Université de Montréal

XML-based Distributed Authoring Tool with Ontology Representing the Domain Knowledge

par

Lidong Wang

Département d'informatique de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès science (M.Sc.)
en Informatique

March, 2002

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

XML-based Distributed Authoring Tool with Ontology Representing the Domain Knowledge

Présenté par :

Lidong Wang

A été évalué par un jury composé des personnes suivantes :

Frasson, Claude	Président-rapporteur
Nkambou, Roger	Directeur de recherche
Kaltenbach, Marc	Membre du jury

Mémoire accepté le : 29 mai 2002

Acknowledgements

I would like to thank Professor Roger Nkambou. As the supervisor, Roger devoted much time and energy on this thesis. He often gave me encouragements and many critiques and suggestions on this work. He proposed many good ideas to improve and extend the project during our discussions. This thesis also contains Roger's many efforts to refine the thesis up to the final.

I should also give my sincere thanks to my wife, Yan Zhang. With her helps and encouragements, I finally finished the thesis. I aware I spent less time with her during the period of writing the thesis.

This thesis contains lots of helps from many people. It would not be possible to recognize and thank separately all those individuals who have contributed to the development of this thesis.

Résumé

Comparé à l'enseignement traditionnel dans les salles de classe, l'enseignement par le biais d'un système tutoriel intelligent (STI) permet un encadrement individuel. Cependant, il s'agit d'un système logiciel complexe, puisqu'il traite non seulement de la matière à enseigner mais également de la psychologie et des caractéristiques pédagogiques du professeur ainsi que des apprenants humains, ce en utilisant les techniques de gestion des connaissances, du multimédia, de réseau et d'intelligence artificielle.

Le curriculum (plan d'enseignement) est une composante très importante dans un STI. Il permet de formuler la représentation des objectifs d'apprentissage et des ressources didactiques, de choisir et d'ordonnancer les activités pédagogique. La représentation des connaissances du domaine est une étape importante au développement d'un curriculum. Elle exige beaucoup de travail, mais il existe peu de composants fonctionnels réutilisables dans cette tâche.

Dans cet ère technologique, le E-learning est devenu un domaine privilégié dans lequel on développe des technologies d'apprentissage humain. Le réseau internet favorise l'enseignement coopératif et l'enseignement à distance. Le Web est ainsi devenu la plate-forme idéale pour une interface d'apprentissage et permet ainsi à un apprenant d'apprendre partout où il se trouve et en tout temps.

Ce mémoire décrit le projet *XML-based distributed authoring tool with ontology representing the domain knowledge*. Cette projet consiste à développer un environnement auteur basé sur XML pour le support à la représentation des connaissances dans un domaine. L'ontologie considérée est celle définie par CREAM-C (un langage de spécification de connaissances d'un domaine d'apprentissage). Cet ontologie a été adaptée pour répondre à la norme IEEE (LOM) et aux spécifications d'IMS. L'outil auteur que nous avons développé permet à un expert dans un domaine donné, de développer et d'organiser les connaissances via un site Web. Les éléments de

connaissances ainsi créés et organisées (par des liens sémantiques) sont encodés dans un fichier XML directement exploitable par les autres modules d'un système tutoriel intelligent.

En offrant une ontologie ouverte, en permettant au développeur d'explorer graphiquement la structure des connaissances d'un domaine et en offrant un support à l'élaboration d'un plan de cours, cet outil contribue à accélérer le processus de développement d'un système de E-learning. Ceci représente un avantage certain lorsqu'on sait que plus de 70% du temps de développement d'un tel système est consacré au développement et à l'organisation des connaissances et des activités pédagogiques.

Mots-clés :

Système tutoriel intelligent
XML
Ontology

Abstract

Compared with traditional teaching in the classroom, teaching using Intelligent Tutoring System (ITS) has more advantages, such as teaching individually, using powerful multimedia resource, saving human efforts, and so on. However, an intelligent tutoring system is a complex software system, since it deals with not only the subject matter to be taught but also the psychology and pedagogical characteristics of human teachers and learners by using knowledge and data management technology, multimedia technology, network technology and artificial intelligence.

A curriculum is a very important part in ITS. The goal of a curriculum in an ITS is to formulate the representation of the materials, selecting and ordering actions in particular activities. Setting up the domain knowledge is the foundation of constructing curriculum. It requires a lot of work, but few functional components are reusable now.

Our society is stepping into the information society and everything, including knowledge, changes so fast. E-learning is the key to deal with continuous change of learners' needs and knowledge. All activities are integrated through the network (Internet) and computer technologies. Through E-learning, we can learn what we want to learn anywhere at any time.

The thesis describes the project XML-based distributed authoring tool with ontology representing the domain knowledge, which is, based on CREAM-C, a part of E-learning and developed under LOM standard and IMS specification. The authoring tool we developed allows a domain expert, in the Internet, to specify domain knowledge involved in the curriculum. The tool can create and query the knowledge developed either in local machine or in remote computers, and establish the knowledge link between knowledge nodes in local computer and remote computers respectively. The established domain knowledge is encoded

into XML file that is convenient for being transferred and shared in Internet and being referenced by other ITS modules.

The authoring tool uses ontology to provide the mean of knowledge representation that is suitable for many domains. It uses visual ways to satisfy the domain knowledge design in curriculum and assists curriculum authors and assist tutors to construct curriculum and plan courses. With the authoring tool, the progress to set up curriculum can be accelerated and avoid doing many jobs which must be done with standalone tutoring system.

Key Words:

Intelligent tutoring system
XML
Ontology

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Compared with traditional teaching in the classroom, teaching using computer-assisted tutoring system has more advantages, such as teaching individually, using powerful multimedia resource, saving human efforts, and so on. However, a computer-assisted tutoring system is a complex software system, since it deals with not only the subject matter to be taught but also the psychology and pedagogical characteristics of human teachers and learners by using knowledge and data management technology, multimedia technology, network technology and artificial intelligence (AI).

Since the beginning of the 1970s, Intelligent Tutoring Systems (ITS) have been developed via computer software integrating many domains such as AI, education science, and cognitive psychology. Some systems, such as SHOLAR, SOPHIE, BIP, Geometry-Tutor STEAMER, GUIDON, WordTutor and GEOCAM, have been developed and concerned with many parts of ITS, like student modeling, domain expertise, integration of teaching strategies, curriculum integration. Now the remaining essential concern in the design of tutoring systems are specifying teaching activities, and efficiently organizing and reusing teaching materials (i.e. curriculum).

A curriculum is a very important part in ITS. The goal of a curriculum in an ITS is to formulate the representation of the materials, selecting and ordering actions in particular activities. [McCalla, 90] gave a more cognitive definition: *the curriculum in ITS represents the selection and arrangement of knowledge with a view to realize teaching goals that are appropriate for the current context and the*

current student. This definition declares the fact that a curriculum should be flexible, and should adapt to the needs of students and the authors of teaching materials. Setting up the domain knowledge is the foundation of constructing curriculum. It requires a lot of work, but few functional components are reusable now.

Our society is stepping into the information society rapidly. In the information society, the dominating issue is the speed of change. Everything changes so fast that people, companies, organization and society do need something that helps them catch up with it. The rapid upgrade, of course, includes that of knowledge. In the information society, some main aspects related to knowledge engineering are listed in the following.

1. Knowledge is constantly updated.

2. Life-long learning is required.

3. Huge amount of knowledge and information are accessible through computers and Internet conveniently.

4. Facing huge amount of knowledge, learning capability and creativity becomes much more necessary.

5. Training in industries becomes more often and more important.

People with a lot of knowledge that is valuable now does not mean that their knowledge will always be up-to-dated, i.e. the senior people who were thought to have more knowledge than juniors, have more difficulty in using computers and Internet, while juniors are the promoters of computer technology.

As huge amount of knowledge becomes available through Internet in the information society, it becomes possible for people to access the knowledge they need when necessary. In such a circumstance, it is more important to know where to and how to find the up-to-dated knowledge, how to understand and master the

new knowledge and how to create knowledge for future use. Education done in schools is very limited since the knowledge upgrades quickly, and is passive since students don't know how the knowledge is used in the real world. Facing the changeable world, life-long learning and anytime & anywhere learning are very necessary. [Guarino 97].

E-learning is the key to deal with continuous change of learners' needs and knowledge. All activities are integrated through network (Internet) and computer technology. Through E-learning, we can learn what we want to learn anywhere at any time. The key ideas in E-learning include:

(1) Knowledge flow running on the Internet makes tutorial materials upgraded as quickly as possible.

(2) Curriculums are directly connected to students' needs.

(3) Modularization and standardization makes knowledge modules highly reusable.

The above means that an E-learning system should dynamically compose of training/teaching materials for each learner through interaction with him/her. To do this, the system has to identify what the learner needs, find knowledge through Internet/intranet, and then configure knowledge modules.

To enable E-learning, many new techniques have been proposed. One of them is ontology engineering. In recent years ontology engineering has been drawing much attention in the domain of intelligent training system [Mizoguchi, 96][Murray, 1998]. Ontology engineering is a research methodology that gives us design rationale. It enables accumulation of knowledge and supports knowledge reusable and sharable. It provides an explicit concept representation at different levels of abstraction. The ultimate purpose of ontology engineering is to "provide a basis for building models of all things in which information science is interested in the world"[Mizoguchi, 2000].

Another core technique is XML. XML (eXtensible Markup Language) has risen from virtual obscurity two years ago to being touted as the format for data interchange. XML tagged data make it easier for computer to access, transfer, search and store files via Internet. XML is gradually becoming standard for data exchange between all platforms, systems and applications. XML is revolutionizing the Internet and E-learning.

1.2 Project objective

In order to find, for everyone, educational materials that are described by various metadata on the web, IEEE proposes a standard LOM (Learning Object Metadata) standard. On the basis of LOM standard [LOM], the IMS global learning consortium, which is specialized in promoting open specifications for facilitating online distributed learning activities, has proposed an IMS Learning Resource Metadata Specification [IMS]. With this technical specification, developers and creators of products and services can work together to share developed materials.

Our project is a part of E-learning and developed under LOM standard and IMS specification. We will develop a distributed authoring tool that is based on XML technology and using ontology to represent domain knowledge.

[Nkambou, 98] proposed a curriculum model CREAM (Curriculum Representation and Acquisition Model) in SAFARI Project where three kinds of knowledge were identified and organized: capabilities (CREAM-C), objectives (CREAM-O), and resource (CREAM-R). CREAM organizes every kind of knowledge as a network with nodes and links, then another network-CKTN (Concepts Knowledge Transition Networks)- is defined to associate the above three spaces to get transition network. CREAM dealt with more kinds of knowledge than previous models. Organizing multiple spaces is flexible for manipulating knowledge. Selection of specific nodes as learning objectives facilitates the organization of tutoring activities.

On the basis of CREAM-C, we are developing an authoring tool allowing a domain expert, in the Internet, to specify domain knowledge involved in the curriculum. With the aid of XML, we can create and query the knowledge developed either in local machine or in remote computers, and establish the knowledge link between knowledge nodes in local computer and remote computers respectively. The established domain knowledge can be referenced by other ITS modules or uploaded on the Internet.

The goals of the authoring tool are 1) providing the means of knowledge representation and organization for domains as many as possible. 2) using visual ways to satisfy the domain knowledge design in curriculum. 3) assisting curriculum authors and tutors to construct curriculum and plan courses. 5) helping tutorial system authors to build more powerful and flexible intelligent tutoring systems.

1.3 Thesis organization

This thesis is divided into seven chapters.

This chapter is an introduction that gives the project objective and thesis organization. Chapter 2 will introduce the development history and modules of intelligent tutoring system, and authoring system by categories. Chapter 3 focuses on the curriculum. We will briefly review the curriculum in some projects, and mainly illustrate curriculum structure in SAFARI project and gives its characteristics and limitations. Chapter 4 gives a detailed description on the ontology and gives the state of art on ontology research. Chapter 5 will give the concept idea on the tool we developed. We will use UML diagrams to describe the system and give reasons why we use some techniques in our project. In chapter 6 we will use an example and screen shots to explain the implementation of the designed tool. Chapter 7 is the conclusion.

Chapter 2

Intelligent Tutoring System

In this chapter, we will review the history and components of intelligent tutoring system, except curriculum. Then we will also introduce the authoring system by categories. The deep introduction of curriculum is put in chapter 3.

2.1 Computer-Assisted instruction system

The Computer-Assisted Instruction (CAI) system appeared in 1950s with a programmed system that placed students in a simple interactive environment. On screens, students answered some questions that the system presented to them. With evaluation tests in certain systems, according to the evaluation of students' responses, the system can decide what to be presented on the next. However, the evaluation was limited only to decide whether the response was correct or wrong. The big weakness of the approach is that there is no deep analysis of domain concepts and students responses. This type of system evolved with the use of pattern-matching techniques to analyze the responses, and later (in 1970s) toward adaptive systems (generating didactic materials, in particular for arithmetic problems and vocabulary acquisition). In such a system, the student model was a record of his/her past behavior parameters. This evolution represents a big step toward individualized teaching, but it remains insufficiency because of the difficulty of analyzing and comprehending students' intentions as well as the inability of reasoning on the teaching domain.

The curriculum in these systems is not taken into account. They used just a simple linear organization (sequences of frame screens) in a predefined order based on students' response. For example, in the system of Kimball, the advice was given to the students according to the quantity of entered data for his or her solution,

rather than the validity of his or her response. Besides, there is no access to the content of a frame, and the teaching domain cannot be reached.

In 1970s, SCHOLAR was proposed and artificial intelligence was used for the time. In SCHOLAR, knowledge to teaching process was separated from the domain knowledge. Thus, the system had a semantic network that represented the fact on the geography of Southern America and a program for managing the interaction with students. This proposal started the evolution of CAI toward the Intelligent Tutoring System.

2.2 Intelligent tutoring system

It is because artificial intelligence was used that Computer-Assisted Instruction system has changed to Intelligent Tutoring System. Computer-Assisted Instruction evolves toward Intelligent Tutoring Systems (ITS) by using the following aspects of intelligence.

1) The subject matter, or domain, must be known by the computer system well enough for this embedded expert to draw inferences or solve problems in the domain.

2) The system must be able to deduce a learner's approximation of that knowledge.

3) The tutorial strategy or pedagogy must be intelligent in that the instructor can implement strategies, which can reduce the difference between expert and student performance.

Generally, ITS includes such components as expert module, student diagnostic module, instruction module, instructional environment and human-computer interface shown in Figure 2.1. The expert module contains the domain knowledge.

The student diagnostic module diagnoses what the student has already known. The instructor module identifies what are deficient in knowledge and selects strategies to present related knowledge. The instructional environment and human-computer interface are responsible for tutorial communication.

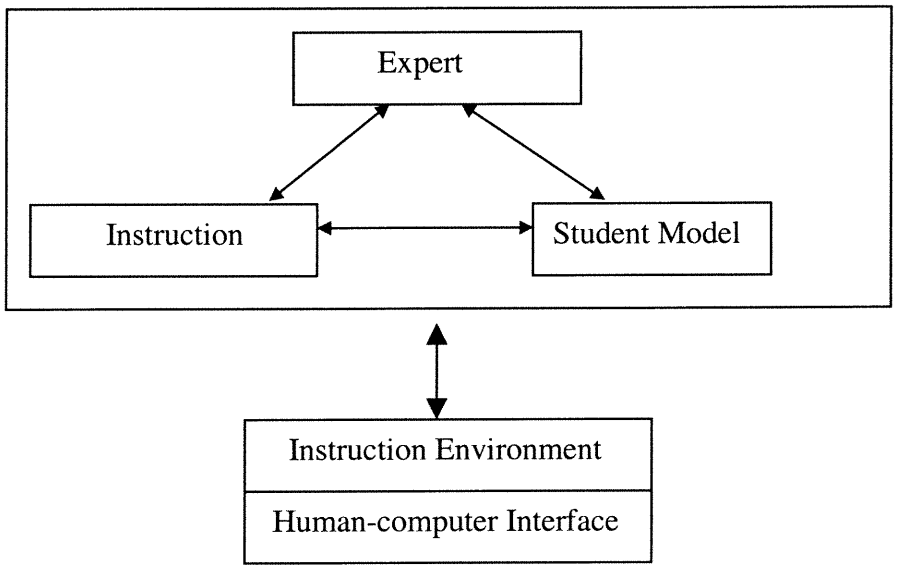

Figure 2.1 Basic Architecture of ITS [Polson 88]

2.3 Expert module

Any expert module must have abundant specific and detailed knowledge derived from people who have years of experience in a particular domain. Consequently, much effort is spent in discovering and organizing the domain knowledge, thus distilling years of experience into a knowledge representation. The enormous amount of knowledge in complex domains as well as the interrelationship of that knowledge means that designing and developing the expert module may be the most demanding chore in building an ITS. Thus, investigating how to encode

knowledge and how to represent such expertise in an ITS remains the central focus of developing an expert module.

Three approaches are common to encode the knowledge in the ITS data structure, each moving toward a more cognitively faithful representation of the content expertise. The first is " black box" expert systems, finding a way to encode the knowledge without actually codifying the underlying human intelligence. A programmer attaches instructions to specific issues observable in the behavior of both the expert and the student within the learning environment.

The second approach involves the building of a "glass box model" to influence the tutorial mechanisms of the system. To do this people must use knowledge-engineering techniques. A knowledge engineer interviews an expert and designs a computational representation for delivering the knowledge, usually a rule-based formalism. This implementation does not necessarily correspond to the way the human expert reasons, especially in novel, unfamiliar situations or when providing explanations. Thus this glass box model allows only for explanations of the information process inherent in the rules of its knowledge base. The rules are typically more strategically aligned with performance rather than explanation, limiting their utility in an instructional setting. However, knowledge-engineering tools and techniques, that is, ways of extracting and codifying information, are becoming more and more useful for ITS development as attention is paid toward making representation more faithful to the breadth and depth of expert reasoning.

The third approach is to encode the domain knowledge and simulate not only the knowledge but also the way that people use the knowledge. This is clearly the most demanding approach to developing an expert module.

2.4 The student diagnosis module

Many ITSs infer a model of the student's current understanding of the subject matter and then use that understanding to adapt the instruction to the student's particular needs. The knowledge structure that depicts the student's current state is the student model, and the reasoning process to develop it is called" student diagnosis." Outputs from student diagnostic modules can be used for variety of purposes, such as advancing through selected curriculums, coaching or offering unsolicited advice, generating new problems, and adapting sets of explanations.

