STUDENT MODELLING AND ADAPTIVITY IN WEB-BASED LEARNING SYSTEMS

A thesis submitted

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science

Massey University, New Zealand

Binglan Han

2001

Abstract

Web-based educational systems are now becoming part of main stream education. As an essential component of the web based educational systems, the student model enables the system to provide individualised course contents and study guidance, and therefore to help the students with different backgrounds and knowledge levels to achieve their learning goals effectively on the web.

A prototype student model was developed in this project for a web based learning system. The architecture of student model is divided in two parts: individual and group student models. The information contained in individual student model includes the student knowledge levels for course contents, study goals, learning styles, preferences, etc. The individual student model is initialised by asking students their behavioural preferences through a questionnaire, and using default information based on stereotyping in the group student model. The model is updated dynamically according to student study times and/or assessment results. The group student model is extracted by a cumulative analysis of the individual student models of various students and is used for giving guidance to the students. Both navigation and content adaptations are provided based on the information maintained in student models.

A web-based educational system was constructed for implementing and testing the student model. The web-based system adopted a three-tier, client-server architecture. The first tier is a set of HTML frames embedded with Java Applets running in the student's web browser to provide course contents and navigation guides. The middle tier consists of Java Servlets, JSP, and application programs to receive student requests, update student model, and send adaptive course contents and navigation guidance information to the client side. The course contents are stored in XML files that are processed to create the individualised course content presentations. The third tier is the relational database for storing the course structures and contents, and the information in the student model.

This study produced a unique two-fold web-based student modelling system that can be applied to intelligently deliver the courses for a wide range of subject domains.

ACKNOWLEDGEMENTS

I would like to thank my chief supervisor, Dr Kinshuk, Associate Professor, Department of Information Systems, College of Business, Massey University. Thank you greatly for your guidance, advice, and inspiring wisdom throughout this enjoyable programme.

I would like to thank my supervisor, Professor Chris Jesshope, Institute of Information Sciences and Technology, College of Sciences, Massey University. Thank you for your advice and help during the course of this project.

I would also like to express my appreciation to Dr Jenny Zhang, Research Officer, NZEDSoft, Massey University, for her assistance and advice during the course of this project.

Finally, I would like to thank my husband and parents for their support and encouragement, without which this project would not have been completed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter1 Introduction

1.1 Introduction

This chapter first describes the basic concepts and advantages of web-based intelligent educational systems and then states the importance of adaptation and student model in these systems. Finally it introduces the design approach adopted in the project and outline of the rest of the thesis.

1.2 Web-based Intelligent Educational System

Intelligent educational systems are computer programs that teach learners in intelligent manner. The techniques from artificial intelligent community are usually used to adapt the teaching processes to the needs of individual students.

Web-based intelligent educational systems (WIES) are those intelligent educational systems that can be accessed through the World Wide Web. A WIES makes an originally stand-alone educational system available to thousands of students all over the world via internet-connected computers.

A *web-based intelligent educational system* helps a wide range of students to achieve their learning goals effectively by delivering knowledge in an adaptive or individualised style through the web.

1.3 Advantages of Web-based Intelligent Educational Systems

The web-based intelligent educational systems inherit modern technologies of both artificial intelligent and web-based systems, so they have distinct advantages:

- *Web-based Intelligent Educational Systems are classroom independent.* The WIES makes the same educational process not only happen in one classroom once, but also around the world over and over again.

- *Web-based Intelligent Educational Systems are cross-platform deliverable.* Any platform that can utilise other web resources can share the materials used in the educational systems.

- *Web-based Intelligent Educational Systems can provide individualised adaptation.* Flexible individualised learning and teaching environment can be established according to the knowledge of the subject domain, information of the individual student or the groups of students, and teaching methodology.

- *Web-based Intelligent Educational Systems can provide adaptive hypermedia.* Learning materials can be dynamically produced and organised using web-based hypermedia and multimedia technologies.

- *Web-based Intelligent Educational Systems can keep centralised maintenance.* The systems need to be maintained only at server side, and the learning materials can be posted to the clients all over the world.

- *Web-based Intelligent Educational Systems are cost effective.* Once the teaching materials have been produced, they can be reused by any number of users.

1.4 Importance of Adaptation in Web-based Intelligent Educational Systems

The intelligence of an intelligent educational system is largely attributed to its ability to adapt to a specific student during the teaching process. As explained below, adaptation is more important for web-based educational systems than other standalone educational systems.

- *The system will be used by a much wider variety of students.* Since students may have very different backgrounds, learning styles, individual preferences, and knowledge levels, a system designed for any particular kind of students is not always suitable for others.

- *The knowledge of a particular student can grow very quickly.* The content page, which is quite complex for a student at the beginning, may soon become quite trivial and boring to the same student. The level and form of knowledge presentation may therefore need to be changed with the student learning process.

- *The system will be used by the students in different places around the world where no teacher is available for face-to-face assistance in most cases.* The system should provide individualised help just as a human tutor would.

- *The systems are generally more appropriate for adopting student centred learning model instead of teacher centred learning model.* The learning materials are presented according to the student's preference and knowledge level, which makes the study processes more interesting and effective.

1.5 Importance of Student Model in Web-based Intelligent Educational Systems

In general, the adaptation process can be described by three stages: getting the information about the student, processing the information to initialise and update a student model, and using the student model to provide the adaptation.

The student model is an essential component in the web-based intelligent education system. The adaptation of a web-based intelligent educational system mainly involves choosing and presenting each successive teaching activity as a function of entire scope of student's knowledge of the subject being taught and other relevant features of the student, which are in turn maintained in a *student model.* Therefore the student model is used to modify the interaction between the system and student to suit the needs of individual student.

1.6 Objective of the Project

This project is a part of the Technology Integrated Learning Environment (TILE) project, which is a web-based learning system being developed in Massey University, New Zealand. The main objectives of the project are as follows:

- *Design and implement a prototype student model used in Web-based Domain-independent Educational Environment.* One of the limitations of most existing adaptive web-based educational systems is that they are course specific. Since the TILE project is a general-purpose tool used to generate and deliver web-based courses for a wide range of subject domains, the techniques for modeling student learning behaviors in certain subjects may not applicable. Therefore, the prototype student model is intended for capturing the characteristics of general learning processes.

- *Design and implement a prototype web-based educational environment to test and evaluate the prototype student model.* According to requirements of the TILE system, the student model should be able to provide adaptive features in various scenarios including:
  - offline with local sever – in this mode, educational materials are provided from a local server in student's computer
  - online without local server – in this mode, educational materials are provided from a remote server
  - online with local server – in this mode, most educational materials are provided from a local server in student's computer, but student still has access to the remote server and local information can be updated from the remote server

  For this purpose, a client-server approach will be adopted for the system. To take advantage of the inferences of server side model (extracted by a cumulative analysis of the individual student models) the overall student model module includes both individual and group student models.

1.7 Research Approach

The following steps summarise the research approach used to satisfy the objectives described above:

- Literature review on the research articles and the existing web-based intelligent educational systems employing student models.

- Design the system architecture, student model, and database based on the literature review.

- Implement the prototype student model and associated prototype web-based educational system.

1.8 Outline of the Thesis

This thesis describes the whole approach of this project. It is divided into three sections:

- The first section presents the existing research in the field, which include the two chapters -- introduction and literature review.

- The second section talks about the system and student model design and implementation, which include three chapters -- system architecture, database, and student model design and implementation.

- The third section is the project results, which include two chapters -- student model adaptation and the student model evaluation and the conclusion.

This thesis outline is shown in Figure 1-1.

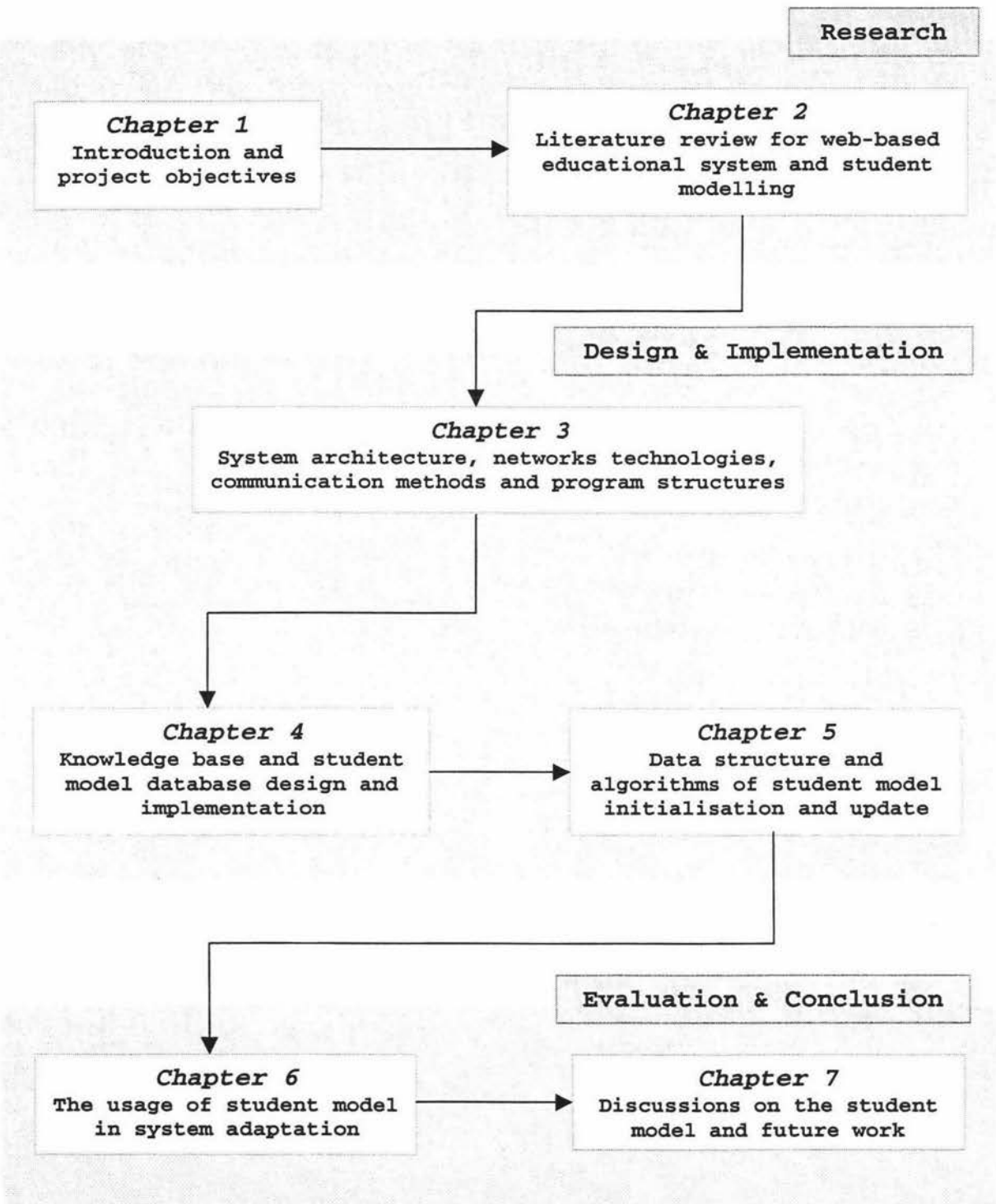

Figure 1-1 Outline of the thesis

Chapter 2 Literature Review

2.1 Introduction

This chapter first introduces the architectures of standalone and web-based intelligent educational systems, and then discusses the information contained in the web-based intelligent educational systems, mainly in the domain model and student model. The next section reviews the student modelling processes namely model initialisation and update. Finally the last two sections investigate the usage of student model, i.e. achieving individualised adaptation based on the data maintained in the student model.

2.2 Architecture of Intelligent Educational Systems

2.2.1 Standalone Intelligent Educational Systems

To achieve effective teaching results, a good human teacher must possess three types of knowledge: the knowledge of subject domains, the knowledge of students, and the knowledge of teaching methodologies. As a simulation of human teachers, a typical *standalone intelligent educational system* (Figure 2-1) usually consists of *an expert module, a student model module, a tutoring module, and a user interface module* (Brusilovsky, 1994).

- The *expert module* comprises the facts and rules of the particular domain to be conveyed to the students, i.e. the knowledge of the experts in this subject.
- The *student model module* represents the educational system's belief of the current state of a specific student, which includes the student's knowledge and skills on the domain, and the student's subject-independent characteristics relevant to the learning processes.
- The *tutoring module* is used to decide which pedagogic activities to be presented (lectures, hints to overcome impasses, advice, support, explanations, practice tasks, tests, etc.) based on the information of individual student in the student model module and its own tutorial structure.

- The *user interface module* is the communication component that controls interaction between the student and the system (Kinshuk, 1996).

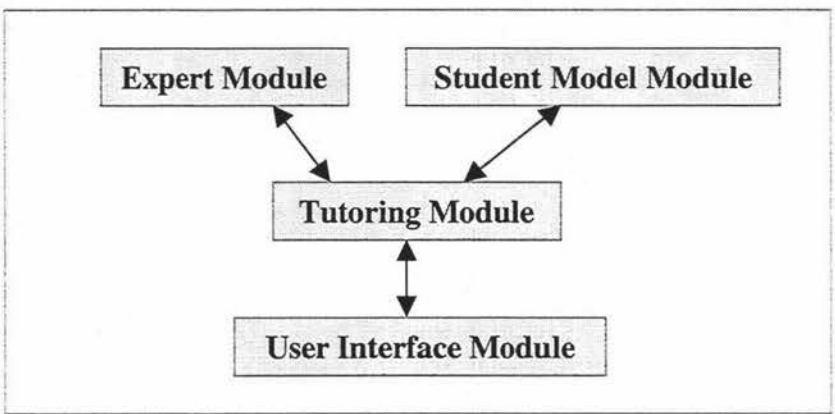

Figure 2-1 The architecture of standalone intelligent educational systems

2.2.2 Web-based Intelligent Educational Systems

The *web-based intelligent educational systems* are generally implemented by rearranging or distributing the standalone intelligent educational components between client and server, and adding some new components to facilitate the client-server communication. According to the location where the system functions are performed, three types of architectures can be specified as follows.

- **Replicated Architecture**

The system with the replicated architecture is an integrated program, for instance a Java applet, which can be downloaded from a specific URL to a student's machine. *The system resides and executes on the client side web browser.* The server is only act as a repository of a tutoring system to be downloaded. The Tutor (Vassileva, 1997) developed in the DCG authoring tool and ADIS (Waendorf & Tan, 1997) used this architecture.

- **Centralised Architecture**

In the centralised architecture, all system functions are performed in the server side, and the client side web browser is only used for displaying the system interface. For instance, the server may communicate with the clients through CGI (common gateway interface) programs on the server side. The students interact with the system using HTML entry forms in a standard web browser. The HTML entry forms can pass the requests and responses of students to the application server. The major disadvantages of this architecture are lack of immediate reaction to a user action and potential difficulties in handling complex client/server communication due to the stateless feature of CGI programs.

The systems employing the centralized architecture include algebra tutor PAT-Online (Ritter, 1997), anatomy tutor (Eliot, 97), SQLT-Web (Mitrovic, 2000), ELM-ART (Brusiovsky, 1996) and CALAT (Nakabayashi et al, 1997).

- **Distributed Architecture**

In the distributed architecture, the system functions reside in both the server and client sides (Figure 2-2). The ways in which the functions are distributed vary from system to system. Normally, the system functions related to user interactions execute on client side, and these functions are usually implemented as Java applets or JavaScript embedded in HTML pages. The communication mechanism between server and clients can be low-level socket connection and remote procedure invocation (RMI).

This architecture is used for teaching scientific inquiry skills (Suthers and Jones, 1997), and the web-based framework ID (Omega group, 2000).

Recent web-based systems tend to adopt CORBA (Common Object Request Broker Architecture) or Java-based distributed infrastructure. For example, in WALTS the client connects to the server by HTTP protocol only once for initial connection and downloading the client side application (JavaScript, Java Applet, etc.). After initialisation, a proprietary protocol

is employed for the client to deal with application server directly with the participation of the web server in the further communication (Choi & Woo, 2000).

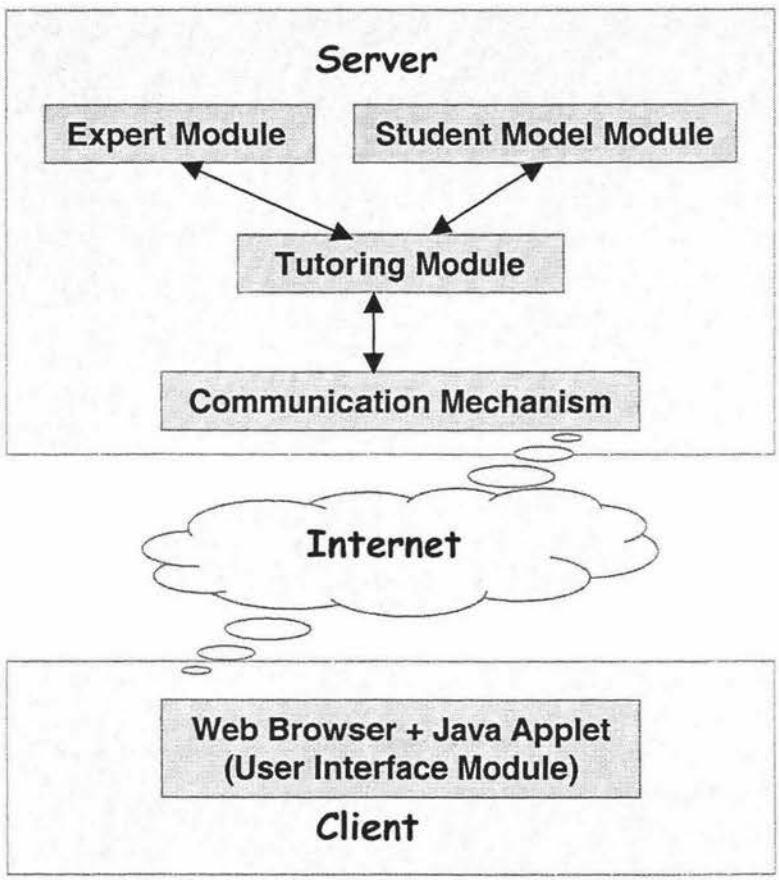

Figure 2-2 Architecture of distributed web-based intelligent educational systems

2.3 Information Handling in Web-based Intelligent Educational Systems

In a web-based intelligent educational system a large amount of information is needed to represent domain knowledge and to model the student learning behaviour.

2.3.1 Representation of Domain Knowledge (Domain Model)

It is obvious that a student model cannot simulate all the information about student's knowledge, and much of the student knowledge would not be even useful for a given domain. A student

knowledge model must be established on the basis of given domain knowledge structure that is also referred as the *domain model.*

From the viewpoint of the domain model structure, *curriculum elements* (CE) may be independent of one another, or more likely are linked to each other by one or more types of relationships. The most common relationships include *prerequisite* (learning a CE requires the mastery of its prerequisites) and *navigational priority* (a CE is presented before another). These relationships may connect all CEs in a subject domain to form a network.

Bayesian belief net is an example of the knowledge element network, which has been applied in several applications to update and infer student knowledge (Far & Hashimoto, 2000; Henze & Nejdl, 2000; Anjaneyulu, 1997).

2.3.2 Information in Student Models

A comprehensive student model should contain information about the student's domain knowledge prior to the use of the educational system, the learner's progress, preferences, interests, goals, and any other information related to the learner (Self, 1994).

Student models have usually been classified according to the nature and form of information contained in the models (Brusilovsky, 1994). Based on the relationship with the subject domain, the information held in student models could be divided into two major groups: *domain specific information* and *domain independent information.*

2.3.2.1 Domain-Specific Information

The model of domain-specific information is named as the *student knowledge model (SKM)* (Brusilovsky, 1994), which represents a reflection of the student's state and level of knowledge and skills in term of a particular subject. In relation to domain knowledge representation, student knowledge models are classified as follows.

- **Scalar Models**

A scalar model is the simplest form of SKM, which describes the level of learner's knowledge on the entire domain by means of a certain integral estimate such as a number ranging from 1 to 5.

- **Overlay Models**

If the entire domain model is made up of a set of knowledge elements or curriculum elements (CE), the overlay model represents the student knowledge as a subset of the domain model. A certain measure is assigned to each curriculum element based on the estimated student's understanding on that element. The measure can be a scalar (an integer, or probability measure, or a flag such as initial acquisition/assimilation/mastery) or a vector estimate (Brusilovsky, 1994; Kinshuk, 1996; Paiva, 1995).

- **Error Models**

Because overlay models cannot represent the errors that the students made, the bug models or error models are developed to define and reflect the reasons of erroneous student behaviours. The error models can be divided into *perturbation models* and *differential models.*

- *Perturbation models* assume one or more perturbations (misconceptions) exist for each curriculum element. The incorrect student behaviours (errors) may be caused by the application of one of misconceptions in place of the related correct knowledge element. The student knowledge is therefore represented by a union of a subset of the domain model and another subset of the misconception set with all misconceptions that the learner may have (Brusilovsky, 1994; Kinshuk, 1996; Paiva, 1995).

- *Differential models* capture misconceptions by only including the entities representing the differences between the expert knowledge and the learner's acquired knowledge (Paiva, 1995).

- **Genetic Models**

Although both *overlay models* and *error models* represent the students' knowledge states, they do not reflect the whole structure of domain knowledge. *Genetic models* represent the student knowledge developing process from simple to complex and from special to general. The genetic model can be described by a genetic graph, and it nodes and the relationships between the nodes represent knowledge elements and their interactions (Goldtein, 1979).

- **Other Domain Specific Information**

The other domain-specific information that may be stored in a student model includes:

- Student's *prior knowledge* about the domain
- *Records of learning behaviours* (number of lectures taken, number of helps asked, frequency of mistakes made while solving problems, reaction/answering time while solving problems, etc)
- *Records of evaluation /assessment* (qualitative and quantitative scores)

2.3.2.2 Domain-Independent Information

A student model also needs to cover a certain amount of domain-independent information in addition to the student's current knowledge level. The domain-independent information about a student may include learning goals, cognitive aptitudes, measures for motivation state, preference about the presentation method, factual and historic data, etc.

- **Goals**

Learning goals answer the questions of why the student uses the educational system and what the student actually wants to achieve. Learning goals are crucial for establishing the correct teaching strategies, as they represent some basic features of a particular learning process. Almost

one third of existing adaptation relies on learning goals (Brusilovsky, 1996a). The goals can be classified as follow:

- The *learning goal* that is a higher level goal and relatively stable.
- The *problem-solving goal* that is a lower-level goal and may change from one problem to another several times within a teaching unit.

- **Cognitive Aptitudes**

Shute (1995) identified a number of specific cognitive aptitudes in an overlay model besides student's general attributes:

- *General knowledge (GK)*
- *Inductive reasoning skill (IR)*
- *Working memory capacity (WM)*
- *Procedural learning skill (PL)*
- *Information processing speed (IPS)*
- *Associative learning skill (AL)*
- *Reflectivity*
- *Risk-taking.*

In the overlay model the curriculum elements were classified as three types: *symbolic knowledge (SK,* something representing something else by association or convention), *procedural skill (PK,* being able to apply rules in solution of a problem), and *conceptual knowledge* (*CK*, high-level relationships among concepts, schemas, and rules relating them together). The mastery of different types of curriculum elements was associated with one or more types of cognitive aptitudes (Capuano et al., 2000).

- **Motivational States**

Motivation State is the force that drives the learner to engage in learning activities. The student motivational state can be measured by a number of long-term and short-term parameters such as

motivation, effort, attention, interest, distraction, persistence, etc. These parameters are in turn associated with other factors including knowledge level, readiness, complexity of topic, learning outcome, etc.

Far and Hashimoto (2000) proposed a learner model that considered both student motivation and knowledge states. The student motivational state was represented in a Bayesian network as shown in Figure 2-3. The graph encoded the causal dependency among motivational aspects (an arrow from A to B is read as 'A influences B').

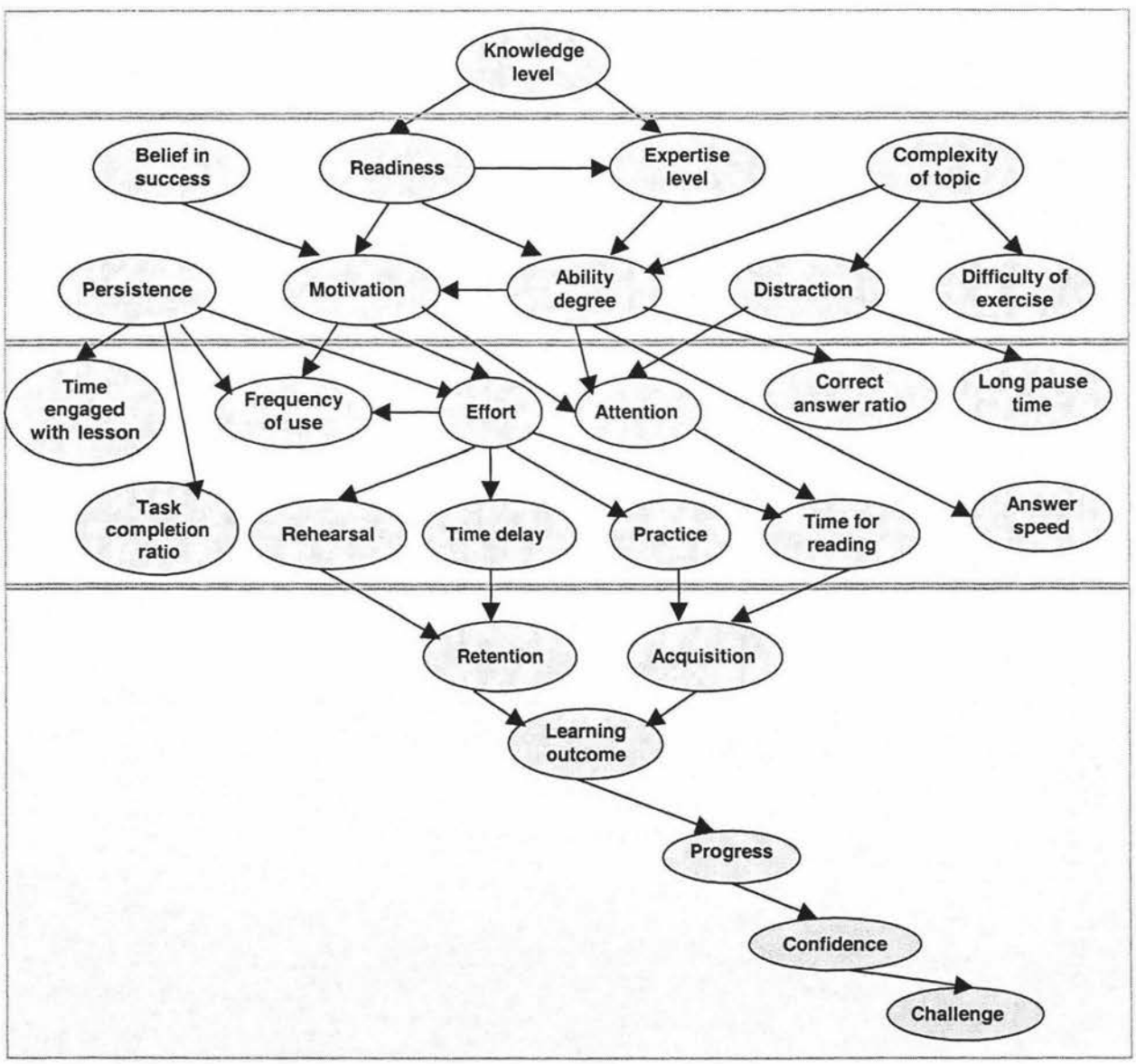

Figure 2-3 Student's motivational states model (Far & Hashimoto, 2000)

- **Background and Experience**

Both *background* and *experience* information can be used as bases for deriving student model parameters.

