Template-driven Teacher Modelling Approach

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science

in Information Science

at Massey University, Palmerston North

Department of Information Systems
College of Business

Yanmin Shi

2004

ACKNOWLEDGEMENTS

I am grateful to Dr. Alexei Tretiakov and Kinshuk for their collaboration in the preparation of our article on template server architecture which was presented at the 3rd IEEE International Conference on Advanced Learning Technologies and published in its proceedings. Also, I am grateful to Roland Kaschek for inviting me to make a presentation on the subject of this thesis at the e-Commerce seminar. I also would like to thank many lecturers, friends, and supporters who have made this happen. My loving wife, Jie Wang, assisted me in my research. Barry Jackson, Peter Blakey, Madre Chrystall, and the other members of academic staff were of invaluable help. Finally, thanks go to the HOD Chris Freyberg, the Secretary Julie, Lyons, Rachael Carruthers, and other supporting staff for providing the adequate, and well-equipped environment in which to work.

ABSTRACT

This thesis describes the Template-driven Teacher Modeling Approach, the initial implementation of the template server and the formative evaluation on the prototype. The initiative of Template-driven teacher modeling is to integrate the template server and intelligent teacher models in Web-based education systems for course authoring. There are a number of key components in the proposed system: user interface, template server and content repository. The Template-Driven Teacher Modeling (TDTM) architecture supports the course authoring by providing higher degree of control over the generation of presentation. The collection of accumulated templates in the template repository for a teacher or a group of teachers are selected as the inputs for the inference mechanism in teacher's model to calculate the best representation of the teaching strategy, and then predict teacher intention when he or she interacts with the system. Moreover, the presentation templates are kept to support the re-use of the on-line content at the level of individual screens with the help of Template Server.

TABLE OF CONTENT

TABLE OF FIGURES

TABLES

REFERENCE

APPENDIX

Chapter1 Introduction

1.1 Background

The broadening acceptance of the World Wide Web as a medium for access to commercial information and services has been accompanied by a growth in the spectrum of pedagogical practice, such as online teaching or distance learning. Education Service Providers (ESPs) are becoming aware of the needs to offer services that satisfy their users, i.e. both teachers and students. Ultimately, the effectiveness and efficiency of the teaching and learning processes become the main concerns in developing and deploying the Web-based educational systems [Specht, 1998]. The Intelligent Tutoring Systems (ITS) indicate the improvements to the effectiveness of the computer-mediated educational systems by introducing the student model [Wenger, 1987]. Based on students' knowledge levels and preferences, the systems cope with issues like personalization and adaptive presentation of contents. Such systems possess a degree of intelligence and adaptive nature. It means that the amount of the content and the navigation to the content could be dynamically adjusted according to student's academic level, preferences and learning ability. The motivation of modelling students in these systems is to overcome the students' tendency to cognitive overloading and enhance the effectiveness of the learning process. However, these systems have a number of problems and limitations when they are used in educational practice.

The first problem is the over-emphasis of on students and insufficient attention to the needs of the teacher. To date, most research efforts in adaptive systems have focused on the students' tutoring and the learning processes, i.e. the recipients of the content. Yet, the contributors of the content, teachers have their own needs when they create the course online. Their demands, which are different from students' requirements, haven't received much attention yet. Currently, most successful courseware authoring products, such as WebCT and BlackBoard, display some degree of adaptive natures and customization when students have access to the course contents. But none of them provides intelligent facilities to help the teachers to create the online education site [Bayne & Cook, 2002], and none of them help the teachers to select the right teaching

strategy in Web environment. In this case, the user model or teacher's model will play a very important role in helping the teacher to create the online courses.

The second problem is the excessive focus on the content sequencing, indexing and selecting in the current adaptive hypermedia system and less attention has been paid to the presentation of the content [Brusilovsky, 1998a]. In these systems, the course developers and teachers either define the navigation styles and choice of document formats for each of the learning units [De Bra, 1999], or otherwise rely on the system to set up the presentation automatically in terms of a few pre-defined templates which are common to all course content units [Carro, et al. 1999]. The teachers of Web-based educational systems find it hard to modify the presentation unless they have enough knowledge on content selection, the architecture of the adaptive systems and underlying structure of student models.

Another problem is the exclusion of the teacher's role in the learning process. In terms of traditional teaching theories, such as "Objectivism", teachers are responsible for selecting teaching methods, setting scenes, providing content, managing the curriculum, and overseeing the learning progress. It is said that the teacher's teaching style and methods are more important than student's learning style [Kinshuk, Patel. Oppermann, Russell, 2001]. In most Web-based educational systems, there is no face-to-face, one-to-one direct interaction between teacher and students. The interactions between teacher and students normally happen in indirect forms that are mediated by the systems, such as email exchanges, or online chat. Teachers usually stay behind the scene. As a result, the interaction between students and content outweighs the interaction between students and teacher. When students interact with the content over the system, the teacher model which represents the human teacher will oversee the student's learning process to provide adequate guidance when necessary.

The lack of a uniform format for data interchange is a common deficiency in the commercial courseware, document authoring and intelligent tutoring systems. These systems normally consist of several modules which exchange data to implement a task. Take the example of the intelligent tutoring system, the content data (document

fragments) and student model data are stored in a database. This data is extracted from the database (using structured query language), calculated in the control program (vendor-specific), and presented in the user interface with a predefined document format. All of these data formats and document format are vendor-specified. It presents some difficulties in integrating one system with another, or exchanging data with another. If the extensible mark-up language XML is adopted, it could make the data exchange much easier [SUN, 2002], as it offers a common, platform–independent, standards–supported basis for formatting data. If standards for educational content formatting based on XML technology, such as IMS [IMS, 2003], are adopted, these standards will definitely have a strong impact on commercial courseware, document authoring and intelligent tutoring systems.

In order to design the robust Web-based educational systems, we need to overcome the shortcomings of the intelligent tutoring system, commercial courseware authoring tools. Templates have been widely accepted in multimedia authoring and commercial document authoring (MS Word, PowerPoint). When templates are described in XML rather than vender-specific formats, they provide a more lasting embodiment of reusable content, because their useful lifetime is no longer bound to the lifetime of the applications used to create them.

In this thesis, we consider the application of templates to the educational content creation. In Web-based educational content creation, templates provide a high-level description of the presentation structure and relations of domain knowledge in the absence of content. Template can also be broken down into template fragments. Template can carry the template fragments attributes such as the name of the template fragments, the incidence of the fragment, and the sequence of the template fragments. As long as we can find out the relationships between the templates, template fragments and teachers' behaviour when they interact with the system, templates and usage patterns can be used to define and train the teacher model. In addition, it also provides a mechanism for sharing and distributing the templates, thus sharing and reusing presentation design ideas.

Here, we suggest a Template-Driven Teacher Modelling (TDTM) architecture which supports the course authoring by providing a higher degree of control over the creation of presentation. The collection of accumulated templates for a teacher or a group of teachers is selected as the inputs for the inference mechanism in the teacher's model to figure out the best representation of the teaching strategy and then to predict teacher's intention when he or she interacts with the system. Moreover, the presentation templates are kept to support the re-use of the on-line content at the level of individual screens with the help of Template Server. With Template Server Architecture, teachers will be able to view the course content in the designated template and document format. Thus, we are filling in what appears to be a gap in the current practice and research in the area of courseware authoring [Tretiakov & Shi, 2003].

1.2 The objective of the research

As has been stated above, the main objective of the research is to enrich the Web-based teaching theories by introducing the template-driven teacher modelling (TDTM) architecture. Based on this architecture and the Template Server Approach, two prototype systems will be partially built to test the new approach. More specifically, these prototype systems will be used to prove the feasibility of the architecture, and collect user's attitudes towards the architecture and template server approaches. Before the formal usability test is initiated on these prototype systems, we will carry out an evaluation on these prototypes. The feedback from this evaluation will help enhance the architecture, supplement the template server approach, and consolidate the online teaching theories.

As long as the teacher model is embedded into a course authoring system, it will greatly reduce a teacher's effort in authoring course content which can be easily adapted to the teacher's teaching style or method. It also allows the teacher to make selections among a variety of templates and multimedia presentation formats (hypertext, graphics, animation, audio, video) or build their own templates in terms of the natures of the discipline and course content, which implicitly represents teaching methods (lecturing, collaboration-oriented, simulation-based, exploration-based etc.).

1.3 Improvement of Adaptability

In order to cater for the teacher's needs in a Web-based education course, the system must possess adaptivity (the system automatically adjusts to the individual teachers) and reusability. It is widely accepted that the intelligent tutoring systems, present two levels of adaptation: content level, and link or navigation level adaptations [Brusilovsky, et al. 1996]. In the first level of adaptation, the system will decide what content will be presented to the user in terms of the user model and task. In the second level of adaptation, the links or navigation to other content documents will be put into certain sequence, be annotated with different colours, icons or dimming, or be disabled in terms of user model (knowledge level goal and task).

All of these adaptation techniques can be adopted in the proposed Template-Driven Teacher-Modelling (TDTM) architecture for the course authoring systems. Instead of manipulating the course content, TDTM focuses on dynamically presenting templates in terms of the teacher model. The course content unit in the intelligent tutoring system contains content fragments. Similarly, template in TDTM consists of template fragments. Various relationships, association rules, clustering can be mined from the usage of templates and template fragments. These rules will be used in the teacher model to implement the adaptation. Like the Intelligent Tutoring Systems, TDTM also possesses two levels of adaptation. In the first level of adaptation, TDTM will decide how the template looks and which group of template fragments will be integrated into the template in terms of teacher's model. In the second level of adaptation, the system will define in which way templates can be found. For example, putting them into certain sequence, or annotating them with different colours, icons or dimming, or disabling them in terms of the teacher model.

1.4 The modelling methods and techniques

Teacher modelling starts with data acquisition (mainly from observing requests to the template repository) regarding to teacher's behaviour when he or she interacts with the system. Then based on the data, a set of data mining techniques will be used to discover association, classification or clustering rules. These rules or algorithms will be used to predict the teacher's actions. Data mining is a process of supervised or unsupervised discovery of interesting, useful, and previously unknown information from this set of data [Kumar, 2002]. The traditional methods and techniques for data mining include classification, discovery of association rules, and clustering. Classification relates to grouping interaction sessions, association rules relate to predicting the usage patterns within a session, and clustering refers to the dynamic approaches to classification, which do not use explicit information about the user.

One of the techniques to facilitate the teacher model is to classify teachers based on teacher's attributes and usage history to the templates and template fragments. Then the system can make predictions about the teacher's intention when he or she interacts with the system. The data of templates and the navigation to (or the selection of) the templates will depend on the settings in these attributes. Some of the potential teacher's attributes are: native language, familiarity to the multimedia authoring tools, teaching experience and teaching history (the number of years in teaching, tertiary or secondary sector). Moreover, a number of environmental attributes, such as organizational attributes, geographical attributes and orientation attributes (space, temporal, belief), could also be associated with teachers and have their impacts on the system by influencing the teacher's behaviour [Iivonen & Sonnenwald, 1998].

However, some of the teacher's attributes are very difficult to describe and quantify in an explicit way, such as different teaching styles (theory first or practice first, procedural or declarative), teaching philosophy (Instructivist or constructivist), social and cultural backgrounds. There are wide variations from individual to individual even though they all belong to the same user group. Furthermore, some attributes show certain dynamic natures. It means that they are changing from time to time when

teachers change their goals and tasks. Modelling implicit aspects of this teacher model presents a real challenge to researchers and developers. The template server approach adopted in this research will take into account the possibility to retrieve implicit information from teacher behaviour as observed by the system by using a decision engine such as Bayesian networks.

We create an architecture (Template-driven Teacher Modelling) which incorporates the template server and teacher model. As the changes of teacher's mind and preferences will be reflected in the usage of the templates, the templates become the key components of template-driven teacher modelling architecture.

1.5 Description of the research

Generally put, a system based on the template-driven teacher modelling architecture will assist teachers in creating or re-constructing the online courses using templates that reflect basic preferences of the teacher, and represent different teaching styles and tasks. The teacher model will play its role reflecting the teacher's preferences and making adequate predictions. The templates will be adapted to represent both the explicit teacher's preferences, and the implicit teacher's teaching style.

In terms of the implementation of teacher model, both implicit and explicit aspects of the teacher model could be stored in the same logical and physical layer on the architecture. It could be stored in a relational database, or embedded in an XML file. We opted for XML because it does not bind us to a particular database management system and it offers a format which can be readily consumed as a data source by a variety of systems without raising the portability issues.

The proposed system will work in the following way. When a teacher logs into the system and initiates a task, several templates appropriate for the task will be offered for selection according to the default data stored in the teacher's model. After one template has been selected and filled with course content, the course content will be stored in a database or an XML file. The selections of templates become part of the teaching

activities. The teaching strategy is a particular combination of teaching activities put together in a sequence [Dee Fink, 2003]. Therefore any major changes in the use of templates may suggest the changes of teacher's teaching strategy. Consequently, the teacher's model will be updated to reflect these changes. The teacher model is maintained in teaching strategy level. The rationale is that teachers could change the teaching strategy in terms of the nature of the content, the academic level and cultural backgrounds of the students [Aikenhead, G. S., 1997]. Considering the system at the highest level of generality, we do not distinguish the teacher model and the data behind it, as we envisage that the teacher model can be modified not only by changing the parameter values, but also by replacing the components and the implementation algorithms. This is the spirit of the Template Method and similar design patterns [Gamma, et al. 1995].

Additionally, the course contents in the database or XML file are globally accessible and reusable. They can be rendered into different document formats (MS Word, PowerPoint, HTML). Moreover, as we are using semantic mark-up to store the course content, any new formats that may emerge in the future can be easily accommodated.

1.6 Schedule of the research

This research project consists of a number of clearly defined stages. In the planning and investigative stage, current literature on Web-based teaching systems and intelligent or adaptive tutoring systems are reviewed. The main purpose of this stage is to be familiar with the research backgrounds, and to initiate the research plan. In the following requirement analysis stage, we analyze and identify the strengths, weaknesses of current Web-based educational systems, and formulate the requirements for the proposed template-driven teacher modelling architecture. Then the framework of the proposed template-driven teacher modelling architecture and the template server are discussed. In the design stage, we seek to answer the question "How will the architecture and the template server approach address the weakness of most Web-based course authoring systems?" The key component of the prototypes is a template server, which serves the template on demand. We also describe the requirements analysis for client applications

that play the vital role in Template-driven teacher modelling architecture. In the implementation stage, software development tools and programming languages, such as Java, JavaScript, XML, XSLT, HTML, CSS, XMLDOM, are used to build the prototype systems that facilitates the teacher model and the template server which serves the templates.

1.7 Expected outcomes

The ultimate outcome of this research is to construct the Template-driven Teacher Modelling Architecture, and to evaluate the template server approach. The architecture is used as the guidelines to build prototype systems (not complete systems) that demonstrate how templates could be served adaptively. Because the templates usage patterns are being used as an input to the teacher model, the template server becomes the key component to implement in this research project. The template server approach is evaluated with the end-users to identify its usefulness and weakness to suggest further improvements. To ensure feasibility, the prototype will address a limited implementation of the Template-driven Teacher Modelling Architecture, as far as the scope of support to teachers is concerned. On the other hand, the architectural discussions will be conducted at a high level of generality.

1.8 The organization of this research thesis

This thesis is organized in the following order:

- In Chapter 1, we outlined the research questions and objectives, schedule, and expected outcomes.
- In Chapter 2 "Education Theories", we take a close look at the teaching and learning processes in context of Web-based education. We also discuss the debate over teaching paradigms such as "Instructivism", "Constructivism" and "Social-cultural theories" within the education community. We focus on their impacts on education practice and their implications for Web-based educational system.
- In Chapter 3 "Web-based educational systems", we discuss the categorizations of Web-based educational systems, such as course authoring and tutoring

systems. Special attention is given to the common features of commercial courseware packages and concept-based research projects. We also address the latest intelligent and adaptive techniques in Web-based educational systems.

- In Chapter 4 "Modelling techniques for Web-based educational systems", the basic user groups of the Web-based educational systems are identified, teachers and students. Various student and user modelling techniques will be discussed extensively. At the end of the chapter, we outline our approach to teacher modelling by suggesting the Template Driven Teacher Modelling architecture.
- In Chapter 5 "Template Server Architecture", Template Server is described in detail. Some of the terms appear frequently in this chapter, such as Template Server, Template Engine, Template Repository and Content Server. The UML (Uniformed Modelling Language) will be used to describe each component in the architecture and the interactions between them.
- In Chapter 6 "Technical Implementation", we discuss the technical issues related to the implementation of the prototype. The role of XML, deployment of components of Template Server, and installation of software packages are described in details.
- In Chapter 7, we design and conduct a user evaluation which is to check the viability of the Template Server Approach and we analyze the results of the evaluation.
- In Chapter 8 "Conclusion, Discussion and Future work", we conclude the research and highlight the main contributions. Then we explore the possible improvement to the proposed Template Server Architecture.

Chapter2 Education Theories

2.1 Introduction

In this chapter, we will first take a close look at the debate over the pedagogical paradigms such as "Instructivism", "Constructivism" and "Social-Cultural theories" within the education community. Then, we will discuss their impacts on education practice, their implications for Web-based education and their implications for our Template-Driven Teacher Modelling Approach.

Tertiary educators feel the pressure to place their courses on the Internet. However, it is widely accepted that using new media requires new approaches to teaching [Horton, 2000]. Putting course content online is more than a matter of converting the syllabus to HTML and placing it on a server. There are important considerations on which content should or should not be placed online, on how it is to be sequenced or organized for navigation, and which authoring tools works best to reach a specific teaching goal. Finally, anyone has to be able to incorporate the knowledge and the know-how accumulated in conventional education by considering the established and the emerging education theories and the existing guidelines to pedagogical research and practice.

In the context of Template-driven Teacher Modelling Approach, education theories will assist our research in the following aspects:

- Clarify the roles and interaction patterns of the participants in the education environment. This helps establish the requirements for the Template-driven Teacher Modelling Architecture
- Assist the classification of teachers in teacher modelling.
- Identify the potential attributes (including social and cultural context) which could affect the selection of teaching strategies.
- Consider possible approaches to the prediction of a teacher's intention.

2.2 Fundamental education processes

Fundamental education processes involve teaching and learning. Currently, there are many schools of thought in education, such as Instructivism, Constructivism, and Social-Cultural theories. Most theorists agree that interaction is the single most important component of any education processes [Vygotsky, 1978]. Moore identified three types of interaction in learning process, student-teacher, student-content and student-student [Moore, 1989].

In this research, we will focus on teacher modelling in the teaching process. Contrary to Moore's discovery, who considered interactions from a learning perspective, we are considering the interactions from a teaching perspective. Therefore, we concentrate on the following three types of interaction in teaching: teacher-student, teacher-content, teacher-teacher.

The direct interaction between teacher and student normally takes place in classroom in the traditional way of teaching. Teacher delivers the instructions, lecture sessions, gives feedback, and asks questions. In on-line learning, the teacher-student interaction is normally indirect, achieved via content either published ("push" model) or delivered in response to student's questions ("pull" model). Ideally, teacher model should serve as an intermediate agent, representing the teacher.

The interaction between teacher and content can be seen when a teacher creates the course content in the form of lecture slides, assignments, and readings. This type of interaction will draw particular attention when we study the teacher modelling in the following chapters. In particular, template server architecture will assist teachers' interaction with the content, and retrieval of information necessary to construct and to update the teacher model.

The interaction between teacher and teacher is considered as a kind of collaboration among teachers to reach a common teaching goal, such as academic meetings, seminars, and discussion related to the teaching. In the context of the architecture we propose,

teacher - to - teacher interaction is realized implicitly, by exchange and aggregation of teacher model data, and explicitly, when teachers share the templates they create.

An implicit interaction between student-teacher can also be achieved by exchanging data between teacher and student models.

In the context of the Web-based education environment, all the interactions will happen on the Internet. The Internet and the system built on it can be looked as the medium that supports the interactions between teacher and students, between teacher and content, and between teacher and other teachers. Another difference between the traditional teaching process and the Web-based teaching process is that the amount of the direct interaction between teacher and students will decrease [Terzi, 2003]. As students will rely heavily on the online content, teachers in distance education are expected to spend more effort on maintaining the course content than on conducting traditional classroom teaching.

2.3 Education theories: Instructivism, constructivism and socio-cultural theories

Over a few decades, a hot debate has been raging about the very basis of teaching theories. Instructivism, Constructivism, and Socio-Cultural theories are the three most influential paradigms [Jonassen, 1991a] [Vrasidas, 2000] [Anderson, et al. 1997]. Studies of their characteristics and of differences among them will provide the research basis for any educational practice.

2.3.1 Instructivism

Historically, teachers have been using the "Instructivist" methods which dominate the process of "knowledge transfer" [Jonassen, 1991a]. The students acquire the knowledge by regurgitating the course content that is presented by the teacher. The teacher is responsible for selecting the content, setting the scenes, choosing the most effective teaching strategies and assessing the student's achievements. This teacher-dominated model is based on "behaviourist psychology". Behaviourists view psychology in terms

of resulting behaviours that can be modified by forming associations between courses of action and their consequences (such as reward and punishment). Teachers taking the instructivist methods are said to treat students as passive information processors. Teachers feed students with information. Then, they monitor and evaluate the feedback from students to adjust the amount and the way of information that is delivered. The Behaviourist movement and the Instructivist teaching dominated American education from about 1920 to 1970 [Hofstetter, 1997].

2.3.2 Constructivism

In the second half of last century, a new trend appeared in education practice to overcome the weakness of traditional education approaches. It is called "constructivism" [Smith, 1999] [Wonacott, 2000] and is based on "cognitive psychology". Under this learner-oriented model, students are viewed as active processors of information and constructors of their knowledge [Rumelhart, 1981]. The Constructivist view of education is particularly relevant to active learning, adult learning, and self-directed learning

According to the principles of constructivism [Brooks, et al. 1999]:

1. Learning is a process of searching for meaning by the learner. Therefore, learning must start with the issues around which the learners are actively trying to build the knowledge by themselves. The educators and the system developers must realise that there are multiple truths or realities, and encourage the learners to construct their own knowledge about the reality.
2. The context is crucial in any constructivist approach. Gaining knowledge requires understanding the domain and the relevant context. The learning process focuses on the primary relevance concepts, not the standalone and isolated facts.
3. In order to teach well, the teacher must place the emphasis not only on the students' prior knowledge but also on his cognitive processes in building the knowledge.

Teacher-Dominated Perspective	Cognitive Perspective
Teacher Centred	Student Centred
Teachers Present Knowledge	Students Discover and Construct Knowledge
Students Learn Meaning	Students Create Meaning
Student as Memorizer	Student as Processor
Learn Facts	Develop Learning Strategies
Rote Memory	Active Memory
Teacher Structures Learning	Social Interaction Provides Instructional Scaffolding
Repetitive	Constructive
Knowledge Is Acquired	Knowledge Is Created
Teacher Provides Resources	Students Find Resources
Individual Study	Cooperative Learning and Peer Interaction
Sequential Instruction	Adaptive Learning
Teacher Manages Student Learning	Students Learn to Manage Their Own Learning
Students Learn Others' Thinking	Students Develop and Reflect on Their Own Thinking
Isolationist	Contextualist
Extrinsic Motivation	Intrinsic Motivation
Reactive Teachers	Proactive Teachers
Knowledge Transmission	Knowledge Formation
Teacher Dominates	Teacher Observes, Coaches, and Facilitates
Mechanistic	Organismic
Behavioralist	Constructivist

Table 1. Comparison of the Teacher-Dominated and Cognitive Perspectives on Education [Brooks et al., 1999]

2.3.3 Two groups of participants in the education process

The radical Instructivism and the radical constructivism are the two extremes in the educational community, from teacher-dominated to student-centred, from behavioural control to cognitive concept building. Actually, most education theories stay in between these two extremes [Vrasidas, 2000]. Sometimes they appreciate a more objective approach. Sometimes they appreciate a more constructivist approach. It depends on the cultural, organisational context, discipline content, resources, and the academic levels of students. Even within the constructivism itself, authors, researchers and theorists articulate differently the constructivist perspective by placing various emphases on the roles of the student and the teacher. In the way in which knowledge is conceived and

acquired, the types of knowledge, skills and activities emphasized and how goals are established.