According to VanLehn's classification [Polson 88], target knowledge types are divided into two types, flat and hierarchical, and declarative knowledge. Specialized strategies for using (or interpreting) knowledge are paired with each type of knowledge. The interpretation process is more difficult to implement for declarative knowledge than for procedural. The interpreter for hierarchical procedural knowledge is harder than for flat procedural knowledge. "Because the difficulty of the student diagnostic process is closely related to the difficulty of the interpretation process, a flat procedural knowledge base makes the student modeling process the easiest, whereas a declarative knowledge base presents the most difficult student modeling problem." [Polson 88]

Most ITS designers use the same knowledge representation scheme as was used in the expert module so that the expert and student modules actually share the same knowledge base. This is called the overlay method of student modeling, where the student's knowledge is represented as a subset of the expert's.

"The next level of complexity in student modeling is to represent misconceptions, erroneous and incorrect knowledge, as opposed to simply incomplete knowledge. In this approach, the overlay model is augmented by a bug library. Bugs, that is, misconceptions or misunderstandings, must typically be collected empirically, but

can be generated computationally from the target procedure, as is done in repair theory [Brown 80]. To reduce the empirical work required for obtaining an exhaustive set of bugs, bugs are sometimes generated form bug part libraries, where bug parts, fragments of production rule clauses, are assembled into bugs. This represents the highest degree of sophistication in student modeling. Success in this is critically tied to the bandwidth issue."[Polson 88]

2.5 The instructional environment

An instructional environment consists of the following elements of an ITS that support what the learner is doing: situations, activities, and tools provided by the system to facility learning.

The activities and tools presented to a learner in an ITS always reflect an underling educational philosophy. The trend, as computers gets faster and as ITS researchers and educators become more creative and clever, is clearly to create more open, more robust, more fulfilling and more effective educational experience. Several principles for building instructional environments have emerged from this trend. An instructional environment should prove that there is more in an ITS than meets the eyes. It should foster constructive learning through activities-tools, games...-designed to use students' prior knowledge and to present students with new information and experiences from which they can construct new knowledge. The environment should emphasize conceptual understanding, not rote procedures. It should attempt to connect in-school and out-school knowledge. It should be designed to let students feel self-monitored, allowing effective learners to assume responsibility for their own learning. The environment should also be developed in the premise that education is a life-long pursuit. From such principles, educational technology community generally believes that computerized instructional environment becomes self-contained

worlds that can enhance and motivate learning-even if the environments themselves are not intelligent.

Research considerations pertaining to instructional environments are: (a) levels of abstraction, (b) fidelity, (c) sequence and (d) help routines. Level of abstraction means what features of the real world are represented in the design of the environment. Fidelity means how closely the simulated environment matches the real world. Here considerations of the different types of fidelity are important; for example, physical fidelity, display fidelity, mechanical fidelity, and cognitive fidelity. Sequence refers to the framework a designer constructs for learning complex skills. A learner progresses through a sequence of increasingly complex micro worlds, each provides new challenges and new sets of achievable goals. By means of help routines, the designer take into account that additional information learner may need for operating the ITS when they are in trouble. For example, help tells a learner what to do.

2.6 The human-computer interface

The learner working with an ITS generally has two problems. First, the learner must learn some subject matter that he or she may not understand. The other problem is that the learner is not very likely an expert user. If the human-computer interaction is poorly designed, a training session will probably be ineffective.

The goal of interface design is to make the interface transparent. The knowledge embedded in the component of an ITS thus evolves knowledge of previous computer systems, human interface research, real objects that are being imitated in the computer system, and knowledge of the entire range of the communication process-perceiving, understanding and creating meaning. Good interfaces embody an understanding and appreciation for the goals and concepts that are important to users and to the domain being taught.

2.7 The instruction and curriculum module

An ITS should have three tutoring characteristics: (a) controls over the representation of the instructional knowledge for selecting and sequencing the subject matter; (b) capabilities for responding to students' questions about instructional goals and content; and (c) strategies for determining when students need help and for delivering the appropriate help. The goal of the instructional and curriculum module is to circumscribe the nature of teaching and implement teaching as a solution to the educational communication problem. Here we only introduce instruction part and chapter 3 will present curriculum in detail.

2.7.1 Instruction

Instructional knowledge routines should allow a student to relate theory to practice, to propose solutions, to develop more effective problem-solving strategies. They should also minimize the load on the student's working memory while new concepts are being internalized.

Thus, the instruction and curriculum module should and can be more than just a by-product of the expert and student modules, and some instructional principles should be robust and explicit enough to generalize across domains. The available literature on instructional theory provides instructional methodologies and can help designers decide such questions as what information to present to what sequence.

Presentation techniques all depend on the instructional objective. Elicitation and explanation help lead learners to an understanding of facts and concepts. Case presentations and simulated entrapment induce learners to formulate rules and to understand relations. Exercises, drills, and examples allow learners to generalize from subskills to the performance of the full task. Seeing the required skills

prepares the student for the real-life situation. All of these strategies should be encompassed in an ITS design when the instructional module is laid out.

Achieving any dynamic flexibility at the instructional level requires designing specific instructional states and means of transitioning from one state to another. The artificial intelligence techniques may be the most useful in the instructional module.

Another challenge to artificial intelligence involves understanding instructional discourse. Such understanding, for example, would include strategies appropriately intervening in the course of a student's problem-solving activity. Intervention, on one hand, allows the ITS control of the tutorial process, but it is also important in keeping a learner on the right track by preventing inappropriate or incorrect learning. Beyond intervention, that is, offering advice, hints or guidance, other strategies are need for answering questions and providing explanations. These kinds of abilities must be incorporated in the design of an adequate instructional module.

2.8 Authoring tools

The restricted domains do not pose the problems linked to the design and organization of teaching. The task of teaching design, also known as instructional design, is a complex process requiring dedicated experts. A theory of instructional design has been developed [Gagné, 93]. Due to the difficult, repeated and costly process, it is necessary to provide effective tools. The required expertise is difficult to acquire, and the integration of such an expertise in an authoring environment is absolutely necessary. The integration leads to the availability of powerful tools well integrated in the authoring environment. In the following paragraphs, we describe the authoring systems that are dedicated to the instructional design and divided by categories stated in [Murray, 99].

Early ITS authoring systems fell into two broad categories: those geared toward device simulation and embodying a "learning environments" instructional metaphor, and those based on a traditional curriculum (or courseware) metaphor. Even though some recent systems combine aspects of both perspectives, the majority of authoring tools fall similarly into two broad categories: pedagogy-oriented and performance-oriented [Murray, 97]. Pedagogy-oriented systems focus on how to sequence and teach content. Performance-oriented systems focus on providing rich learning environments in which students can learn skills by practicing them and receiving feedback.

1. Curriculum sequencing and planning

Authoring systems in the Curriculum Sequencing and Planning category organize instructional units (IUs, or "curriculum elements") into a hierarchy of courses, modules, lessons, presentations, etc., which are related by prerequisite, part, and other relationships. The instructional units typically have instructional objectives. Some systems include IUs that address misconceptions or remedial material. The content is stored in canned text and graphics. These systems are seen as tools to help instructional designers and teachers design courses and manage computer-based learning.

Intelligent sequencing of IUs (or content, or topics) is at the core of these systems. To the student, tutoring systems built with these tools may seem identical to traditional computer-based instruction. Screens of canned text and pictures are presented, and interactions tend to be limited to multiple choice, fill-in, etc. Of course, the difference is that the sequencing of the content is being determined dynamically based on the student's performance, the lesson objectives, and the relationships between course modules. Because domain knowledge is not represented in a very "deep" fashion, any arbitrary domain can be tutored (just as a textbook can be about any domain). But the depth of diagnosis and feedback in tutors built with these authoring tools is limited by the shallowness of their domain knowledge representation. This makes them more appropriate for building

tutors that teach conceptual, declarative, and episodic types of knowledge, and less pedagogically powerful for building tutors that teach procedural or problem solving skills.

Authoring systems in the Curriculum Sequencing category are, generally speaking, the most "basic," or minimally functional (though each system in this category has certain very evolved signature features or capabilities). Several of the other categories described below contain these minimal capabilities (such as curriculum sequencing or IU planning) and add additional functionality.

2. Tutoring strategies

Systems in this category excel at representing diverse teaching strategies. They tend to be similar to the Curriculum Sequencing systems described above, in that content is stored in canned text and graphics and domain knowledge representation is shallow. However these systems also encode fine-grained strategies used by teachers and instructional experts. Systems in the Curriculum Sequencing category above tend to focus on the "macro" level of instruction--i.e. the sequencing of topics or modules, while systems in this category also address the "micro" level of instruction. Instructional decisions at the micro level include when and how to give explanations, summaries, examples, and analogies; what type of hinting and feedback to give; and what type of questions and exercises to offer the student. Systems in the Tutoring Strategies category have the most sophisticated set of primitive tutorial actions, compared with systems in other categories. Also characteristic to systems in this category is the ability to represent multiple tutoring strategies and "meta-strategies" that select the appropriate tutoring strategy for a given situation.

As in Curriculum Sequencing systems, students using tutors built with these authoring tools will see screens of limited interactivity--they will be learning by reading and thinking, rather than learning by doing. However, the availability and intelligent interjection of small grain sized components such as explanations,

multiple levels of hints, and analogies can make the tutor appear quite responsive, at times even conversational, compared to tutorials built with Curriculum Sequencing systems.

3. Device simulation and equipment training

For tutors built by authoring tools in this category, the student is shown a piece of equipment and is asked to identify its components, perform operating steps, perform maintenance steps, or diagnose faulty device behavior and fix or replace the implicated parts. These types of skills are relatively widespread and generic, so authoring tools that specialize in this area should be widely usable. The expert knowledge for component locations and operational scripts is straightforward to model. Performance monitoring and instructional feedback is also straightforward (e.g. "That is not the Termination Switch," and "You should have checked the safety valve as your next step"). Thus authoring tools can be built which closely match the needs of the author (and student). The most difficult authoring task with these systems is building the device simulation. But once the simulation is authored, much of the instructional specification comes "for free." Component location and device behavior "what if" activities can be generated automatically. However, the device operation procedures must be authored.

In contrast to the previous two categories of authoring tools, students using tutors built with tools in this category will be "learning by doing." Introductory or conceptual instruction is absent or limited, and it is assumed that students have a basic familiarity with important concepts and procedures in the domain before using the tutor. These tutors are learning environments in which to practice skills. Specific feedback is given for each skill step, and task difficulty is increased as students' progress.

The major differentiating factor among systems in this category is the depth and fidelity of the device simulation. Authoring tools range from those supporting static expression-based relationships between device components to those supporting runnable but shallow simulation models, to those supporting deeper,

more causative or cognitive models of how the device works. The tools also vary widely in the types of devices and physical processes that they can model.

4. Expert systems and cognitive tutors

The important classes of intelligent tutors are those that include rule-based cognitive models of domain expertise. Such tutors, often called model tracing tutors [Anderson 1991], observe student behavior and build a fine-grained cognitive model of the student's knowledge that can be compared with the expert model. Authoring tools have been prototyped for such tutors. In this category authoring tools are included, which use traditional expert systems (built to solve problems, not to teach) and produce "value added" instruction for the encoded expertise. These systems are similar to model tracing systems, except the expert system is based on performance competency, rather than cognitive processes. Some systems include buggy or novice-level rules that capture common mistakes, allowing the tutorial to give feedback specific to those errors.

Students using these systems usually solve problems and associated sub-problems within a goal space, and receive feedback when their behavior diverges from that of the expert model. Unlike most other systems described above, these systems have a relatively deep model of expertise, and thus the student, when stuck, can ask the tutor to perform the next step, or to complete the solution to the entire problem. Authoring an expert system is a particularly difficult and time-intensive task, and only certain tasks can be modeled in the manner.

5. Multiple knowledge types

Instructional design theories classify knowledge and tasks into discrete categories, and prescribe instructional methods for each category. They tend to be limited to

types of knowledge that can be easily defined, such as facts, concepts, and procedures. Though the knowledge types and instructional methods vary for different theories, they typically prescribe instruction similar to the following. Facts are taught with repetitive practice and mnemonic devices; concepts are taught using analogies and positive and negative examples progressing from easy prototypical ones to more difficult borderline cases; procedures are taught one step at a time. Instruction for these knowledge types includes both expository presentations of the knowledge, and inquisitory exercises that allow for practice and feedback. Straightforward instructional strategies for how to sequence content and exercises, and how to provide feedback, are defined separately for each knowledge type. The pre-defined nature of the knowledge and the instructional strategies is both the strength and the weakness of these systems. Domain knowledge for each knowledge type can be easily represented for authors, who fill in templates for examples, steps, definitions, etc. The tools support the decomposition of complex skills into elementary knowledge components, and links between knowledge components can be authored and used in instruction. Since instructional strategies are fixed and based on knowledge types they do not have to be authored. Of course, not all instruction fits neatly into this framework, but it has significantly wide applicability.

Authoring systems in the Multiple Knowledge Types category are diverse in many respects, but they all use a knowledge/skill classification scheme and represent and instruct differentially based on knowledge type. Also, they all cite classic instructional design literature as part of the basis for their pedagogical approach. For the student, tutors built with these systems are similar in character to those in the Multiple Teaching Strategies category. The main difference is for the authors, whose task is more constrained, and thus both easier and less flexible.

6. Special purpose systems

In this category are authoring tools that specialize in particular tasks or domains. Systems in the Device Simulation and Multiple Knowledge Types categories are also for particular types of tasks, but systems in the Special Purpose category focus on more specific, less general tasks. There is a rough principle that authoring tools tailored for specific tasks or instructional situations can better support the needs of the student and author for those situations. Systems in this category were designed by starting with a particular intelligent tutor design, and generalizing it to create a framework for authoring similar tutors. Authoring is much more template-like than in other categories of authoring tools. Authors are usually given prototypical examples that help them fill in the blanks. One potential problem with special purpose authoring is that once a task and its instructional approach have been codified enough to become a template, the resulting system reflects a very particular approach to representing and teaching that task; one that may only appeal to a limited authoring audience. On the other hand, preferred design and pedagogical principles can be strictly enforced, since the author has no influence over these aspects.

Since the only thing that systems in this category have in common is that they support particular types of tasks or domains, we cannot say anything in general about the types of tutors built or about students' experience using the tutors.

7. Intelligent/adaptive hypermedia

As adaptive hypermedia systems and web-based tutors become more sophisticated, they increasingly incorporate methods and models from the field of intelligent tutoring. The functions of these systems overlap with those from the Curriculum Sequencing and Tutoring Strategies categories above (depending on whether the focus is on instruction at the macro or micro level). As is the case with most web-based systems today, the level of interactivity and fidelity available to the student is low for tutors built with these authoring tools. Unlike systems in the other categories, these systems must manage the hyperlinks

between units of content (as well as the form and sequencing of the content itself). The links available to the student can be intelligently filtered, sorted, and annotated based on a student model or profile [Brusilovsky, 98]. Link filtering can be based on prerequisites, cognitive load, topic appropriateness, difficulty, etc.

Chapter 3

Curriculum

In the educational domain, the development of a curriculum passed by three stages: (1) the identification of teaching content, (2) the definition of teaching objectives and (3) the determination of necessary means for deriving the instruction steps needed to meet the defined objectives.

Several studies on these curriculum development stages have been undertaken. The key is to know on what stage it starts to obtain an optimal curriculum. Some people propose to start by building the objectives before identifying the representation of content. The consequence of this approach is that one can find objectives with weak content. However, when the content deriving materials precedes the identification of objectives, the result is tangible and thus is more significant.

Determination and classification of teaching content consists of identifying and defining the knowledge that is used in the teaching environment. The determination of content is based on two phases: the utilization of strategies for the identification of relevant knowledge, and the classification and representation of retrieved knowledge.

3.1 Strategies determining teaching content

The strategies, from more subjective to more objective, are: philosophic foundation, introspection, functional approach, and task analysis. The utilization of one or another strategy in a process of curriculum development depends on the particular knowledge domain.

The strategy, philosophic foundation, consists of using a specific philosophy or a set of philosophies as the foundation for the choice of content. It is based on verbal assertions from educators. For example, consider the following assertion, all content of curriculum of driving theory do not touch the practical driving environment. Once the curriculum developers agree with this assertion, the relevant content to this assertion has been identified. This strategy may be more subjective, but it is used widely in the development of curriculum in academic domains.

An introspection strategy is based on examining the thinking and beliefs of people while implied in the development of a curriculum relative to a knowledge domain. It implies that one or more teachers or domain experts pose the following foundational question: whether this or that item seems pertinent to be included in the curriculum? The different contribution will be analyzed later in order to select the most pertinent content. Thus, the content follows from a discussion between a group of professors. The advantage of this approach is that one can get a curriculum with rich content. More specific approaches within this identification process can be:

-Seeing the description of domain knowledge as it is progressively linked into a curriculum;

-Identifying the general groups of implied abilities;

-Identifying the specific performance (or behavior) for each group of general abilities;

-Structuring the performance in a significant learning sequence;

-Specifying the levels of competence for each task performance;

The functional approach is based on the definition of functions learning environment in terms of the operations executed in the environment. The content is thus identified in terms of functions.

The task analysis strategy involves the identification and verification of tasks executed in an environment, with a view of including them as the content of a curriculum. Several methodologies exist for task analysis [John 94].

The strategies for content determination are very diverse. The question is that given certain resources, which strategy could be the most appropriate for determining the content of a specific curriculum? In general, the utilization of a unique strategy cannot always cover all information implied in a domain; several strategies should be used for identifying significant content. This can render the final curriculum capable of covering the needs of all students.

3.2 Classification of content

Classification of knowledge is extremely important in the cognitive design of teaching and learning since it orders the strongly different teaching strategies and the representation in human memory. We introduce the notion of capability, in the sense of Robert Gagné, for describing the outputs of learning carrying on content. By capability, what one acquires is the ability developed, allowing a person to succeed in the exercise of an intellectual, professional, or physic activity. It deals with information structure (knowledge of cognitive unit) stored in the long-term memory of the student and that give possibly a performance. It represents what is learned. Several theories have been developed in the education domain, in the psychology and in the artificial intelligence for characterizing the product of learning.

Gagné's Capability Theory

Gagné identified five big categories of capabilities [Gagné, 85] that can produce most of human activities: verbal information, intellectual skills, motor skills, attitudes and cognitive strategies.

1) Verbal information is the kind of knowledge we are able to state. The processors of this kind of capabilities can explain, describe, or name objects or facts.

2) Intellectual skills enable individuals to interact with their environment in terms of symbols or conceptualizations, i.e. solving an equation, programming, etc. Gagné divided intellectual skills into several sub-categories ordered by the complexity of mental operations: discrimination, concrete concepts, defined concepts, rules and high-order rules.