- *Background information* is about the student previous experience that may have impact on student learning achievement, such as profession, relevant work experience, perspectives etc. EPIAIM, C-Book, and Anatom-Tutor included the background information in student models for adaptation (Brusilovsky, 1996).

- *Experience information* is about how familiar the student is with the learning environment. The students who are quite familiar with the subject domain may be novices in using the educational systems and vice versa. This information is helpful in selecting appropriate adaptive navigation methods.

- **Preferences**

The users may have different *preferences* over a range of aspects of an education system. These preferences could be domain related or domain independent. Student preferences are considered different from other information stored in student models in that they cannot be deduced by the system. The students have to inform the system directly or indirectly about his or her preferences. It is important for a web-based educational system to present and organise the learning materials in the way that is both effective and preferred by the student. Individual student preferences can also be accumulated to form group student preferences in a group student model. The most important part of student preferences is the *learning style* that is correlated with *multiple intelligence*.

- **Multiple Intelligence**

Howard Gardner's most current research defines eight distinct intelligence forms stated as follows (Lane, 1998):

1) *Verbal/linguistic intelligence* – plays with word
2) *Logical/mathematical intelligence* – plays with questions
3) *Visual/spatial intelligence* – plays with pictures
4) *Musical/rhythmic intelligence* – plays with music
5) *Bodily/kinaesthetic intelligence* – plays with moving
6) *Intra-personal intelligence* – plays alone
7) *Interpersonal intelligence* – plays with socialising
8) *Naturalist intelligence* – plays with categories

Gardner suggested that everyone possesses all above intelligence but in varying degrees, consequently one person can show low ability in one domain area but high ability in another domain. According to the *multiple intelligence theory*, intelligent educational system should be individualised so that every student can be guided to achieve his or her maximum potential (Lane, 2000).

- **Learning Style**

A *learning style* is defined as the unique collection of individual skills and preferences that affect how a student perceives, gathers, and process learning materials (Johnson & Orwig, 1998). Multiple intelligence determines multiple learning styles. Just as everyone has his/her unique ways to see, hear or experience the world, every learner has different preferences for how, when, where and how often to learn knowledge (Lane, 2000).

As everyone has multiple learning styles, presenting information to students in only one style will not meet the need of all students. If the learning materials are presented in the way that best fits the learning style and multiple intelligence of an individual student, the student can learn better and faster, and consequently resulting in creativity increasing and academic achievement.

In a web-based learning system, various interaction methods can be provided to satisfy different learning styles by employing multimedia objects (video, audio, text etc). Unfortunately, most learners have not discovered the learning styles that suit themselves best.

Canfield created a learning style inventory to help students to determine preferred learning conditions (peer, goal setting, competition, instructor detail), areas of interest (numeric, qualitative, inanimate, people), modes of learning (listening, reading, iconic, direct experience), and course expectations (Lane, 2000).

Litzinger and Osif (1992) proposed that everyone has preferred and consistent learning behaviours and methods. They broke the learning process into cognition, conceptualisation and affective. Kolb catalogued the learning styles in details and Harman (1995) gave some examples of suitable teaching methods (Lane, 2000):

1) *Concrete experience*--- laboratories, fieldwork, observations, and trigger films.
2) *Reflective observation*---logs, journals, and brainstorming.
3) *Abstract conceptualisation*---lectures, papers, and analogies.
4) *Active experimentation*---simulations, case studies, and homework.

- **Factual and Historic Data**

A student model may contain a number of *factual* and *historic data* about an individual student such as *name, age, parents, ID, past education, interests*, etc. These are necessary for initialising an individual student model.

2.4 Student Modelling

A *student model* is the system's beliefs about the learner's knowledge, interests and goals. The student model should be constantly updated in accordance with the dynamic features of the student knowledge acquisition process (Brusilovsky, 1994). Student modelling was well documented by Self (1974), Brusilovsky (1994), Paiva (1995), and Kinshuk (1996).

A student model is *executable* if its present state can be utilised to simulate the behaviours of the modelled student when the student is solving a problem. The executable models are also referred to as *procedural models* (Brusilovsky, 1994).

There are a variety of on-line educational systems currently using student modelling techniques, include OLAE (Online Assessment of Expertise) (Martin & VanLehn, 1993), POLA (Probabilistic Online Assessment) (Conati & VanLehn, 1996), and University of Saskatchewan's MicroWeb project (Stauffer, 1996).

2.4.1 Initialisation of Student Models

A student model can be initialised in three ways: *explicit questioning*, *initial testing,* or by *stereotyping* (Self, 1994).

- **Explicit Questioning**

The initial student models are usually constructed by directly questioning the learner. This method is a very effect way to obtain general information about a student (Paiva, 1995).

- **Initial Testing**

By asking the learner to take a test, the initial parameters in the student models can be obtained by analysing the test results. In order to control the length of the test, concept of neighbourhood of knowledge states may be applied (if curriculum elements A and B are in the same neighbourhood, mastery of A implies mastery of B).

- **Stereotype**

Stereotype is defined as a representation of a collection of attributes that occur in a certain group of people. The stereotype is usually used to transfer explicitly acquired information about a group of people into guesses about an individual. This group of people should share the same interest according to a set of criteria. The information in a stereotype is adopted as default assumptions for initial parameters in the student models (Self, 1994). Stereotypes may be modified according to newly gathered group information. The user modelling prototype UMIE

built user stereotypes based on the theory that a user does not cease to belong in a stereotype since the triggers of the stereotypes are facts that do not change (Benaki et al, 1997).

2.4.2 Update of Student Models

Both *information sources* and *update methods* are important for updating student models.

2.4.2.1 Information Used for Updating Student Models

The information used for updating a student model may be categorised as follow according to the information sources.

- Information currently stored in the student model
- Information currently stored in other components of ITS, e.g. domain model (structure of expert domain knowledge) may be maintained in the components other than the student model module within an ITS.
- Information gained through interaction between student and system.

Above information may be obtained in following ways (Brusilovsky, 1994; Kinshuk, 1996).

- *Implicit acquisition,* by observing the actions of a student while being engaged in learning processes
- *Explicit acquisition,* by direct dialogues between the system and the student e.g. explicit questioning.
- *Structural acquisition*, by analysing the interrelations between curriculum elements, e.g. if curriculum element A is a prerequisite of B, mastery of B means mastery of A, and lack of knowledge on A implies lack of knowledge on B.
- *Historical acquisition,* by generalized estimate of the student knowledge and capabilities or the student previous experience.

2.4.2.2 Methods Used for Updating Student Models

The most important information for updating the student model is derived by analysing student responses to the questions presented by the system, problems solving processes and operations within the system. These analytic processes can be called *cognitive diagnosis* that is also defined as the process of inferring a person's cognitive state from his or her performance (Paiva, 1995).

- **Analysis of Student Responses**

Analysis of student responses is also called *performance measuring* (Brusilovsky, 1994). The questions in a tutoring system can be divided into two categories:

- Simple questions requiring the student to master only one curriculum element
- Complex questions requiring the student to master more than one curriculum elements

When handling the response to a simple question, correct answer results in an increase in the measure of the related curriculum element in the student model, and the incorrect answer leads to a decrease in the measure.

An analysis of the response to a complex question needs more efforts. In case of correct answer it may be enough to increase the measures of all related curriculum elements. For an incorrect answer, additional analysis (e.g. based on structure of domain model) and test may be needed to further identify the acquired and not acquired curriculum elements. If a perturbation error model is employed, in cases of incorrect answers one should also determine which perturbations of the relevant curriculum elements are responsible for the error.

- **Analysis of the Process of Problem Solution**

The *model tracing* technology is usually applied to analyses a problem-solving episode. The system contain all possible correct rules that can be used by the student in solving the problem together with a catalogue of misconceptions responsible for possible errors that may occur. By

solving the problem with the student, the system is able to detect all correct rules and misconceptions used by the student in every step of the problem-solving process (Brusilovsky, 1994).

- **Analysis of Student Actions**

Analysis of the student actions is the most complex process. In certain domains it is possible to define every student action as a result of acquisition of a set of curriculum elements or misconceptions. Without knowing the problem to be solved, the system is able to identify acquired curriculum elements or misconceptions by simply tracing student actions. This method is usually referred to as *issue tracing* (Brusilovsky, 1994).

- **Discounting Old Data**

Data ageing reduces the values of old data in the student model and gives greater importance to the data derived from the recent evidence. Data ageing is based on the assumption that the more time has elapsed from the formation of the old data, the greater the probability the evidence supporting the data is no longer relevant (Webb & Kuzmycz, 1998).

2.5 Usage of Student Model

In a web-based intelligent educational system the data in the executable student model are employed to drive the teaching processes. The basic usage of student models is to provide *individualised adaptation.*

2.5.1 Usage of Student Model in Intelligent Educational Systems

In an intelligent educational system, the tutoring module can perform a wide range of functions that may be conducted by a human tutor in traditional training, and the student model can be used to adaptively execute each of these functions. These functions include *knowledge development, error remediation, domain content representation, exploration space control,*

knowledge diagnostics (accurate definition of student's knowledge state), *strategic functions* (selection of global teach plans or strategies), *prediction functions* (student learning behaviour and learning path), *and assessment functions* (student assessment and ITS assessment) (Brusilovsky, 1994; Kashihara et al., 2000; Kinshuk et al., 2000; Self, 1988).

2.5.1.1 Knowledge Development

The process of an intelligent educational system to help a student knowledge development consists of several stages stated as follows.

- **What to Teach**

In 'what to teach' stage the student model provides information on what knowledge the student lacks. The tutoring module then will be able to decide *curriculum sequence* in goal-oriented tutoring mode, or provide *active help and feedback* during problem-solving process, or provide *passive help* under student requests.

- **Curriculum Sequence**

Curriculum sequence is used to provide the student with the most suitable individually planned sequence of knowledge units to learn and sequence of learning tasks (examples, questions, problems, etc.) to work with. It can help the student find an optimal path through the learning materials (Brusilovsky, 2000). Curriculum sequencing techniques can be subdivided into two types:

1) *High-level sequencing (knowledge sequencing)* uses student model and domain knowledge to select next teaching concept or topic.

2) *Low-level sequencing (task sequencing)* uses the student knowledge level recorded in the student model to determine next learning task or teaching operation (problem, example, test, and questions) within current topic.

The student models were used for curriculum sequencing in ITEM-IP (Brusilovsky, 1992b) ELM-ART (Brusilovsky et al., 1996), CALAT (Nakabayashi et al., 1997), InterBook (Brusilovsky & Schwarz, 1997), AST (Specht et al., 1997), MANIC (Stern et al., 1997), Medtec (Eliot et al., 1997), and DCG (Vassileva, 1997).

- **Active Help and Feedback**

After updating the student model by analysing student behaviours, the system can provide active help and feedback, which can be categorised as follow.

1) *Intelligent feedback to student solutions* provides students with extensive error feedback after analysing the student solution to a simple or complex problem. A recent example is ELM-PE (Weber & Möllenberg, 1995).

2) *Interactive problem solving support* provides students with intelligent help such as hint or remind on every step of problem solving by monitoring and examining student actions. The ACT Programming Tutor (Corbett & Anderson, 1992) and GRACE (McKendree et al.1992) provided this type of supports.

3) *Example-based problem solving support* provides students with helpful examples related to their earlier study record during the process of problem solution. An example using this technique is ELM-PE (Weber & Möllenberg, 1995).

- **Passive Help**

Whenever requested by a student, the system provides the students with passive help such as hint or answer to a question, or some extra explanation based on the student knowledge levels stored in the student model.

- **When to Teach**

In 'when to teach' stage the student model is used to choose an appropriate moment for knowledge development. This is particularly important when active help is needed during solution of problems.

- **How to Teach**

In 'how to teach' stage the student model is required for selecting proper *teaching actions* such as *explanations, tests, examples, and problems*. The selection is normally carried out according to the student learning style or preference as well as the student domain related knowledge level.

- **Implementation of Teaching Actions**

In this stage, certain teaching actions may be modified according to the state of student model. The degree of details involved in explaining a conception may be decided by means of the student model, for instance, the more the concept is studied, the less the details are provided in the explanation.

2.5.1.2 Error Remediation

Self (1988) identified the following remediation methods.

- *Error definition*: word description of error and recommendation for correction
- *Explicit remediation*: explicit presentation of correct knowledge
- *Implicit remediation*: prompts of correct knowledge or actions
- *Counter examples*: system-generated situations or problems
- *Demonstration of a solution method*
- *Access to previous experience*: previous experience stored in the student model
- *Repeat attempt: for persistent students*
- *Tactical retreat*

2.5.1.3 Domain Content Representation

Kinshuk et al (2000) proposed a *Multiple Representation Approach* for providing guidelines for multimedia objects manipulation. The approach recommends selection of multimedia objects based on learner's domain competence level.

2.5.1.4 Exploration Space Control

While a student is navigating in the domain space, the ITS automatically controls the exploration space by limiting information resources, number of searching paths and tools, and amount of presented information. The control is used for reducing the student's cognitive load, and is conducted according to the student competence level, experience, etc. (Kashihara et al., 1997).

2.5.2 Usage of Student Model in Web-based Educational Systems

Web-based intelligent educational systems may be thought as the intelligent educational systems that are ported to the web and may employ the extra educational methods powered by various web technologies. The adaptation of a web-based intelligent educational system is the implementation of the adaptation technologies for the intelligent educational systems in a web-based hypermedia environment. Because web pages have the nature of hyper-space, it provides a good platform for intelligent educational systems to practice their adaptation activities using hypermedia objects.

Most of the usage of student model in intelligent educational systems described above can be ported to web-based systems easily and some of them can be extended due to the advanced web-based technology.

- **Curriculum sequencing**

In the context of Web-based education, proper *curriculum sequencing* should be performed so that the learners with little or no external assistance are not lost in a large amount of hypermedia

objects and links. Web-based hyper-space can make it more flexible to guide students going through available information. Currently, it is used by almost all the web-based intelligent educational systems.

- **Intelligent feedback of student solutions**

Because intelligent feedback to student solution needs only one interaction between browser and server during a complete solution, it is a very suitable technology in the slow networks. The recent web-based intelligent educational systems using this technology are ELM-ART (an ITS for programming in LISP) (Brusilovsky et al., 1996) and WITS (an ITS for differential calculations) (Okazaki et al., 1997).

- **Interactive problem solving support**

This technology is not widely used in Web-based systems as in standalone systems, since it is usually difficult for the server-based applications to be interactive enough to support both watching the student actions and providing help on each step. Each interaction between browser and server may take a visible amount of time, and thus performing interaction on each step may make the problem solving process unbearable. The situation is improving with fast advances in web technologies.

Several systems demonstrated that the interactive problem solving support technology could work on the WWW. PAT-Online (Brusilovsky et al, 1997; Ritter, 1997), Belvedere (Suthers & Jones, 1997), ADIS (Warendorf & Tan, 1997). In above systems, reasonably small Java applets provided nice interactive interfaces, and the intelligent and adaptive parts resided on the server.

- **Example based problem solving**

Example based problem solving does not require extensive client-server interaction and can be naturally used in intelligent educational systems on the Web. ELM-ART applied this technology on the web (Brusilovsky et al., 1996).

2.6 Adaptation in Web-based Intelligent Educational Systems

By using student model various visible aspects of the educational systems can be adapted to the students. The same teaching materials can be presented differently to the students with different information stored in the corresponding student models. In a web-based hypermedia context, the contents of the web pages and the links from the web pages can be adapted, which are referred to as *adaptive presentation* and *adaptive navigation* respectively.

2.6.1 Adaptive Presentation

The goal of the *adaptive presentation technology* is to adapt the content of a hypermedia page to the user's goals, knowledge and other information stored in the user model (Brusilovsky, 2000). Boyle and Encarnacion (1994) demonstrated that adaptive presentation improved student performance. In a web-based hypermedia environment, not only the text but also the multimedia objects can be presented adaptively.

In a system with adaptive presentation, web pages can be dynamically produced or assembled from several pieces for each student or group of students to fit their special requests. Generally, qualified students receive detailed and in-depth information, while novices receive extra explanation (Beaumont 1994; Boyle & Encarnacion, 1994). The tutoring systems adopting the adaptive presentation techniques include ITEM/IP (Brusilovsky, 1992b), Anatom-Tutor (Beaumont, 1994), C-Book (Kay & Kummerfeld, 1994b), De Bra's adaptive course on Hypertext (Calvi & De Bra, 1997) Medtec (Eliot et al., 1997), ELM-ART, AST, and InterBook (Diego et al., 2000).

2.6.1.1 General Approaches of Adaptive Presentation

- **Additional Explanations**

Some parts of information about a concept may be hidden from a particular student if the information is not suitable for the student knowledge level (Brusilovsky, 1996a). As a result, in

addition to the basic presentation, some category of students can get additional information that is specially prepared for them but will not be shown to other students. This method was used in MetaDoc (Boyle & Encarnacion, 1994), KN-AHS (Kobsa et al., 1994), ITEM/IP (Brusilovsky, 1992b), EPIAIM (de Rosis et al., 1993), and Anatom-Tutor (Beaumont, 1994).

- **Prerequisite Explanations**

Before presenting the explanation of a concept, the explanations of all its prerequisite concepts whose contents are not fully understood by the student are inserted. C-book applied this technique (Kay & Kummerfeld, 1994b).

- **Comparative Explanations**

While presenting a concept, the comparative explanations for the already learned similar concepts are provided to stress similarities and differences between the two concepts. ITEM/IP, Lisp-Critic, and C-book applied this technique (Kay & Kummerfeld, 1994b).

- **Explanation Variants**

The system stores several variants for some parts of the page content, and the student gets the proper variant based on his or her student model data. This approach was adopted in Lisp-Critic (Fischer et al., 1990), Hypadapter (Hohl et al., 1996), SYPROS (Gonschorek & Herzog, 1995)

- **Sorting**

The fragments of information about a concept are sorted so that the pieces most relevant to user's background and knowledge level are placed toward the front. This method was implemented in Hypadapter (Hohl, Böcker & Gunzenhäuser, 1996) and EPIAIM (de Rosis et al., 1993).

2.6.1.2 Implementation of Adaptive Presentation

- **Conditional Text**

All information about a concept is divided into several chunks of texts. Each chunk is associated with a condition on the student knowledge level maintained in the student model. When presenting the information about the concept, the system presents only the chunks where the related conditions are true. This technique is very flexible, and can be employed to implement all the explanation adaptations listed above excluding sorting. ITEM/IP (Brusilovsky, 1992b), and C-book (Kay & Kummerfeld, 1994b) use it.

- **Stretch-Text**

It can also turn off and on different parts of the content according to the student knowledge level. In stretch text, there are a number of hot words within a content text, each of which is associated with an extra hidden piece of text for detailed in-depth explanation. A hot word is activated if the student knowledge level reaches a prescribed standard. This technique was proposed in the MetaDoc (Boyle & Encarnacion, 1994) and developed further in KN-AHS (Kobsa et. al., 1994).

- **Page Variants**

A system keeps two or more variants of the same page with different presentations of the same content. As a rule, each variant is prepared for one of several possible user stereotypes, and the system selects the page variant according to the user stereotype. This technique was applied in Anatom-Tutor (Beaumont, 1994) with the background stereotypes, and in C-book (Kay & Kummerfeld, 1994b) with the stereotypes reflecting user proficiency.

- **Fragment Variants**

The system stores several variants of explanations for each concept and the user gets the page that includes fragment variants corresponding to his or her knowledge about the concepts presented in the page. This idea was supported by the work of Paris and Anatomy-Tutor (Brusilovsky, 1996a).

- **Frame-based Technique**

All the information about a particular concept is represented in form of a frame. Slots of a frame can contain several explanation variants of the concept, links to other frames, examples, etc. Special presentation rules are used to decide which slots should be presented to a particular student and in which order. The frame-based technique was applied in Hypadapter (Hohl et. al., 1996), and EPIAIM (de Rosis et al., 1993). This technique and can be utilised to implement all explanation adaptations mentioned above except prerequisite and comparative explanations.

2.6.2 Adaptive Navigation

The goal of the *adaptive navigation technology* is to support the student in hyperspace orientation and navigation by changing the appearance of visible links. In particular, the system can adaptively sort, annotate, or partly hide the links of the current page to make it easier for users to choose next link. In a WWW context where hypermedia is a basic organisational paradigm, adaptive navigation support can be used very naturally and efficiently.

2.6.2.1 General Approaches of Adaptive Navigation

- **Global Guidance**

Global guidance helps students find the shortest way to the information they want. The most direct method is to provide the student with the links to follow at every browsing step. This method was used in WebWatcher (Armstrong et al., 1995), Adaptive HyperMan (Mathé & Chen,

1996), and HYPERFLEX (Kaplan et al., 1993). The student's study goal and domain knowledge level will be the basis of global guidance. Curriculum sequence can provide global guidance with dynamic button "next".

- **Local Guidance**

Local guidance helps students move one navigation step by suggesting the most relevant links from the current node according to the student preferences, knowledge level, and background. Methods used in educational hypermedia are sorting links and direct guidance according to the user knowledge (Brusilovsky & Weber, 1996; Kushniruk & Wang, 1994; Pérez et al., 1995).

- **Global Orientation**

Global orientation helps the students to understand the overall course structure and his/her absolute position in the hyper-space. It can be implemented by applying hiding and annotation technologies. Hiding reduces the size of the visible hyper-space and can simplify both orientation and learning. Keeping a node in the same annotation at different pages makes the student recognise the nodes he or she met before and understand the current position easily. This method was used in ISIS-Tutor (Brusilovsky & Pesin, 1994), HyperTutor (Pérez et al, 1995; Clibbon, 1995), Hypadapter (Hohl, Böcker & Gunzenhäuser, 1996), ISIS-Tutor, SYPROS (Gonschorek & Herzog, 1995), HyperTutor and Hynecosum (Vassileva, 1996).

- **Local Orientation**

Local orientation helps the user by pointing out what is around and his or her relative position in the local hyper-space. It can also be implemented by annotation and hiding.

Annotation provides additional information about the nodes available from the current node. The current states of the nodes behind visible links can be displayed by showing several gradations of link relevancy, reflecting several levels of user knowledge of the nodes, (Brusilovsky & Pesin, 1994; Brusilovsky & Zyryanov, 1993; de La Passardiere & Dufresne, 1992; Schwarz et al.,

1996), outlining the links related to the current goal (Brusilovsky & Pesin, 1994), and providing special annotation for links not ready to be learned.

Hiding limits the number of navigation opportunities to decrease the cognitive overload and let the students concentrate themselves on analysing the most relevant links. Hiding the links to the nodes that are not yet ready to be learned was used in ISIS-Tutor (Brusilovsky & Pesin, 1994), HyperTutor (Pérez et al., 1995; Clibbon, 1995), and Hypadapter (Hohl, Böcker & Gunzenhäuser, 1996). Hiding the links to the nodes that belong to the educational goals of subsequent lessons is done in ISIS-Tutor and SYPROS (Gonschorek & Herzog, 1995).

2.6.2.2 Implementation of Adaptive Navigation

Direct guidance, sorting, hiding, and *annotating* are the primary technologies for adaptive navigation support. Most existing adaptation techniques use exactly one of these ways to provide adaptive navigation support. However, these technologies are not contradictory and can be used in combinations. For example, ISIS-Tutor (Brusilovsky & Pesin, 1994) used direct guidance, hiding, and annotation and Hypadapter (Hohl, Böcker & Gunzenhäuser, 1996) used sorting, hiding, and annotation. In particular, the direct guidance technology can be naturally used in combination with any of the three other technologies" (Brusilovsk, 1996).

- **Direct Guidance**

Direct guidance is the simplest way to provide adaptive navigation support, but it provides limited or no support for the students who would not like to follow the system's suggestion. It can be implemented by visually outlining the link to the "best" node as in Web Watcher (Armstrong et al., 1995) or by presenting an additional dynamic link connected to the "best" node as in ISIS-Tutor (Brusilovsky & Pesin, 1994), SHIVA (Zeiliger, 1993), HyperTutor (Pérez et al., 1995), and Land Use Tutor (Kushniruk & Wang, 1994).

- **Sorting**

Sorting is to reorder all the links of a particular page according to the student model and to some user-valuable criteria, such as, the more close to the top, the more relevant the link is. The limitation is that it can only be used with non-contextual links, and makes the order of links non-stable, but the stable order of options is important for novices (Debevc, Rajko & Donlagic, 1994; Kaptelinin, 1993). It is useful and can significantly reduce navigation time in information retrieval applications (Armstrong et al., 1995; Kaplan et al., 1993; Mathé & Chen, 1996, Kaplan et al., 1993), on-line documentation systems (Hohl, Böcker & Gunzenhäuser, 1996), and educational hypermedia (Tomek et al., 1993).

- **Hiding**

Hiding restricts the navigation space and reduces students cognitive overload by hiding links to irrelevant pages or nodes. It can be used with all kinds of non-contextual, index, and map links by hiding buttons or menu items (Brusilovsky & Pesin, 1994), and by transferring contextual links from clickable "hot words" to normal text (Gonschorek & Herzog, 1995; Pérez et al., 1995). Hiding is also more transparent to the student and looks more "stable" than sorting as the links are usually added incrementally, but not removed or reordered (Brusilovsky, 1996).

