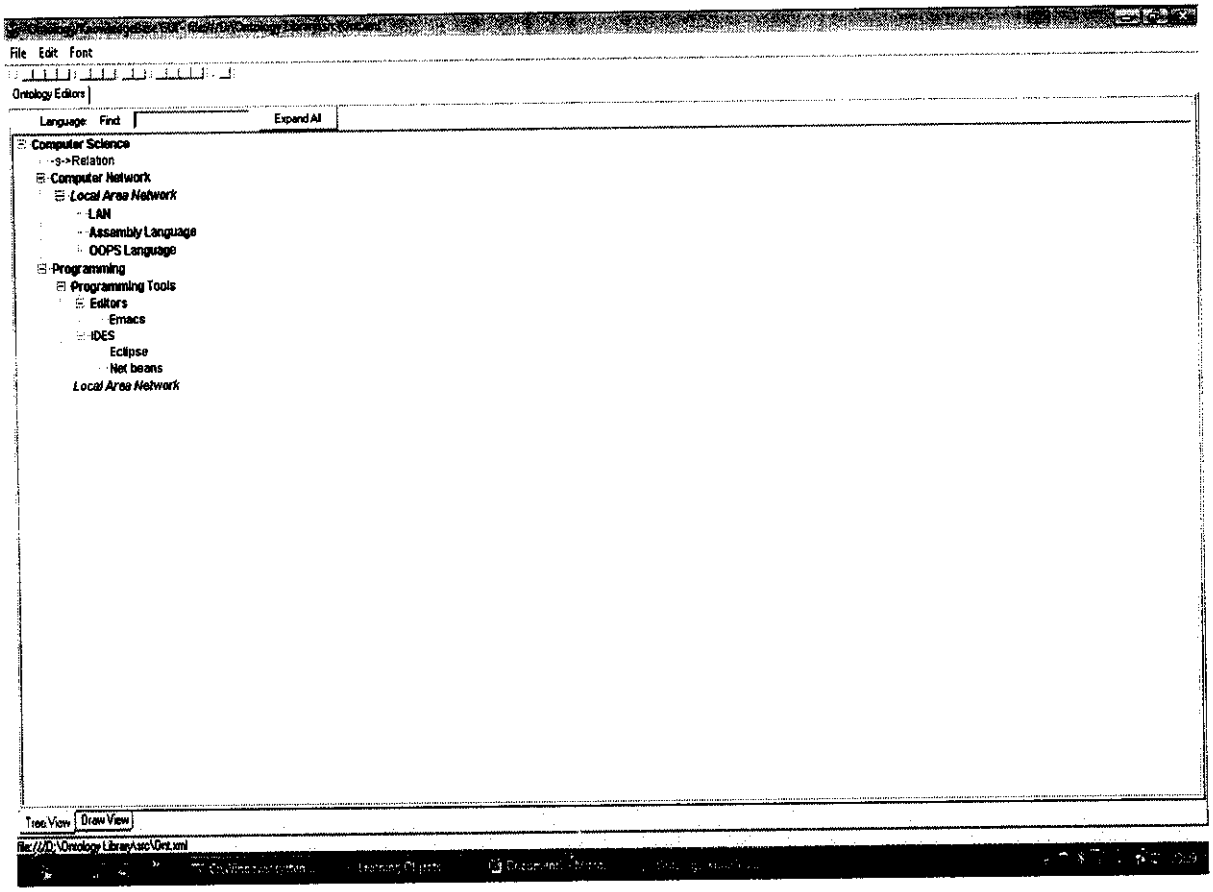


Figure 5.7 Creation of Concept Map

The main idea for designing concept map is to analyze the related links for a particular domain with its sub concepts.