3) Cognitive strategies are the kind of capabilities that allow acquisition of other types of capabilities and their management in the reasoning and the resolution of problems (learning strategies, problem-solving strategies)

4) Attitudes are the internal states that influence the choice and the actions of a person that posses them. For example, a favorable attitude to classic music influence the selection of information that one wants to listen.

5) Motor skills, which remain too little consideration in the goal of learning, are a class of capabilities (output of learning) linked to the activities concerning the movement: driving a car, typing, and sport activities. A curriculum can sometimes imply the acquisition of motor skill.

One of advantages of this classification is that, the particular conditions for favoring the acquisition of all types of knowledge in this taxonomy have been defined by Gagné [Gagné, 93].

3.3 Representation of domain knowledge

Representation based on rules

This approach consists of modeling the expertise of a domain with the help of a rule base; which can represent the domain knowledge and make reasoning on the domain knowledge. The system is often modeled with the help of production rules representing domain knowledge or strategy knowledge. The domain or strategy knowledge is compiled in a rule packet. This approach has been used in several ITS like Geometry Tutor, BUGGY, GUIDON. The advantage of this approach is that it is easy to exploit and allows a good representation of procedural knowledge. The disadvantage is that it is difficult to represent structured knowledge such as concepts and relationships between them.

Representation based on networks

It used network to represent domain knowledge. This representation takes into account the cognitive model of domain. It allows representing most types of knowledge, but it is more appropriate for the representation of declarative knowledge. The advantage of this representation is intuitive for domain experts; it is easier for an expert to express his knowledge with this form than with the rule form. Several varieties of this type of representation have been proposed in AI: semantic networks conceptual graph and the frame networks.

Semantic networks in SCHOLAR were used for presenting domain knowledge. Another example using network representation approach is the subject networks proposed by Murray and Woolf [Murray, 93]. This is a kind of semantic network that is used to represent domain knowledge and different relations among them. Knowledge in a semantic network may be facts, concepts, procedures, principles, etc., and is represented by nodes of network. Murry and Woolf identified different types of links between knowledge: familiar, deep familiar, part of, error of critic

concept, simple prerequisite, and typical prerequisite. The system KAFITS [Murray, 91] provided a browser that is used for building this representation.

Representation based on Layers

Schreiber and his colleagues [Schreiber, 93] proposed an approach of knowledge representation on four layers: domain, inference, task, and strategy. The theory of domain contains concepts, relations between concepts, expressions and relations between expressions. According to the authors, the theory models the static knowledge and declarative domain. There are two types of relations between concepts: aggregation relations (of sub-components) and relations of specification. An expression represents a property of a component (a concept and its states) or a test (and result of the test). Two types of relations exist between expressions: the relation of causality and the relation of indication. If we are in the context of systems, the layers represent the static and the functional aspect, introduced by the expressions and the relations between expressions. The mixing of static and functional levels can cause some problems at the time of exploitation of representation.

Most previous representation approached make abstraction of the aspect of knowledge organization for final teaching, yet it is necessary if one wants to teach [Webster, 94].

Representation based on ontology

Mizoguchi proposed ontology to represent knowledge. It uses well-designed common vocabulary to represent knowledge. Ontology makes it possible to reuse knowledge. The next chapter will introduce ontology in detail.

3.4 Selection and sequencing

Curriculum includes two parts. Formulating a representation of the material is one part of curriculum, and selecting and sequencing particular concepts from that representation is the other.

The differences between expository tutors and procedure tutors are evident in the problems associated with selecting and sequencing material. For expository tutors, the problems are those of maintaining focus and coherence and of covering the subject matter in an order that supports later retrieval of the concepts being taught. Procedure tutors have the additional problem of properly ordering the sub-skills of the target skill and selecting exercises and examples to reflect that order.

Topic selection in expository tutors

Curriculums in expository tutors must deal with two sources of constraints. One set of constraints arises from the subject matter. Topics must be selected to maintain coherence and to convey the structure of the material being taught. A second set of constraints comes from the tutoring context. Selection of some topic or fact for discussion must reflect student's reaction to previous tutoring events.

Exercise and example selection in procedure tutors

Procedural skills are nearly always taught by exercises and examples. The major curricular issue is to choose the correct sequence of exercises and examples. Ideally, the choice of exercises and examples should be dedicated by a learning model that is precise and powerful enough to support a tutoring system. Research on the selection and sequencing of exercises has suggested several standards.

1. Manageability

Every exercise should be solvable and ever example should be comprehensible to students who have completed previous parts of the curriculum.

2. Structural transparency

The sequence of exercises and examples should reflect the structure of the procedure being taught and should thereby help the student induce the target procedure.

3.Individualization

Exercises and examples should be chosen to fit the pattern of skills and weakness that characterizes that student at the time the exercise or example is chosen.

Selection and sequencing criteria

The curriculum designers should care more the goals of curriculum construction than to the principles for design in particular situations. Curriculums for tutoring situations serve several functions:

1. A curriculum should divide the material to be learned into manageable units. These units should address at most a small number of instructional goals and should present material that will allow students to master them.

2. A curriculum should sequence the material in a way that conveys its structure to students.

3. A curriculum should ensure that the instructional goals presented in each unit are achievable.

4.Tutors should have mechanisms for evaluating the student reaction to instruction on a moment-to-moment basis and for reformulating the curriculum.

3.5 Some typical curriculum systems

Curriculum in BIP

BIP (Basic Instructional program)[Barr, 76] is a tutoring system of an introduction course to BASIC programming. It individualizes the teaching sequence via the appropriate selection of series tasks in a set of 100 examples of problems. BIP put its selection strategy on the content information in a network called Curriculum Info Network (CIN) that connects tasks in the curriculum to domain objectives.

In BIP, the curriculum is divided into three conceptual levels. The super level contains the principal themes of expertise call techniques that are ordered in advanced following the prerequisite relations. These techniques are composed of knowledge units of immediately lower level called skills. The skills are not linked each other. The last level contains tasks that ate connected to skills, allowing thus to exercise the last one.

Curriculum in SCENT

SCENT [Bretch, 90] is an adviser system for learning of LISP. The curriculum in SCENT is considered as the result of the planning of content. The approach of curriculum corresponds to the view of McCalla [McCalla, 90], according to which the curriculum corresponds to a plan of course adapting a particular student. The notion is taken back in PEPE [Bretch, 90] where an extended structure of the granule theory [McCalla, 94], with the introduction to the prerequisite link, in addition to existing links (aggregation and generation), is used for generating the curriculum.

Curriculum in Sherlock

Sherlock [Lesgold, 92] is an advice-based tutoring system in which the goal is to supervise and to help the students in airplane who have taken the theoretic courses to learn the knowledge in the detection of breakdown on the F15. The learning environment includes a simulation of test station and equipment to be tested.

One of the powerful points of Sherlock is that it is based on a solid analysis of tasks (done with the help of experts of domain) in detecting the breakdown that posed problems concerning learners' difficulties. The analyses of task allow identifying the skills (curriculum issues) that distinguish the expert technicians from novice technicians. The curriculums have been grouped into three categories: the strategies of detection, the strategies of repairing, and the strategies of resolution and making decision. Sherlock had led to 61 curriculum issues, which are implemented as the pedagogic goals.

3.6 Curriculum in SAFARI

In the project SAFARI, Nkambou [Nkambou, 98] proposed a curriculum model: CREAM. CREAM organizes learning outcomes, objectives and didactic resources as three separated networks. A global association network, Curriculum Knowledge Transition Network (CKTN), is responsible of forming tutoring activities based on the three-seperated networks.

- CREAM Architecture

The architecture of CREAM is shown in Figure 3.1

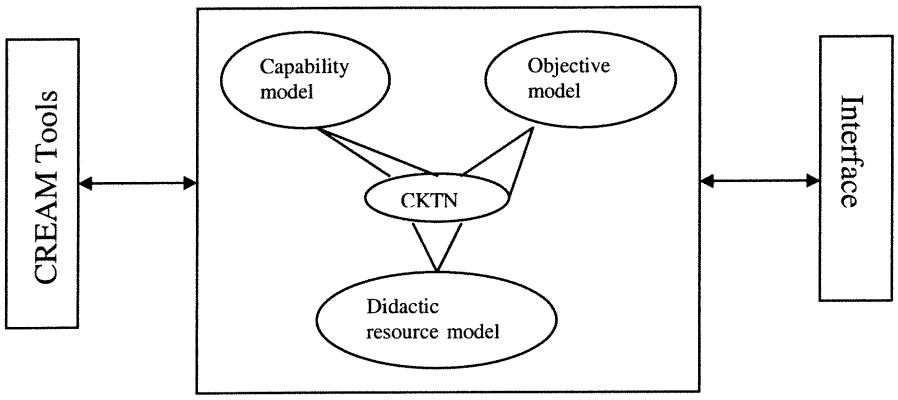

Figure 3.1 Architecture of CREAM

Four models in CREAM are shown in the central rectangle of the architecture. Capability model represents and organizes domain capabilities as learning outcomes. Objective model deals with objectives to teach capabilities. Resource model contains all kinds of didactic resources. CKTN is the core of CREAM that combines the capability, objective and resource models together to get a knowledge transition network that is used for creating tutorial actions. Two interface parts include the didactic workshop used for the acquisition of curriculum knowledge, and the interaction between CREAM and other tutorial components in SAFARI such as a planner. The CREAM based teaching development process is shown in Figure 3.2.

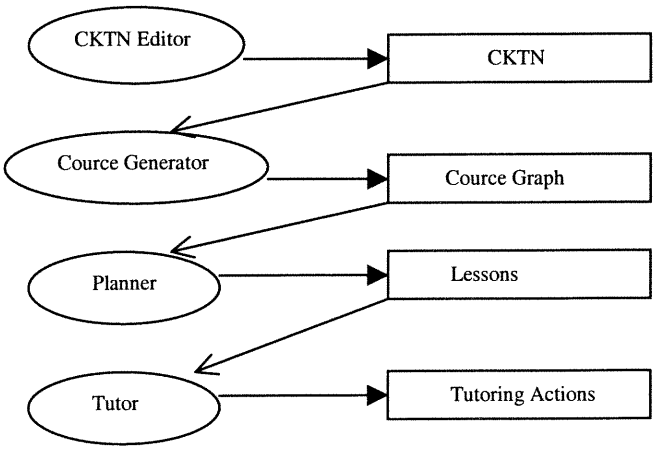

Figure 3.2 CREAM based teaching development process

• Capability Model

CREAM identifies learning outcomes based on the classification of capabilities by Gagne, intelligent skills, verbal information, cognitive strategies, motor skills and attitudes. Most of them also contain some subclasses.

In the Capability Model, CREAM defines five of relationships between capabilities including analogy, generalization, abstraction, aggregation and deviation. When build a capability space, a curriculum author should describe all relationship between capabilities.

• Objective Model

According to the instructional principle of Gagne, performance objective is to show progress toward the instructional goals. Three objectives are:

1) To provide a means for determining whether the instruction relates to the accomplishment of goals;

2) To provide a means for focusing the lesson planning upon appropriate conditions of learning;

3) To guide the development of measures for learner performance;

4) To assist learners in their study efforts;

CREAM defines an objective as a description of a set of performances that a student should be able to demonstrate after learning. It contains information about the levels according to Bloom, acquired skills, the context students should demonstrate, and success criterion.

The context in an objective refers to the concrete means to guide the tutoring. The attribute Criterion of Success gives the standard for evaluating the acquisition levels of students, which depends on the specific context attributes.

There are three main relationships among objective, this is, prerequisite relationships, pretext relationships and contribution relationships. The prerequisite relationships may be also classified as obligatory prerequisite desirable prerequisites.

• Didactic Resource Model

Didactic resources comprise all kinds of tutoring materials that support the tutoring to achieve certain objectives for acquiring certain capabilities.

In CREAM, didactic resources are classified into three types: tutorial resources, intelligent resources and dumb resources. Tutorial resources refer to the resources how to guide tutorial activities. Intelligent resources consist of problems, demonstrations, exercises, electronic documents, intelligent videos, tests, exams, etc. Dumb resources are primitive tutorial materials such as texts, images sound, static multimedia and physical resources, as well as the dynamic resources, for instance, animation, simulations, hypermedia, and dynamic physical resources. Organization of resources is based on the following six types of relations: similarity (analog), abstraction, particular cases, utilization, auxiliary and equivalence.

• Partial Association Spaces

In order to carry effective tutoring, CREAM identifies partial association spaces between models: Capability->Object (C-O), Objective->Capability (O-C), and Objective->Resource (O-R).

The C-O association contains two prerequisite relationships, i.e. obligatory prerequisite and desirable prerequisites. Contribution relationship from objectives to capabilities, in O-C association, contains three levels: strong, medium and weak contributions. They represent the degree that an objective supports some capabilities. The O-R association reflects the relations between objectives and resources. There are two relations in this subspace, i.e. critical resources and accessory resources. The accessory resource is used for the alternative resources when a student cannot achieve certain capabilities after the critical resources have been used.

• Global association

CKTN (Concept Knowledge Transition Network) is a global association space that combines the above three models together. In CKTN a transition node connects capability nodes, objective nodes and resource nodes. The relationship from a capability node to an objective node is a prerequisite relation, and from an objective node to a capability node with a contribution relation. Resource nodes links to objectives are real materials of tutoring activities.

Characteristics and limitations of CREAM

Compared with most tutoring systems, CREAM exploits general-purpose curriculum development environment and enhances the curriculum as the central role in ITS. This motivation is more advanced than specific-domain systems. Learning outcomes in CREAM are capability theories of Gagné that reflects the conditions of human learning. CREAM uses multiple spaces to separate capabilities, objectives and resources explicitly, which make it possible to develop

a curriculum in parallel. Meanwhile, this enhances flexibility of knowledge utilization in teaching delivery stage. For example, analogy relationships between capabilities in Capability Model can be used by tutor module to remedy learners' errors.

However, some limitations exist in CREAM. At first, multiple spaces increase the complexity of system architecture and realization. With CREAM, a real curriculum would consist of a number of nodes. For instance, the curriculum for teaching EXCEL includes 500 capability nodes. Both learners and curriculum authors are easy to lose in such large space. If the system can supplies visual navigation ability, it will enhance usability of system. Second, few functional component can be shared with other intelligent tutoring systems and files containing domain knowledge is not encoded to the format which is easy to be transferred in the Internet. If the system uses the common vocabularies with others' and the established domain knowledge is in the Internet-oriented file format, the workload is much less to build other curriculums if some defined nodes are used. Third, the system is not suitable for running at Internet. This restricts the range of users' locations, and further increases users' inconvenience.

Chapter 4

Ontology

Although Intelligent Tutoring Systems have been discussed for many years and there have been application systems in some domains, building an intelligent tutoring system still has many difficulties. First, building an intelligent tutoring system always starts from scratch. It requires a lot of time and work. Second, knowledge embedded in most intelligent tutoring systems does not accumulate well. Third, few knowledge and functional components in intelligent tutoring systems are reusable and can be shared with other intelligent tutoring systems. Recently, researchers have designed and developed some authoring tools for building intelligent tutoring systems. With the help of these tools, the efficiency of building an intelligent tutoring system has been improved a lot, but it still needs a lot of work. It is still not easy for authors to represent the knowledge in the form specified by the tool. In order to design an Intelligent Tutoring System (ITS), designers have to know what instructional theories and models provide appropriate principles and strategies, what functional components are necessary to reach instructional objectives, what pedagogical actions are appropriate for each situation, what architecture and strategy are appropriate to support these actions, how to achieve a coherent learning support process, and how to organize domain knowledge. However, these fundamental characteristics of an ITS is often implicit or ill formed. From the knowledge engineering point of view, this was because these authoring tools did not provide a sophisticated vocabulary that enable him/her to represent the knowledge about the tutorial system at a right level of abstraction. To solve this problem the ontology engineering is a good candidate.

ontology engineering is a research methodology that gives us the design rationale of a knowledge base, kernel conceptualization of the event we are interested in, the strict definition of basic concepts together with sophisticated theories and

technologies enabling accumulation of knowledge which is dispensable for modeling the real world. ontology consists of task ontology, which characterizes the computational architecture of knowledge-based systems, and domain ontology, which characterizes the domain knowledge.

We begin the chapter by discussing what ontology is. How to use ontology is one of the crucial issues in ontology research. Therefore, we analyze the ontology in eight levels followed by the discussion on what concrete advantages ontology can give in the real-world problem. The next topic is the classification of ontologies. On the basis of the discussion made thus far, we present the scope of ontology engineering. Finally, we exemplify ontology engineering by summarizing state of art.

4.1 What is ontology?

4.1.1 Simple definitions

Three simple definitions extracted from literature are given below.

(1) Ontology is a term in philosophy and the meaning is "theory of existence".

(2)A definition of ontology in AI community is "An explicit representation of conceptualization"[Gruber, 92].

(3) A definition of ontology in knowledge base community is "a theory of vocabulary/concepts used as building artificial systems"[Mizoguchi, 93].

The above definition can give us a general idea but it is not sufficient for in-depth understanding what ontology is. A more comprehensive definition is given in the section 4.1.2.

4.1.2 Comprehensive definitions

(1) Ontology: Capital letter "O" is used to distinguish the "Ontology" in philosophy from other area. "Ontology" is a theory which can answer questions such as "what is existence", "What properties can explain the existence", "How these properties explain the differences of existence", etc.

(2) ontology: According to Mizoguchi, the definition of ontology is an explicit and less ambiguous description of concepts and relations among them appearing in the target object.

(3) Formal ontology: Axiomatic description of ontology. It can answer questions about the capability of ontology.

4.2 Typology of ontology

We here discuss this issue considering the usage levels of ontology. According to [Mizoguchi, 2000], ontology is divided into the following types from the knowledge reuse point of view as follows:

Ontology:

Workplace ontology[Vanwelkenhuysen 94, 95b]

This ontology is for workplace. It affects task characteristics by specifying several boundary conditions, and characterizes and justifies problem-solving behavior in the workplace. Workplace ontology collectively specifies the context in which domain knowledge is interpreted and used during the problem solving. *Examples (Circuit troubleshooting): Fidelity/Efficiency/Precision/High reliability/etc.*

Task ontology [Mizoguchi 92, 95a][Hori 94][Wielinga 93]

Task ontology is a vocabulary system that describes problem-solving structure with all the existing tasks domain-independently. It does not cover the control structure but cover components or primitives of unit inferences that are taking place during performing tasks. Task knowledge in turn specifies domain knowledge by giving roles to each objects and relations between them. *Examples (Scheduling tasks): Schedule recipient/schedule resource/goal/constraint/availability/load/select/assign/classify/remove/relax/add/etc.*

Domain ontology

Task-dependent ontology [Mizoguchi 95b]

T-Domain ontology

A task structure requires not all the domain knowledge but some specific domain knowledge in a certain specific organization. We call this special type of domain knowledge T-domain ontology because it depends on the task. *Examples (Job-shop scheduling): Job/order/line/due date/machine availability/ tardiness/load/cost/etc.*

Task-independent ontology

Because object and activity ontologies are related to activities, we call them activity-related ontology and call field ontology activity-independent ontology.