Nevertheless, the teachers and the learners (or students) are the two important groups of participants in this educational process. Although some educational systems appear to be more teacher-centric (as the Instructivism), while others are student-centric (as constructivism), both have to be present. Understanding the role of teacher and learner from the perspectives of the instructivism and the constructivism provides a useful vantage point from which to grasp how the educational theories impact on pedagogical practice.

2.3.4 The weakness of constructivism

The radical constructivism has led some educators to believe that it supports spontaneous, uncontrolled learning, in contrast to the systematic, organized instruction of knowledge employed by the Instructivist tradition. The open-endedness of constructivist questioning can be daunting for the entry-level students. Similarly, it might be hard for the teacher to incorporate radical constructivism into her teaching strategies, as students will in charge of their own studies, and the teacher may feel uneasy about losing control. The efficiency and the reliability of the evaluation methods are also questioned, as the constructivist learning environments are difficult to evaluate. Finally, in view of radical constructivism that knowledge cannot be transferred, but has to be constructed by the student; therefore each student works individually to build his or her perception, and does not take advantage of the teacher's guidance and classroom collaboration. This contradicts the Socio-Cultural theories (to be covered in the following section) in which the learned concepts (knowledge) evolve in the process of social interaction [Vygotsky, 1978].

2.3.5 The Third Voice: Socio-Cultural Theory

It has been pointed out that constructivism raises serious questions from an epistemological position [Noddings 1990]. It is hard to believe that all the individual

constructions can be considered as "knowledge". Professor Noddings gives an example of a student, called Benny, who had developed a particular process of calculation, which satisfied himself. But from the viewpoint of professional mathematician who has wider experience and knowledge, this process could be seen as inadequate. Is there any sense in which Benny's process could be regarded as "knowledge"?

In Vygotsky's view [Vygotsky, 1978], peer interaction, scaffolding (by the teacher), and modelling (models are provided by the teacher for a problem solution) are important ways to facilitate individual cognitive growth and knowledge acquisition. Socio-cultural theorists are challenging the radical constructivism putting stress on individual perceptions [Lave & Wenger, 1991]. In their opinion, students often have different views of a situation. If these views seem incompatible, inadequate and incorrect, there is a need for reconciliation which can lead to the social mediation of individual knowledge, such as correction by teachers.

There also have been widespread debates on the question such as whether people growing up under different cultural circumstances would differ in the basic intellectual capacities that they develop [Luria, 1979]. In Luria's experiments, she found that "changes in the practical forms of activity, and especially the re-organization of activity based on formal schooling, produced qualitative changes in the thought processes of the individuals studied." Moreover, "the basic changes in the organization of thinking can occur in a relatively short time when there are sufficiently sharp changes in social-historical circumstances".

In the context of education, both the teacher's teaching and the student's learning activity involve certain thought processes. In these thought processes, teachers will choose the most effective teaching strategy and the most appropriate content influenced by his or her cultural and social circumstances. The same will be true for students.

2.4 Impacts on teaching and learning processes

The influences of Instructivism, Constructivism and Socio-cultural theories on educational community are extensive, and have affected the development of curriculum, teaching methods and assessment; and the evaluation of their impacts is largely driven by common understandings of what constitutes the reliable indicators of the successful education practice.

2.4.1 Curriculum

According to the Instructivism, teachers are the "God of knowledge". They represent the unchanging, structured external world. Consequently, most institutions have very tight curricula. They provide the basis for organizing the course content in a predefined sequence. As the content that represents the external world is well structured, it can be divided into small manageable units. However, it greatly reduces the flexibility of individual educator to respond to new opportunities and challenges. The designer of the new Web-based educational systems should overcome the rigidity of traditional curriculum to provide flexible features to teachers and students.

The Constructivism calls for the elimination of a standardized curriculum. Instead, it promotes using curricula customized to the students' prior knowledge. Also, it emphasizes hands-on problem solving skills.

The socio-cultural theorists think that knowledge is acquired through social interaction. In their curricula, attention is given to both content-specific and interdisciplinary aspects of courses, such as integrating natural science and social science contents. Students are encouraged to take courses which combine both theory and practice, involve collaborative activities which affect their studies within the school and beyond.

2.4.2 Teaching and Instruction methods

The Instructivists claim that teachers don't have to change their teaching methods unless they are convinced by observable and measurable outcomes [Burns, 1980]. All students should acquire the knowledge in the same way. The Instructivist teaching

strategy normally starts by presenting simple concepts and then proceeds to the more elaborate ones [Abtar, 1997]. Their strategies rely on courseware using the concept-example-practice sequence pattern. The questions are normally close-ended.

Under the theory of constructivism, teachers focus on students making connections between facts and thus achieving new understanding, rather than on memorizing the facts. Teachers tailor their teaching strategies for each student and encourage students to explore, analyze, interpret, and predict information [Jonassen, 1991b].

Socio-cultural theorists suggest that teachers should promote extensive dialogue among students and between teachers and students. This is often called cooperative learning strategy [Ronkowski, 2002]. As social and cultural settings may heavily influence both teacher's teaching and student's learning processes, education administrators should offer opportunities for teachers to diverse their teaching strategies. For example, they could allow teachers to mix and match their teaching strategies in collaborative teaching. On the other hand, students should be provided with the course content in different formats and in different learning environments.

2.4.3 Assessment

The Instructivist assessment strategies are mainly based on criteria. The teacher measures the student's performance in terms of a predefined set of criteria. The purpose of assessment is to help student reinforce memory and identify areas where improvement or correction are needed.

The Constructivism calls for the elimination of grades and standardized testing. Instead, assessment becomes part of the learning and analysis process so that students play a major role in judging their own progress.

The Socio-cultural theorist's view of learning assessment is regarded as more broadly based than the Instructivism and the Constructivism, and involves peer assessment and teacher's comments.

2.5 Finding the balance

Teachers, irrespective of the role they play, whether instructive or supportive, are facing the dilemma of which paradigm to use. Actually, most education theories stay in between the extremes [Vrasidas, 2000].

In real practice, researchers of education theory try to find the balance between any two or among these three examples. For instance, the social constructivism is the balance between constructivism and socio-cultural theory, in which a student constructs understanding through many channels: reading, listening, exploring and experiencing his or her social and cultural environment [McMahon, 1997].

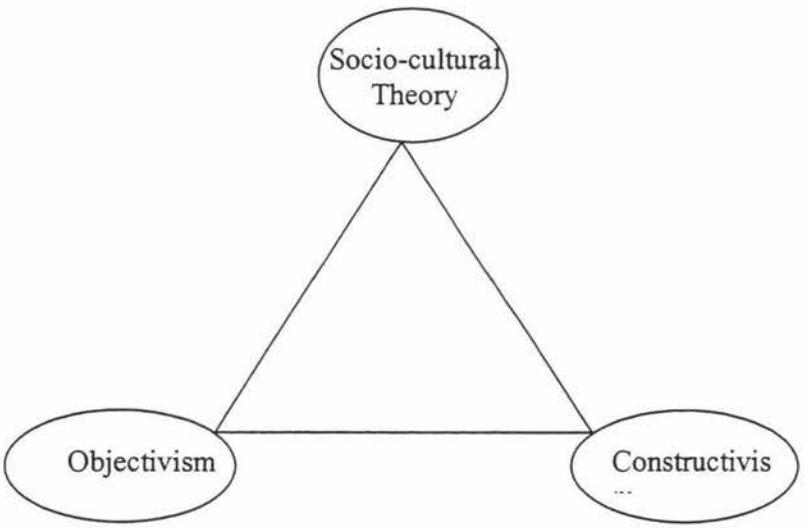

Figure 1. Three poles of education theories.

In response to teacher's commitment to accommodate the required course content, externally applied assessments and student's cognitive characteristics, most educators are finding the appropriate balance between the existing Instructivist instructional methods and the new constructivist instructional methods [Dick, 1995].

The balance has to be constantly reassessed, because the social, cultural, and organizational contexts are changing from time to time.

2.6 Implications for Web-based Education:

As far as the educational delivery is concerned, we are still at the early stage of the Internet age, as people gradually discover the potential of the Internet as education media. The distributed, asynchronous, collaborative natures of the Internet make it an ideal tool to store and deliver the knowledge content, making it valuable from the Instructivist perspective. At the same time, the Internet is a powerful tool for the knowledge search and navigation (the Constructivist perspective), and for the communication (the Socio-cultural theory perspective).

2.6.1 New technology asks for the changes in education theories

The Internet offers profoundly new approaches to the learning and teaching:

1) The Internet provides the direct access to the information and the course content anytime and anywhere, thus reducing the need for the teacher to be a lecturer of a course;
2) The Internet provides new forms of social interactions (e-mail, chat, discussion forums, video conferencing);
3) The Internet creates a new educational environment which supports the teamwork and collaboration.

These changes are gradually affecting the existing practices, and redress the balance between the three main paradigms in education.

2.6.2 Various teaching and learning models on Web-based education

For Internet-based learning, education theorists proposed various teaching and learning models based on the Instructivism, Constructivism and Socio-cultural theories. For example, "Symbol processing model" [Sherry, 1995] is one of the Instructivist models

in Web education. The Web course designer set all of the instructional objectives. It is assumed that all of the Web course learners participating in the instruction are either intrinsically or extrinsically motivated to learn the behaviour set in the instructional goal. Individual differences among learners are often either ignored or generalized.

On the other hand, the "Situated learning model" [McLellan, 1996] follows from a constructivist model of learning in which students actively construct their own knowledge. Another similar model is "Cognitive Flexibility and the Hypermedia Design Model". It suggests a metaphor of a "crisscrossed landscape" with its suggestion of a nonlinear and multidimensional traversal of a complex subject that is well supported in the Internet environment, especially using the hypermedia of the World Wide Web in conjunction with one of the Net's discussion facilities [McManus, 1995].

Berryman [Berryman, 1993] introduces the cognitive apprenticeship model as a means of delivering instruction via the Internet. Previous studies [Bransford, et al. 1989] have shown that traditional approaches to instruction (such as readings, lectures, and demonstrations of key points) often focus on declarative and procedural information. As a result, it produces "inert knowledge". Inert knowledge refers to previously learned knowledge that is not spontaneously applied to a relevant problem. In other words, students may learn something in one context but may not be able to apply it to a relevant problem in another. Cognitive apprenticeship [Collins, et al., 1989] is an instructional innovation which was introduced to address the problem of inert knowledge. This approach is based on the underlying principles of apprenticeship learning and focuses on the use of such strategies as modelling expert behaviour and coaching students to mimic expert skills until they are competent in their performance.

Social interaction and cultural diversity considerations have also drawn a lot of attention in Web-based education. "The design of Web-based instruction is not culturally neutral, but instead is based on the particular epistemologies, learning theories and goal orientations of the designers themselves." [McLoughlin & Oliver, 2000] Recently, education professionals have argued for a cultural dimension in the course design process and the need to provide culturally sensitive learning environments.

2.7 Influence on Template-Driven Teacher Modeling

Though most proposed new teaching and learning models are aiming at solving some deficits which come to existence when applying the traditional education theories (mainly Instructivism) to the emerging Web-based or computer-mediated education environment, nobody claims the full elimination of teacher's role in the teaching process.

The Template-Driven Teacher Modelling assumes that teachers are using templates to create course material. Their choice of templates will be dictated by the prevailing paradigm to the teaching strategies they are going to take. We assume that templates can be created in support of each of the approaches. Hence, the template selection provides us with information on the teachers' preferences in that respect. Once teacher preferences are established, the appropriate guidance can be provided. Teacher preferences are constantly re-evaluated, so that an adaptive response can be provided dynamically. It should be noted that teacher is not "assigned" to a particular paradigm, but rather the appropriate balance is reached by offering a mix of templates.

At the same time, certain explicit teacher attributes may also be of significance. For example, for teachers applying the socio-cultural paradigm, the cultural backgrounds of teachers as well as those of the students may be taken into account when the teacher is prompted for template selection. Here are the justifications. We have made the assumption that the selection of templates is influenced by the choice of the teaching strategy. Therefore, when the teacher changes the teaching strategy in terms of the nature of the content, the academic level and cultural backgrounds of the students [Aikenhead, G. S., 1997], it will have an impact on the template selection. On the other hand, according to Luria's discovery [Luria, 1979], any changes in the practical forms of activity, and especially the re-organization of activity (such as changes of teaching strategy and selection of templates) are based on formal schooling and produce qualitative changes in the thought processes of the individuals studied.

Instructivism, constructivism, and social-cultural theories put their own emphasis on various interactions between the participants in the education process. Yet, none of them sufficiently takes into account teacher-teacher interactions (although socio-cultural theory does cover them to a degree). We believe that with the advent of Internet, teacher-teacher interactions will gain in importance, particularly via sharing and exchange of course material. Therefore, the Template-Driven Teacher Modelling architecture incorporates teacher-teacher interaction at two levels which has been outlined in section 2.2.

Chapter3 Web-based educational systems

3.1 Introduction

Web-based educational systems exhibit different images to the users of systems. Students view the systems as the learning environment. They use the Web-based educational systems to learn the knowledge. Teachers view the systems as the teaching environment. They normally use the Web-based educational system to publish the course content. Benefits of Web-based education are: the independence of course authoring, teaching and learning with respect to time and space, hypermedia and cross-references which allow the students and teachers to "jump" from one part of the course to another part of the course as desired, no limit on the number of users and unified client application (Web browser). Web-based educational systems installed and maintained in one place may be viewed by a huge number of users all over the world. The evolution of a Web-based educational system can be divided into a number of stages. Initially, they are only the replication of lecture notes that require very limited involvement of the teacher. The effect of these systems is disappointing because the course contents are static and there are no teacher-to-student, and student-to-student interactions. From the perspective of education theorist, these systems do not meet requirements of teachers who adopt either Instructivism or constructivism. This generation of Web-based systems is quickly replaced by systems with multimedia and simulation. The systems, which are rich in multimedia and simulations, are the ideal tools to implement constructivist model [Dabbagh & Schmitt, 1998]. Students and teachers are provided with much more freedom and options to interact with the content. However, these systems require the increasing investment from the teacher to create more complicated multimedia and simulation objects. The latest Web-based educational systems incorporate intelligent or adaptive facilities, such as student model and teacher model, to accommodate individual requirements from students and teachers. It is becoming one of the fastest growing areas in educational technology research and development [Khan, 1997].

In this chapter, we will review some commercial Web-based educational systems and concept-based research projects to discuss the basic features that these systems provided. We will be focusing on how these systems improve the interactions between teachers and students, students and students, teachers and content, students and content.

3.2 Web courseware tools

Web-based courseware tools are the good examples of Web-based educational systems. They can be further put into two categories: commercial courseware packages and specific concept-based research products. The commercial courseware packages are developed to meet general requirements from the tertiary educators. They focus on the system reliability, comprehensive functionalities and cost effectiveness to support day-to-day, multi-discipline pedagogical requirements. The primary users are tertiary students and educators. Another group of web-based course authoring systems are the specific concept-based research products which are more exploratory and experimental. They concentrate on the exploration of the new architectures to renovate and supplement the current education practice.

3.2.1 Commercial courseware packages—WebCT vs Blackboard

Firstly, let us look at the commercial Web-based courseware packages. WebCT and Blackboard are the most widely used software for the implementation and management of Web-based educational environments [Bayne & Cook, 2003] [Britain & Liber, 2000]. More than 2600 Institutions in 84 countries currently are licensed to use WebCT [WebCT, 2002]. Both of these packages can be used to create entire online courses, or to simply publish materials which supplement existing courses. In terms of the support to the teaching and learning processes, these software packages provide the necessary means to facilitate the interactions between teachers and students, teachers and content, students to content. WebCT and Blackboard come with the following tools: bulletin board, online chat, online quizzes, calendar, self-evaluation, threaded discussions, synchronous communication (real-time chat and whiteboard), student assessment tools, and collaborative work groups.

3.2.1.1 The tools for teachers

The tools for teachers in the courseware tools are designed to support the teacher's routine commitments related to the teaching process, such as the course content creation, academic assessment, and student management. In general, these teaching processes involve the interactions between teachers and the course content, teachers and students, teachers and other teachers.

1) Content Creation

In the distance education process, the basic interactions happened between teachers and content. The creation and modification of course content becomes the key function of a Web courseware package. It means that the teachers can easily publish and modify the course syllabi and handouts without knowing anything about the underlying technologies, such as HTML coding or Web scripting. Correspondingly, links between the course syllabus and the calendar should be automatically incorporated into the materials created. Courseware usually accepts plain text (such as ASCII and JIS) or graphics (such as GIF and JPEG), and automatically encodes it for HTML display.

2) Gradebook

As in Instructivist, grades are an essential means for modifying student behaviour, and this view is normally supported by deeply engrained regulations and practices, it is imperative to have some tools that help teachers manage grades. Grading information is often stored in a grade database. Then the data are presented by various reporting facilities such as spreadsheet printouts. Courseware can serve as a communication channel allowing students to see in on-line grade-books both the grades and the relevant teacher's comments.

3) Assessment Tools

Online assessment is becoming more important as teachers are becoming more experienced with Web-based teaching and instruction. It is bound to enhance the interactions between teachers and students. Courseware products usually include some forms of online test and examination. For example, a multiple choices test, randomized tests with test banks combining multimedia files and text for testing. Access to tests can be controlled so that students have only one opportunity to take each test. The tests can be timed. Ideally, several item formats for the same concept are available for students to choose from. For example, student can choose to sit the pure textual test or test with animation and sound.

4) Student management tools

Online tracking allows teachers to keep records about the student's use of courseware. These records include date and time of access, time spent in the system, number of postings to the bulletin board, the number of articles student has read and the difficulty level of the article. This information is crucial for assessing the academic progress of students.

3.2.1.2 The tools for students

The tools for students are the courseware tools designed to support the student's day-to-day learning processes, such as the file exchange, asynchronous communications, and collaborations.

1) File Exchange

Students submit assignment papers for grading. Teachers and instructors grade, comment upon, and return the assignment papers. Courseware systems develop strategies for supporting these exchanges. Effective courseware tools provide the means

for facilitating these exchanges. Also, it usually permits teachers to comment on assignment papers directly within the program, so that teacher comments stand out when the papers are returned to the student.

2) Asynchronous Communication (bulletin, discussion board and mailing list)

In asynchronous discussions, students can come and go whenever or wherever they like. It is said to be a principal advantage of Web teaching. One way to handle this is to create a mailing list. Any student who sends an email to the automatic mailer will end up mailing to all other students on that list. An alternative is to form a news group (bulletin or discussion board). The principal difference is that users of newsgroups peruse titles of messages and choose which ones to download. While subscribed to a mailing list, students automatically receive all messages posted. In a newsgroup, students may choose not to download the message at all.

3) Synchronous Communication (Real—Time Chat and video conferencing)

Teachers value in-class discussions, and often see asynchronous discussions as deficient. In the Internet world, so-called chat-rooms have filled the need of those who seek real time exchanges. Courseware often includes synchronous communication features, known as chats. Another form of synchronous communication is video conferencing.

Advantages of video conferencing are obvious. It allows visual communication and creates social presence and a comfortable environment for teaching and learning, students can interact with teacher in a very natural and spontaneous way. However, video conferencing requires the use of highly advanced technology and broadband telecommunication networks. Up to now, neither WebCT nor Blackboard includes video conferencing feature.

4) Collaborative Work Groups.

According to the social-cultural theorists, social interaction and collaboration are the most effective ways of teaching and learning. In classroom settings, teachers often have students gather into smaller working groups within the classroom space. In Web education settings, it can be more difficult for students to meet in groups of three or four than to attend a traditional class. For this reason, courseware tools have features that support the collaborative work of small groups.

3.2.2 Limitations of current commercial Web-based educational systems

1) Time-consuming

Experience has shown that even after materials are created, the process of converting them into user friendly presentation forms that work effectively in the Web education environment is very time consuming. Apart from learning the specific architecture of the system, teachers often need to receive special training on how to effectively use different the components. The management of the course is even more demanding and time-consuming. Teachers must stimulate discussion and the posting of assignments on time, facilitate discussions, assign students to groups, assign roles and monitor group interaction, arrange for visiting experts, organize support staff, assign and record grades.

2) Non-reusable

Course content on the Web-based educational system currently comes in various forms, lectures, handouts, key points, tutorials, examples, quizzes and assignments. Content also comes in different formats, such as text, graphics, animations, audio or video clip. Teachers must create and edit materials in all of these disparate formats.

3) Homogeneous user

Most courseware tools are designed with a particular class of users in mind (as it usually done for on-campus courses) [Brusilovsky 1998a]. They may not suit other users because of "inflexibility". On-campus students are reasonably homogeneous, well-prepared and well-motivated. They have the access to on-campus teachers and assistants to fill possible gaps and resolve misunderstandings.

On the other hand, adult students learning part-time in distance education schemes are likely to be in a very different environment, and under very different circumstances (e.g. the mother of small children vs. a soldier on overseas assignment).

3.3 Specific concept-based educational systems

The specific concept-based educational systems are those experimental educational systems developed by research groups. They are designed to outperform the commercial educational systems in providing new paradigms such as intelligence, adaptivity and new metaphors. But, the trade-offs also come with these systems. For example, they normally have very complicated architectures that made them costly to build and difficult to implement.

3.3.1 WINDS

The Web-based Intelligent Design and Tutoring System (WINDS) [WINDS, 2002] is a European project with the objective to implement a learning environment integrating an intelligent tutoring system, a courseware management system, and a set of cooperative tools. The environment is designed to build a large knowledge base supporting architecture and civil engineering design courses and to experiment a comprehensive Virtual University of Architecture and Engineering Design. The basic concepts of the system, the design of its authoring environment as well as the strategy to implement adaptation are described in the following sections.

3.3.1.1 Learning Objects

To overcome the limitations in commercial courseware package, it is essential to store and collect the authoring material in a way that the content can be reused and accessed globally. Learning objects are introduced to the computer-based and Web-based education environment as the instructional elements, which are inspired by the object-oriented paradigm of computer science. Object-orientation technology highly focuses on the creation of "objects" which can be reused in multiple contexts. Here is the fundamental idea behind learning objects: teachers can build small instructional elements that can be reused many times in various learning and teaching contexts. Moreover, learning objects are delivering services over the Internet, meaning that any number of students can access and use them simultaneously. There are significant differences between learning objects and the traditional instructional media such as CDROM and tapes, as they can be dynamically integrated and remotely managed over the Internet.

The benefits of using learning objects in WINDS are summarised here:

1) Providing the basis for a consistent content generation.
2) Enabling for maximum of reusability of the course material.
3) Allowing for a maximum of flexibility for the dynamic generation of the course.
4) Facilitating the course content management.

Showcase learning objects (for learning astronomy) can be found at
http://www.astro.uiuc.edu/projects/data/

3.3.1.2 Authoring Environment

The visible component in WINDS authoring environment is the interface with which the teacher authors the course. The interface contains two parts: a searching frame on the left and an editing frame on the right. The searching frame contains a navigational tree which provides an overview of all courses authored and a pool of Learning Elements where the teacher can search for related topics and share learning units created by other teachers. In the editing frame, teachers are able to add, move, copy and delete

learning objects. Also, teachers can also connect or disconnect the links between learning objects.

3.3.1.3 Adaptive Learning Environment in WINDS

Adaptation is especially important for Web-based educational system for at least two general reasons [Brusilovsky, 1998a]. First, most Web-based applications are to be used by a much wider variety of users than any standalone application. "A Web application which is designed with a particular class of users in mind may not suit other users". As the users of Web course authoring tools, teachers vary sharply in teaching strategies, teaching experience and knowledge level about Web technology. Second, in many cases the student is "alone" working with a Web application. The human assistance that a colleague or an expert typically provides in a normal situation is unavailable.

As discussed above, adaptive nature is one of the key requirements for modern Web educational systems. In WINDS, a student model is created to represent the student's knowledge of domain, learning experience and preferences. Then the student model enables the adaptive navigation and adaptive content presentation.

3.3.2 Interbook and ELM-ART

InterBook [InterBook, 1999] is another specific concept-based tool for authoring and delivering adaptive courses on the Internet. It applies some research results in the area of Adaptive Hypertext and Hypermedia. These research results indicate that adaptive navigation support can make exploratory learning more productive and protect students from "being lost" in hyperspace. It is designed to provide students with adequate guidance, and the providers of learning materials with standards and authoring tools for an efficient utilization of the Web as an intelligent learning support media.

