Knowledge Management Support for Web Based Learning

Jia Zhao

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

January 2002

0-612-68493-8

ABSTRACT

Knowledge Management Support for Web Based Learning [1]

Jia Zhao

The overlapping area of *Knowledge Management* (KM) and *Web Based Learning* (WBL) deserves more attention from researchers. The WWW is becoming more and more popular for disseminating distance learning programs; and KM technologies have been largely utilized in industrial and business corporations in contrast to their application in the educational sector. Current WBL environments, whose major role should be delivering and presenting knowledge, still lack systematic KM support. Motivated by intention to apply KM strategies to the area of WBL, this thesis aims: (1) to perform a systematic analysis and categorization of KM strategies that can be utilized in the field of WBL; (2) to propose a system model for developing a WBL environment with KM support; (3) to explore in depth two KM strategies: namely *Subject Knowledge Modeling* and *Dynamic Course Planning-cum-scheduling*. As the practical contribution of this thesis, two prototype systems are developed and tested. One is called *Online Course Presentation System* and the other is called *Online Course Planning System*. Both prototypes have been tested by a group of 10 students in a practical environment.

1. Computer Assisted Learning could also be in the context of a Web. Thus Web Based Learning (WBL) is the term used synonymously with Computer Assisted Learning in this thesis.

Acknowledgement

I would like to express my sincere gratitude to my two supervisors - Dr. Franz Kurfess and Dr. Thiruvengadam Radhakrishnan, for accepting me as their student and introducing me into this field. Their valuable insights and persistent guidance throughout my research have made the success of this thesis possible.

I am thankful to Dr. Thiruvengadam Radhakrishnan for his financial support.

I also acknowledge Mr. Chunbo Yang's hearty cooperation in implementing the system prototypes. His kindness is invaluable.

Specially, I would like to thank my parents, for their encouragement and deepest dearest love.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Defining Knowledge Management

Our culture now accepts such notions as "knowledge workers", "knowledge media", and "knowledge economics", etc. These terms suggest a shift towards a knowledge-focused society. The title of this thesis is *Knowledge Management (KM) Support for Web Based Learning (WBL)*. Such a thesis cannot proceed without explaining what is *knowledge* and *Knowledge Management.*

Different practitioners have their own beliefs of what constitutes "knowledge". Karl Wigg (The Knowledge Management Forum 1996) and his colleagues presented a "working definition" of the term "knowledge": "Knowledge consists of facts, truths, and beliefs, perspectives and concepts, judgments and expectations, methodologies and know-how". Further, he says, "Knowledge is accumulated and integrated and held over a period of time to handle specific situations and challenges." Maarten Sierhuis's (The Knowledge Management Forum 1996) view is that knowledge is always "situated." So, rather than follow the cognitive scientific view of knowledge as symbolic representations stored in the brain, he views that knowledge is created in a situation. "Knowledge is fluid, tacit, and forever changing", he remarks, and "is never again used in exactly the same way." Eunika Mercier-Laurent (The Knowledge Management Forum 1996a) considers knowledge from a "representation" perspective: "Knowledge is information in a suitably represented form (conceptual model, objects, frames, constraints, cases, rules, graphs, etc), and different kinds of reasoning (decision making, learning, etc) can make use of it."

There is also a large number of definitions for the term "Knowledge Management (KM)", proposed by scholars and researchers from various backgrounds. Thomas Bertels (The Knowledge Management Forum 1996) provides a lengthy definition of KM: "Knowledge management is the management of the organization towards the continuous renewal of the organizational knowledge base - this means, for example, creation of supportive organizational structures, facilitation of organizational members, putting IT-instruments with emphasis on teamwork and diffusion of knowledge (as e.g. groupware) into place." Denham Grey (The Knowledge Management Forum 1996) offers another lengthy interpretation: "Knowledge management is an audit of "intellectual assets" that highlights unique sources, critical functions and potential bottlenecks which hinder knowledge flows to the point of use. It protects intellectual assets from decay, seeks opportunities to enhance decisions, services and products through adding intelligence, increasing value and providing flexibility." Comparatively, Daniel E. O'Leary (O'Leary 1998) provides a simple interpretation of the term, but it is too broad: "Knowledge management is the formal management of knowledge for facilitating creation, access and reuse of knowledge, typically using advanced technology."

In my opinion, it is hard to define KM precisely and in a generic fashion, because in practice, knowledge and KM are always associated with a domain of application. The definition of KM should be based on what kind of knowledge that domain contains, what kind of processes exists in that domain, and what kind of supports that domain asks for. Thus, it is reasonable that we first choose and analyze the domain to which KM will be applied. Then we know the content of KM within that context. On the other hand, regardless of what domain KM will be applied, there is a consensus within the academia of computer science that KM incorporates a collection of processes and strategies that govern the creation, dissemination, and utilization of knowledge. Researchers keep encountering similar situations in different domains. So far, generic KM research work has been devoted to both universal applications and specific applications.

1.2 Knowledge Management and Web Based Learning

Currently, domain-specific KM research mainly focuses on the areas of industry and business. Industrial factories and business corporations are where most KM systems were implemented and most KM strategies have been utilized. This is because knowledge has become the most important resource in those organizations. The success of an organization depends on its ability to transform the personal knowledge of employees, as well as knowledge stored on passive information carriers, such as handbooks, into organizational knowledge and make it widely available according to special needs. Other domains, in which knowledge is no less central than in industry and business, deserve more KM support than what they receive now. Web Based Learning (WBL) is one of those domains.

The WWW has become a very popular means of disseminating distance-learning programs. The basic role of WBL is two-fold: first, to deliver Web-based materials; secondly, to provide interaction among instructors and students. However, due to all kinds of technological barriers, in order to attract novice users and make them stay with the system, the main point is not how much information a WBL tool carries, or how diverse a WBL tool's functions are, but how effectively it can help the end users in their learning. Today's "off the-shelf " WBL environment has integrated many tools and supports every activity in traditional learning environments. But the problem is that they concentrated too much on simulating the multifunctional aspect of the traditional learning environment, and did not explore enough about how to improve learning effectiveness through this new method - KM.

One promising way to improve learning effectiveness with WBL is to introduce KM strategies into this field. WBL is a domain which contains a large amount of knowledge about subjects, courses, and students, and in which almost every process involves a certain portion of knowledge. For the course author, he is a knowledge creator and organizer, who is looking forward to a efficient approach to structure knowledge for different subjects. For the instructor, he is a knowledge presenter, who is always seeking a better way to arrange course material and activities so that the knowledge will be presented thoroughly. For the student, he is a knowledge receiver, who should not only have access to the most up-to-date resources, but also be guided to accomplish various learning goals effectively. I believe that each of these processes can be a potential source for KM support. For example, if the system possesses the knowledge about its users, it can provide personalized tips more specific to that situation. As another case, if the system possesses the knowledge of scheduling rules instead of simply a fixed schedule, then it can schedule and reschedule to adapt to the current situation.

1.3 Objectives of the Thesis

There exists a gap between KM and WBL. I have the intention of bringing them together, and devote this thesis to the exploitation of how KM will best support WBL. To start with, there is a hypothesis: If good knowledge management is applied, students can benefit a lot in Web Based Learning and the converse is also true, that is bad management can lead to no extra benefit arising from Web Based Learning.

The objectives are:

1. Since both the topic areas KM and WBL are very broad, the first thing is to narrow the scope of the problem. To achieve this, two questions are raised in surveying the two topic areas.
   Which part of KM is most suitable for WBL?
   Where does WBL especially need KM support?

2. Deeply examine the knowledge content and knowledge flow within a general WBL environment. Propose a KM system model for the specific application of WBL.

3. If the above two objectives are to be viewed at a macro level, then I would like to switch to micro level by proposing and focusing on one or two specific KM strategies dedicated to WBL. A real world case of learning the Assembly Language in COMP228 or COMP520 at Concordia University will be used to study those KM strategies.

4. To prove the effectiveness of the proposed KM strategies, I will develop a prototype of the intended system and evaluate it with real users.

1.4 Thesis Organization

The thesis begins by examining the theoretical foundations and related research work in Chapter 2. In this chapter, I survey various systematic views of KM processes and strategies and give my own summary. Then I move on to the WBL area and summarize different types of learning tools that can be integrated into the WBL environments. Corporate Memories, Knowledge Description Ontologies, and Knowlets are presented.

In Chapter 3, I analyze the problem domain by asking the following questions: What classes of knowledge exist in the field of WBL? What principles contribute to learning effectiveness? What value does KM intend to bring to WBL? What KM strategies can be applied to WBL? I identify two kinds of KM support for WBL: Content-Centered Knowledge Management (CCKM) and Process-Centered Knowledge Management (PCKM). CCKM concentrates on the structure and organization of domain knowledge concept and knowledge media, their creation, storage, and reuse in computer-based environment; PCKM focuses on the view of KM as a social communication process which can be improved by various aspects of groupware support. Finally I propose a system model for WBL environment which serves as a framework to incorporate the above two kinds of KM support.

Chapter 4 and Chapter 5 address two specific KM strategies. Chapter 4 addresses *Subject Knowledge Modeling*, a KM strategy which belongs to CCKM. This strategy considers organizing and structuring the subject knowledge of a course. This has three aspects. First of all, it is about finding a structure which contains the hierarchical classification of the subject knowledge as well as the dependencies between 'represented knowledge-units'; secondly, it is about how to relate actual media contents to the structure, since knowledge is not just an empty structure; finally, how to implement such a structure in the computer system so that it can be easily presented and maintained.

Chapter 5 presents *Dynamic Course Scheduling-cum-Planning*, one of the PCKM strategies. It deals with dynamic priority assignment by a student, since different activities have different deadlines and demands. It deals with student's characteristics, his/her demands on time, and preferences. Different students manage their time differently. Finally, there is a relationship between CCKM and PCKM. Since most activities deal with certain amount of content-based knowledge, the sequencing of these activities is restricted by knowledge dependencies. This KM strategy was tested using a real-world case -- planning and scheduling for an Assembly Language course. The general planning and scheduling principles are analyzed and adapted to the WBL environment. Certain constraints are taken into consideration. A heuristic algorithm is introduced to dispatch activities and generate final outputs: dynamic plans and schedules.

The thesis work also includes the working prototype of two systems: Online Course Presentation System and Online Course Planning System. The Online Course Presentation System takes advantage of the Subject Knowledge Modeling strategy introduced in Chapter 4, and utilizes the knowledge tree to encapsulate and present subject knowledge for Assembly Language class. The Online Course Planning System deploys the Dynamic Planning and Scheduling strategies discovered in Chapter 5, using a timeline to encapsulate learning activity-related knowledge. The objective of the planning system is to provide dynamic scheduling and prevent the student from violating his/her plans. Several use cases are devised and demonstrated in the demo to drive home the hypothesis stated in Section 1.1. All the implementation and evaluation issues are addressed in Chapter 6.

Chapter 7 concludes the thesis and addresses the related future work to do.

2. Background

2.1 Theory and Conceptual Foundations

2.1.1 Overview of KM Processes and Strategies

"Knowledge Management: Great concept but what is it (Angus 1998) ?" To establish a theory and conceptual foundations towards an understanding of KM, a systematic overview is more useful than abstract definitions. Many researchers have proposed diverse systematic views of KM processes and strategies. The following sections introduce some popular ones.

2.1.1.1 Four-process View

Jeff Angus and Jeetu Patel (Angus 1998) describe a *four-process view* of KM. The four processes are *gathering*, *organizing*, *refining* and *disseminating*. This view actually simulates KM engineering, since each process represents a stage in the knowledge life cycle.

2.1.1.2 Two-track View

Karl E. Sveiby (The Knowledge Management Forum 1996) contrasts two tracks of thought on the subject at two different levels.

Track 1: KM = Management of Information

Several researchers of KM have a background in AI, reengineering, groupware etc. To them "knowledge = objects" that can be identified and handled in information systems. They focus on the research and practices within an organization.

Track 2: KM = Management of People

Researchers and practitioners in this track have their education in philosophy, psychology, sociology or business/management. They are primarily involved in assessing, changing and improving human individual skills and/or behavior. To them "knowledge = processes", a complex set of dynamic skills, know-how etc, that is constantly changing. The focus of research and practice are on an individual.

2.1.1.3 Converting and Connecting View

By defining: "KM is a process of converting knowledge from the sources accessible to an organization and connecting people with that knowledge", O'Leary (O'Leary 1998) classifies KM processes into six categories:

1. Converting individual to group-available knowledge;
2. Converting data to knowledge;
3. Converting text to knowledge;
4. Connecting people to knowledge;
5. Connecting people to people;
6. Connecting knowledge to people.

A contribution of O'Leary's view of KM is that he introduces connecting knowledge to people, thus breaking the assumption of classic KM systems that knowledge has to be pulled from the system, and suggests that there exist alternative strategies that push the knowledge to the user.

2.1.1.4 Three-Layered Model

A *three-layered model* for processing knowledge, as illustrated in Figure 2.1 is introduced by Andreas Abecker (Abecker 1998). *Object level* comprises manifold information and knowledge sources; *Knowledge-description level* enables a uniform, intelligent access to a diversity of object-level sources; *Application level* links the information model and the concrete application situation. This three-layered model of KM is derived from a general KM system architecture.

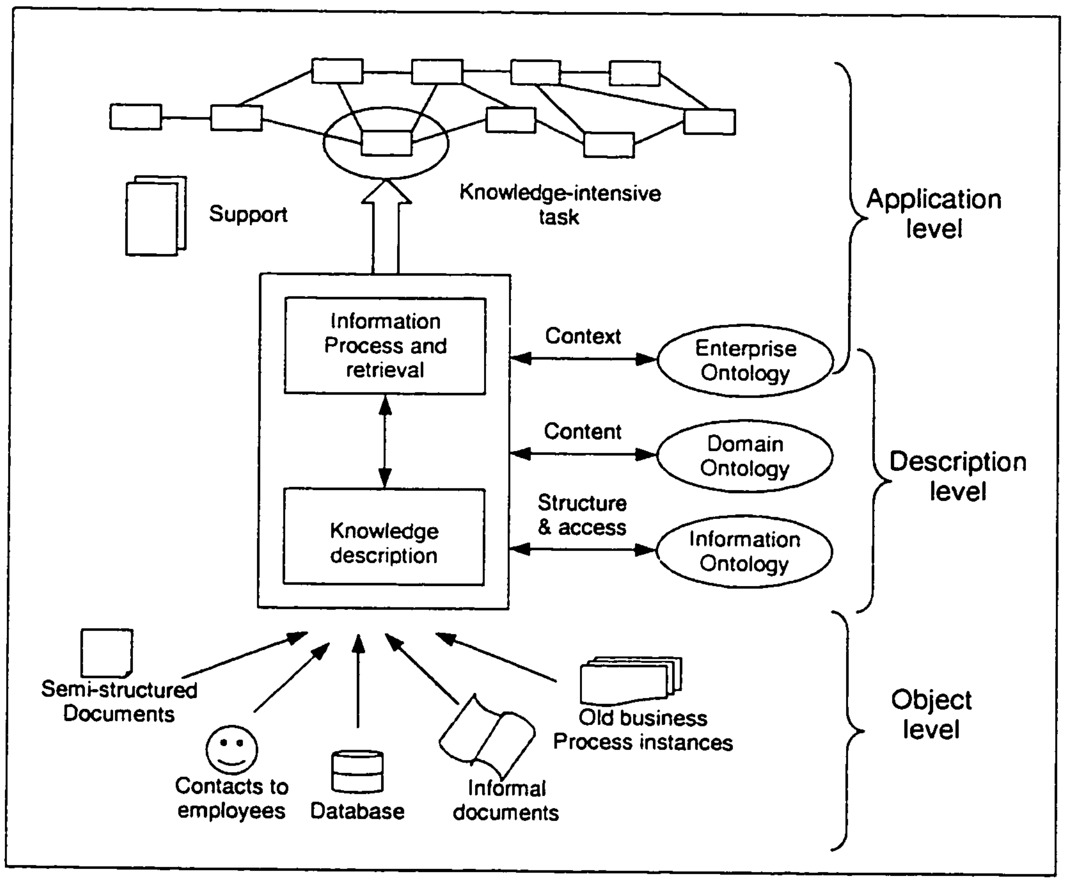


Figure 2.1: Three-layered Model of KM

2.1.1.5 Summary

In order to summarize various KM strategies in an integrated manner, I chose to combine the three-layered model (Abecker 1998) with six KM processes, as illustrated in Table 2.1. It should be pointed out that KM strategies are not limited to those presented in Table 2.1. A domain specific KM system adopts KM strategies which meet the requirements of its domain, and if necessary, it can modify some of the KM strategies and even introduce new ones.

Layer	KM Process	KM Strategy
Object Level	Acquisition	• Data entry • OCR and scanning • Voice input • Document analysis • Information filtering
	Preservation	• Data warehousing • Database • Knowledge base
Description Level	Representation	• Ontologies • Knowledge Component • Information modeling • Conceptual graph • Description logic
	Organization	• Cataloging • Indexing • Linking • Searching • Mapping
Application Level	Dissemination	• Flow • Sharing • Alert • Push
	Utilization	• Workflow management • Personal assistant • Cooperative information system • Electronic performance support system • Case-based reasoning • Intelligent information retrieval • Data mining

Table 2.1: Summary of KM Processes and Strategies

- **Corporate or Organizational Memory**

A narrow definition of corporate memory is "an explicit, disembodied, persistent representation of the knowledge and information in an organization (Heijst 1998) supporting sharing and reuse of individual and corporate knowledge and lessons learned (Abecker 1998)." However, it is presumed that any information that, if well managed, will enhance the organization's competitiveness can be considered as part of the corporate memory. In practice, the corporate memory is a centralized, well-organized information depository located at the core of an organization. Intelligent knowledge management services are built around it to assist the user working on a knowledge-intensive operational task by providing all the information necessary and useful for fulfilling this task (see Figure 2.2) (Abecker 1998).

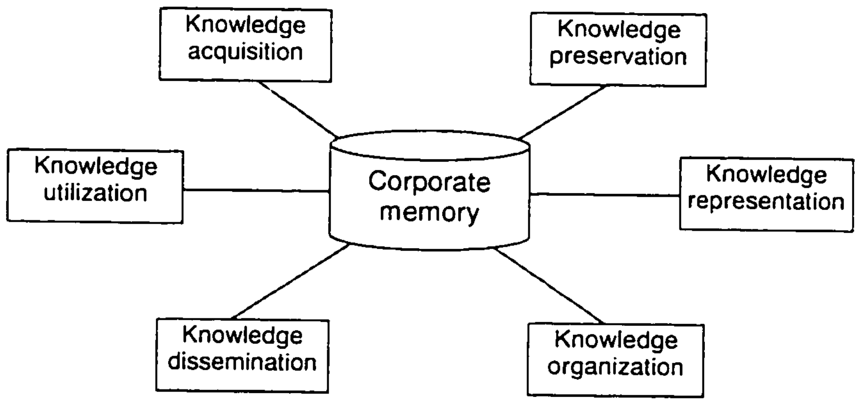


Figure 2.2: Corporate Memory Assists in the Basic KM Processes (Abecker 1998)

- **Ontology**

An *Ontology* is a very important concept at knowledge description level. In order to understand *ontology*, we should first understand *conceptualization*. We can see conceptualization as a world view with respect to a given domain, which is often conceived as a set of concepts (e.g. entities, attributes, processes), their definitions and their inter-relationships. Ontologies generally appear as a taxonomic tree of conceptualizations, from very general and domain-independent at the top levels to increasingly domain-specific further down in the hierarchy (Chandrasekaran 1999). A significant number of KM systems are ontology-based. The first step in developing such a KM system is to perform an effective ontological analysis of the domain. A KM system for WBL is not an exception. Current research on ontologies can be divided into three categories: effort to create large ontologies, to define expressive languages for representing ontological knowledge, and to implement systems that support ontology-based applications (Chandrasekaran 1999).