Activity-related ontology

This ontology is related to activities taking place in the domain. There are two major activities exist in a domain. One is behavior

of an object and the other is organizational or human activity. Verbs play an important role in this ontology; however, they are different from those in task ontology. The subjects of the former verbs are objects, components, or agents involved in the activities of interest, while those of the latter are domain experts.

Object ontology[Vanwelkenhuysen, 95a]

This ontology covers the structure, behavior and function of the object. *Examples (Circuit boards): component/connection/line/chip/pin/gate/bus/state/role/etc.*

Activity ontology (Enterprise ontology)[Gruninger, 94]

*Example*s:use/consume/produce/release/state/resource/commit/enable/complete/disable/etc.

Activity-independent ontology

Field ontology

This ontology is related to theories and principles, which govern the domain. It contains primitive concepts appearing in the theories and relations, formulas, and units constituting the theories and principles. *Exemples(units): mole/kilogram/meter/ampere/radian/etc.*

General/Common ontology

Examples: Things/Events/Time/Space/Causality [Lenat, 90]

4.3 Roles of ontology engineering

4.3.1 ontology as a design rationale

In the mechanical design, previous designs are often used as a reference for new products designer. One of the critical issues in such case is that how to understand the intentions and justifications of various decisions made by different designers. They are collectively called design rationale. Design rationale information is often implicit and the implicitness often causes difficulties in reusing the prior designs. Thus, Design rationale is as important as design drawings.

ontology plays a role similar to design rationale in reusing knowledge bases. In order to reuse knowledge in a knowledge base, we have to know underlying conceptualization, which reflects the assumptions and requirements in the problem using the knowledge base. Although many knowledge bases have been built to date, no such information has been described. ontology as the design rationale information of knowledge bases will contribute to reuse knowledge bases and play the roles of backbones on knowledge bases. The future knowledge bases should be built with explicit representation of ontologies.

4.3.2 How to use ontology

Although there have been many discussions on ontology, how to use it has not been fully discussed. [Mizoguchi, 2000] divided the usage of ontology into eight levels. The following is a list of how to use ontology.

Level 1: Used as a common vocabulary for communication among distributed agents.

Level 2: Used as a conceptual schema of a relational database. Structural information of concepts and relations among them is used. Conceptualization in a database is conceptual schema. Data retrieval from a database is easily done when there is an agreement on its conceptual schema.

Level 3: Used as backbone information for a user of a certain knowledge base.

Level 4: Used for answering competence questions.

Level 5: Standardization

5.1 Standardization of terminology (at the same level of Level 1)

5.2 Standardization of meaning of concepts

5.3 Standardization of components of target objects (domain ontology).

5.4 Standardization of components of tasks (task ontology)

Level 6: Used for transformation of databases considering the differences of the meaning of conceptual schema. This requires not only structural transformation but also semantic transformation.

Level 7: Used for reusing knowledge of a knowledge base using design rationale information.

Level 8: Used for reorganizing a knowledge base based on design rationale information.

4.3.3 Functions of ontology

As discussed before, ontology engineering does make a critical contribution to the innovative knowledge processing technology in the information society. We need to know how to treat real-world knowledge effectively. To enable just-in-time learning, materials need sophisticated knowledge processing. Semantically annotated documents with semantic tags that are carefully designed in a principled manner.

First of all, ontology provides a set of terms that should be shared among people in the community, and hence could be used as well-structured shared vocabulary.

These terms enable us to share the specifications of components' functionalities, tutoring strategies and so on and to compare different systems properly.

ontology explicitly represents the underlying conceptualization that has been kept implicit in many cases. ontology is composed of a set of terms and relationships with formal definitions in terms of axioms. Such axioms are declarative, and hence such ontology represents the conceptualization declaratively. Thus, ontology is the source of intelligence of ontology-based system.

Another role of ontology is to act as a meta-model. A model is usually built in the computer as an abstraction of the real target. Ontology provides concepts and relationships that are used as the building blocks of the model. Axioms give semantic constraints among concepts. Thus, ontology specifies the models to build by giving guidelines and constraints that 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.

A shared ontology is the first step towards standardization. Not only informal definitions of terms/concepts but also intermediate concepts are made explicit by ontology. The structure usually employs *is-a* and *part-of* links to related concepts. The structure obtained in ontology represents an understanding about the domain of the developer. It is usually much more informative than definition of a term. Ontology cannot instantly become a standard but it 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.

4.4 Scope of ontology engineering

ontology engineering should cover philosophy, knowledge representation, ontology design, standardization, EDI, reuse and sharing of knowledge, media integration... which are the essential topics in the future knowledge engineering. Definitely, they should be constantly refined through further development of ontology engineering.

Basic division

-Philosophy (Ontology, Meta-mathematics)

ontology which philosophers have discussed since Aristotle is discussed as well as logic and meta-mathematics.

-Scientific philosophy

Investigation on Ontology from the physics point of views, e.g., time, space, process, causality, etc. is made.

-Knowledge representation

Basic issues on knowledge representation, especially on representation of ontological stuff, are discussed.

Division of ontology design

-General (Common) ontology

General ontology such as time, space, process, causality, part/whole relation, etc. are designed. Both in-depth investigations on the meaning of every concept and relation and on formal representation of ontologies are discussed.

-Domain ontology

Various ontology in, say, Plant, Electricity, Enterprise, etc. are designed.

Division of common sense knowledge

-Parallel to general ontology design, common sense knowledge is investigated and collected and knowledge bases of common sense are built.

Division of standardization

-EDI (Electronic Data Interchange) and data element specification

Standardization of primitive data elements should be shared among people for enabling full automatic EDI.

-Basic semantic repository

Standardization of primitive semantic elements should be shared among people for enabling knowledge sharing.

-Conceptual schema modeling facility (CSMF)

-Components for qualitative modeling

Standardization of functional components such as pipe, valve, pump, boiler, register, battery, etc. for qualitative model building.

Division of Data/knowledge interchange

-Translation of ontology

Translation methodologies of ontology into another are developed.

-Database transformation

Transformation of data in a database transforms into another of different conceptual schema.

-Knowledge base transformation

Transformation of a knowledge base into another built based on a different ontology.

Division of knowledge reuse

-Task ontology

Design of ontology for describing and modeling human ways of problem solving.

-T-domain ontology

Task-dependent domain ontology is designed under some specific task context.

-Methodology for knowledge reuse

Development of methodologies for reusable knowledge uses the above two ontology.

Division of knowledge sharing

-Communication protocol

Development of communication protocols between agents that can behave cooperatively under a goal specified.

-Cooperative task ontology

Task ontology design for cooperative communication

Division of media integration

-Media ontology

ontology of the structural aspects of documents, images, movies, etc. are designed.

-Common ontology of content of the media

ontology common to all media such as those of human behavior, story, etc. are designed.

-Media integration

Development of meaning representation language for media and media integration through understanding media representation are done.

Division of ontology design methodology

-Methodology

-Support environment

Division of ontology evaluation

Evaluation of ontology designed is made using the real world problems by forming a consortium.

4.5 How ontology provide solutions

As stated before, ontology consists of the task ontology which characterizes the computational architecture of a knowledge base system which performs a task, and the domain ontology which characterizes the domain knowledge where the task is performed, such as diagnosis, monitoring, scheduling, or design. Instruction is a task, as is supporting the learning process. Task ontology might provide an effective methodology and vocabulary for both analyzing and synthesizing knowledge base systems to which ITSs belong. The benefits of

using ontology are: a common vocabulary, making knowledge explicit, systematization, standardization, and meta-model functionality. This functionality suggests the possibility of an ontology-aware authoring functionality that could be very intelligent in the sense that it would know what model would help authors.

Since an ITS needs terms/concepts concerning pedagogical actions to ground the functionality in concrete actions, the justification should be given by theories, and the source of intelligence of systems should come from the knowledge bases containing this knowledge. Easy access to educational theories would be valuable to both human and computer agents. ontology Engineering helps specify higher-level functionality of ITSs: it bridges the gap between human knowledge and knowledge in the knowledge bases. ontology could pave the way for the building of an authoring environment for ITSs. An authoring agent could explain relevant theories in response to an author's request; it could give the author some possible justifications for teaching and learning strategies from a theoretical point of view.

In order to do meaningful mediation using common vocabulary, the first challenge is to have computers mediate the sharing of our knowledge, with a common vocabulary for representing the knowledge. The level 1 ontology plays a sufficient role for this goal, which is to share primitive concepts in terms that can describe the knowledge and theories. The second challenge is to extend this sharing from among humans to among computers. Level 2 introduces definitions of each term and relations richer than in level 1 by using axioms. An axiom relates a couple of concepts semantically, which makes computers partially understand the rationale of the configuration of the interest events. The operationalization of this knowledge leads to the building of IIS s. This requires 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). The knowledge server communicates with humans who need help in finding knowledge appropriate for their goals. Thus, such authoring environments can discuss with authors about the appropriateness of strategies

adopted with the help of the knowledge server. ITSs developed in this way would behave in a seamless flow of knowledge from designers onto learners.

4.6 Survey on the research of ontology

4.6.1 Theory

N. Guarino and J. Sowa have been independently conducting research on theories of ontology. Both share the attitude towards philosophy and both incorporate the results obtained in philosophy as principles to design the top-level ontology. In the case of building ontology for a large-scale knowledge base, the validity of the knowledge base necessarily is justified in terms of wider rage of tasks, that is, it needs to show its generality rather than task-specific utility. Compliance with the principled top-level ontology provides a good justification. Thus, top-level ontology is important.

Sowa's ontology [Sowa 95, 98] is based on J. S. Peirce's idea. It is defined without assuming such things as human, iron, etc and provides an environment or context like family, school, etc. He introduces two important concepts, continuant and concurrent in addition to the three and obtains 12 top-level categories by combining the seven primitive properties.

Guarino's [Guarino, 97] top-level ontology is more extensively incorporates philosophical consideration. It is designed based on mereology (theory of parts), theory of identity, and theory of dependency. His ontology consists of two subjects: ontology of Particulars such as things that exist in the world and universals that include concepts we need when we describe Particulars.

4.6.2 Machine-readable dictionary: MRD

Development of machine-readable dictionaries has been done extensively in natural language processing community where ontology has also been discussed as the upper level model of the words/concepts structure. Typical examples include WordNet [Miller, 93] [WordNet], EDR [Yokoi, 95], EuroWordNet [EWN], Generalized Upper Model [GUM].

4.6.3 Metadata, XML, Tag, and intelligent player

Another activity to note is that about standardization about metadata such as Dublin Core [DC 97], MCF: Meta Content Framework [MCF 97] and RDF: Resource Description Framework [RDF 97] in W3C. Dublin Core is the first de facto standard of the metadata descriptors. MCF was proposed to W3C as a candidate of the framework for metadata representation and extended and elaborated into RDF that is becoming as a standard. RDF introduces XML for its syntax.

XML is a simplified version of SGML, which is a powerful markup language definition language and has been widely used in document description for years. XML is equipped with several powerful hyper-reference functions to fit the Internet environment. The good things of XML is that the users can define their own tags which shows not only structure information of the document but also its semantic information for various uses of the document to enable semantic interoperation. For example, we can design an intelligent instructional player of a teaching material as an XML document by sharing a set of tags for explicating the roles of the portion of document and controlling the interpretation. It can "Play" the XML document adaptively to the performance of the learners. Further, we could formalize the set of tags to specify the performance of such instructional players. While tags in metadata description generally form a level one ontology, such tags can be those at the second level. Intelligent players with plug & play capability with shared XML tags is expected to be promising with Java implementation.

4.6.4 Developing methodologies

We here present some of the developing methodologies and environments.

IDEF5 [IDEF5]

Knowledge Based Systems Inc. has developed a comprehensive methodology for developing ontology, IDEF5, especially for enterprise modeling together with graphical and representation languages. IDEF5 tries to follow the normal and typical methodology and adopts the steps such as term extraction, term definition, and relation among terms extraction followed by formalization of them. Documentation is well done and contains helpful guidelines to follow.

TOVE [TOVE]

TOVE is also one for enterprise ontology like IDEF5. Its remarkable characteristics include:

1.Requirements to the ontology are formulated, as "competency questions" which the resulting model based on the ontology has to answer. Thus, competence of the ontology is formally specified.

2.In TOVE, axioms are represented in Prolog and answer the competency questions.

The approach is thus very formal and follows the standard approach in software engineering. TOVE thus axiomatizes time and activity to be able to investigate many characteristics of the model through conceptual-level execution.

DODDLE, Aspect theory and CLEPE&AFM

DODDLE (A Domain Ontology rapiD DeveLopment Environment) [Yamaguchi 97] is an environment for ontology development that has been used to build a legal ontology. Its characteristic is that WordNet is incorporated in the

environment as the upper model to give the users guidelines for identifying and defining domain-specific concepts.

Aspect theory [Takeda, 95] is a framework for combining several ontologies built based on different viewpoints. ontology is called an Aspect and a set of logical operation among them is defined to form a theory. ASPECTROL is a language to implement the theory.

CLEPE [Seta 97] is an environment for task ontology [Mizoguchi, 93], a level 3 ontology. It is designed assuming three kinds of users such as basic ontology authors, task ontology authors and task model authors. Models are built under the guide of task ontology to maintain their consistency.

AFM: Activity-First Method [Mizoguchi 95a,b] is an environment for ontology building where task analysis is first done to elicit verbs that specify objects necessary for performing the task of interest. Nouns used as objects of the verbs are then extracted with the roles played by them. The building process begins by document analysis managing several intermediate products obtained during the course of ontology building to enable sophisticated support. The ontology built by AFM is of the level 2. AFM is going to be augmented to enable users compare multiple ontologies built by different persons to come up with a partial agreement of them and to build a maximally agreed ontology.

Chapter 5

Project concept description

The better effect for students has gained by using face-to-face tutoring style, but it is hard to be realized in most situations. In an attempt to bring some of the benefits of the face-to-face tutoring style to a broader audience, researchers have produced computer-based intelligent tutoring systems with the goal to let computers mimic human tutors. Nonetheless, this endeavor has existed for over a quarter of a century with little impact on traditional educational practice. In addition to the difficulties such as the expense and complexity involved in building systems to behave like human tutors, disseminating such applications to large audiences is a very tough work. In the middle of 1980s, the occurrence of the World Wide Web offers an unprecedented opportunity for computer–based intelligent tutoring systems and also stimulates the enthusiasm of both researcher and producer. With the easily accessed World Wide Web, we now have appropriate conditions to combine the newest computer technology and the personal tutor experience to a broader audience in a variety of settings in and out of the classroom.

With the World Wide Web, people can upload knowledge nodes developed by themselves and download knowledge nodes done by others. By using the World Wide Web, our progress to set up curriculum can be accelerated and avoid doing some jobs which must be done with standalone tutoring system. With the help of World Wide Web, we can develop the parts that are unique for us, find other common parts on the web, and then establish links among them. The aim of our project is to implement such an authoring tool that can allow a domain expert, in the Internet, to establish the domain knowledge. It uses visual means to satisfy the domain knowledge design and further help curriculum authors and assist tutors. In this chapter, we focus our efforts on giving a concept description on our project.

5.1 Technical summary

The following items list the technical summary of our project.

(1) ontology is used to define the domain knowledge.

(2) Document Type Descriptor (DTD) embedded into the XML file is employed to represent defined domain knowledge nodes.

(3) JAVA is used to develop an authoring tool, which enables users in the World Wide Web to create a domain knowledge based on the ontology defined in (1).

4) XML-Query engine is used to extract information from XML-files either from local machine or from outside servers.

(5) The links can be set up among knowledge nodes either developed by users or imported from outside.

(6) On the basis of the established links, the contents of domain knowledge nodes can be browsed can editable.

(7) The established domain knowledge can be either in the relational database tables or encoded into XML format.

5.2 Technical considerations and selections

In this section, we will give reasons why we chose some techniques to be used in the project and illustrate these techniques in detail.

5.2.1 Why choose ontology

Knowledge is domain-dependent, and hence knowledge engineering that directly investigates such knowledge has been suffering from rather serious difficulties, such as domain-specificity and diversity. Further, much of the knowledge in

expert systems is heuristics, which makes knowledge manipulation more difficult [Mizoguchi 00]. However, in ontological engineering, we investigate knowledge in terms of its origin and elements from which knowledge is constructed. Hierarchical structure of concepts and decomposability of knowledge enable us to identify portions of concepts sharable among people. Exploitation of such characteristics makes it possible to avoid the difficulty knowledge engineering is facing. The following is an enumeration of the benefits we can get from using ontology engineering:

1. *Common vocabular*ies. The vocabularies to describe the target objects are agreed by people involved.

2. *Explication.* Presuppositions/assumptions are usually implicit. Any knowledge base approach is based on a conceptualization possessed by the builder and is usually implicit. But ontology can explicate the representation of assumptions and conceptualization.

3. *Systematization of knowledge.* Knowledge systematization requires well-established vocabulary/ concepts to describe phenomena, theories and objects. ontology can provide a backbone for the systematization of knowledge.

4. *Standardization.* The common vocabularies and knowledge systematization bring us standardized terms/concepts.

5. Sharing. Sharing common understanding of the structure of information among people or software agents is one of the more common goals in developing ontology. For example, suppose several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology on the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to give more complete answers according to users' queries.

6.Reuse of domain knowledge. Enabling reuse of domain knowledge was one of the driving forces behind recent surge in ontology research. For example, models for many different domains need to represent the notion of time. This representation includes the notions of time intervals, points in time, relative measures of time, and so on. If one group of researchers develops such ontology in detail, others can simply reuse it for their domains. Additionally, if we need to build a large ontology, we can integrate several existing ontologies describing portions of the large domain.

Since the above benefits, we decide to use ontology to represent domain knowledge in our project.

5.2.2 Why use XML?

HTML, XML are markup languages, which make the plain information on the web into computer-understandable data. HTML is extensively used by people who are willing to make tagged document on the web. HTML makes computer-processable documents only in the sense of display structure of them on the screen. However the following defects of HTML prevent itself to be used to define ontology.

- *HTML is a presentation technology only.* HTML does not necessarily reveal anything about the information to which HTML tags are applied. For example, we know that <h2>Apple</h2> has a definite, predictable appearance in a web browser, but is it a computer company? A fruit? A last name? A recording company? HTML doesn't tell. HTML tag names don't describe what content is. They only imply how content appears.
- *HTML has a fixed tag set.* You can't extend it to create new tags that are meaningful and useful to you and others. Only the W3C (World Wide Web Consortium) can do that.