5.7 VIEWING THE CONCEPT MAP

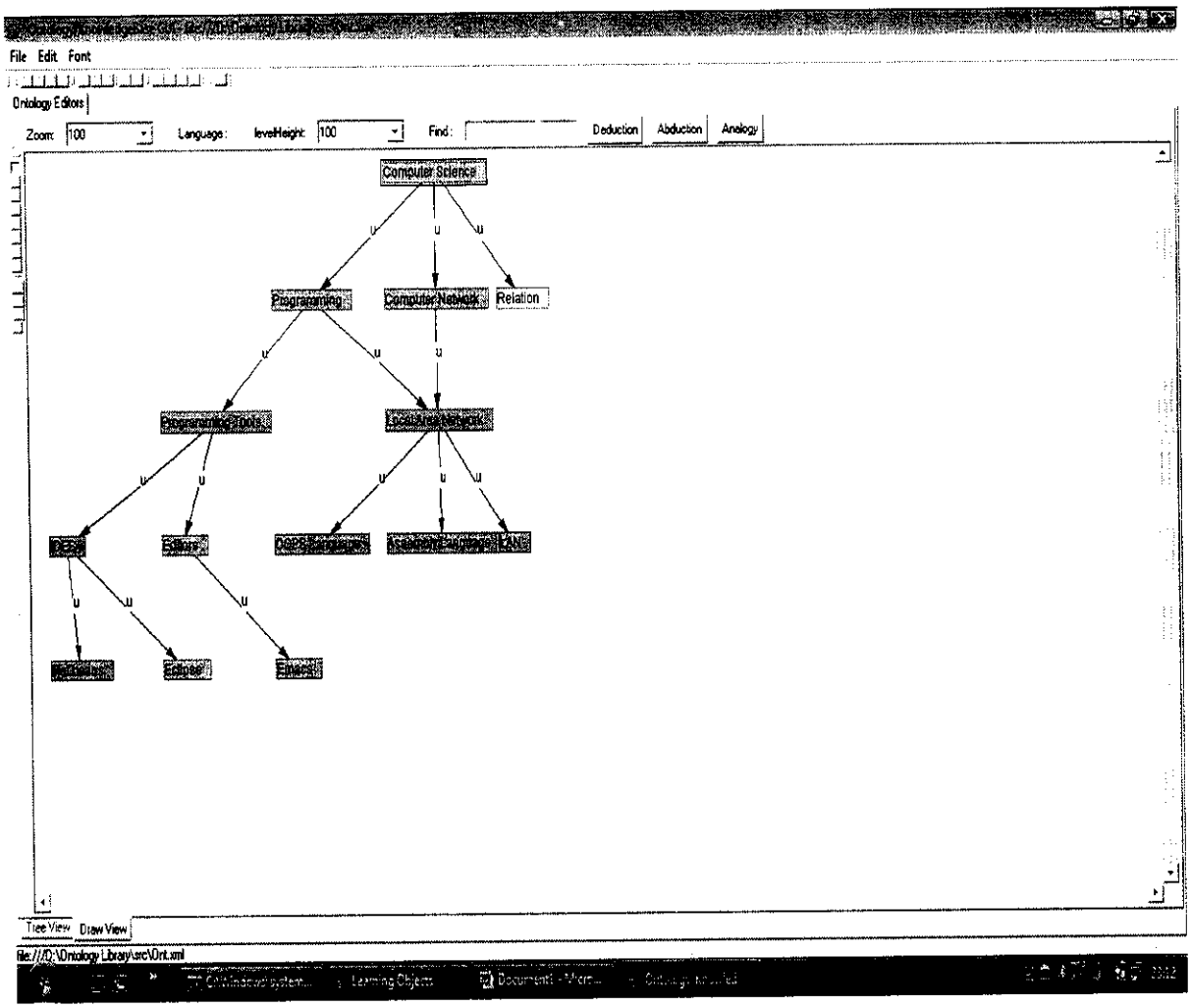


Figure 5.8 Viewing the Concept Map

The above screen shot depicts, the related links for a particular domain (computer science)