2.1.3 Overview of WBL

The objective of learning is not only to acquire *theoretical knowledge*, but also to reconstruct *tacit knowledge* by applying the theory to practice. This application of theory therefore constitutes part of the learning process. No matter what instructional model a WBL application adopts, (e.g., drill and practice, tutorials, simulations, problem solving), it should involve interactive practices besides the delivery of learning materials. A WBL application can be called *integrated WBL environment* when it simulates the following minimal set of activities of the real world learning environment:

- enrollment/registration;
- provision of course advice;
- provision of course materials;
- provision of feedback;
- provision of responses to administrative and academic queries;
- conduct of assessment;
- provision of assessment results.

Table 2.2 illustrates various tools that can be integrated into a learning environment. Although these tools are not delegated to WBL, they may incorporate web-specific features that other forms of computer-assisted learning cannot provide. My thesis will mainly concentrate on the impact of KM on two of those learning tools: *structured content* and *course plan and schedule.*

Category	**Tools or features**
Communication tools	• Chat • E-mail • Bulletin board • Mailing list • White board • Newsgroup • Multiple user domains (MUDs) • Web-based conferencing system • Web-based presentation
Course material tools	• **Structured content** • Syllabus • Glossary • Searching • Index • Print • Algorithm animation • Database search • References
Assessment tools	• Assignment • Self-test • Quiz/test • Exams
Student support tools	• Calendar • Learning tips • Learning goals • Orientation • Learning skill • **Course plan and schedule** • Personal counseling • Question/answer • Support for disabled students • Record and progress monitoring

Student/course management tools (for instructor)	• Student registration and profile • Progress tracking/reporting • Access control • Student management • Course planning/authoring/backup • File management

Table 2.2: Categories of WBL Tools

- **Structured content**

This tool is used to accomplish one major role of WBL -- delivery of course materials. The most popular form of structured content we see on the Web is the online book, a digitized version of a real book, using the same structure and text content. However, structured content should not be regarded as equal to online book. It does not have to be text based only either. A well-structured content should reflect the underlying knowledge organization of a domain, and the Web can deliver knowledge in different kinds of media. Moreover, the Web makes any modification to the domain knowledge accessible to users as soon as it has been made - a feature that other types of computer-assisted learning tools may not contain.

- **Course plan and schedule**

Course plan and schedule are not equal to a digitized course outline and static "to-do list". They should be regarded as dynamic outputs of two important WBL processes: planning and scheduling, which are able to incrementally adapt their behavior to a heterogeneous variety of users. Such processes demand underlying modeling of users, whether built manually by the users or built by the system through monitoring user behavior and acquiring unobtrusively other knowledge about users. Implemented as Web applications instead of stand-along applications, this tool may incorporate advanced features. For example, it may trace each student's progress through the Web, calculate the average class progress, and provide personal-specific suggestions and warnings based on the class average.

2.2 Existing WBL Environments

2.2.1 WebCT

WebCT (see WebCT homepage) has been developed by the Department of Computer Science at the University of British Columbia and was originally intended to support the department's own online programs. After it was licensed and commercialized, it developed into a robust product that facilitates the creation of sophisticated WBL environments. It offers a variety of features including a conferencing system, online chat, tracking of student progress, organization of group projects, student self-assessment, maintenance and publishing of grades, automatic index generation, course content searches, a course calendar and student home pages.

Although WebCT is a quite mature WBL environment, it still has some limitations. First of all, WebCT provides a simplified model for learning. It does not "remember" the student's characteristics and learning preferences, thus it cannot give any individualized feedback concerning a particular student's progress. It just records some static data, such as grades and the visiting times of a page, and then waits for the student to "pull" this information out. Another drawback of WebCT is in its way of organizing course content. It supports a limited "online text book" style, and does not leave much space for media content. Finally, WebCT does not provide any planning and scheduling tool for students.

2.2.2 LearningSpace

LearningSpace has been built on top of the groupware product Lotus Notes and the Domino Web Server software (see LearningSpace homepage). It takes advantage of the features that Lotus Notes offers for conferencing, e-mail and scheduling. However, LearningSpace is more than just a groupware product. It provides an online environment for unparalleled support of all three delivery modes of e-learning — self-directed, asynchronous collaboration, and "virtual classroom" learning. LearningSpace comprises five distinct databases:

1. Schedule -- directs students through a course;
2. Media Centre -- for distribution of courseware;
3. Course Room -- a discussion area;
4. Profiles -- provides information on students;
5. Assessment Manager -- supports administration of tests.

LearningSpace has a feature which WebCT does not have -- activity tracking. It tracks and reports on each activity within a course, including the number of times the student launched the activity, the amount of time spent, status, score, and the student's current location in the activity. A student can access this information. However, LearningSpace does not incorporate dynamic or incremental generation of a schedule of activities.

2.3 Related KM Research and Work

2.3.1 Corporate Memory Reference Architecture

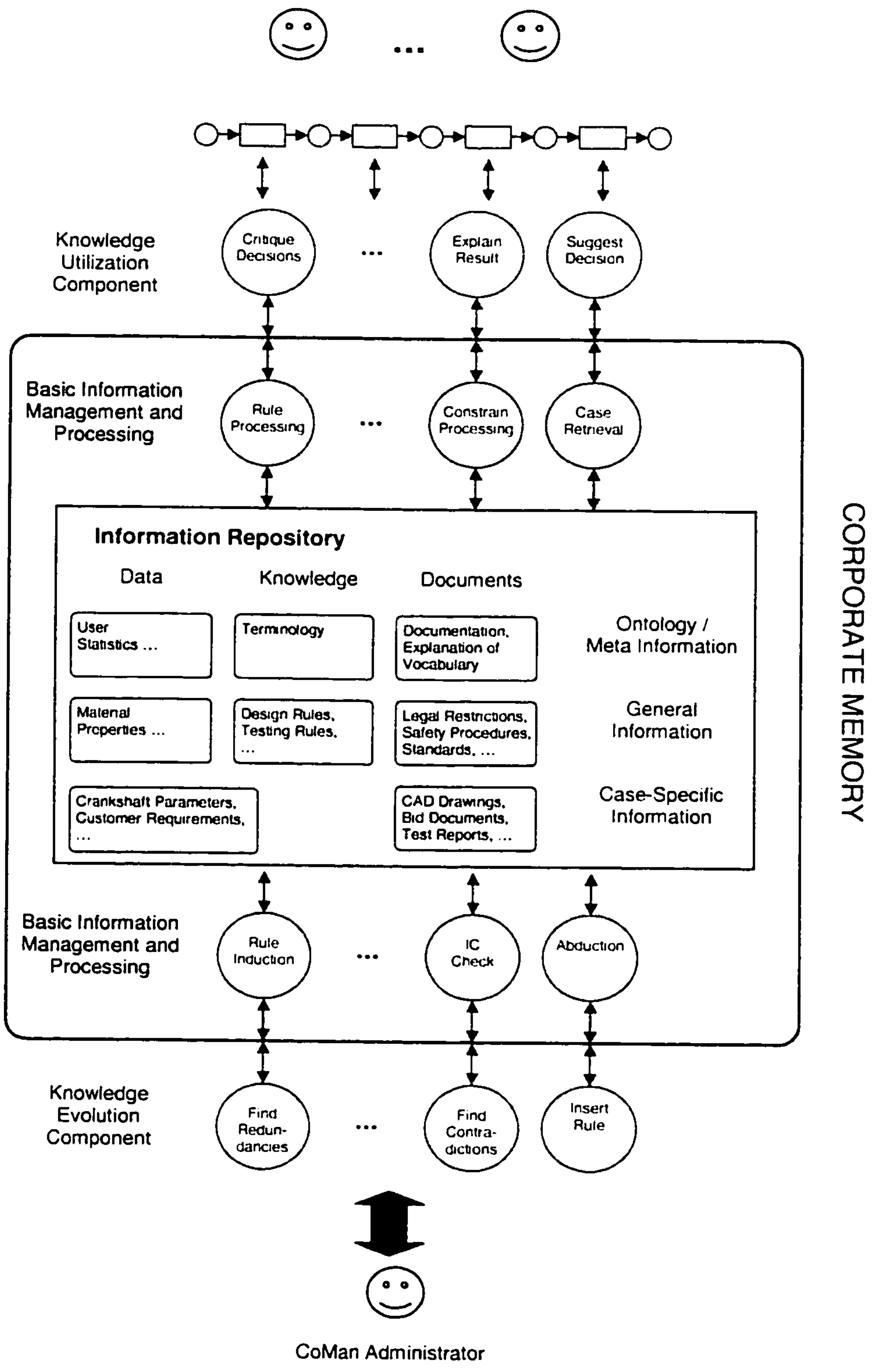
Kuhn and Abecker (Kuhn 1998) suggested a general architecture for a Corporate Memory (see Figure 2.3). *Information Repository* stands at the core of the system. The contained information is structured with regard to two dimensions. The first dimension is based on the representation format of information, distinguishing data, knowledge, and documents. In an organization, *data* and *documents* usually are already stored in databases, paper-based archives, or in hypertext forms. The *knowledge* – characterized by complex formal representations that can be processed with complex inferences – has to be captured perhaps by using AI techniques. The second dimension is based on the level of abstraction, its role in the problem solving process, and the temporal stability of information. It distinguishes three information layers:

1. Case-specific information that describes processes and control information used or generated during concrete executions of the operational task under consideration.

2. General information or abstracts from specific task instances describes how to use and interpret the concrete case data.

3. Ontological, or meta information layer manages the most abstract kind of information which represents the basic assumptions underlying the system development and representation (Kuhn 1998).

The central information repository is accessed by a number of *Basic Information Management and Processing Services*. Those services are utilized by *Knowledge Utilization and Evolution Components* employing the basic mechanisms for solving subtasks occurring in the work processes to be supported (Kuhn 1998).



Knowledge Utilization Component
Critique Decisions
...
Explain Result
Suggest Decision
Basic Information Management and Processing
Rule Processing
...
Constrain Processing
Case Retrieval
Information Repository
Data
Knowledge
Documents
User Statistics ...
Terminology
Documentation, Explanation of Vocabulary
Ontology / Meta Information
Material Properties ...
Design Rules, Testing Rules, ...
Legal Restrictions, Safety Procedures, Standards, ...
General Information
Crankshaft Parameters, Customer Requirements, ...
CAD Drawings, Bid Documents, Test Reports, ...
Case-Specific Information
Basic Information Management and Processing
Rule Induction
...
IC Check
Abduction
Knowledge Evolution Component
Find Redundancies
...
Find Contradictions
Insert Rule
CoMan Administrator
CORPORATE MEMORY

Figure 2.3: Corporate Memory Reference Architecture (Kuhn 1998)

2.3.2 Knowledge Description Ontologies

Abecker (Abecker 1998) provides a general information modelling scheme for organizations by applying three kinds of knowledge description ontologies. The *enterprise ontology* is used to describe contexts and the intended utilization of contexts. The *information ontology* contains generic concepts and attributes that apply to all kinds of information. The *domain ontology* is used for content description, providing a view of the internal structure of the domain knowledge. Figure 2.4 illustrates a simple information model which is based on above scheme.

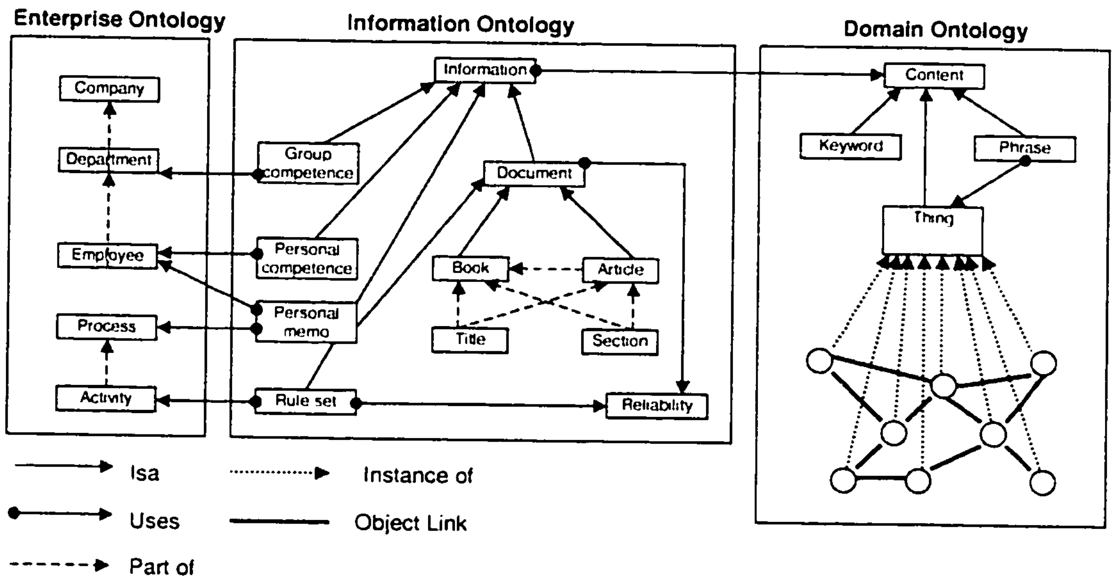


Figure 2.4: Simple Knowledge Description Ontologies

2.3.3 Component-Based Knowledge Management (CBKM)

Kurfess (Kurfess 1999) has been working on the component-based organization of knowledge. He defined a structure called *knowlet* to represent knowledge components. Knowlets have the following features:

- Structured hierarchically
- Hyperlinks across knowlets
- Different relationship based on domain classification, similarity and usage.
- Have granularity (amount of info)
- Can be accessed according to contents
- Mobile, change location
- Cloned easily

Kurfess also designed the architecture of knowlets to support the above features, which is presented in Table 2.3:

Parts	Contents	description
Actual Content	Other knowlets	Composite knowlets can be formed by encapsulating smaller knowlets to represent large amount of information.
	Related atomic items	Information of various types such as text document, image, sound; or different formats such as PDF, PP slides, QuickTime movies, etc.
Meta-information	Administrative information	Listings of names, types, formats, sizes, access permissions
	Usage-related information	Creation and modification dates, users, utilization of specific parts of methods.
	Content-related information	Keywords, descriptors like meta-tags, references to ontologies or related knowlets.
Specific methods		Simple functions for manipulation of contents, or full-fledged software components on themselves.

Table 2.3: Architecture of Knowlet

Besides, a knowlet registry was introduced to perform centralized organization of all the knowlets in its domain. It collects content-based and usage-based information to support fast identification and similarity-based access to knowlets, and it also supports operation on sets of knowlets.

Poorna R. Bhimavarapu, a Master's student implemented a Component Based Knowledge Management System (CBKMS) for operation systems concepts (Bhimavarapu 1999). As illustrated in Figure 2.5, the system consists of four major components. 1) Knowledge Container, a database designed to store knowlets. 2) Knowledge manager, a front-end tool for managing the Knowledge Container. 3) Knowledge Broker, middle-ware for knowledge retrieval. 4) Knowledge Browser, a user-friendly interface to browse the knowledge.

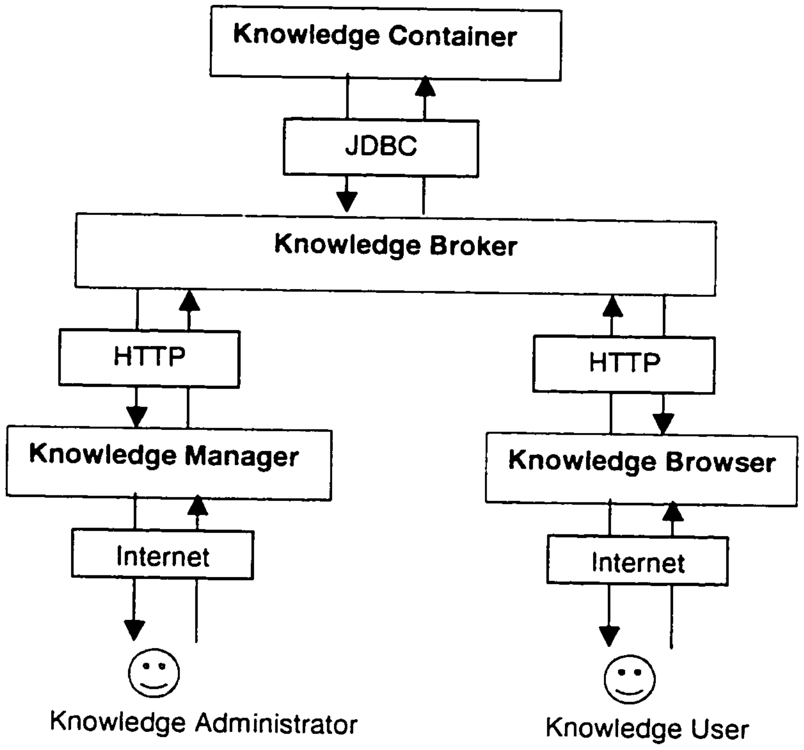


Figure 2.5: Data Flow Diagram

3. KM Support for WBL – A General Study

In Chapter 2, we summarized the categories of tools that are useful in a WBL environment. However, tools and functionalities are not the major subject of this thesis, rather knowledge is. Thus, in this chapter, we explore the following questions: what classes of knowledge are involved in WBL? How to organize knowledge for WBL? What kind of KM supports can be provided for WBL?

3.1 Representing Knowledge in WBL

Delivery of Web-based learning programs involves much more than the delivery of learning resource materials. Similarly, the knowledge in WBL environment compromises more than just the knowledge of course subjects. A knowledge repository for WBL must be identified. The WBL environment can be viewed as a virtual organization which deserves a particular Corporate Memory for its own. Within WBL, any piece of knowledge or information that contributes to the *learning effectiveness* of a WBL environment could be stored in its Corporate Memory.

Inglis (Inglis 1999) proposes several design principles for distance learning programs contributing to learning effectiveness:

- Anticipate the preconditions of the students, and respond appropriately.
- Time, pace, and place flexibility.
- Level of the student's access to learning support.
- Learning style preferences.
- Formative assessment, including self-assessment and summative assessment.
- Alternative entry points. Levels and depths of tuition topics and paths, forms of assessment and points of exit.
- Partial completion of a session; record of progress for each user.

For industrial practice, the content of Corporate Memory includes knowledge about products, production processes, customers, marketing strategies, financial results, strategic plans and goals, etc; For WBL, it obviously should be replaced by knowledge about students and courses related issues. By applying Borghoff and Pareschi's (Borghoff 1998) general architecture for a Corporate Memory to the specific application of WBL environment, I illustrate in Figure 3.1 the Corporate Memory architecture for WBL. Notice that this time the core information repository contains various kinds of knowledge related to WBL.

The case-specific information layer contains students' learning characteristics, study plans and progresses, etc., which differ from person to person. The general information layer contains more sharable data such as question and suggestion bases, planning and scheduling rules, course-related documents and media files. The ontological, or meta information layer is where Abecker's (Abecker 1998) ontological approach of information modelling can be applied. Three knowledge description ontologies – the organization ontology, the domain ontology, and the information ontology, represent the basic assumptions of the underlying WBL system development from three different aspects.