- *Web browsers were viewed as potential application platforms*, but Java technology frankly needs more to chew on than HTML offered in order to fulfill that vision.

XML makes documents' semantic structures understandable by a computer to some extent, which makes an extraordinary difference in the value of information on the web. As presented before, tags in HTML are concerned only with structural information such as headings, fonts, itemization, etc., those in XML family are for representing any information about the document, typically some amount of its meaning such as author, topics dealt with, etc. XML tags have two major advantages over HTML:

(1) explication of class (e.g., city) for each specified text (e.g., Montreal) in the document

(2) define arbitrary "data structure" for interpretation of the multiple fragments of texts. These lead us to the idea of metadata that is a data of the data (document) and are extensively used for information retrieval on the web. By using XML tags, users can explicate semantic information explicitly contained in the document as much as possible.

Except arbitrary extension of document's tags and attributes, XML provides many good things such as storage (XML document), schemas (DTDs, XML schema language) query language, programming interfaces (SAX, DOM) and so on. All of these are used in the project and will be presented in the following.

DTD

A DTD (Document Type Descriptor) is a formal description in XML Declaration Syntax of a particular type of document. It sets out what names are to be used for the different types of element, where they may occur, and how they all fit together. For example, if you want a document type to be able to describe `Lists`

which contain `Items`, the relevant part of your DTD might contain something like this:

```
<!ELEMENT List (Item)+>

<!ELEMENT Item (#PCDATA)>
```

This defines a list as an element type containing one or more items (that's the plus sign); and it defines items as element types containing just plain text (Parsed Character Data or PCDATA). Validating parsers read the DTD before they read your document so that they can identify where every element type ought to come and how each relates to the other, so that applications which need to know this in advance (most editors, search engines, navigators, databases) can set themselves up correctly. We can create the list like:

```
<List><Item>Chocolate</Item><Item>Music</Item><Item>Surfing</Item>
</List>
```

How the list appears in print or on the screen depends on your style sheet: you do not normally put anything in the XML to control formatting like you had to do with HTML before style sheets. This way you can change style easily without ever having to edit the document itself.

A DTD provides applications with advance notice of what names and structures can be used in a particular document type. Using a DTD when editing files means you can be certain that all documents which belong to a particular type will be constructed and named in a consistent and conformant manner.

With DTD, the XML file can carry a description of its own format with it. With DTD, independent groups of people can agree to use a common DTD for interchanging data. We also can use standard DTD to verify that the data received from the outside world is valid.

In this project, we use DTD to control the format of ontology we developed and embedded DTD into XML file.

SAX or DOM?

Once we have XML file, accessing information inside need the help of SAX or DOM.

What is DOM?

The Document Object Model (DOM) is a set of platform and language neutral interfaces that allow programs to access and modify the content and structure of XML documents. This specification defines a minimal set of interfaces for accessing and manipulating XML document objects. The Document Object Model is language neutral and platform independent. A DOM object is used to extract information from an XML document in a Java program (using a Java XML Parser).

What is SAX?

SAX stands for the Simple API for XML. Unlike DOM (Document Object Model) that creates a tree-based representation for the information in your XML documents, SAX does not have a default object model. This means that when creating a SAX parser and read in a document, we will not be given a nice default object model. A SAX parser is only required to read a XML documents and fire events based on the things it encounters in the XML document. Events are fired when the following things happen:

- open element tags are encountered in the document
- close element tags are encountered in the document
- #PCDATA and CDATA sections are encountered in the document
- processing instructions, comments and entity declarations, are encountered in the document.

SAX (Simple API for XML) and DOM (Document Object Model) were both designed to allow programmers to access their information without having to write a parser in their programming language of choice. By keeping the information in XML format, and by using either SAX or DOM APIs it is free to use whatever parser it wishes. This can happen because parser writers must implement the SAX and DOM APIs using their favorite programming language.

DOM gives the access to the information stored in your XML document as a hierarchical object model. Based on the structure and information in a XML document, DOM creates a tree of nodes and we can access information by interacting with this tree of nodes. The textual information in the XML document gets turned into a bunch of tree nodes. Thus DOM forces people to use a tree model to access the information in the XML document. This works out really well because XML is hierarchical in nature. This is why DOM can put all information in a tree (even if the information is actually tabular or a simple list).

In the case of DOM, the parser does almost everything such as read the XML document in, create a Java object model on top of it and then give a reference to this object model (a document object) so that people can manipulate it. SAX doesn't expect the parser to do much, SAX requires that the parser should read the XML document, and fire a bunch of events depending on what tags it encounters in the XML document. People are responsible for interpreting these events by writing an XML document handler class, which is responsible for making sense of all the tag events and creating objects in the object model.

Comparing DOM and SAX, we found that if XML documents contain document data, then DOM is a completely natural fit for the solution. For example, if a system can index and organize information that comes from all kinds of document sources (like Word and Excel files), DOM is well suited to allow programs access to information stored in these documents.

However, we are dealing mostly with structured data (the equivalent of serialized Java objects in XML), so DOM is not the best choice.

The information in our project is stored in XML documents that are machine-readable data, so SAX is the right API for giving our programs access to this information. The tasks in our project to generate machine-readable data include:

- object properties stored in XML format;
- queries that are formulated using some kind of XML query language;
- result sets that are generated based on queries (this includes data in relational database tables encoded into XML);

Since parse does not do much on SAX, we have also to write:

- the custom object model to "hold" all the information in the XML document.
- a document handler that listens to SAX events and makes sense of these events to create objects in the custom object model.

5.2.3 XML query language

Given its flexibility, XML will facilitate exchange of huge amounts of data on the Web, just as HTML enabled vast numbers of documents. Dozens of application of XML already exist, including a Chemical Markup Language for exchanging data about molecules and the Open Financial Exchange for exchanging financial data between banks or banks and customers. However, the availability of huge amounts of XML data poses several technical questions that the XML standard does not address. In particular,

- How will data be extracted from large XML documents?
- How will XML data be exchanged, e.g., by shipping XML documents or by shipping queries?

- How will XML data be exchanged between user communities using the same ontologies?

XML query language is the core to address above questions. Data extraction, transformation, and integration are all well-understood database problems. Their solutions often rely on a query language, either relational (SQL) or object-oriented (OQL). These query languages do not apply immediately to XML, because the XML data differs from traditional relational or object-oriented data. XML data, however, is very similar to a data model recently studied in the research community: the semi structured data model. Several XML query languages have been designed and implemented, and they can express queries, which extract pieces of data from XML documents, as well as transformations, which, for example, can map XML data between DTDs and can integrate XML data from different sources.

In this project, we chose XQL as XML query language that was recognized by W3C (world wide web consortium) as a de facto standard for query XML documents. In order to interpret the XML files and get what we want, XML-Query Engine developed by Fat Dog Company is employed to parse XML files and extract the information inside them. XML Query Engine (XQEngine for short) is a full-text search engine component for XML. It searches small to medium-size collections of XML documents for Boolean combinations of keywords, much as web-based search engines let you do for HTML. Queries are specified using XQL that is nearly identical to the simplified form of XPath. We embed XQEngine in the class xmlquery.

5.3 Defined domain knowledge ontology

The domain knowledge is a very important part in intelligent tutoring system. To make it easier to manipulate much knowledge, it is important to characterize it in terms of a few concepts. There are two viewpoints to represent the domain knowledge of the training, that is, the educational viewpoint and the pure-domain viewpoint. From the former viewpoint, the domain concepts should be captured based on how trainees treat the topics in training context. On the other hand, from the latter, the concepts should be represented in the form of executable knowledge-representation. These two viewpoints, in principle, should be integrated into ontology.

Developing ontology has two stages, informal stage and formal stage. Most distinguish between an informal stage, where the ontology is sketched out using either natural language descriptions or some diagram technique, and a formal stage where the ontology is encoded in a formal knowledge representation language, is machine computable. As an ontology should ideally be communicated to people and unambiguously interpreted by software, the informal representation helps the former and the formal the latter.

In the informal stage, we identify the purpose of ontology under developing is for the domain knowledge. In order to exactly extract vocabularies, which can refer to concepts, relations and attributes in domains as many as possible, we study a wide range of knowledge holders such as specialist; database metadata; standard textbooks; research papers and other ontologies. The following is the expression of the ontology we developed.

Domain knowledge ontology:

Nodes

Title

Concept

Objects

Relations

Facts

Rules

Condition

Hypothesis

Conclusion

Actions

Principles

Links

NodeTitles

RelationTypes

Analogy

Generalization

Abstraction

Aggregation

Derivation

Domain knowledge DTD:

In order to let ontology be interpreted by software unambiguously, we employ the DTD to represent it.

<!DOCTYPE Domainknowledge [

<!ELEMENT Domainknowledge(Nodes+, Links+)>

<!ELEMENT Nodes(Title, Concepts, Facts, Rules, Principles)>

<!ELEMENT Title (# PCDATA)>

<!ELEMENT Concepts(Objects+, Relations+)>

<!ELEMENT Objects (# PCDATA)>

<!ELEMENT Relations (# PCDATA)>

<!ELEMENT Facts (# PCDATA)>

<!ELEMENT Rules(Conditions| Hypothesis| Conclusions| Actions)>

<!ELEMENT Conditions (# PCDATA)>

<!ELEMENT Hypothesis (# PCDATA)>

<!ELEMENT Conclusions (# PCDATA)>

<!ELEMENT Actions (# PCDATA)>

<!ELEMENT Principles (# PCDATA)>

<!ELEMENT Links(NodeTitles, RelationType+)>

<!ELEMENT RelationType(Analogy| Generalization| Abstraction| Aggregation| Derivation)>

<!ELEMENT Analogy (# PCDATA)>

<!ELEMENT Generalization (# PCDATA)>

<!ELEMENT Abstraction (# PCDATA)>

<!ELEMENT Aggregation (# PCDATA)>

<!ELEMENT Derivation (# PCDATA)>

]>

5.4 System architecture

The previous sections give a detail description on the techniques used in the software. In this section, we will use UML (Unified Modeling Language) to illustrate the software system we developed.

In the Internet there is much knowledge that has been defined. As stated before, one main benefit of using ontology is reusing some domain knowledge developed by others, so we don't need to spend time to develop each knowledge node. On the basis of the same ontology, we can set up our own domain knowledge by define some which are special for us and are not on the web, and then set links between the knowledge we developed and others on the web. The system we developed can define the knowledge nodes according to the ontology stated in the section 5.3 and establish the links between the knowledge nodes we did and outside ones in the Internet to save time and energy.

5.4.1 Use case diagram

The emphasis of use case diagram is on what the system does from the standpoint of an external observer, we want to use use case diagram to give a general picture of the system. The system's use case diagram is shown in Figure 5.1. Use case diagram are closely connected to scenarios that is what happens when someone interacts with the system. From the figure, we clearly understand there are five scenarios to be performed by the system such as

(1) Create domain knowledge nodes;

(2) Query XML files;

(3) Setup links among knowledge nodes;

(4) Browse and edit the established domain knowledge;

(5) Convert defined domain knowledge to XML files;

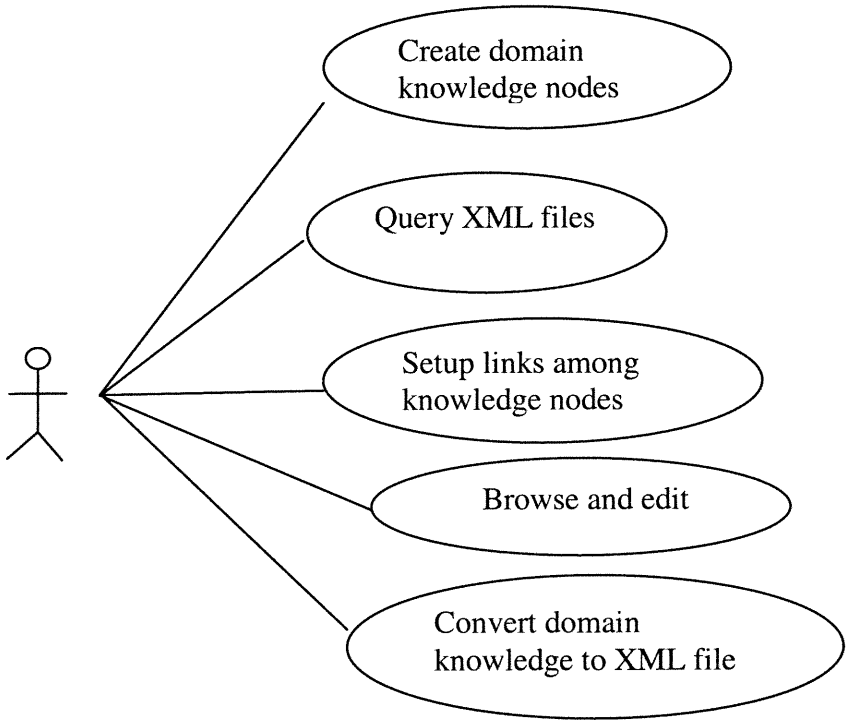

Figure 5.1 Use case diagram

5.4.2 Class diagram

Use case diagram give us what the system can do. Now we go deeper into the class level and use class diagram to describe the system. The class diagram gives us an overview of a system by showing its classes and relationships among them. Shown in the Figure 5.2, the system consists five core classes: *ontology, toxmlfile, setup_link, xmlquery and browse_edit, and* two auxiliary classes *ActionListener and ItemListener*. The navigability arrow on the figure shows which direction the association can be traversed or queried.

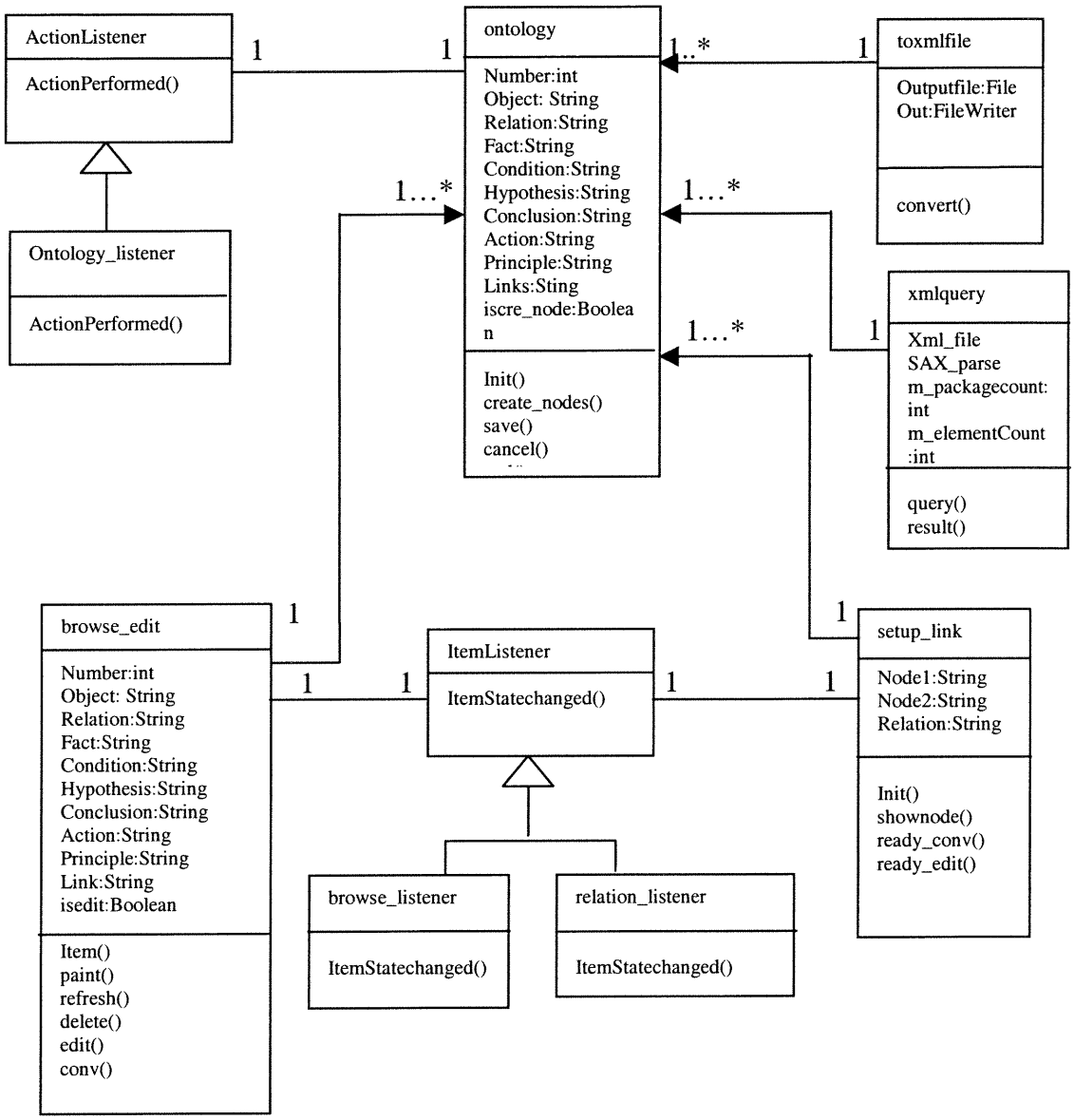

Figure 5.2 Class diagram

(1) Class *ontology* provides not only an interface for users to define the contents of domain knowledge nodes under the structure of defined ontology but also the facility to save them in the relational database tables. The class also provides functions to let users modify the contents they enter and cancel the operation when they found what they did are redundant or wrong.

(2) Class *toxmlfile* is responsible for converting the defined domain knowledge from relational database tables to XML files that can be uploaded to the Internet or used by other ITS modules.

(3) Class *xmlquery* can extract information from XML files by query them either from remote or local machines. Since XML files can be interpreted by JAVA via parse, this class is the bridge linking the outside world with the local machine. The XML query engine is embedded in the class and this class also provides the facility to store the information parsed from XML files into a database. The parse we used is SAX. The reasons why use it is illustrated in the previous section.

(4) Class *setup_link* is responsible for establishing links among knowledge nodes either from local or from remote machines. According to the ontology, the links among them have five types such as analogy, generation, abstraction, aggregation and derivation. Users can choose one of them according to the reality.

(5) After links are set, class *browse_edit* can let users to browse the whole defined domain knowledge including the graphic link representation and the contents of the knowledge nodes. Under this stage, users can check the correctness of links and contents of each node. Users can delete knowledge nodes that are redundant, delete link between nodes, and change the contents of the nodes

(6) Class ActionListener and ItemListener are the auxiliary classes to detect action events and perform some operations.