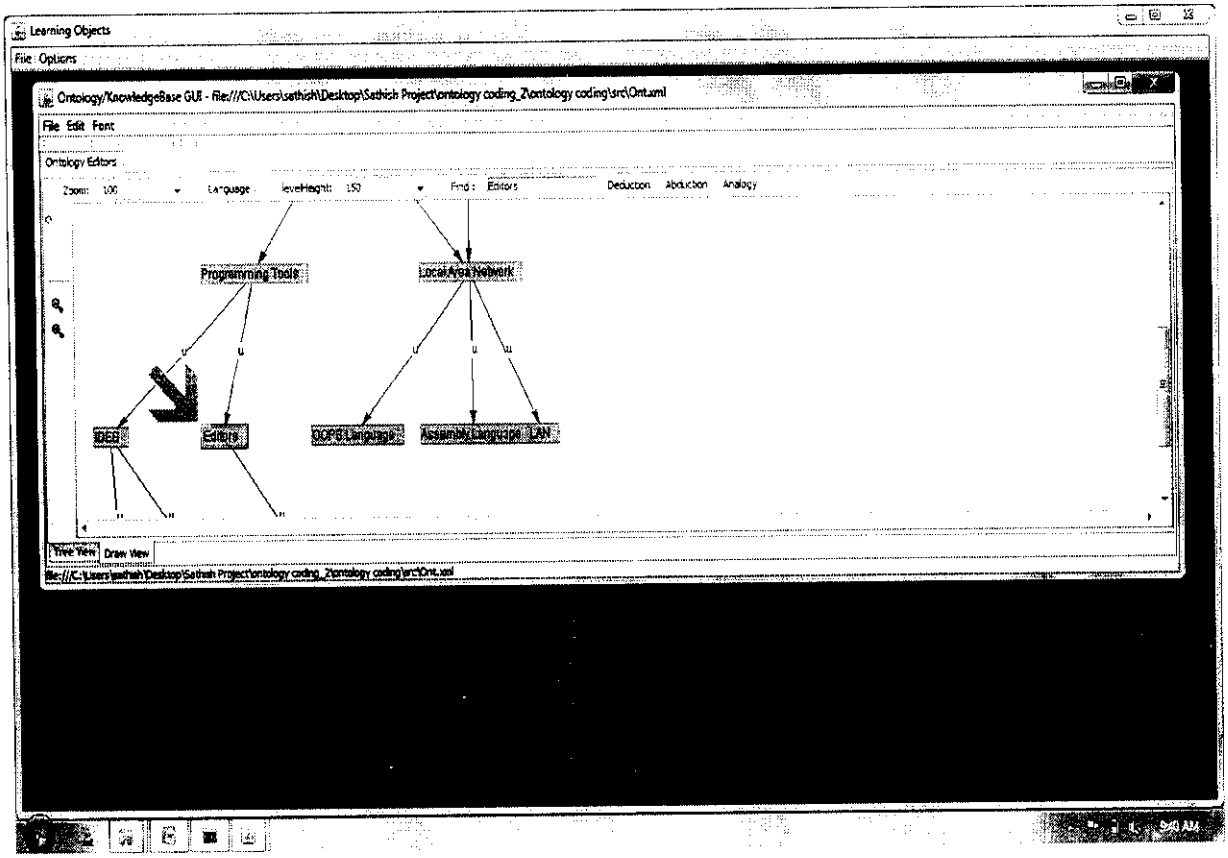


Figure 5.9 Highlighting the sub concepts or domain that is given in the search engine

The above screen shot highlights the related sub concepts for the domain that is given in the search engine. For instance, local area network is given in search engine and it is been highlighted.