- **Annotation**

Adaptive annotation can tell students more about the current state of the annotated links by using some forms of comments. The annotation can be implemented in the form of text or visual cues, for example, varied icons (de La Passardiere & Dufresne, 1992; Schwarz et al., 1996), colours (Brusilovsky & Pesin, 1994; Brusilovsky & Zyryanov, 1993), or font sizes (Hohl, Böcker & Gunzenhäuser, 1996).

Annotation can distinguish several levels of relevancy (not just relevant and non-relevant as in hiding) as it implemented in Hypadapter (Hohl, Böcker & Gunzenhäuser, 1996), ELM-ART

(Brusilovsky et al., 1996), InterBook (Brusilovsky & Schwarz, 1997), WEST-KBNS (Brusilovsky, Eklund & Schwarz, 1997), and AST (Specht et al., 1997)

2.6.3 Adaptive Collaboration Support

Adaptive collaboration support is a very new technology that was developed with development of network-based educational systems. It can use system's knowledge about different students to form a matching collaborating group. Existing examples include formation of a group for collaborative problem solving at a proper moment of time (Hoppe, 1995; Ikeda, Go & Mizoguchi, 1997) and finding the most competent peer to answer a question about a topic (Bishop, Greer & Cooke, 1997; McCalla et al., 1997).

2.7 Summary

The intelligent educational system usually consists of an expert module, a student model module, a tutoring module, and a user interface module. Web-based educational systems arrange these modules in client or server side as replicated, centralised or distributed architectures. Web-based educational systems contain data of both domain model and student model. These data will be used for student model initialisation and may be updated by student modelling process. As a result, web-based intelligent educational systems can provide individualised adaptation based-on student model. The basic structure and operations of the student model module and its relationship with other components in an intelligent educational system are illustrated in Figure 2-4.

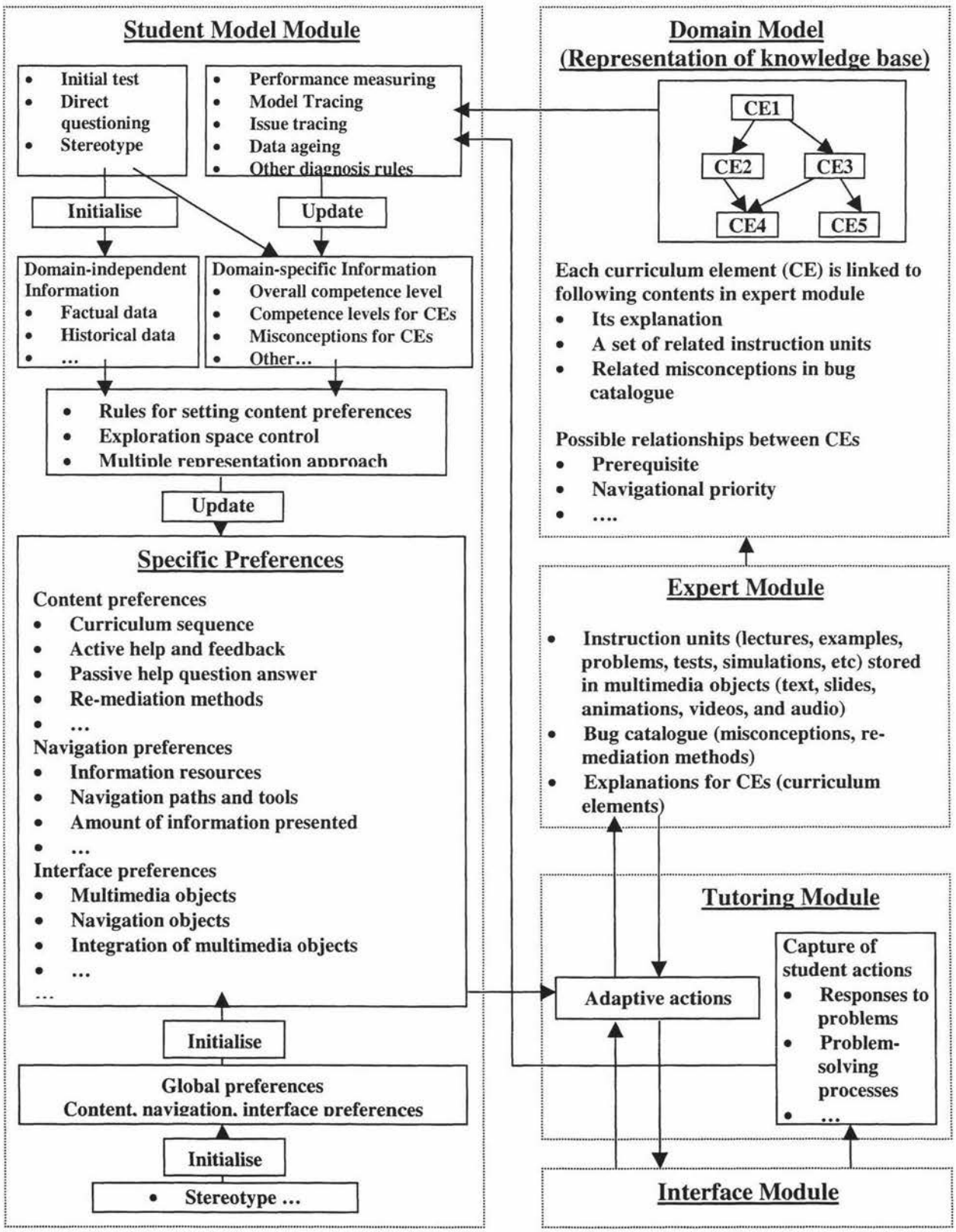

Figure 2-4 Structure of student model module and its relationship with other modules

Chapter 3 System Architecture Design and Implementation

3.1 Introduction

This chapter presents the basic architecture of the web-based educational system for testing the student model. The Java related techniques and client-server communication methods used in the system are discussed. After introducing the design of the system interfaces, the Java implementation of the system is described in details.

3.2 System Architecture in Relation with TILE Project

The Technology Integrated Learning Environment (TILE) project is designed to provide an integrated system for the management, authoring, delivery and monitoring of education at a distance (Jesshope et al., 2000). The modular framework of the system includes the modules of a typical Intelligent Educational System (IES) --- *an expertise module, a student model module, a tutoring module, and a user interface module.* These modules are to be developed as a range of tools to be integrated into the whole system.

The overall architecture of the TILE project is based on *distributed, three-tire client-server* architecture. The client is only concerned with user interface and connection to the server, most of the course delivery logic is located in the server side, from which the database can directly be accessed (Gehne et al., 2000). A relational database is built in the server side for flexible distribution, update and synchronisation of course materials, and easy standardisation of the interface of various tools.

The system architecture of this project is mainly based on the framework of TILE project. Because the other modules of the TILE project have not been established yet, the knowledge base

and the course delivery logic have to be built along with the prototype student model in this project for testing the prototype student model.

Figure 3-1 describes the scope of this study in relation to the three-tier architecture, in which the system implementation details and technologies used for the implementation are emphasised.

3.3 Java and Related Network Technologies Used in this Project

To implement the three-tier architecture, Java and Java-related open source tools are utilised in this project. The *server* employs the public-domain tools such as *Apache, Tomcat*, and *InstantDB*. The *client* is a collection of HTML pages embedded with Java applets for client-server communication and the student model presentation. Java and the related network technologies used in the system are discussed below.

3.3.1 Java

Java is a powerful, platform-independent programming language. Java has the benefits of being simple, object-oriented, distributed, interpreted, robust, secure, architecturally neutral, portable, high-performance, and dynamic. No programming language makes it as easy to access the Internet as Java. With Java's java.net package, elegant stream-based I/O classes, and easy-to-use multithreading capability, network programming becomes straightforward.

Due to its object-orientated features and powerful networking capabilities, Java is considered as the most natural and efficient platform for developing and delivering this distributed web-based application.

Jbuilder is an integrated Java development environment that provides visual tools and wizards for easy and rapid application development. *Jbuilder* supports for the latest Java standards. *Jbuilder4* is used to develop the prototype student model and the related testing system.

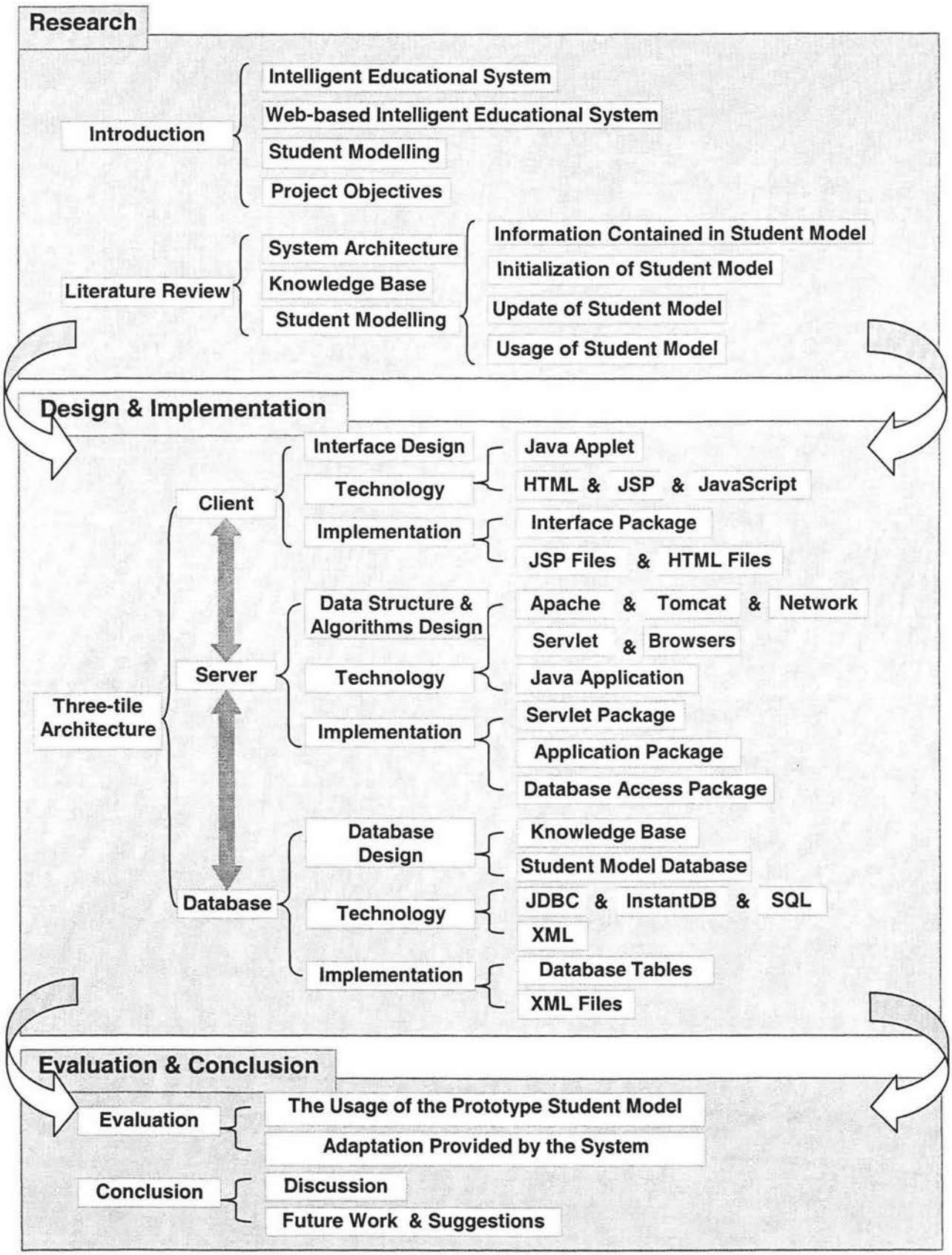

Figure 3-1 Scope of this study in relation to the three-tier architecture

3.3.2 Applet

An *applet* is a Java program that can be included in an HTML page and run in multiple web browsers without installation. When a Java enabled browser is used to view a web page that contains an applet, the applet's code is transferred to the browser and executed by its Java Virtual Machine (JVM).

Java Plug-in software enables Java applets to run using Sun's Java Runtime Environment (JRE), which provides a Java compatible environment for web browsers. By this way, applets can take full advantage of the latest features of the Java platform and be assured that they will run reliably and consistently.

Because the Java applet can provide a fully interactive graphical user interface running locally on the client browser, it is the best choice for creating dynamic visualisation of the student model data that are constantly changing and maintained in the server side.

This system used a Java applet class (ControlApplet) to build a distributed graphical user interface (including content tree, concept list, and suggested links) that can run in the Web browser on any student computer with the help of Java plug-in. Since the server-side Java programs perform the tasks of tracking student behaviours and implementing the student model update algorithms, the small and nimble Java applet can deliver relatively sophisticated demonstrations on the processes of the student model update.

3.3.3 Servlet

Servlets are server-side Java programs that provide a component-based, platform-independent method for building web-based applications. Since the Java servlet technology not only provides the features of traditional CGI scripts but also has the benefits of efficiency and portability, it is used to develop the server-side components for client-server communication in this project. The advantages of Java servlets in relation to this project are summarised as:

- The segmented *design* of Java servlets enables and encourages handling requests with a cascade of specialised objects instead of doing everything in one big script as in CGI programs. Therefore Java servlets are easier to be revised, maintained, and reused than CGI scripts. This project implements several servlet classes, each of which only deals with one type of client-server communication.

- *A servlet* is only instantiated once by the servlet engine when the client first accesses the URL it resides, and all subsequent accesses are put through that one object. This improves performance by keeping the response time of servlets lower than that of CGI programs that must be run once for every access. In this project the performance improvement by using servlets is obvious due to the frequent client-server communication during the student working process.

- Java servlet technology provides means to make certain objects to be unique to a user session and others to be shared among all users. In this project, the data structures representing the student model data are maintained exclusively for each student learning session.

- *Servlets* allow dynamic interaction between a web browser and components on the server, which accept data via HTTP and send back HTML code. This feature meets the request of dynamically tracking student behaviours in this study.

- *Servlets* can access all Java APIs, including use JDBC API for uniform database access. In this project, two servlets (LoginServlet and SaveServlet) were used to load data from and save data into the database through JDBC.

3.3.4 JSP

JSP (JavaServer Pages) technology is an extension of the servlet technology that enables the rapid development of dynamic web pages. A JSP file is actually compiled into servlets by the

servlet engine when the JSP file is first required from a client. Since it is tedious to output HTML using servlets, the JSP pages are written for producing the complex HTML outputs.

3.3.5 JDBC

JDBC technology is an API that allows access to the tabular data sources from Java programming language. It provides cross-DBMS (database management system) connectivity to a wide range of SQL databases. To use the JDBC API with a particular database management system, a JDBC technology-based driver is needed to mediate between JDBC technology and the database. In this project, JDBC was employed to access the student model database that is managed by InstantDB - a relational database management system written purely in Java.

3.3.6 Tomcat

Tomcat is a Java servlet running environment or a servlet engine that manages execution of servlets and server pages. It is an open-source implementation of Java Servlet and JavaServer Pages technologies developed at the Apache Software Foundation. Tomcat can be used as either a stand-alone container (mainly for development and debugging) or as an add-on to an existing web server (currently Apache, IIS and Netscape servers are supported). Since Tomcat is considered as the most stable and efficient open-source servlet container, it was selected as the servlet running shell in cooperation with Apache HTTP server to perform client-server communication tasks.

3.3.7 XML

Extensible Mark-up Language (XML) is a universal format for structured documents and data on the Web. XML documents simply mark up pages with descriptive tags whose meaning need not be described or explained. Java JAXP enables the applications to parse and transform XML documents in a way that is independent of a particular XML processing implementation. SAX is the event-driven, serial-access mechanism for accessing XML documents. SAX is the simplest and least memory-intensive mechanism for dealing with XML documents when compared with

other technologies such as COM and XSLT. Therefore the SAX technology supported by Java JAXP was used in this study for converting XML teaching materials into individualised HTML pages adapted to the student model data. In future development the XSLT technology may be used to convert HTML teaching materials to XML files for storage and adaptations.

3.3.8 Java Scripts

JavaScript is a scripting language that can be inserted into an HTML page. JavaScript is supported by both Netscape and Internet Explorer. Because JavaScript can put dynamic text into an HTML page and can be set to execute in response to a number of web browser events, it is used to perform the tasks needed when certain student actions take place. For instance, a JavaScript function was written for sending message to inform the server to save student model data back to the database when the student closes the web browser.

3.4 Basic Structure Design of the Three Tier Architecture

The system in this project is implemented with a *three-tier, client-server* architecture. The client is the presentation interfaces that are implemented as HTML frames embedded with Java applets and run in a web browser. The Java application programs for performing the student model update and adaptation reside in the middle layer server. The server-side application components communicate directly with the backend database storing information of course materials and student model. Figure 3-2 shows the three-tier architecture of the system.

3.4.1 Client Tier

The client tier is a web browser running in the student computer. The browser functions as the user interface of the tutoring model and is responsible for handling adaptive presentation of course materials and providing adaptive navigation support.

The main interface uses an HTML frame that is divided into two parts. A front-end applet is placed on the left side of the main HTML frame. The applet is used to dynamically track

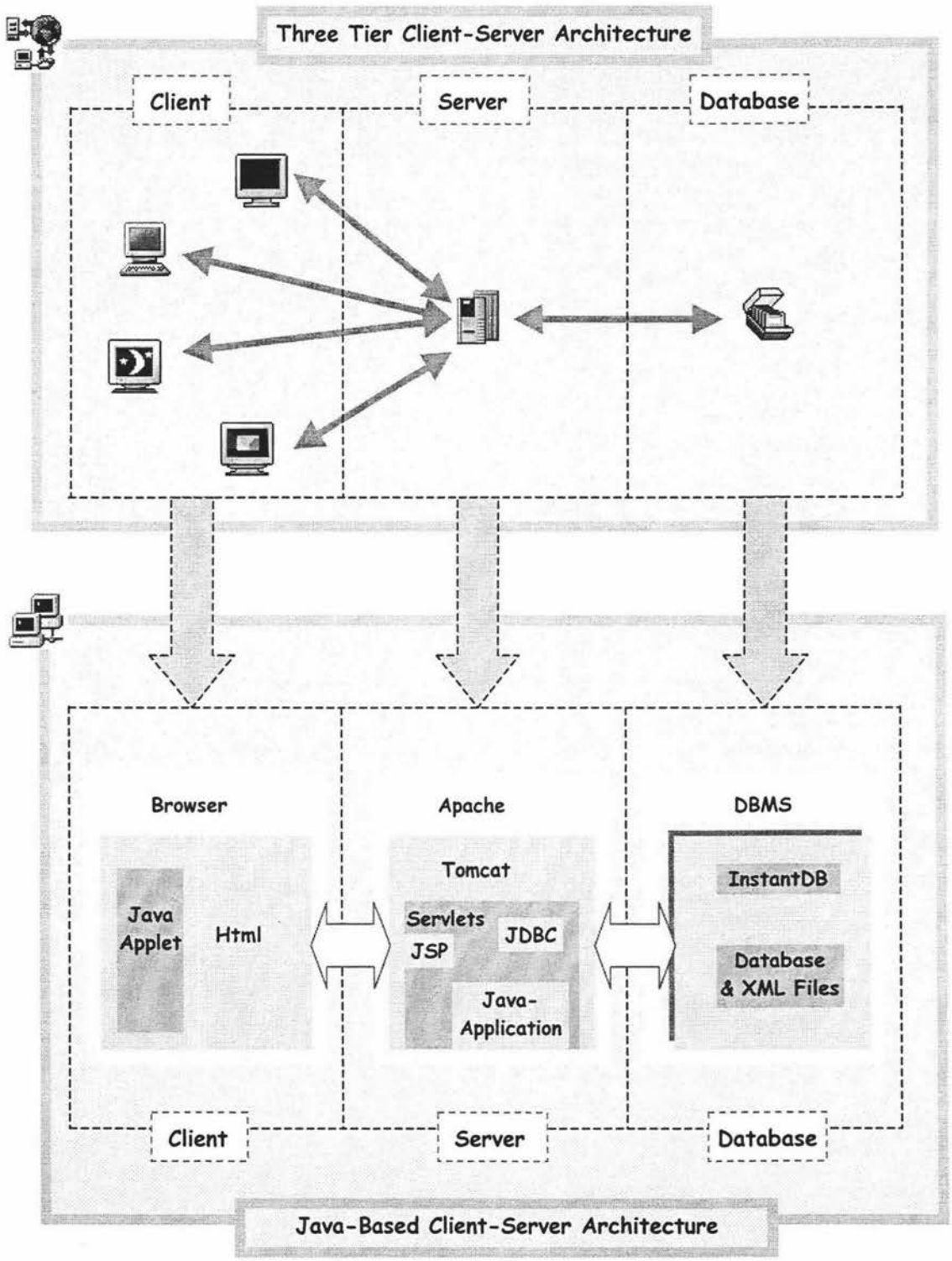

Figure 3-2 Three-tire architecture of the system

student working process and to provide the student with navigation adaptation support. The course materials are presented as HTML files on the right side. The HTML forms are used to submit the student's answer to questions to the corresponding servlet.

3.4.2 Middle Tier

The middle tier that resides in the server side handles the student model initialisation and update logic. It is responsible for receiving client requests, processing the data contained in the requests, applying student model initialisation and update logic to the data, and generating a client response based on the updated student model. The middle tier consists of the servlets objects communicating with client tier, the application objects simulating the knowledge base and student model, and the database access objects. The servlets process the requests from the client-side applets, invoke methods in the application objects, and send the produced presentations back to the client. The application objects initialise and update the student model data, and generate adaptive HTML pages from the XML files. The application objects use the database access objects to retrieve data from the database during initialisation and to save the modified data back to the database after the student finished a working session.

3.4.3 Data Management Tier

The database management system (DBMS) stores the data required by the middle tier. It also resides on the server side and is composed of a back-end database server - InstantDB. By using JDBC API, the Java application components can access the database by using the SQL call-level interface.

3.5 Client-Server Communication

This three-tier architecture needs two-way communications between Java applets in the client side and the servlets in the server side. The applets capture the student actions and pass the

information to the servlets for student model update. The servlets send the adaptive contents generated according to modified student data back to the applets.

The communication mechanism between clients and server is mainly implemented by opening an HTTP socket connection between the applet and a relevant servlet. This includes two steps:

- Firstly, the applet has to open a connection to the specified servlet URL.

- Secondly, the applet can send an output stream to or get an input stream from the servlet.

The applet can send data to the Servlet by GET or a POST method. The POST method is more powerful because any form of data can be sent through setting the content type in the HTTP request header.

A high level of data abstraction, *communicating with object serialisation,* is used in this project. The serialised Java objects are passed between applet and servlet instead of individual parameters. For example, the student model application package has a Student class that encapsulates all of the domain independent information about a student. Instead of passing each parameter of student information (such as student ID and password) as name value pairs, a true Java object *Student* is passed between an applet (loginApplet) and its corresponding servelet (CheckloginServlet).

3.6 Interface Design of the System

The system interfaces (Figure 3-3) are designed to demonstrate the student model working process, which include the login interface, main course content presentation interface, and several display windows.

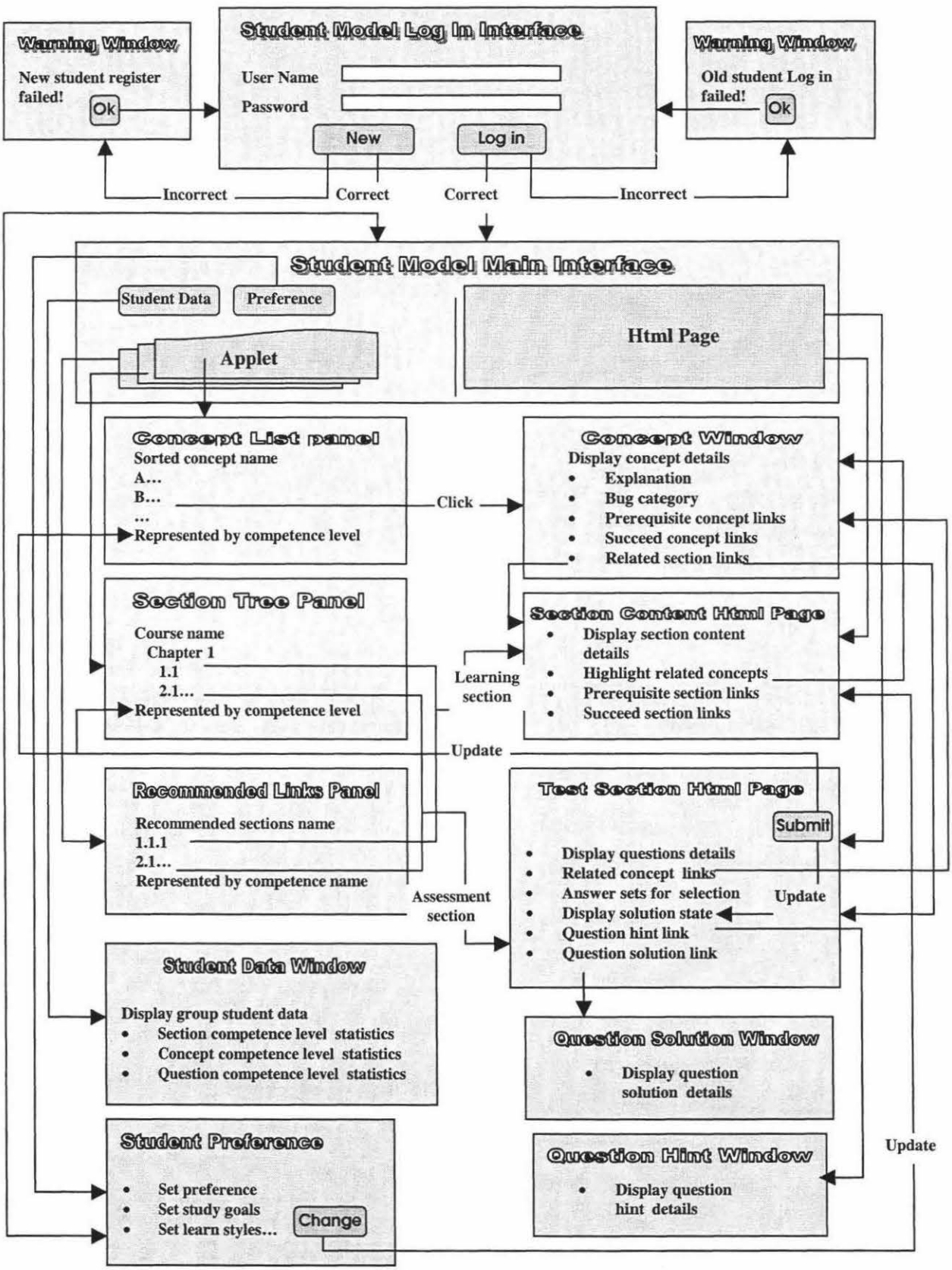

Figure 3-3 Design system interfaces

- **Login Interface**

The login interface is used for students to specify IDs and select courses. The student data are sent to the server for verification and student model initialisation.