5.4.3 Packages diagram

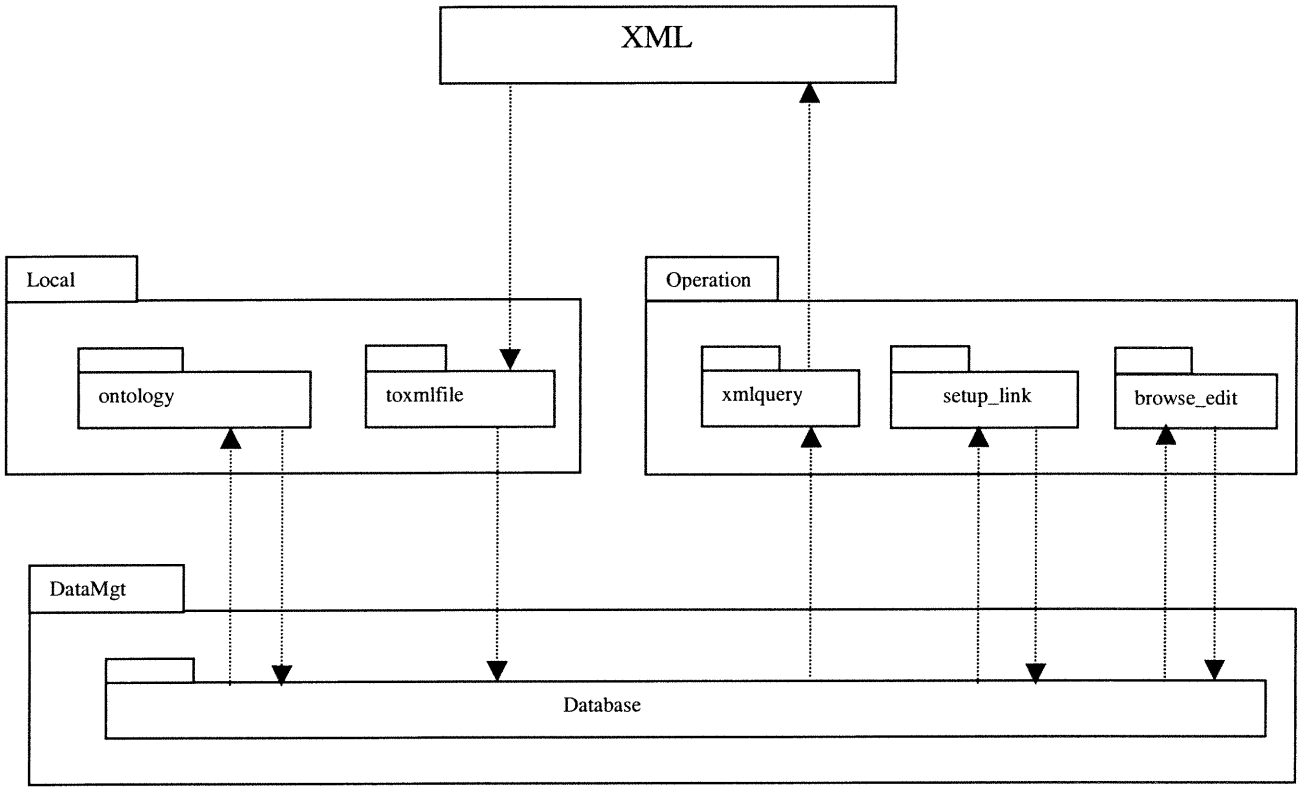

Figure 5.3 Packages diagram

To simplify complex class diagram, we group classes into packages such as Local, Operation and DataMgt. The dotted arrows in Figure 5.3 represent dependencies: one depends on another if changes in the other could possibly force change in the first. Local package is in charge of developing knowledge nodes and converting to XML file. Operation package focuses on querying XML files, setting up link, and

browse and edit established domain knowledge. DataMgt package manages the database.

5.4.4 Sequence diagram

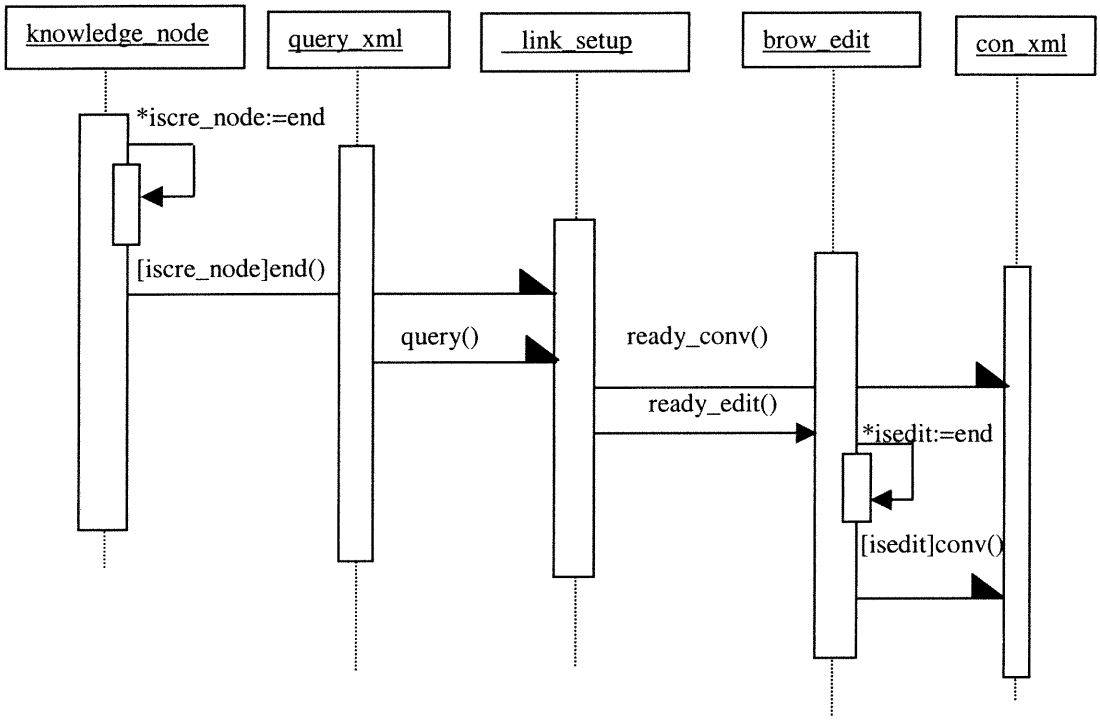

Figure 5.4 Sequence diagram

A sequence diagram is an interaction diagram that details how operations are carried out—what messages are sent and when. Sequence diagrams are organized according to time. Figure 5.4 is the sequence diagram for the system we developed.

knowledge_node, query_xml, link_setup, brow_edit and con_xml are the object of class ontology, xmlquery, setup_link, browse_edit and toxmlfile respectively. The right triangle above the lines represents asynchronous message. For example, there are two messages 1) When creating knowledge nodes is finished, the

message end () is sent to link_setup. 2) When querying XML file is done, the message query () is also sent to link_setup. The order in which these two messages are sent or completed is irrelevant. From the sequence diagram we know, the procedure of making domain knowledge nodes is independent of querying outside knowledge nodes. That means this tool can let users have three working styles: 1) create all knowledge nodes by the user self and set up links to establish domain knowledge. 2) create part of knowledge nodes by the user and get and query other part from outside, then set up links to establish domain knowledge. 3) get all knowledge nodes from outside and query them, then set up links to establish domain knowledge. All domain knowledge established from the three ways can be converted to XML file.

The entire programs are developed by Java and they reside in the server side. Users in the client side, by using web browser, interact with the programs in server. The information entered by the users in client side, such as creating domain knowledge nodes, editing domain knowledge nodes, and establishing links, are saved in the database of server. The user can select the option in the main menu to convert it to XML file.

Chapter 6

Implementation

The concept ideas of the authoring tool have been proposed in the previous chapter. The tool can setup domain knowledge and convert the established domain knowledge to XML format. When the outside knowledge nodes in XML files are suitable for our usage, we can use the tool to parse them and set up links among the knowledge nodes getting from outside and ones established by us. In addition, the tool provides visual facility to let us view the whole picture of the established domain knowledge and perform the checks and changes. In this chapter, we deal with the implementation of the authoring tool. We will use an application combined with some screen shots to explain the functions of the tool.

Application: we want to establish domain knowledge on how to calculate power (in mathematics area, i.e. the second power of 2 is 4) and output it in XML file. To be simplified, assume the index number is a positive integer and the base number is a real number. So the architecture of knowledge nodes links is as follows.

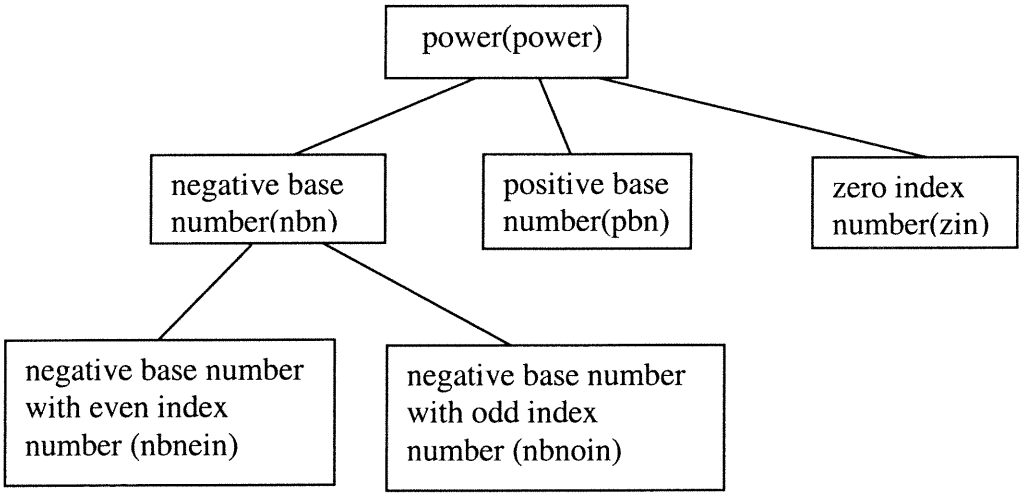

Figure 6.1 Architecture of knowledge nodes links

In Figure 6.1, the texts in parentheses like nbn, pbn... are abbreviations of node titles and they are used as node IDs later. Assume nodes pbn and zin are got from outside in XML format. So we need to create nodes power, nbn, nbnein and nbnoin. The flow chart of establishing domain knowledge on "power" is as follows.

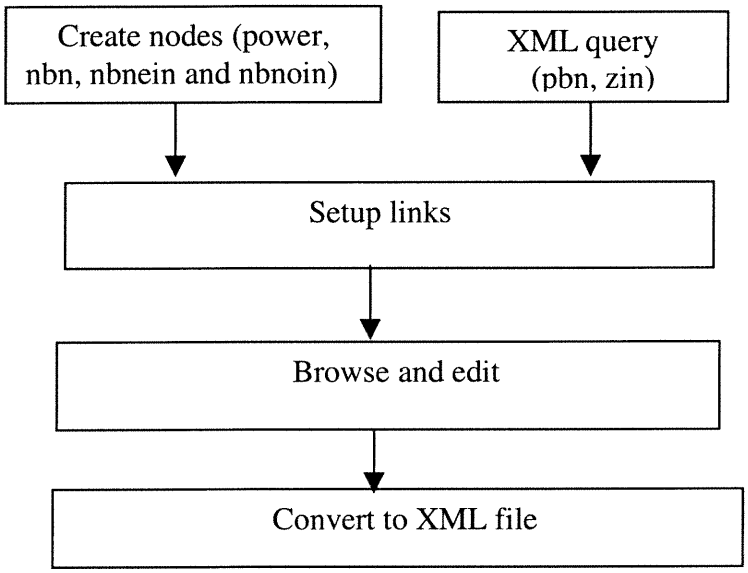

Figure 6.2 The flow chart of constructing domain knowledge

By using the application, we begin to illustrate the authoring tool by following the step shown in Figure 6.2.

When we use the Internet browser to open the tool, the main menu shown in the Figure6.3 has five options: 1) create domain knowledge nodes, 2) parse XML file, 3) establish links among knowledge nodes, and 4) browse and edit domain knowledge 5) convert to XML file.

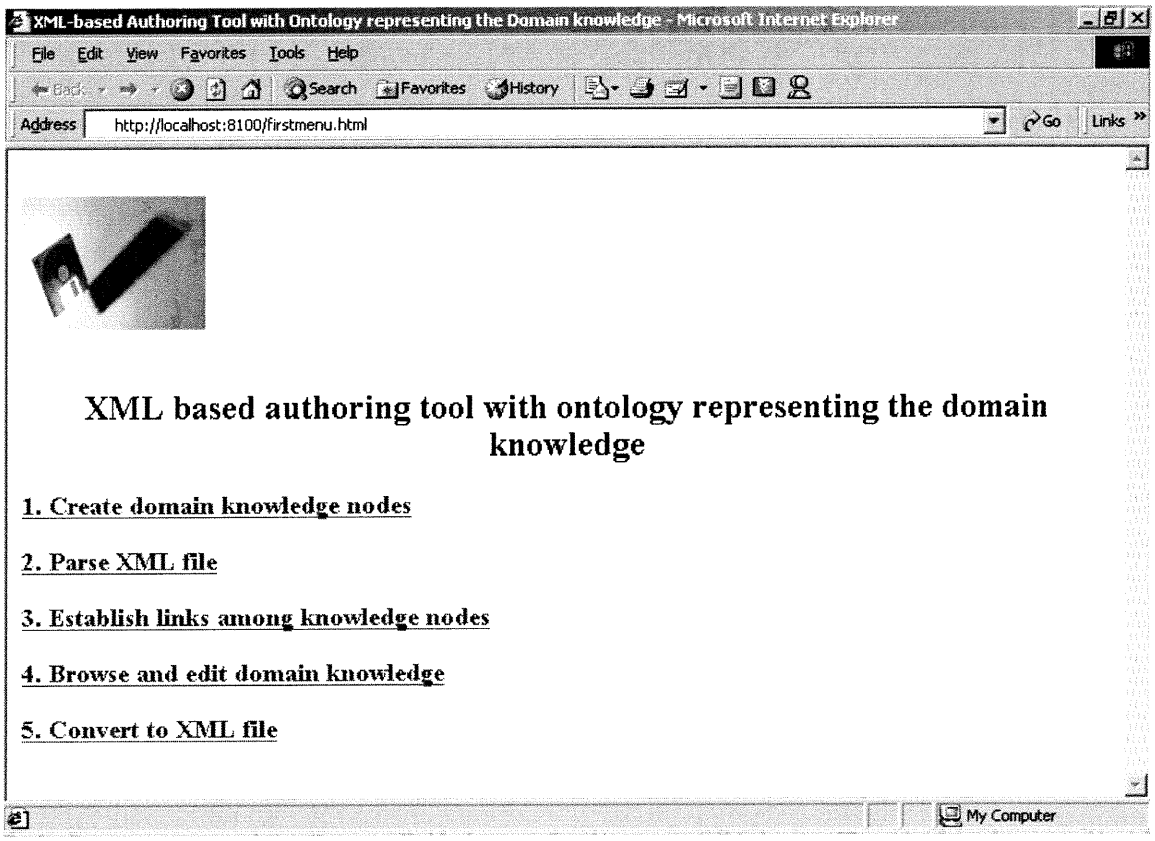

Figure 6.3 Main menu

Step 1: create domain knowledge nodes

When we click the first option, *create domain knowledge nodes*, the new web page is popped up. In the new web page (Figure 6.4), the ontology we defined previously to describe the domain knowledge appeared on the left of the page. At this time we type the content on the blank to define the first knowledge node *power*, according to the ontology. The content of node *power* are shown in Figure 6.4. Once finishing entering content, we can press NEXT to save what we typed and to create another node with blank areas waiting for typing. If we want to

cancel what we typed before save, just press CANCEL and the text fields on this node will be blank. After finishing creating knowledge nodes *power, nbn, nbnein, nbnoin*, just click the cross on the top right to close the page.