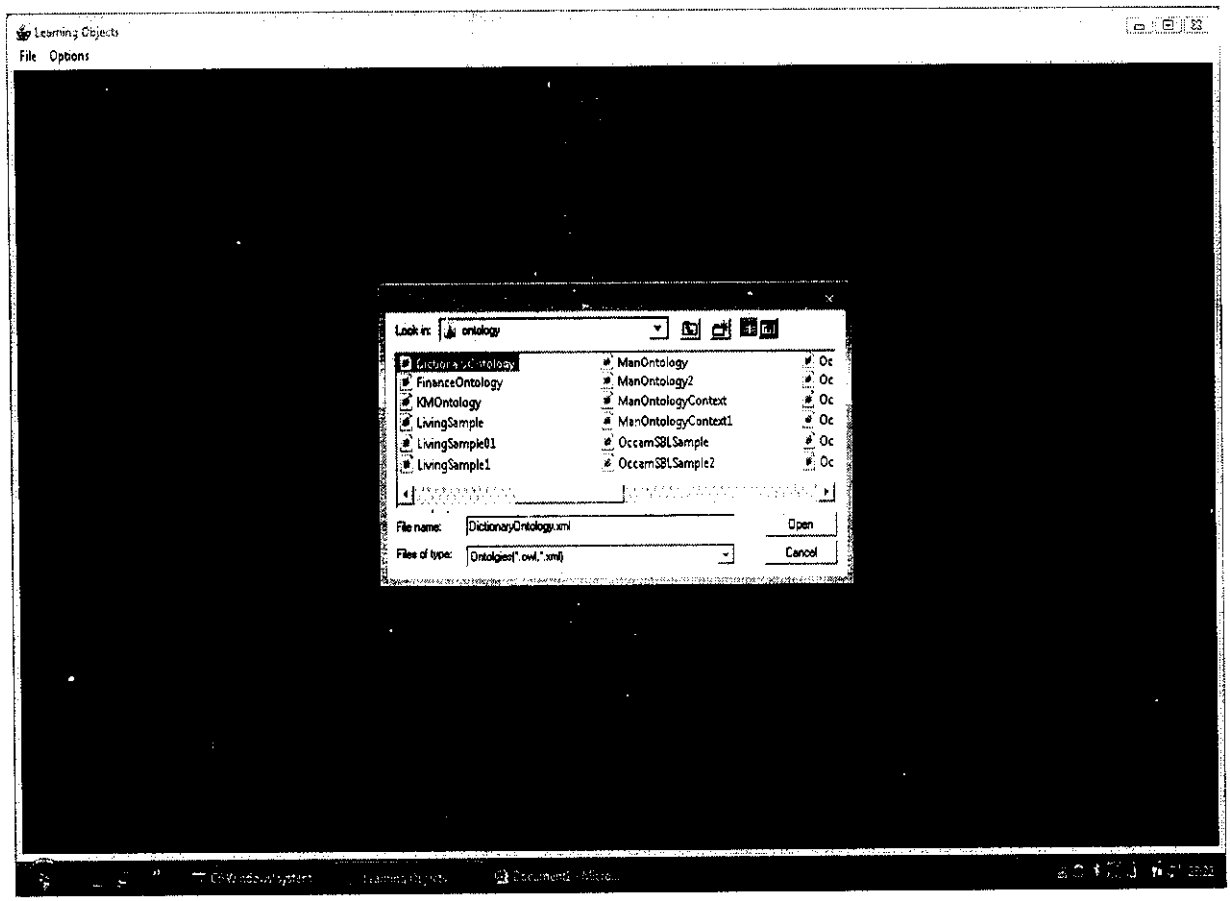


Figure 5.10 Domain Extraction

The above process is repeated when we choose other dataset of different domain or field