- **Course Content Presentation Interface**

The content presentation interface is an HTML frame including a control frame and a content frame.

The control frame consists of a control applet that has the following functions.

- Display *content tree,* from which the student can select any section title for showing its content in the content frame.
- Display sorted *concepts list* from which the student can select any concept for showing its description in the concept window.
- Display the list for the *recommended sections* from which the student can select any recommended section title for showing its content in the content frame.
- Graphically display student competence levels of sections and concepts.
- Open student preferences window.
- Open student data window.

The content frame is used to display the content of a section.

- For a non-test section the links for prerequisite sections, succeeding sections, and related concepts are provided.
- For each question in a test section the links for related concepts are provided.

- **Concept Window**

The concept window is employed for displaying the detailed explanation of a concept.

- **Hint and Solution Window**

This window is used for displaying a hint or solution of a question.

- **Student Preference Window**

Preferences can be different for different students and can be different for the same student at different time. This window is used for students to modify their preferences.

- **Student Data Window**

This window displays group student model data.

3.7 Implementation Details of the Three Tier Architecture

The program files and classes are organised into several packages according to their functions and positions in the three-tier architecture.

3.7.1 JSP Files and Student Model Interface Package

There are several *JSP files* used for presenting student model interfaces and embedded Applets, their functions are described in the table 3-1. *Student model interface package* includes all the applets, panels and the classes used by them. Table 3-2 describes the functions of each class in this package.

Table 3-1 JSP Files and HTML Files

File Name	File Function
Login.jsp	Call Applet -- *LoginApplet.java*
Course.jsp	Present the HTML Frame as the main interface
Control.jsp	Call Applet -- *ControlApplet.java*
GroupModel.jsp	Call Applet -- *GroupModelApplet.java*
Content.HTML	Inform the user while loading the course content

Table 3-2 Student Model Interface Package

Class Name	Class Function
LoginApplet	▪ Get student name and password ▪ Check if the student name and password are in valid form ▪ Present massage dialogue boxes for login information ▪ Sent student input data (name, password, and courseID) to the server side program (*CheckLoginServlet*)
ControlApplet	▪ Open student preference window ▪ Open group student data window ▪ Call *GroupModelApplet.java* ▪ Add *SectionViewTreePanel*, *ConceptListPanel*, and *RecommendedLinkedListPanel* to a tabbedPane ▪ Create a *UpdateThread* object and run it
GroupModelApplet	▪ Present group student model data about sections, concepts, and questions based on statistics obtained by summarising the data of all students in the group. ▪ Provide the explanations for the graphic presentation of statistic results
SectionViewTreePanel	▪ Present the content tree of a course ▪ Display section competence levels ▪ Display update process of section competence levels ▪ Provide the explanations for the graphic presentation of section competence levels

Table 3-2 -- Continued	Student Model Interface Package
Class Name	**Class Function**
RecommendLinkPanel	▪ Present recommended section links of a course ▪ Display the competence levels of the recommended sections ▪ Provide the explanations for the graphic presentation of section competence levels
ConceptListPanel	▪ Present concept list of a course ▪ Display the concepts competence levels ▪ Provide the explanations for the graphic presentation of concept competence levels
SectionView	▪ Contain data necessary for displaying the section information (section id, course id, section title, and section competence level)
ConceptView	▪ Contain data necessary for displaying the concept information (concept id, concept name and concept competence level)
UpdateThread	▪ Updates section competence levels according to study time
Utilities	▪ Set URL address for easy use in other classes

3.7.2 Student Model Application Package and Servlets Package

According to the different roles of the classes involved in the server-side operations, the middle tier classes are grouped into two packages. *Student Model Application Package* contains the classes for executing the student model initialization and update logic (Table 3-3). These classes consist of the basic course units -- *Sections, Concepts, Questions* and the data structures for knowledge base and course-related student data -- *SectionTable, ConceptTable,* and *QuestionTable.* The *Servlets Package* contains all the servlets classes used for client-server communication and HTML files output (Table 3-4).

Table 3-3 Student Model Application Package	
Class Name	**Class Function**
Concept	▪ Contain detailed information on a concept (ID, related courses, related sections, related questions, and student competence level) ▪ Load concept from the database by calling methods of the relevant database access class ▪ Save concept back to the database by calling methods of the relevant database access class
ConceptTable	▪ A hash table for storing the data of all concepts in a course and their related competence levels for a specific student ▪ Initialise concept table by loading all relevant concepts from the database ▪ Update the concept competence levels according to predefined algorithms ▪ Save concept table back to the database
BugCategory	• Contain a concept and its related error category details
Section	▪ Contain detailed information on a section (ID, related courses, related concepts, related questions, and student competence level) ▪ Load section from the database by calling methods of the relevant database access class ▪ Save section back to the database by calling methods of the relevant database access class
SectionTable	▪ A combination of tree and hash table for storing the data of all sections in a course and their related competence levels for a specific student ▪ Initialise section table by loading all relevant sections from the database ▪ Update the section competence levels according to predefined algorithms ▪ Save section table back to the database
AdaptiveContent-Handler	• Process *XML files* to produce adaptive contents based on student model data

Table 3-3 -- Continued Student Model Application Package	
Class Name	**Class Function**
Question	▪ Contain detailed information on a question (ID, related courses, related concepts, related section, and student solution state) ▪ Load question from the database by calling methods of the relevant database access class ▪ Save question back to the database by calling methods of the relevant database access class
QuestionTable	▪ A hash table for storing the data of all questions in a course and their related solution states for a specific student ▪ Initialise question table by loading all relevant questions from the database ▪ Update the question competence levels according to predefined algorithms ▪ Save question table back to the database
Solution	• Contain the solution to a specific question
Hint	• Contain the hints to a specific question
Student	▪ Contain detailed information on a student (learning style, preferences, study goals, experience, overall competence level, course Id, student Id, password, etc.) ▪ Load student data from the database by calling the relevant database access method ▪ Save student data back to the database by calling the relevant database
GroupConcept	▪ Summarise statistics on the competence levels of a concepts for a group student
GroupSection	▪ Summarise statistics on the competence levels of a section for a group student
GroupQuestion	▪ Summarise statistics on the solution states of a questions for a group student

Table 3-4 Servlets Package

Class Name	Class Function
CheckLoginServlet	▪ Check if a student logs in successfully ▪ Create the objects of *SectionTable, ConceptTable, QuestionTable,* and *Student.* ▪ Save the created objects in the session of the browser
InitialisationServlet	▪ Call the methods in *SectionViewTree, ConceptList,* and *RecommendedLinks* for initialisation
UpdateServlet	▪ Call the methods in *SectionTable, ConceptTable* and *QuestionTable* for updating competence levels and solution states
ContentPageServlet	▪ Open Concept window ▪ Open Hint and Solution window ▪ Present HTML forms for test sections ▪ Call *AdaptiveContentHandler* for non-test sections ▪ Provide links to the Answer and Hint windows
QuestionHandlerServlet	▪ Update question solution state ▪ Reload Applet *ControlApplet*
ConceptPageServlet	▪ Present concept details in HTML files
HintServlet	▪ Present the hint of a question in HTML files
SolutionServlet	▪ Present the solution of a question in HTML files
PreferenceServlet	▪ Present student preference selection items in an HTML file

Table 3-4 -- Continued Servlets Package	
Class Name	**Class Function**
PreferenceHandlerServlet	▪ Set student domain independent data according to the values submitted by the student preference form
GroupModelServlet	▪ Create arrays of *GroupSection*, *GroupConcept,* and *GroupQuestion* ▪ Output the above data to *GroupApplet*
DataSaveServlet	▪ Save the competence levels and other data in data structure *SectionTable, ConceptTable, QuestionTable,* and *Student* into relevant database tables

3.7.3 Student Model Database Access Package

Student Model Database Access Package includes the classes used for performing database related functions. These classes are executed in the server side and can directly access and operate on the database. The database related functions include *add, delete, insert, search,* and *update* with a specific cell or row in a database table. The functions of the classes in this package are described in table 3-5.

Table 3-5 Student Model Database Access Package	
Class Name	**Class Function**
StudentDataAccess	▪ Check if the student is a new student ▪ Check if the student is a old student ▪ Load *student data* and *OverallCompetenceLevel* from database tables ▪ Update database table -- *Student* ▪ Update database table -- *OverallCompetenceLevels*

Table 3-5 -- Continued	Student Model Database Access Package
Class Name	**Class Function**
ConceptDataAccess	▪ Load *ConceptTable* from database (tables *Concept* and *ConceptCompetenceLevel*) ▪ Get bug categories from database (table *BugCategory*) ▪ Get array of Concepts from database (table *ConceptCompetenceLevel*) ▪ Get prerequisite concept Ids from database (table *ConceptPrerequisiteRelation*) ▪ Get succeed concept Ids from database (table *ConceptPrerequisiteRelation*) ▪ Get related question Ids from database (table *QuestionConceptRelation*) ▪ Update concept competence levels in the database (table *ConceptCompetenceLevel*)
SectionDataAccess	▪ Load section tree from database (table *Section*) ▪ Get section details from database (table *Section* and *SectionCompetenceLevel*) and add the data to *SectionTable* ▪ Get array of Sections from database (table *SectionCompetenceLevel*) ▪ Get prerequisite section Ids from database (table *SectionPrerequisiteRelation*) ▪ Get succeed section Ids from database (table *SectionPrerequisiteRelation*) ▪ Get related concept Ids from database (table *Concept*) ▪ Get related question Ids from database (table *Question*) ▪ Update section competence levels in the database (table *SectionCompetenceLevel*)
QuestionDataAccess	▪ Get array of questionStates from database (table *QuestionSolutionState*) ▪ Get questions details from database (table *Question* and *QuestionSolutionState*) ▪ Get related concept Ids from database (table *QuestionConceptRelation*) ▪ Update question solution state in the database (table *QuestionSolutionState*)
SMDataBaseCreation	• Create all the Student model database tables ▪ Create all the representation of knowledge base tables

3.8 System Working Processes

The classes in same or different packages have to communicate and co-operate with each other to perform all the system functions. The system working processes are described below.

3.8.1 Student Login Process

- The applet *LoginApplet* is loaded by accessing the URL of JSP file *Login.jsp* from a web browser. This applet presents a user interface for the student to input his/her name and password, and select the course ID that he/she would like to work on.

- When the student submits the entered data (name, password and courseID), the client-server connection is established via a socket between the applet *LoginApplet* and the servlet *CheckLoginServlet*.

- *CheckLoginServlet.java* checks if the received student information is correct. If it is correct, several data structures (*SectionTable, ConceptTable, QuestionTable, and Student)* are created and stored in an HTTP session managed by the servlet container Tomcat.

- The data structures are created via three layers of methods calls between the related classes as described in Figure 3-4.

3.8.2 Student Model Initialisation Process

- When a user logins to the system successfully, *CoursePage.jsp* are called to present an HTML frame with two parts.

- The left part presents a course structure interface produced by *ControlApplet* embedded in *Control.jsp*. The course structure interface consists of three layers of tabbed panels produced by classes *SectionViewTreePanel, ConceptListPanel,* and *RecommendedLinkedListPanel.* During the initialization of *ControlApplet*, an HTTP socket connection is established from *SectionViewTreePanel* to *InitialisationServlet* for retrieving the data needed to initialise the three tabbed panels.

- The right part presents *Content.html* first to show a simple message while loading the left part Applet. After loading the applet, the HTML page of the first section is presented.

3.8.3 Student Model Update Process

- *UpdateServlet* is called through an HTTP socket connection from *ControlApplet* to implement student model update logic whenever a student studies a section for a certain amount of time or change to another section.

- *UpdateThread* executes concurrently with *ControlApplet* for calculating study time.

- *ContentPageServlets* is called through an HTTP socket connection from *ControlApplet* whenever a student selects a new section. *ContentPageServlet* outputs the section content if the section is a test section. For a non-test section, *ContentPageServlet* calls the methods in *AdaptiveContentHandler* to convert the XML file to the adapted HTML output.

- *QuestionHandlerServlet* is called by submitting the HTML form that contains the answered questions. After updating the questions solution states and relevant concept and section competence levels, *ControlApplet* will be reloaded to present updated student competence levels.

- Whenever a concept link is selected from the concept list or any content page, *ConceptPageServlet* is called to present the concept details in a new window. *SolutionServlet* and *HintServlet* are called whenever the solution and hint buttons are pressed from a test section. The solution or hint is displayed in a new window.

- *PreferenceServlet* is called from *ControlApplet* whenever the preference button is pressed. It presents an HTML form in a new window for the student to specify his/her preferences etc. When the student submits this form, *PreferenceHandlerservlet* is called to update the student domain-independent data.

- When a student exits the main presentation HTML frame, *DataSaveServlet* is called for saving the student model data stored in the data structures back to the database.

Figure 3-4 shows the whole system working process described above.

3.9 Summary

This project is designed as a *distributed three-tier, client-server* architecture in which all the program classes are organised into different packages according to their functions. The package that deals with the client side interaction consists of the applet classes, JSP, and HTML files. The student model application package includes the classes for performing the student model update and adaptation logic. The servlets package is responsible for the client-server communication. The database access package is made up of the classes that facilitate the database operations for the application package.

Java is used as the only programming language in this project. Several Java and network related techniques are used, which include Applets, Java servlets, JDBC, JSP, XML, JavaScript and Tomcat.

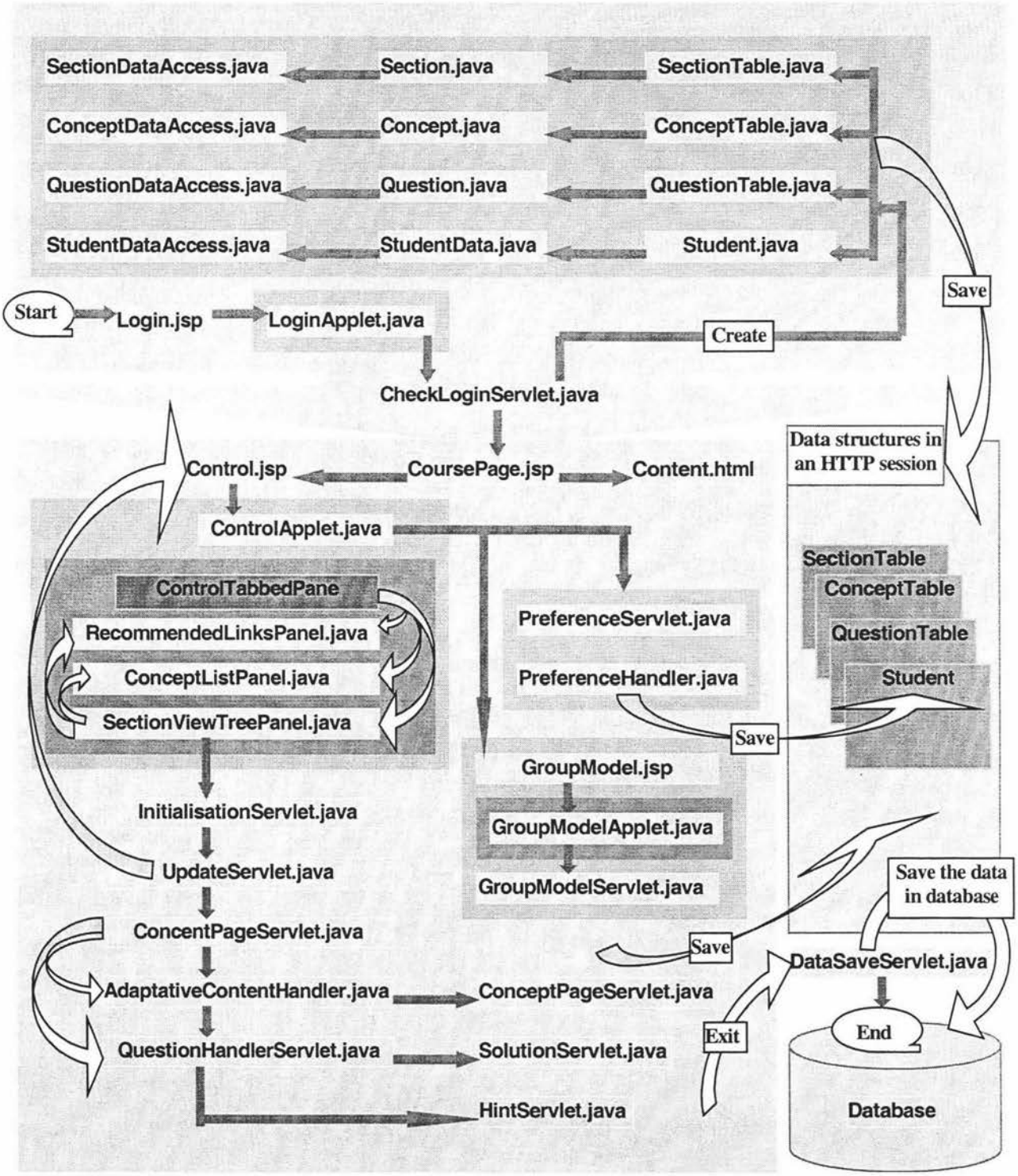

Figure 3-4 System working process

Chapter 4 System Database Design and Implementation

4.1 Introduction

This chapter presents the analysis and design of the knowledge representation database and student model databases. Both individual student model database and group student model database are discussed. The database logic views and schemas are based on Entity- Relationship Model.

4.2 Knowledge Representation Database

The knowledge base or domain model is needed for representing course contents and the relationship between the content elements. The knowledge representation database also forms the foundation of student model database.

4.2.1 Components of Knowledge Representation Database

The knowledge representation database consists of two main components: the content tree and the concept network.

4.2.1.1 Content Tree

The course materials are hierarchically organised into units of different levels as a *content tree,* which is similar to a table of content in a textbook as shown in Figure 4-1. The nodes in the content tree represent the chapters or sections in a textbook. Only the sections represented by the lowest level nodes, i.e. leaves, in the content tree contain the course contents (text, examples, questions, etc.), and are referred to as *content sections*. The sections associated with non-leaf nodes in the content tree only act as the references for the course structure, and are called *non-content sections*. The content sections are categorised into two types: *assessment section* and *learning section*. An *assessment section* is the content section that only holds a set of questions

for students to solve. A *learning section* is the content section that contains no questions but the other learning materials. A learning section may be connected to one or more concepts that are taught in the section. This *content tree* structure is displayed in the student interface along with graphic indicators for the competence levels of all the sections.

Each leaf node in the content tree is stored as a XML file that is converted to an HTML output, and then presented to the student whenever the unit is selected.

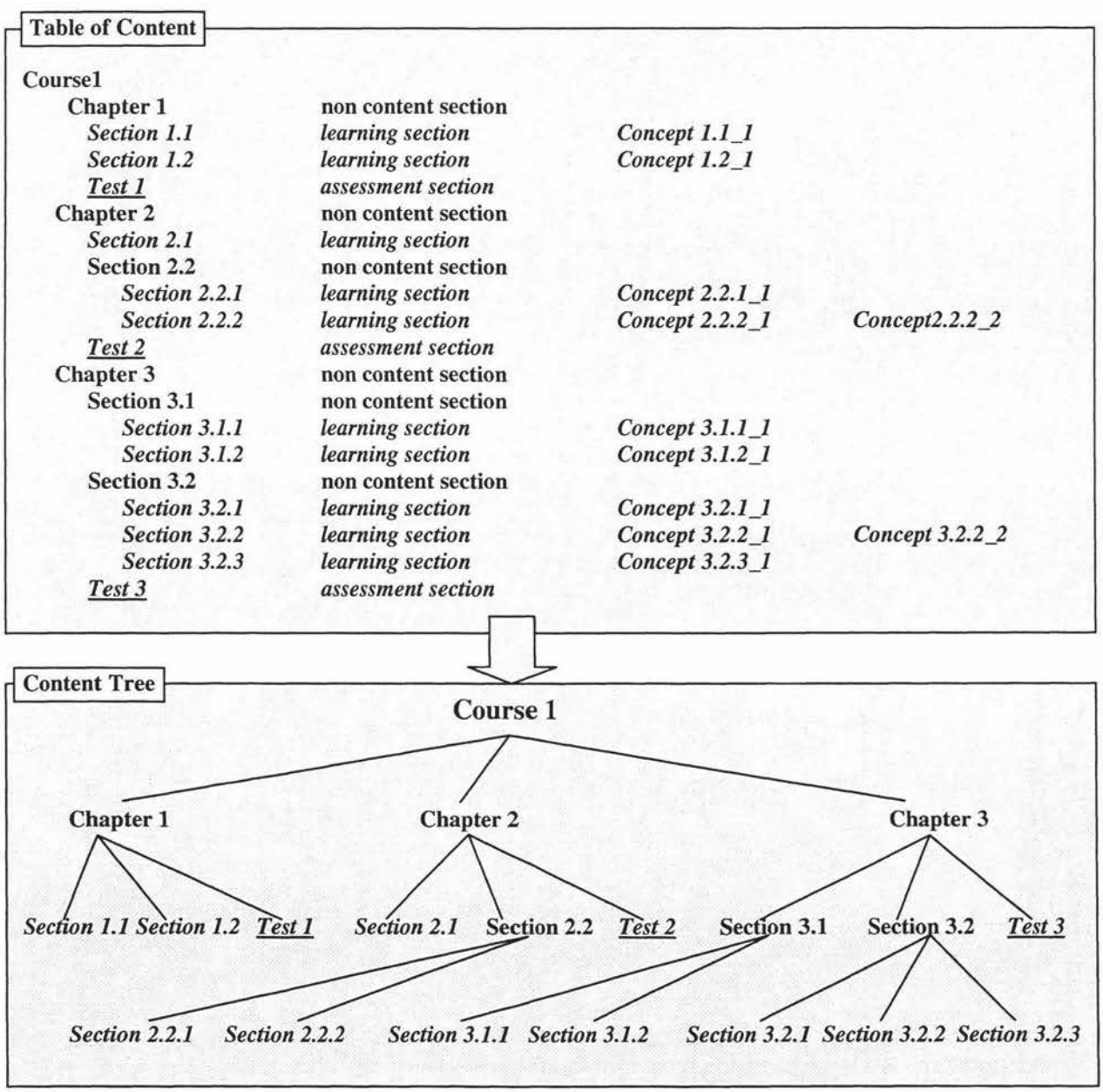

Figure 4-1 Example of a content tree

4.2.1.2 Concept Network

The *concept network* represents the pedagogical structure of the course materials. It is the backbone of the student model because whether a student has mastered the content of a course section is finally judged by if he/she mastered the concepts related to the section.

Concepts are linked together by prerequisite relationship to form the network as shown in Figure 4-2. *Concept network* provides the reference access to the course material. One or more concepts are associated with a learning section. A concept may be linked to a number of questions that are used to check if the student has mastered the concept. A bug catalogue (misconceptions, remediation methods, etc) may also be related to a concept. Figure 4-3 shows the relationships of sections, concepts, and questions.