InterBook is a result of cooperation between researchers of the School of Computer Science at the Carnegie Mellon University and the Department of Psychology at the University of Trier. Conceptually, it is based on a knowledge-based approach to

creating adaptive and interactive electronic textbooks [Brusilovsky, et al. 1996]. This approach was originally applied in ELM-ART, a Web-based Intelligent Tutoring System for learning the programming language LISP.

3.3.2.1 Adaptive Learning Environment in InterBook

Like most Web-based applications, InterBook relies on software programs on a Web server for adaptive delivery of courses over the Internet. For each registered student, the program maintains an individual student model of student's knowledge and applies this model to provide adaptive guidance, adaptive navigation support (implemented as coloured bullets and font variations) to the student, and adaptive help. The "link annotation" technology provided by InterBook serves to both adaptively individualize, and to multimedia-enrich the student experience.

3.3.3 KBS Hyperbook System

KBS Hyperbook system [Henze & Nejdl, 1999] is a framework for designing and maintaining open, adaptive hypermedia systems on the Internet. It gives students the ability to define their own learning goals, propose next reasonable learning steps to take, support project-based learning, gives alternative views, and can be extended by documents written by the students. KBS Hyperbook contains Web server, Hyperbook servlet, Visualization module, Navigation module and Storage module.

In KBS, domain concepts are related to each other on the base of a conceptual model of the Hyperbook. Each Hyperbook unit is indexed with some knowledge concepts. For each domain, a separate knowledge model is constructed to contain the knowledge concepts of the domain and their learning dependencies (such as the prerequisite knowledge). A glossary containing the concepts used in the knowledge model is generated. For each glossary item, links to examples in Hyperbook units, and to pages from other courses available on the Web are generated. A page-sequencing algorithm

taking into account stated student goals and the level of knowledge established in on-line tests generates the learning sequence.

3.4 The Future of Web-based Educational system

Most Web-based Educational systems have layered or modular architectures, such as the presentation layer, student model layer and adaptation layer. One of the suggested improvements to the system is to add another layer to the existing architecture, which is called the "teacher model layer". This "teacher model layer" consists of teacher model with pedagogical rules or considerations.

Metaphors play a very important role in course authoring systems. According to the definition in Cambridge Dictionary, metaphor is an expression that describes a person or object in a literary way by referring to something that is considered to possess similar characteristics to the person or object being described [Cambridge Dictionary, 2002]. It helps both course designers and students understand the system easily, if the system uses metaphors, presented as realities from a familiar universe of discourse.

The common metaphors in course authoring systems are card, page, frame, object etc. It is said that new metaphors are needed to match the new practice in Web-based education [Hedberg & Harper 1998].

3.4.1 Introducing teacher model layer to authoring system

The purpose of introducing a teacher model layer to an authoring system is to enable the design of more intelligent and flexible teaching and learning environments. As discussed in Chapter 2, various education theories have great impacts on modern education practice. The Web-based education environment involves interactions between human participants, such as students, teachers and course administrators, and interactions between human and course content such as HTML, hyperlinks and multimedia. The intersection between teacher's pedagogical considerations and the attributes of Web-educational impact will draw increasing attentions from Web-based

education practitioners. One way to introduce the teacher model layer to the authoring system is to build teacher models which model his/her pedagogical expertise.

3.4.2 Intelligent Tutoring and Adaptive Learning Environment

Numerous attempts have been made to embody Intelligent Tutoring Systems (ITSs) to the Web-based teaching/learning systems. In order to achieve more powerful, adaptive and flexible authoring paradigms, these systems are typically developed with a four-module paradigm: student model module, adaptation module, application and client application module [De Bra & Calvi, 1998]. The key module is the student model. It is what makes an ITS an adaptive system since the student model is used in order to individualize the output to accommodate the needs of the student being tutored. The system can monitor the student's progress through a particular knowledge base and figure out which level the student is in and provide the adaptive feedback on how he or she should proceed. Currently, two categories of adaptations have been implemented: adaptive presentation and adaptive navigation [refer to Chapter1].

3.4.3 New metaphors for course authoring system

The authoring metaphor used to create a multimedia courseware is the methodology by which the authoring tool relates its components and processes to the objects in daily activities. The similarities between their characteristics will help users understand the complicated technical components and processes to accomplish the task. Authoring metaphors available to create courses include card/page-based, time-based, icon/flow-based, frame-based, and object-based [Kozel, K. 1997].

With the introduction of interactive multimedia learning materials and the movement towards constructivism, the organizing metaphor of the authoring system has become crucial to the effective design of the final learning environment. New metaphors for authoring systems need to be developed to match new practice in Web-based education

[Hedberg & Harper, 1998] and that of all the metaphors mentioned above are likely to survive. The objects-based metaphor stands the greatest chance because they reflect an evolutionary improvement. The object-based authoring metaphor is visually represented by the embedded objects and iconic properties. These media objects are mini multimedia applications which can be tied together in various ways.

Learning Objects Architecture is a framework that is developed to facilitate the object-based metaphor in online teaching and learning. IEEE learning Technology Standards Committee defines learning as: "any entity, digital or non-digital, which can be used, reused or referenced during technology supported learning." [LTSC, 1998]. Other standards are: ADL (Advanced Distributed Learning) Sharable Courseware Object Reference Model [ADL, 2002] and Dublin Core Metadata Element Set. These models play important roles in many Web-based educational systems, and deserve further discussions in the coming chapters.

3.4.4 Supporting teachers in course authoring systems

To date, most research efforts in adaptive and adaptable systems have focused on the students' tutoring and learning processes, i.e. the recipients of the content. As the contributors of the content, teachers have their own needs when they make the course online. Their requirements, which are different from students', haven't received much attention yet. Currently, most successful courseware authoring products, such as WebCT and BlackBoard, display some degree of adaptability nature (adaptability calls for system flexibility that allows user to perform modification) [Costabile, M. F. Piccinno, A., Fogli, D. Mussio, P. 2003] and customization abilities when students access the course sites. But none of them provides intelligent facilities and adaptivity (adaptivity calls for a system capable of monitoring user's behaviour and other contextual attributes, and use different approaches to automatically adapt itself) to help teachers create the online education site [Bayne & Cook, 2002] and help teachers select the right teaching strategy in Web environment.

Chapter4 Modelling techniques for Web-based educational systems

4.1 Introduction

Elaine Rich said, "Most systems that interact with human users contain, even if only implicitly, some sort of model of the creatures they will be interacting with." [Rich, 1983]. User modelling has been introduced to enhance the effectiveness of usability of software systems in a wide variety of situations. In these software systems, user models can be seen as the representation of properties of a single user or a group of users. To meet user's individual requirements, the user model acquires its information in an explicit and implicit manner [Virvou, 2001]. As a result, a complete user model should consist of both explicit aspects and implicit aspects about the user. Up to now, two categories of techniques for user modelling have been developed and evaluated by researchers. The easier technique is to construct the model manually, such as asking the user straightforward questions regarding the personal background. The user model attributes obtained in this way are normally explicit and static. Another way is to allow the system to learn the users' record, users' data and to build a model automatically using techniques like data mining. It is done based on the interaction between user and systems without any notice of user. The user model attributes and relationships obtained in this way are normally implicit and dynamic.

As we know, there are generally two main groups of users in Web-based educational systems, students and teachers. Students use the system to learn the knowledge while the teachers use the system to teach. Correspondingly, there are two categories of the user model: student model and teacher model [Virvou, 2001]. The objective of this chapter is to discuss the necessity of teacher model in Web-based course authoring systems. We will also discuss the modelling techniques and components in teacher models and how teacher models are incorporated into Web-based course authoring tools.

4.2 General Architecture of Adaptive Tutoring System

The Adaptive Tutoring System is an example of the computer mediated learning environment. Examining the Adaptive Tutoring System will help us understand the general architecture of the adaptive system. "For the system to be adaptive, it must improve its interaction with the user" [Langley, 1997]. The adaptive tutoring systems are generally made up of a few components or modules: the student model module, the adaptation module, the application and the client application module.

The student model contains all information that the system knows about the students such as their academic level, knowledge, goal, preferences, and experience of using the system. They are generally initialized either with default values or by querying the students in the form of questionnaires. Thereafter, these student's details are maintained by the system, although the students may be able to review and edit the explicit and static part of the student model. The dynamic part of the student model which maps student's actions and performances at various levels, such as task completion and requests for help, are also recorded by the system to enrich the student models.

An adaptation module (sometimes called adaptive engine) is responsible for dynamically changing the content presentation and navigation based on the student data. It takes in the student information from the student model, and derives adaptive outputs that suit the individual student needs. The adaptation engine particularly contains pre-defined algorithms which control all the changes to the outputs. These pre-defined algorithms could vary from simple statistical equation to more complicated inference such as Bayesian Networks. As the adaptation engine constantly communicates with the student model and the course content repository to provide the personalized content presentation and navigation, these three components, student model, adaptation engine and content repository, can be considered as the "Adaptive Content Server" which serves the content.

The client application focuses on rendering the presentation of the content which is the output of the "Adaptive Content Server". Another function of the client application is to monitor the student's performance, and then pass it onto the "Adaptive Content Server". This data will be analyzed and stored in the student model for use.

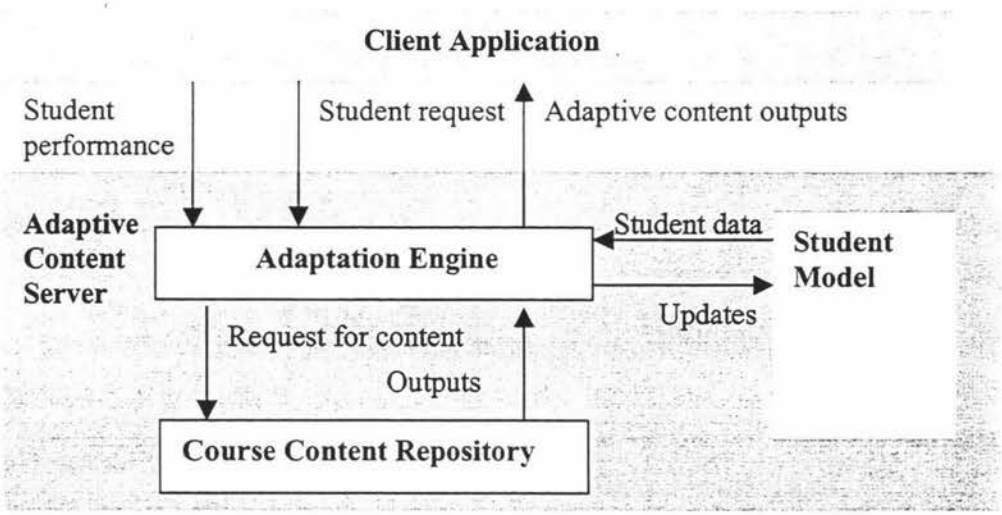

Figure 2. Architecture of the Adaptive Tutoring System

4.3 User Models

There are generally two main groups of users in the Web-based educational systems, students and teachers. Students use the system to learn the knowledge while the teachers use the system to create the online courses and teach students. Correspondingly, there are two categories of user model: student model and teacher model [Virvou, 2001].

The Web-based educational systems exhibit two different images to the users: course authoring images and tutoring images. Teachers and instructors view the systems as the course authoring systems when they publish course content, and conduct teaching over

the systems. The students and learners view the systems as the tutoring systems which focus on assisting student's learning in a Web-based environment. However, the role of the teacher's model in the tutoring systems has also been noticed [Kinshuk & Patel, 1996]. The teacher models in these images share some similarities. For example, they try to embed teacher's teaching strategies in their systems. In the course authoring system, the teaching strategy of a teacher is identified by monitoring teacher's behaviour when he/she creates the sites.

In a tutoring system, the question is how the teacher model helps student learn. It is said that the teacher model will greatly enhance the effectiveness of the intelligent tutoring systems (ITS) [Virvou, 2001]. This could happen if the student model mediates the teacher model or teacher model mediates the student model. Then the result of interaction could be passed onto the system to make adequate changes on the fly.

4.4 Student Model

The role of student model in Web-based educational system, more specifically the Web-based tutoring system, has been widely recognized [Stauffer, 1996]. And the frameworks containing the student model are getting close to commercial tools for the intelligent Web-based tutoring systems which provide adaptable content and presentation to students [Brusilovsky, 2001]. On the other hand, the teacher/instructor/author modelling in Web-based course authoring tools is just receiving attention.

The student model is the representation of properties of a single student or a group of students. It also reflects the current state of knowledge of the student or the group of students. Professionals in ITS have identified a number of key aspects of the student model [Brusilovsky, et al. 1996] [Sison, 1998] [Vassileva, 1996].

4.4.1 Student's knowledge

Students use the Web-based educational system to learn the knowledge of the subject. The effectiveness of this learning process can be evaluated by checking the current state of the student' knowledge on the subject. Therefore, the student's knowledge is the most important feature of the student model for all Intelligent Tutoring Systems. Student's knowledge is the variable ingredient in the student model. Firstly, each student is on the different level of the academic level with regard to a specific subject because of his/her study strategy, academic ability, the time spent on study or for other reason. Secondly, when the student uses the system to study the subject, his state of knowledge is constantly changing or improving. In this case, the knowledge representation in the student model must also change accordingly. "Overlay" and "stereotype" are the two modelling techniques which are applied to the student's knowledge, and they will be discussed in the following section of this chapter.

4.4.2 Student's preferences

The student's preferences are the component which represents personal likeness, such as favourable colours, text fonts, or particular types of questions (multiple choice or true-false). In terms of the modelling technique, the student's preferences can be modelled more easily than with other aspects of the student model as they can be explicitly quantified in a numeric way. For instance, decimal number 0-15 or binary 0000 – 1111 represents 16 different colours. The preferences are normally obtained directly from students by asking them questions. This data resides in the student model and does not change frequently. Another important characteristic of the student's preferences is that a group of single student's preferences can be aggregated into a student group model [Kaplan, 1993]. This feature is particularly useful in a collaborative learning environment.

4.4.3 Student's goal

The student's goals are the motivation to act in using the system. They can be further divided into sub-goals, which are closely related to the context of a student's work. Because of these divisions, the representation of student's goals will be of a tree or hierarchical structure [Vassileva, 1996]. Student's goals differentiate from each other in terms of their natures. Some common goals are: problem-solving goals, collaboration goals, concept-searching goals, test and verification goals. It is reasonable to distinguish between local low-level goals which change very frequently and general high-level goals which are relatively stable [Brusilovsky, et al. 1996]. Unlike to student's preferences, student's goals are changeable in general. They are sometimes difficult to be described by students themselves and hard to be identified by systems. As long as the student's current goal is recognized, overlay model in the system could be used to match the possible outputs to the goals.

4.5 Teacher Model

In order to make effective courses and to teach effectively from a teacher's perspective, it is imperative to incorporate a teacher's model into the Web course authoring tools and tutoring systems [Kinshuk, Patel, Oppermann, Russell, 2001]. The teacher modelling component in the system monitors each teacher's interactions with the system and constructs and/or updates his/her user model [Virvou, 2001]. The information stored in a typical teacher model may contain: teacher's knowledge about the subject and the system, teacher's task, teaching strategy and special preference. The teacher information will be used to help the teacher reuse the course artefacts and to set the appropriate teaching strategy and environment.

4.6 Benefits and trade-offs of teacher model

There are a number of advantages to have teacher models in Web-based course authoring systems. Firstly, the teacher model will benefit the teacher by improving the

quality of teaching. The "intelligent help" facility that comes with the teacher model will dramatically reduce teacher's time in planning the course as it can recommend the right teaching strategy and predict what the teacher intends to do. Secondly, the teacher model serves as an agent to help the student learn the subject. The content and navigation presented to the students are actually manipulated by both the student model and the teacher model. Moreover, it also provides a collaborative environment in which teachers can share data in their models. It is especially useful for the novice teacher. The quality of Web-based course authoring tools can also be significantly improved by incorporating the teacher model into the system.

However, incorporating the teacher models into the system requires significant human and financial investment. The great amount of communication between each component in model-based system will definitely slow down the response from the system to the user's request. The systems containing the teacher model are normally very complicated in architecture (like Bayesian Networks), and consumes a lot more hardware resources than the traditional Web-based educational systems.

4.6.1 Enhancement of the authoring process

It is widely accepted that course authoring is an iterative process that includes creation, evaluation and modification of the course content. It means that teachers might alter the course content and presentation after they evaluate the authoring outcomes. The teacher model in this respect will provide the historical records of teacher behaviours and actions when they interact with the system to modify the course. Teachers can make a reference of their previous actions, which are recorded in the teacher models. When teachers are well informed about the previous modification of their course and experience using the authoring tools, they will make better decisions.

4.6.2 Intelligent help

Adaptive nature is one of the main characteristics of the latest Web-based educational systems. The parameters or attributes contained in the teacher model can be used to provide the customized help. For example, these helps could be crucial when teachers encounter difficulties in using the tools, or what teachers should do when they generate the similar course. The more sophisticated teacher model could make predictions about what the teacher is going to do, and offer the relevant help.

The Web-based course authoring systems could also check the alignment of the setting of teaching goals with the choice of teaching strategies. For example, at some stage, the teacher's goal is to give entry-level students knowledge about a certain domain, it is suggested to adopt the Instructivist strategy. But the teacher offers students a complicated simulation practice which is suggested by constructivism. In this case, the authoring tools should inform the teacher about the incompatibility.

Alternatively, the student will benefit from teacher model as the teacher model represents the teacher in the system. This will be discussed in the following section.

4.6.3 Enhancement of collaboration among teachers

Learning to use new tools of technology (Web-based Course Authoring System) for curricular development requires collaborations on teaching and technical issues. The collaboration among teachers could be enhanced if the Course Authoring Systems come with teacher models as the key components. Teachers who have been successful in Web teaching have found that building a support community is essential [Rennebohm, & Pullman, 2001]. No single teacher can manage to be an active participant in this environment alone. The data kept in an experienced teacher's model will be extremely useful for a novice teacher who just starts making these Web-based courses.

With the help of a group of teacher models, teachers can execute tasks that are too complex for one individual teacher to undertake [Hamada & Scott, 2000]. It provides opportunities for teachers to participate in cross-discipline, cross-cultural group collaborations, to articulate their teaching strategies and teaching goals, and to manage

their workflow amid a high degree of uncertainty about how the effective web teaching be done. At the end, they must create an intellectual product collaboratively.

4.6.4 Enhancement of Pedagogical Innovation

When data kept in a group of teacher models is accumulated and aggregated, various data mining techniques can be applied to find underlying rules in the Web teaching practice, such as the relationship between the outcomes and different teaching strategies and the different student academic levels. These results provide the convincible evidence for education administrators to promote good teaching practice in certain setting, such as natural and social science, entry-level or senior level students.

4.6.5 Enhancement of teacher-student interaction

A typical adaptive system contains 4 main components: the domain knowledge model, the student model, the adaptation model and the client application. When the teacher model is incorporated into the adaptive educational system, the teacher model can exchange data with the student model. As discussed in the last section, the student model contains the student's progress, this data could be used to compare with the data in the teacher model and verify the effectiveness of this teaching practice [Virvou, 2001].

4.7 Modelling techniques

A variety of user modelling and analysis techniques have been developed to support specific applications. Two groups of techniques for the common user modelling have been developed and evaluated by researchers. The easier way is to construct the model manually, such as in asking the user questions regarding their personal background. Another way is to allow the system to learn the user and to build the model automatically. It is done based on the interaction between user and systems without any notice of the user. The ultimate goal to model users is to understand the underlying relationship and associations among the user's attributes (including interactions and

behaviours) and his/her expectations. Then the system will then predict the user's action based on the user's information needs or goals.

The following bullet list provides a sample of model attributes [User Modelling, 1997].

Typical attributes maintained in the computer user model are:

- User preferences, interests, and personal attitudes.
- Proficiencies (e.g. task domain knowledge, proficiency with system)
- Interaction history (e.g., interface features used, tasks performed/in progress, goals attempted/achieved, number of requests for help)

These attributes are in line with the previous discussion on the student's models. As this research does not focus on the detailed attributes in user models, the task will be left to the future research.

4.7.1 Classification and clustering approach

User classification and clustering are widely used approaches to model computer application users. In most occasions, they are called: overlay and stereotype. These two techniques have been mentioned in previous section and are discussed in detail here.

By overlay modelling [Goldstein, 1982], the user's state of knowledge is described as a subset of the expert's knowledge of the domain. And it is determined by comparing the User's knowledge to the expert's knowledge. It is critical to estimate user's initial knowledge by observing user's behaviour. In addition, a user's misconceptions of some knowledge concepts must be detected and modelled.

Stereotypes are also used to classify users [Garlatti, et al. 1999]. By categorizing or clustering users, the designer can treat them as a single unit, simplifying the design as well as processing load at run-time. A simple system may support only a single stereotype for each user. More sophisticated systems support multiple, possibly conflicting, stereotypes.

Here is a simple example of classification. If we use T_i to represents a set of Templates in our system, the attribute Y_j represents the set of teaching history (in years) of a teacher. we will have:

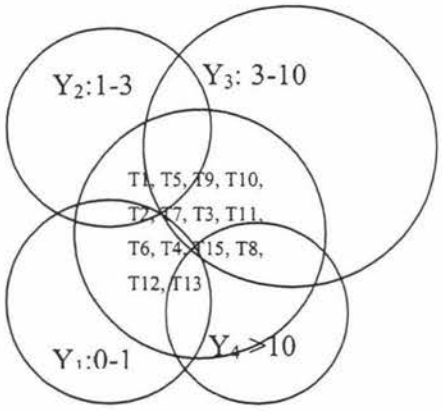

Figure 3. User modelling: Classification

When a teacher response to a question like "how many years of teaching experience do you have?" our system could tell which group of templates will be presented to the teacher. Obviously, the real user model will be much more complicated than this. The model includes many attributes as teaching strategy to come with a single result.

4.7.2 Static and dynamic user attributes

User attributes in the model are what we know about a user. They are the basis for classifying and categorizing the users. Apart from the discussion on the user's or teacher model, such as user's knowledge, preferences and goals, it is noticeable that all these user attributes can also be roughly put into two groups: static and dynamic attributes. The set of static user attributes contains, the age of user, nationality, language, teaching history and so on. The value of these static attributes can be acquired from explicit user modelling techniques such as questioning and surveying. The set of

dynamic user attributes contains values which are generated, while user interact with the system, such as the frequency of use, the time of the last access, the type of interface accessed so far. These attributes are coming from user's knowledge about the subject, his/her experience in using the system and goals.

4.7.3 Explicit and Implicit User Modelling

Explicit user modelling involves asking the user straightforward questions. For example, if a user log on to a Web site to apply for a free email account, they might be asked to fill in a form, which contains his/her nationality, gender, age, education history and special interest. The information will then be used to help establish a personalized email account, such as native language support. The implication of this kind of user modelling is the attempt to get initial information about a new user before the interaction and learning process starts. The explicit user modelling can be further individualized by implicit user modelling [Boyle & Encarnacion, 1994].

Implicit user modelling extracts information from the user's behaviour and interaction with the system. When a user operates on a system, the system can observe user actions without the user being aware of this. Most of the time the user's task is unknown in advance, or his/her goal is hard to describe. Implicit user modelling is particular useful when the system knows little about the user. It can give valuable information about the user's knowledge regarding to the system. However, it is not easy to find a way to model the implicit and dynamic attributes in the user model.

4.7.4 Probabilistic approach - Bayesian Networks

Here, let's say a few words about Bayesian Networks that are the most widely used probabilistic approach to user modelling [Pearl, 1988]. In the real world, there are many examples where the probability of one event is conditionally related to the probability of another event/s. A Bayesian Network can be represented as a directed acyclic graph [Murphy, 2001] in which the nodes represent variables, events or actions, for example,

the attributes in a user model. These variables consist of a set of mutually exclusive and exhaustive propositions (called hypotheses). For each of these hypotheses, the probability of being true is maintained. All of these probabilities together are called a probability distribution, also known as the belief. The dependencies between the propositions are represented by links between nodes, called branches and networks. When applied in a human-computer interaction, Bayesian Networks are used to estimate the probability of unknown user behaviour given existing knowledge about the user.

For example, if we still use the templates as the components to build our teacher model. "Select T1" stands for the selection of Template1. Then we will likely have a probability distribution networks P(T) in the teacher model.