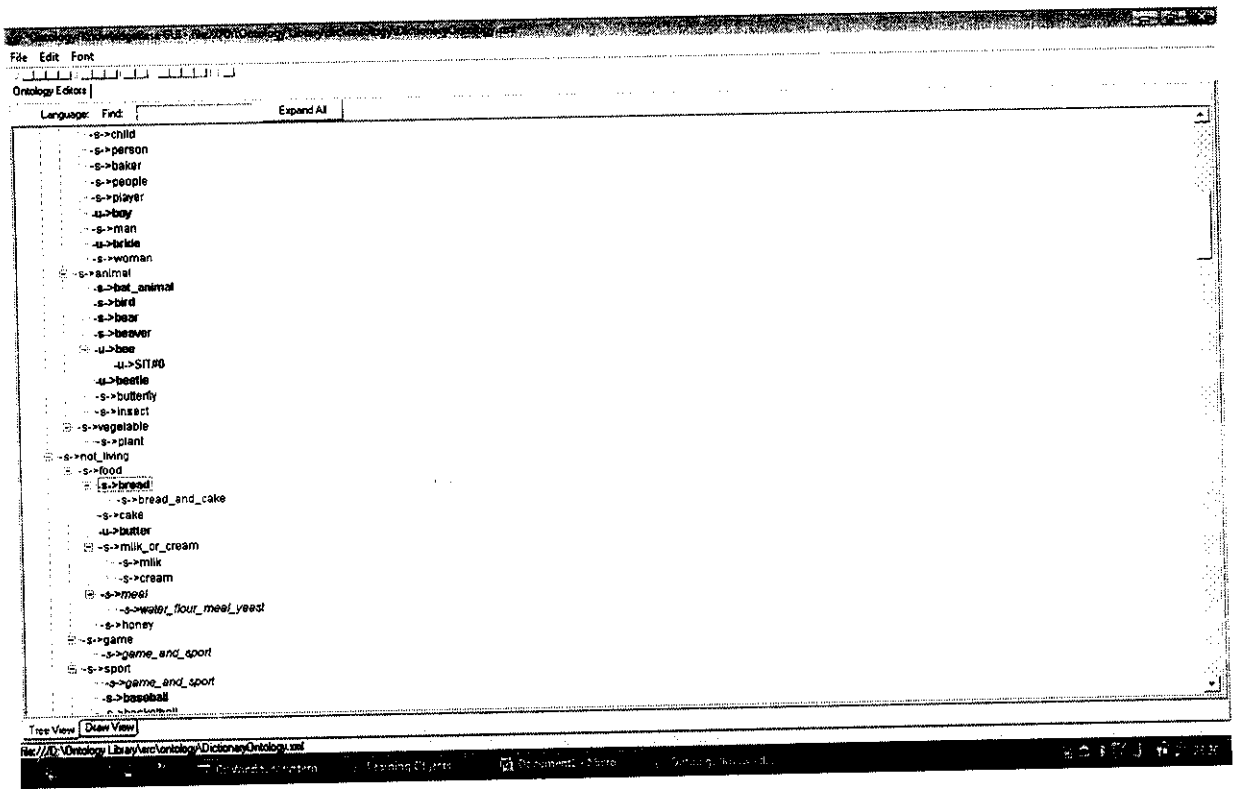


Figure 5.11 Hierarchical View (Or) Tree View of Domain

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

The Knowledge Puzzle production subsystem enables the automatic domain ontology generation from learning object content through TEXCOMON, and how annotation and edition of assets, rules, competences, and instructional roles are performed through ONTO-AUTHOR. This gives an OM's content that the Knowledge Puzzle Exploitation Subsystem can exploit for the on-the-fly generation of active, independent, reusable, and theory-aware learning knowledge objects. The LKOs can then be used in any training environment, including standards platforms and intelligent tutoring systems. This solutions contrast with current practices in the e-Learning area and represent a way to bridge the gap between classical e-Learning systems and intelligent tutoring systems.

6.2 FUTURE ENHANCEMENTS

The LKO concept offers a great opportunity to go beyond the actual limited view of LOs by allowing the exploration of their content and making the integration of pedagogical knowledge more flexible in a given learning context. In this project also performed a semantic evaluation based on human experts and compared the tool TEXCOMON with Text-To-Onto, one of the state-of-the-art tools in domain ontology learning from text. This evaluation showed improved extraction results with our tool.