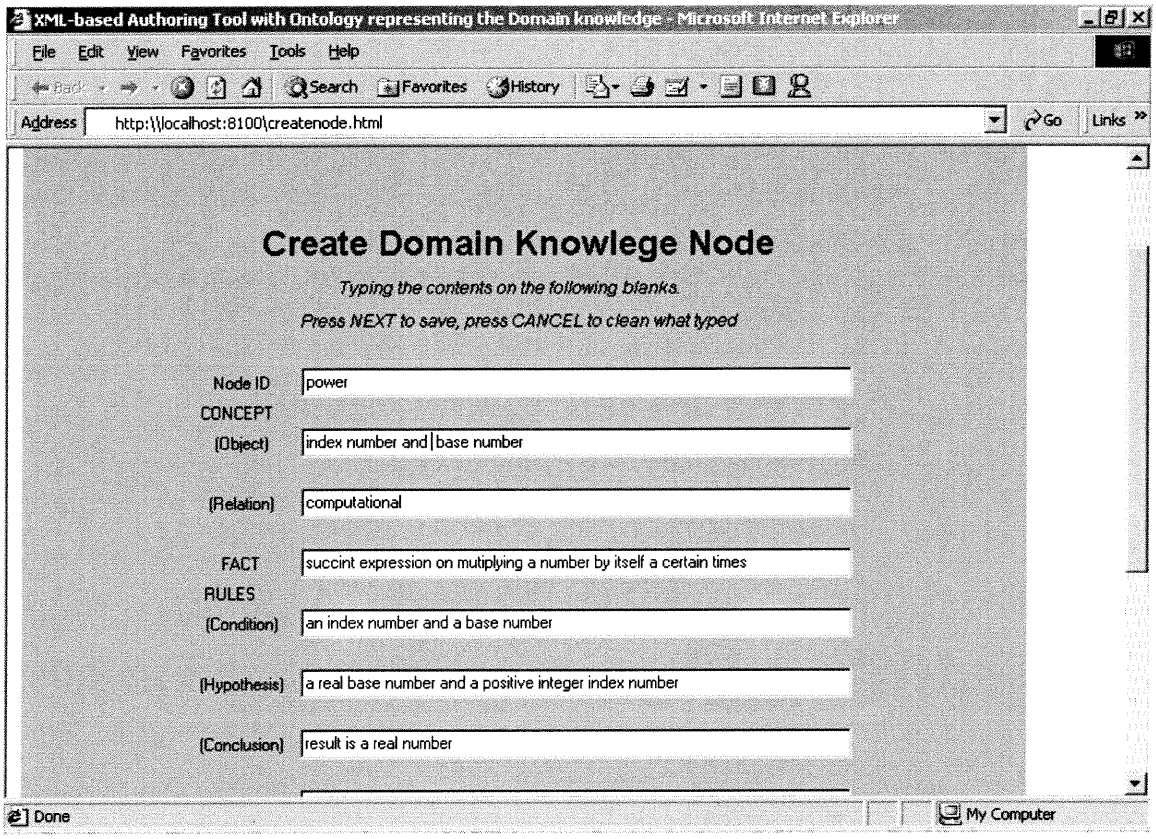

Figure 6.4: Create domain knowledge node (1/2)

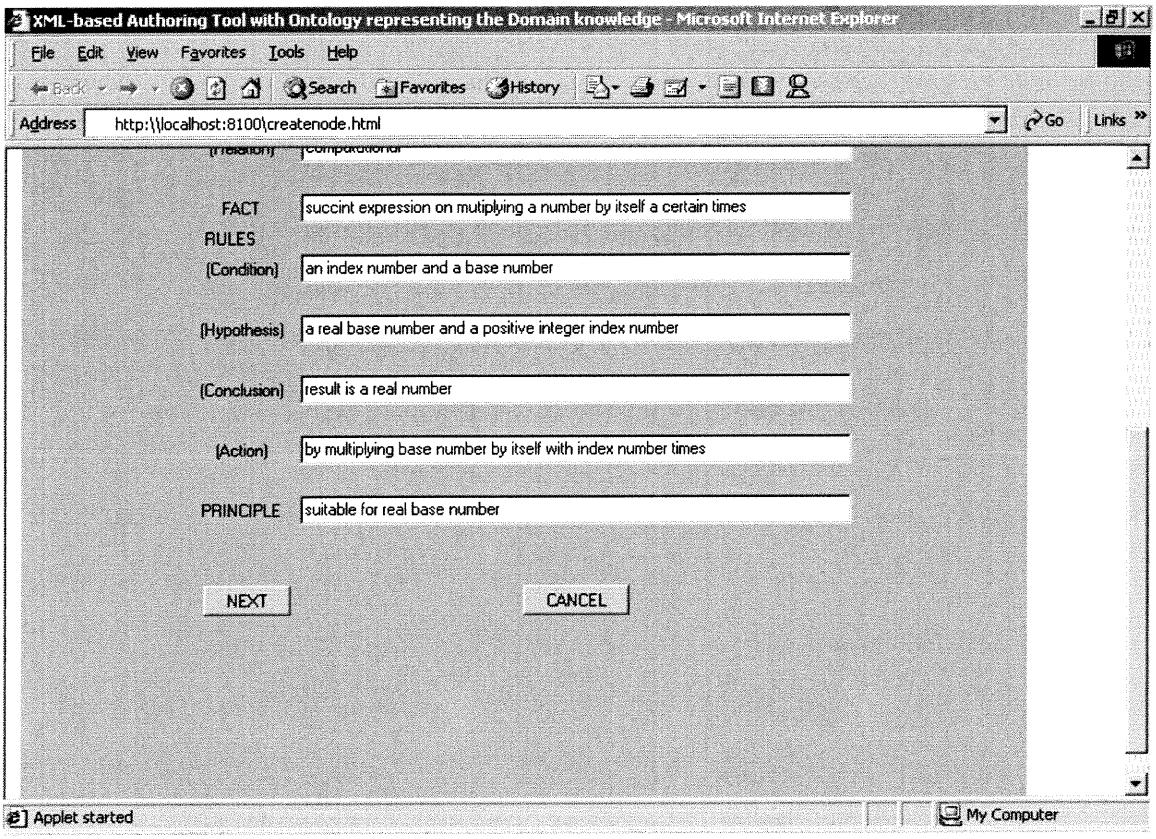

Figure 6.4: Create domain knowledge node (2/2)

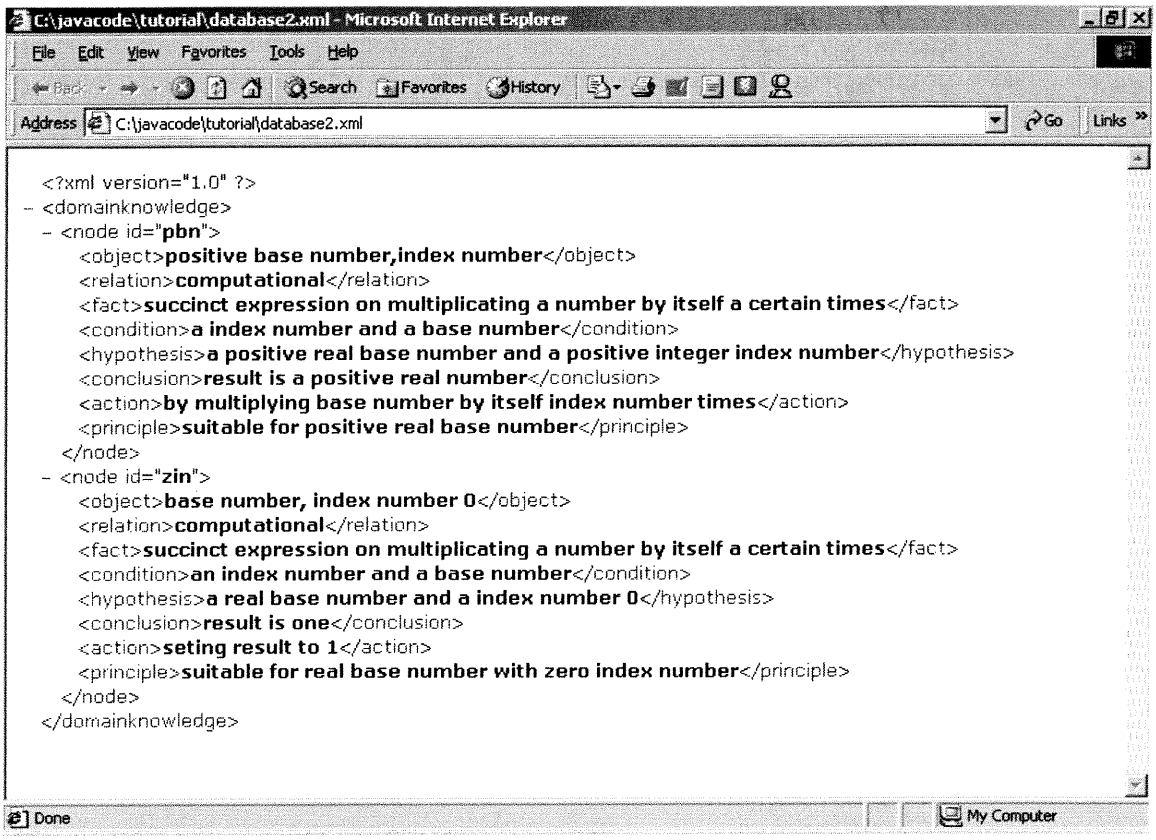

Figure 6.5 XML file from outside

Step 2: Parse XML file

In the Internet there are many defined knowledge nodes in XML format and suppose the knowledge nodes *pbn* and *zin* are in the Internet. The downloaded XML file is shown in Figure 6.5. We select option 2 to let the tool to parse it and get information. The program will save the queried information to the database.

Step 3: Establish links among knowledge nodes

When we create the knowledge nodes and get outside knowledge nodes, we will click the third option to establish the links among them. As shown in Figure 6.6, all knowledge node IDs appear on the left choice box. When we select one from the left choice box, all IDs except the one selected in the left choice box will appear in the middle choice box. Finishing selecting two IDs, we click the rightmost choice box to choose what relation type they have. Five relation types can be chosen such as analogy, generalization, abstraction, aggregation and derivation. According to Figure 6.1 we establish their links as generalization respectively and press Save to save the links.

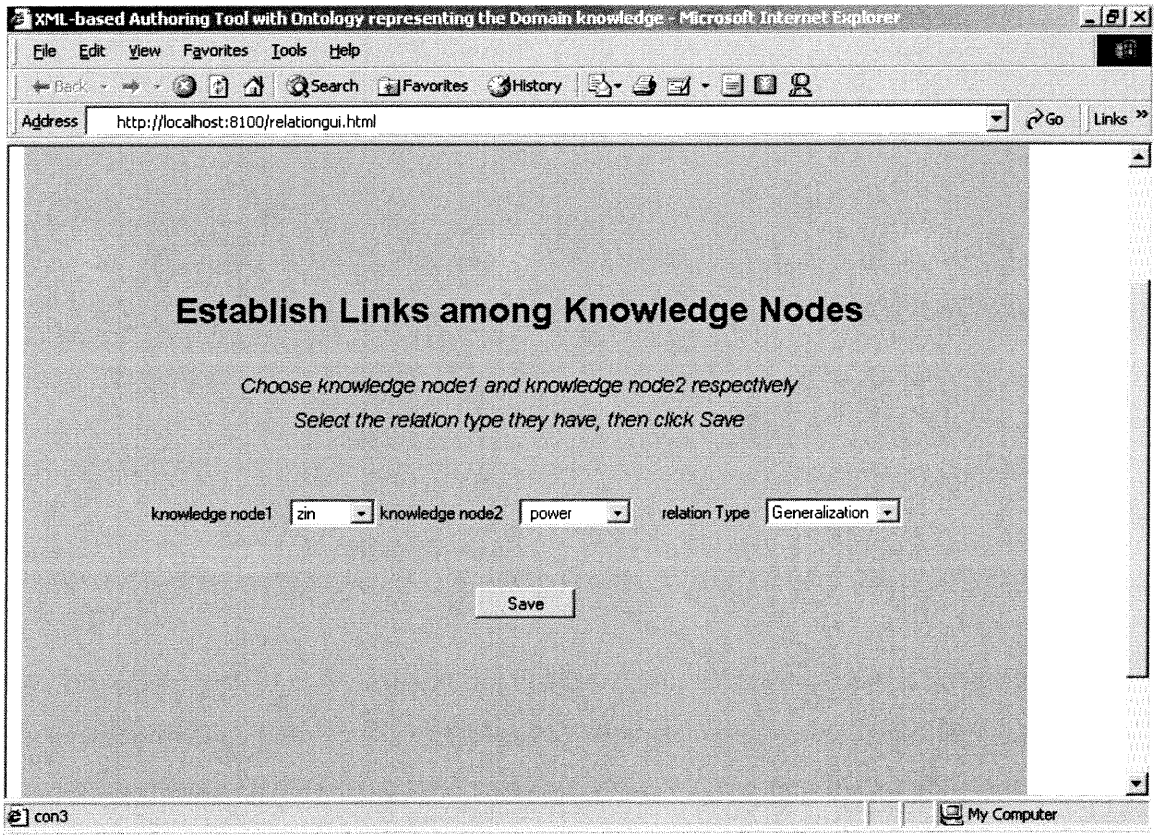

Figure 6.6 Establish links among knowledge nodes

Step 4: Browse and edit the established domain knowledge

So far, we have established six linked knowledge nodes. Here, we can press option 4 to view the whole picture. After press option 4, the page showed in Figure 6.7 pops up. We can find two parts on the page. Part one is architecture of nodes' relation. We use five different colors to represent different relation types. If checking the content of each node, we can go to part two. In part two, on the uppermost choice box, just click the choice box to choose the knowledge node ID we want to browse. Then, as shown in Figure 6.7, the content and link with other knowledge nodes are shown on the page according to the ontology we defined before. This time, we can perform the operations like delete node and change content and link. After change or delete, press "save change" to save the latest version. When we press refresh button, the picture in part one will be refreshed according the latest information.

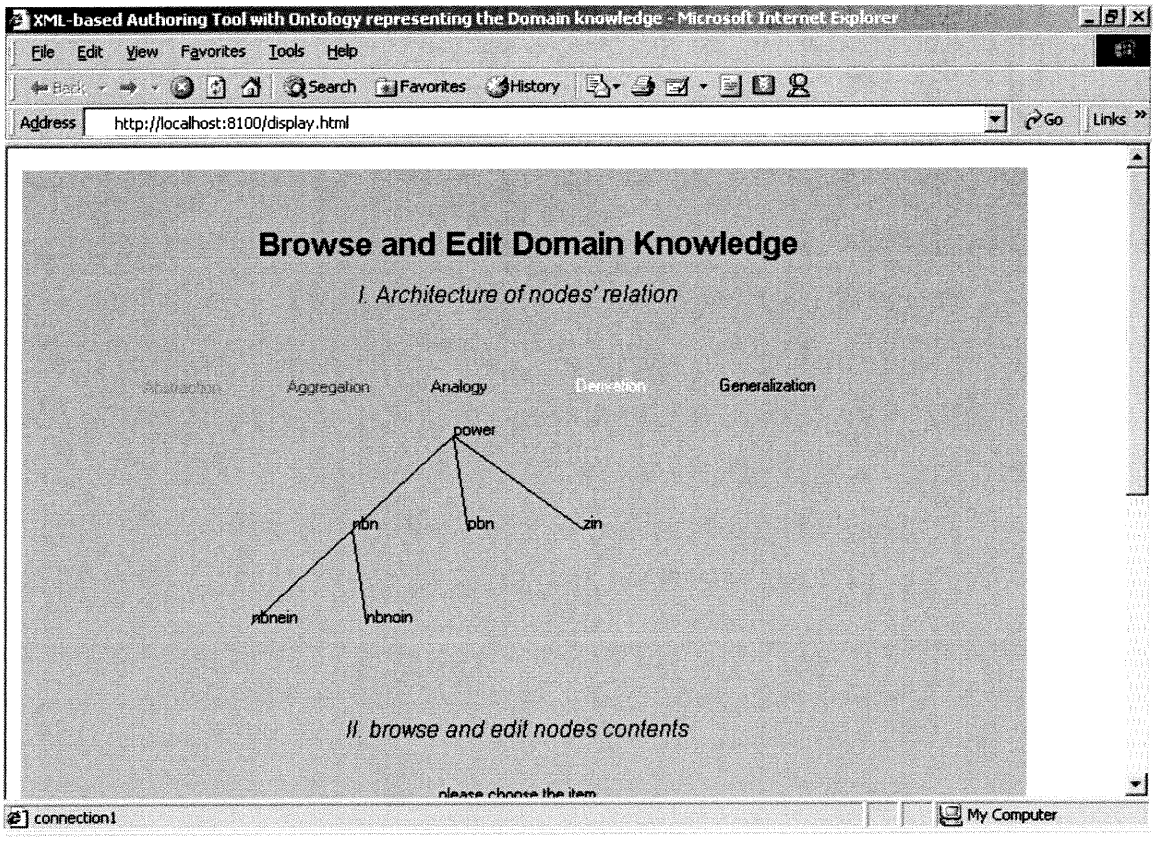

Figure 6.7 Browse and edit domain knowledge (1/2)

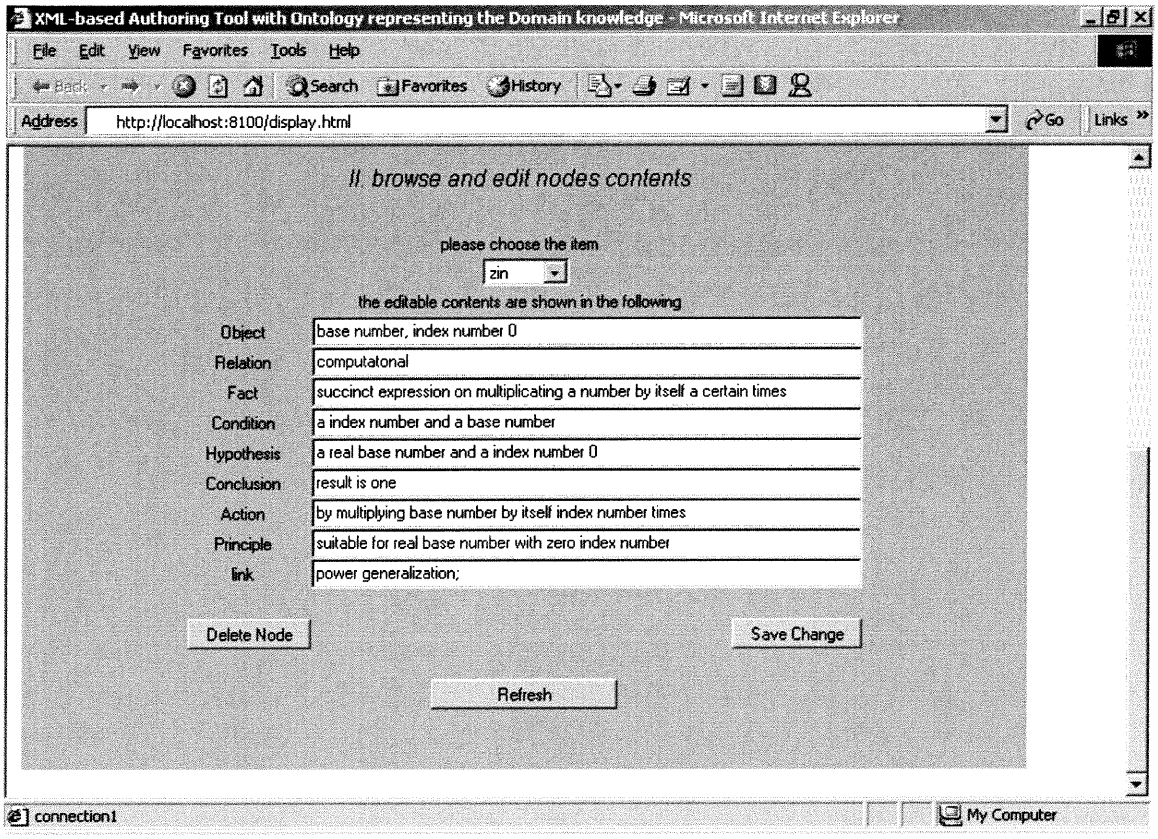

Figure 6.7 Browse and edit domain knowledge (2/2)

Step 5: Convert to XML file

Since the XML is becoming the file standard format in the Internet, the file must be converted to XML format if we want the file to be shared with other people. The content of knowledge nodes we typed are stored in database. When we choose option 5, the domain knowledge we developed before is converted to the XML file shown in Figure 6.8.