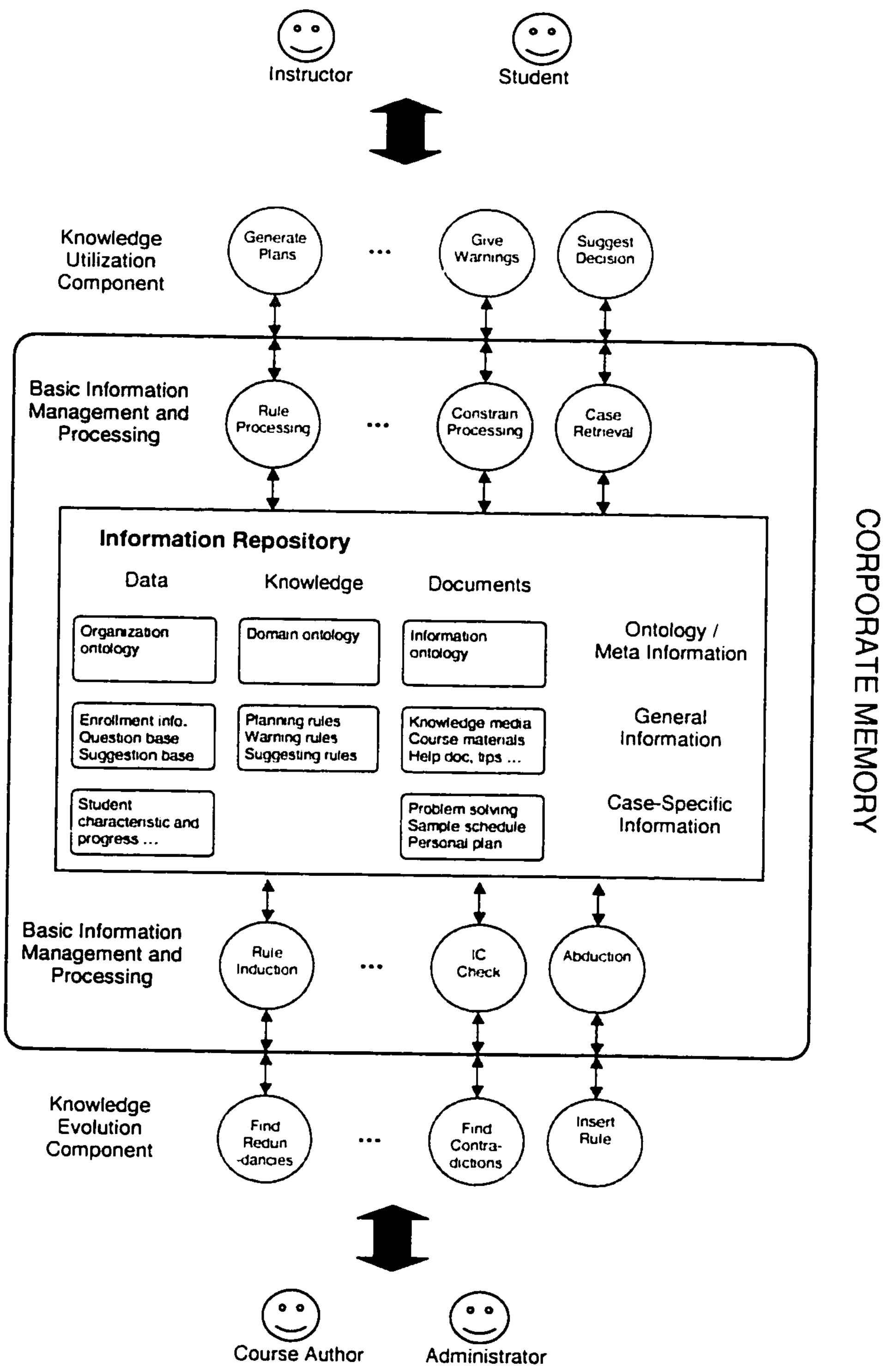

Instructor
Student
Knowledge Utilization Component
Generate Plans
...
Give Warnings
Suggest Decision
Basic Information Management and Processing
Rule Processing
...
Constrain Processing
Case Retrieval
Information Repository
Data
Knowledge
Documents
Organization ontology
Domain ontology
Information ontology
Ontology / Meta Information
Enrollment info. Question base Suggestion base
Planning rules Warning rules Suggesting rules
Knowledge media Course materials Help doc, tips ...
General Information
Student characteristic and progress ...
Problem solving Sample schedule Personal plan
Case-Specific Information
Basic Information Management and Processing
Rule Induction
...
IC Check
Abduction
Knowledge Evolution Component
Find Redun -dancies
...
Find Contra-dictions
Insert Rule
Course Author
Administrator
CORPORATE MEMORY

Figure 3.1: Corporate Memory Architecture for WBL

We further analyzed the structure of the three knowledge description ontologies for WBL as a domain of application (Figure 3.2).

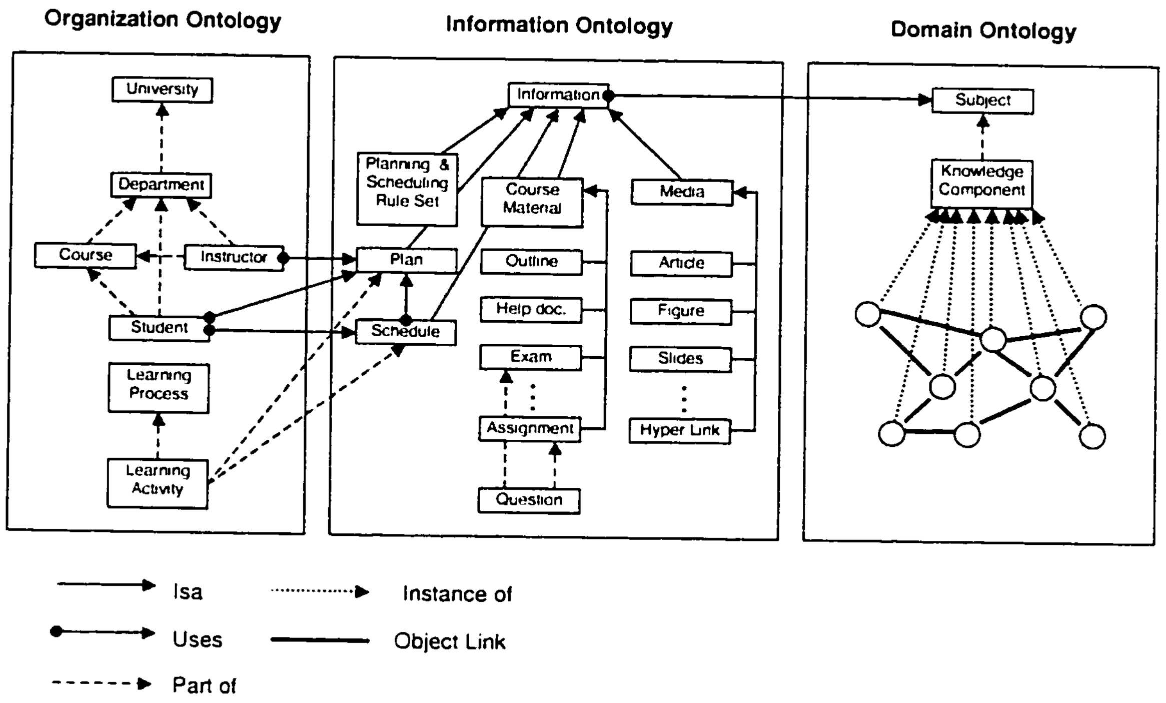


Figure 3.2: Knowledge Description Ontologies for WBL

3.2 CCKM and PCKM

Borghoff and Pareschi (Borghoff 1998) divide KM into process-centered and product-centered views. The former view of KM is a social communication process which can be improved by various aspects of groupware support; the latter refers to knowledge documents, their creation, storage and reuse in computer-based corporate memories. Although the above ideas are most likely generated based on the researchers' study of industry or business organizations, as the word 'product' suggests, it nonetheless provides a way to understand knowledge in WBL. We can roughly divide the KM support for WBL into two broad categories: *Content-Centered Knowledge Management* (CCKM) and *Process-Centered Knowledge Management* (PCKM).

The author of a course prepares knowledge pertinent to a course, his major task is to gather materials available from different sources and then arrange them into topics in the domain field. Future changes might be to update some of the materials, and to modify part of the knowledge structure. These activities all require knowing the actual content of the knowledge and its related materials. I refer to this as Content-Centered Knowledge Management (CCKM). On the other hand, during the course presentation period, most of the scheduled learning activities are time-related and time-bounded: lectures based on different topics are given weekly, assignments are given out with deadlines, quiz, midterm and final exams take place on certain days, etc. Supplemental resources and activities like lectures might not be compulsory, but they might have suggested periods of availability. Other factors are a student's personal learning characteristics, unpredictable events, etc. A good management of those constraints and personalization in constraint satisfaction could contribute to learning effectiveness. Since this perspective of KM is concerned with the learning process and its emphasis is more on human factors, I refer to this as Process-Centered Knowledge Management (PCKM).

3.3 Knowledge Management Supported System Model for WBL

CCKM is a long-term job for a course author, who works at a very detailed level to deal with the structure of subject knowledge and its creation and organization of knowledge media. The knowledge related to CCKM is relatively persistent and stable. On the other hand, PCKM consists of short-term jobs which will not last for more than a whole semester. Most likely, the course schedule for a student will be decided at the beginning of a semester; and a student's personal study plans should not take too much time to make and are expected to reflect the most current changes. Support and suggestions should be provided in time and based on student's characteristics and progresses.

Among the existing WBL tools, support for these two categories of KM are either not clearly identified or not provided. Thus, I propose in this thesis a KM supported system model for WBL incorporating both CCKM and PCKM.

3.3.1 System Architecture and Overview

Figure 3.3 is the architecture diagram for the proposed KM system for WBL.

A KM system for WBL combining CCKM and PCKM together has many advantages. First of all, it supports knowledge sharing between course authors (creators) and instructors (delivery people). Secondly, it will support reuse of knowledge over semesters. Dates of lectures, assignments and exams change in each semester, but their contents may not change. Most of the course planning software we can find on the Web is actually syllabus planning software for instructors. But for the students, this kind of software is almost useless, or just as useful as a course outline. Rather than teaching-oriented planning tools, students would prefer a system incorporating a learning-oriented planning tool, that is a tool that allows them to develop and customize their own study plans.

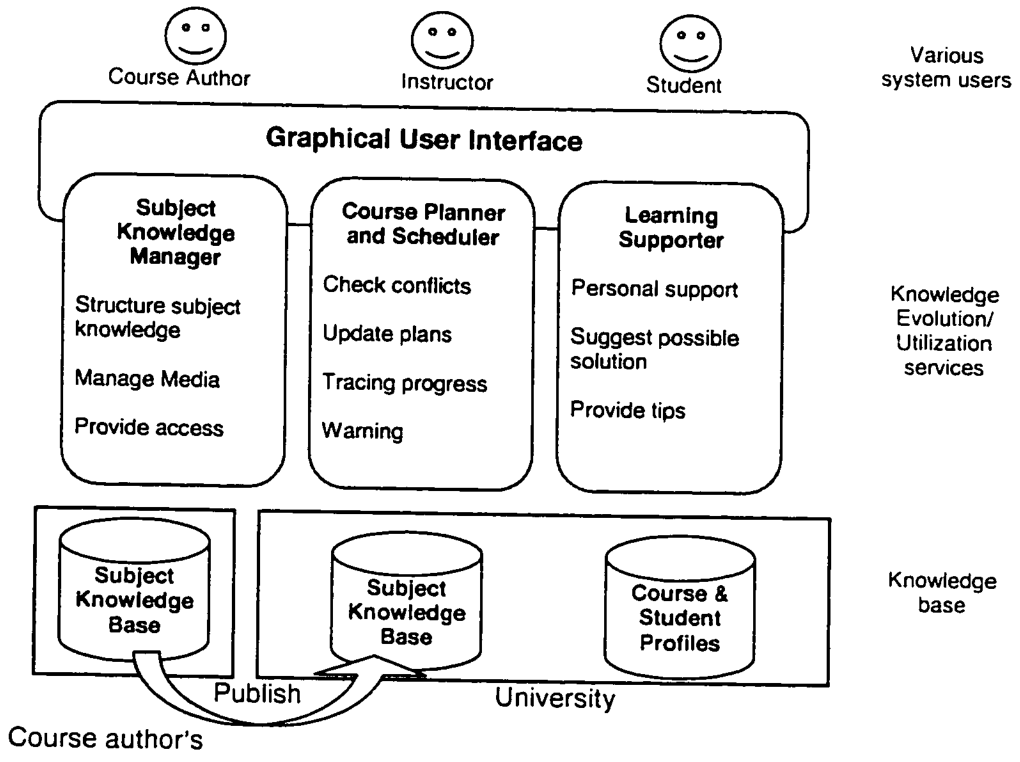


Figure 3.3: KM Supported System Architecture

The role of the publisher is to disseminate the domain knowledge developed by the course authors to the instructors in various universities. Like publishing a book, the publisher should first proofread and correct the author's work. But finally the domain knowledge is published as CD-ROMs or online libraries instead of printed books.

3.3.2 User Profiles and Requirements

Three categories of users will be involved in this project. They are *course author*, *instructor* and *student*. Although course author and instructor can be the same person, these are two different roles and should be separated. Compared to instructors, course authors have much greater freedom for organizing knowledge. Students form a diverse variety of Web-based learners.

Table 3.1 lists the top-level description of the tasks performed by these three categories of users.

Course Author	Instructor	Student
• Create/structure/update subject knowledge • Create/update knowledge media • Manage knowledge meta-information • Search for knowledge	• Access subject knowledge • Create/add knowledge as supplements • Create/update course materials • Create/update course plans (time line) • Manage student progress • Provide personal advice to students • Evaluate students performance	• Access subject knowledge • Access course plans • Access course materials & create personal study plans • View learning progress • Get suggestions and tips • Self testing

Table 3.1: User Tasks

3.3.3 Knowledge Evolution/Utilization Services

The core of the system consists of the following three parts.

1. **Subject Knowledge Manager**

The Subject Knowledge Manager is a middleware component containing the whole business logic of subject knowledge and controls the access to the Subject Knowledge Base. It is responsible for the creation and organization of knowledge structure, knowledge media, and knowledge meta-information. It handles different access privileges. For example, a course author may have full access (add/delete/update) to the knowledge of several subjects stored in the Subject Knowledge Base. An instructor cannot modify the knowledge structure but can add knowledge media to the subject he is teaching. A student usually has read-only access to the Subject Knowledge Base.

2. Course Planner & Scheduler

The Course Planner & Scheduler is responsible for maintaining course plans and schedules. It uses the rules stored in the knowledge base to check the contradictions among multi-level plans, updates schedules to reflect most resent changes, and outputs warnings and suggestions concerning students' progresses.

3. Learning Supporter

The Learning Supporter provides all kinds of supports to students in order to improve their learning effectiveness. These supports include: providing help to access documents and learning tips; retrieving similar cases from the knowledge base to solve problems; provide suggestions based on student's progress and characteristics, etc.

3.3.4 Database and Knowledge Base

- **Subject Knowledge Base**

The underlying Subject Knowledge Base contains the domain knowledge for various courses. It stores knowledge components as well as the relationships among knowledge components in order to support different views of the domain knowledge. Both course author and instructor can access the Subject Knowledge Base. The Subject Knowledge Base used by the course author is located at the course author's private place. It may contain both structured knowledge and unstructured knowledge. The Subject Knowledge Base used by the instructor is located locally at a university or place of the use.

- **Course and Student Profiles**

Course and Student Profiles contains all the course related information such as syllabus, lectures, labs, assignments, projects, exams, course schedules etc. In order to provide support to students, it should also maintain information about students, including personal study plan, learning preference, etc.

In the following two chapters, we will explore two KM strategies in more depth: Subject Knowledge Modeling which belongs to CCKM, and Dynamic Course Planning-cum-Scheduling which is of the PCKM type. Dynamic Planning-cum-Scheduling is also the key process connecting CCKM and PCKM.

While addressing the two KM strategies, we use an course in the Computer Science Department at Concordia University (COMP 520 Computer Organization and Assembly Language) as a case study. Dr. V. Rujaraman and Dr. Radhakrishnan's book *Assembly Language Principles for IBM PCs* (Radhakrishnan 2000) will be used as the major reference to structure subject knowledge as well as the source of text-based course materials.

4. KM Strategy: Subject Knowledge Modeling

In this chapter, we address the following questions: (1) How to represent the well-structured knowledge of a course subject? (2) How to incorporate knowledge dependencies into the knowledge structure? (3) How to incorporate media contents into the knowledge structure?

4.1 Knowledge Tree As Structure

Since the subject knowledge is usually structured hierarchically, I propose using a *knowledge tree* (K-tree) to visualize the structured subject knowledge, which is easy to understand and easy to realize in a computer system. A K-tree is a tree where the contents of each parent node are composed of the contents of all its child nodes, and no two nodes in the same level have overlapping content.

4.1.1 Defining Tree Node

Each node in the tree is a self-contained structure encapsulating all of its sub-nodes as well as the following information:

- **Parent K-tree**

The K-tree containing the node.

- **Type**

All the nodes in the knowledge tree fall into one of four categories: *subject, modules, sub-modules* and *topics*. Subject (S) is represented as the root of the knowledge tree. Topics (T) are represented as terminal nodes or leaves. Modules (M) are non-terminal nodes that connect directly to the root. Sub-modules (SM) are all the non-terminal nodes between modules and topics.

- **Object ID**

Each node is associated with an *Object ID* (OID) indicating its hierarchical position in the K-tree (see Figure 4.1).

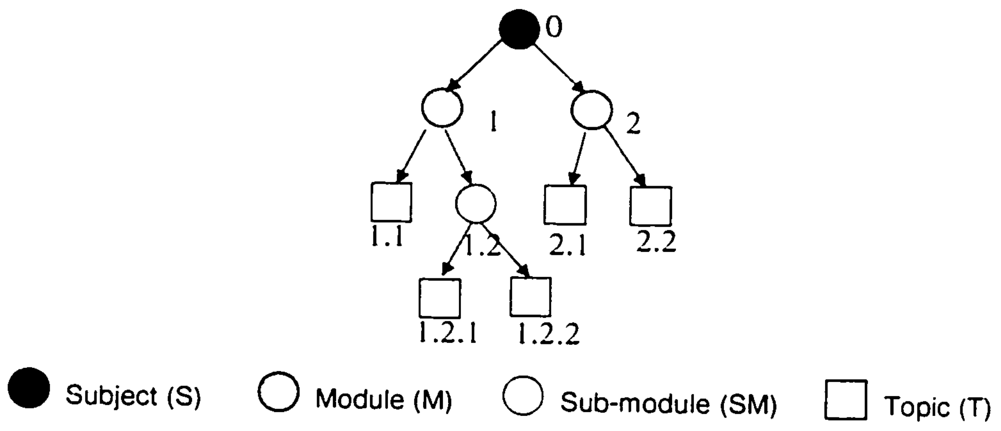


Figure 4.1: A Simple K-Tree with OID

In the real implementation, using OID to represent the hierarchical structure of K-tree is more advantageous than using pointers. First, it saves space in database and memory. Because for each node, using OID only requires one extra field whereas using pointers demand at least two: one pointing to the node's parent and the other pointing to its next sibling. Secondly, with OID, it is easier to retrieve a sub-tree from the database, which is a very frequent operation. We can get all the qualified nodes only by one query – select the nodes whose OID begin with the OID of the root node of the sub-tree. It is much more efficient than traversing through all the pointers.

- **Abstract**

The abstract of a node is a brief text description summarizing the knowledge contents of that node.

- **Weight**

The weight associated with a node allows an instructor and student to have an idea about how much time it takes to learn the knowledge contained in a certain node. It is useful for making course plans and learning schedules. Weight is not constant. Rather it depends how much a node and all its children are actualized by media contents. A conceptual node with no actual media contents has a weight of zero. When more and more media are implemented for a node and its sub-nodes, its weight grows. Weight can be normalized to be from 0 to 10.