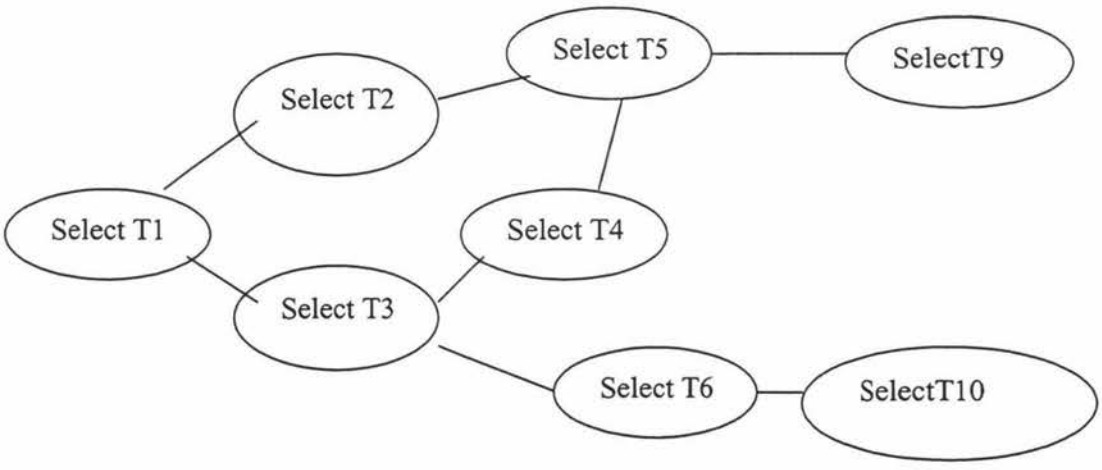

Figure 4. User modelling: Bayesian Networks

If we have got a set of data regarding to a teacher's selection of templates, we can train the teacher model with Bayesian Networks. P(T) represents the prior belief of selection

of templates set. P(D) represents the probability of set of data captured in the interaction between teacher and system. P(T|D) represents the probability of selection of a specific set of templates regarding to the foundation Bayes rule [Buntine, 1996]

$P(T|D) = P(T)\ P(D|T) / P(D)$

If we have two teacher models or two sub models M_1, and M_2, we will have:

$P(T=M_1|D) = P(T=M_1)\ P(D|\ T=M_1) / P(D)$

$P(T=M_2|D) = P(T=M_2)\ P(D|\ T=M_2) / P(D)$

But since P(D) is constant and appears in both equations, this can be simplified to

$P(T=M_1|D) = P(T=M_1)\ P(D|\ T=M_1)$

$P(T=M_2|D) = P(T=M_2)\ P(D|\ T=M_2)$

The resulting fraction (called Bayesian factor) is used to decide which model to choose: if it is over 1, model M_1 is selected, otherwise model M_2 is preferred.

Bayesian Networks is such a tool that encodes implicit, uncertain and probabilistic relationships among events, actions and variables of interest (such as the relationships between user's actions and goals). When used in conjunction with statistical techniques, the Bayesian Networks has a number of advantages for the normal data analysis. Firstly, because the model encodes dependencies among all variables, it can handle situations where some data entries are missing. Secondly, a Bayesian Networks can be used to learn implicit relationships, and hence can be used to gain understanding about a user's behaviour and his domain knowledge, goals, and motives. Therefore these relationships can be used to predict user's actions.

Uncertainty in the user-system interactions arises in a variety of situations such as:

- Uncertainty inherent in the domain being modelled.
- Uncertainty as to the accuracy of user attributes.

- Uncertainty about user's expectations
- Uncertainty as to the actual availability of user attributes in the model.

Despite the great power of Bayesian Networks in probability analysis, two problems are causing concern. The first one is the computational difficulty of exploring a previously unknown network. To calculate the probability distribution of any branch of the network, all branches must be calculated. It might take a lot of time and resources to perform, or may even be impossible to perform.

The second problem concerns the quality and extent of the prior probability distribution used in Bayesian inference processing. A Bayesian network is only as useful as this prior knowledge is reliable. Either an excessively optimistic or pessimistic expectation of the quality of these prior probability distributions will distort the entire network and generate invalid results.

4.8 Template and User model

Templates have been widely used in content authoring such as Word, PowerPoint and FrontPage. Here, templates are seen as the skeleton of documents, which users may use when they create new documents. Previously, it is called template-driven approach. But template has another use in the course authoring. Templates serve as the artefacts which can be manipulated by user models. It means that the elements in the template are controlled in terms of the relationships to the attributes of the user model. Not only does it accommodate the static and dynamic attributes in the user model, but it also offers a solution to the implicit user modelling.

4.8.1 Template attributes

Templates can be looked on as a set of skeleton document objects. Each set of templates and template fragments, which are the elementary components of the template, should contain some attributes which allow distinguishing the use of them in a way that is relevant to the suitability of different categories of users. The attributes of these

templates include the names of the templates, URL of the templates, Keywords in the templates, the language in which the templates are defined, popularity of a template, the topology (if we describe templates in a hierarchical way) or relationships of the template and template fragments. There are two different groups of template attributes in terms of their states in the user modelling process: static and dynamic attributes.

4.8.2 Static vs. Dynamic template attributes

Some template attributes show static features. For example, the names of the templates could indicate the subject domain (such as social_science_template1), or the language in which the templates are defined indicates the nationality of users. These static template attributes are corresponding to the relevant attributes in the user/teacher model which do not often change their value.

Dynamic template attributes refer to those unstable attributes, such as the popularity of the template (frequency of use by all users, or by some pre-defined groups of users) and ranking of template, topology or correlations of templates in a template set. Template set is a set of templates which are logically related to each other. If a user continuously uses a template, the popularity of the template increases. The ranking of the template set will change correspondingly. The template, which serves as the representation of user model, can address both static and dynamic features of user model.

4.8.3 Implicit representation of user model

The main objective to introduce the Template-driven teacher modelling approach is to implicitly represent the user/teacher model. When the teacher uses the templates to create courses, he/she should have motives or goals (semantic intention) in mind though sometimes these motives and goals are difficult to describe. Suppose that a teacher chooses a teaching strategy to conduct the online tutorials for the computer science students on the topic of database. Although the semantic intention is decided, it is still hard to predict his or her physical actions to build such a teaching module. He may click on a series of buttons in the Microsoft FrontPage or Macromedia DreamWeaver to

create a Web page. With templates, we hide the details of individual applications. We look at templates as the vehicle to transport the semantic meanings. If we present a set of templates to the teacher, and let him/her choose the template that he/she likes, we actually implicitly train the teacher model. With templates, the implicit information about the teacher can be explicitly recorded and quantified. Then we can use some algorithms, such as statistical techniques or Bayesian Networks, to set up the relationship between the teacher's goal and his behaviour. Consequently, the system with the teacher model and Bayesian Networks could predict teacher action according to his goals.

We can also move one step further to add teaching strategies to this teacher model. As we have discussed in Chapter 2, there are a number of different teaching strategies such as "Instructivism", "Constructivism" or "Social-Cultural theories". It is hard to find teachers who adopt only one teaching strategy. In most cases, they choose hybrid approaches. A teacher may select a more "objective' approach to one teaching module, then choose a more "constructive" approach to another teaching module. If we decide to build a number of teaching strategies for each teacher model rather than a single teaching strategy for a teacher model, it will be possible to learn more accurately about a teacher. Associated with each teaching strategy, there is an URL which we call a template index or directory page. In each template index page there is a list of templates which are put in order or rendered in different colours.

Teacher	Teaching Strategy	URL of Templates
A	Objective	http://ispost.massey.ac.nz/templates/Template.html
A	Constructive	http://ispost.massey.ac.nz/templates/Template1.html
A	Social-Cultural	http://ispost.massey.ac.nz/templates/Template2.html
B	Objective	http://ispost.massey.ac.nz/templates/Template3.html
B	Constructive	http://ispost.massey.ac.nz/templates/Template4.html
B	Social-Cultural	http://ispost.massey.ac.nz/templates/Template5.html

Table 2. Teacher model and Teaching strategy

Figure 5. Example of template index page

4.8.4 Content presentation and navigation with Template-driven teacher modelling

Template-driven approach in the course authoring process is a modelling technique to represent a teacher's model. It tries to relate both explicit and implicit attributes in a teacher model to templates in order to provide better presentation and navigation of documents in which the course content are published. Simply put, the objective is to use template and template fragments to represent the implicit nature of a teacher's expectations in the course authoring process. With the help of the teacher's expectations which is derived from the calculation of attributes on the teacher model, the system will automatically predict the teacher' action, serve the right template, and make the most "appropriate" templates the most accessible to the teacher instead of providing access to all available templates.

The most widely used technique to implement the user models is the prioritizing. Prioritizing the templates in terms of the attributes in teacher models involves two steps.

The first step is to determine the priority for templates. The second step is to determine the way to express the priority in the client application. Some examples are suggested.

1. Prioritizing according to the frequency of use (the more frequent, the higher priority).
2. Prioritizing according to the correlations of templates in a set of templates.
3. Prioritizing according to the time the last access (the more recent, the higher priority).
4. Explicit matching (language, organizational regulations).

Chapter5 Template Server Approach

5.1 Introduction

With Web-based Course Authoring system, teachers are able to select their favourite templates, create, and modify the course contents which are to be presented in some forms of documents. Traditionally, course designers use the WYSWYG tools, such as Microsoft Word and PowerPoint, to create documents, process text and present multimedia. This approach offers a high level of control over the appearance of a document but they do not offer reusability and portability among different applications [Tsai, 1998]. Some difficulties can occur when exchanging data or documents between applications. Nowadays, the global availability is crucial to Web-based educational systems. To present the same content in another document format, user has to generate new document without taking the advantage of the previous document. The solution is to separate the content (data-oriented) and presentation (application-oriented).

The Extensible Mark-up Language XML provides an ideal tool to separate the content and presentation. It offers an approach to the content reusability. XML is suitable for the exchange of data/content. It is well structured, self-describing, easily parsed. With semantic mark-up, very high typographic quality can be achieved, and users do not need to spend time on formatting the content for different applications. XML is also an ideal tool to describe a document template.

Here we propose an approach [Tretiakov & Shi, 2003], which is the key concept in the Template-driven Teacher Modelling Architecture. The Template Server Approach differs from the traditional WYSWIG. The Template Server Approach we suggest is not entirely new. It is inspired from some existing architectures and practices, such as Web publishing, 3-tier architecture, template in Microsoft PowerPoint. Actually, some researchers even suggested a new language called "Template Mark-up Language"(TML) for Web-based applications [Kristensen, 1999]. The Template Mark-up Language (TML) is an application of XML which defines a generic and flexible set of template mark-up elements and tags.

To clearly present the proposed Template Server for the Template-driven Teacher Modelling Architecture, the UML (Unified Modelling Language) is adopted in this thesis to describe the architecture. The UML has been widely used in recently years. It is the recommended notion for visually expressing the system models and system architecture in an objected-oriented fashion.

5.2 The Template Server Concept and the Prototype

5.2.1 The description of the concept

Templates are normally used for creating documents in a fast manner. But they can also be used in another way. In the Template-driven Teacher Modelling Architecture, an intelligent program will provide the user with a set of templates (in template index) which will potentially meet the user's requirements for entering content.

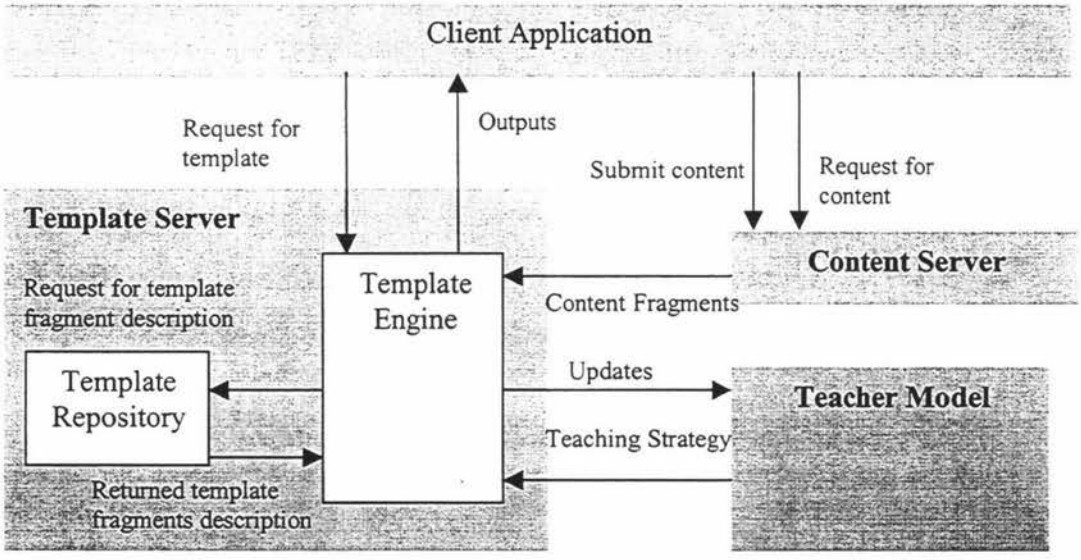

Figure 6. Architecture of Template Driven Teacher Modelling for course authoring

The Template Server in the Template-driven Teacher Modelling Architecture consists of two major components, Template Engine and Template Repository. The Template Engine takes in the user requirements and negotiates with the Template Repository and Teacher Model to dynamically generate or serve the templates. The template repository

consists of a collection of template and template fragments which are normally stored as documents written in a certain mark-up language, such as XML and XSLT.

We use the same term for an organization hosting HTML pages on Web Servers within its scope of control. A Web Server actually serving HTML pages in response to client's HTTP requests for appropriate URLs. The "Template Server" responds to the client's request in a similar way. It serves templates rather than the Web page (the document).

5.2.2 The difference between Template Server Approach and the commercial document authoring packages

One could argue that the most commercial office software packages like MS Office also provide templates. Then what are the differences between these two kinds of templates? As mentioned earlier, the templates in those packages are normally in the vender-specific document formats (such as .dot for Word and .pot for PowerPoint) instead of the universal accepted format XML. When the teacher modelling is carried out, the template repository serves as the data source for this process. But we know that the semi-structured documents are not suitable for data extraction. Although we can use the program to convert the documents to XML, it would be very expensive to develop such a software application that is able to convert all software document formats into XML. Also this kind of application requires a file called DTD (Document Type Definition) beforehand [Frank, 2003]. On the other hand, the XML-based template matches the standards for uniformed data storage and interchange.

5.2.3 The brief of the template server

In terms of its functions, the Template Server resembles the Controller in MVC architecture [SUN, 2002]. It translates user interactions with the client interface into actions to be performed by the model. Apart from the Template Server, the proposed prototype is also composed of other modules: Teacher Model, Client interface and Content Server. The teacher model is the module which contains data about the teacher,

his/her usage pattern of templates and algorithms to predict the teacher's action. This data is used as the inputs for rules and algorithm which govern the selection of templates and the creation of new templates and make predictions about which group of templates the user will want to see. The Client interface contains the view that presents the templates and document content. It maintains the consistency in its presentation when the teacher model changes the output. The "Content Server" is the program which provides content in response to the teachers' request when teachers need to update the course contents.

5.3 Template Server

Template Server is the most important module in Template-driven Teacher Modelling Architecture. It consists of "Template Engine" and "Template Repository". Following paragraphs give the full descriptions of these components.

5.3.1 Template Engine

Situated in the centre of "Template Server", the "Template Engine" actually is the program which serves templates in response to a teacher's request. To create a course on the Web, a teacher needs to put the content into the skeleton documents. Therefore the teacher should be able to create template, store it in the repository, and select it (the integration of template fragments) from the repository. The "Template Engine" then plays its role in transforming the template or template fragments descriptions into a template document which can be viewed in client application, such as HTML forms.

The second role of "Template Engine" is to adaptively serve the templates, such as hide some inappropriate template from the teacher or prioritize some favourable template. The "Template Engine" negotiates with "Teacher Model" to select the specific set of templates and template fragments descriptions. This will determine which set of templates and template fragments descriptions are to be selected and which relationship is presented among them in terms of the output of adaptation rules or Bayesian Networks. The templates are generally presented in a way to prioritize the most

favourable templates, such as putting them into different sequence or annotating them in different colours. To implement the adaptation, the "Template Engine" talks to the "Teacher Model" which stores the access history, usage pattern, preferences and other teacher characteristics and rules and algorithms to predict a teacher's behaviour. It mainly assists the teacher in choosing the appropriate templates.

5.3.2 Template Repository

"Template Repository" is the storage of all templates or template fragment descriptions. After receiving inputs from "Client application", the "Template Engine" communicates with "Template Repository" to generate the template document which is requested by the teacher. These templates, or template fragment descriptions stay in the "Template Repository" as individual files. It is necessary to find a way to organize these files when a real system is designed. There are three ways to manage the templates in "Template Repository". The first option is the traditional file system. All computer operating systems provide the basic file management facility. Files are stored in folders in hierarchical structure. The advantage of file system is the simplicity of implementation. It supports the content searching. But the file system does not support the advanced querying like Relational DBMS (database management system). The second option is the relational database approach. Each file can be stored in the database as a "record". Each record contains a number of "fields" which store a piece of information such as author, title, the group, date of creation, and physical address (URL) of the template. The database management system is more powerful than traditional file system. But it is relatively expensive to build and maintain. Another option is the object database. The object database is ideally suited to handling the XML data. In the object database, the XML data is stored as a DOM object which maintains the data, method, relationship and semantics of the model. Moreover, the World Wide Web Consortium has announced the query specification for XML, XML-QL [W3C, 1998], XQuery [W3C, 2002]. It means that users can retrieve XML data from an object just as what they are doing in relational database using SQL.

5.4 Content Server

The "Content Server" is the program which provides content in response to the teachers' request when teachers need to update the course contents. The "Content Server" keeps all the contents and content fragments in the "Content Repository", which is created by teachers in the template domain. Content Repository represents the knowledge which the teachers wish to teach. These contents could be created, updated and deleted in the system. Similar to the "Template Repository", "Content Repository" is made up of content and content fragments in the form of XML files.

5.5 Document Engine

The "Document Engine" is the intermediary component between "Template Server" and "Client application". It can be embedded in either "Template Server" or "Client application", such as Web browser. The main function of "Document Engine" is to render the template document and display it in the target document format. When rendering, "Document Engine" allows the user to render, download and present the template document in a variety of document formats, e.g. PDF, HTML or Microsoft Word. New document formats can be accommodated as they appear. It is the responsibility of the "Template Sever" to create the templates. And the responsibility of the "Document Engine" in our prototype is to offer a presentation service which renders templates in HTML.

5.6 Client application

Client application is the component in the architecture that facilitates the dialog between user and system, such as taking user's inputs, and presents the template document outputs to the user. The client application should have two basic functions. Firstly, it should display the documents with content or template document without content. If it is the template document without content, the client application should be able to accept the content inputs. The HTML form is an option. Secondly, the client application should display the adaptive template directory, offer previews or hints of templates for selection, and accept the user's selections.

For example, in a HTML form, the user enters the content data necessary to complete the form. Once the user chooses to submit his request, the content data in the HTML form will be encoded and sent to the Web server. The Web server passes the encoded data to the "Content Server". The "Content Server" decodes the data and stores them in the "Content Repository".

To present the adaptation result, the client application takes in the output from "Template Server" which negotiates with "Teacher model" to select the specific set of template or template fragments. The widely used approach is to prioritize the most favourable templates. Such a series of negotiations could be triggered explicitly by the user, or automatically by the client application immediately on start-up. The specific templates are then selected to present to the user. All the descriptive information about these templates, not only the names of the template but also structure and suggested hints for usage, will be extracted from the "Template Repository". The information could then be used to provide the graphical previews and textual descriptions. In regard to providing the previews of templates based on the textual data, SVG (Scalable Vector Graphics) is an ideal solution. In SVG, graphics can be described by plain text in the form of XML [SVG, 2003]. The textual messages in SVG files can also be searched for use.

5.7 Implementation of Teacher Model

The teacher model has been discussed extensively in the previous chapter. Here the teacher model has a more specific meaning. It stores the data about the teacher's preferences, such as the usage of each template, the algorithms to classify the teacher in terms of the data. Then the rules in the teacher model will be used to predict the teacher's preference to choose a certain set of templates. Our solution is to provide a template index page to the teacher. Each index page is associated to a single teaching strategy. Each teacher could have a number of teaching strategies. In this way, we fine-tune the teacher model.

5.8 A Brief introduction of UML

As stated at the beginning of this chapter, UML (Unified Modeling Language) is adopted to visually describe the Template-driven Teacher Modelling Architecture. Here we need to give a short description about UML. The story started from the mid-1990s, three of the former contestants in object-oriented community (Grady Booch, Jim Rumbaugh, and Ivar Jacobson) finally worked together at the Rational corp. and submitted the first version of Unified Method documentation to the Object Management Group (OMG). Since then, UML becomes the standard object-oriented modelling language.

5.8.1 Various views of a system

A model is a representation of a system or architecture which may contain interrelated elements. A good model allows different viewers to see various levels of details or various aspects of a system at different times or states. To reflect all these differences, UML groups all its modelling methods in a set of interlocking views. Each view reveals a particular set of aspects that are of concern to a particular viewer group. For example, the Use Case view focuses on the scenarios executed by human users (actors) and external systems. Another example is the Static view (Object Diagram). It captures the overall structure of the system in terms of classes. The Static view could describe the static natures of objects as the object unifies the data (attributes) and behavioural features (methods) into one single entity. But the Static view does not contain any representation of the dynamic behaviour. Dynamic Views include the Sequence Diagram and Collaboration Diagram.

5.8.2 Template-Driven Teacher Modelling Architecture in UML diagrams

In each view, UML defines a set of common notations which form the basic elements to build a set of different diagrams in each view. These notations and elements are created in a way that they are easily understood and reusable. These diagrams may emphasize

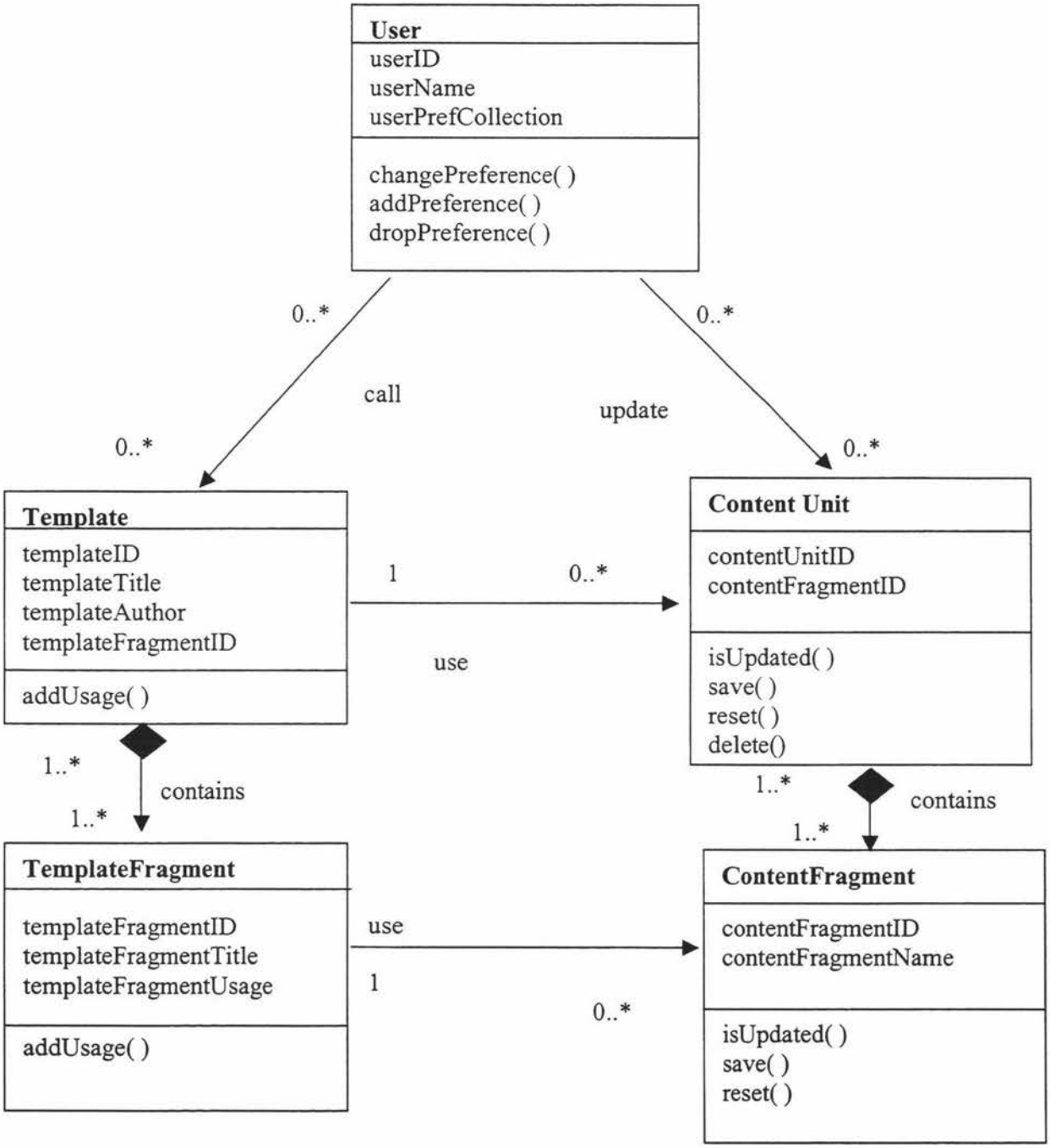

Figure 7. Class Diagram

on various aspects. For example, in the interaction or dynamic view, the Sequence Diagram and Collaboration Diagram both demonstrate the interactions of objects. The Sequence Diagram clearly shows the time sequence but do not show the relationship between objects. Statechart in UML is another means of describing the dynamic behaviour of elements. In the contrast, the collaboration diagram demonstrates the relationship very clearly.