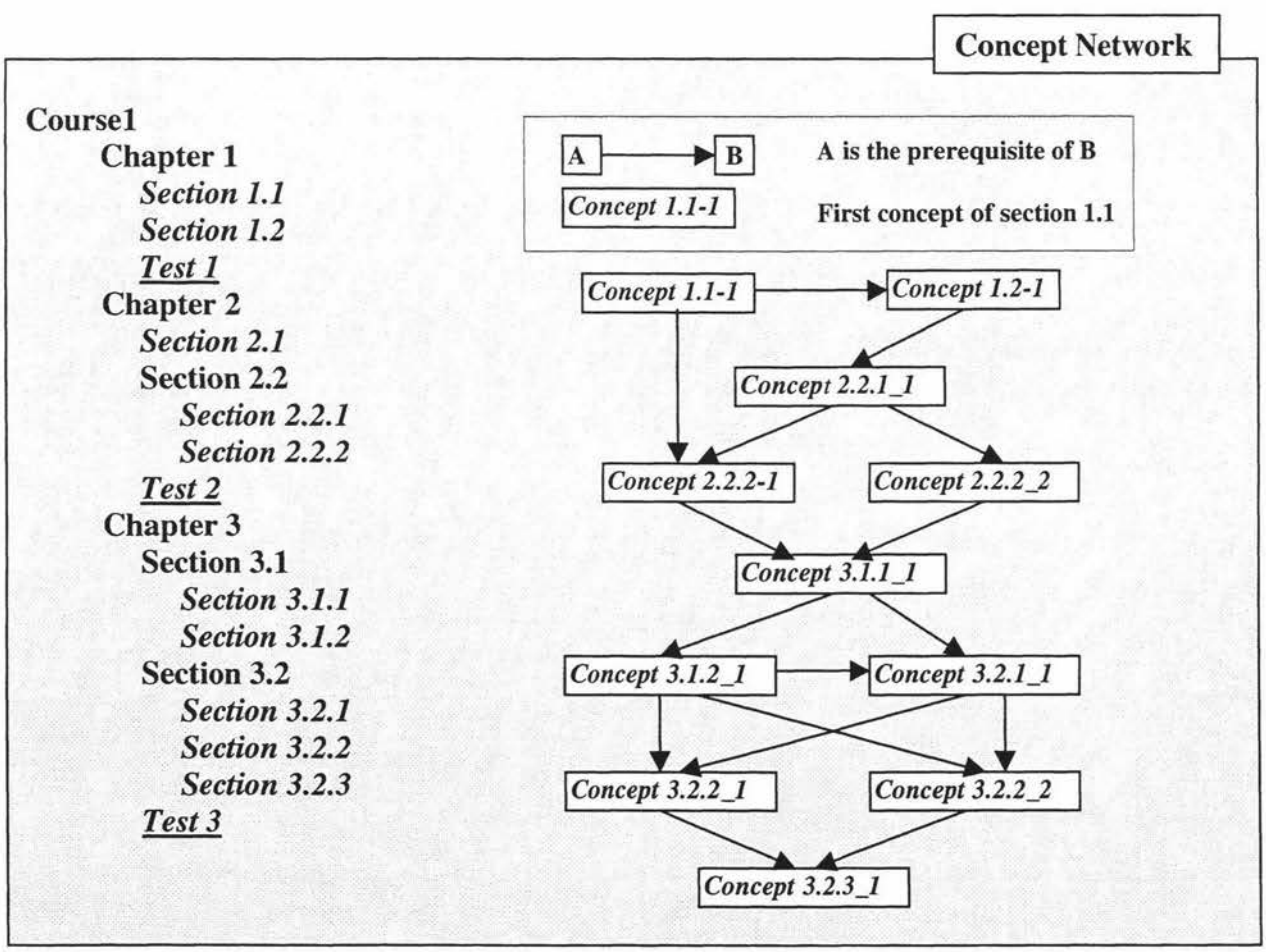

Figure 4-2 Example of concept network

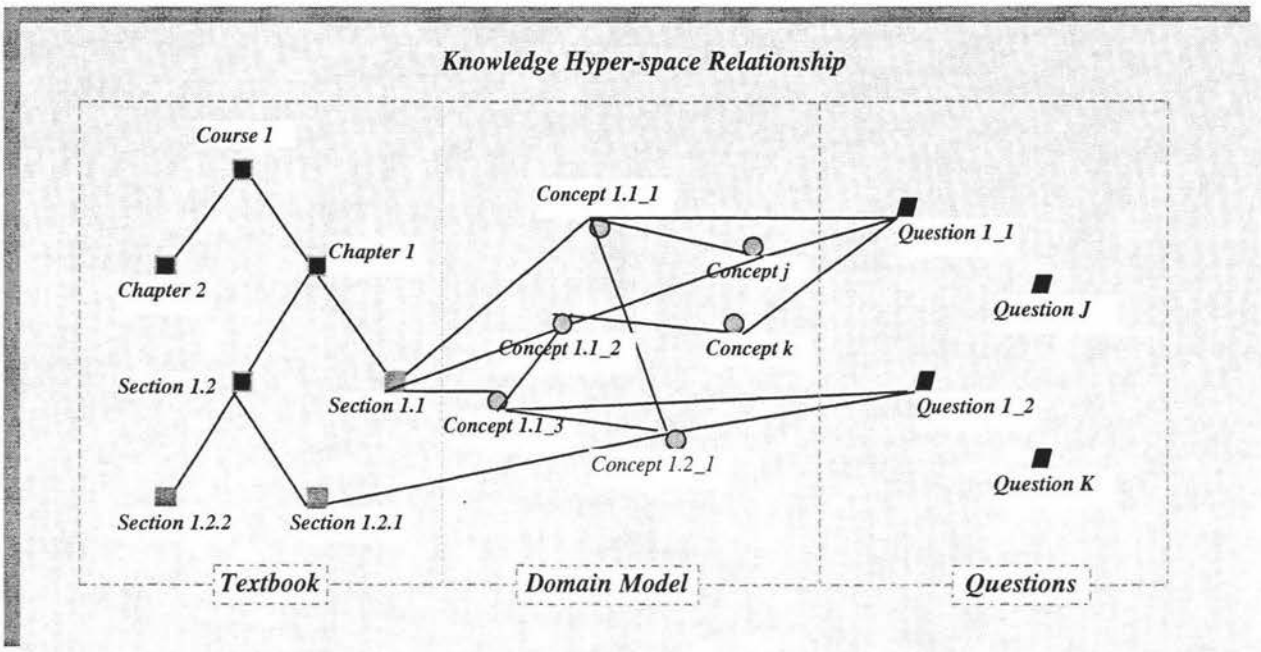

Figure 4-3 Example of section-concept-question relationship

4.2.2 Requirements for Knowledge Representation Database

Sections, concepts, and questions are the major components of the representation of knowledge base. The relationships of these components form the foundation of the *content tree* and *concept network.*

The requirements of a section unit are summarised as follows.

- A section can be a content section or non-content section
- A content section can be a assessment section or learning section
- Only a learning section may be related to concepts.
- A learning section may be related to none, one, or more concepts.
- A section may have none, one, or more prerequisite sections or courses.
- An assessment section has a number of questions.
- A content section must have a required study time (set by teacher at authoring time, the required study time for an assessment section is set to 0).

The requirements for a concept are described as follows.

- Each concept must be related to only one learning section.
- No concept should be related to a non-content section.
- Each concept may be related to none, one, or more questions.
- Each concept may have none, one, or more bug categories.
- Each concept may have none, one, or more prerequisite concepts.

The requirements for a question are described as follows.

- Each question must be related to one or more concepts
- Each question must have a answer
- Each question may have a solution
- Each question may have none or one hint

4.2.3 Design of Knowledge Representation Database

The knowledge representation database is designed by using *Entity-Relationship Model*. The logical view of the model is shown in Figure 4-4. The entities and their attributes are defined in Table 4-1.

4.3 Individual Student Model Database

The *individual student knowledge database* is represented by an *overlay model*, in which the current state of a student's knowledge level is described as a subset of the domain model.

Logical View of Knowledge Base

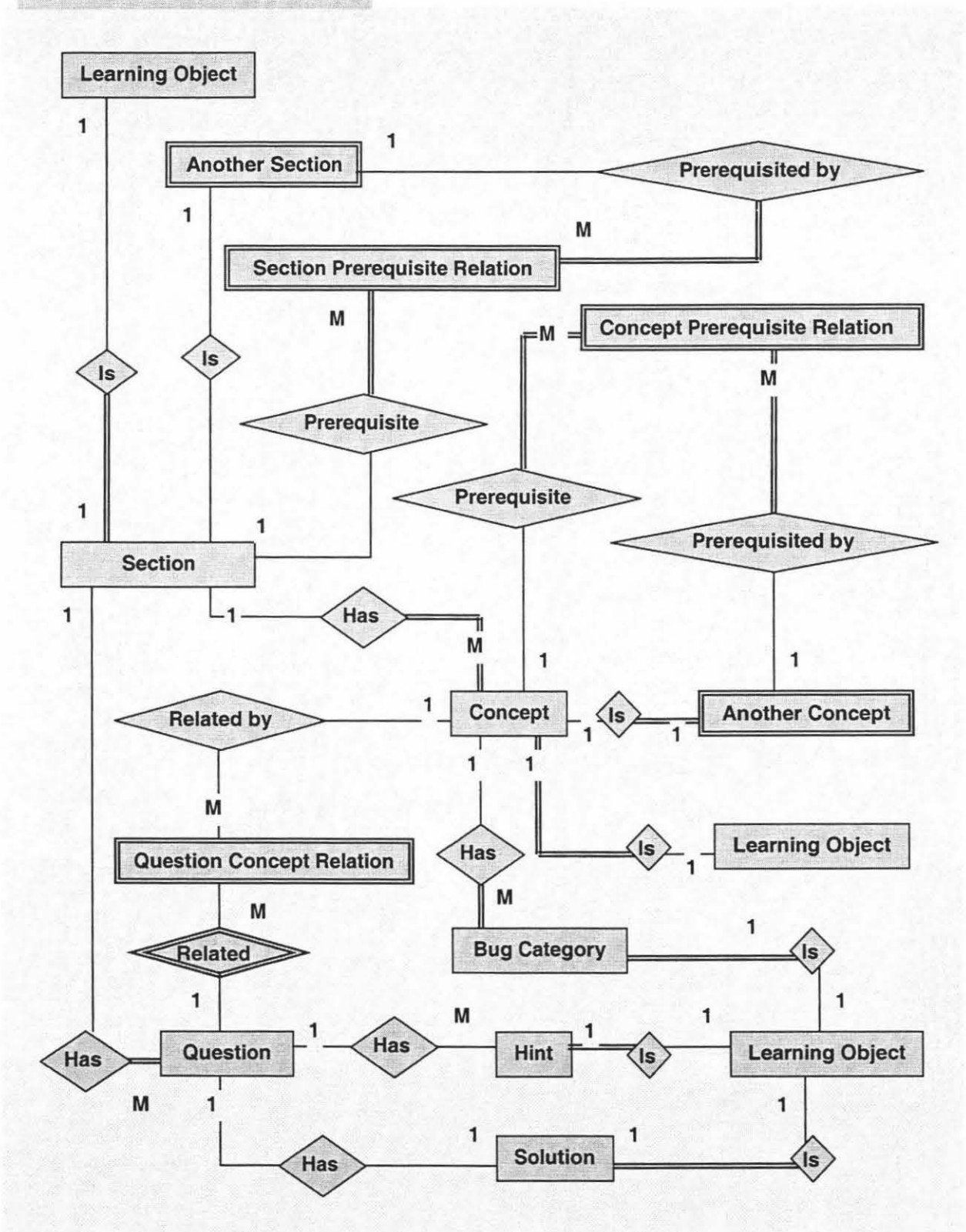

Figure 4-4 Logic view of the knowledge representation database

Table 4-1 Entities and their attributes of the knowledge representation database

Entity	Attribute	Data Type	Constraint	Null Value
Section	CourseID	CHAR(15)	Primary key	No
	SectionID	CHAR(15)	Primary key	No
	SectionTitle	CHAR(50)		No
	ParentSectionID	CHAR(15)		
	RequiredStudyTime	INERGER		
	Sequence	INTEGER		
	Type	INTEGER		No
	LearningObjectID	CHAR(20)		No
Concept	CourseID	CHAR(15)	Primary key	No
	SectionID	CHAR(15)	Primary key	No
	ConceptID	CHAR(20)	Primary key	No
	ConceptName	CHAR(50)		No
	LearningObjectID	CHAR(20)		No
	QuestionNumber	INTEGER		
Question	CourseID	CHAR(15)	Primary key	No
	SectionID	CHAR(15)	Primary key	No
	QuestionID	CHAR(15)	Primary key	No
	Answer	CHAR(20)		No
	SolutionObjectID	CHAR(20)		
	HintObjectID	CHAR(20)		
Section Prerequisite Relation	CourseID	CHAR(15)	Primary key	No
	SectionID	CHAR(15)	Primary key	No
	PrerequisiteCourseID	CHAR(15)	Primary key	
	PrerequisiteSectionID	CHAR(15)	Primary key	
Question Concept Relation	CourseID	CHAR(15)	Primary key	
	QuestionID	CHAR(15)	Primary key	
	ConceptID	CHAR(15)	Primary key	
Concept Prerequisite Relation	CourseID	CHAR(15)	Primary key	No
	ConceptID	CHAR(15)	Primary key	No
	PrerequisiteCourseID	CHAR(15)	Primary key	
	PrerequisiteConceptID	CHAR(15)	Primary key	
Bug Category	CourseID	CHAR(15)	Primary key	No
	ConceptID	CHAR(15)	Primary key	No
	BugCategoryID	CHAR(15)	Primary key	
	BugCategoryName	CHAR(30)		
	LearningObjectID	CHAR(20)		
Hint	CourseID	CHAR(15)	Primary key	No
	HintID	CHAR(15)	Primary key	No
	LearningObjectID	CHAR(15)		No
Solution	CourseID	CHAR(15)	Primary key	No
	SolutionID	CHAR(15)	Primary key	No
	LearningObjectID	CHAR(15)		No

4.3.1 Components of Individual Student Model Database

The individual student model database consists of two main components: and *domain-independent data* and *domain-specific data*.

4.3.1.1 Domain Independent Data

The *domain-independent data* includes student personal information, background, experience, goals, and preferences of learning style.

- **Student's Study Goal**

A student can have one or both of the following study goals (represented by integers 1 and 2) in the actual database.

- *Overview* -- the study goal of the student is to have the overview of the subject domain.
- *In-depth* – the study goal of the student is to learn the course materials in depth.

- **Student Non-Domain Related Experiences**

A student can have one of four types of non-domain related experiences (represented by integers 1 to 4 in the actual database).

- *None* -- the student does not have any experience of working with similar systems at all.
- *A-little* -- the student has a little experience of working with similar systems.
- *Moderate* -- the student has some experience of working with similar systems.
- *A-lot* -- the student has a lot of experience of working with similar systems.

- **Student Learning Style**

A student can prefer one or both of the following learning styles (represented by integers 1 and 2 in the actual database).

- *Concrete* – the student prefers concrete experience to abstract conceptualisation, prefers interactive activities to explanatory descriptions and examples, and prefers active experimentation to reflective observation.

- *Abstract* -- the student prefers abstract conceptualisation to concrete experience, prefers explanatory descriptions and examples to interactive activities, and prefers reflective observation to active experimentation.

A student's presentation preferences on multi-media objects are divided into following two groups and are represented by integers 1 and 2 in the actual database.

- **Student Text-Video Preference**

- *Text* – the student prefers textual presentation to visual presentation.
- *Video* – the student prefers visual presentation to textual presentation.

- **Student Text-Audio Preference**

- *Text* -- the student prefers textual presentation to audio presentation.
- *Audio* -- the student prefers audio presentation to textual presentation.

4.3.1.2 Domain Specific Data

The *domain-specific data* are summarised as follow.

- **Student Study Time for Content Sections in Content Tree**

The student study time for a content section is the total time that a student spent on the content section.

- **Student Competence Levels for Sections in Content Tree**

A student can have one of following five "section competence levels" represented by integer 1-5 in the actual database.

- *Not-ready* -- the section is not ready to be learned yet because necessary prerequisite is not meet by the student.
- *Ready* -- the section is ready to be learned because the student has studied all of its prerequisites.
- *Visited* -- the section or one or more of its sub-sections have been visited by the student for a short period of time (set by teacher at authoring time).
- *Learned* -- the section has been studied for certain amount of time (longer than the required study time) or all of its sub-sections are learned or mastered by the student.
- *Mastered* -- the content of this section has been mastered by the student.

- **Student Competence Levels for Concepts in Concept Network**

A student can have one of following two "concept competence levels" (represented by integers 1 and 2 in the actual database).

- *Not-mastered* -- the concept has not been mastered by the student.
- *Mastered* -- the concept has been mastered (if a concept is mastered, all of its prerequisite concepts are mastered).

- **Student's Question Solving Status for the Questions in a Course**

A student can have one of following five "question solving status" (represented by integers 1 to 5 in the actual database).

- *Unsolved* -- the student has not tried to answer this question.
- *Unsolved-but-studied-hint* -- the student has studied the hint of the question, but has not tried to answer this question.
- *Solved* -- the student has answered the question correctly without studying hint.
- *Solved-with-help-of-hint* -- the student has answered the question correctly with the help of hint.
- *Failed-or-studied-solution* -- the student provided the incorrect answer, or the student studied the solution before answering the question.

- **Student's Overall Competence Levels for a Course**

A student can have one of following three "overall competence levels" (represented by integers 1 to 3 in the actual database).

- *Beginning* -- the student is a beginner to this subject domain.
- *Intermediate* -- the student has some basic knowledge of this subject domain.
- *Advanced* -- the student has advanced knowledge of this subject domain.

Table 4-2 describes all the integer attributes that define the entities in the student model database.

4.3.2 Design of Student Model Database

Based on above analysis, the student model database is designed by *an Entity-Relationship Model*. The logical view of student model database is shown in Figure 4-5. The schema and contents of individual tables are presented in Table 4-3.

Table 4-2 Integer attributes specification of student model database

Entity	Attribute	Description	Value	Meaning	Default
Domain Independent Data	StudyGoals	The student's purpose of learning the course	1 2 3	Overview In-depth Both	1
	Experience	The student's non-domain related experience	1 2 3 4	None Little Moderate A lot	1
	VTPreference	The student's preference on video or text presentation	1 2 3	Prefer Video Prefer Text Prefer Both	2
	ATPreference	The student's preference on audio or text presentation	1 2 3	Prefer Audio Prefer Text Prefer Both	2
	CAStyle	The student's learning style is concrete or abstract	1 2 3	Concrete Abstract Both	1
Overall Competence Level	OverallCompetenceLevel	The student's competence level of this particular course	1 2 3	Beginning Intermediate Advanced	1
Section Study Time	SectionStudyTime	The student's actual study time on this particular section	Integer	Seconds used	0
Section Competence Level	SectionCompetenceLevel	The student's competence level of this particular section	1 2 3 4 5	Not-ready Ready Visited Learned Mastered	1
Concept Competence Level	ConceptCompetenceLevel	The student's competence level of this particular concept	1 2	Not-mastered Mastered	1
Question Solving Status	QuestionSolvingStatus	The student's solution state of the particular question	1 2 3 4 5	Unsolved Solved-with-hint Solved Unsolved-with-hint Failed	1

Logical view of the student model database

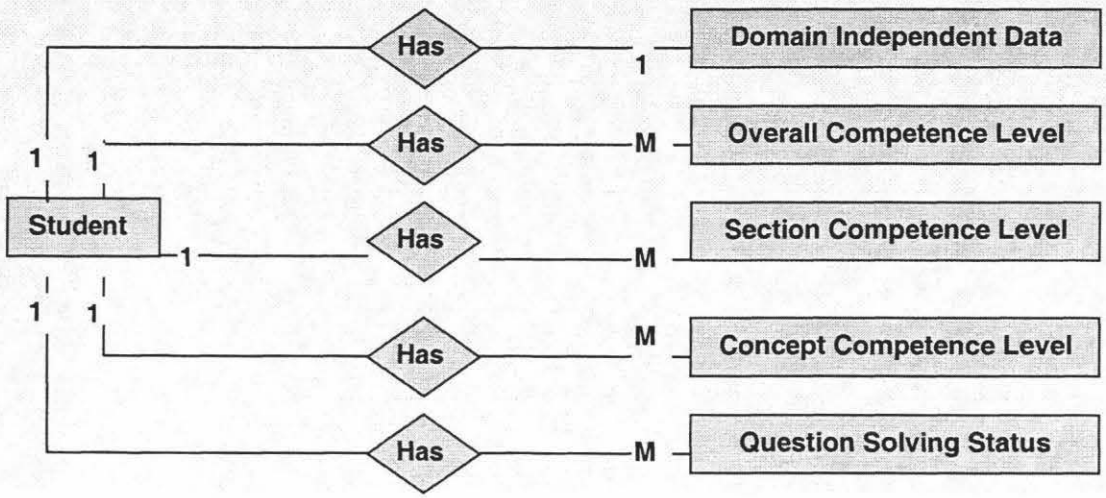

Figure 4-5 Logic view of student model database

Table 4-3 Entities and their attributes of the student model database

Entity	Attribute	Date type	Constraint	Null value
Domain Independent Data	StudentID	CHAR(20)	Primary key	No
	StudentPW	CHAR(20)	Primary key	No
	StudyGoals	INTEGER		No
	Experience	INTEGER		No
	VTPreference	INTEGER		No
	ATPreference	INTEGER		No
	CAStyle	INTEGER		
Overall Competence Level	StudentID	CHAR(15)	Primary key	No
	CourseID	CHAR(15)	Primary key	No
	OverallCompetenceLevel	INTEGER		No
Section Competence Level	StudentID	CHAR(15)	Primary key	No
	CourseID	CHAR(15)	Primary key	No
	SectionID	CHAR(15)	Primary key	No
	SectionCompetenceLevel	INTEGER		No
	StudyTime	INTEGER		No
Concept Competence Level	StudentID	CHAR(20)	Primary key	No
	CourseID	CHAR(15)	Primary key	No
	ConceptID	CHAR(20)	Primary key	No
	ConceptCompetenceLevel	INTEGER		No
Question Competence Level	StudentID	CHAR(20)	Primary key	No
	CourseID	CHAR(15)	Primary key	No
	QuestionID	CHAR(20)	Primary key	No
	QusestionSolvingStatus	INTEGER		No

4.3.3 Referential Constraints

In student model database, all primary keys are foreign keys from the database for knowledge representation. These foreign keys link child entity instances in the student model database to the parent entity instances in the database for knowledge representation. The "cascade" operation is performed in case of deletion of parent entity instances, i.e., if a parent entity instance is deleted, and the parent entity contains the key value referenced by the corresponding foreign keys in the child entity instances, all related child entity instances will be automatically deleted.

4.4 Group Student Database

The previous description on student model database was for an individual student. The database for a group student model is constructed by averaging corresponding values in the models for the individual students within a group. The group student model data obtained from previous student groups may be used to initialise current individual student models if same specific information is not reliably available for that student.

The *group student data* includes the statistical results of the individual student competence levels on concepts and sections, and the solving status of questions. These statistics are produced dynamically from individual student models. The group student data is presented in the percentage form in several tables described as follows.

- **Table of Section Competence Levels**

- *Group student competence levels for each section* -- Indicate how many percent of the students in a group have the competence level of not-ready, ready, visited, learned or mastered for each section.
- *Group student competence levels for all the course sections* -- Indicate how many percent of all the sections in a course have been mastered by the students as a group.

- **Table of Concept Competence Levels**

- *Group student competence levels for each concept* -- Indicate how many percent of the students in a group have the competence level of not-mastered or mastered for each concept.
- *Group student competence levels for all the course concepts* -- Indicate how many percent of all the concepts in a course have been mastered by the students as a group.

- **Table of Question Solution Status**

- *Group student competence levels for each question* -- Indicate how many percent of the students in a group have the solving status of unsolved, unsolved-but-studied-hint, solved, solved-with-help-of-hint or failed-or-studied-solution for each question.
- *Group student competence levels for all the course questions* -- Indicate how many percent of all the questions in a course have been solved successfully by the students as a group.

4.5 Summary

This system includes both *Knowledge Representation Database* and *Student Model Database.* The knowledge representation database, which includes the content tree and concept tree, provides the basic structure and content of a course. It is also the foundation of individual student model database that consists of domain independent and domain specific information used for modelling the learning behaviours of individual student. The information for a group student model is dynamically extracted from the models of individual students in the group.

Chapter 5 Student Modelling in Web-Based Learning Systems

5.1 Introduction

In this web-based learning system both individual and group student modelling are included. This chapter first describes the initialisation and update of individual student model, then discuss the processes of group student modelling.

5.2 Initialisation of Individual Student Model

After receiving a student login message from the client side, the servlet in the server side performs ID verification, and then loads the student model information and knowledge base information from the databases to memory. The loaded knowledge base data mainly include course content tree and concept network. As described in chapter 4, the information contained in the individual student model is classified as domain-independent and domain specific. Initialisation of the two types of student model data is described as follows.

5.2.1 Initialisation of Student Domain-Independent Information

The domain independent information are student study goal, non-domain related experience, learning style, and preferences of multi-media objects. These data are initialised by explicit questioning. When a student accesses the system for the first time, a questionnaire window (Figure 5-1) is presented to the student to specify their domain independent information. If the student chooses not to specify a certain data, the default value will be used for the data. The default values for domain independent data were summarised in the previous chapter (Table 4-2).

5.2.2 Initialisation of Student Domain-Specific Information

Most domain-specific data are initialised using default values. The initialisation processes are described as follows.

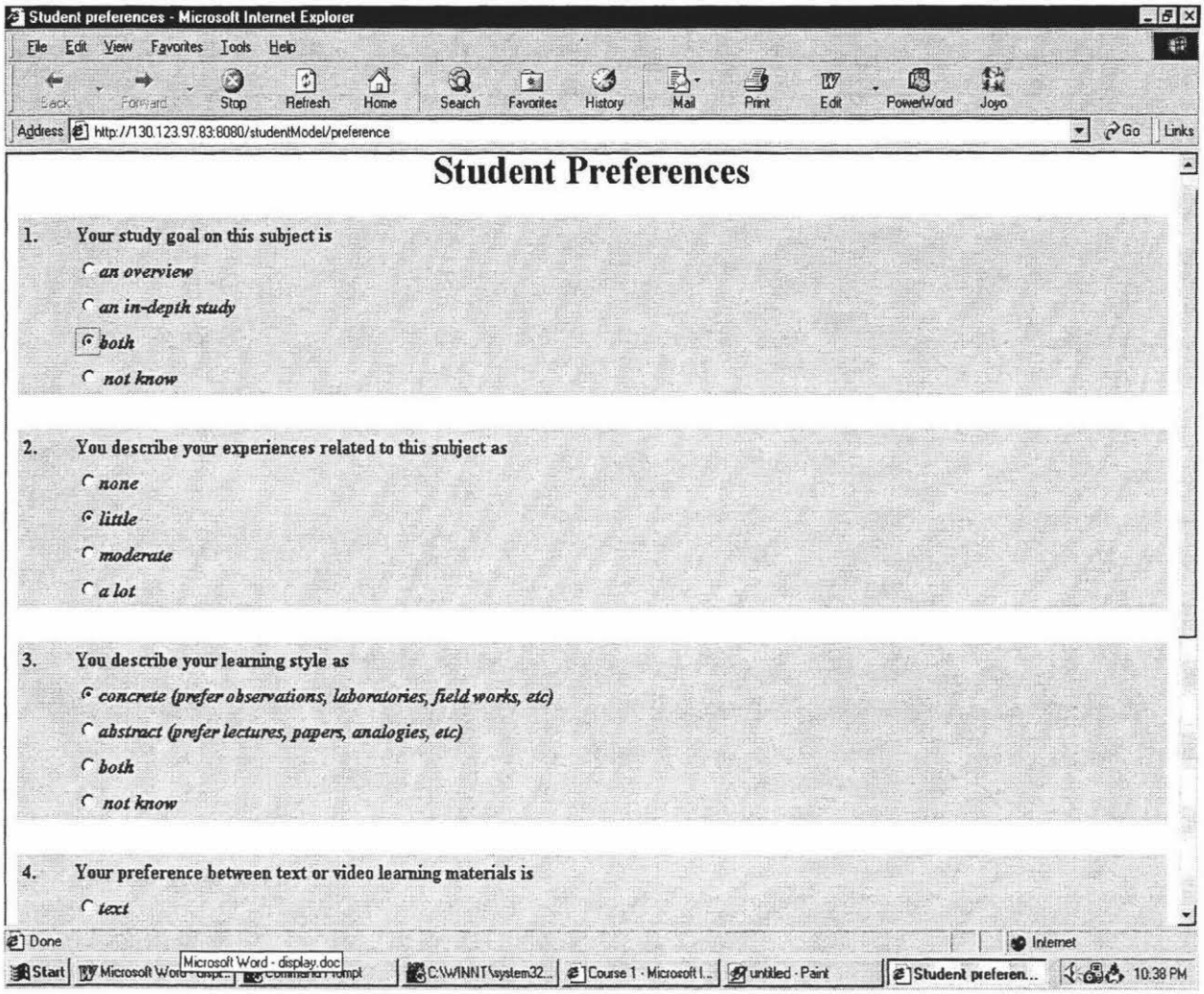

Figure 5-1 Student interface of questionnaire

- **Student Study Times for Content Sections in Content Tree**

Initial value for the student study time for a content section is set as 0.

- **Student Competence Levels for Sections in Content Tree**

The initial competence level of a section is set to "*ready*", if the section satisfies the following conditions:

- The section does not have any prerequisite section, or all its prerequisite sections do not belong to the current course.
- The section does not have any related concepts whose prerequisite concepts are "*not-mastered*".