In the production part, text mining and pattern extraction can be enhanced in order to decrease the generated noise. Furthermore, integration of more complex pedagogical scenarios and instructional theories must be undertaken to enrich the composition process.

APPENDIX

SAMPLE CODING

```
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;
import javax.swing.JOptionPane;
public class HierarchyWndCG extends Hashtable<Object, Object> {
private static final long serialVersionUID = -5251388973128935985L;
public MainFrame mainFrame;
public GraphDrawFrame WndRacine = null;
public GraphDrawFrame WndActive = null;
public HierarchyWndCG(MainFrame f) {
super(10);
mainFrame = f;
}
public GraphDrawFrame getRoot() {
return WndRacine;
}
public void setRoot(GraphDrawFrame wd) {
Vector<GraphDrawFrame> wndPerFils = new Vector<GraphDrawFrame>(5, 2);
```

```
wndPerFils.addElement(wd);
put(wd, wndPerFils);
WndRacine = wd;
wndPerFils = null;
}
public void newChild(GraphDrawFrame wdPere, GraphDrawFrame wdFils) {
Vector<GraphDrawFrame> vect = (Vector<GraphDrawFrame>) get(wdPere);
if (vect != null)
vect.addElement(wdFils);
vect = new Vector<GraphDrawFrame>(5, 2);
vect.addElement(wdPere);
put(wdFils, vect);
vect = null;
}
public GraphDrawFrame findFrameForCG(Graph graph) {
GraphDrawFrame wdGC = null;
for (Enumeration e = keys(); e.hasMoreElements();) {
wdGC = (GraphDrawFrame) e.nextElement();
if (wdGC.getGraphDrawPanel().vgraph.getGraph() == graph)
return wdGC;
}
```

```
return null;
}
public void repaintAllActiveWnd() {
GraphDrawFrame wdGC;
for (Enumeration e = keys(); e.hasMoreElements();) {
wdGC = (GraphDrawFrame) e.nextElement();
if (!wdGC.isIcon()) {
wdGC.getGraphDrawPanel().repaint();
}
}
}
public void clear(boolean all) {
closeFrame(WndRacine, all, false);
}
public GraphDrawFrame getFatherFrame(GraphDrawFrame wdGC) {
GraphDrawFrame father = null;
for (Enumeration<Object> e = this.keys(); e.hasMoreElements() && father == null;) {
GraphDrawFrame wd = (GraphDrawFrame) e.nextElement();
Vector wdChild = (Vector) this.get(wd);
if (wdChild.contains(wdGC))
father = wd;
```

```
}
return father;
}
public void closeFrame(GraphDrawFrame wdGC, boolean all,
boolean bConsultUser) {
if (WndRacine == null)
return;
int Rep = 0;
if (bConsultUser)
Rep = JOptionPane.showConfirmDialog(wdGC,
"Do you want the children frames to be closed ?",
"Question", JOptionPane.YES_NO_OPTION);
if (!bConsultUser || Rep == JOptionPane.YES_OPTION) {
Vector vWndPrFils = (Vector) this.get(wdGC);
if (vWndPrFils != null && vWndPrFils.size() > 1) {
Enumeration e = vWndPrFils.elements();
for (GraphDrawFrame wd = (GraphDrawFrame) e.nextElement(); e
.hasMoreElements();) {
wd = (GraphDrawFrame) e.nextElement();
closeChildrenFrames(wd);
}
```

```
vWndPrFils.clear();
}
vWndPrFils = null;
}
remove(wdGC);
if (wdGC == WndRacine) {
if (mainFrame instanceof CGDrawingFrame) {
int result = JOptionPane.showConfirmDialog(null,
"Save before closing ? ", "Warning",
JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE);
if (result == JOptionPane.YES_OPTION)
mainFrame.File_Save();
}
clear();
/*****************************************************************
* CG gc = null; if (WndRacine.getGraphDrawPanel().vgraph != null)
* gc = (CG) WndRacine.getGraphDrawPanel().vgraph.getGraph();
*
* if (gc!=null) gc.clear();
*****************************************************************/
if (all) {
```

```
WndActive = null;
WndRacine = null;
wdGC.dispose();
} else {
this.setRoot(WndRacine);
WndActive = WndRacine;
}
mainFrame.updateMenuItems1And2(true);
mainFrame.updateMenuItems();
} else {
WndActive = this.getFatherFrame(wdGC);
wdGC.dispose();
}
}
void closeChildrenFrames(GraphDrawFrame wd) {
Vector vWndPrFils = (Vector) get(wd);
try {
Enumeration e = vWndPrFils.elements();
for (GraphDrawFrame wd1 = (GraphDrawFrame) e.nextElement(); e
.hasMoreElements();) {
wd1 = (GraphDrawFrame) e.nextElement();
```

```
closeChildrenFrames(wd1);
}
vWndPrFils.clear();
vWndPrFils = null;
remove(wd);
wd.dispose();
} catch (Exception exc) {
}
}
}
```