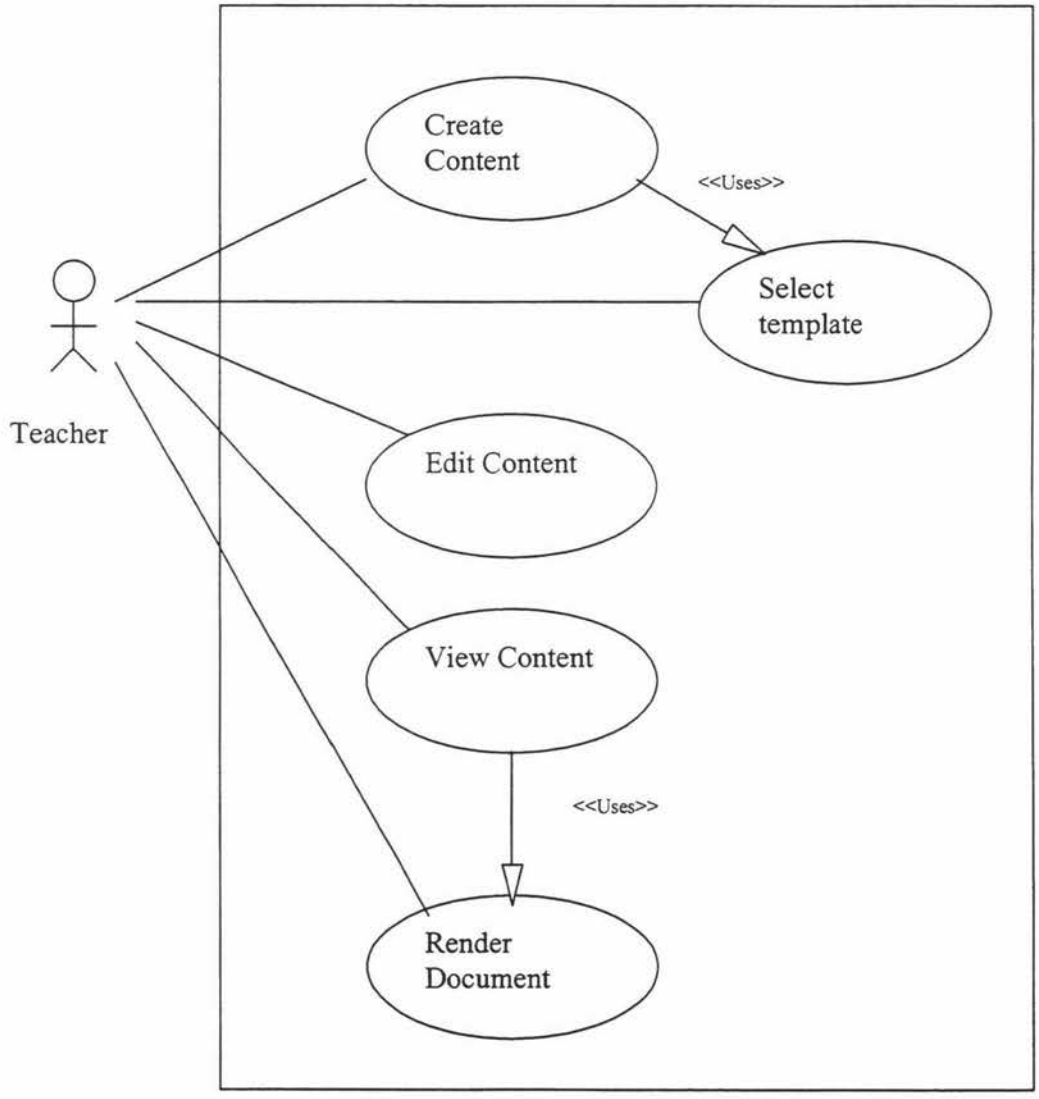

Figure 8. Use Case Diagram

5.9 Use cases

We use the Use Case Diagram to model the classes and the typical user's actions on the prototype system. On the client side, the client application includes Web browser. On the server side, we keep all the other components, Template Server, Document Engine, Content Repository and Teacher Model. The template and contents are stored as XML files in "Template Repository" and "Content Repository" respectively.

5.9.1 User login

The first thing that the user should do is to login the system. The system will identify the user and find all the information about this particular user. Then the user specifies a task to do, such as creating a new section or chapter. This will trigger the system to calculate the user's data in the Teacher Model. The calculation of this data helps the system locate the particular template or a group of templates. Corresponding to each teaching strategy, which serves for different pedagogical requirements, sometimes, a single user may have a number of different template sets in the template repository. The data the user provides will help the system find the specific template sets.

5.9.2 To select a template

In order to create the course content, users need to request template from the "Template Server". "Template Engine" in the "Template Server" adaptively the template index or directory based on the data (personal data and task data) which the user gives when he/she logs in. The "Template Engine" contacts the "Teacher Model" to classify user's teaching strategy, which is then used to select the appropriate set of templates which are listed on a template index page. The user chooses the template that he/ she needs from the template index or directory, and then commits to use it. The Client Application translates the selection into the address of the template, and "post" it to the "Template Engine". The "Template Engine" locates the XML and XSLT files in the Template Repository". The "Template Engine" then transforms them into a visible template in a

document format for inserting the course content. By default, an empty HTML form is displayed to the user.

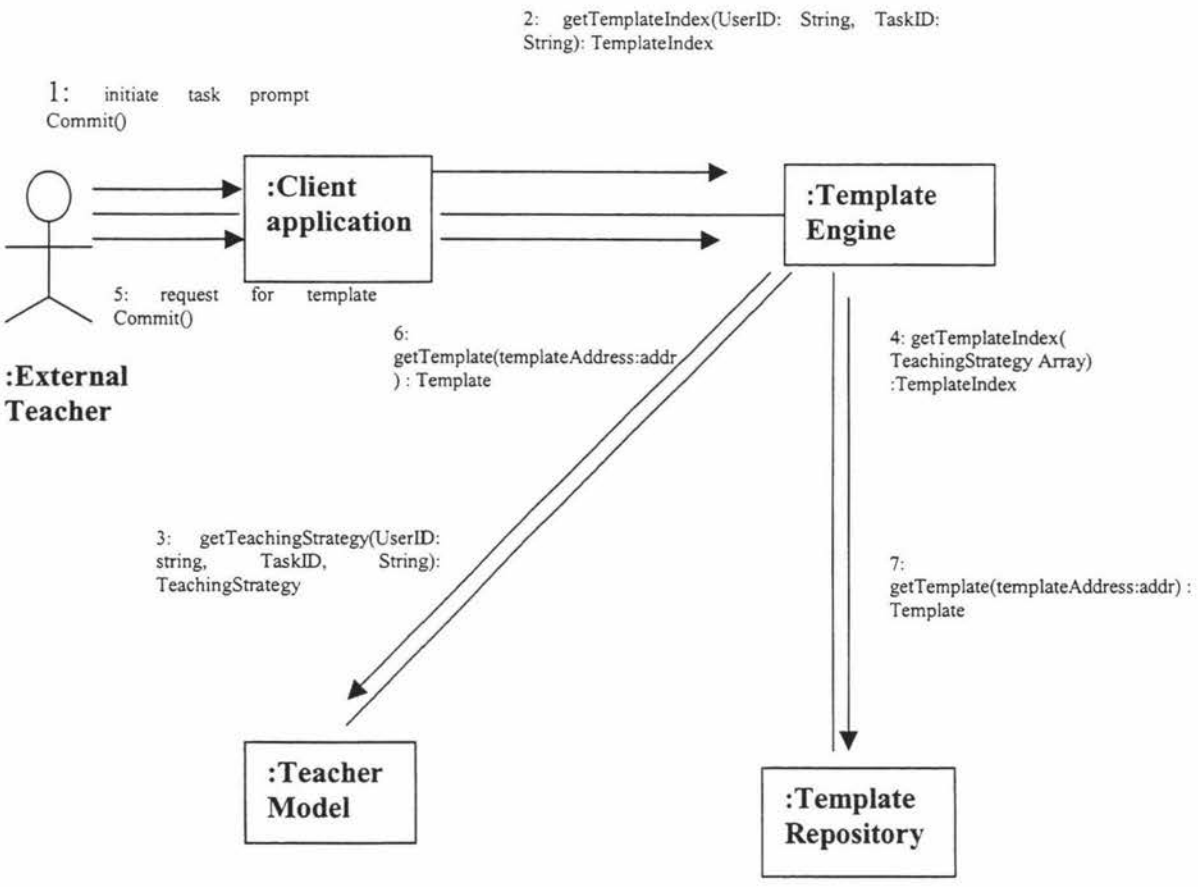

Figure 9. Collaboration diagram (Select Template)

5.9.3 To create content

Firstly, the user needs to select a template. This is the same as what happened in the section "Select Template". The difference is that system will not adaptively change the template index in terms of the teacher model. After the empty template is presented as a skeleton document to the user, the user will be able to enter the information required (e.g. fill in text fields in the form), and submit the populated template to the "Content

Server". The "Content Server" extracts the content data from the populated document and stores them as content files in the form of XML in the "Content Repository".

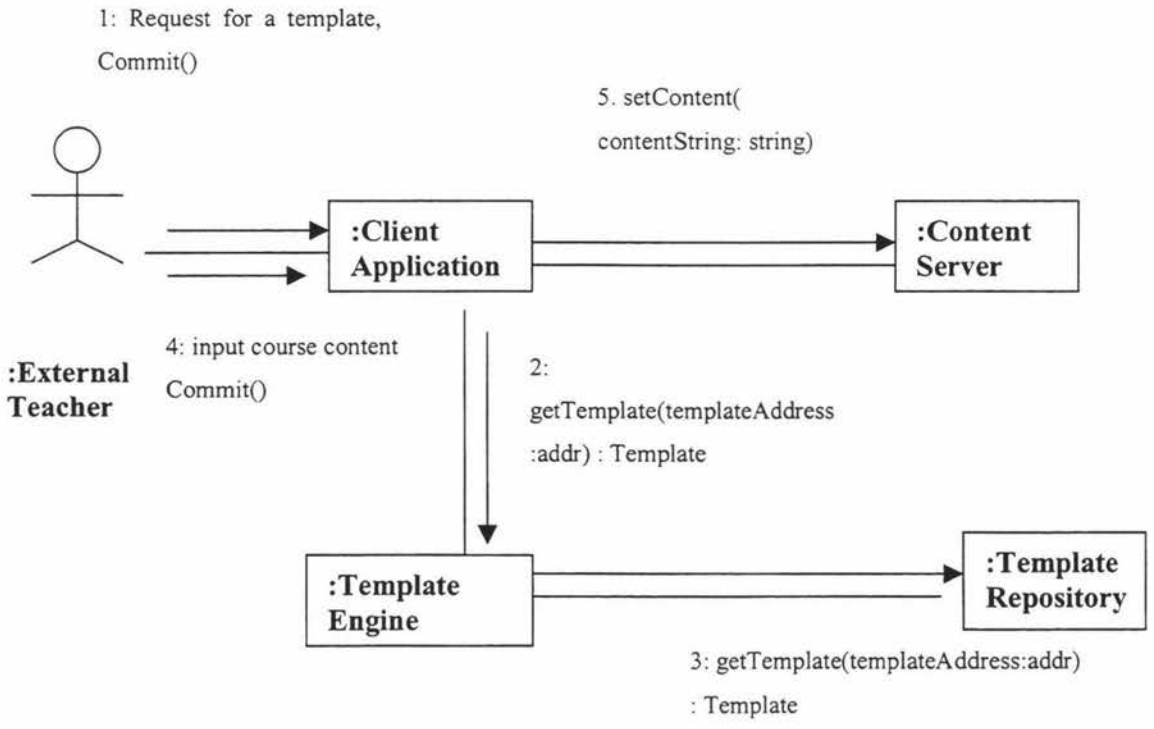

Figure 10. Collaboration diagram (Create Content Unit)

5.9.4 To view content

The user requests a summary of his/her content files. Then he/she selects a content file, and submits a request to view it. The information of the content file is automatically sent to the "Content Server", which returns to the user with the requested content. The content is displayed in the default template and document format, normally HTML form.

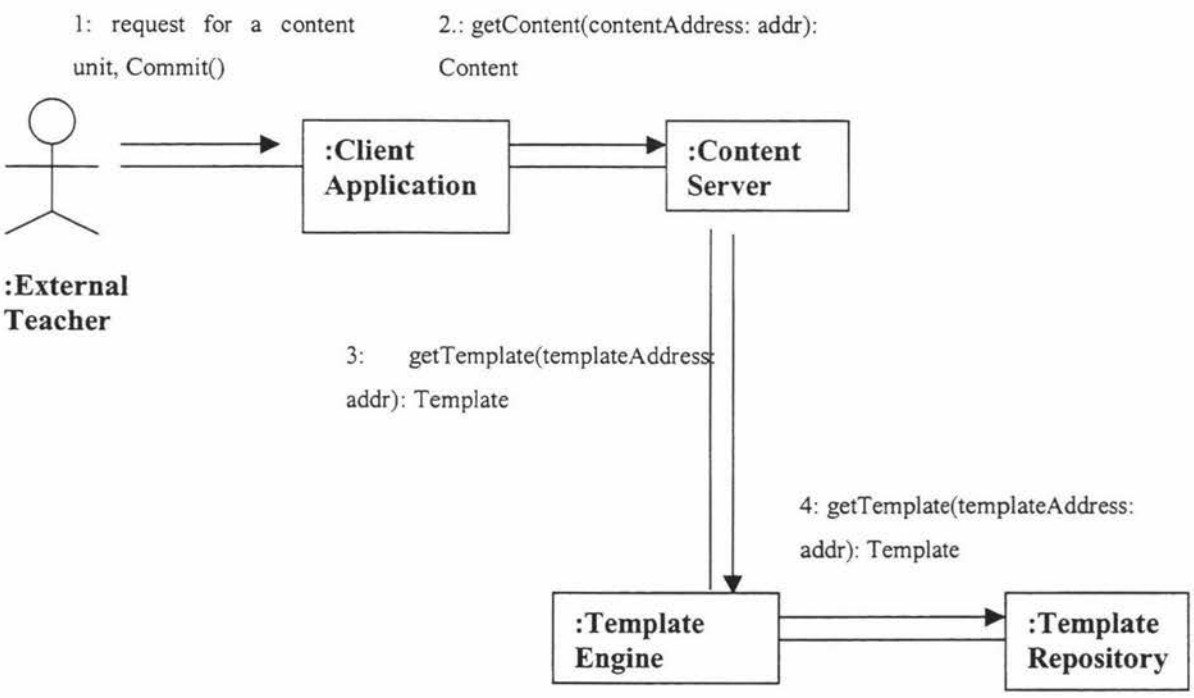

Figure 11. Collaboration diagram (To view content Unit)

5.9.5 To edit content

The user requests a summary of his/her content files. The user selects a content file, and submits a request to modify it. The information of the content file is automatically sent to the "Content Server", which returns to the user with the requested content. The content is displayed in the template document format when the content is created, normally in the HTML form. The user makes the changes that he/she intends to, and re-submits the content data to the "Content Server". The "Content Server" extracts the data, and stores them to the "Content repository", which is a component of "Content Server". The content can be stored in the "Content Repository" as a new version or the same version as the previous content file.

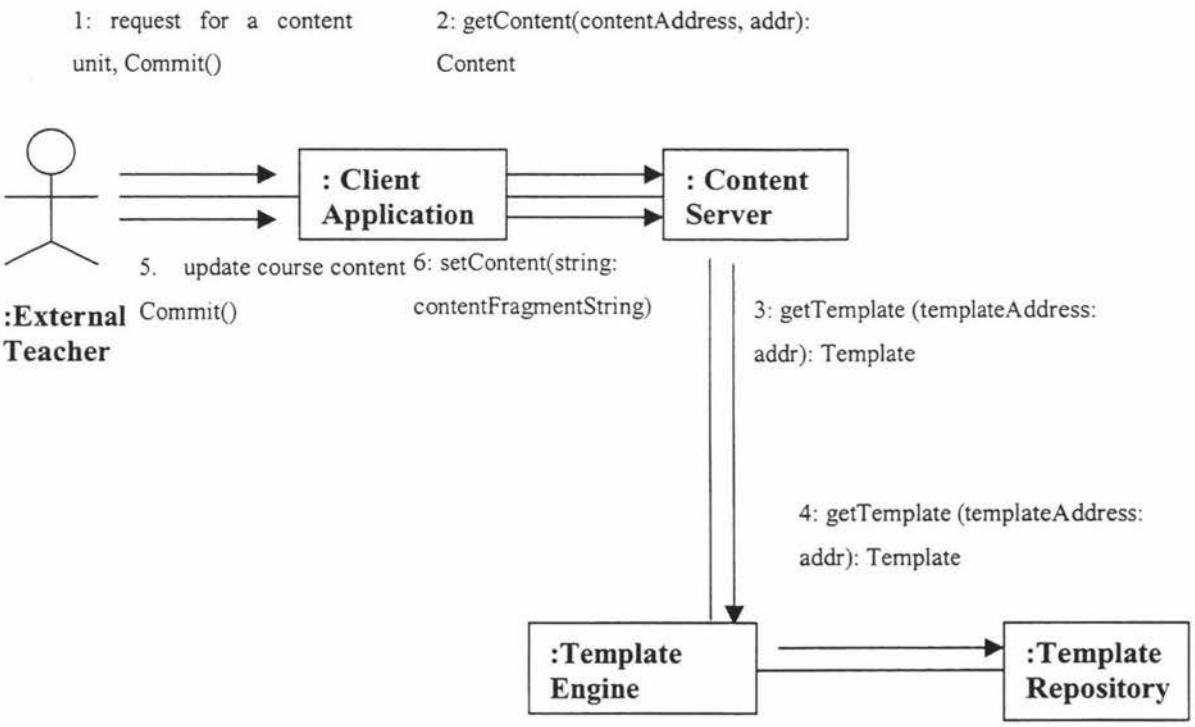

Figure 12. Collaboration Diagram (Edit Content Unit)

Chapter6 Technical implementation

6.1 Introduction

Based on the Template Server Approach thich we proposed, we developed two prototype systems, prototype A and prototype B. They are all Web-based, 3-Tier systems. Both of them contain the template server, client application, content server and teacher model which are hosted on a server with Apache Web Server and Tomcat Servlet container. The template server has been configured for adaptively serving templates and tracking user's actions. The implementation of these two prototypes includes following steps:

- Developing template engine which serves the templates
- Developing template repository
- Building client application
- Building teacher model
- Building content repository
- Designing user login and implementing user tracking
- Developing adaptation engine which contains algorithm and assists the template engine

6.2 3-Tier Architecture

Most of the sophisticated Web based applications, which involve data entry, data processing and data storage, are based on a 3-Tier architecture. The key characteristic of a 3-Tier system is the separation of a distributed application into presentation, functionality, and data components. It provides a flexible, scalable, extensible architecture with a manageable environment, and facilitates rapid development of robust applications. The 3 tiers are:

- Presentation tier (mainly Client application). This tier interfaces with the user, such as taking the input and presenting the output.
- Functionality tier. This tier provides functionality, transformation, connectivity

- and adaptability nature. In a Web environment, it consists of the Web Server/Application Server.
- Data tier. This tier contains data or models which represent the data requirements. This data can be stored in the Database or embedded in XML files.

In fact, all Web-based systems consist of data in the database (or XML) and algorithms in the application. At some stages, the data must be validated. The Data validation (such as XML transformation) and separation can happen in any or all of these 3 tiers depending on the needs of the application. The pros and cons to validating data in each tier will be described in the following sections.

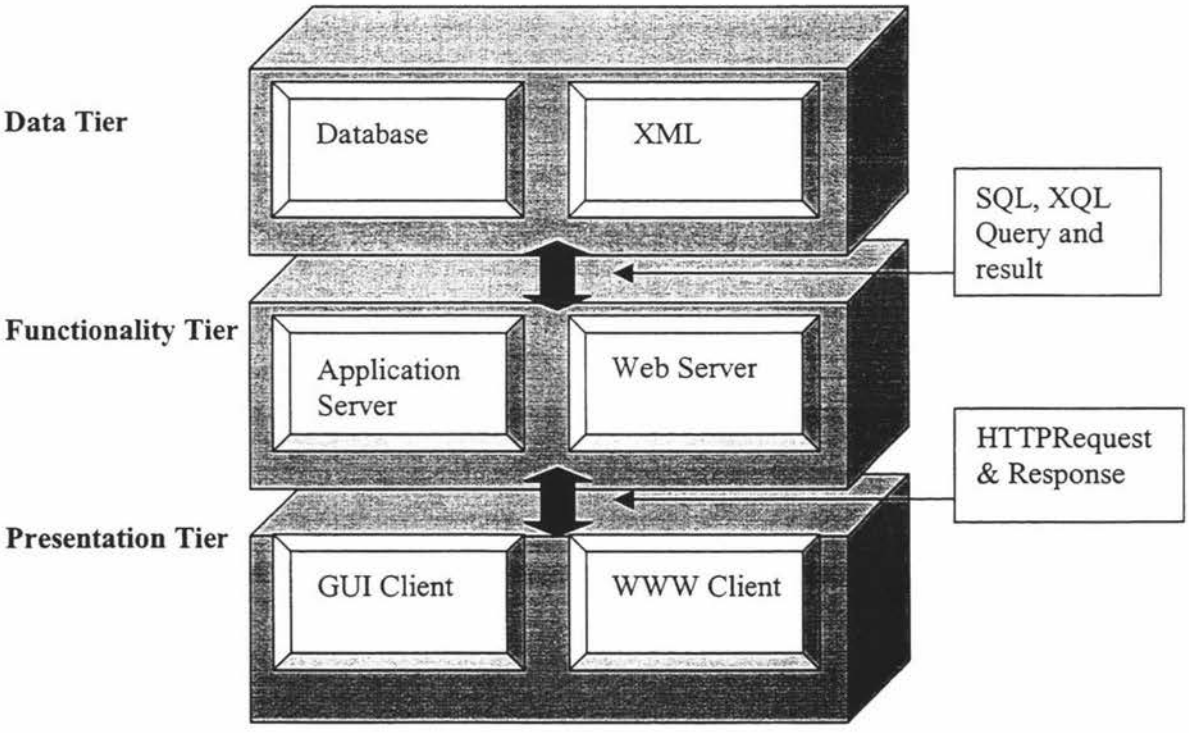

Figure 13. 3-Tier Architecture

6.2.1 Presentation tier (on Client)

The presentation tier mainly interfaces with the user on client computer, such as taking the input and presenting the output. In most cases, the client computer runs a Web browser (Internet Explorer or Netscape). Validating the user's data input from the Client provides the most immediate feedback to the user and is a focus of this project. JavaScript, Java Applets and ActiveX controls are client side components which can be used to accomplish this validation. All Web Applications are Event driven. The Server receives HTTP Requests from the client. An HTTP Request is the Event. Specifically, it is a "GET" or a "POST" action.