The initial competence levels of the sections that do not meet above requirements are set to "*not-ready*".

- **Student Competence Levels for Concepts in Concept Network**

The initial competence levels of all concepts are set to "*not-mastered*".

- **Student's Question Solving Status for the Questions in a Course**

The initial question solving statuses of all questions are set to "*unsolved*".

- **Student's Overall Competence Level for a Course**

By default, the initial overall competence level is set to "*beginning*". It may also be decided by checking the student grades of prerequisite courses, if the information is available.

5.3 Update of Individual Student Model

5.3.1 Update of Student Domain-Independent Information

Student can modify his/her domain-independent data from the questionnaire window at any time. By clicking the "Preferences" button on the main interface, the student can open the questionnaire window, and then the student can select new values to reset the domain-independent data. When the student submits his or her questionnaire, the student record in the server-side is changed according to the submitted values, and the control applet is reloaded. As a

result, the course materials are presented according to the newly specified preferences, learning style, and study goal.

5.3.2 Update of Student Domain-Specific Information

The student competence levels for sections and concepts are updated by tracking student study time in a section and student problem solving behaviours. The details of updating processes are described as follows.

- **Update of Student Study Times for Content Sections in Content tree**

- The student study time for a content section is update when the student switches to another content section. The time that the student spent on the content section is added to the previous total study time to get a new total study time.

- **Update of Question Solving Statues**

- Based on a student's answer to a question and whether the student studied the hint and/or solution to the question, the solution status of a question may be updated to *"unsolved-but-studied-hint", "solved", "solved-with-help-of-hint",* and *"failed-or-studied-solution"* from the initial value *"unsolved"*. The exact meanings of these values are defined in the previous chapter.

- **Update of Concept Competence Levels**

- The initial status of a concept competence level is *"not-mastered"*.

- For a concept that is related to one or more questions, the concept competence level is updated by student's performance on solving the concept-related questions. Each of these concepts has a teacher-customisable parameter that defines the number of questions the student should solve correctly before that concept is treated as "mastered". The concept

competence level is changed to "*mastered*" from the initial status "*not-mastered*' if the student has obtained the score represented by the parameter.

- A discount coefficient, which is also set by the teacher, is applied if the student solved a question by studying its hint, so the student needs to solve more questions in order to get the required score for mastering a concept.

- For a concept without any related question, its competence level is set as "*mastered*" after its related section has been studied for over the required study time for the section.

- Each concept may have one or more prerequisite concepts. If a concepts has a "*mastered*" competence level, the competence levels of all its prerequisite concepts are set as "*mastered*" even the student has not solved enough questions.

- **Update of the Competence Levels of Learning Sections**

The student competence level on a learning section in the content tree is decided by the student study time and the competence levels of the section-related concepts.

- The initial state of a learning section competence level is "*not-ready*" or "*ready*".
- After a student studies a learning section for a short period of time, which is a time value set by teacher at authoring time to ensure that the student really starts to study the section instead of just browsing, the competence level of the section is set as "*visited*".
- After a student studies a learning section for over the required study time for the section, which is predefined by the teacher, the competence level of the section is set as "*learned*".
- For a "*not-ready*" learning section, if the competence levels of all its prerequisite sections are "*learned*" or "*mastered*", the competence level of the section is set as "*ready*".
- For a learning section related to one or more concepts, the competence level of the section is set as "*mastered*" if the competence levels of all its related concepts are "*mastered*".
- For a learning section that does not have any related concept, a "*learned*" section becomes a "*mastered*" section immediately.

- **Update of the Competence Levels of Assessment Sections**

- For a "*not-ready*" assessment section, if the competence levels of any of its prerequisite sections are "*learned*" or "*mastered*", the competence level of the assessment section is set as "*ready*".
- If the competence levels of all the concepts related to all the questions in the assessment section are "*mastered*", the competence level of the assessment section is set as "*mastered*".

- **Update of the Competence Levels of Non-Content Sections**

The competence level of a non-content section is determined by the competence levels of its child sections in the content tree. No concept is related to a non-content section.

- The initial state of a non-content section competence level is "*not-ready*" or "*ready*".
- If the competence levels of any its child sections become "*ready*", the competence level of the non-content section is set to "*ready*".
- If the competence levels of any its child sections become "*visited*", the competence level of the non-content section is set to "*visited*".
- If the competence levels of all its child sections become "*learned*" or "*mastered*", the competence level of the non-content section is set to "*learned*".
- If the competence levels of all its child sections become "*mastered*", the competence level of the non-content section is set to "*mastered*".

Figure 5-2 shows part of the student competence levels update process in the student model.

- **Update of Overall Competence Level**

- If the competence levels of less than 20% of concepts in the course are "*mastered*", the overall competence level is set to "*beginning*".

- If the competence levels of more than 20% and less than 50% of concepts in the course are "*mastered*", the overall competence level is set to "*intermediate*".

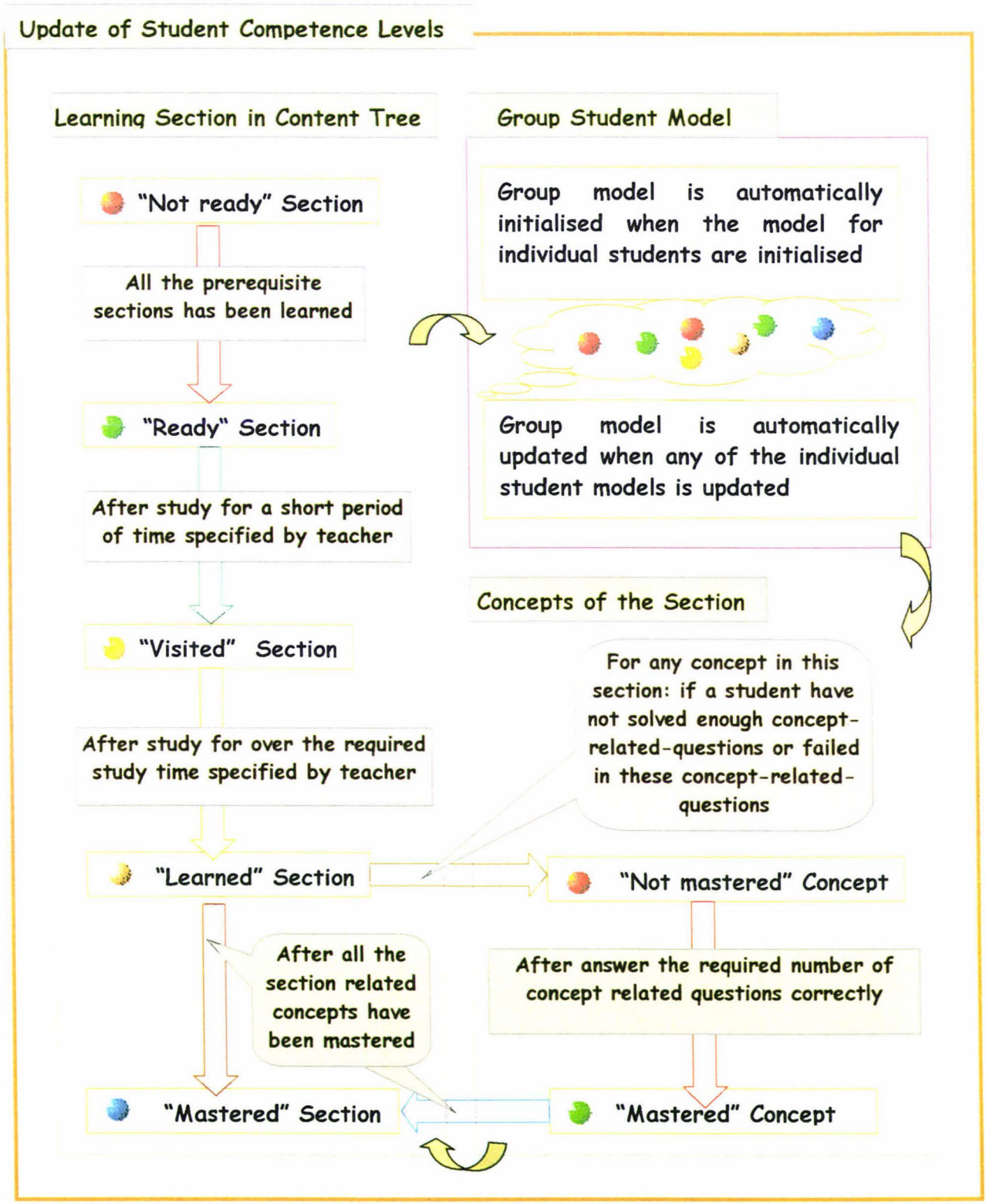

Figure 5-2 Student competence levels update process

- If the competence levels of more than 50% of concepts in the course are "*mastered*", the overall competence level is set to "*advanced*".

5.4 Update Algorithms of Individual Student Model

The algorithms for updating competence levels are described in the following program structures.

- **Algorithms for Updating Section Competence Level**

```
//update content section competence level according to study time
Calculate the content section study time
if      content section study time >= required study time
        The section competence level is set to "learned"
if      0<content section study time < required study time
        The section competence level is set to "visited"

//update section-related concepts when the concepts have no related questions

if      the content section competence level is "learned"
        if      section-related question number is equal to "Null" (a non-test section)
                for     each related concepts of this content section
                        if      the concept related question is equal to Null
                                The concept competence level is set to "mastered";

//update the section competence level when the section is an assessment section

if      the content section related question number is not equal to "Null" (an assessment section)
        for     each question
                for     for each question related concept
                        if      the concept competence level is not equal to "mastered"
                        then    the section competence level is set to "visited";
                        else    the section competence level is set to "mastered";
```

```
//update the section competence level when the section is not an assessment section

if      the section related question number is  equal to "Null" (a learning section)
        if      this section has child sections
                if      the section required study time is equal to "0" (a non-content section)
                        and the competence level of any of its child sections  is equal to "visited" or
                        "learned" or "mastered"
                then    the section competence level is set to "visited";
                else if the section required study time is equal to"0" (a non-content section)
                        and the competence levels of  all of its child sections  are equal to "learned" or
                        "mastered"
                then    the section competence level is set to "learned";
                else if the section required study time is equal to "0" (a non-content section)
                        and the competence levels of  all of its child sections are equal to "mastered"
                then    the section competence level is set to "mastered";
                else if the section required study time is not equal to "0" (a content section)
                        if  the competence levels of the section related concepts are equal to  "mastered"
                        then the section competence level is set to "mastered";


//update the succeed sections

if      the section competence level is equal to "mastered"
        for     each of its succeed section
                check if  the section competence level is "ready";
if      the section competence level is equal to "learned"
        for     each of sibling section
                check if  the section competence level is "ready";


//update parents section

if      the type of parent section is not equal to "top" or  "Null"
                update  the competence level of the parent section;
```

- **Algorithms for Checking if a Section is Ready to Learn**

```
if      the section competence level is not equal to "not ready"
        return;
if      the section is not a content section
        for     all the children of this section
                if      the competence level of  any of its child sections  is equal to "ready "
                then    the section competence level is set to "ready"
if      the number of prerequisite sections is not equal to "Null"
        if      the section related question number is  equal to "Null" (a learning section)
                if      the competence levels of all of its prerequisite section are equal to "mastered" or
                        "learned"
                then    the section competence level is set to "ready"
        else if the section related question number is  not equal to "Null" (an assessment section)
                if      the competence level of any of its prerequisite section is equal to "mastered" or
                        "learned"
                then    the section competence level is set to "ready"
```

- **Algorithms for Updating Concept Competence Level**

```
if      the concept competence level is equal to "mastered";
then    Return;
//check  succeed concepts
if      any of the succeed concepts' competence level is equal to "mastered"
then    the concept competence level is set to "mastered";
//check states of the related questions
else
        for     all related questions
                if      the question is correct answered without help of hint
                then    the score of the concept plus one;
                else if the question is correct answered with help of hint
                then    the score of the concept plus the "HintCofficient";
        if      score >=  the required score of the concept
        then    the concept competence level is set to "mastered";
```

//update related concept and sections

if the concept competence level changed to "mastered"

then update the competence levels of all of its related sections;
update the competence levels of all of its prerequisite concepts (repeat above process) ;
check if ready-to-learn for the sections which are related to succeed concepts

5.5 Initialisation and Update of the Group Student Model

As described in the Section 4-4, the information in the group student model is dynamically extracted from the models of individual student in the group. When the model for individual students are initialised, the group model is automatically initialised. Similarly the group model is automatically updated when any of the individual student models is updated. Therefore, no specific initialisation and update processes are needed for the group student model.

5.6 Summary

This chapter presented the processes of student model initialisation and update. The domain independent information of individual student model is initialised and updated using a questionnaire, in which a student can specify or modify his/her study goal, experience, learning style, and preferences for multi-media objects. The individual student competence level on each concept is updated according to his/her performances on solving the concept-related questions. The individual student competence level on a content learning section is decided by the study time and the competence levels of its related concepts. The individual student competence level of a non-content section is set based on the competence levels of its child sections. As a group student model is dynamically constructed from the individual student models, the group student model is automatically initialised or updated after initialisation and update of the individual student models.

Chapter 6 Usage of Student Model in System Adaptation

6.1 Introduction

This chapter describes the adaptations provided in this system based on the prototype student model. It then analyses how the adaptation methods and techniques are used in this system, and presents the actual student interfaces used to demonstrate the adaptation.

6.2 Usage of Student Model

The prototype student model designed in this project is used to provide adaptations to students. Two types of adaptation are implemented in the system: *navigation adaptation* that mainly decides the sequence of the presentation, and *content adaptation* that involves selectively presenting learning materials in various hypermedia forms.

The *domain independent information* in student model supports content adaptation. The system is able to create an individualised content presentation based on the student experience, study goals, learning styles, and preferences of multimedia objects. The *domain specific information* in student model supports both content and navigation adaptation. The competence levels of the sections in the content tree are used to decide the recommended learning paths.

6.3 Navigation Adaptation

Navigation adaptation is achieved via list of recommended links, graphic presentations of section and concept competence levels, and cross-reference links. These methods implemented a number of navigation adaptation techniques, which include *direct guidance, sorting, hiding,* and *annotation.*

6.3.1 List of Recommended Links

Based on the section competence levels, the system generates a *list of recommend links*, which includes a number of sections that have not been learned and are most suitable to be studied after the current content section in the content tree (Figure 6-1).

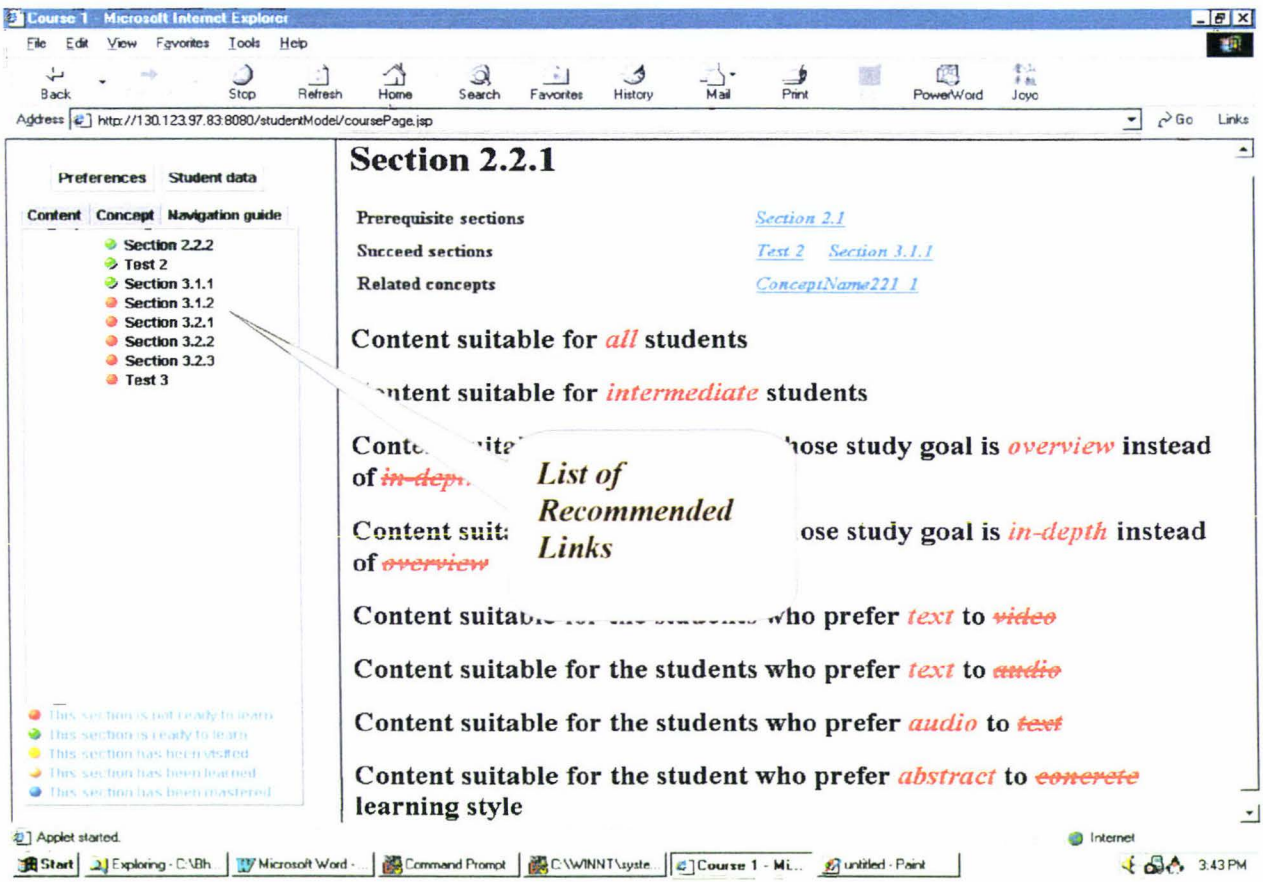

Figure 6-1 Student interface of list of recommended links (right panel shows what kind of content will display in actual learning system)

The list of recommended links is arranged according to the following rules:

- The sections are initially listed according to their competence levels: "*ready*" content sections come first, "*visited*" content sections second, "*not-ready*" content sections last. The non-content sections and "*learned*" and "*mastered*" content sections are not listed.

- For the sections with the same competence levels, the positions of the sections in the content table are used to decide the list order.

List of Recommended Links implement the following adaptation methods and techniques:

- **Direct guidance** is implemented by directly presenting all of the recommended section links and directly telling the student in which order these sections should be learned.
  - **Global guidance** is provided by suggesting the student a group of browsing steps within the context of the all teaching materials after each selection.
  - **Local guidance** is provided by suggesting the most relevant section link to follow from the current node.

- **Adaptive sorting of links** is implemented by displaying an ordered list of section links sorted according to their relevance to the current section node and their competence levels.
  - **Global guidance** is provided by helping the student to find the shortest way to the most relevant "*ready*" sections among all the teaching materials with minimal floundering.
  - **Local guidance** is provided by suggesting the student the first selection of all the recommended links.

- **Hiding** is implemented 1) by keeping visible only those links that are suitable to the student current knowledge level and hiding the other irrelevant links from the student. (for example, the links to mastered sections are hidden by this way) and 2) by limiting the length of the list (for example, at most display the first ten recommended links).
  - **Global orientation support** is provided by simplifying both orientation and learning with the reduced size of the recommended linked list.
  - **Local orientation support** is provided by decreasing the student cognitive load and making him/her concentrate on relevant links with the limited number of navigation opportunities.
  - **Additional support for novices** is provided by not showing recommended links until the student set the current task more exactly by selecting a content section in the course hierarchy.

- **Annotation** is implemented by using coloured balls along the section titles to annotate different competence levels of the links (different colours represent different competence levels as shown in the lower part of left panel in Figure 6.1).
- **Global orientation support** is provided by keeping the colour of a section link the same as its colour in the content tree.
- **Local orientation support** is provided by informing the student if a link can be learned immediately.

6.3.2 Graphic Presentation of Section Competence Level

The competence levels of sections are illustrated with coloured balls in the content tree (Figure 6-2). The meanings of the coloured balls are explained under the content tree (Figure 6-2). These presentations give clear indications on student learning directions.

Graphic Presentation of Section Competence Level implements the following adaptation methods and techniques:

- **Annotation** is implemented by using five colored balls to represent five section competence levels. Different color represents different competence level as shown in Figure 6-2.
- **Global orientation support** is provided by keeping the color of a section the same as its color in the recommended linked list.
- **Local orientation support** is provided by informing the student the current competence levels of all the sections in a course.

6.3.3 Graphic Presentation of Concept Competence Level

The competence levels of concepts are illustrated with coloured balls in concept list. The presentations give clear indications on student knowledge level (Figure 6-3).

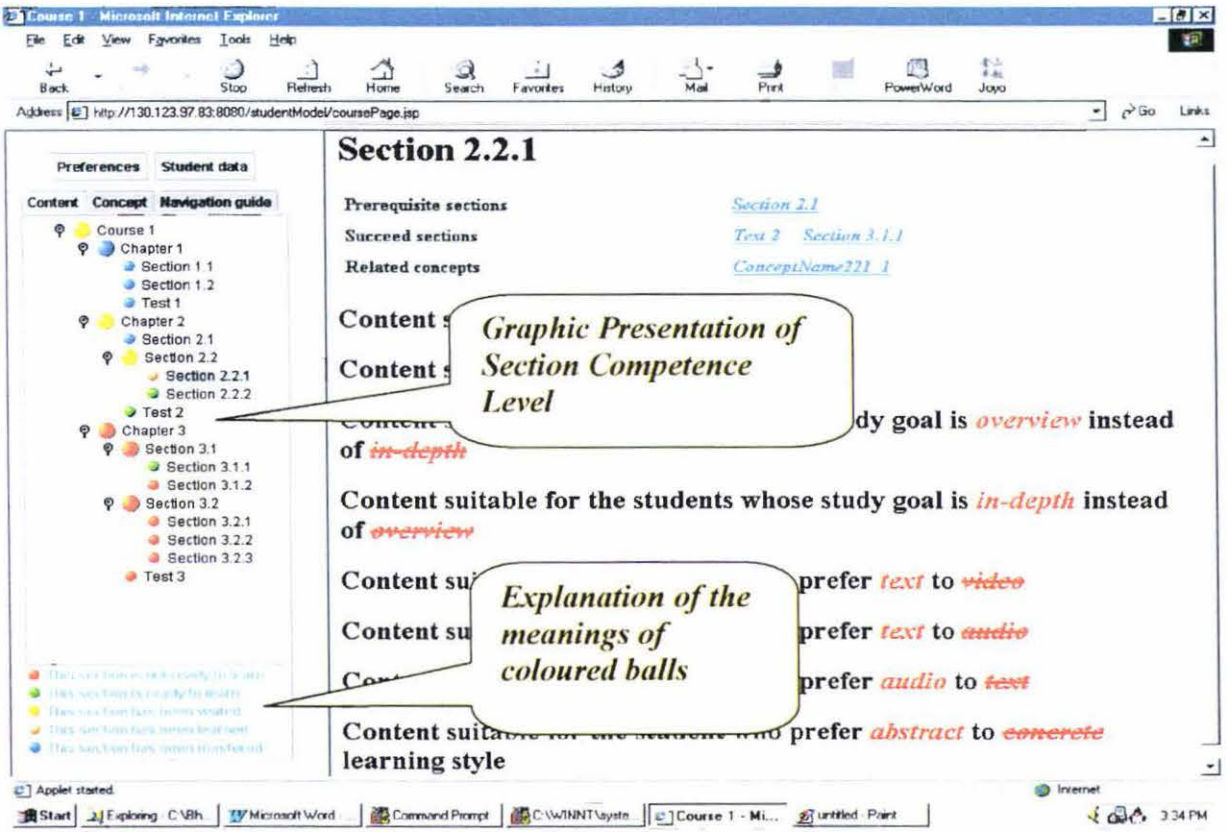

Figure 6-2 Student interface of displaying section competence level

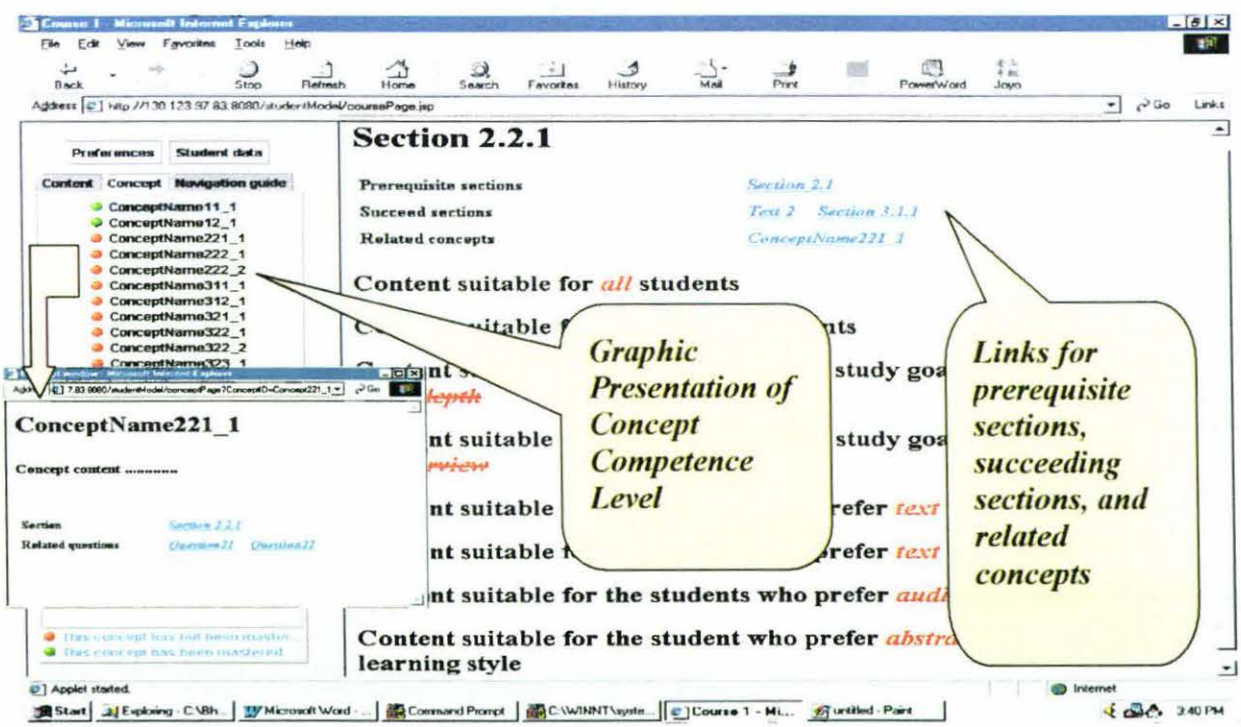

Figure 6-3 Student interface of displaying concept competence level

Graphic Presentation of Concept Competence Level implement the following adaptation methods and techniques:

- **Direct guidance** is implemented by opening a new window to present concept explanation when a concept being clicked.
- **Local guidance** is provided by presenting concept details whenever a student requested.