- **Access permissions**

Several levels of access permissions have been identified:

1. *Full Structure Access* – User have full access to the tree node, which means user can modify the structure of the whole sub-tree rooted at that node. User can also add/delete/update its media contents.
2. *Full Media Access* – User can add/delete/update media content of a node, but can not modify its structure.
3. *Add-media Access* – As the name implied, user can only add media contents but cannot delete and modify them.
4. *Read-only Access* – User can only read the node.

Generally, the course author has Full Access to the tree nodes; an instructor has Full Media Access or Add-media Access; a student has Read-only Access.

- **Modification log**

Whenever a new node is added, or the position of a node is shifted, or a new media file is added to a node, user should be notified so as to get the most up-to-date knowledge. The modification log records all the related information.

- **Media prototypes**

K-tree serves as a map not only for indicating to the students how different modules, sub-modules and topics are arranged, but also for providing entries to access related media contents. Each node is associated with a collection of media prototypes implementing that node. Section 4.3 explains this in more details.

A sample tree node is illustrated in Table 4.1.

Type	Topic (T)
Parent K-tree	Assembly Language
Object ID	2.2
Abstract	Central Processing Unit (CPU)
Weight	3
Access permissions	Full Access
Modification log	Date: 20/1/02; User: Jia Zhao; New media added: Figure/CPU Architecture
Media prototypes	Article/CPU Overview Figure/CPU Architecture

Table 4.1: A Sample Tree Node

4.1.2 A Sample Knowledge Tree

The textbook *Assembly Language Principles for IBM PCs* (Radhakrishnan 2000) is converted to a knowledge tree for learning Assembly Language. The knowledge tree is composed of eight Modules and each of them is composed of several Topics. The following list gives Types, OIDs and names of all the nodes in the knowledge tree.

Assembly Language

M1 Introduction

T1.1 What is Assembly Language

T1.2 High Level Language and Assembly Language

T1.3 Why Do We Need Assembly Language

T1.4 When Do We Use an Assembly Language

T1.5 Major Parts of an Assembly Language

T1.6 Assembly Language for the IBM PC

M2 Hardware and Software Overview

T2.1 IBM PC Hardware Architecture
T2.2 Central Processing Unit
T2.3 Model for Main Memory
T2.4 CPU Registers
T2.5 Basics of Assembly Language
T2.6 Instruction Set
T2.7 Codes for Data Allocation
T2.8 System Software

M3 Assembly Process and Program Execution

T3.1 Symbol Table
T3.2 Op-code Table
T3.3 Assembly Process
T3.4 Linking Stage
T3.5 Executable Module
T3.6 Executing a Program
T3.7 Software Interrupts

M4 Addressing Modes

T4.1 Why Addressing Mode
T4.2 Immediate Addressing
T4.3 Register-Indirect Addressing
T4.4 PC Relative Addressing
T4.5 Displacement Oriented Addressing

M5 Stack

T5.1 Stack and Operations for It
T5.2 The Use of Stack
T5.3 Calculating Value of an Arithmetic Expression

M6 Arrays, Arithmetic and More

T6.1 Defining Arrays

T6.2 Conversion: ASCII-Binary

T6.3 Multiplication and Division

T6.4 Conversion: Uppercase ASCII

T6.5 NEAR and FAR Points in Segment

M7 Subroutines

T7.1 Why Subroutines

T7.2 Calling Subroutines

T7.3 Passing Parameters to Subroutines

T7.4 Recursive Subroutines

T7.5 Calling Subroutines in Other Segments

M8 Macros

T8.1 Defining and Using a Macro

T8.2 Expending a Macro

T8.3 The LOCAL Directive

T8.4 Passing Parameters

T8.5 Macro and Subroutines

Figure 4.2 displays the knowledge tree structure for Assembly Language.

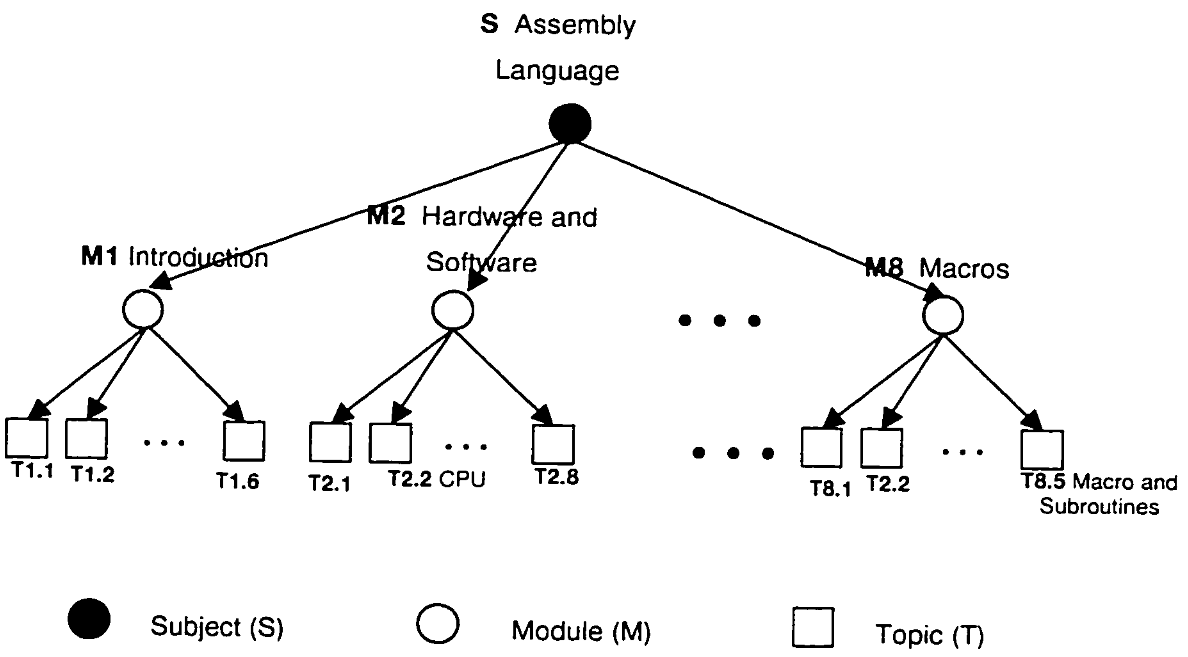


Figure 4.2: Knowledge Tree for Assembly Language

4.2 Knowledge Dependencies

A simple knowledge tree may only contain part-whole (or parent-child) relationships, as the one illustrated in Figure 4.1. However, a knowledge tree can be more generalized by incorporating other relationships, among which the *knowledge dependency* is especially important. Knowledge dependency means that a tree node establishes the foundation of learning the material related to another node. When we say node A depends on node B, it suggests that node A refers to node B, or node B is the prerequisite of node A.

4.2.1 DAG Diagrams

In order to improve learning effectiveness, it is necessary to present knowledge dependencies to instructors and students. Instructors are largely concerned with knowledge dependencies during the course planning process, because he wants to make sure that the temporal arrangement of the leaning activities does not conflict with the knowledge dependencies. Chapter 5 addresses the issues about course planning. Knowledge dependencies are crucial to students, too. They allow students to trace back the related knowledge to a specific problem.

For knowledge dependencies, I find Directed Acyclic Graph (DAG) to be simple and intuitive. Taking the K-tree of Assembly Language for example, Figure 4.3 shows the DAG representing the top-level knowledge dependencies between modules.

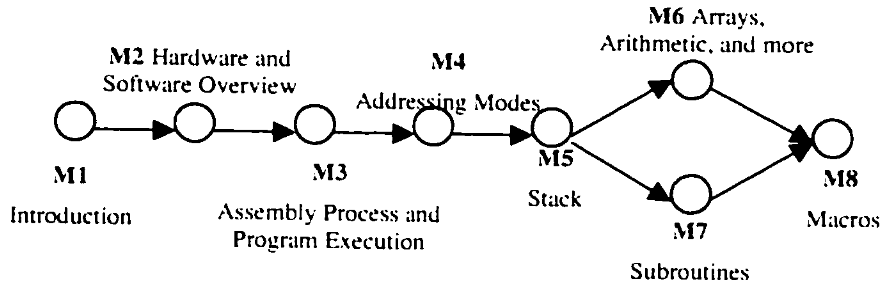

M6 Arrays, Arithmetic, and more
M2 Hardware and Software Overview
M4
Addressing Modes
M5
Stack
M1
Introduction
M3
Assembly Process and Program Execution
M7
Subroutines
M8
Macros

Figure 4.3: Top-level Knowledge Dependencies for Assembly Language

Knowledge dependencies not only exist between modules, but also between topics, and between modules and topics. To represent those dependencies, a much more complex DAG need to be generated. This is out of the scope of my thesis and will not be addressed here.

4.2.2 Dependency Matrix and Dependency Pair List

DAG diagrams are just visual representations of knowledge dependencies to users. In the computer system, knowledge dependencies can be implemented in two ways, by associating a *Dependency Matrix* with a K-Tree, or by associating a *Dependency Pair List*. Both the Dependency Matrix and Dependency Pair List are conversions of the DAG diagram. Table 4.2 is a Dependency Matrix converted from Figure 4.2. Cell value 1 means the column **directly** depends on the row.

Supporter \ Dependent	M1	M2	M3	M4	M5	M6	M7	M8
M1	0	1	0	0	0	0	0	0
M2	0	0	1	0	0	0	0	0
M3	0	0	0	1	0	0	0	0
M4	0	0	0	0	1	0	0	0
M5	0	0	0	0	0	1	1	0
M6	0	0	0	0	0	0	0	1
M7	0	0	0	0	0	0	0	1
M8	0	0	0	0	0	0	0	0

Table 4.2: Dependency Matrix for Assembly Language

Table 4.3 shows the Dependency Pair List corresponding to Figure 4.2.

Dependent	Supporter
M2	M1
M3	M2
M4	M3
M5	M4
M6	M5
M7	M5
M8	M6
M8	M7

Table 4.3: Dependency Pair List for Assembly Language

Compared to a Dependency Pair List, the Dependency Matrix is easier to implement and easier to operate on, but it occupies more space. It is suitable if we only consider the top-level dependencies between modules. However, if lower level dependencies among sub-modules and topics are to be considered, it is not efficient to generate a Dependency Matrix for every sub-tree, as there are too many nodes involved. Then Dependency Pair List is a good substitution.

4.3 Media Prototypes

A knowledge tree without any media contents but only a skeleton is as useless as a library without any books that has only catalogues. Students learn from explanations provided in different media form, by connecting a concept to the real world cases, by visualizing and practicing, etc. Those are achieved through using media. In order to improve learning effectiveness, a student should have access to various media atomic items. The way those media atomic items are arranged and accessed is crucial. I suggest attaching all the media atomic items to the knowledge tree, and each of them, whether being text, image, animation, PowerPoint slides, etc. can be regarded as a *media prototype* of a tree node based on its content. By visiting a tree node, a student can access a collection of media prototypes implementing that node. Each media prototype is associated with the following information:

- **Abstract**

The abstract of a media prototype is a brief text description summarizing its content.

- **Parent node**

The node implemented by this media prototype.

- **Media type**

Type of the media prototype, such as text, slide, image, animation, movie, sound, etc.

- **Content Type**

Type of the media content, such as article, exercise, example, algorithm, etc.

- **Format**

File format of the media prototype, such as TXT, JPG, DOC, PPT, GIF, WAV, etc.

- **Access permission**

Who created the media prototypes? Both course authors and instructors are authorized to do so. When a knowledge tree is published, it already contains many articles, figures, examples and exercises created by the course author, which usually can be accessed publicly. While the knowledge tree is being used for a course, the instructor can also add media prototypes as well as specify the access permission of those media prototypes. Generally there are four basic access privileges:

1. *Public* – The media prototype is visible to all the users.
2. *Course* – The media prototype is visible to instructors and students of a specific course.
3. *Section* – The media prototype is visible to instructors and students of a course-specific section.
4. *Personal* – The media prototype is only visible to its creator.

- **Presentation time**

Presentation time gives the user an estimate of how much time it takes to use a media prototype once. For some type of media, such as sound and animation the media presentation time is fixed. For others, such as text document, image and exercise, the media presentation time can only be estimated. It should be pointed out that presentation time is different from learning time. While presentation time is independent of a user's characteristic, learning time is always associated with the user's cognitive style, memory type, learning motivation, etc. and cannot be predefined. But it can be estimated, especially if statistics from previous users are available.

- **Creation information**

Creation information includes the author, creation date of a media prototype.
A sample media prototype is show in Table 4.4.

Name	CPU overview
Parent node	Topic 2.2 CPU
Media type	Text
Content type	Article
Format	HTML
Access Permissions	Public
Presentation time	20 min.
Creation info.	Radhakrishnan 2000/4/2

Table 4.4: A Sample Media Prototype

Figure 4.4 illustrate the K-tree for Assembly Language with two media prototypes added to Topic 2.2 on CPU. A published K-tree will have much more media prototypes.

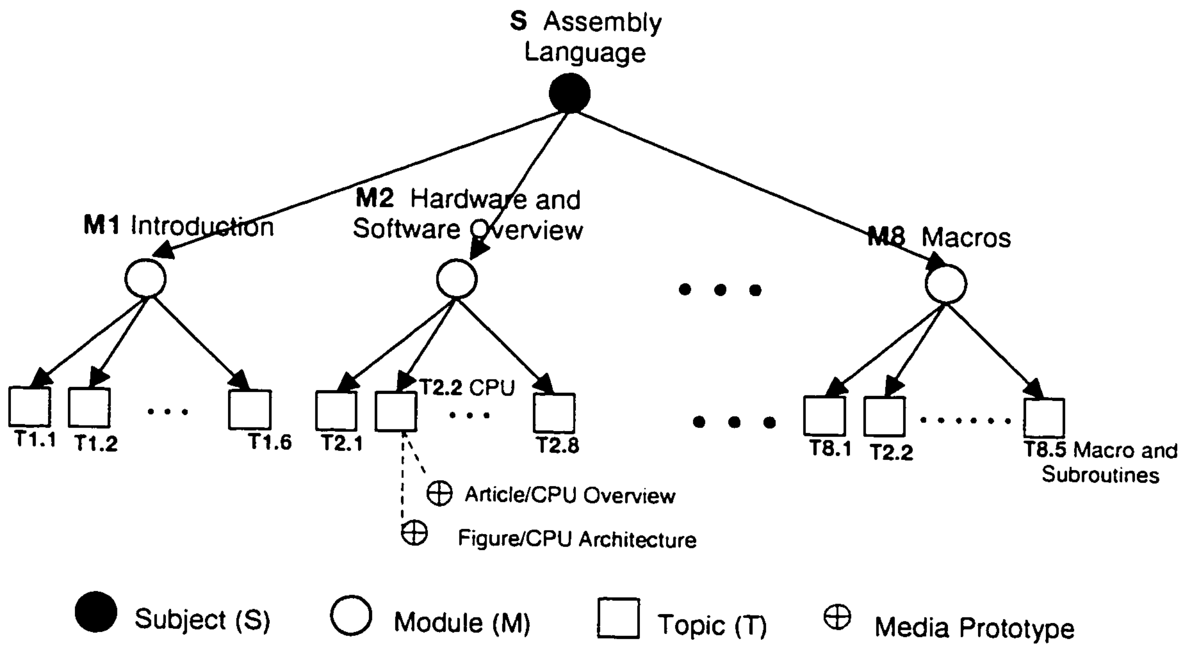


Figure 4.4: Knowledge Tree with Prototypes

5. KM Strategy: Dynamic Planning-cum-Scheduling

5.1 Overview of Scheduling and Planning

5.1.1 Defining Planning

Both plans and planning are well studied in the Artificial Intelligence area. Shinghal (Shinghal 1992) defines: "A *plan* provides an appropriate sequence of actions to meet the goal of a character in the story." Rich (Rich 1991) gave a definition for *planning*: " There are methods that focus on ways of decomposing the original problem into appropriate subparts and on ways of recording and handling the interactions among the subparts as they are detected during the problem-soling process. The use of these methods is often called *planning*."

5.1.2 Defining Scheduling

The difference between planning and scheduling is that while planning is concerned with "what" is to be done, "where," by "whom," and "when"; scheduling additionally deals with "how" things are to be done. However, a schedule shouldn't be regarded as a fixed timetable of activities, but just determine "when" a certain task will be performed relative to other tasks. We deal with scheduling on an ongoing basis, and the degree to which we carry it out and the techniques we use vary depending upon the complexity of our situations and our needs and objectives (Hinze 1998).

5.1.3 Heuristic Search

Heuristic search is important to planning and scheduling. Artificial Intelligence (AI) has a tradition of employing human-like problem solving knowledge which can sometimes be expressed as heuristic. In AI the challenge has been to encode and control such knowledge. *Heuristics*, by definition is problem-solving-knowledge, and thus is problem specific. Procedures which employ domain specific information have been proven most effective in constructing heuristics (Brown 1995). Project planning, transportation scheduling and production scheduling are traditional application areas.

5.2 Course Planning

WBL involves course planning. In order to achieve the major goal, that is successfully completing a course, both instructor and student make course plans. The instructor's planning process involves decomposing a course into various required learning activities, such as lectures, quizzes, exams, assignments and projects, deciding their contents, and arranging them into an appropriate chronological sequence respecting the knowledge dependencies. The student's planning process involves incorporating self-defined learning activities into the instructor's plans, and defining "when" and "where" those activities will be accomplished.

5.2.1 Downward compatibility

Within a university, there exist different levels of plans. Their downward compatibility can be represented by a tree structure as shown in Figure 5.1.

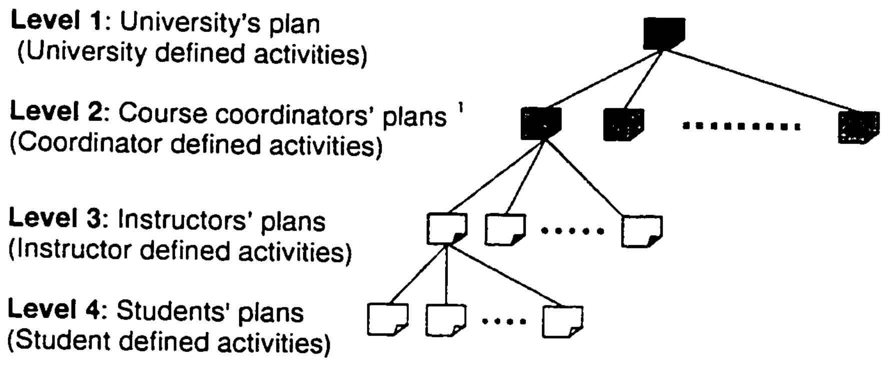

Level 1: University's plan
(University defined activities)
Level 2: Course coordinators' plans [1]
(Coordinator defined activities)
Level 3: Instructors' plans
(Instructor defined activities)
Level 4: Students' plans
(Student defined activities)

Figure 5.1: Upward Compatibility of Plans

1. A coordinator coordinates a multi-section course to ensure uniformity across multiple sections.

Downward compatibility means that the lower level plans should incorporate the activities defined in the upper level plans. For example, usually the beginning and end date of each semester, school holidays, mid-term break and final examination period are defined in the university's plan. Thus all the course-related activities defined at lower levels should not violate those dates. Similarly, if the course coordinator decided which topics should be covered in the course, instructors are not allowed to omit those topics although they are allow to add other topics up to a certain point. Moreover, each activity defined in an instructor's plan is a milestone or sub-goal for students to respect in their personal plans. Upward compatibility also means that any modification in a high level plan will cause corresponding changes in the low level plan.

5.2.2 Instructor's Plan

This thesis focuses on two levels of plans: instructor's plan and student's plan. An instructor's plan should at least contain the following information:

- What types of activities and how many of them should the students perform in order to obtain a "pass" in the course? General types of activities are lecture, assignment, lab practice, quiz, exam, project and reading report, etc.
- What is the knowledge content of each activity? If it is a lecture, which topics will be covered? If it is an assignment, what questions will be answered? If it is a reading report, what readings should be assigned?
- What is the date of lectures, lab practices, quizzes and exams?
- What is the submission deadline for assignments, projects and reading reports?
- How is a final mark computed and what is the penalty if a student misses a deadline or fails to accomplish an activity?