Advantages:

- Provides immediate feedback to the User
- Avoids additional error in data transmission
- Mimics traditional dedicated applications

Disadvantages:

- Data validation rules are separated from the Data tier
- Maintenance vigilance

6.2.2 Functionality Tier (on Server)

The middle tier in the architecture is the Functionality Tier which normally hosts Web/Application server. This includes the Web Server and the application specific code which is executed by the Web server. The code could be in many forms including Perl scripts, Java Servlet, JSP, C/C++ programs, ASP, etc. It could be a combination of all these things. In our project, we keep the Template Engine at this tier.

Advantages:

- Can be implemented in many languages
- Can sometimes reuse existing code
- More responsive to the user than Data validation on the 3rd tier

Disadvantages:

- User must "submit" the data before it is validated.
- Data checking is done separate from data storage.

6.2.3 The Data Tier (on Server)

The 3rd tier is the Data tier which normally runs a Database Server in most cases. However, the Data can also be embedded in XML. XML eliminates the need of DBMS in the relational databases. The XML provides the common interface for different applications so it is an ideal file format to transfer data. Another prominent solution is to store the data as an object in an Object-Oriented database. The Database or XML in the 3rd tier can be located on a different system other than the Web/Application server. This is because the data is often used by more than one application or process. Even if it is located on the same system, it is still a logically separated tier. Typical relational database servers used today are Microsoft's SQL Server, Access, Oracle and MySQL.

Advantages:

- Single point of validation if many different applications use the same database.
- Easy to manage
- Data consistency rules stored with the actual data.

Disadvantages:

- Furthest point of validation from the User. (Slowest response.)
- Sometimes difficult to implement complex cross-field validation.
- Reports only one error at a time (A User could enter several invalid fields and only be notified of one error at a time resulting in many round-trip attempts to "submit" the data)

6.3 Developing Template Namespace, Template Schema and Template Description in Template Repository

A set of definitions of formal and common vocabulary will be beneficial to the interpretation and construction of the templates. There are two main reasons why a common vocabulary is required to create a template. Firstly, any template, like the document, needs to have a good structure. And each element in the template may appear in different sequence. For example, in a standard research thesis, the title and introduction section will come first, and a summary or conclusion section will come last. All the authors and readers know the conceptual meanings of these two sections. Secondly, a template may have course specific sections in the structure. For example, a template for the course called "Database Concept" may have sections like "Tables", "Record" and "Query". Other courses in Information Systems may share the same set of template and template fragment names. When more fragments are needed for constructing course templates, how can we name them differently? These issues are regarded as namespace in the conceptual modelling process.

6.3.1 XML and Namespace

The XML user-defined tags are the ideal media for describing a template document structure. In terms of common document vocabulary, the developer or teacher of the course may use tags like "<Heading>", "<Title>", "<Definitions>", "<Assignment>" and "<Summary>". In terms of the course specific template tags, developers or teachers may design their own tags, such as "<Database>", "<Table>", "<Relationship>", "<Query>" and "<Schema>".

In order to avoid the confusion over the usage of the template tags and the attributes, we need to adopt XML namespace. An XML namespace is a collection of element type and attribute names. The collection itself is not important -- in fact, the XML namespaces don't actually exist as physical or conceptual entities. What is important is the name of the XML namespace, which is specified by a URI. This allows XML namespaces to provide a two-part naming system for element types and attributes. The first part of the

name is a prefix to identify the different XML namespace -- the namespace name. The second part is the element type or attribute name itself -- the local part, also known as the local name. Together, they form the universal name.

```
<Template>
   <Block>
   <database:Database xmlns:database="http://www.massey.ac.nz/templates/database">
     < database:table>Academic Record</database:table>
     < database:table>Performance Record</database:table>
     </database:Database>
  </Block>
 </Template >
```

6.3.2 Template schema in Template Repository

The Template schema in the Template Repository consists of such a set of XML files which describe the structure of templates. There are normally one or more Template schema files for each template. An example of such a file is listed here.

```
<xsd:element name='template' type='templateType'/>
<xsd:complexType name='templateType'>
   < xsd:element name='table' type='string'/>
   < xsd:element name='relationship' type='string'/>
</xsd:complexType>
```

Logically, a template schema file is separated from the Template Repository which mainly contains the templates. But in implementation, the template schema can be considered as a part of the Template Repository.

6.3.3 Developing Template Description

Here we introduce a few terms: template description, and template document. Sometime, we use template description instead of template because template is a very abstract and general term. It could mean any skeleton documents (output), and documents (input) that are used to make the skeleton documents. It works well in system modelling and conceptual analysis stages when we do not differentiate them. However, when we discuss the physical implementation, it is better to use the specific term to refer to the real documents in the system. Since we are going to use XML and XSLT documents to make the skeleton document, these XML and XSLT documents can be seen as the description of the template. The template document here refers to the skeleton document which is made from template descriptions

A standard Template description file relies on a Template namespace and a template schema to describe a template. At this stage, we are going to build a prototype which does not cover many disciplines. Therefore, we simplify the syntax of our XML based template description. Examples of the template description files can be found in Appendix A. When the template is expressed in XML format, another set of template description files are needed. They are XSL (eXtensible Stylesheet Language) files, the companion of XML. The XSL files are used to transform XML files into another set of XML files, HTML files, or text files. It means that XSL can tailor a template description file (XML) into what users want.

6.4 Developing Template Repository

In Web practice, various "Style Sheets" are widely used to customize the look of documents and templates. For example, CSS (Cascading Style Sheet) is used to render templates for HTML page. CSS contains style rules which define certain presentation parameters for elements (like, fonts, colours and margins). Unlike CSS, XSLT is more widely used to manipulate various documents which are in the form of XML. We choose to represent the template schema and template description in XML, the XSLT will be used to tailor templates depending on the user's requirements.

6.4.1 Example of XSLT file

An example of such a XSLT file is listed in Appendix B.

6.5 Developing Template Engine

In our design, both template description files (XML and XSLT) are stored in the template repository. In order to transform them into a template document, we use Xalan [Apache, 2002] as the "Template Engine". It is a high-performance XSLT stylesheet processor. To read and update - create and manipulate - an XML document, an XML parser is needed. There are basically two categories of XML parsers. One is called DOM. The other is called SAX. In order to respond to a user's request on the Web, another Java technology "Servlet" is used to read the parameters and implement the XML transformation which is the Template Engine.

6.5.1 DOM Vs. SAX

Document Object Model (DOM), a programming interface specification being developed by the World Wide Web Consortium (W3C), allows a programmer to create and modify HTML pages and XML document as full-fledged program object. The Microsoft XML parser is a COM component that comes with Microsoft Internet Explorer 5.0. Once IE 5.0 is installed, the parser is automatically available to scripts inside HTML documents such as Javascript, VBscript and ASP. Here is an example showing how to use Microsoft XML parser in Javascript.