ONTOLOGY MAIN

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.util.Hashtable;
import javax.swing.JDesktopPane;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
```

```
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.UIManager;
import javax.swing.filechooser.FileFilter;
public class OntologyMain extends JFrame implements ActionListener {
private static final long serialVersionUID = -2244318580580158356L;
static JDesktopPane deskView;
static Hashtable<String, String> leanHash;
static Hashtable<String, BufferedImage> leanImageHash;
public OntologyMain() {
toInitialize();
setJMenuBar(addMenubar());
setDefaultCloseOperation(EXIT_ON_CLOSE);
Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
setSize(dimension.width, dimension.height - 30);
setContentPane(deskView);
setTitle("Learning Objects");
setVisible(true);
}
private void toInitialize() {
```

```
try {
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
} catch (Exception e) {
e.printStackTrace();
}
deskView = new JDesktopPane();
leanHash = new Hashtable<String, String>();
leanImageHash = new Hashtable<String, BufferedImage>();
}
public void actionPerformed(ActionEvent ae) {
if (ae.getActionCommand() == "Exit") {
System.exit(0);
}
if (ae.getActionCommand() == "Load Datas") {
JFileChooser chooser = new JFileChooser(".");
chooser.setFileFilter(new FileFilter() {
public boolean accept(File f) {
String fileName = f.getName();
if (f.isDirectory() || fileName.endsWith("xml")
|| fileName.endsWith("owl"))
return true;
```

```
else
return false;
}
public String getDescription() {
return "Ontolgies(*.owl,*.xml)";
}
});
chooser.showOpenDialog(this);
if (chooser.getSelectedFile() != null) {
try {
new OntologyKBGUIFrame(Ontology
.loadOntologyFromXML("file:///"
+ chooser.getSelectedFile().getAbsolutePath(), CG.cgCSDescr));
} catch (Exception e) {
//e.printStackTrace();
}
}
}
}
private JMenuBar addMenubar() {
JMenuBar menuBar = new JMenuBar();
```