The instructor's plan is usually made at the beginning of each semester and should be relatively fixed, since it mainly serves as a series of goals for students, and shifting goals will confuse the students and hinder their learning process. However, there do exist some special cases that may cause an instructor to modify the course plan during the semester.

For example, an instructor may extend the deadline of an assignment for a student who got a medical note.

5.2.3 Student's Plan

As we've seen, the instructor only defines the constraints for some activities. When and how to accomplish those activities is up to students to decide. Different students will make different plans based on their characteristics and preferences. A student's plan generally contains the following information:

- The information inherited from instructor's plan.
- For those activities having a deadline, such as assignments, projects, and reading reports, it is infeasible to specify exactly when and how much time should be spent on them. For each activity, a student can at least allocate a *time-span* before the deadline but not too far from it. By definition, time-span is several continuous days within which the corresponding activity would better be accomplished in order to incur the least penalty.
- For those activities happening on particular dates, some have to be prepared, such as quizzes, exams; some others would be better prepared if the students have enough time, such as lectures and lab practices. Thus what's the time-span for those preparation activities?
- Extra learning activities defined by the student him/herself.
- Predictable and unpredictable *events* which hold up the learning process should also be incorporated into the student's plan. For example, a school trip is a predictable event and an illness is an unpredictable event.

Compared to the instructor's plan, a student's plan is much more flexible and is re-planned on an ongoing basis. Because, in practice, students' characteristics, or learning preferences, the events they encounter will all influence the way they plan. They also keep modifying the plans to adapt to current situations. As long as the plan finally leads to a successful completion of the course, there are many ways to plan. Another thing that needs to be pointed out here is that if the instructor makes a change to his plan, the student's plan needs to be refreshed immediately in order to reflect the change.

5.2.4 Defining Learning Activities

Based on the above analysis, I summarize in Table 5.1 the information encapsulated in learning activities, which distinguishes them from each other.

Name	Name of the activity.
Type	Lecture, lab, quiz, assignment, project, exam, report or any other type defined by instructors and students.
Plan	Which plan the activity belongs to. (Who makes the plan for which course and section in which semester?)
Deadline/date	The date required to submit the results of an activity, if applicable, after which a penalty will probably be applied. Or the date when a single-day activity happens.
Count for Evaluation	The percentage the activity constitutes of the final mark.
Knowledge Content	Link to the nodes in the corresponding K-tree.
Time Span	Several continuous days within which the activity would better be accomplished in order to cause the least penalty.
Workload	An estimated weight of the activity, which might be represented by the number of daily work units.
Late Penalty	The percentage of marks the students will loose per day if they miss the deadline.

Table 5.1: Properties of Learning Activities

5.2.5 Planning with Knowledge Tree – A Case Study

As we previously mentioned, planning is not just about "when" to do, but also "what" to do and "how" to do. When a plan incorporates knowledge, it becomes more detailed and more useful. Usually it is up to the instructor to decide the content of each learning activity. The instructor should take the knowledge tree and knowledge dependencies into consideration while introducing and arranging activities. It seems easy to assign knowledge contents to applicable activities such as lectures - by linking each activity to one or more tree nodes. However, the difficulty is in making sure that the arrangement does not conflict with knowledge dependencies.

Although with a K-tree and DAG diagram at hand, an instructor can always refer to them during the planning process to check knowledge conflicts, it is preferable that a software tool assists in this job. Following is a proposed solution, in which I only consider the top-level dependencies between modules. Solutions for more complex dependencies will be considered in future work.

First, let's define a *Knowledge Allocation Table* for each instructor's plan, which stores the current values of two Boolean attributes of all the nodes in the corresponding K-tree. The two attributes are "*assigned*" and "*assignable*". For each node in the knowledge tree, "assigned" being true means the node has been assigned to or will be learned in previous lectures. The attribute "assignable" being true means two possibilities: 1) All the node's all supporting node have been assigned to or will be learned in previous lectures so that this node becomes assignable; 2) the node does not depend on any other nodes. The computer system will also keep a copy of the Dependency Matrix for each instructor's plan and keep modifying it to reflect the changes during the planning process.

An example: the instructor is planning for the Assembly Language course using the corresponding K-tree (Figure 4.1), and suppose M1, M2, M3, and M4 have already been assigned. The current Knowledge Allocation Table and Dependency Matrix are show in

Table 5.2 and Table 5.3 respectively. Note that in the Dependency Matrix, a row of zeros means the node representing that row is assigned, and a column of zeros means the node representing that column is assignable. Thus, Table 5.3 is consistent with Table 5.2.

	assigned	assignable
M1	T	T
M2	T	T
M3	T	T
M4	T	T
M5	F	T
M6	F	F
M7	F	F
M8	F	F

Table 5.2: Knowledge Allocation Table (Before)

Dependant / Depended	M1	M2	M3	M4	M5	M6	M7	M8
M1	0	0	0	0	0	0	0	0
M2	0	0	0	0	0	0	0	0
M3	0	0	0	0	0	0	0	0
M4	0	0	0	0	0	0	0	0
M5	0	0	0	0	0	1	1	0
M6	0	0	0	0	0	0	0	1
M7	0	0	0	0	0	0	0	1
M8	0	0	0	0	0	0	0	0

Table 5.3: Dependency Matrix (Before)

Now suppose the instructor assigns M5. Responding to that, the system changes all the cells in row M5 to zero in the Dependency Matrix. Next, it checks every column and finds that now all the cells in columns M6 and M7 have became zeros. This means node M6 and M7 become assignable. The Knowledge Allocation Table and Dependency Table after these changes are shown in Table 5.4 and Table 5.5 respectively. Again, they are consistent.

	assigned	assignable
M1	T	T
M2	T	T
M3	T	T
M4	T	T
M5	~~F~~ T	T
M6	F	~~F~~ T
M7	F	~~F~~ T
M8	F	F

Table 5.4: Knowledge Allocation Table (After)

Dependant / Depended	M1	M2	M3	M4	M5	M6	M7	M8
M1	0	0	0	0	0	0	0	0
M2	0	0	0	0	0	0	0	0
M3	0	0	0	0	0	0	0	0
M4	0	0	0	0	0	0	0	0
M5	0	0	0	0	0	~~1~~ 0	~~1~~ 0	0
M6	0	0	0	0	0	0	0	1
M7	0	0	0	0	0	0	0	1
M8	0	0	0	0	0	0	0	0

Table 5.5: Dependency Matrix (After)

Since Modules are sub-divided into Topics, another question arises: in practice, instructors are more likely to assign Topics to activities rather than Modules. How to make sure a whole Module has been assigned? To solve this problem, we can add another property *child-count* to tree node. The initial value of child-count equals to the number of children the node has. Every time a child node is assigned, the child-count will count down by one. Once the child-count reaches 0, the node would become assignable.

5.3 Dynamic Learning Activity Scheduling

I discussed course planning in last section, mainly focusing on how to decompose a course into various learning activities respecting the knowledge dependencies. Once a course plan is arrived at, it must be used in the real learning process. Here arises the scheduling problem concerning the exact sequential execution of the predefined learning activities. Scheduling the learning activities is crucial to students in WBL. A feasible schedule will lead to successfully completing the course, while an infeasible one can only become a burden to the student. What kind of schedule is feasible? First it should incorporate the student's characteristics, constraints, and learning preferences. Secondly, it should be adaptable to new situations and changes. Thirdly, it should provide flexibility and violate as few constraints as possible. Forth, it should always let the student know what is likely to happen in the next time interval based on the current situation.

5.3.1 Dispatching Rules and Critical Degree

A *dispatching rule* is a kind of heuristics used to select the next job to be processed from a set of jobs awaiting service (Brown 1995). A simple dispatching rule is to select the job with the closest due date. Applying a well-defined dispatching rule to a collection of well-defined activities will always yield the same execution sequence. However, in real learning practice, there is obviously more than one route to complete a course, each student choosing one differing from others. While it is impossible to force all the students to follow a single route of learning, the routes they choose should be checked from time to time to verify if they still satisfy all kinds of constraints. Thus, an optimal solution is to generate a referential schedule – a sequence of executions the students need not strictly follow – using a proper dispatching rule. Moreover, each activity is associated with a *critical degree* representing its current urgency. In practice, the student does not necessarily follow the generated sequence to accomplish the activities. Rather, based on the critical degrees, he is allowed to optimize his own executions while not to put himself into the danger of failing an activity or even failing the course.

If we use such a dispatching rule that selects the activity with the shortest due date for next execution, there would be a problem. For example, if an assignment and a project have the same due dates, then based on the dispatching rule they will have the same priorities. However, usually the workload of a project is much heavier than that of an assignment and ideally the former should be started earlier. To solve this problem, I designed a dispatching rule for learning activity scheduling which takes two other activity properties into consideration: <u>time-span</u> and <u>workload</u>.

A proposed dispatching rule is to select the activity with the highest *critical degree*. By definition, critical degree (C) has direct ratio with the *current workload* (W) and inverse ratio with the *current time-span* (T): C=CW/CT. I add "*current*" before both workload and time-span because scheduling is an on going process. If the date is within a predefined time-span, then the current time-span should be counted from the current date. Also, if the student has finished several work units of an activity, then the current workload should be the left over work units. Thus, the critical degree of an activity keeps changing with time. Depending on the value of critical degree, Table 5.6 defines five activity states.

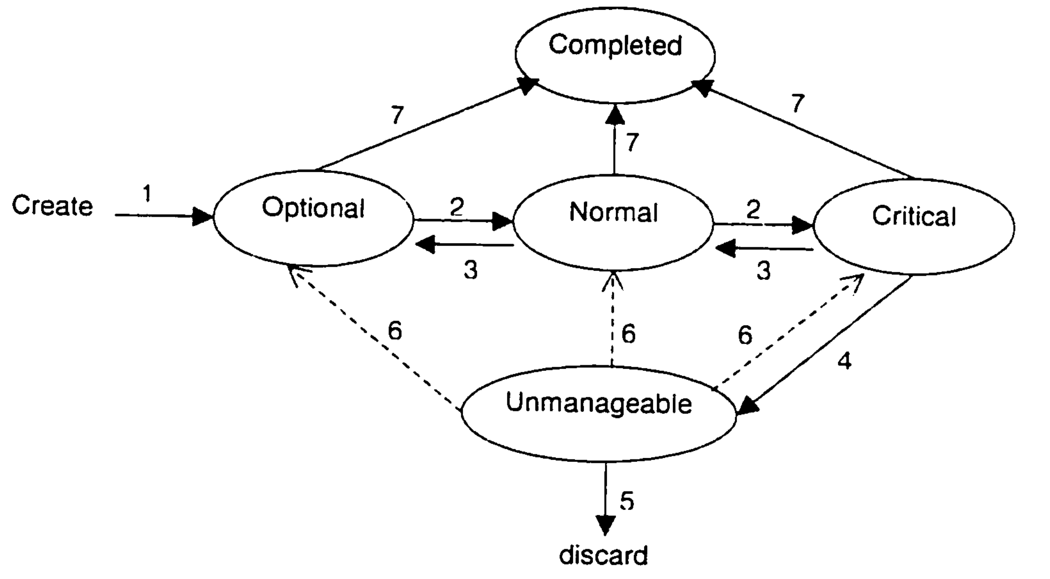
Critical degree	State of the activity
$0 < C < 1$	*Optional* -- student has more than enough time to finish the activity, and it is OK for him/her to postpone doing the work.
$C = 1$	*Normal* – student gets just enough time to finish the activity, postponement will lead to changing the state of the activity to 'Critical'.
$C > 1$	*Critical* - student is short of time, and he/she should do it in an accelerated manner in order to finish it in time.
$C = 0$	*Completed* - the activity has been done successfully within the time span.

T = 0 && W > 0	*Unmanageable* - the activity wasn't done successfully within the time span, and normally cannot be redeemed.

Table 5.6: Activity States

5.3.2 Activity Life Cycle

Based on the previous specification of various activity states, Figure 5.2 illustrates the life cycle of an activity in the dynamic scheduling process.



1 External events cause this
2 Passage of time
3 Partial work
4 No timely-attention given
5 Removal / penalty
6 Deadline is extended
7 Timely work put in adequately

Figure 5.2: Activity Life Cycle

The solid arrow lines represent the most common route an activity may traverse. When an activity is created, it is normally in the 'Optional' state. If it is left undone, the state will gradually change from 'Optional' to 'Normal', and then from 'Normal' to 'Critical'.

Conversely, once an activity is in the 'Critical' state, it can also change back to 'Normal' and 'Optional' states if the student notices the critical situation and works hard enough to catch up the lost time. The student is free to choose to complete an activity in whichever states among 'Optional', 'Normal', and 'Critical', although it is reasonable to assume that 'Critical' activities will have higher priority. Normally, when a 'Critical' activity changes to 'Unmanageable', it cannot traverse back and should be discarded. But an exception does exist - that is when the instructor extends the deadline of an activity, and thus brings the 'Unmanageable' back to life. In Figure 5.2, such a situation is represented by dotted arrow lines.

5.3.3 State of the Scheduling Process

In WBL, scheduling is a dynamic and on-going process. This is because the major input of scheduling (that is activities) is always changing with time, both in number and in its properties. This means: Firstly, while the predefined activities are finished one after another, new activities may be introduced into the course plan. Secondly, the critical degree of each activity is changing from day to day, or even from moment to moment. Thirdly, since hardly any aspect of the real world is completely predictable, while a schedule is being executed, unforeseen things may happen and disrupt the schedule.

To adapt to the continuously changing world, one solution is to keep a reschedule-execute-reschedule cycle. When the situation changes, we just throw away the rest of the schedule and start the scheduling process over, using the current situation as the new initial state. For a certain activity, the reschedule-execute-reschedule cycle may proceed toward a *point of no-return*, which means no matter how we schedule, it is impossible to accomplish the job (missing the deadline for example). In this case, we can simply abandon the failed activity and start over again. Figure 5.3 is a state diagram for scheduling.

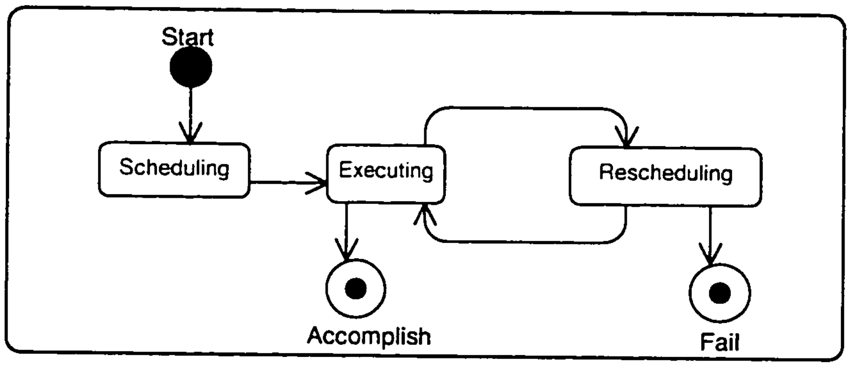


Figure 5.3: State Diagram for Scheduling

5.3.4 Scheduling Inputs and Outputs

The inputs and outputs of the scheduling process are illustrated in Figure 5.4.

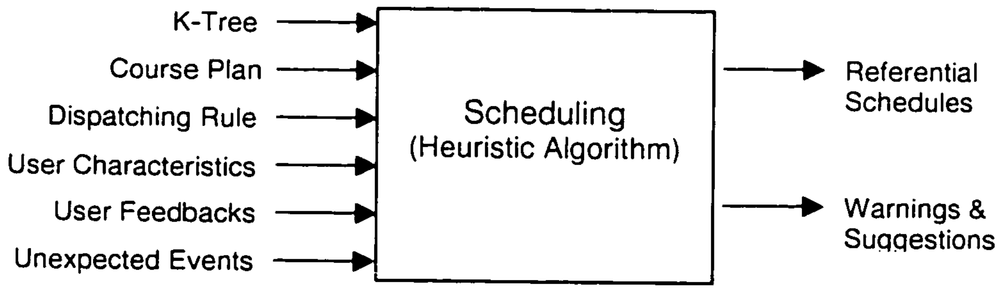


Figure 5.4: Scheduling Inputs and Outputs

Most of the inputs in Figure 5.4 are explained in the previous sections, such as K-Tree, course plan and dispatching rules. User characteristics, although another very important issue, are outside the scope of this thesis and are treated as a part of future research. User feedbacks and unexpected events are explained through the case study in the following section.

Regarding the outputs, we do not expect fixed step-by-step schedules from the scheduling process. Rather, referential schedules are more preferable. Basically, a referential schedule is both flexible and feasible. Being flexible means the schedule suggests various possible ways of working; being feasible means the schedule is always kept up to date and will prevent the student from violating the course plan to the best of its ability. Thus, a referential schedule should at least present the student with the following three categories of information:

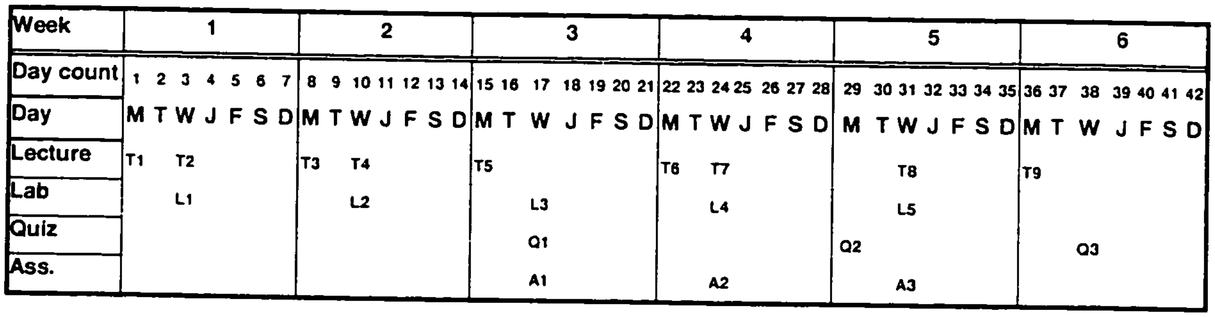
- **In the past**
    - Which activities are successfully completed?
    - Which activities are not completed?
    - What are the penalties that have been caused by delay or absence?
- **At present**
    - What's due today?
    - Which activities can be done now?
    - What are the critical degree and states of those activities?
- **In the future**
    - What's due tomorrow?
    - What will happen if I postpone an activity?

5.3.5 Scheduling: A Case Study

In this section, we examine scheduling manually for a specific case: student X taking the assembly language course. In this case, four inputs will be considered: (1) The K-Tree for assembly language illustrated in Chapter 5; (2) The dispatching rules introduced in Section 5.3.1; (3) A course plan which I create manually in the next section; (4) A special event which is introduced later. Several assumptions are also made here: (A) Student X takes only one course; (B) The K-Tree is fixed during the scheduling process; (C) User characteristics are excluded from this case study. In short, I try to keep the case simple because it aims at showing what will be a typical scheduling process.

5.3.5.1 A Concrete Course Plan as Input

Suppose the instructor has decided that the Assembly Language course would involve the activities as illustrated in Figure 5.5 (timeline) and Table 5.7 (activity properties). A logical *day count* is added to each day because using it to represent a deadline is much more straightforward than using physical dates.