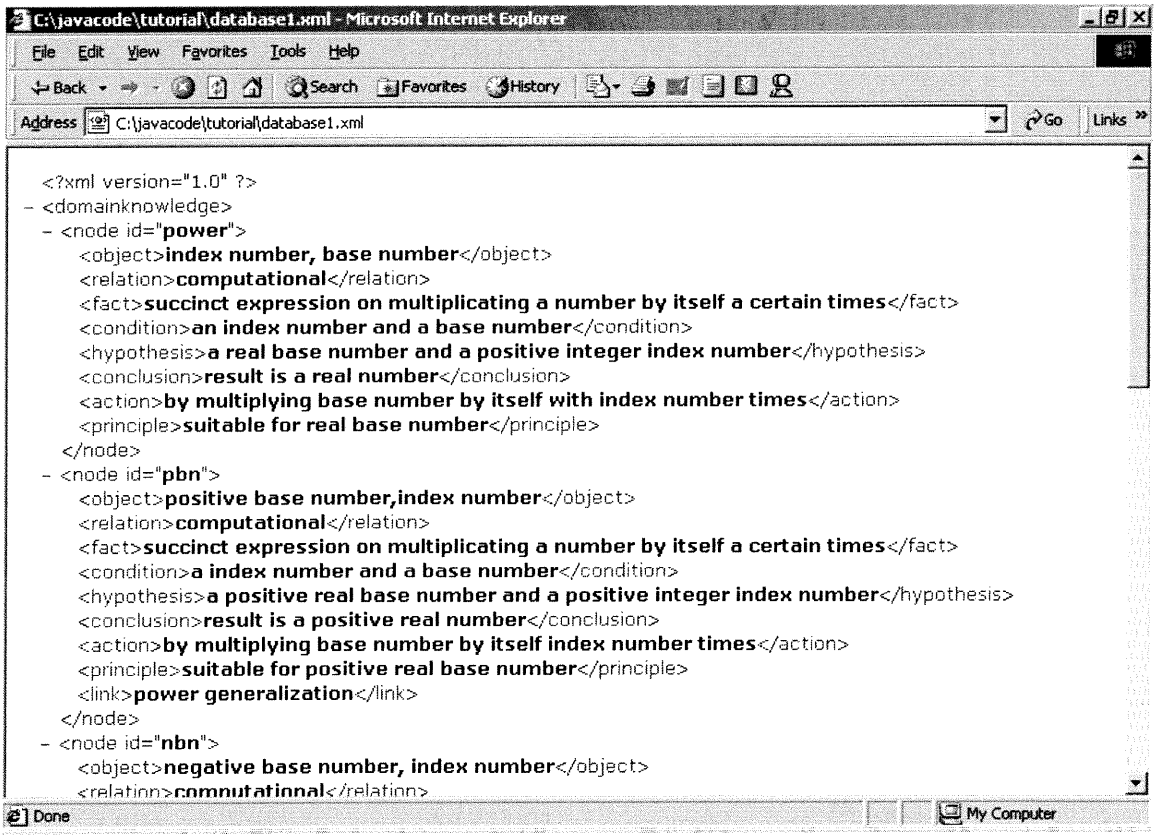

Figure 6.8 Domain knowledge XML file (1/2)

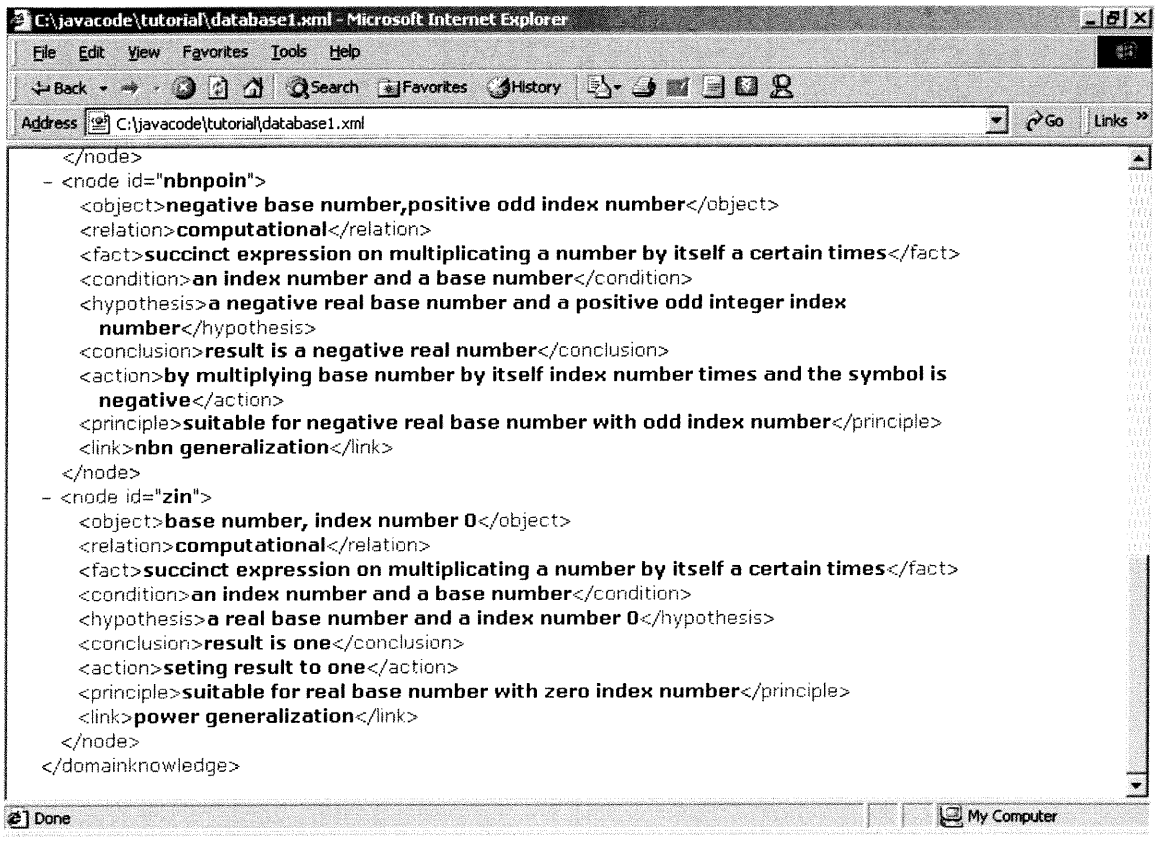

Figure 6.8 Domain knowledge XML file (2/2)

Chapter 7

Conclusion

This project was proposed under the circumstance that our society is stepping into the information society quickly. In the information society, everything changes so fast, especially in the knowledge area. A huge amount of knowledge becomes available through Internet and E-learning is gradually popular since it is the key to deal with continuous change of learners' needs and knowledge.

Contributions

The contributions of the authoring tool include:

1. Domain experts can use Internet to establish his/her own domain knowledge by either creating knowledge nodes himself/herself or querying the knowledge nodes developed by others, and establishing links among them.

On the basis of CREAM-C and under LOM standard and IMS specification, we developed the authoring tool allowing domain experts, in the Internet, to specify domain knowledge involved in the curriculum. The tool provides facilities to let the user create the knowledge nodes himself or herself and query the knowledge nodes developed by others. All these knowledge nodes can be either in local machine or in the remote computers. Users can also use the tool to establish the knowledge links between knowledge nodes in local computer and those of remote computers. The knowledge links are divided into five types, such as abstraction, aggregation, analogy, derivation and generalization.

2. Ontology is used to represent knowledge, which makes representation standardized.

The authoring tool uses ontology to provide the mean of knowledge representation. Ontology brings us standardized terms that are suitable for many domains and makes it possible to share domain knowledge developed by different people. It uses visual ways to satisfy the domain knowledge design in curriculum and assists curriculum authors and assist tutors to construct curriculum and plan courses. With the authoring tool using ontology, the progress to set up curriculum can be accelerated and avoid doing many jobs which must be done with standalone tutoring system.

3. The established domain knowledge is encoded into XML file that can be transferred in the Internet and referenced by other ITS modules easily.

With the help of the tool, the established domain knowledge is encoded into XML file that is convenient for being transferred and shared in Internet and being referenced by other ITS modules. The XML is the file format that is programming language-independent and platform-independent. That means XML file can be parsed no matter what programming languages and platforms are used. When ITS modules need to use the domain knowledge encoded into a XML file, without worrying about by what programming language the modules are developed and what platform they are in, the works are just parsing it and extracting the information. This makes it easy to use shared domain knowledge.

Limitations

The authoring tool we developed is a part of the whole intelligent tutoring system. Owing to the tight time schedule, there are some limitations such as,

1) Lack of some other support modules in ITS, i.e. student module and interaction development environments, to verify the interaction features with other modules.

2) Lack of more examples from different domains to verify its ontology versatility.

3) From practical view, some interfaces are not very friendly.

Future works

The future works to refine the system include:

1) Since the interfaces of the authoring tool are separated and not very friendly, the future work should provide a unified interface for each option and make the interface more convivial possibly by using of icons.

2) Since the option titles are very abbreviated, users maybe not very clear on what woks they do. Some help explanations on each option should be added to illustrate each usage.

3) For the users first using the tool to create the knowledge node, he (she) may be not familiar with what those ontology represent. So some introducing examples should be added to assist new users to be accustomed to this tool quickly.

4) Testing the system combined with other modules in a real course with real students to find errors and correct them.

5) For the ontology development, some tools can be used, such as OIL and DAML, to provide a rich set of ontology constructor.

References:

[Anderson 91] Anderson, J. R. and Pelletier, R., A *development system for model-tracing tutors.* In Proc. of the International Conference on the Learning Sciences, Evanston, IL, pp. 1-8, 1991.

[Barr 76] Barr, A and Atkinson,R.C, *The computer as a tutorial laboratory; the Standard BIP project.* International Journal of Man-Machine Studies, vol.8,567-596.

[Bretch 90] Bretch, B.J., *Determining the focus of instruction: content planning for intelligent tutoring systems.* Ph.D thesis, University of Saskatchewan, 1990.

[Brown 80] Brown, J.S., Vanlehn, K, *Rapir theory: A generative theory of bugs in procedural skills,* Cognitive Science, 4, 379-426.

[Brusilovsky 98] Brusilovsky, P., *Methods and Techniques of Adaptive Hypermedia.* In P. Brusilovsky, A. Kobsa, and J. Vassileva, editors, Adaptive Hypertext and Hypermedia, Chapter 1, pp. 1-44, Kluwer Academic Publishers, The Netherlands, 1998.

[DC 97] Dublin core metadata; http://purl.org/metadata/dublin_core, 1997.

[Deutsch 98] Deutsch, Alin and Fernandez, Mary, *XML-QL: A Query Language for XML* World Wide Web Consortium'1998.

[EWN] http://www.let.uva.nl/~ewn/

[Gagné 85] Gagné, R.M. *The condition of learning and the theory of instruction,* CBS college publishing, 4th edition, 1985.

[Gagné 93] Gagné, R.M., *Computer-based instructional guidance.* Automating instruction design: concept and issues, Educational Technology Publish, Englewood Cliffs, NJ, 133-146,1993.

[Gomez 94] Gomez-Perez, A., *Some ideas and examples to evaluate ontologies.* Technical report KSL-94-65, Knowledge systems laboratory, Stanford, 1994.

[Gruber 92] Gruber, T., *A translation approach to portable ontology specifications*, Proc. of JKAW'92, pp. 89-108, 1992.

[Gruber 93] Gruber, T., *Towards principles for the design of ontologies used for knowledge sharing.* In Roberto Poli Nicola Guarino, editor, International workshop on formal ontolofy, padova, Italy, 1993.

[Gruber 94] Gruber, T., *An ontology for engineering mathematics*, Proc. of Comparison of implemented ontology, ECAI'94 Workshop, W13, pp.93-104, 1994.

[Gruninger 94] Gruninger, R. and Fox, M., *The design and evaluation of ontologies for enterprise engineering*, Proc. of Comparison of implemented ontology, ECAI'94 Workshop, W13, pp.105-128, 1994.

[Guarino 97] Guarino, N., *Some organizing principles for a unified top-level ontology*, Working Notes of AAAI Spring Symposium on Ontological Engineering, Stanford, 1997.
[GUM]http://www.darmstadt.gmd.de/publish/komet/gen-um/newUM.html

[Hori 94] Hori, M. and Nakamura, Y., *Reformulation of problem-solving knowledge via a task-general level*, Proc. of the Third Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop: JKAW'94, pp.1-15, 1994.

[IDEF5] http://www.kbsi.com/idef/idef5.html

[IMS] http://www.imsproject.org

[John 94] John, B.E, Kierras, D.E, *The GOMS family of analysis techniques: tools for design and evaluation*. Carnegie Mellon University, 1994.

[Jonassen 96] Jonassen, D.H. & Reeves, T.C., *Learning with Technology: Using Computers as Cognitive Tools.* In D.H. Jonassen, (Ed.) Handbook of Research on Educational Communications and Technology. New York: Scholastic Press, Chapter 25.

[Lenat 90] Lenat, D. and R. Guha, *Building Large Knowledge-Based Systems*, Addison-Wesley Publishing Company, Inc., 1990.

[Lesgold 92] Lesgold, A and Lajoie, S, *A coach practice environment for an electronics troubleshooting job.* Computer Assisted Instruction and Intelligent Tutoring System: shared goals and complementary approaches. 201-238,1992.

[LOM] http://ltsc.ieee.org

[McCalla 90] McCalla, G. and McCalla, I, *The search for adaptability, flexibility, and individualization: approaches to curriculum in intelligent tutoring systems.* In Adaptive Learning environments: Foundations and Frontiers, Berlin, Springer-Verlag 91-112,1990.

[Mizoguchi 00] Mizoguchi, R., *IT Revolution in Learning Technology,* Proc. of the SchoolNet2000, pp.46-55, Pusan, Korea, August 4-5, 2000.

[Mizoguchi 2000] Mizoguchi, R., et al. *Using Ontological Engineering to Overcome AI-ED Problems,* International Journal of Artificial Intelligence in Education, Vol.11, No.2, pp.107-121, 2000.

[Mizoguchi 96] Mizoguchi, R., et al. *Knowledge engineering of educational systems for authoring system design—A preliminary results of task ontology design.* In: Prof. EuroAIED96, 329-335, 1996.

[Mizoguchi 95a] Mizoguchi, R. and Vanwelkenhuysen, Johan, *Task ontology for reuse of problem solving knowledge*, Proc. of KB&KS'95, pp.46-59, 1995.

[Mizoguchi 95b] Mizoguchi, R, Ikeda, M., Seta, K., et al., *Ontology for modeling the world from problem solving perspectives*, Proc. of IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995.

[Mizoguchi 93] Mizoguchi, R., *Knowledge acquisition and ontology*, Proc. of the KB&KS'93, Tokyo, pp. 121-128, 1993.

[Mizoguchi 92] Mizoguchi, R. et al., *Task ontology and its use in a task analysis interview systems -- Two-level mediating representation in MULTIS* --, Proc. of the JKAW'92, pp.185-198, 1992.

[Mars 94] Mars, N., *An ontology of measurement, Proc. of Comparison of implemented ontology*, ECAI'94 Workshop, W13, pp.153-162, 1994.

[MCF 97] *Meta Content Framework using XML*; http://www.w3.org/TR/NOTE-MCF-XML-970624, 1997.

[Miller 93] Miller, G., et al.: *Five papers on WordNet*, CSL Report 43, Cognitive Science Laboratory, Princeton University, 1993.

[Murray 91] Murray, T., Wqoolf, B.P., *A knowledge acquisition tool for intelligent computer tutor,* SIGART Bulletin, Vol.2, No2, 1-133, 1991.

[Murray 93] Murray, T., *Design and implementation of an intelligent multimedia tutor*, AAAI'93.

[Murray 97] Murray, T., *Expanding the knowledge acquisition bottleneck for intelligent tutoring systems*. International J. of Artificial Intelligence in Education. Vol. 8, No. 3-4, pp. 222-232. 1997.

[Murray 98] Murray, T., *Authoring knowledge base tutors: tools for content, instructional strategy, Student Model, and Interface Desig*n. J. of the Learning Sciences, 7, 1, 5-64, 1998.

[Murray 99] Murray, T., *Authoring intelligent tutoring systems: an analysis of the state of the art*, International J. of Artificial Intelligence in Education Vol. 10, pp 98-129, 1999.

[Nkambou 98] Nkambou, R., Frasson, C., Gauthier, G., *A new approach to ITS-curriculum and course authoring: the authoring environment*, Computers &Education, 31,105-130, 1998.

[RDF97] Introduction to RDF Metadata; http://www.w3.org /TR/NOTE-rdf-simple-intro-971113.html, 1997.

[Polson 88] Polson, M., Richardson, J., *Foundation of intelligent tutoring systems*, Lawrence Erlbaum associates publishers, 1988.

[Sasajima 95] Sasajima, M., Kitamura, Y., Ikeda, M., and Mizoguchi, R., *FBRL: A Function and behavior representation language*, Proc. of IJCAI-95, pp. 1830-1836, Montreal, 1995.

[Seta 97] Seta, K et al.: Capturing a Conceptual Model for End-user Programming -Task Ontology as a Static User Model-, Proc of UM'97; http://www.ei.sanken.osaka-u.ac.jp/pub/seta/seta-um97.html.

[Schreiber 93] Schreiber, G., Windinga, B.&Breuker, J, *Kad, a principle approach to knowledge-based system development,* Academic press, 1993.

[Sowa 95] Sowa, J, *Distinction, combination, and constraints*, Proc. of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.

[Sowa 98] Sowa, J, *Knowledge representation: logical, philosophical and computational foundation,* PWS Publishing Comp, Boston 1998.

[Takeda 95] H.Takeda, et al.: Agent organization and communication with multiple ontologies, International Journal of Cooperative Information Systems, Vol.4, No.4, pp.321-337, 1995.

[TOVE] http://www.ie.utoronto.ca/EIL/tove/toveont.html

[Uschold 94] Uschold, M and Gruninger, M, *ontologies: principles, methods and applications*. Knowledge engineering review, 11(2) June 1996.

[Vanwelkenhuysen 94] Vanwelkenhuysen, J. and Mizoguchi, R., *Maintaining the workplace context in a knowledge level analysis*, Proc. of JKAW'94, Hatoyama, Japan, pp.33-47, 1994.

[Vanwelkenhuysen 95a] Vanwelkenhuysen, J. and R. Mizoguchi, *Ontologies and guidelines for modeling digital systems*, Proc. of 9th KAW, Bannf, Canada, 1995.

[Vanwelkenhuysen 95b] Vanwelkenhuysen, J. and Mizoguchi, R., *Workplace-Adapted Behaviors: Lessons Learned for Knowledge Reuse*, Proc. of KB&KS '95, pp.270-280, 1995.

[Webster 94] Webster, J.G., *Instructional objectives and bench examinations in circuits laboratories.* IEEE Transition on Education, Vol. 37, No.1, 111-113, 1994.

[Wielinga 93] Wielinga, B. et al., *Reusable and sharable knowledge bases: A European perspective*, Proc. of KB&KS'93, Tokyo, pp.103-115, 1993.

[WordNet] http://www.cogsci.princeton.edu/~wn/

[Yamaguchi 97] Ymaguchi, T et al.: A legal ontology rapid development environment using a machine-readable dictionary, LEGONT'97, pp.69-73, 1997.