- **Sorting** is implemented by sorting all the concepts of a course in alphabetic order.
- **Global guidance** is provided by helping the student to find the concept links he/she really want quickly.

- **Annotation** is implemented by annotating concepts with different competence levels in different coloured balls (red ball – *"not mastered"*, and green ball – *"mastered"*).
- **Local orientation support** is provided by indicating the current competence levels of all concepts in a course.

6.3.4 Cross Reference Links

On each section content pages and concept description pages, the structures of knowledge base are clearly presented by *cross-reference links.* The section or concept prerequisite relations, the section-concept relations, and the question-concept relations are illustrated by the links. The *cross-reference links* include:

- For a learning section, the links for *prerequisite sections*, *succeeding sections*, and *related concepts* are provided (Figure 6-3).

- For each question in an assessment section, the links for *related concepts* are provided (Figure 6-4).

- Hint and solution window can be opened from assessment sections for displaying a *hint or solution of a question* (Figure 6-4).

- The concept window includes the links for the *concept related learning section and the concept related questions* (Figure 6-5).

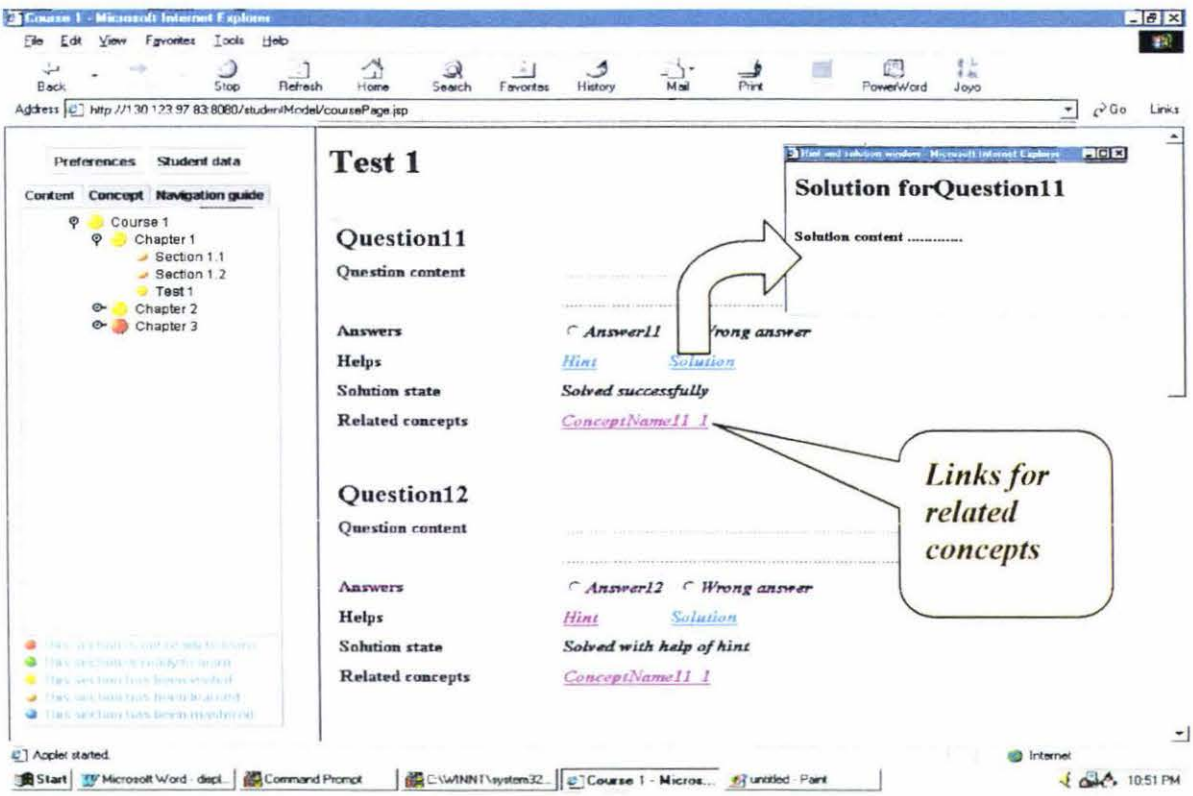

Figure 6-4 Student interface of displaying assessment section (the right panel shows what kinds of components or questions are required in the system)

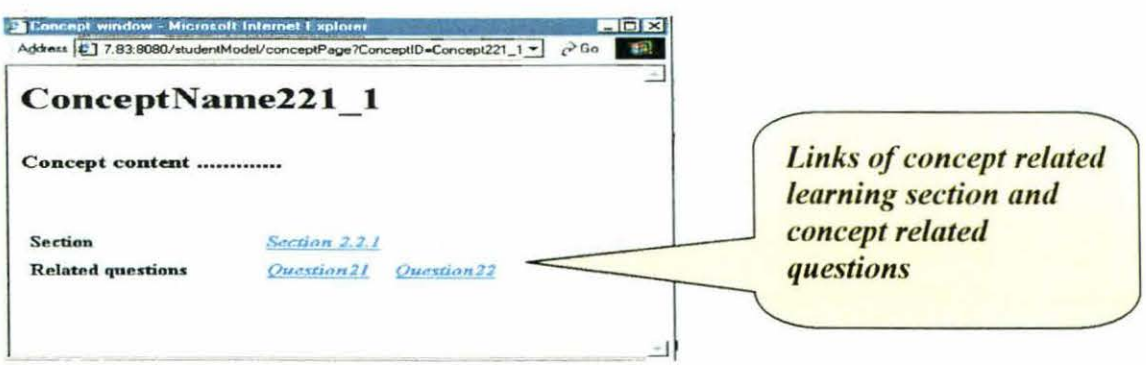

Figure 6-5 Student interface of concept window

Cross Reference Links implement the following adaptation methods and techniques:

- **Direct guidance** is implemented by providing the relevant links directly.
- **Local guidance** is provided by helping the student get the information related to the links easily.

- **Sorting** is implemented by arranging the related links as the same order as they are in the course structure.
- **Local guidance** is provided by helping the student find the shortest way to the desired link.

- **Hiding** is implemented by hiding irrelevant links.
- **Local guidance** is provided by making the student concentrate on relevant links and therefore decreasing the student cognitive load.

- **Annotation** is implemented by underlining links to attract the student attention.
- **Local orientation support** is provided by reminding the student to check and learn the related information.

Table 6-1 is the summary of all the navigation adaptation methods and techniques used in the system.

6.4 Content Adaptation

6.4.1 Content Adaptation Based-on Individual Student model

The system is able to create an individualised content presentation based on the student study goals, experience, learning styles, preferences of multimedia objects, and overall competence levels.

Table 6-1 Summary of navigation adaptation methods that are used in this system

	To Provide: \ Using:	*Direct Guidance*	*Sorting*	*Hiding*	*Annotation*
List of recommended links	*Global guidance*	Yes	Yes		
	Local guidance	Yes	Yes		
	Global orientation support			Yes	Yes
	Local orientation support			Yes	Yes
Graphic presentation of section competence level	*Global guidance*				
	Local guidance				
	Global orientation support				Yes
	Local orientation support				Yes
Graphic presentation of concept competence level	*Global guidance*		Yes		
	Local guidance	Yes			
	Global orientation support				
	Local orientation support				Yes
Cross-reference links	*Global guidance*				
	Local guidance	Yes	Yes	Yes	
	Global orientation support				
	Local orientation support				Yes

Each content section in the content tree is stored as an XML file in which the HTML learning objects suitable for a particular type of students are wrapped as CDATA sections. Because predefined XML tags are used to specify what type of students that a section is targeted at, the XML processing program do not have to check the internal structure of the learning content. The XML files are unwrapped by Java SAX programs into HTML files based on the student domain independent data and the overall competence level. Figure 6-6 shows this update process by using the student video/text and audio/text preference as an example.

Almost all the content adaptation methods and techniques in literature can be used to define the XML tags. By this way, presenting adaptive content suitable to a particular student will be straightforward. As shown in Figure 6-7, the individualised course contents are presented according to domain-independent data and the overall competence level. For example, the contents dealing with difficult details are hidden from students with lower competence levels, and additional explanations are provided. Whenever the information in the individual student model is updated, the course content page is reloaded to adapt to the modified information.

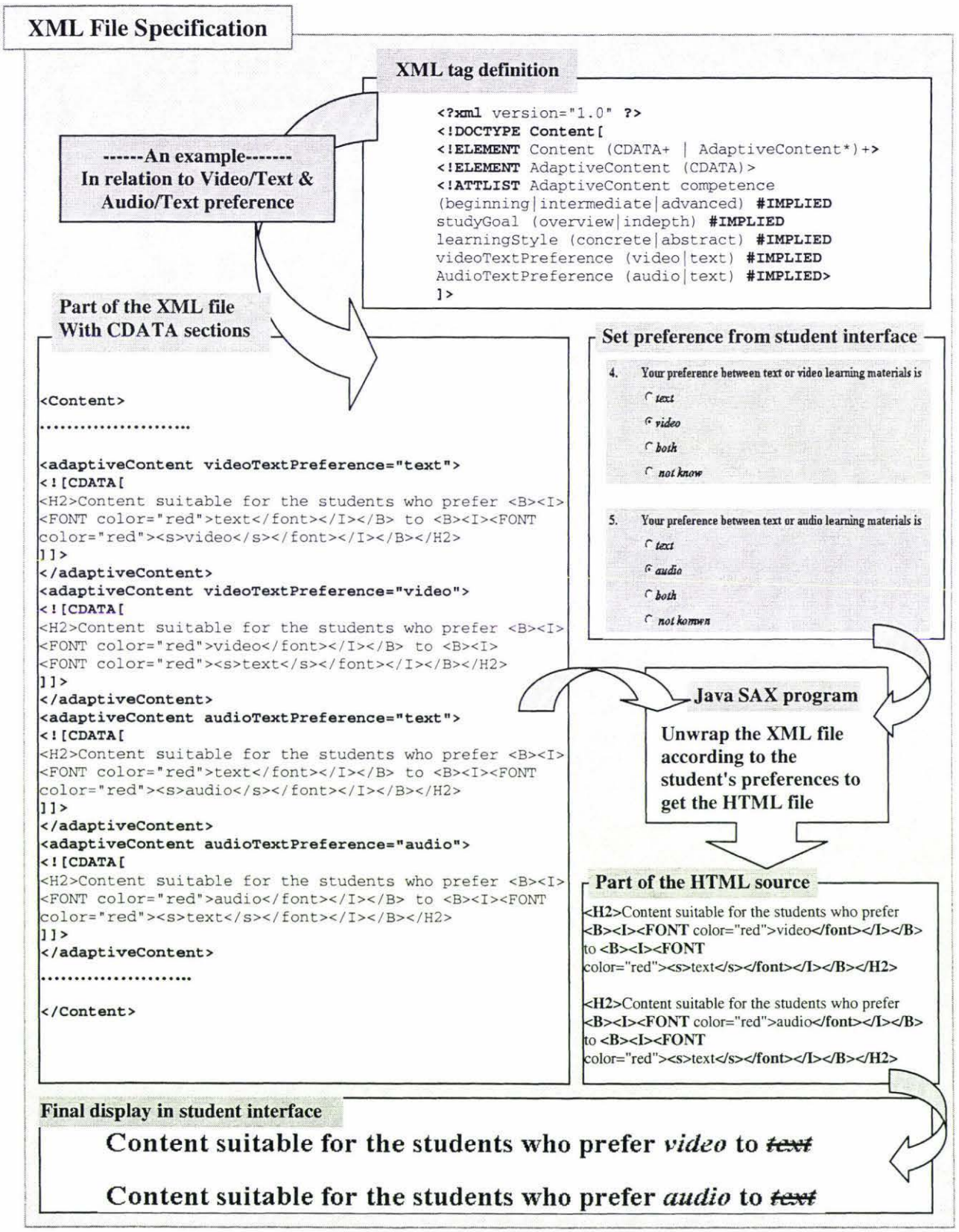

Figure 6-6 XML files processing steps

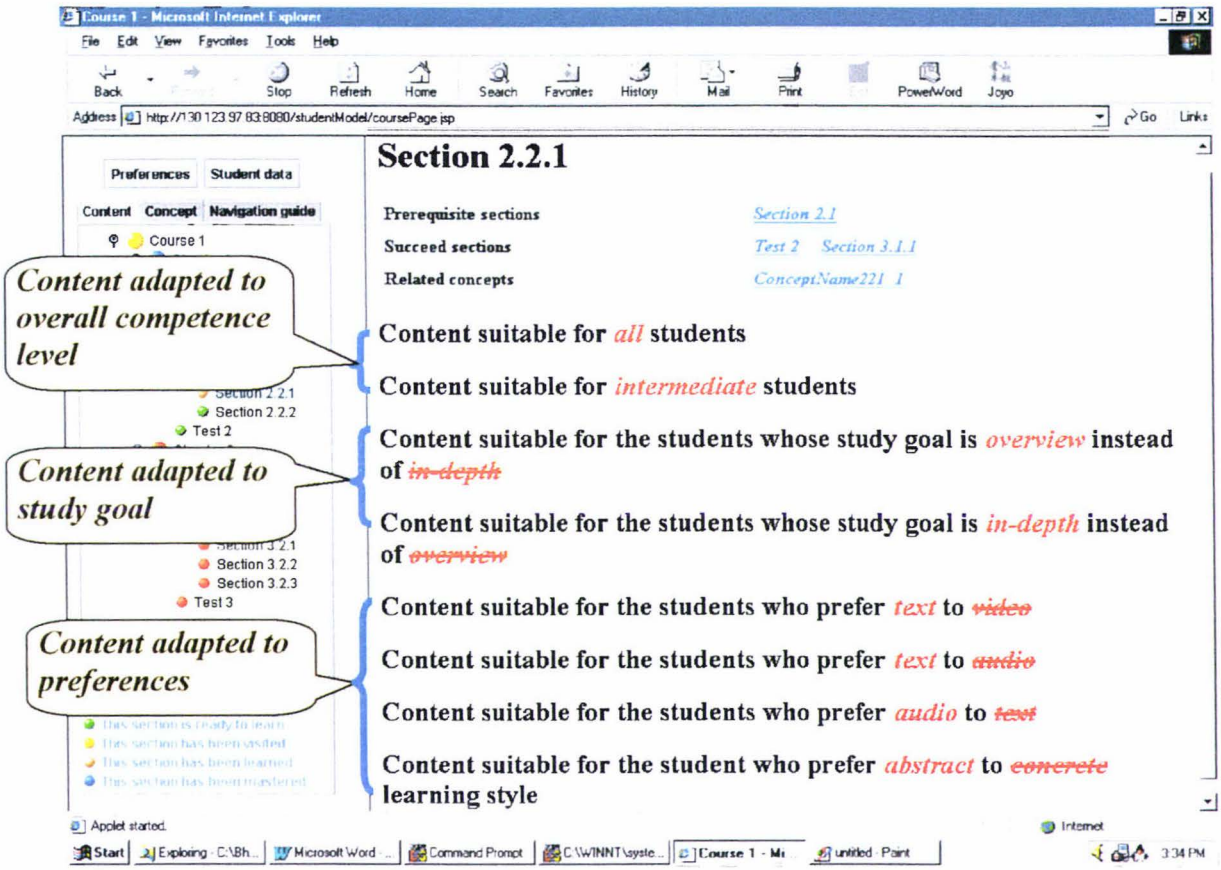

Figure 6-7 Student interface of content presentation

Table 6-2 is the summary of the content adaptation methods and techniques that can be used by a teacher to prepare individualised course materials.

Table 6-2 Content adaptation methods that can be used in the system

		Conditional text	*Stretch-text*	*Fragment variants*	*Page variants*
▪ **Adaptively presentation of course materials in learning sections**	***Additional explanations***	Yes	Yes	Yes	Yes
	Prerequisite explanations	Yes		Yes	
	Comparative explanations	Yes		Yes	
▪ **Dynamically display questions' solution state in assessment sections**	***Explanation variants***	Yes	Yes	Yes	Yes
	Sorting			Yes	

6.4.2 Content Adaptation Based-on Group Student Model

Group student data is the statistic results of individual student data (Figure6-8). It can be used by a teacher to reorganise the learning materials and provide adaptive content suitable to the knowledge levels of the students in this group. For example, if most of the students can not master most of the course sections, additional explanations and examples should be added to the content before it can be presented to a new student in the same group.

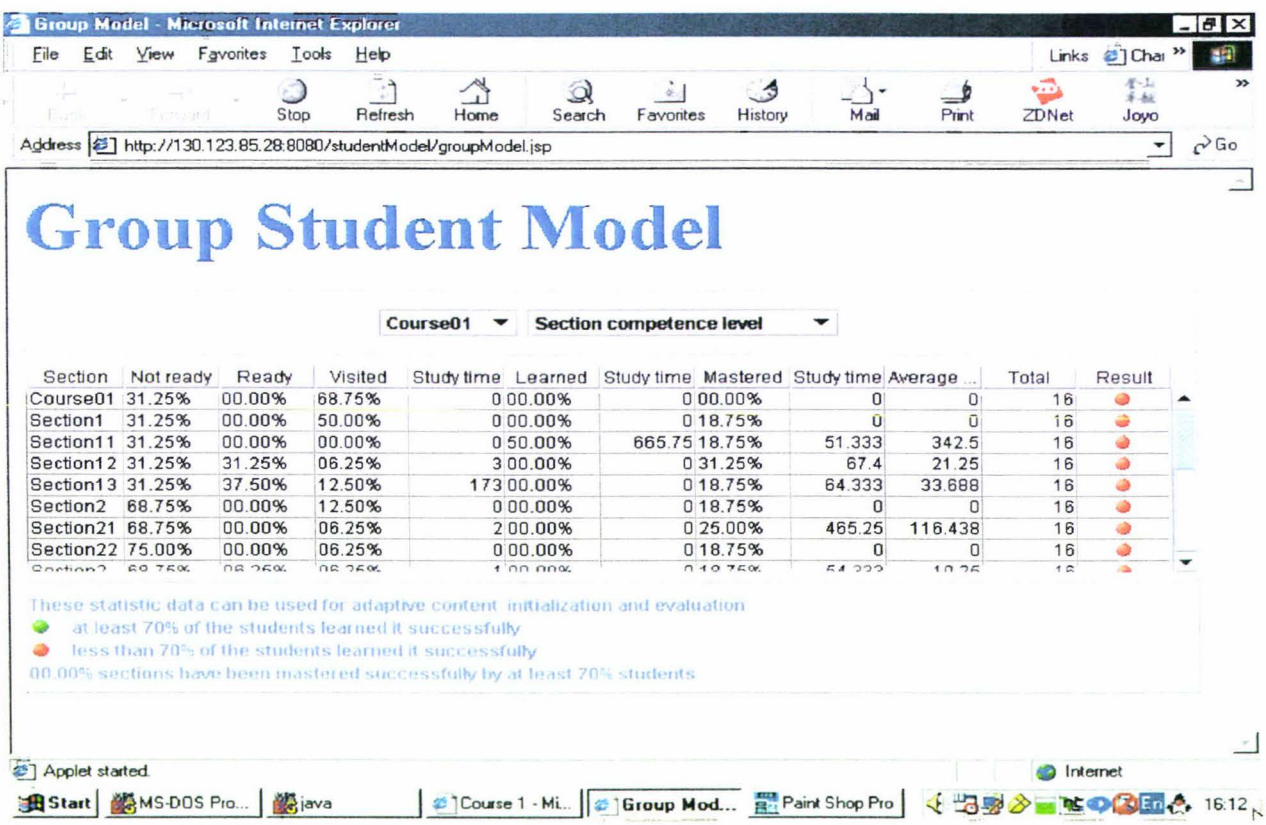

Figure 6-8 Student interface of group student model data

6.5 Summary

This system provides navigation adaptation and content adaptation based on student domain-independent data and domain-specific data. A number of adaptation methods and techniques are utilised for achieving the navigation and content adaptations, which have been illustrated via the presentation interfaces.

Chapter 7 Discussions and Conclusion

7.1 Introduction

This chapter is a summary of the thesis. First the prototype student model and its related adaptation approaches are discussed, then the future work for improving the existing student model is proposed, and finally the conclusions on this research are drawn.

7.2 Discussions

The prototype web-based system developed in this project contained various types of information to represent domain knowledge, and student learning progress and behaviours. The information is constantly updated according to student learning actions. The presentation of the teaching materials is adapted to the information in the individual student model. Following sub-sections summarised various components of the system.

7.2.1 System Architecture

The prototype web-based system in this project is implemented with the *distributed, three-tier, client-server* architecture. The client is a set of HTML files presented in a web browser with embedded Java applets. The Java application programs used for the student model update and adaptation reside in the middle tier server which can communicate directly with the third tier database. This type of system architecture is flexible, and easy to manage and maintain.

7.2.2 Domain Model

The domain model mainly consists of content tree and concept network. The content tree represents the vertical organisation of teaching materials, and the concept network is formed by the prerequisite relations between the concepts. The content tree organises the course contents in

the textbook fashion familiar to most students, and the concept network provides the underlining relations between the course contents.

The domain model possesses a straightforward structure, which can be easily adopted for organising the contents of most disciplines. The structure provides a simple yet sufficient basis for further establishing the student model database.

Although the concepts prerequisite relationship is the most natural and fundamental relationship between curriculum elements in a course, other relations, such as navigational priority, may be included in the concept network to provide more guidance or flexibility during the delivery of courses.

7.2.3 Student Model

Both individual and group student models are implemented in the system, so the system is able to monitor and adapt to the learning behaviours and progresses of an individual student and a group of students.

The individual student model includes *domain-specific information* and *domain-independent information.* The domain-specific information form an overlay model, which is a subset of the domain model, and mainly contains the student competence levels for each concept, section, and overall domain. The *domain-independent information* includes student's background, experience, study goals, and preferences of learning styles. Other types of domain-independent information can also be added to the student model database, such as cognitive aptitudes, motivational states, and distinct intelligent forms. However, the relationships between the student learning behaviours and above information are too complex to be implemented in a generalised student model.

Both explicit questioning and stereotyping are used for student model initialisation. Then the student model update takes place mainly according to the study time on each section and the

progress made by the student in assessment. In the web-based student modelling system, the student learning actions captured at the client side are transferred to the server side for student model update. Delay in transporting large amount of data between the client and server may result in the degradation of the system performance. Therefore only a limited amount of data should be associated for student modelling purposes. The time that a student spent on a section is considered the most relevant data for assessing student efforts. The assessment section provides the best measurements for student competence levels. Therefore above two types of data are captured and sent to the server for updating student model.

The information in the group student model is dynamically extracted from the models of the individual students in the group. Most data in the group student model are computed by averaging the corresponding data in the individual student models. The model data for a group of students is used to initialise the individual models of a similar student group.

7.2.4 Adaptation Approaches

The system provides both navigation adaptation and content adaptation based on the information maintained in the student model. The methods used for achieving the navigational adaptation include the content table, concept list, recommended section list, annotated competence levels, and cross-reference links. The individualised contents are presented according the student competence level, cognitive goals, learning styles, and preferences for learning methods.

The student model uses XML technology for adaptation purposes. It is extremely flexible and powerful to use XML tags to wrap the hyper-media course materials as specific adaptive contents. The teacher can easily define a set of specific tags for the intended adaptation operations. The XML-wrapped course material can be easily processed to present individualised content presentations.

7.3 Future Work

The prototype student model should be validated by the real world educational practices. By using the student model to simulate the student behaviours when they are studying in the real academic environment, the accuracy and effectiveness of the student model can actually be assessed. Based on the assessments, further improvements may be attempted. The assessment – improvement iterative processes should be carried out until the student model can satisfactorily simulate student learning processes.

To improve the prototype student model and its related web-based educational system, the following future work is identified.

7.3.1 Task-Based Stereotyping

Group based stereotyping can be extended to the concept of task-based stereotyping where system observes the action patterns of various students and monitors the results obtained through those task patterns. If a number of students follow a certain task pattern with certain outcome, the system could use this information to give guidance in future when a new student starts to follow that action pattern.

7.3.2 Utilisation of More Student Learning Actions

Some domain-independent information may be updated according to the student learning actions. For instance, the system may record how many times the student studied text objects as compared to the audio objects for deciding his/her text-audio preference.

7.3.3 More Flexible Content Adaptations

The educators may define their own operations for content adaptations. A set of user interfaces may allow the course controllers to specify a set of student model data for specific content adaptations. The additional information may be initialised using explicit questionnaire and stereotype, and updated by capturing certain student learning actions. The system should be able to automatically add the newly defined information into the student model database. A number of student actions may be captured by the system if the educator makes such requests. The content adaptations can be performed based on the added student model data.

7.3.4 Multiple Representation Approach

Multiple Representation Approach (Kinshuk et al, 2000) controls the presentation of multimedia domain content based on learner's domain competence level and a set of rules for using multimedia objects. For instance, a novice in a subject should get direct instruction for knowledge by text or animation rather than complex interactive multimedia object such as simulations. Each multimedia object has its own characteristics, for example, text is good to convey details, audio is good to stimulate imagination, and pictures are good to convey ideas.

7.3.5 Exploration Space Control

Exploration space control (Kashihara et al., 1997) provides a way to the system to automatically control the learning space by limiting information resources, number of searching paths and tools, and amount of information presented. The control may reduce the student cognitive load, and can be based on the student competence level, experience, etc.

7.3.6 Deactivation of Adaptation

Students or teachers may turn on/off part or all of the system adaptation functions. A student may choose to shut down all adaptation processes of student modelling. Alternatively the student may

only switch off the content adaptation, which means all course materials in a section are displayed without adaptive hiding of the content that is originally assumed by the system as not suitable for the student.

7.4 Conclusion

As an important component of the web-based educational system, a student model is mainly employed to provide adaptive course contents and study guidance, and therefore to suit the needs of individual students with different knowledge levels and learning habits.

This study produced a two-fold web-based student modelling system, which is able to model and adapt to the learning actions and progresses of an individual student or a group of students in the web-based learning systems. The learning efforts of an individual student are estimated by the time that the student spent in the course contents. The learning outcomes of a student are assessed by her/his answers to a set of questions that are associated with particular concepts or contents. The group student model is constructed dynamically from the information contained in the individual student models. Both navigation and content adaptations are provided based on individual and group student models.