Week	1							2							3						
Day count	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Day	M	T	W	J	F	S	D	M	T	W	J	F	S	D	M	T	W	J	F	S	D
Lecture	T1		T2					T3		T4					T5						
Lab			L1							L2							L3				
Quiz																	Q1				
Ass.																	A1				

Week	4							5							6						
Day count	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
Day	M	T	W	J	F	S	D	M	T	W	J	F	S	D	M	T	W	J	F	S	D
Lecture	T6		T7							T8					T9						
Lab			L4							L5											
Quiz								Q2									Q3				
Ass.			A2							A3											

Figure 5.5: Instructor's Plan in Timeline

Activity	Deadline	Knowledge Content	Count for Evaluation	Late Penalty
Lecture1	Day 1	M1		
Lecture 2	Day 3	M2		
Lecture 3	Day 8	M3		
Lecture 4	Day 10	M4		
Lecture 5	Day 15	M5		
Lecture 6	Day 22	M6		
Lecture 7	Day 24	M7		
Lecture 8	Day 31	M8		
Lecture 9	Day 36	M1-M8		
Lab 1	Day 3			
Lab 2	Day 10			
Lab 3	Day 17			
Lab 4	Day 24			
Lab 5	Day 31			
Quiz 1	Day 17	M1-M3	20%	100%
Quiz 2	Day 29	M4-M6	20%	100%
Quiz 3	Day 38	M7-M8	20%	100%
Ass. 1	Day 17		10%	10%
Ass. 2	Day 24		15%	10%
Ass. 3	Day 31		15%	10%

Table 5.7: Properties of Activities in Instructor's Plan

Student X create his personal study plan based on the instructor's plan as illustrated in Figure 5.6 (timeline) and Table 5.8 (activity properties). Figure 5.6 differs from Figure

5.5 in that it not only shows the deadline of each activity, but also suggests a time span – the gray areas – to prepare or review the activity. Similarly, although Table 5.8 inherits a lot of information from Table 5.7, it introduces two more columns: Time Span (days allocated for preparing or reviewing an activity if applicable) and Workload (number of work units for each activity if applicable). Those two properties can only be defined in student's plan because they largely depend on the student's learning preference and personal characteristics.

Week	1							2							3							4							5							6						
Day count	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
Day	M	T	W	J	F	S	D	M	T	W	J	F	S	D	M	T	W	J	F	S	D	M	T	W	J	F	S	D	M	T	W	J	F	S	D	M	T	W	J	F	S	D
Lecture1	T1			4																																						
Lecture 2			T2			4																																				
Lecture 3								T3			4																															
Lecture 4										T4			4																													
Lecture 5															T5			4																								
Lecture 6																						T6			5																	
Lecture 7																								T7			5															
Lecture 8																															T8			4								
Lecture 9																																				T9						
Lab 1	2		L1																																							
Lab 2								4		L2																																
Lab 3															4		L3																									
Lab 4																						4		L4																		
Lab 5																													4		L5											
Quiz 1													7				Q1																									
Quiz 2																									7				Q2													
Quiz 3																																		7				Q3				
Ass. 1													7				A1																									
Ass. 2																				7				A2																		
Ass. 3																											7				A3											

Figure 5.6: Student X's Plan in Timeline View

Activity	Deadline	Knowledge Content	Time Span (review/preparing)	Workload	Count for Evaluation	Late Penalty
Lecture1	Day 1	M1	Day 2-5 (4)	3		
Lecture 2	Day 3	M2	Day 4-7 (4)	3		
Lecture 3	Day 8	M3	Day 9-12 (4)	3		
Lecture 4	Day 10	M4	Day 11-14 (4)	3		
Lecture 5	Day 15	M5	Day 16-19 (4)	3		
Lecture 6	Day 22	M6	Day 23-27 (5)	4		
Lecture 7	Day 24	M7	Day 25-29 (5)	4		
Lecture 8	Day 31	M8	Day 32-35 (4)	3		
Lecture 9	Day 36	M1-M8				
Lab 1	Day 3		Day 1-2 (2)	1		
Lab 2	Day 10		Day 6-9 (4)	2		
Lab 3	Day 17		Day 13-16 (4)	2		
Lab 4	Day 24		Day 20-23 (4)	2		
Lab 5	Day 31		Day 27-30 (4)	3		
Quiz 1	Day 17	M1-M3	Day 10-16 (7)	5	20%	100%
Quiz 2	Day 29	M4-M6	Day 22-28 (7)	5	20%	100%
Quiz 3	Day 38	M7-M8	Day 31-38 (7)	5	20%	100%
Ass. 1	Day 17		Day 10-16 (7)	6	10%	10%
Ass. 2	Day 24		Day 17-23 (7)	6	15%	10%
Ass. 3	Day 31		Day 24-30 (7)	6	15%	10%

Table 5.8: Properties of Activities in Student X's Plan

5.3.5.2 Outputs

Suppose today is day 10. Based on Figure 5.7, we know that from day 9 to day 10, the situations have changed a lot for student X. 1) It left the time span for preparing "Lab 2". 2) Two activities – reviewing "T4" and preparing "L2" enter into the last day within their time span. 3) Another two activities – preparing "Quiz 1" and writing "Assignment 1" just enter into their time span. 4) "Lecture 3" shifts from the first day to the second within its time span for review. Let's presume that all the activities on the left of day 10 have been completed successfully, and on day 9 student X didn't review "Lecture 3" at all. The rescheduling process will exclude any irrelevant activity and only consider the activities related to the current day, that is those due today and those having their time span overlapped with today. Based on the above consumption, after rescheduling on day 10, the timeline would be like in Figure 5.7:

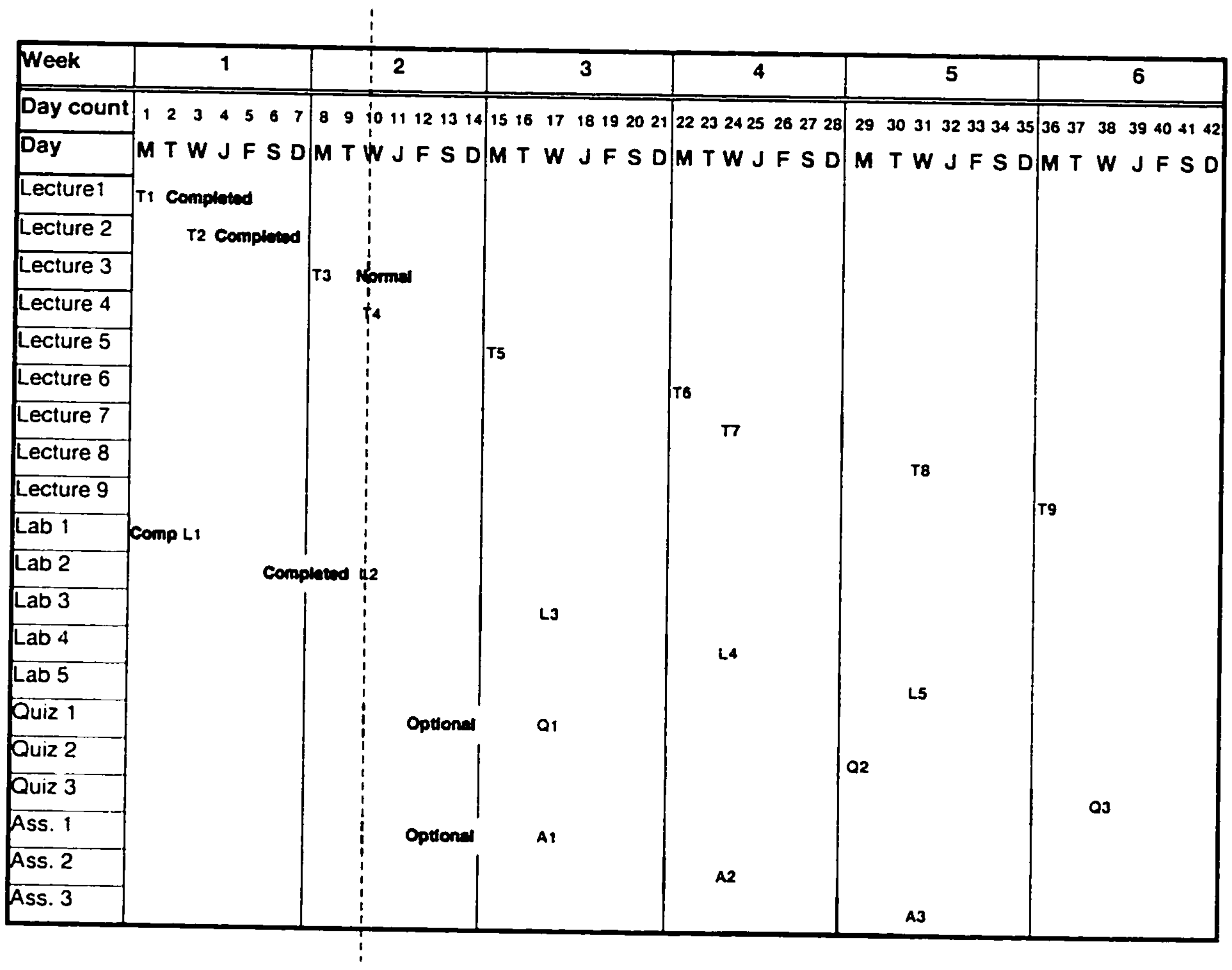


Figure 5.7: Student X's Timeline on Day 10

Visually, the timeline gives student X the following information:

```
Today is day 10 (Wed. Week 2)
________________________________________________

You have no uncompleted activities in the past
The accumulated penalty is 0
........................................................................
You have following activities due today
        1.  Attend Lecture 4
        2.  Attend Lab 2
You have no critical activity to do today
You have 1 normal activity to do today
        1.  Review Lecture 3
            -   Current workload: 3 unit
            -   Current time span: 3 days
            -   Current critical degree: 1
You have 2 optional activities to choose from:
        2.  Write Assignment 1
            -   Current workload: 6 units
            -   Current time span: 7days
            -   Current critical degree: 0.86
        3.  Prepare Quiz 1
            -   Current workload: 5 units
            -   Current time span: 7 days
            -   Current critical degree: 0.71
........................................................................
You have nothing due tomorrow
```

Figure 5.8: Scheduling Outputs on Day 10

Assume that an unexpected event occurred, namely student X was sick and didn't do any thing from day 10 to day 15. Thus On day 16, the timeline would look like that in Figure 5.9.

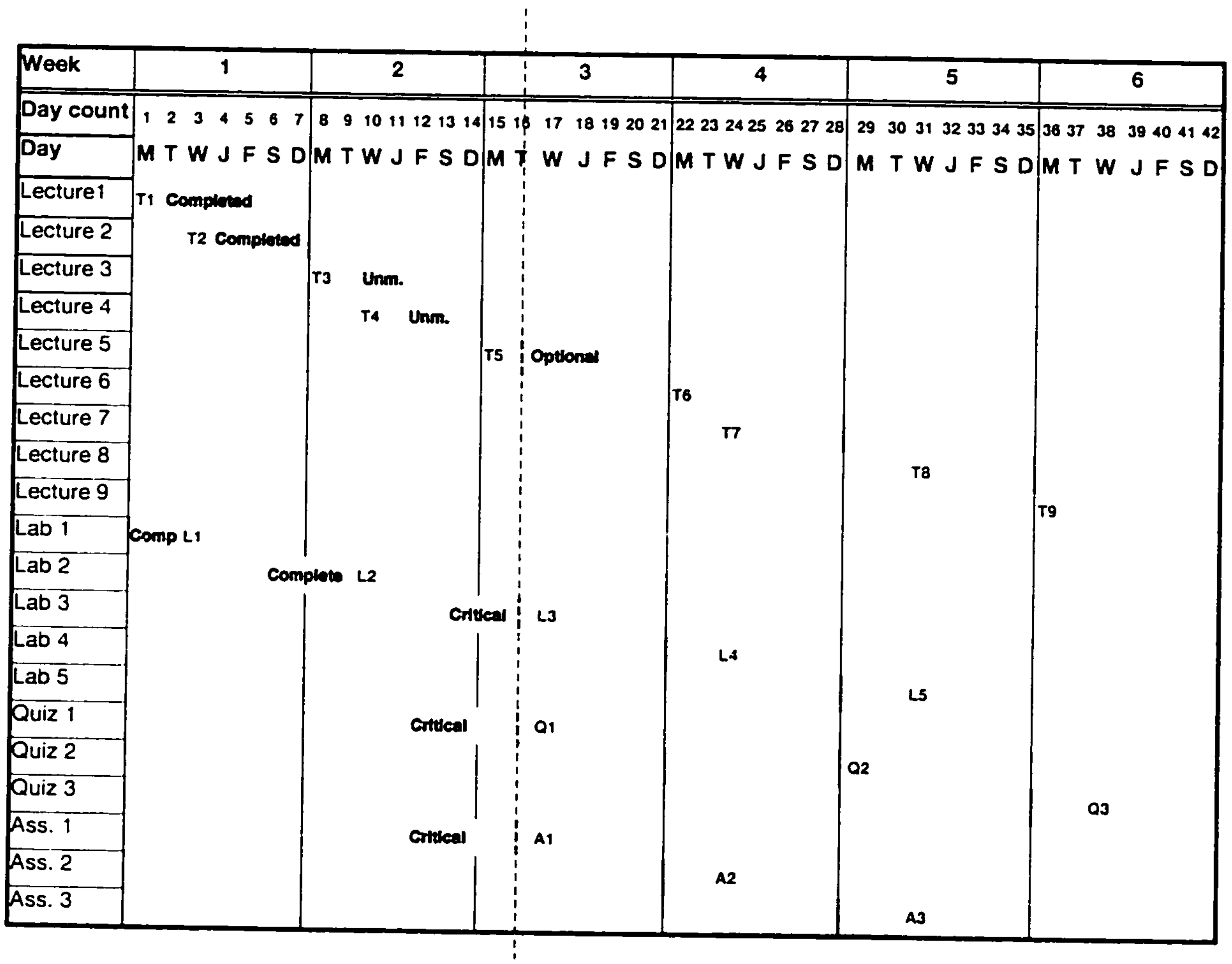
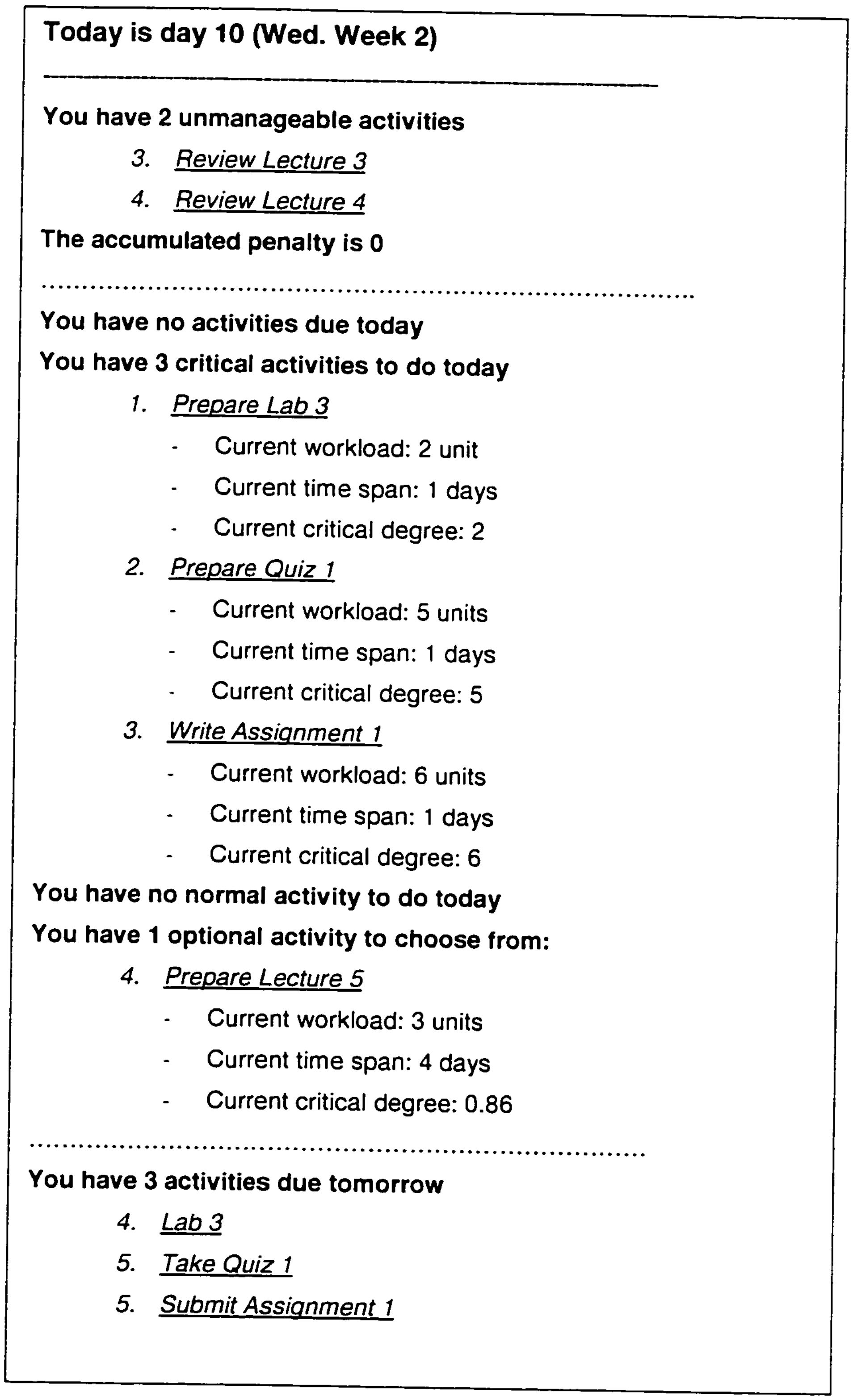


Figure 5.9: Student X's Timeline on Day 16

Apparently the current situation has changed considerably and becomes very critical because of the student's absence, as summarized in Figure 5.10:



```
Today is day 10 (Wed. Week 2)
________________________________________________

You have 2 unmanageable activities
        3.  Review Lecture 3
        4.  Review Lecture 4
The accumulated penalty is 0
.........................................................................
You have no activities due today
You have 3 critical activities to do today
        1.  Prepare Lab 3
              -   Current workload: 2 unit
              -   Current time span: 1 days
              -   Current critical degree: 2
        2.  Prepare Quiz 1
              -   Current workload: 5 units
              -   Current time span: 1 days
              -   Current critical degree: 5
        3.  Write Assignment 1
              -   Current workload: 6 units
              -   Current time span: 1 days
              -   Current critical degree: 6
You have no normal activity to do today
You have 1 optional activity to choose from:
        4.  Prepare Lecture 5
              -   Current workload: 3 units
              -   Current time span: 4 days
              -   Current critical degree: 0.86
.........................................................................
You have 3 activities due tomorrow
        4.  Lab 3
        5.  Take Quiz 1
        5.  Submit Assignment 1
```

Figure 5.10: Scheduling Outputs on Day 16

Students may find it is easy to do planning and scheduling simply within their own memory space, or at most using some paper and pens, if they only register for one course a semester. However, the extra work gradually becomes overwhelming and error-prone when they take 3 or 4 courses simultaneously, which are quiet often the real world cases, and computer based planning and scheduling tools become handy. Although this chapter deals with course planning and scheduling on one course base, the proposed rules and methodologies fit properly to multi-courses.

6. Implementation and Evaluation

With regard to implementation, two modules were implemented to test the concepts proposed. One is the *Online Course Presentation System*, which deploys the idea of knowledge tree as an encapsulation of various knowledge forms to be presented to the student. The other one is the *Online Course Planning System*, which deploys the idea of dynamic planning and scheduling as a way to support the student's learning process.

6.1 Online Course Presentation System

6.1.1 System Architecture

The Online Course Presentation System is a Web-based client/server application. Programs run on both the client and server computers. Figure 6.1 shows the system architecture of the Online Course Presentation System.