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM"); //create an XMLDOM object
xmlDoc.async="false"; //set synchro mode
xmlDoc.loadXML(xmlfile); //load the xml into xmldom
```

SAX (Simple API for XML) like DOM (Document Object Model) gives access to the information stored in XML documents using any programming language (and a parser for that language).

However, DOM and SAX take very different approaches to providing access to the document information: DOM creates a tree of nodes (based on the structure and information in XML document) and interacts with this tree of nodes. When a document is processed, all the nodes will be read and stored in computer memory. If the document is relatively large in size, it will consume a great amount of system resources. With SAX, even-based approach, the parser tells the application what is in the document by notifying the application of a stream of parsing events. Application then processes those events to act on data. SAX is very useful when the document is large.

6.5.2 Java Servlet

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server, such as Apache. A Servlet is normally thought of as an applet that runs on the server side. They are server- and platform-independent. This leaves developers free to select a "best of breed" strategy for their servers, platforms, and tools. For sure, the Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Servlets can also access a library of HTTP-specific calls and receive all the benefits of the mature Java language. Up to now, Java Servlets have already been widely used in many Web applications.

6.5.3 Example code of template engine

In order to process the template description (XML file and XSLT file), a transformer is needed which is able to transform the template description into the target document. That is what the template engine is designed for. The key component in the transformer is an XML parser. Xerces/Xalan is among the family of parsers for XML from the Apache XML Project. And it translates XML documents into HTML, text or other XML document types. It implements the W3C Recommendations for XSL Transformations (XSLT) and the XML Path Language (XPath). It can be used from the command line, in an applet or a servlet, or as a module in other program. In our

prototype, we adopt Xerces/Xalan in Java servlet to implement the transformation of an XSLT file and an XML file into a HTML document.

An example of the code is listed in Appendix C.

6.6 Building Client application

The client applications in our prototypes are in Web browser interface.

6.6.1 Prototype A

The client application of Prototype A contains a menu tree which serves as a navigation directory. It gives the user a clear picture of where he/she is staying. Users can choose the chapters or sections in each chapter. In the centre of the screen, it shows the content of the chapter or section. On the right side of the screen, the user can select an action from the dropdown list (Modify). A number of commands can be found in this list, such as add chapter, add section, delete, move up, and move down. In order to create a new section, the user will be presented with a series of screens. The user will be asked to choose a template from a list of pre-defined templates, such as Bulleted list form, Paragraph with title form, and Text with picture.

The adaptive prompting in the template selection is realized by changing the sequence of the templates which are available, with the more recently accessed templates located at the top of the list. Figure 15 shows the template selection screen.

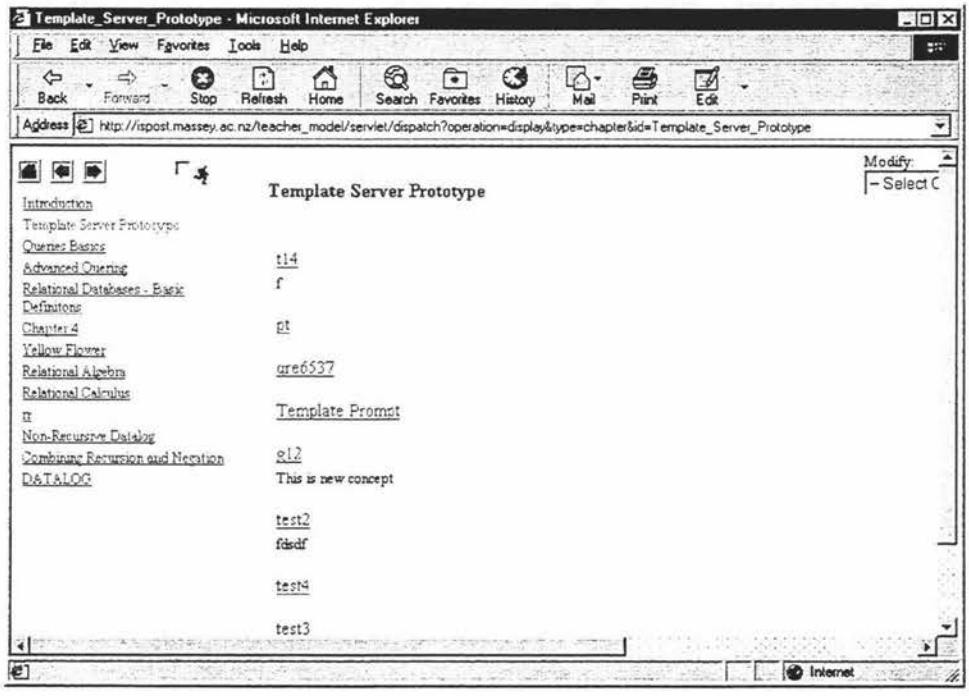

Figure 14. The Interface of Prototype A [Tretiakov & Shi, 2003]

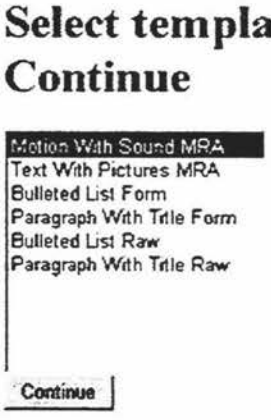

Figure 15. The list of templates

6.6.2 Prototype B

Prototype B followed a design which is slightly different from Prototype A. The client Interface contains a number of frames. One frame shows the adaptive list of templates. At the beginning, a pre-defined set of templates (template index) is displayed. Then, the sequence of the templates changes whenever the user can make a selection on the template list. Another frame is the content frame which presents the related contents in content repository after the user has made his/her selection of the template. The file names of related contents will be displayed here. Otherwise, the contents files for the default templates are displayed. The most important frame is the document view which shows the empty HTML form for entering content and the HTML form with content for updating.

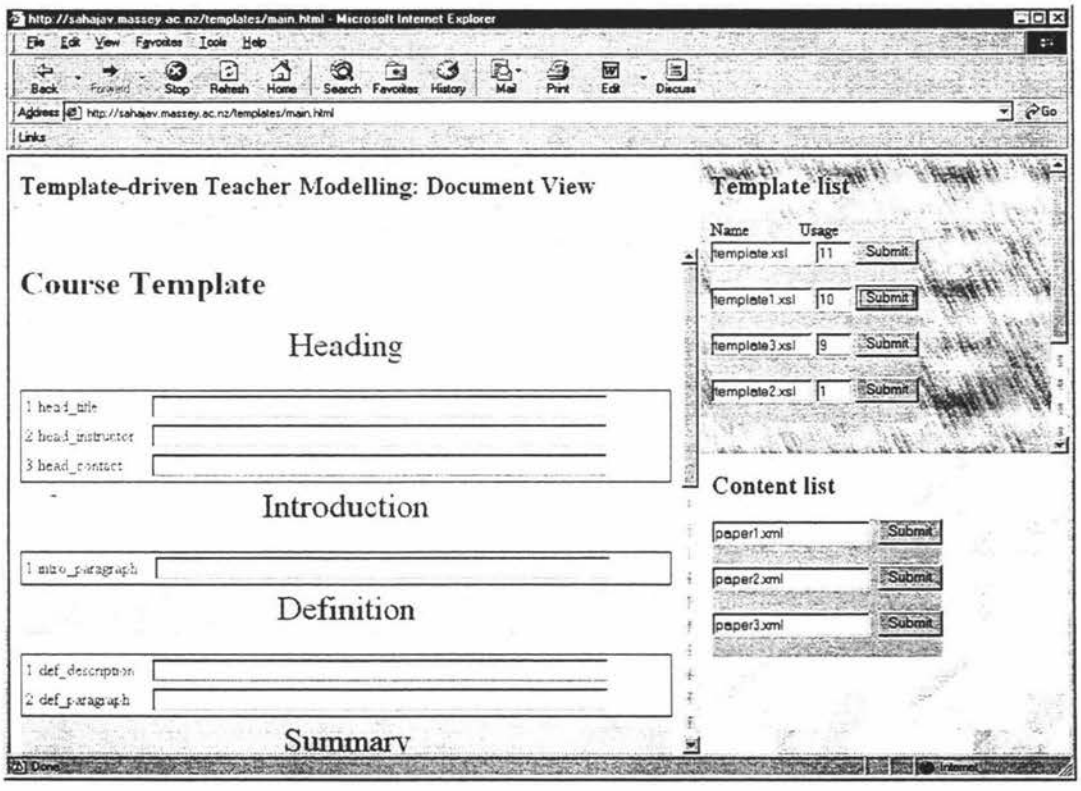

Figure 16. The interface of Prototype B (empty HTML form)

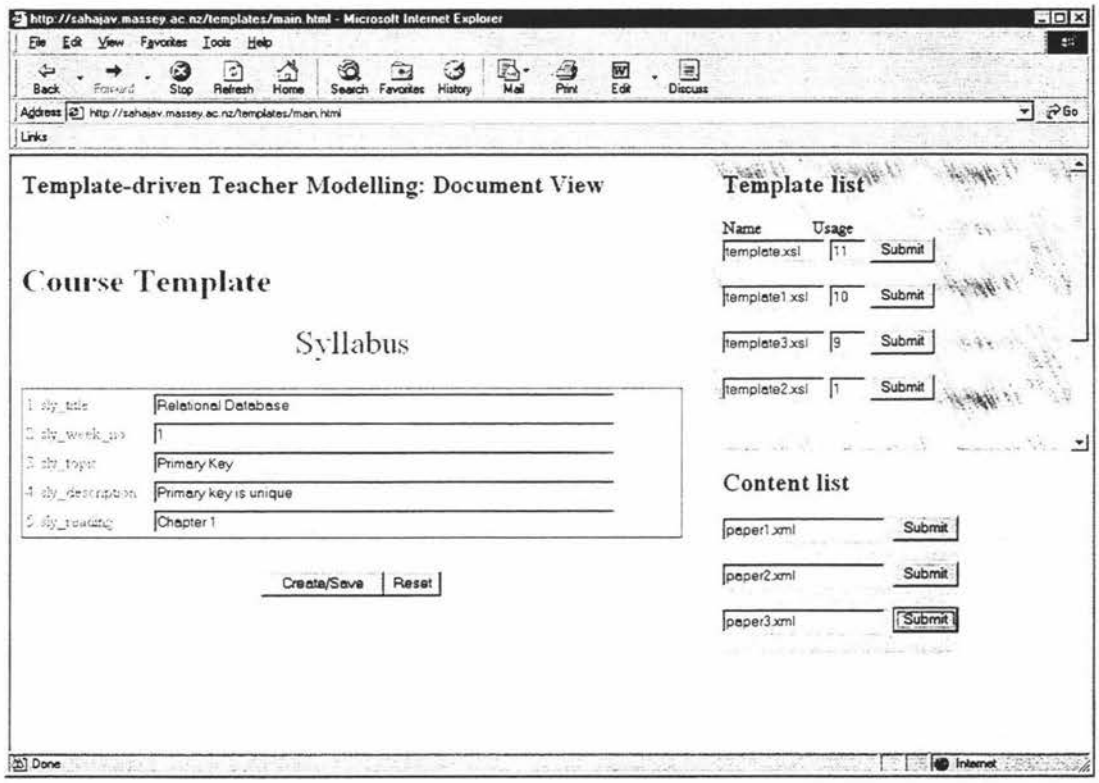

Figure 17. The Interface of Prototype B (with content)

6.7 Building teacher model

The objective of teacher modelling is to keep not only the teacher's explicit characteristics but also implicit aspects regarding his/her behaviour, environmental and dynamic constraints. The teacher profile (or traditional view of teacher model) keeps all the explicit characteristics of teacher, while the data extracted from the template repository can indirectly reflect the implicit aspects of teacher model. As the teacher profile is data intensive, it is ideal to place it on the data tier of the system but it is also beneficial to store it on the presentation tier in the client application. It will reduce the amount of overheads to download or upload all the data over the network. On the other hand, template repository is file-based and is also ideal to be placed on the data tier.

Even in the traditional view of teacher modelling, there is still a wide variety of attributes in the teacher model that could potentially affect the teacher when he/she

teaches. Most system developers solve this problem by selecting only a subset of the available attributes (normally those regarded as the relatively important attributes). Sometimes, an algorithm will be used to assign different weight to individual attribute to classify the users. As a result, the outcome of this attribute selection and design of algorithms process vary from project to project, from developer to developer, and from organization to organization.

Once these attributes are decided, the actual acquisition of data will start. And the data can be store in a relational database or simply an XML file. As building the teacher model is not the main focus of this research, it is unnecessary to create a full-fledged teacher model. Instead, we can create a simple XML to store the basic data about the teacher such as his/her usage of the templates, and use a simple algorithm to calculate the order of the templates in the index page.

As discussed above, the templates in the template repository will play a more important role in this teacher modeling. To simplify the research, we only consider occurrence of a template that represents the preference of a teacher in certain pedagogical circumstances. There are a number of other dynamic, implicit aspects that are related to the use of templates, such as the context of template or the sequence of a set of templates. One template might be a prerequisite of another template.

The occurrence of a template can be easily captured on client application. It can then be stored in a small database or an XML. In our case, we choose XML to store the occurrence of each template.

6.8 Building adaptation engine

The adaptation engine basically is the algorithm which predicts a teacher's action based on the system's knowledge of him/her. The complexity of this algorithm depends on the number of attributes or dimensions in a teacher's model and their relationships. It could be a simple statistical equation or complicated Bayesian Networks, which represents each dimension as a vector. In this research, we choose a single dimension. That is the

usage occurrence of each template by a user. This data will be fed into a sorting loop to select the template which has been used most often. This is a simplified situation, and may not suit many experienced users.

In prototype B, the actual data regarding to the usage of each template is stored at a server initially. It is then downloaded to the client, or specifically in the Web browser. It is written in XML and embedded in JavaScript. In the JavaScript code, we declare an XMLDOM object. The XML file is then loaded into the object for parsing. In order to present the list of templates, the system will carry out a sorting process to decide which template has the highest ranking in the list. Each time, when the teacher selects a template, the system will go through the same sorting process to update the list.

Chapter7 Evaluation of the Template Server Approach

7.1 Introduction

In this chapter, we try to answer the following questions:

1. *Why* do we need to evaluate the Template Server Approach and the prototypes?
2. *What* do we evaluate?
3. *How* do we evaluate?
4. *What* is the result of the evaluation?

7.2 Initiative of Evaluation

The Template Server Approach is an attempt to make an innovative contribution to Web-based education practice. It incorporates a number of new techniques such as teacher modelling, template server, and template reuse. Such a new approach should go through a preliminary and formative evaluation by the education practitioners. The outcomes of the evaluation on the approach and the prototype that it is based on are expected to highlight the strength and weaknesses of the approach in order to make further improvements.

7.3 Definition of Evaluation

In general, evaluation involves judgment of the feasibility of a plan, the innovation of a new approach, or the quality of a product against certain specified criteria. Whether evaluating the policies for the government, or an approach for an educational program, the researchers in the evaluation are observing, or collecting information, and comparing or interpreting that information in order to make a decision as to the feasibility, the innovation, or the value of the object. Formative evaluation is the process of evaluating the effectiveness and efficiency of an approach or a prototype during the development process. Summative evaluation, in contrast, is the evaluation of the completed, final product [Bhola, 1990].

7.4 Stages of Full Evaluation

Before the formative evaluation starts, a physical implementation of the prototype is required. Although we focus on the evaluation on the Template Server Approach, we do need a system which can be presented to the users. A prototype, as we learned from the implementation of information systems, can be a Hi-Fi or Low-Fi model. In this case, a Low-Fi prototype refers to a mock-up or a paper template representing the sample screens while the Hi-Fi Prototype is the real and physical software system or interface which allows teachers to access it for early feedback.

In general, once the prototype is set up, a pilot test on the prototype is expected. The pilot test is conducted on a one-to-one or small group basis. The instructional developers test the prototype with teachers or even final year students who are doing an education degree. The primary aim at this stage is to identify if there are any major deficiencies in the approach and questions of evaluation.

Normally a formative evaluation consists of a second stage which is called field test or trial. The conditions of the field trial will be as close as possible to the actual operational conditions. Taking the example of our case, the Web-based educational system equipped with Template Server should be put in place to simulate the teaching environment as closely as possible.

Once the formative evaluation is complete, a summative evaluation can be initiated to ensure that all the requirements for the product have been met. However, field tests often require the de-bugging, re-designing and improving on the product. Thus the formative evaluation process could be iterative, and circular in order to confirm the usability of the Template Server Approach, and to optimize the effectiveness and efficiency of the prototype.

Due to the relatively tight schedule of this project, it is appropriate to combine two stages: the pilot test and the field test, in the formative evaluation into one stage. The summative evaluation will be written when the whole system is complete.

7.5 The selection of the prototype

As we have built two prototypes, we could select either of them, or both of them as the system we present to the users. Prototype A is more mature than prototype B. It has implemented most of the features that are predicted. Therefore, we choose Prototype A as the system to be evaluated.

Prototype A and B are built using the same Template Server Approach. However, they incorporated different technologies. In Prototype A, templates are expressed in HTML format on the Template Server while they are presented in XML format in Prototype B. The flexibility, scalability and extensibility of XML make it the preferred way to implement adaptability.

The main initiative of this evaluation is to evaluate the Template Server Approach, rather than the underlying technologies. The participants will not notice the difference between HTML-based templates and XML-based templates on the server because what they will see are Web pages. Based on this assumption, using Prototype A or B will not generate any difference in the evaluation result, at least in this current stage.

7.6 The selection of data collection technique

In our evaluation process, a variety of qualitative techniques can be used to gather the data, ranging from direct observations, interviews, and questionnaires. The qualitative evaluation techniques explore the underlying issues, ideas, and values driving users attitudes and behaviours. Another data collection technique is the quantitative approach, such as quantifying the interaction between the user and the prototype, typically when the user clicks on a certain link in a specific period of times. Because our initiative is to explore the new approach to help the teacher publish the course contents, it is suitable to choose qualitative techniques, instead of quantitative. It can be justified from another consideration. As our template server architecture is still in its early stage, adopting the quantitative approach means that a great number of participants are required.

7.7 The design of the evaluation questionnaire

This evaluation is designed to be a formative evaluation and subjective measurement (user's perception). We will not go through all the functionalities the prototype possesses. Most evaluators of the Web-based systems pay their attention to the functionalities and the qualities of the application that can be evaluated by performance measurement, such as laboratory test and examination. However the main focus of this evaluation will be given to the usefulness of Template Server Approach that this project demonstrates. The questionnaire in the evaluation is considered as the recording and collecting of information about the usefulness of the Template Server Approach. Therefore, as the designers, we have endeavoured to avoid bias when constructing the questionnaire.

7.7.1 The selection of the question type

The first choice was to decide the type of questions to be asked. We considered the three kinds of questionnaires, factual-typed questions, opinion-typed questions, and attitude questions [Kirakowski, J]

- **Factual-type questions:** Ask for the observable information. Such as: what education did the respondent get? Normally used for collecting objective data. It would be uneconomic to obtain them in another way.

- **Opinion-type questions:** To discover the thoughts about something or someone. They are used to check out the popularity of something or someone. They direct the respondent's thought outwards.

- **Attitude questions:** Directing thoughts inward, determining the satisfaction with an artefact or design detail. These questions are widely adopted to ask the users about their experiences with Information Systems, Web-based, or multimedia systems.

In our evaluation, we seek to discover the user's attitudes towards working with the Template Server Approach and it's prototype. These attitudes are put into several categories:

- User's feeling of being efficient when he/she use the approach and the prototype.
- The degree to which the user likes the approach and the prototype.
- How helpful the user feels the approach and the prototype
- Does the user feel comfort when he/she use the approach and the prototype
- To what extent the user feels the approach and the prototype improve his/her work.

7.7.2 Likert Scaling

In order to apply the attitude-type questions to our evaluation, we choose Likert scaling. Likert scaling is widely used in the evaluation questionnaires for the research of Information Systems [Liao, 2000]. It often comes with 1-5 or 1-7 scale. The respondent is asked to give 'weight' to the statement from likely to unlikely. Finally, the researcher can add the score on each statement from each respondent to get a user's overall attitude to a product or an approach. Or he/she can also calculate the average score for each statement when there are many respondents in a standard evaluation.

7.8 The reliability and validity of the evaluation

The questions in a poorly designed evaluation will be insensitive to the differences among various applications. It means that the same set of questions can be used for evaluating different applications. That will lead to a serious reliability problem. When we design the questions in the evaluation, we reference many evaluation formats in HCI usability test (QUIS, PUEU, CSUQ, and PUTQ) [Perlman, G. 1998]. We then go

through each question and ask whether it is related to the Template Server Approach and the prototype, and whether it is consistent to the objective of our evaluation.

Another common threat to the reliability of the evaluation is the inadequate selection of participants in the evaluation or respondents to the questions. As this evaluation is designed as a pilot test, the participants will be mainly internal staff. But we try to recruit participants from diverse backgrounds, i.e. different teaching experience and different disciplines. We first produce a list of candidates. Then we randomly select participants from this list to remove the bias in the selection.

The validity of the evaluation is the degree or extent to which the evaluation is actually measuring or collecting the data that the designers of the evaluation intend to do [Kirakowski, J.]. In designing the evaluation, we did a number of pre-test to ensure that the data we eventually obtained was the data we wanted.

7.9 The manipulation of the prototype

We have discussed the client application or interface of Prototype A in previous chapters. We also covered the underlying architecture of the Template Server Approach. But the user of the prototype still needs to know the structure of the teaching or course unit before he/she can confidently use the prototype.

A teaching unit contains a number of chapters in Prototype A. Each chapter then consists of a few sections. The user is able to create chapters, sections and concepts. In most cases, the user can edit the sections and concepts. When the user creates a new section, he or she will be presented with a number of templates, such as the bulleted list form, text with pictures, and paragraph with title form.

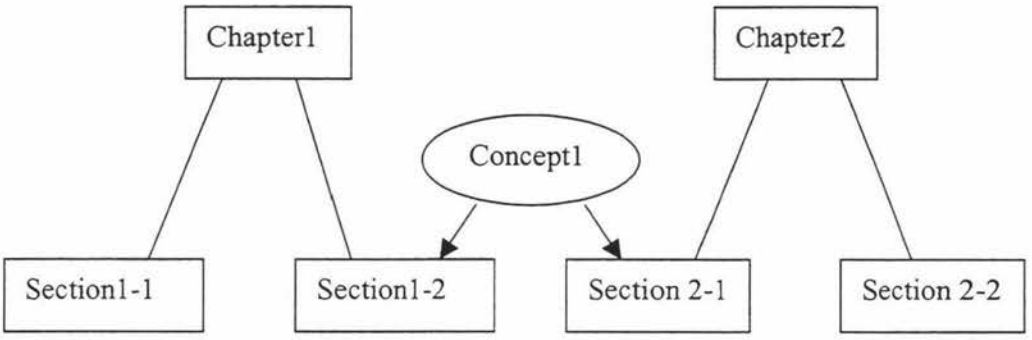

Figure 18. A graphical description of the course structure (a teaching unit) in the Prototype A

7.10 The result and analysis of the evaluation

The evaluation for the Template Server Prototype is administered between July 10, 2003 to August 14, 2003 at Department of Information Systems, College of Business, Massey University. 7 teaching staff participated the evaluation. The data reported here report on what respondents think about the Template Server Approach and the prototype.

The 13 statements in the evaluation were responded on to a 1-7 Likert Scale.

1= Strongly Disagree

7= Strongly Agree

The neutral value is 4. Any value between 4 and 7 represents a favourable response.

The evaluation statements and their average responses are listed in the following:

Q1.I would like to use templates served by template server over the networks

Q2.I would like to use templates designed by others.

Q3.I would like to use more templates than that are offered by the prototype.

Q4.I would appreciate the way that the order of the templates is rearranged each time.

Q5.Using the Template Server Approach would make it easier to do my job.

Q6.Using the Template Server Approach would minimize the chance I make mistakes in my job.

Q7.Learning to operate the prototype would be easy for me.

Q8.I would find it easy to get the Template Server to do what it is supposed to do.

Q9.Understanding the Template Server Approach would be easy and straightforward.

Q10. I would find the Template Server to be flexible to interact with.

Q11. Using the Template Server Approach would improve my overall job performance.

Q12. Using the Template Server Approach in my job would increase my productivity

Q13. I would adopt the Template Server Approach in my teaching practice.

Though we use the qualitative approach to gather the data, we use the quantitative approach to analyse the data. This is because of the difficulty to generalize the respondents' overall satisfaction towards the prototype if we analyse the open-ended questions in the data collection process. The answers to those open-ended questions sometimes are not comparable to each other. Since we use the likert-scale (1-7) to quantify the respondents' reply to the questions, we could rely on the statistical tools to conclude our evaluation.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	
Participant1	7	7	7	7	7	7	7	6	6	6	6	7	6	
Participant2	6	6	NA	6	7	2	4	5	4	6	5	6	6	
Participant3	7	7	7	3	7	7	7	5	6	5	6	6	6	
Participant4	6	5	7	3	7	6	7	6	6	6	7	7	6	
Participant5	5	3	5	4	5	5	5	5	5	5	5	5	5	
Participant6	7	6	7	5	4	4	7	6	6	5	2	2	4	
Participant7	5	2	2	NA	6	2	7	6	6	5	5	5	5	
Participant8														
Average score	6.14	5.14	5.83	4.67	6.14	4.71	6.29	5.57	5.57	5.43	5.14	5.43	5.43	5.50
Satisfaction	88%	73%	83%	67%	88%	67%	90%	80%	80%	78%	73%	78%	78%	79%
Average Deviation	0.67	1.11	0.64	1.33	1.11	1.50	1.11	0.50	0.67	0.50	1.17	1.33	0.67	
Standard Deviation	0.82	1.51	0.89	1.63	1.33	1.94	1.33	0.55	0.84	0.55	1.72	1.87	0.84	

Table 3. Evaluation result

We can see from the data table that the overall satisfaction towards the prototype and Template Server Approach is 5.51 or 79%. The satisfaction to Question1, Question 3, Question 5 and Question 7 are around 90%. It indicates that most users like to use Template Server Approach to publish the course content over the networks. They agree that the Template Server prototype is easy to learn and will make their work easy to do. Apparently, users would like to see more templates on the server.

The Question 4 received the lowest satisfaction rate (67%). Some users particularly do not like the way that the order of the templates is re-adjusted according to user's behaviour. It is a part of our initiative to adaptively reflect the user's intention to help user using the system in a more efficient way. It does not indicate deficiencies in the Teacher Modelling and the implementation of the Teacher Modelling. We are required to search for better solutions to the presentation of the Teacher Modelling.

We can see from the answers to Question 6, 11 and 12 that responses are diversely distributed. These three questions reflect the features of the prototype rather than the principle of Template Server Approach. One of the explanations is that the prototype is still not mature enough to compare to most commercial courseware products. The users may feel disappointed when they can not find the features that are common to those products. It does not meet their expectations of increasing their productivity and minimising the chances of making mistakes.

The participants also gave us many suggestions during the evaluation. But most of them are related to the physical aspects of the prototype instead of the underlying Template Server Approach, such as:

- The back button is not working in the browser.
- Should provide features that allow the user to choose the file folder to save their documents.
- Need the navigation facilities.

Chapter8 Conclusion, Discussion and Future Work

8.1 Abstract

So far, a new Template-driven Teacher Modelling Architecture has been described. And two prototypes based on the new architecture are also partially created to demonstrate the new architecture and the Template Server Approach. The objective of this chapter is to provide a short conclusion of this project and also gives a preliminary discussion on the future improvements to the architecture and prototypes.

8.2 Main contributions of this thesis

Due to the complexity of the teaching processes, knowledge acquisition and teacher modelling, most research projects in Web-based education start with a modular and scalable framework or architecture to explain the abstract concepts of the system. Then a prototype system will be built based on the proposed framework or architecture to demonstrate the feasibilities of the system. In our project, we:

- Propose a new Template-driven Teacher Modelling Architecture which is designed to benefit teachers in making more flexible and effective Web-based courses. In this architecture, we create a new concept called Template Server Approach. The Template Server Approach encourages teachers requesting templates over the networks. Teachers could select templates from the adaptive template prompt (adaptive template index) that is under the control of the teacher model.
- Suggest teacher modelling techniques based on templates usage. The collection of accumulated templates in the template repository for a teacher or a group of teachers are selected as the inputs for the inference mechanism in teacher's model to calculate the best representation of the teaching strategy, and then predict teacher intention when he or she interacts with the system.
- Build two prototypes demonstrating the initial implementation of the proposed architecture. The prototypes are created in such a way that users can play with

them, and get familiar with the main features of Template-driven Teacher Modelling Architecture.

- Evaluate one of the prototypes (prototype A) to collect users' attitudes towards our new Template Server Approach. (The reason why prototype A is chosen for evaluation is discussed in 7.5 "The selection of the prototype")

8.3 The benefit of XML in Course Authoring

XML is extensively adopted in this project. Using XML-based technologies in course authoring systems provides an open, powerful infrastructure that allows for the specification and implementation of the course content, the course templates and the user models. Storing the information in XML also provides the flexible approach to its presentation in different document formats. Transformation via XSLT gives the facility to build various templates and documents that meet the requirements from different users. Furthermore, new technologies are emerging in XML family. For example, XLink allows the building of rich hypermedia. XQL or XML-QL (a query language for XML) [W3C, 1998] allows intelligently retrieving the XML data.

We have used XML for storing and transferring data among applications, such as Javascript code in the browser and Java servlet on the server. They are running on different platforms (Windows and Unix). XML has demonstrated the advantages of global accessibilities and portability. One of the weaknesses of the XML is that it takes longer time to process XML files when it is big in size. Because of the demonstrative nature of our prototype, the XML files are relatively small in size. We can tell how slow it would be when the file size increases. There are also other factors, such as the capacity of RAM of the machine, which can slow down the processing speed. Regarding to XML editors, we have used Cooktop, HTML-Kit, and XMLEditPro in our project. Cooktop is the most comprehensive XML editor among them. The XSLT transformation that comes with Cooktop impresses us intensively.

8.4 Teacher Model

Apart from the template Server Approach, teacher modelling is another focus of this project. As we mentioned in the previous chapters, we simplified the teacher model by choosing one attribute from the teachers' interaction with the system. That is the usage of each template in the system. Obviously, many other factors and decision-making processes will affect the teacher's mind in selecting template. We can perceive the teacher model as a subsystem which takes in data, processes the data, and comes up with possible speculations on what template the teachers will select. This involves many issues in building the system. Firstly, designers should agree on what attributes (factors) will be considered in the teacher model, and their weights in the calculation because some attribute and factor are more influential than the others. Secondly an appropriate algorithm will be chosen to calculate the data to classify and cluster the teachers. And it will come up with the possible result. The ideal solution is the Bayesian Networks in Artificial Intelligence research which could lead to the next step of this project.

To fine-tune the teacher model, we also suggested teaching strategy concept in teacher modelling. Since teachers tend to choose hybrid approaches in their teaching, and sometimes change the ways they teach in terms of the course nature and student academic level. We proposed a number of teaching strategies under each teacher model. It doesn't mean that there are only three teaching strategies, "Instructivism, Constructivism and Social-Cultural theories".

Apart from the discussion on the creation of teacher model, we also need to look at the teacher model from the perspective of its usage. We have discussed the importance of teacher model in the course authoring system which focuses on planning and designing a course. We haven't discussed the effect of teacher models in the tutoring system which focuses on assisting student's learning activities in a Web-based environment. The teacher models in these two systems share some similarities. For example, they both tried to embed teacher's teaching styles into their systems. In the course authoring system which employs the Template-driven Teacher Modelling Architecture, the teaching style into a teacher is identified by monitoring the teacher's behaviour when he or she creates the sites. And the teaching style is implicitly indicated by the highest occurrence of a group of templates in our project.

In a tutoring system, the question is how the teacher modelling helps student learn knowledge. If the teacher's teaching style has been incorporated into the Web-based course, is that possible to shift the teacher's previous focus away from the style that students prefer when it is delivered? This could happen if the student model negotiates with the teacher model or teacher model negotiates with the student model. Then the result of interaction could be passed onto the system to make adequate changes while the system is still running.

A system model assumes a specific intention of the person whom it represents. How to judge the effectiveness of a teacher model in a Web-based educational system becomes crucial when the system is fully implemented. The solution we suggested is a usability test. It works well to find faults and suggest improvements. But it doesn't indicate how accurate the teacher model in the course authoring system anticipates the teacher's intention.

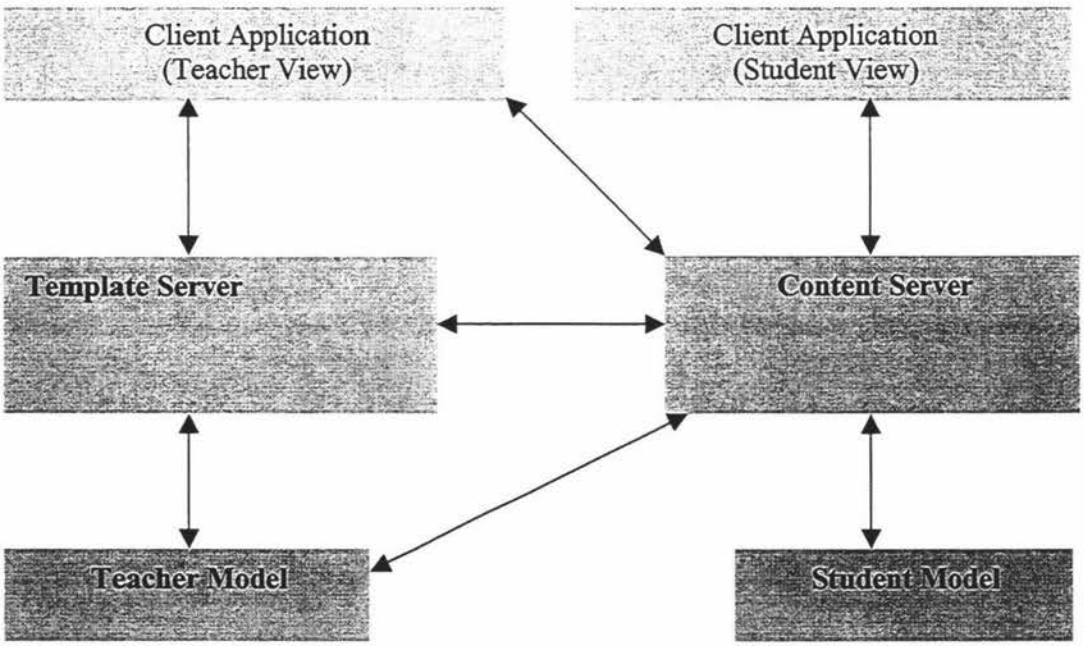

Figure 19. Integration of teacher model and student model in Web-based educational system

8.5 Integration of student model and teacher model

The course authoring and tutoring modules are the two main images of Web-based educational systems. Up to now, we have discussed how adaptive content presentation and student model in intelligent tutoring system support students' learning, and how the template server and teacher model support teachers' making courses on the Web. It has not been previously attempted to integrate the student model and teacher model to make the fully functional Web-based educational system. In this system, the student model and teacher model will be able to interact with each other to change the content and the presentation template. Students will benefit from this new system.

8.6 Evaluation and Usability Test

The objective of the evaluation (pilot-test) is to find the strength and weakness of the prototype and suggest improvements to fine-tune the prototype. Afterward a field test or usability test will be conducted. Human participation is vital in these evaluations and field tests, and they cannot be substituted by any other solutions. When human participants (real teachers) are involved in the formal usability test, serious ethical concerns may be raised, such as anonymity, confidentiality, potential harm and conflict of interest. Consequently, a sufficient ethical evaluation is also required and an application for such an evaluation will be presented to the Ethical Council of the institute.

In terms of the procedures of the usability test, all the participants will be informed about the objective and the nature of this project. If they are interested in testing the new teacher modelling technique, corresponding architecture and the usability of the client application, they will be requested to sign a consent form. Then, the participants can log into the system which is hosted on the server. Participants can log into the system as anonymous users, or any user names they make up. Therefore, there are no

ways to find the true identity of users. When they submit feedback, no real names and other identity information is required. The pilot test, after the participants finished interacting with the system, they are requested to answer a questionnaire which contains another set of questions which is more comprehensive than those in pilot test. They are also invited to report on their experience of the test and give suggestions about it.

8.7 Future Work

Future work will concern with the full implementation of the prototype and usability test of the system with focus on the reflection of effective teacher modelling on the template. Moreover, new techniques will be considered when we examine the actual behaviour of a number of teachers and fine-tune the template server and the teacher models accordingly. Furthermore, it is the responsibility of the designers of the Web-based educational system to find out how to judge the effectiveness of a teacher model.

Reference:

Abtar, K. (1997). "Smart Learning: Multimedia Design Beyond Objectivism" *Proceedings of the ICCE 1997*. Kuching, Sarawak, P354-363.

ADL, (2002) "SCORM Overview" at http://www.adlnet.org/ Accessed on 4/7/2002
LTSC, (1998) "Learning Object Model Document" IEEE Learning Technology Standards Committee. http://itsc.ieee.org/doc/wg12/LOMdoc2_5a.doc

Ainsworth, S.E., Underwood, J.D. & Grimshaw, S.K. (1999) Formatively Evaluating REDEEM: An authoring environment for Intelligent Tutoring Systems. *In Lajoi, S. & Vivet, M. (Eds) Artificial Intelligence in Education Open Learning Environments: New Computational technologies to support learning, exploration and collaboration*, pp93-100. Amersterdam: IOS press.

Aikenhead, G. S. (1997) "Teachers, Teaching Strategies, and Culture" in *Globalization of Science Education: International Conference on Science Education*, pages 133-136, Seoul, South Korea

Anderson, B. (2001) "What is an ontology?" Ontology works, Inc. accessed at: http://www.ontologyworks.com/docs/what-is-ontology.pdf on August 21, 2002

Anderson, J. A., Reder, L. M., & Simon, H. A. (1997) "rejoinder: Situative versus cognitive perspectives: form verse substance." *Educational Researcher 26(1), 18-21.*

Apache, (2002) accessed at http://xml.apache.org/xalan-j/ on August 21, 2002

Banathy, B. H. (1992) "A system view of education: Concepts and principles for

effective practice" from "Web –based instruction". *Educational Technology Publication, Inc.* Englewood cliffs, New Jersey 07632.

Bayne, S. & Cook, J. (2002) "WebCT vs BlackBoard? An Evaluation of Two Virtual Learning Environments" University of Bristol at http://www.ltss.bris.ac.uk/interact21/in21p04.htm (accessed May 27, 2002)

Bennett S. (2001) "Schaum's Outline of UML" McGraw-Hill International (UK) ISBN 0-07-709673-8. Printed in Italy by L.E.G.O.Spa, Vincenza

Berryman, Sue E. (1993) "Designing Effective Learning Environments: Cognitive Apprenticeship Models". New York: *Institute on Education and the Economy*. At: http://watserv1.uwaterloo.ca/~acpalmer/berryman.html accessed on 20/06/2002

Bhola, H. S. (1990). Evaluating "Literacy for development" projects, programs and campaigns: Evaluation planning, design and implementation, and utilization of evaluation results. Hamburg, Germany: UNESCO Institute for Education; DSE [German Foundation for International Developement]. xii, 306 pages.

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

Bransford, J.D., Franks, J.J., Vye, N.J., & Sherwood, R.D. (1989). "New approaches to instruction: Because wisdom can't be told". In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning.* (pp. 470-497). New York, NY: Cambridge University.

Bayne, S and Cook, J (2003) "An Evaluation of Two Virtual Learning Environments"

Learning Technology Support Service, University of Bristol, United Kingdom, accessed at http://www.ltss.bris.ac.uk/interact21/in21p04.htm on 21/11/2003

Britain, S. and Liber, O. (2000), A Framework for the Pedagogical Evaluation of Virtual Learning Environments, at ALT-C 2000, UMIST

Brooks, J. G & Brooks, M. G. (1999) "In Search of Understanding: The Case for Constructivist Classrooms (revised edition)"*Association for Supervision & Curriculum Development* ASCD; ISBN: 0871203588;

Brusilovsky, P. (1995), "Adaptive learning with WWW": The Moscow State University Project, in: P. Held and W. F. Kugemann (Eds.), *Telematics for Education and Training.* IOS, Amsterdam, 1995, pp. 252–255.

Brusilovsky, P. (1996) "Methods and techniques of adaptive hypermedia". *User Modeling and User Adapted Interaction.* 6, 2-3, 87-129

Brusilovsky, P. (1998a) "Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies"

Brusilovsky, P. (2001) "Adaptive Educational Hypermedia" In: *Proceedings of Tenth International PEG conference, Tampere, Finland, June 23-26, 2001*, pp. 8-12.

Brusilovsky, P., Eklund, J., and Schwarz, E. (1998b) "Web-based education for all: A tool for developing adaptive courseware". *Computer Networks and ISDN Systems* (Proceedings of Seventh International World Wide Web Conference, 14-18 April 1998) 30 (1-7), 291-300.

Brusilovsky, P., Schwarz, E., and Weber, G. (1996) A Tool for Developing Adaptive Electronic Textbooks on WWW. In *Proceedings of WebNet'96 - World Conference of the Web Society, October 16-19, 1996*. San Francisco, CA, AACE. - pp. 64-69.

Buntine. W, (1996)
A guide to the literature on learning probabilistic networks from data.
IEEE Transactions on Knowledge and Data Engineering, 8:195-210, 1996.

Burns, R. B. (1980) "Essential Psychology" MTP Press Ltd P4.

Cambridge Dictionary, (2002) Cambridge Dictionaries Online, accessed on 10th, November 2002, at http://dictionary.cambridge.org/define.asp?key=metaphor*1%200

Carro, R. M., Pulido, E., Rodriguez, P. 1999. "Dynamic generation of adaptive Internet-based courses" In *Journal of Network and Computer applications, 22*, pp.249-257

Costabile, M. F. Piccinno, A., Fogli, D. Mussio, P. (2003) "Software shaping Workshops: Environments to Support End-User Development" Accessed at http://giove.cnuce.cnr.it/chi/doc/PositionPaper-EUD-CHI-Costabileetal.pdf on 30/12/2003

Collins, A., Brown, J. S., & Newman, S. E. (1989). "Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics". In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum Associates.

Dabbagh, N. H. and Schmitt, J (1998) "Redesigning Instruction Through Web-based Course Authoring Tools". *Educational Media International* v.35 n2, p. 106-110.

De Bra, P. & Calvi, L. (1998) "AHA: a Generic Adaptive Hypermedia System" *Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia*

HYPERTEXT'98, Pittsburgh, USA, June 20-24

De Bra, P. 1999 "Design Issues in Adaptive Hypermedia Application Development" *In Proceedins of 2nd Workshop on Adaptive Systems and User Modelling on World Wide Web*. Toronto and Banff, Canada, pp. 29-39

Dee Fink, L. 2003 "Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses" Jossey-Bass, ISBN: 0-7879-6055-1

Derry, S. (1996). Cognitive Schema Theory in the Constructivist Debate. In *Educational Psychologist*, 31(3/4), 163-174.

Dick, W. (1995). "Instructional design and creativity: A response to the critics." *Educational Technology,* July/August *35* (4), 5-11.

Dick, W. and Carry, L. (1990), "The Systematic Design of Instruction" Harper Collins Publishers.

Ernest, P. (1995). The one and the many. In L. Steffe & J. Gale (Eds.). *Constructivism in education* (pp.459-486). New Jersey: Lawrence Erlbaum Associates,Inc.

Frank, (2003). Document Conversion to XML Accessed as http://www.frank-it-beratung.de/doc2xml/tutorials/tutorial1_en.html on 21st August, 2003

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

Garlatti, S., Iksal, S. & Kervella, P.(1999) "Adaptive On-Line Information System by means of a Task Model and Spatial Views." In Brusilovsky, Peter, and Paul De Bra (eds.), *Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW*, 1999, pp. 55-69. Available from the World Wide Web: http://wwwis.win.tue.nl/asum99/contents.html.

Goldstein, I. (1982) "The genetic graph: A representation for the evolution of procedural knowledge". *Intelligent Tutoring Systems*. Sleeman D. and Brown, S. J. editors. Computers and People Series. Academic Press, Inc., London.

Gruber, T. R. (1993) "A translation approach to portable ontologies." *Knowledge Acquisition*, 5(2):199-220, 1993

Hamada, K. and Scott, K. (2000) "Anthropology and International Education via the Internet: A Collaborative Learning Model". *The Journal of Electronic Publishing*, September, 2000, Volume 6, Issue 1, University of Michigan Press

Hedberg, J. G., & Harper, B. M. (1998) "Visual Metaphors and Authoring". ITFORUM, Paper No 25. at http://itech1.coe.uga.edu/itforum/paper25/paper25.html accessed on 15/06/2002

Henze, N and Nejdl, W (1999) "Adaptivity in KBS Hyperbook System" accessed at http://www.contrib.andrew.cmu.edu/~plb/WWWUM99_workshop/henze/henze.html on 2/7/2002

Hofstetter, F. T (1997) "Multimedia Literacy" McGraw-Hill Companies, PO Box 182605, Columbus, OH 43218-2605

Horton, S (2000) "WEB TEACHING GUIDE" *A practical approach to creating course Web sites*, Yale University Press

IMS, (2003) IMS Learning Resource Meta-data Specification. Accessed at http://www.imsglobal.org/metadata/index.cfm#version1.2.2 on 5/9/2003

Iivonen, M. & Sonnenwald, D. H. (1998) "Analyzing and Understanding Cultural Differences: Experiences from Education in Library and Information Studies" *64th IFLA (International Federation of Library Associations) General Conference,* August 16 - 21, 1998, Amsterdam, at http://www.ifla.org/IV/ifla64/077-155e.htm (accessed August 7,

2002)

InterBook 1999 "Adaptive educational Hypermedia on WWW" accessed at http://www.contrib.andrew.cmu.edu/~plb/InterBook.html on 1/7/2002

Jonassen, D. (1991a). "Objectivism vs. Constructivism: Do we need a new philosophical paradigm?". *Educational Technology Research and Development*, 39(3), 5-14.

Jonassen, D. (1991b). "Evaluating constructive learning". *Educational Technology*, 31 (9), 28-33.

Kaplan, C. (1993) "Adaptive hypertext navigation based on user goals and context". *User Models and User Adapted Interaction* 3(3), 1993-220.

Khan, B. H. (ed.) (1997), "Web Based Instruction." Englewood Cliffs, New Jersey: *Educational Technology Publications.*

Kinshuk & Patel, A. (1996) "Intelligent Tutoring Tools: Redesigning ITSs for Adequate Knowledge Transfer Emphasis" In Lucas, C. (ed): *Proceeding of 1996 international conference on Intelligent and Cognitive systems. IPM, Tehran* (1996) 221-226

Kinshuk & Patel, A. Oppermann, R., Russell, D (2001) "Role of Human Teacher in Web-based Intelligent Tutoring system". *Journal of Distance Learning*, 6(1), 26-35.

Kinshuk, Tretiakov, A. Hong, H. Patel, A. (2001) "Human Teacher in Intelligent Tutoring System: A Forgotten Entity!" *ICALT 2001,* 227-230

Kirakowski, J. Questionnaires in Usability Engineering: A list of Frequently Asked Questions (3rd ed.), http://www.ucc.ie/hfrg/resources/qafq1.html

Kozel, K. (1997). "The classes of authoring programs". *EMedia Professional* . Vol. 10, No. 7

Kristensen, A. 1999 "Template resolution in XML/HTML" accessed at http://www-uk.hpl.hp.com/people/sth/doc/trix.html" on 16/09/2002

Kumar, V. 2002 "Data Mining Algorithms" University of Minnesota/AHPCRC

Langley P. (1997) "User Modeling in Adaptive interfaces". In *Proceedings of the 21th German Annual Conference on Artificial Intelligence, Freiburg, Germany, Springer, 1997.*

Lave, J. and Wenger, E. 1991 "Situated learning: legitimate peripheral participation" New York: *Cambridge University Press.*

Liao, Z. 2000 An Empirical Study on Organizational Acceptance of New Information Systems in a Commercial Bank Environment, Proceedings of the 33rd Hawaii International Conference on Systems Sciences.

Luria, A. R. (1979) "The Making of Mind: A Personal Account of Soviet Psychology." *Harvard University Press 1979. Revised as "Cultural Differences in Thinking*" at http://marxists.org/archive/luria/works/1979/culture.htm accessed on 18/6/2002

McLellan, H. (1996) Being digital; Implications for Education. *Educational Technology,* November/December 5-20.

McLoughlin, C. and Oliver, R. (2000) "Designing learning environments for cultural inclusivity: A case study of indigenous online learning at tertiary level" *Australian Journal of Educational Technology* 2000, 16(1), 58-72.

McMahon, M (1997) "Social Constructivism and the World Wide Web - A Paradigm for Learning." *ASCILITE 97 conference papers,* The Australian Society for Computers in Learning in Tertiary Education, at http://www.curtin.edu.au/conference/ASCILITE97/papers/Mcmahon/Mcmahon.html

McManus, T. (1995). "Special considerations for designing Internet based education". *Technology and Teacher Education Annual, 1995,* Willis, D., Robin, B., Willis, J. (Eds); Charlottesville, VA: Association for Advancement of Computing in Education.

Moore, M. G. (1989) "Three types of interaction." *The American Journal of Distance Education*, 3(2), 1-6.

Murphy, K. (2001) "A Brief Introduction to Graphical Models and Bayesian Networks" access at http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html on 15/09/2002

Noddings, N. (1990) "Constructivism in mathematics education" in Davis, Maher and Noddings (Eds.) *Constructivist views on the teaching and learning of mathematics*. JRME Monograph, Reston, Virginia, NCTM

Pearl, J. (1988), Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.

Perlman, (G. 1998) Web-Based User Interface Evaluation with Questionnaires, accessed at http://www.acm.org/~perlman/question.html on June 5, 2003

Rennebohm, K. and Pullman, F. (2001) "Teacher's Guide to International Collaboration on the Internet", Washington. Accessed on 24/6/2002 at: http://www.ed.gov/Technology/guide/international/startingaproject.html

Rich, E. (1979): "User Modeling via Stereotypes". *Cognitive Science*, 3:329-354

Rich, E. (1983) Users are Individuals: Individualizing User Models. *Int.Journal of Man-Machine Studies*, 3, No.18, pp.23-46

Robie, J. (1999) "The Design of XQL" Texcel Research, accessed at http://www.ibiblio.org/xql/xql-design.html on 1/9/2002

Romiszowski, A. J. (1997) "Web-based Distance Learning and Teaching: Revolutionary Invention or Reaction to Necessity?" from "Web –based instruction". Educational Technology Publication, Inc. Englewood cliffs, New Jersey 07632.

Ronkowski, S (2002) "TAs as Teachers: A Handbook for Teaching Assistants at UCSB" Chapter 3, TA Development Program, Office of Office of Instructional Consultation, University of California. at http://www.id.ucsb.edu/IC/TA/hdbk/main.html accessed on 19/6/2002

Rumelhart, D. E. (1981) "Schemata: The building blocks of cognition" in Guthrie J. T. (Ed.), *Comprehension and Teaching: Research Reviews* (pp. 3-26). Newark, DE: International Reading Association.

Sherry, L. (1995). "Issues in Distance Learning". *International Journal of Distance Education*, 1 (4).

Sison, R. (1998) "Student Modeling and Machine Learning" *International Journal of Artificial Intelligence in Education*, 9, 128-158

Smith, M. (1999) "Constructivist Theory in Instructional Design: Evaluating the Role of the Learner". *The Internet Source for Schools, Volume III, Fall 1999. International School Services*, Princeton, New Jersey

Specht, M. (1998) "ATS-Adaptive Teaching System" *ERCIM News No.34* - July 1998

Stauffer, K. (1996) "Applications of Student Modeling" Accessed at http://ccism.pc.athabascau.ca/html/students/stupage/Project/sm_app.htm on August 12th, 2002.

SVG, (2003) W3C "Scalable Vector Graphics (SVG) 1.1 Specification" Accessed at http://www.w3.org/TR/SVG/ on November 30th, 2003

SUN, (2002) "Java Blue Prints—MVC" Accessed at http://java.sun.com/blueprints/patterns/MVC-detailed.html on August 12th 2002.

Terzi, S. (2003) "Teacher-Student Interactions in Distance Learning" III International Education Technology Conference and Fair, North Cyprus, 2003

Tretiakov, A. & Shi, Y. (2003) "Template Server Architecture" ICALT 2003, The 3rd IEEE International Conference on Advanced Learning Technologies, Athens, Greece, July 2003.

User Modeling, (1997) UM 97 Reader's Guide." *User Modeling: Proceedings of the Sixth International Conference, UM97.* On-line proceedings, 1997. Available from the World Wide Web: http://www.um.org

Vassileva, J. (1996) "A task-centered approach for user modeling in a hypermedia office documentation system". *User Modeling and User Adapted Interaction*, **6**, (2-3), 185-223.

Vrasidas, C. (2000) "Constructivism versus objectivism: Implications for interaction, course design, and evaluation in distance education" *International Journal of Educational Telecommunications*, 6(4), 339-362.

Virvou, M. & Moundridou, M (2001) "Student and Instructor Models: Two Kinds of User Model and Their Interaction in an ITS Authoring Tool". *Bauer, M. Gmytrasiewicz P.J. & Vassileva, J. (eds), User Modeling 2001:* 8th International Conference, Springer-Verlag, Berlin Heidelberg.

Vygotsky, L. S. (1978). "Mind in Society: The Development of Higher Psychological

Processes." *Harvard University Press.* P52 -91.

W3C (1998) "XML-QL: A Query Language for XML" at http://www.w3.org/TR/NOTE-xml-ql/ accessed on 11/8/2002

WebCT, (2002) "Who uses WebCT?" at http://www.webct.com/company/viewpage?name=company_webct_customers accessed on 1/7/2002

Wenger, E. (1987). "Artificial Intelligence and Tutoring Systems". *Morgan Kaufmann, Los Altos, CA 94022.*

WINDS, (2002) "Web-based Intelligent Design and Tutoring System" http://www.fit.fraunhofer.de/projekte/winds/index_en.xml, accessed on 17/06/2002

Wonacott, M. E. (2000) "Web-Based Training and Constructivism" *National Dissemination Center for Career & Technical Education.* Accessed at http://www.nccte.org/publications/infosynthesis/ in-brief/inbrief02-webtraining.pdf on August, 8, 2002

Appendix A Example of template description file (XML) for Prototype B

Paper1.xml