The approach in the project is general in a sense that can be extended in a number of ways as discussed in previous future work section. A number of more sophisticated student modelling and adaptation functions may be added to the existing system to form a more intelligent web-based educational system.

References

Anjaneyulu, K. (1997). Concept Level Modelling on the WWW. In Brusilovsky, P., Nakabayashi, K. & Ritter, S. (Eds.) Proceedings of Workshop "*Intelligent Educational Systems on the World Wide Web*" at AI-ED'97, 8th World Conference on Artificial Intelligence in Education, ISIR, 26-29,
http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Anjaneyulu.html.

Armstrong, R., Freitag, D., Joachims, T. & Mitchel, T. (1995). WebWatcher: A learning apprentice for the World Wide Web. *AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments*, Stanford, CA,
http://www.isi.edu/sims/knoblock/sss95/mitchell.ps.

Asnicar, F. A. & Tasso, C. (1997). ifWeb: A prototype of user model-based intelligent agent for document filtering and navigation in the World Wide Web. In Brusilovsky, P., Fink, J. & Kay, J. (Eds.) Proceedings of Workshop "*Adaptive Systems and User Modelling on the World Wide Web*" at 6th International Conference on User Modelling, Carnegie Mellon Online, 3-11,
http://www.contrib.andrew.cmu.edu/~plb/UM97_workshop/Tasso.html.

Beaumont, I. (1994). User modelling in the interactive anatomy tutoring system ANATOM-TUTOR. *User Modelling and User Adapted Interaction*, 4 (1), 21-45.

Benaki, E., Karkaletsis, V. A. & Spropoulos, C. D. (1997). User Modelling in the WWW: the UMIE Prototype. Paper presented at the workshop "*Adaptive Systems and User Modelling on the World Wide Web*" at 6th International Conference on User Modelling, Chia Laguna, Sardinia, Italy, June 2-5, 1997.

Bishop, A. S., Greer, J. E., & Cooke, J. E. (1997). The co-operative peer response system: CPR for students. In Müldner, T. & Reeves, T. C. (Eds.) Proceedings of *ED-MEDIA/ED-TELECOM'9 World Conference on Educational Multimedia/Hypermedia and World Conference on Educational Telecommunications*, AACE, 74-79.

Boyle, C. & Encarnacion, A. O. (1994). MetaDoc: an adaptive hypertext reading system. *User Models and User Adapted Interaction,* 4 (1), 1-19.

Bra, P. D. (1999). Design issue in adaptive web-site development. Paper presented at the 2nd workshop "*adaptive systems and user modelling on the www*" at 7th international conference on user modelling, Banff, Canada, June 20-24, 1999.

Brusilovsky, P. (1994). The construction and application of student models in intelligent tutoring systems. *Computer and System Sciences International*, 32(1), 70-89.

Brusilovsky, P. (1995). Intelligent tutoring systems for World-Wide Web. Presented at *Third International WWW Conference*, Darmstadt, April 10-14, 1995.

Brusilovsky, P. (1996a). Methods and techniques of adaptive hypermedia. In Brusilovsky, P. & Vassileva, J. (Eds.) *Special Issue on Adaptive Hypertext and Hypermedia: User Modelling and User-Adapted Interaction,* 6 (2-3), 87-129.

Brusilovsky, P. (1996b) A tool for developing adaptive electronic textbooks on www. Paper presented at *WebNet'96 - World Conference of the Web Society,* San Francisco, CA., October 16-19, 1996.

Brisilovsky, P. (2000). *Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies,*
http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/Brusilovsky.html.

Brusilovsky, P. L. (1992a). A framework for intelligent knowledge sequencing and task sequencing. In Frasson, C., Gauthier, G. & McCalla, G. I. (Eds.) *Intelligent Tutoring Systems,* Berlin: Springer-Verlag, 499-506.

Brusilovsky, P. L. (1992b). Intelligent Tutor, Environment and Manual for Introductory Programming. *Educational and Training Technology International,* 29 (1), 26-34.

Brusilovsky, P., Eklund, J. & Schwarz, E. (1997a). Adaptive Navigation Support in Educational Hypermedia on the World Wide Web. In Howard, S., Hammond, J. & Lindgaard, G. (Eds.) *Human-Computer Interaction,* New York: Chapman and Hall, 278-285.

Brusilovsky, P., Eklund, J. & Schwarz, E. (1998). Web-based education for all: a tool for development adaptive courseware. *Computer Networks and ISDN Systems,* 30 (1-7), 291-300.

Brusilovsky, P. & Pesin, L. (1994). ISIS-Tutor: An adaptive hypertext learning environment. In Ueno, H. & Stefanuk, V. (Eds.) *Japanese-CIS Symposium on knowledge-based software engineering*, EIC, 83-87.

Brusilovsky, P., Ritter, S. & Schwarz, E. (1997b). Distributed intelligent tutoring on the Web. In du Boulay, B. & Mizoguchi, R. (Eds.) Proceedings of *AI-ED'97, 8th World Conference on Artificial Intelligence in Education*, Amsterdam: IOS, 482-489.

Brusilovsky, P. & Schwarz, E. (1997). User as student: Towards an adaptive interface for advanced web-based applications. In Jameson, A. C. Paris & Tasso, C. (Eds.) Proceedings of *6th International Conference on User Modelling,* Wien: Springer-Verlag, 177-188.

Brusilovsky, P., Schwarz, E., & Weber, G. (1996). ELM-ART: An intelligent tutoring system on World Wide Web. In Frasson, C., Gauthier, G. & Lesgold, A. (Eds.) *Intelligent Tutoring Systems.* Lecture Notes in Computer Science, Vol. 1086, Berlin: Springer Verlag, 261-269.

Brusilovsky, P. & Zyryanov, M. (1993). Intelligent tutor, environment and manual for physical geography. Paper presented at 7th International PEG Conference, Edinburgh, July 1993.

Calvi, L. & De Bra, P. (1997). Using dynamic hypertext to create multi-purpose textbooks. In Müldner, T. & Reeves, T. C. (Eds.) Proceedings of *World Conference on Educational*

Multimedia/Hypermedia and World Conference on Educational Telecommunications, AACE, 130-135.

Capuano, N., Marsella, M. & Salerno, S. (2000). ABITS: An agent based intelligent tutoring system for distance learning. Paper presented at the International Workshop on "*Adaptive and Intelligent Web-based Educational Systems*", Montreal, Canada, June 19th, 2000.

Choi, J. & Woo, C. (2000). Adaptive programming Language tutoring systems on the web. Paper presented at *International Conference on Computer in Education/International Conference on Computer Assisted Instruction 2000 (ICCE/ICCAI 2000)*, The Greate Hotel, Tai Pei, Taiwan, November 21-24, 2000.

Clibbon, K. (1995). Conceptually adapted hypertext for learning. Paper presented at 1995 Conference on *Human Factors in Computing Systems (CHI'95),* Denver, Colorado, USA, May 7 - 11, 1995.

Conati, C. & VanLehn, K. (1996). POLA: A student modelling framework for probabilistic on-line assessment of problem solving performance. Paper presented at *5th International Conference on User Modelling* (UM-96), Kailuna-Kona, Hawaii, USA, Jan. 2-5, 1996.
http://www.cs.brandeis.edu/~bgoodman/um_96_toc.html

Corbett, A. T. & Anderson, J. A. (1992). Knowledge tracing in the ACT programming tutor. Paper presented at 14th Annual Conference of the Cognitive Science Society-Indian University, Laurence Erlbaum Associates, 1992.

Debevc, M., Rajko, S. & Donlagic, D. (1994). Adaptive bar implementation and ergonomics. *Informatics : Journal of Computing and Informatics* 18, 357-366.

de La Passardiere, B. & Dufresne, A. (1992). Adaptive navigational tools for educational hypermedia. In Tomek, I. (Ed.) *Computer Assisted Learnin,* Berlin: Springer-Verlag, 555-567.

de Rosis, F., de Carolis, B. & Pizzutilo, S. (1993). User tailored hypermedia explanations. *INTERCHI'93 Adjunct proceedings*, Amsterdam, 169-170, http://wwwis.win.tue.nl/ah94/deRosis.html

Diego, J., Rivera, Z., Jim, E. & Cooke, J. (2000). An XML-Based Tool for Building and Using Conceptual Maps in Education and Training Environments. Paper presented at *International Conference on Computer in Education/International Conference on Computer Assisted Instruction 2000* (ICCE/ICCAI 2000), Tai Pei, Taiwan, November 21-24, 2000.

Eliot, C., Neiman, D. & Lamar, M. (1997). Medtec: A Web-based intelligent tutor for basic anatomy. In Lobodzinski, S. & Tomek, I. (Eds.) Proceedings of *World Conference of the WWW, Internet and Intranet* (WebNet'97), AACE, 161-165.

Far, B.H. & Hashimoto, A. H. (2000). A Computational Model for Learner's Motivation States in Individualised tutoring System. Paper presented at *International conference of Computer in Education/International conference on computer assisted Instruction 2000* (ICCE/ICCAI 2000), Tai Pei, Taiwan, November 21-24, 2000.

Fink, J., Kobsa, A. & Schreck, J. (1997). Personalised hypermedia information provision through adaptive and adaptable system features: User modelling, privacy, and security issues. In Brusilovsky, P. Fink, J. & Kay, J. (Eds.) Proceedings of Workshop "*Adaptive Systems and User Modeling on the World Wide Web*" at 6th International Conference on User Modelling, Carnegie Mellon Online, 43-53, http://www.contrib.andrew.cmu.edu/~plb/UM97_workshop/Fink.html.

Fischer, G., Mastaglio, T., Reeves, B. & Rieman, J. (1990). Minimalist explanations in knowledge-based systems. *IEEE Computer Society*, vol. 3, 309-317.

Gehne, R., Jesshope, C. & Zhang, J. (2000). Technology integrated learning environment---a web-based distance learning system. Paper resented at *IASTED International Conference 2001,*

Internet and Multimedia Systems and Applications, Honolulu, Hawaii, USA, August 13-16, 2001.

Goldstein, I. P. (1979). The genetic graph: A representation for the evolution of procedural knowledge. *Intern. J. Machine Studies*, Vol. 11, No. 1, 1979

Gonschorek, M. & Herzog, C. (1995). Using hypertext for an adaptive help system in an intelligent tutoring system. Paper presented at *the 7th World Conference on Artificial Intelligence in Education (AI-ED'95)*, Washington, DC, August 1995.

He, X. & Cabirc, D. (2001). *Sever-side scripting languages as tools for collecting and organising data,*
http://ltol.scnu.edu.cn/fullpaper/415-f.html.

Henze, N. & Nejdl, W. (2000). *Bayesian modelling for adaptive hypermedia systems,* http://www.kbs.uni-hannover.de/Arbeiten.../bayesianModeling/bayesianApproach.html

Hohl, H., Böcker, H.-D. & Gunzenhäuser, R. (1996). Hypadapter: An adaptive hypertext system for exploratory learning and programming. In Brusilovsky, P. & Vassileva, J. (Eds.) *Special Issue on Adaptive Hypertext and Hypermedia, User Modelling and User-Adapted Interaction* 6 (2-3), 131-156.

Höök, K. (1997). Evaluating the utility and usability of an adaptive hypermedia system. In Moore, J., Edmonds, E. & Puerta, A. (Eds.) Proceedings of *1997 International Conference on Intelligent User Interfaces*, ACM, 179-186.

Hoppe, U. (1995). Use of multiple student modelling to parameterise group learning. In Greer, J. (Ed.) Proceedings of *7th World Conference on Artificial Intelligence in Education* (AI-ED'95), AACE, 234-249.

Ikeda, M., Go, S. & Mizoguchi, R. (1997). Opportunistic group formation. Paper presented at *8th World Conference on Artificial Intelligence in Education: Knowledge and Media in Learning Systems* (AI-ED'97), Kobe, Japan, August 18-22, 1997.

Jesshope, C., Heinrich, E. & Kinshuk (2000). Technology integrated learning environments for education at a distance. Paper presented at *DEANZ 2000 Conference*, Dunedin, New Zealand, April 26-29, 2000.

Johnson, C. & Orwig, C. (1998). *What is learning style,* http://www.sil.org/lingualinks/library/Llearning/CJ0625/CJ0676.html.

Kaplan, C., Fenwick, J. & Chen, J. (1993). Adaptive hypertext navigation based on user goals and context. *User Modelling and User Adapted Interaction,* 3 (3), 193-220.

Kaptelinin, V. (1993). Item recognition in menu selection: The effect of practice. Paper presented at *INTERACT '93 and CHI '93 conference companion on Human factors in computing systems*, Amsterdam, The Netherlands, April 24 - 29, 1993.

Kashihara, A., Kinshuk, Oppermann, R., Rashev, R. & Simm, H. (1997). An Exploration Space Control as Intelligent Assistance in Enabling Systems. In Halim, Z., Ottmann, T. & Razak, Z. (Eds.) Proceedings of *International Conference in Computers in Education*, AACE, VA, 114-121 (ISBN 983-9151-03-7).

Kashihara, A., Kinshuk, Oppermann, R., Rashew, R. & Simm, H. (2000). A Cognitive Load Reduction Approach to Exploratory Learning and Its Application to an Interactive Simulation-based Learning System. *Journal of Educational Multimedia and Hypermedia*, 9 (3), 253-276 (ISSN 1055-8896).

Kay, J. & Kummerfeld, B. (1994a). Adaptive hypertext for individualised instruction. Paper presented at "*the Workshop on Adaptive Hypertext and Hypermedia*" at User Modelling '94, Cape Cod, August 1994,

http://www.cs.usyd.edu.au/~bob/adapt_hyper.html

Kay, J. & Kummerfeld, R. J. (1994b). An individualised course for the C programming language. Paper presented at *Proceedings of Second International WWW Conference*, Chicago, IL, October 17-20, 1994,
http://archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/Educ/kummerfeld/kummerfeld.html

Keller, R. M., Wolfe, S. R., Chen, J. R., Rabinowitz, J. L. & Mathe, N. (1996). A bookmarking service for organising and sharing URLs. Paper presented at *6th International World Wide Web Conference*, Santa Clara, CA, April 1997.

Kettel, L., Thomson, J. & Greer, J. (2000). Generating individualised hypermedia applications. Paper presented at International Workshop on "*Adaptive and Intelligent Web-based Educational Systems*", Montreal, Canada, June 19, 2000.

Kinshuk. (1996). *Computer-Aided Learning for Entry-Level Accountancy Students,* PhD Thesis, De Montfort University, England.

Kinshuk, Oppermann, R., Patel, A. & Kashihara, A. (2000). Multiple representation approach in multimedia based intelligent educational systems. In Lajole, S. P. & Vivet, M. (Eds.) *Artificial Intelligence in Education IOS Press*, Amsterdam, 259-266.

Kobsa, A., Müller, D. & Nill, A (1994). KN-AHS: An adaptive hypertext Client of the user modelling system BGP-MS. Paper presented at *4th International Conference on User Modelling*, Hyannis, Cape Cod, Massachusetts U.S.A., August 15 - 19, 1994.

Kushniruk, A. & Wang, H. (1994). A hypermedia-based educational system with knowledge-based guidance. Paper presented at the *World conference on educational multimedia and hypermedia* (ED-MEDIA'94), Vancouver, Canada, June 25-29, 1994.

Lai, M.-C., Chen, B.-H. & Yuan, S.-M. (1995). Toward a new educational environment. Paper presented at *4th International World Wide Web Conference*, Boston, USA, December 11-14, 1995, http://www.w3.org/pub/Conferences/WWW4/Papers/238/.

Lane, C. (1998) *Gardner's multiple Intelligence,*
http://www.tecweb.org/eddevel/gardner.html

Lane, C. (2000). *Learning styles and multiple intelligence in distributed learning/IMS projects,*
http://www.tecweb.org/pbs/lsmidl.pdf

Martin, J. D. & VanLehn, K. (1993). OLAE: Progress toward a multi-activity, Bayesian student modeller. In Brna, P., Ohlsson, S. & Pain, H. (Eds.) Proceedings of *the World Conference on Artificial Intelligence in Education.* Edinburgh, Scotland: AACE, 410- 417.

Mathé, N. & Chen, J. (1996). User-centred indexing for adaptive information access. *User Modelling and User-Adapted Interaction*, July 1996, 6(2-3), 225--261.

McCalla, G. I., Greer, J. E., Kumar, V. S., Meagher, P., Collins, J. A., Tkatch, R. & Parkinson, B. (1997). A peer help system for workplace training. In Boulay, B. D. & Mizoguchi, R. (Eds.) *Artificial Intelligence in Education: Knowledge and Media in Learning Systems*, Amsterdam: IOS, 183-190.

McKendree, J., Radlinski, B. & Atwood, M. E. (1992). The Grace Tutor: a qualified success. In Frasson, C., Gauthier, G. & McCalla, G. I. (Eds.) Proceedings of *Second International Conference*, ITS'92, Springer-Verlag, 677-684.

Micarelli, A. & Sciarrone, F. (1996). A case-based toolbox for guided hypermedia navigation. Presented at *5th International Conference on User Modeling*, UM-96, Kailua-Kona, Hawaii, Jan. 2-5, 1996.

Mitovic, A. (2000). Porting SQL-Tutor to the web. Paper presented at the Workshop on "*Adaptive and Intelligent Web-based Educational Systems*", Held in Conjunction with ITS 2000, Montreal, Canada, June 19, 2000,
http://www.cosc.canterbury.ac.nz/~tanja/sqltw-its.htm

Montgomery, S. M. & Groat, L. N. (2000) *Student learning styles and their implications for teaching*,
http://www.crlt.umich.edu/occ

Mukherjea, S. & Foley, J. D. (1995). Visualizing the World-Wide Web with navigational view builder. *Computer Networks and ISDN Systems,* 27 (6), 1075-1087.

Mukherjea, S., Foley, J. D. & Hudson, S. (1995). Visualizing complex hypermedia networks through multiple hierarchical views. Paper presented in *CHI'95*, Denver, Colorado, USA, May 7 -11, 1995.

Murray, T., Shen, T., Piemonte, J. & Condit, C. (2000). Adaptivity in the metalinks Hyper-book authoring framework. Paper presented at International Workshop on "*Adaptive and Intelligent Web-based Educational Systems*", Held in Conjunction with ITS 2000, Montreal, Canada, June19, 2000

Nakabayashi, K., Maruyama, M., Kato, Y., Touhei, H. and Fukuhara, Y. (1997). Architecture of an intelligent tutoring system on the WWW. In Boulay, B. D. & Mizoguchi, R. (Eds.*) Artificial Intelligence in Education: Knowledge and Media in Learning Systems,* Amsterdam: IOS, 39-46.

Nomoto, T., Matsuda, N., Hirashima, T. & Toyoda, J. i. (1997). Toward Learning from Surfing. In Brusilovsky, P., Nakabayashi, K. & Ritter, S. (Eds.) Proceedings of Workshop *"Intelligent Educational Systems on the World Wide Web"* at AI-ED'97, 8th World Conference on Artificial Intelligence in Education, ISIR, 40-46,
http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Nomoto.html.

Okazaki, Y., Watanabe, K. & Kondo, H. (1997). An Implementation of the WWW Based ITS for Guiding Differential Calculations. In Brusilovsky, P., Nakabayashi, K. & Ritter, S. (Eds.) Proceedings of Workshop "*Intelligent Educational Systems on the World Wide Web*" at AI-ED'97, 8th World Conference on Artificial Intelligence in Education, ISIR, 18-25, http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Okazaki/Okazaki.html.

Paiva, A.M. (1995). *About user and learner modelling –an overview,* http://www.cbl.leeds.ac.uk/amp/MyPapers/overviewSUMChap.ps

Pérez, T., Gutiérrez, J. & Lopistéguy, P. (1995). An adaptive hypermedia system. Paper presented at *7th World Conference on Artificial Intelligence in Education (AI-ED'95),* Washington, DC, August 1995.

Peylo, C., Thelen, T., Rollinger, C. & Gust, H. (2000). A web-based intelligent educational system for PROLOG. Paper presented at *International Workshop on Adaptive and Intelligent Web-based Educational Systems*, Held in Conjunction with ITS 2000, Montreal, Canada, June19, 2000

Ritter, S. (1997). Pat Online: A Model-tracing tutor on the World-Wide Web. In Brusilovsky, P., Nakabayashi, K. & Ritter, S. (Eds.) Proceedings of Workshop "*Intelligent Educational Systems on the World Wide Web*" at AI-ED'97, 8th World Conference on Artificial Intelligence in Education, ISIR, 11-17, http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Ritter/Ritter.html.

Rivlin, E., Botafogo, R. & Shneidermann, B. (1994). Navigating in hyper-space: designing a structure-based toolbox. *Communications of the ACM* 37 (2), 87-96.

Schwarz, E., Brusilovsky, P. L. & Weber, G. (1996). World-wide intelligent textbooks. In P. Carlson & F. Makedon (Eds) Proceedings of *EDTELEKOM 96 - World Conference on Educational Telecommunications*, Charlottesville, VA: AACE, 302--307.

Self. J. A. (1988). Student models: what use are they? Artificial Intelligence Tools in Education. Paper presented at *IEP TC3 Working Conference on AI Tools in Education*, Frasacai, May 26-28, 1987.

Self, J. A. (1994). Formal approach to student modelling. In McCalla, G. I. & Greer, J. E. (Eds.) *Student Modelling: the Key to Individualised Knowledge-Based Instruction*, NATO-ASI Series F, Berlin: Springer-Verlag, 1994, 3-35,

Shute, V. J. (1995). *SMART:* student modelling approach for responsive tutoring. *User Modelling and User-Adapted Interactions* 5, Kluwer Academic Publisher, 1-44.

Signore, O., Bartoli, R., & Fresta, G. (1997). Tailoring Web pages to users' needs. In Brusilovsky, P., Fink , J. & Kay, J. (Eds.) Proceedings of Workshop "*Adaptive Systems and User Modelling on the World Wide Web*" at 6th International Conference on User Modelling, Carnegie Mellon Online, 85-90,
http://www.contrib.andrew.cmu.edu/~plb/UM97_workshop/Signore.html.

Specht, M., Weber, G., Heitmeyer, S. & Schöch, V. (1997). AST: Adaptive WWW-Courseware for Statistics. In Brusilovsky, P. Fink, J. & Kay, J. (Eds.) Proceedings of Workshop "*Adaptive Systems and User Modelling on the World Wide Web*" at 6th International Conference on User Modelling, Carnegie Mellon Online, 91-95,
http://www.contrib.andrew.cmu.edu/~plb/UM97_workshop/Specht.html.

Srern, M. (1997). The difficulties in web-based tutoring, and some possible solutions. Paper presented at the workshop *"Intelligent educational systems on the world wide web"*, 8th world conference of the AIED society, Kobe, Japan, 18-22 August 1997.

Stauffer, K. (1996). *Applications of Student Modelling,*
http://ccism.pc.athabascau.ca/html/students/stupage/Project/sm_app.html

Stern, M., Woolf, B. P. & Kuroso, J. (1997). Intelligence on the Web? In Boulay, B. D. & Mizoguchi, R. (Eds.) *Artificial Intelligence in Education: Knowledge and Media in Learning Systems,* Amsterdam: IOS, 490-497.

Suthers, D. & Jones, D. (1997). An architecture for intelligent collaborative educational systems. In Boulay, B. D. & Mizoguchi, R. (Eds.) *Artificial Intelligence in Education: Knowledge and Media in Learning Systems,* Amsterdam: IOS, 55-62.

The Omega group (2000). Adaptive Course Generation and Presentation. Paper presented at the International Workshop on "*Adaptive and Intelligent Web-based Educational Systems*", Held in Conjunction with ITS 2000, Montreal, Canada, June19, 2000.

Thomas, C. G. & Fischer, G. (1996). Using agents to improve the usability and usefulness of the World Wide Web. Paper presented at *5th International Conference on User Modelling* (UM-96), Kailua-Kona, Hawaii, Jan. 2-5, 1996.

Tomek, I., Maurer, H. & Nassar, M. (1993). Optimal presentation of links in large hypermedia systems. Paper presented at the *World conference on educational multimedia and hypermedia (ED-MEDIA'93)*, Orlando, FL, June 1993.

Vassileva, J. (1996). A task-centred approach for user modelling in a hypermedia office documentation system. *User Models and User Adapted Interaction 6* (1996), 185--223,

Vassileva, J. (1997). Dynamic Course Generation on the WWW. In Boulay, B. d. & Mizoguchi, R. (Eds.) *Artificial Intelligence in Education: Knowledge and Media in Learning Systems,* Amsterdam: IOS, 498-505.

Warendorf, K. & Tan, C. (1997). ADIS - An animated data structure intelligent tutoring system or Putting an interactive tutor on the WWW. In Brusilovsky, P., Nakabayashi, K. & Ritter, S. (Eds.) Proceedings of Workshop "*Intelligent Educational Systems on the World Wide Web*" at AI-ED'97, 8th World Conference on Artificial Intelligence in Education, ISIR, 54-60,

http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Warendorf/Warendorf.html.

Webb, G. I & Kuzmycz, M. (1998). Evaluation of Data Aging: A Technique For Discounting Old Data During Student Modelling. Paper presented at the 4th International Conference on Intelligent Tutoring Systems, San Antonio, Texas, August 17-19, 1998.

Weber, G. & Möllenberg, A. (1995). ELM-Programming-Environment: A Tutoring System for LISP Beginners. In Wender, K. F., Schmalhofer, F. & Böcker, H. D. (Eds.) *Cognition and Computer Programming*, Norwood, NJ: Ablex, 373-408.

Weber, G. & Spcht, M. (1997). User modelling and adaptive navigation support in www-based tutoring systems. Paper presented at UM-97, Cagliari, Italy, June 2-5, 1997.

Wilson, P. & Coghill, G. (2000). Student learning issues: factors to consider prior to designing computer-assisted learning for higher education. Paper presented at *ICCE 2000*, LosAngeles, CA, USA, June 13-15, 2000.

Zeiliger, R. (1993). Adaptive testing: contribution of the SHIVA model. In Leclercq, D. & Bruno, J. (Eds.) *Item banking: Interactive testing and self-assessment.* NATO ASI Serie F, Berlin: Springer-Verlag, Vol. 112, 54-65.

Zukerman, I. & McConachy, R. (1993). Consulting a user model to address a user's inferences during content planning. *User Models and User Adapted Interaction*, 3 (2), 155-185.