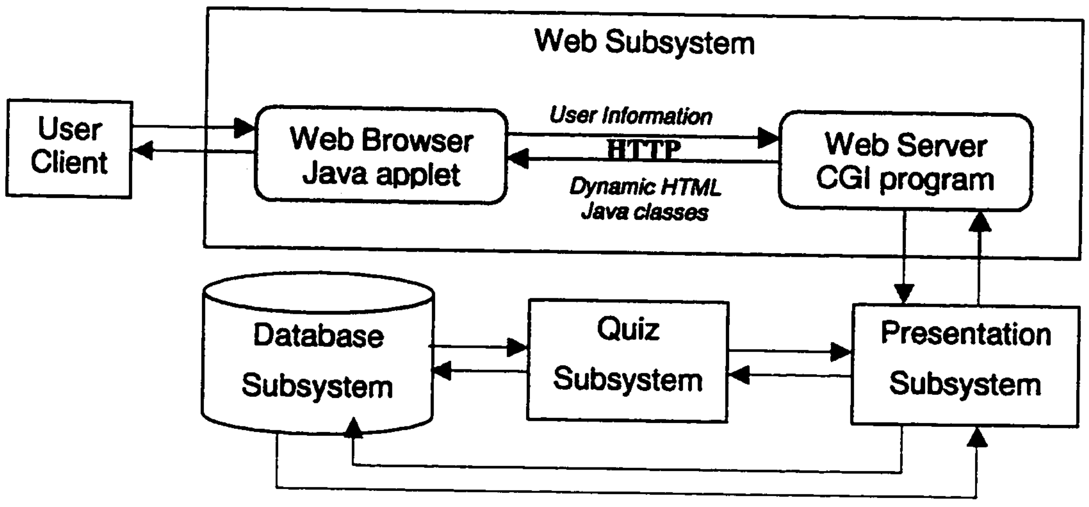


Figure 6.1: Online Course Presentation System Architecture

- **Web Subsystem**

This subsystem consists of four parts: Web Browser (client), Web Server, HTML page and CGI program. The user launches a Web browser on his computer to connect to the Web server through the Internet/Intranet. The HTML page will gather the user information or request and send them as the parameters of a CGI program which runs on the Web server. The CGI program will generate dynamic HTML web pages according to the user's inputs. Within these HTML pages, the Java classes will be loaded from the Web server to the browser and start running on the client side.

- **Presentation Subsystem**

The presentation Subsystem is the most important part of the Online Course Presentation System. It controls all the GUI components and the links between them and the database. It also controls the UI to show the quiz information. The Presentation Subsystem takes full advantage of Java's features to integrate UI, database and multimedia technologies. This subsystem has three parts: knowledge tree, articles/slides show and audio playing. The knowledge tree is a three-level tree – course level, module level and topic level – and serves as a table of contents for the course. Different levels of nodes being selected will cause different knowledge content to be presented. For example, if a module level node is selected, a tutorial consisting of a slide show and audio will be presented. On the other hand, if a topic level node is selected, only a text-based page (in HTML) will be presented.

- **Quiz Subsystem**

The Quiz Subsystem is used as a self-evaluation method to check the effectiveness of self-study. The quiz is designed as multiple-choice questions. The user has to select the only correct answer from four choices. The user can choose between getting an *instant response* (getting the feedback immediately after answering a question) or postponed response (getting the feedback after finishing the whole quiz). The provided feedback includes the correctness of the answer, and the links to the related knowledge in the K-tree, specific to that question.

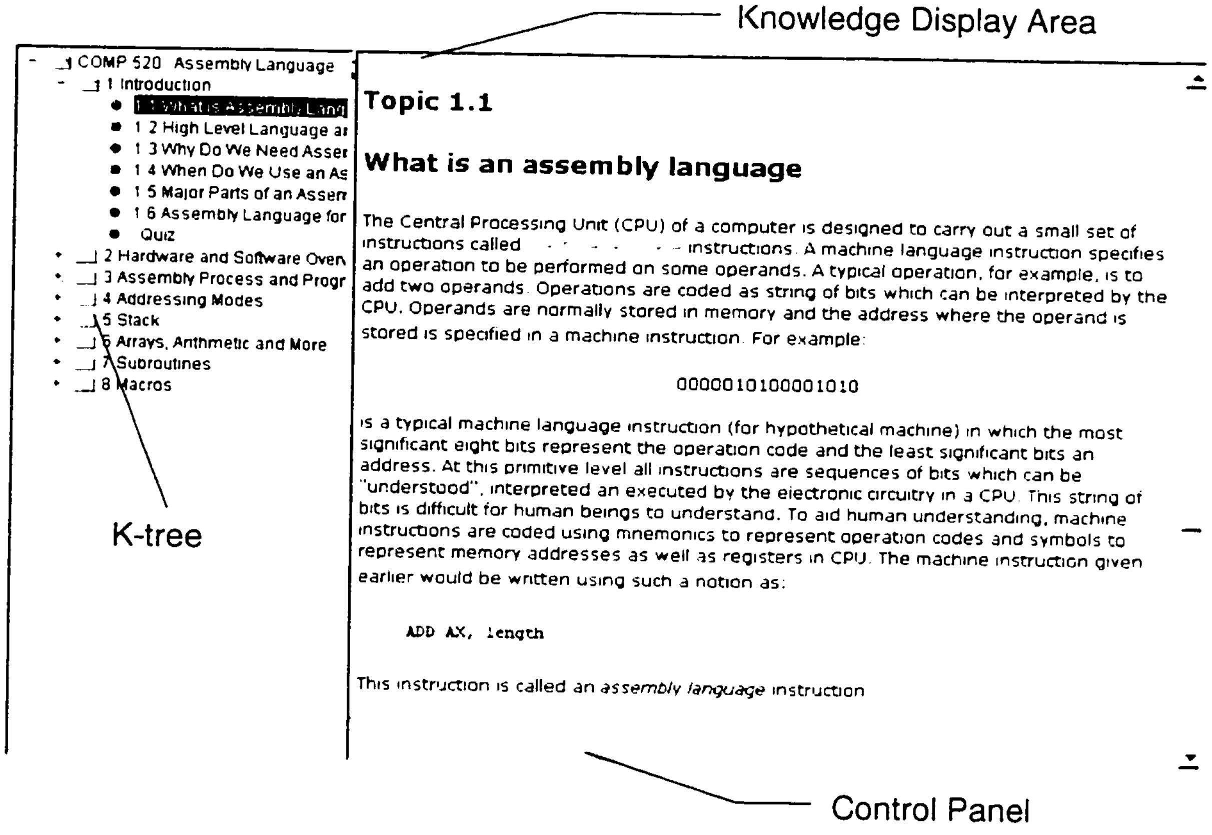
- **Database Subsystem**

Oracle 8i was chosen to implement this subsystem not only because Oracle is popular, but also because Oracle 8i supports multimedia data types. JDBC is used to connect to the DBMS. The database subsystem has *Course, Student, KTree* tables which save course information, student information and the knowledge tree respectively.

6.1.2 Graphic User Interface (GUI) Design

The UI design of the Online Course Presentation System follows the real world metaphor. The knowledge tree can be regarded as a book which is organized into chapters (modules) and topics; there is a slides/audio tutorial at the beginning of every chapter and there is also a quiz at the end of every chapter. So, to learn the course, the student may begin with listening to the tutorial, then read the articles, try out the quiz and review the knowledge he/she still isn't familiar with.

As you can see in Figure 6.2, the GUI of the Online Course Presentation System consists of three parts: the knowledge tree, the knowledge display area and the control panel.



- COMP 520 Assembly Language
  - 1 Introduction
    - 1 1 What is Assembly Lang
    - 1 2 High Level Language ar
    - 1 3 Why Do We Need Asser
    - 1 4 When Do We Use an As
    - 1 5 Major Parts of an Assen
    - 1 6 Assembly Language for
    - Quiz
  - 2 Hardware and Software Oven
  - 3 Assembly Process and Progr
  - 4 Addressing Modes
  - 5 Stack
  - 6 Arrays, Arithmetic and More
  - 7 Subroutines
  - 8 Macros

Topic 1.1

What is an assembly language

The Central Processing Unit (CPU) of a computer is designed to carry out a small set of instructions called [illegible] instructions. A machine language instruction specifies an operation to be performed on some operands. A typical operation, for example, is to add two operands. Operations are coded as string of bits which can be interpreted by the CPU. Operands are normally stored in memory and the address where the operand is stored is specified in a machine instruction. For example:

0000010100001010

is a typical machine language instruction (for hypothetical machine) in which the most significant eight bits represent the operation code and the least significant bits an address. At this primitive level all instructions are sequences of bits which can be "understood", interpreted an executed by the electronic circuitry in a CPU. This string of bits is difficult for human beings to understand. To aid human understanding, machine instructions are coded using mnemonics to represent operation codes and symbols to represent memory addresses as well as registers in CPU. The machine instruction given earlier would be written using such a notion as:

```
ADD AX, length
```

This instruction is called an *assembly language* instruction

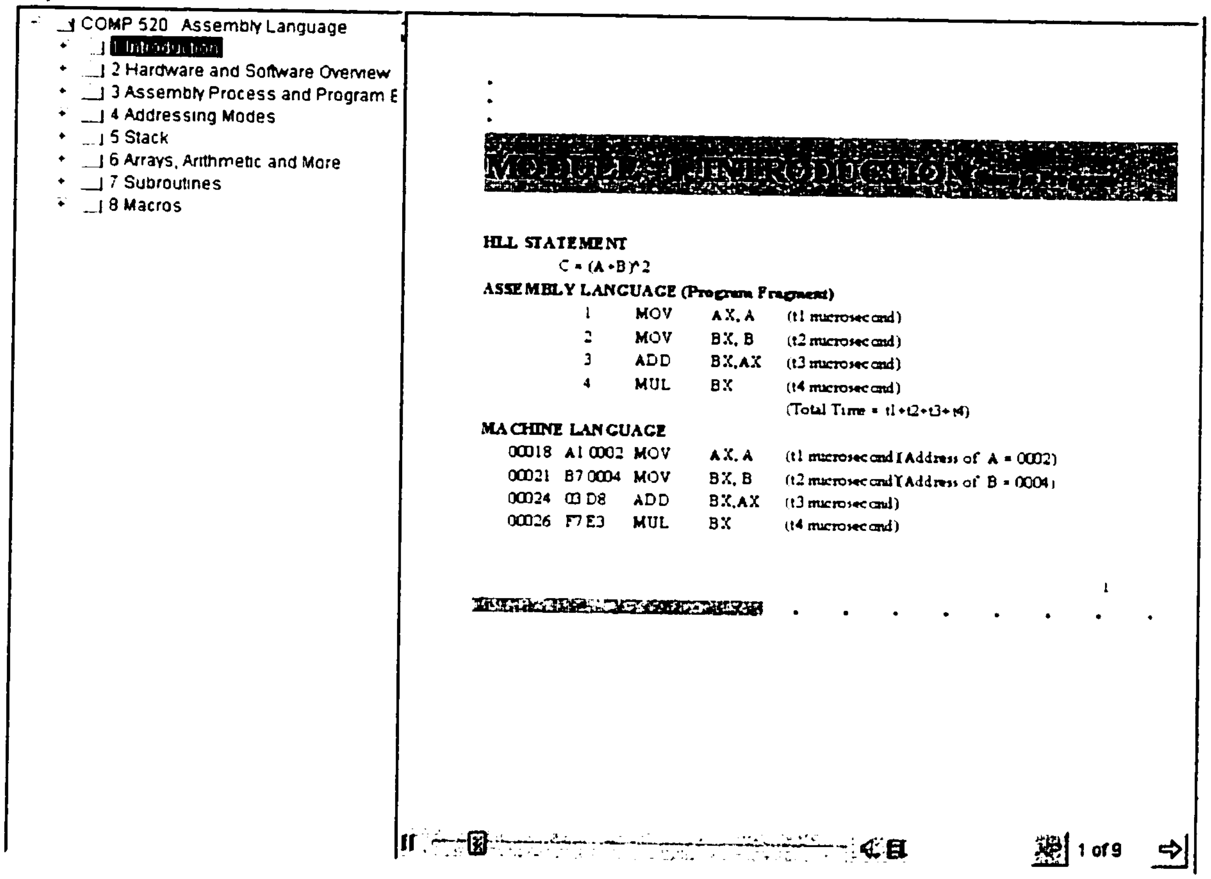
Figure 6.2: GUI for Online Course Presentation System – Article

- **Knowledge Tree**

The knowledge tree has a common tree's look and feel in the Windows system. The knowledge tree can be expanded and three-levels of a tree node will be shown: 1) the tree root represents the course; 2) the folders represent module level nodes; 3) the black nodes represent topic level nodes. Different levels of nodes being selected, the corresponding information will be shown in the knowledge display area.

- **Knowledge Display Area**

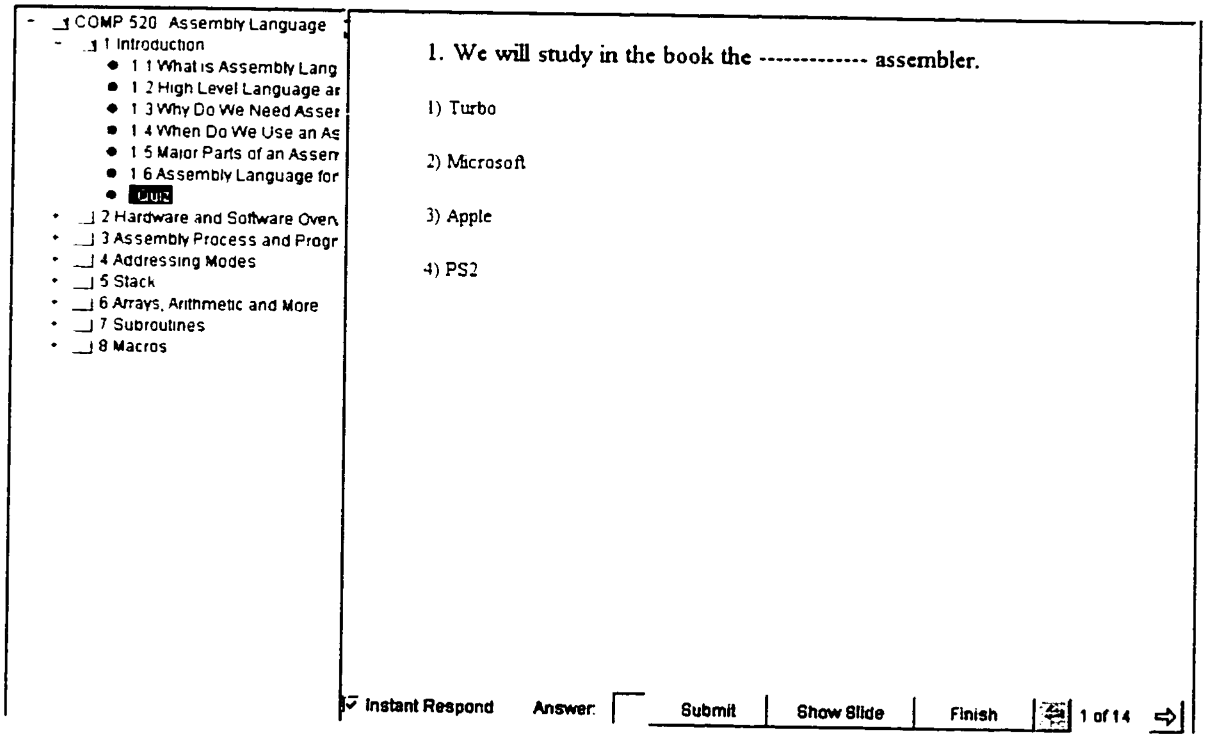
The knowledge display area is actually an HTTP window, and depending on which node is selected in the knowledge tree, it will display various knowledge contents. In Figure 6.2, the topic "What is assembly language?" is selected in the knowledge tree, and the HTTP window is displaying an article corresponding to this specific topic. If a module is selected in the knowledge tree, it will start the tutorial for that module. This time, as illustrated in Figure 6.3, the HTTP window is used to display slides.


COMP 520 Assembly Language
1 Introduction
2 Hardware and Software Overview
3 Assembly Process and Program E
4 Addressing Modes
5 Stack
6 Arrays, Arithmetic and More
7 Subroutines
8 Macros
MODULE 1 INTRODUCTION
1 of 9

```
HLL STATEMENT
        C = (A+B)^2
ASSEMBLY LANGUAGE (Program Fragment)
            1     MOV     AX, A     (t1 microsecond)
            2     MOV     BX, B     (t2 microsecond)
            3     ADD     BX,AX     (t3 microsecond)
            4     MUL     BX        (t4 microsecond)
                                    (Total Time = t1+t2+t3+t4)
MACHINE LANGUAGE
   00018  A1 0002  MOV    AX, A     (t1 microsecond)(Address of A = 0002)
   00021  B7 0004  MOV    BX, B     (t2 microsecond)(Address of B = 0004)
   00024  03 D8    ADD    BX,AX     (t3 microsecond)
   00026  F7 E3    MUL    BX        (t4 microsecond)
```

Figure 6.3: GUI for Online Course Presentation System – Tutorial

In the third case, when the node "Quiz" is selected in the knowledge tree, the HTTP window is used to display questions in the quiz. See Figure 6.4.



COMP 520 Assembly Language
1 Introduction
1 1 What is Assembly Lang
1 2 High Level Language ar
1 3 Why Do We Need Asser
1 4 When Do We Use an As
1 5 Major Parts of an Assem
1 6 Assembly Language for
Quiz
2 Hardware and Software Overv
3 Assembly Process and Progr
4 Addressing Modes
5 Stack
6 Arrays, Arithmetic and More
7 Subroutines
8 Macros
Instant Respond
Answer:
Submit
Show Slide
Finish
1 of 14

1. We will study in the book the ------------ assembler.

1) Turbo

2) Microsoft

3) Apple

4) PS2

Figure 6.4: GUI for Online Course Presentation System – Quiz

- **Control Panel**

The look and feel of the control panel at the bottom changes with the contents in the knowledge display area (the HTTP window). It is hidden when articles are displayed, (see Figure 6.2). When the HTTP window is displaying tutorial slides, the control panel appears to provide the transition between slides and the playback control for audio files, as shown in Figure 6.3. When the HTTP window is displaying quiz questions, the control panel changes its appearance to provide interaction between the user and the quiz subsystem, as shown in Figure 6.4.

The Online Course Presentation System is a Web based application that differs from most current course material delivering tools, since it takes most advantage of the Web and incorporates following features: (1) Knowledge tree is used to encapsulate and present course materials. (2) It lets the students to learn through interaction with the system. (3) Multimedia course materials are presented. (4) It immediately reflects any changed made to the knowledge tree by the instructor.

6.2 Online Course Planning System

6.2.1 System Architecture

The Online Course Presentation System is also a Web based client/server application. Figure 6.1 shows the system architecture of the Online Course Planning System.