```
JMenu mnuFile, mnuOptions;
JMenuItem mtmExit, mtmLoad;
mnuFile = new JMenu("File");
mtmExit = new JMenuItem("Exit", 'x');
mnuFile.add(mtmExit);
mnuFile.setMnemonic('F');
menuBar.add(mnuFile);
mnuOptions = new JMenu("Options");
mtmLoad = new JMenuItem("Load Datas");
mnuOptions.add(mtmLoad);
mnuOptions.setMnemonic('O');
menuBar.add(mnuOptions);
mtmExit.addActionListener(this);
mtmLoad.addActionListener(this);
return menuBar;
}
public static void main(String[] args) {
new OntologyMain();
}
}
```

REFERENCES

1. Amal Zouaq and Roger Nkambou (2009), 'Enhancing Learning Objects with an Ontology-Based Memory', IEEE transactions on Knowledge and data Engineering, vol.21, no. 6.

2. M.-H. Abel, A. Benayache, D. Lenne, C. Moulin, C. Barry, and B. Chaput (2004) , 'Ontology-Based Organizational Memory for e-Learning', J. Educational Technology & Soc., vol. 7, no. 4, pp. 98-11.

3. S. Amershi and C. Conati (2006), 'Automatic Recognition of Learner Groups in Exploratory Learning Environments', Proc. Eighth Int'l Conf. Intelligent Tutoring Systems, pp. 463-472.

4. J. Bourdeau, R. Mizoguchi, V. Psyche´, and R. Nkambou (2004), 'Selecting Theories in an Ontology-Based ITS Authoring Environment', Proc. Seventh Int'l Conf. Intelligent Tutoring Systems, pp. 150- 161.

5. P. Buitelaar, P. Cimiano, M. Grobelnik, and M. Sintek (2005), 'Ontology Learning from Text', Proc. Tutorial at European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2005).

6. T. Berners-Lee, J. Hendler, and O. Lassila (2001), 'The Semantic Web', Scientific Am. Magazine, vol. 5, no. 1, pp. 34-43.

7. K. Cardinaels, M. Meire, and E. Duval (2005), 'Automating Metadata Generation: The Simple Indexing Interface', Proc. 14th Int'l Conf. World Wide Web, pp. 548-556.

8. M.J.Connor, H. Knublauch, S.W. Tu, B. Grossof, M. Dean, W.E. Grosso, and M.A. Musen (2005), 'Supporting Rule System Interoperability on the Semantic Web with SWRL', Proc. Fourth Int'l Semantic Web Conf. (ISWC '05).

9. M.-C. De Marneffe, B. MacCartney (2006), and C.D. Manning, 'Generating Typed Dependency Parses from Phrase Structure Parses', Proc. Fifth Conf. Language Resources and Evaluation, pp. 449-454.

10. E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, and C.G. Nevill- Manning (1999), 'Domain-Specific Key Phrase Extraction', Proc. 16th Int'l Joint Conf. Artificial Intelligence, pp. 668-673.

11. R.M. Gagne´, L.J. Briggs, and W.W. Wagner (1992), 'Principles of Instructional Design', fourth ed. HBJ College Publishers.

12. M. Hearst (1992), 'Automatic Acquisition of Hyponyms from Large Text Corpora', Proc.14th Int'l Conf. Computational Linguistics, pp. 539- 545.

13. IMS ePortfolio (2007) Specification, www.imsglobal.org/ep/index.html.

14. J. Jovanovi_c, D. Gasevic, and V. Devedzic (2006), 'Ontology-Based Automatic Annotation of Learning Content', Int'l J. Semantic Web and Information Systems, vol. 2, no. 2, pp. 91-119.

15. D. Klein and C.D. Manning (2003), 'Accurate Unlexicalized Parsing', Proc. 41st Meeting Assoc. for Computational Linguistics, pp. 423-430.

16. M.D. Merrill (1999), 'Instructional Transaction Theory (ITT): Instructional Design Based on Knowledge Objects', Instructional Design Theories and Models New Paradigm of Instructional Technology, C.M. Reigeluth, ed., Lawrence Erlbaum Assoc.

17. R. Mizoguchi and J. Bourdeau(2000), 'Using Ontological Engineering to Overcome Common AIED Problems', J. Artificial Intelligence and Education, Special Issue on AIED, vol. 11, pp.107-121.

18. J.D. Novak and A.J. Can (2006) as, 'The Theory Underlying Concept Maps and How to Construct them', technical report, Florida Inst. for Human and Machine Cognition.

19. P.R. Polsani (2004), 'Use and Abuse of Reusable Learning Objects', J. Digital Information, vol. 3, no. 4, Article No. 164, http:// jodi.tamu.edu/Articles/v03/i04/Polsani.