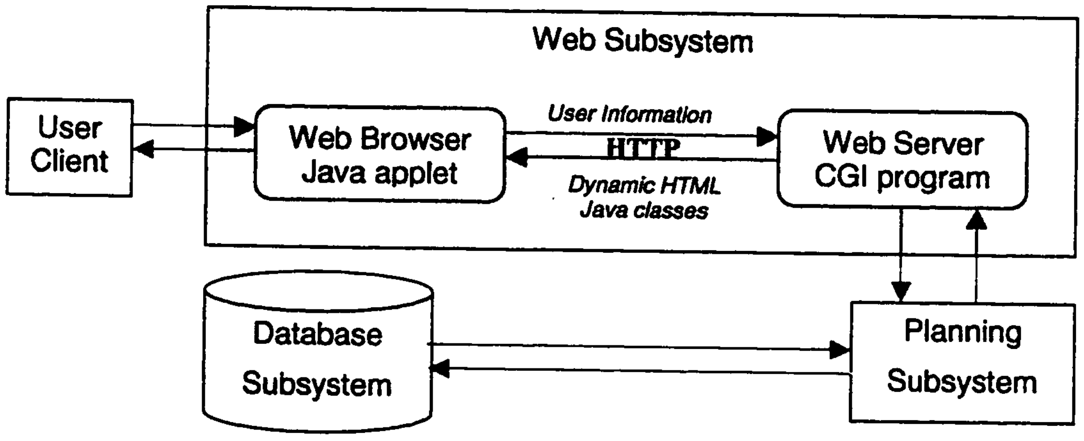


Figure 6.5: Online Course Planning System Architecture

The web subsystem in the Online Course Planning System works similar to the Online Course Presentation System, and will not be addressed repeatedly here. The Planning Subsystem is the core part of the Online Course Planning System. It controls all the GUI components and the links between them and database. The Planning Subsystem manages all aspects of learning activities, including modifying activity properties, monitoring user's progress, updating activity states, refreshing schedules, and bringing up the knowledge contents related to each activity. Noticeably here, the activity-knowledge relationship establishes a connection between the Planning Subsystem and the Course Presentation System. The Database Subsystem has two tables: *CoursePlan* contains the properties of activities at instructor's level, while *StudentPlan* contains the properties of activities at student's level.

6.2.2 Graphic User Interface (GUI) Design

The major part of the GUI for Online Course Planning System is a timeline as shown in Figure 6.6. Each column represents a certain day, and each row represents an activity in progress. To be more intuitive, instead of using a number-based critical degree, the GUI displays each activity in one of four colors, each representing one of four states: gray represents "finished"; red represents "unmanageable"; yellow represents "critical"; green represents "optional". One of the columns is orange and shows where "today" is. It keeps moving from the first day to the last day in the timeline while the time passes. The width of an activity tells the user its time span.

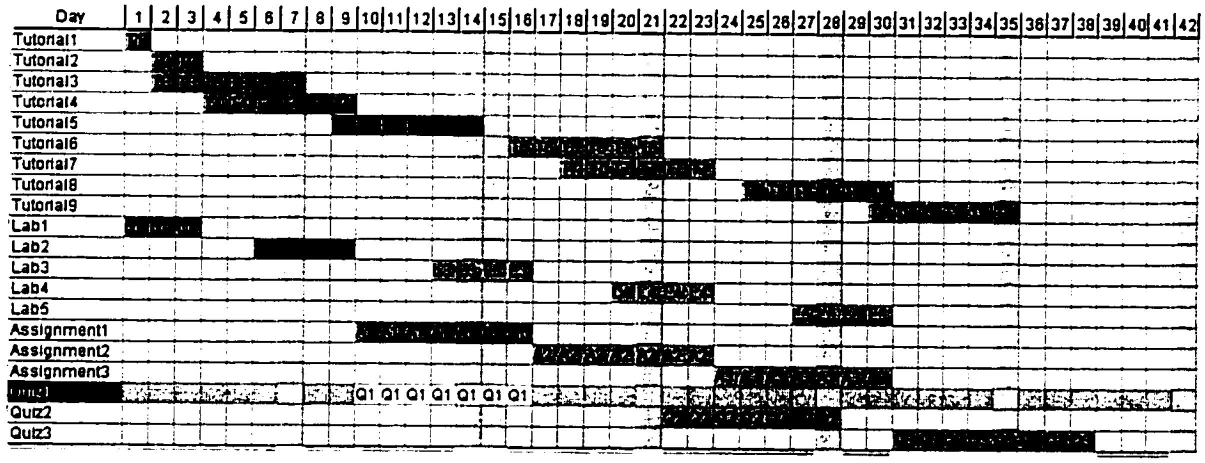
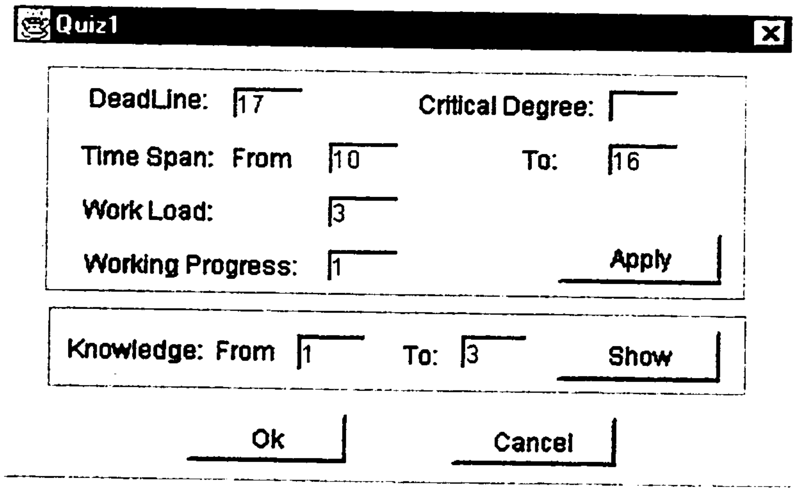


Figure 6.6: GUI for Online Course Planning System – Timeline

Clicking on an activity will bring up a window displaying the various properties of that activity, (see Figure 6.6). If the user is an instructor, he can modify the deadline, time span and work load. If the user is a student, he can only modify the progress work. Changing the progress of work may cause the critical degree to change color.

Furthermore, the "Show" button will lead the user to the knowledge tree node that relates to this specific activity.


Quiz1

DeadLine: 17　Critical Degree:

Time Span: From 10　To: 16

Work Load: 3

Working Progress: 1

Knowledge: From 1　To: 3

Figure 6.7: GUI for Online Course Planning System – Activity Property

The Online Course Planning System is a Web-based application that differs from most current course planning tools, since it takes most advantage of the Web and incorporates following features: (1) It automatically plans on an ongoing base for each student based on his/her progress, and always reflects the most current situation. (2) Student's plan immediately reflects any changes made to the course plan by the instructor. (3) Students are provided with maximum flexibility to learn.

6.3 Evaluation

Evaluation is an important part of work that has been done for this thesis, in order to prove that the research results are useful, and also to test the performance of the implemented systems. The following sections present the details of the evaluation plan, procedure, and results.

6.3.1 General Evaluation Plan

- Evaluation objectives
  - Summative evaluation, which takes place after implementation, mainly to test the proper functioning of the final system.
  - Improve the interface, find the good/bad parts.
  - Identify potential problem areas and suggest possible solutions for future work.

- Evaluation methods
  - Before the test, use a questionnaire to investigate the user characteristics;
  - During the test, perform laboratory studies with user participation; use direct observation to measure user performance; use thinking-aloud protocol to get empirical data.
  - After the test, interview users to get supplementary data, their opinions and suggestions.

- User profiles
  - 10 students in computer science or computer engineering;
  - Have taken Assembly Language course;
  - At least two female students and minimum 4 male students;
  - At least 3 Chinese students;

  - Use questionnaire for more detailed user characteristics (name, age, nationality, computer knowledge, experience with Web-based learning software, attitudes, etc).

- Setup and run the test
  - Prepare testing documents, including a pre-test questionnaire (see Appendix I), testing scenarios (see Appendix II), observation log, and a set of questions for the after-test interviews (see Appendix III);
  - Ask the user to fill out the questionnaire;
  - Simply present the system to the user;
  - Present the testing methods and rules to the user;
  - Ask the user to follow the scenarios and complete the required tasks (see Appendix II); fill out the observation log simultaneously;
  - Interview the user after the test;
  - Repeat the test for other users.

- Evaluation environment and apparatus
  - In the computer lab, communication with tester allowed;
  - Windows NT system, keyboard and mouse.

6.3.2 Evaluation Results

Through direct observation during the test, I found the final system functions properly. All the users can accomplish the required tasks within an hour. The users seem to understand the system very well after listening to a simple 5-minute introduction of the system. The error rate is lower than 3%. Through the interviews after the test, I found out that the users' general opinion towards the system is "very satisfied". On average, the users gave the system following evaluations:

- Easy of understanding: 85%
- Easy of use: 82%

- Usefulness: 90%
- Satisfaction: 82%

While comparing with other similar WBL tools, some users said that the knowledge tree structure is commonly seen in other systems. One user pointed out that the Quiz part is similar to a Quiz tool he used in his math class. But most of the users like the way multimedia knowledge is integrated into the Online Knowledge Presentation System, and found that the knowledge connections established between the two systems are very useful. On the other hand, most of the users said the Online Course Planning System was refreshing and helpful, especially when they have to handle several courses at the same time, and they had never used a similar system before.

As for the GUI, the users think the knowledge tree and the planning timeline are visually self-explanatory. They also found it is easy to interact with Quizzes and Tutorials. Although no serious GUI problem was found during the test, the observation log did track down many possible ways to further improve the GUI and to meet more of the users' expectation. Here are some examples:

- In the activity property pop-up window, most of the properties are represented as numbers. This is not very intuitive and would better be substituted with words familiar to users or proper visual components.
- Currently the GUI provides only linear access to slides and quiz questions. The users would prefer direct access.
- Currently, the critical degree is represented by colors. It should also be represented with numbers so that two activities can be compared even if they have the same color.
- Currently, the planning timeline displays the whole semester. This is overwhelming for some users. It is better to provide a zoom facility to focus on a certain week.

Besides improving the GUI from various detailed aspects, the test and the after-test interviews also discovered many other functional expectations from the user. Here are some major ones:

- Add more media contents and more interactive features;

- Provide more quizzes and allow a user to restart quizzes;
- Count penalty and grades for users;
- Compare progress with other users;
- Provide basic communication tools with the instructor.

The major dissatisfaction I received from the users towards the system is the problem of speed. This is partially due to the network configuration, partially due to the software environment – the system was developed using Java 2 and was running with Netscape 6, both of which cause slow response.

7. Conclusion and Future Work

Through systematic analysis of the domain of Web Based Learning (WBL), it is concluded that two classes of Knowledge Management (KM) support can be provided to WBL: Content-Centered Knowledge Management (CCKM) and Process-Centered Knowledge Management (PCKM). CCKM emphasizes the knowledge structure, and mainly concerns how subject knowledge is acquired, represented, organized and reused. PCKM emphasizes the knowledge flow control. It is concerned that the learning processes, human factors, and how to assess, change and improve individual learning effectiveness. Comparatively, CCKM is more general while PCKM is more domain-specific. Thus, to provide CCKM support to WBL, current research in KM can be applied, such as ontologies and knowledge components. PCKM can support WBL, by examining the domain specific and user specific factors. Planning and scheduling are typical PCKM problems, considered in this thesis.

Specific KM strategies can improve learning effectiveness. Subject Knowledge Modeling provides better representation and organization of the subject knowledge, by deploying the idea of knowledge tree as an encapsulation of various knowledge concepts and media to be presented to the student. Dynamic Planning-cum-Scheduling effectively supports user's learning by providing flexible and feasible schedules, early warnings and suggestions.

Two subsystems of WBL, namely Online Course Presentation System and Online Course Planning System are implemented and evaluated with a small set of 10 students. These early results indicate that the KM support is easy to use and is helpful. The evaluation needs more usability test.

The future work can be divided into two parts: short-term future work and long-term future work. The short-term future work is to improve and further develop the two systems already implemented. By analyzing the evaluation results, it is concluded in

Section 6.3.2 many ways to improve the system both from the GUI side and the functional side. WBL is such a broad field that besides subject knowledge modeling and course planning, a lot more places may also be a potential KM supported area. So the long-term future work is to explore KM strategies that are suitable for many other aspects of WBL. One possibility is to personalize the learning support, and the main knowledge involved here would be the user model and user characteristics, student records and progress, suggestions and tips, etc.

Appendix I Questionnaire

'Who are you?' – Individual user characteristics

Date ____________________ Time ________________

Name __________________ Tel No ________________

Physical Characteristics

Height __________

Weight _____________

Left-handed ☐ or right- handed ☐

Visual acuity	good ☐	normal ☐	bad ☐
Auditive acuity	good ☐	normal ☐	bad ☐

....

Psychological Characteristics

Cognitive style verbal ☐ analytic ☐ spatial ☐ intuitive ☐

Adventurous ☐ or timid ☐

Quick ☐ or slow ☐ learner

Good ☐ or bad ☐ memory

Motivation high ☐ moderate ☐ low☐

....

Socio-cultural Characteristics

Age __________

Gender M ☐ F ☐

Ethnic background ___________

First language ___________

Other languages ____________________

....

Prior Experience & Knowledge

Please indicate your educational background.

Have you taken the Assembly Language course? Y☐s ☐o

Are you familiar with the Windows system?

Have you use Web-CT or any other Web-based learning applications? If so, indicate their names.

Appendix II Testing scenarios

'What you do' – User's tasks to perform during test

Scenario 1 – Learn Subject Through the Knowledge Tree

- Before going to the next step, spend 5 minutes to explore the knowledge, and discover as much as you can;
- Start the tutorial by selecting one of the modules (the folder) from the tree;
- Listen to the tutorial;
- Skip any content you feel you are already familiar with;
- Try to disable the sound and enable it again;
- Finish the tutorial;
- Back to the tree, open the folder to see a list of topics;
- Browse through the topics;

Scenario 2 – Take a Quiz

- In the tree, select a module other than the one you've just learned;
- Open it, and at the end of the list of topics, find 'Quiz';
- Select 'Quiz' to start a Quiz;
- You see the first question in the right window;
- Spend 3 minutes to explore the interface, read how to take a Quiz before proceed to the next step;
- Switch to the Quiz of the module you've just learned;
- Make sure that at the bottom of the right window, 'Instant response' is not selected;
- Answer the question by typing input the corresponding number in the text box next to 'Answer', and click button 'Submit';
- Click the button 'Show Slide' to see the corresponding slide to the question;
- Click the button 'Back to Quiz' to return to the Quiz;
- Traverse back and forth to make any modification; always click 'Submit' after you modify an answer;
- After finishing the Quiz, click the button 'Finish' to see the result;
- Click the button 'Review' to review your answers.

Scenario 3 – Understanding Planning Process

- Listen to a short presentation of the planning module, to understand how the timeline works;
- View the plan of day 15;
- What's the situation now? What's the status of the activities in the timeline?
- What will happen next?
- What should you do?
- Select A1, a window will pop up showing the current properties of this activity;
- Make some changes to 'Working Progress', see how it change the color next to 'Critical Degree';
- Now presume you had a emergency and cannot get any work done until day 22;
- The tester will open the plan of day 22 for you;
- Look at the timeline, what's the situation now?
- What will you do next?
- If the instructor gave you an extension for A1, what do you think will happen to your time line?

Appendix III Interview Guideline

'What you feel and how you change' – User's opinion

User Name ____________________ Interview time ____________________

What's your general opinion toward the software? (Take 10 points as a full scale. High points represent "good", while low points represent "bad".)

• Easy of understanding	1	2	3	4	5	6	7	8	9	10
• Easy of use	1	2	3	4	5	6	7	8	9	10
• Usefulness	1	2	3	4	5	6	7	8	9	10
• Satisfaction	1	2	3	4	5	6	7	8	9	10

What's the major problem of the software in your opinion?

Does the software meet your expectation?

What do you expect more if the software will be further developed?

Is the software similar to other systems you familiar with? If so, is it comparable?

What's your attitude toward Web Based Learning (WBL)?

References

Abecker, A., A. Bernardi, K. Hinkelmann, O. Kuhn and M. Sintek (1998) 'Toward a Technology for Organizational Memories', *IEEE Intelligent System*, **13**, 3, pp 40-48.

Angus, J., J. Patel, and J. Harty (1998) 'Knowledge Management: Great Concept But What Is It?', *Information Week*, 673, March 16, pp 58-65.

Bhimavarapu, P. R. (1999) 'Component Based Knowledge Management Of Operating System Concepts', Project Report, New Jersey Institute Of Technology, Newark, NJ, 1999.

Borghoff, U. M., R. Pareschi (1998) Infomation Technology for Knowledge Management, Springer-Verlag, Berlin, Germany.

Brown, D. E., W. T. Scherer (1995) 'A Survey of Intelligent Scheduling Systems', *Intelligent Scheduling System*, pp1-40.

Chandrasekaran, B., J. R. Josephson and V. R. Benjamins (1999) 'What Are Ontologies, and Why Do We Need Them?', *IEEE Intelligent System*, pp20-26.

Eisenstadt, M. and T. Vincent (1998) 'Knowledge Media takes off on the Web: Introduction and Overview of the Book', in M. Eisenstadt and T. Vincent, *The Knowledge Web*, Knowledge Media Institute (KMi), ISBN 0-7494-2726-4, Kogan Rage, London, UK, pp 1-18.

Heijst, G., R. Spek, and E. Kruizinga (1998) 'The Lessons Learned Cycle', in U. M. Borghoff and R. Pareschi (1998) *Information Technology for Knowledge Management*, Springer-Verlag, Berlin, Germany, pp 17-33

Hinze, J. W. (1999) *Construction Planning and Scheduling*, Prentice Hall, Upper Saddle River, New Jersey.

Inglis, A., P. Ling and V. Joosten (1999) *Delivering digitally: managing the transition to the knowledge media*, Kogan Page, London.

The Knowledge Management Forum (1996) 'What is Knowledge Management?', available at http://www.km-forum.org/what_is.htm

The Knowledge Management Forum (1996) 'Knowledge vs. Information', available a http://www.km-forum.org/t000008.htm

Kuhn, O., A. Abecker (1998) 'Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Callenges', in Borghoff Uwe M. Pareschi, R. (1998) *Information Technology for Knowledge Management*, Springer-Verlag, Berlin, Germany, pp 183-205.

Kurfess, F. and L. Jololian (1999) 'Knowlets: Components for Knowledge Management', New Jersey Institute Of Technology, Newark, NJ.

LearningSpace homepage available at http://www.lotus.com/home.nsf/welcome/learnspace

Masterton, S. (1998) 'The Virtual Participant: A Tutor's Assistant for Electronic Conferencing', in M. isenstadt and T. incent, *The Knowledge Web*, Knowledge Media Institute (KMi), ISBN 0-7494-2726-4, Kogan Rage, London, UK, pp 249-266.

O'Leary, D. (1998) 'Knowledge Management Systems: Converting and Connecting', *IEEE Intelligent System*, **13**, 3, pp 30-33

Radhakrishnan, T. and V. Rajaraman (2000) *Essentials of Assembly Language Programming for The IBM PC*, Prentice-Hall of India Private Limited, New Delhi.

Rich, E. (1983) 'Users are Individuals: Individualizing User Models', *International Journal on Man-Machine Studies* 18, pp 199-214.

Rich, E. and K. Knight (1991) *Artificial Intelligence*, ISBN 0-07-052263-4, McGraw-Hill, New York, U.S.A, pp 330.

Shinghal, R. (1992) *Formal Concepts in Artificial Intelligence: Fundamentals*, ISBN 0-442-31444-2, Chapman & Hall, London, UK, pp 260.

Stutt, A. and E. Motta (1998) 'Knowledge Modeling: An Organic Technology for the Knowledge Age', in M. Eisenstadt and T. Vincent, *The Knowledge Web*, Knowledge Media Institute (KMi), ISBN 0-7494-2726-4, Kogan Rage, London, UK, pp 212-224.

Summer, T. and J. Taylor, (1998) 'Media Integration Through Meta-learning Environments', in M. Eisenstadt and T. Vincent, *The Knowledge Web*, Knowledge Media Institute (KMi), ISBN 0-7494-2726-4, Kogan Rage, London, UK, pp 63-78.

Valente, A. and j. A. Breuker (1996), 'Towards Principled Core Ontologies', in *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*.

WebCT homepage, available at http://www.webct.com