```
<?xml version='1.0'?>
<template>
      <block type='Heading'>
            <item name='title'/>
            <item name='instructor'/>
            <item name='contact'/>
      </block>

      <block type='Introduction'>
            <item name='paragraph'/>
      </block>

      <block type='Definition'>
            <item name='description'/>
            <item name='paragraph'/>
      </block>

      <block type='Summary'>
            <item name='paragraph'/>
      </block>

      <block type='Figure'>
            <item name='img'/>
            <item name='description'/>
      </block>

      <block type='Assignment'>
            <item name='title'/>
            <item name='paragraph'/>
            <item name='due_date'/>
            <item name='full_mark'/>
      </block>

      <block type='Syllabus'>
            <item name='title'/>
            <item name='week_no'/>
            <item name='topic'/>
            <item name='description'/>
            <item name='reading'/>
      </block>

</template>
```

Appendix B Example of template description file (XSLT) for Prototype B

Template.xsl

```
<?xml version= "1.0"?>

<!—This template description file could be used to transform any
paper?.xml into a template -->

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/"> <!—Match the root tag-- >

   <HTML><LINK href="styles.css" rel="stylesheet" type="text/css"/>
   <FORM name="frmTempalte">

<xsl:apply-templates/> <!— Start to apply the template -->

    <P><INPUT type="button" value="Create/Save" name="Create"
onclick="cmdCreateFileClicked()"></INPUT>

       <INPUT type="reset" value="Reset" name="reset"></INPUT></P>
   </FORM>
   </HTML>
</xsl:template>

<xsl:template match="block"> <!—search for the tag "block" -->

   <DIV>

<xsl:attribute name="id"> <!—find the attribute and take the value -->

<xsl:value-of select="@type"/>

</xsl:attribute>

<xsl:value-of select="@type"/></DIV>

   <TABLE class="border">
   <TBODY>

<xsl:for-each select="item"> <!—access each occurrence of a specified
node "item"-->

    <TR>
    <TD>

<xsl:value-of select="position()"/>

    </TD>
    <TD>
<xsl:value-of select="@name"/>

    </TD>
    <TD>
```

```
    <INPUT type="text" size="60">
<xsl:attribute name="value">
<xsl:value-of select="."/>
</xsl:attribute>
<xsl:attribute name="name">
<xsl:value-of select="@name"/>
</xsl:attribute>
      </INPUT>
      </TD>
      </TR>
</xsl:for-each>
      </TBODY>
      </TABLE>
</xsl:template>
</xsl:stylesheet>
```

Appendix C Template engine (transformer servlet) for Prototype B

XML and XSLT transformation Servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

//*****************
//The function of this servelt is to transform template description
//files into template (HTML)
//presented by Yanmin Shi
//*****************

public class SampleXSLTServlet extends javax.servlet.http.HttpServlet
{

      // Respond to HTTP GET requests from browsers.
  public void doGet (javax.servlet.http.HttpServletRequest request,
                     javax.servlet.http.HttpServletResponse response)
    throws javax.servlet.ServletException, java.io.IOException
  {
      // Set content type for HTML.
    response.setContentType("text/html; charset=UTF-8");
      // Output goes to the response PrintWriter.
    java.io.PrintWriter out = response.getWriter();
    out.write("<H1>Course Template</H1>\n");

      // pick up the parameters
    String paraValue1 =
"http://post:8080/templates/"+request.getParameter("xmlfile");
```

```
    String paraValue2 =
"http://post:8080/templates/"+request.getParameter("xslfile");

 try{
      javax.xml.transform.TransformerFactory tFactory =
                javax.xml.transform.TransformerFactory.newInstance();
      // Get the XML input document and the stylesheet, both in the
      //servlet
      // engine document directory.

      javax.xml.transform.Source xmlSource =
                new javax.xml.transform.stream.StreamSource
                (new java.net.URL(paraValue1).openStream());
      javax.xml.transform.Source xslSource =
                new javax.xml.transform.stream.StreamSource
                (new java.net.URL(paraValue2).openStream());
      // Generate the transformer.
      javax.xml.transform.Transformer transformer =
                            tFactory.newTransformer(xslSource);
      // Perform the transformation, sending the output to the
      //response.

      transformer.transform(xmlSource,
                 new javax.xml.transform.stream.StreamResult(out));
      }
      // If an Exception occurs, return the error to the client.
    catch (Exception e){
      out.write(e.getMessage()); //display error message
      e.printStackTrace(out);
                      }

      // Close the PrintWriter.
```

```
        out.close();
    }
}
```

Appendix D Example of XML processing at client application for Prototype B

(Template.html)

```
<html>

<body background="rain.jpg">
<h2>Template list</h2>

<table  border="0" cellpadding="0" cellspacing="0" bgcolor="#CCCCAA">
<tr><td>Name      Usage</td></tr>
<tr>
<td>
<form id="fm0"
action="http://sahajav.massey.ac.nz/templates/servlet/SampleXSLTServle
t" method="get" target ="Document" onSubmit="return add0(this)">
      <input type="hidden" name="xmlfile" value="paper.xml">
      <input id="xsl0" type="text" name="xslfile" size="12" value=" ">
      <input id="usg0" type="text" name="usg" size="3" value=" ">
      <input type="submit" name="Submit" value="Submit">
</form>
</td></tr>
<tr>
<td>
<form id="fm1"
action="http://sahajav.massey.ac.nz/templates/servlet/SampleXSLTServle
t" method="get" target ="Document" onSubmit="return add1(this)">
      <input type="hidden" name="xmlfile" value="paper.xml">
      <input id="xsl1" type="text" name="xslfile" size="12" value=" ">
      <input id="usg1" type="text" name="usg" size="3" value=" ">
      <input type="submit" name="Submit" value="Submit">
</form>
</td></tr>
<tr>
<td>
<form id="fm2"
action="http://sahajav.massey.ac.nz/templates/servlet/SampleXSLTServle
t" method="get" target ="Document" onSubmit="return add2(this)">
      <input type="hidden" name="xmlfile" value="paper.xml">
      <input id="xsl2" type="text" name="xslfile" size="12" value=" ">
      <input id="usg2" type="text" name="usg" size="3" value=" ">
      <input type="submit" name="Submit" value="Submit">
</form>
</td></tr>
<tr>
<td>
<form id="fm3"
action="http://sahajav.massey.ac.nz/templates/servlet/SampleXSLTServle
t" method="get" target ="Document"onSubmit="return add3(this)">
      <input type="hidden" name="xmlfile" value="paper.xml">
      <input id="xsl3" type="text" name="xslfile" size="12" value=" ">
      <input id="usg3" type="text" name="usg" size="3" value=" ">
      <input type="submit" name="Submit" value="Submit">
</form>
</td></tr>
```

```
</table>

<textarea id="textarea" rows="10" cols="10" style="visibility:hidden">
<USAGE>
<TEMPLATE><NAME>template.xsl</NAME><COUNT>5</COUNT></TEMPLATE>
<TEMPLATE><NAME>template1.xsl</NAME><COUNT>3</COUNT></TEMPLATE>
<TEMPLATE><NAME>template2.xsl</NAME><COUNT>1</COUNT></TEMPLATE>
<TEMPLATE><NAME>template3.xsl</NAME><COUNT>2</COUNT></TEMPLATE>
</USAGE>
</textarea>

</body>
</html>

<script type="text/javascript" for="window" event="onload">

      //read the xml file from textearea
var text=textarea.innerText;

      //load the xml into xmldom
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
      xmlDoc.async="false";
      xmlDoc.loadXML(text);

      // ..... processing the document goes here

var nodes=xmlDoc.documentElement.childNodes;


      // put template index orderly

sorting(nodes);

</script>

<script language="javascript">

function sorting(nodes){

      // sort the templates depending on the usage

xmlxsl=new Array(4);
xmlusg=new Array(4);

      for (i=0; i<=3; i++) {

      //pass the template name and usage to dom
            xmlxsl[i]= nodes.item(i).firstChild.text;
            xmlusg[i]= nodes.item(i).lastChild.text;
            }

      // define array to store the data from xml

compxsl=new Array(4);
compusg=new Array(4);

      //.........put into order
```

```
for (i=0; i<=3; i++) {

//define the object to keep the names of template and usage for
//calculation

      compxsl[i]= xmlxsl[i];
      compusg[i]= Number(xmlusg[i]);
                  }

tempusg=new Number(0);
var tempxsl;
tempxsl=" ";

//calculate the biggest
for (i=0; i<3; i++) {

      for (j=0; j<(3-i); j++) {


            if (compusg[j]<compusg[j+1]) {
                  tempusg=compusg[j];
                  tempxsl=compxsl[j];

                  compusg[j]=compusg[j+1];
                  compxsl[j]=compxsl[j+1];

                  compusg[j+1]=tempusg;
                  compxsl[j+1]=tempxsl;
                                          }
                              }
                  }

for (i=0; i<=3; i++) {

//pass the ordered data back to dom
      xmlxsl[i]= compxsl[i];
      xmlusg[i]= compusg[i];
                        }

// ..... set the values for inputs

//assign objects

      htmlxsl0= document.getElementById("xsl0");
      htmlusg0= document.getElementById("usg0");

      htmlxsl1= document.getElementById("xsl1");
      htmlusg1= document.getElementById("usg1");

      htmlxsl2= document.getElementById("xsl2");
      htmlusg2= document.getElementById("usg2");

      htmlxsl3= document.getElementById("xsl3");
      htmlusg3= document.getElementById("usg3");

//assign values
```

```
            htmlxsl0.setAttribute("value", xmlxsl[0]);
            htmlusg0.setAttribute("value", xmlusg[0]);

            htmlxsl1.setAttribute("value", xmlxsl[1]);
            htmlusg1.setAttribute("value", xmlusg[1]);

            htmlxsl2.setAttribute("value", xmlxsl[2]);
            htmlusg2.setAttribute("value", xmlusg[2]);

            htmlxsl3.setAttribute("value", xmlxsl[3]);
            htmlusg3.setAttribute("value", xmlusg[3]);
                        }

</script>

<script language="javascript">

function add0(){        //add 1 to the first template

      htmlxsl0= document.getElementById("xsl0");
      htmlusg0= document.getElementById("usg0");

      htmlxsl1= document.getElementById("xsl1").value;
      htmlusg1= document.getElementById("usg1").value;

      htmlxsl2= document.getElementById("xsl2").value;
      htmlusg2= document.getElementById("usg2").value;

      htmlxsl3= document.getElementById("xsl3").value;
      htmlusg3= document.getElementById("usg3").value;

      newusg=Number(htmlusg0.value)+1;
      newxsl= htmlxsl0.value;



      text="<USAGE><TEMPLATE><NAME>";
      text= text+newxsl;
      text= text +"</NAME><COUNT>"+newusg+"</COUNT></TEMPLATE>";
      text= text
+"<TEMPLATE><NAME>"+htmlxsl1+"</NAME><COUNT>"+htmlusg1+"</COUNT></TEMP
LATE>";
      text= text
+"<TEMPLATE><NAME>"+htmlxsl2+"</NAME><COUNT>"+htmlusg2+"</COUNT></TEMP
LATE>";
      text= text +
"<TEMPLATE><NAME>"+htmlxsl3+"</NAME><COUNT>"+htmlusg3+"</COUNT></TEMPL
ATE></USAGE>";

      //read the xml file from textearea

alert(text);

xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
```

```
      //load the xml into xmldom

xmlDoc.async="false";
xmlDoc.loadXML(text);


      // ..... processing the document goes here

var nodes=xmlDoc.documentElement.childNodes;


sorting(nodes);
      //.........put into order

return true;

            }


function add1(){  //add 1 to the second template

      htmlxsl0= document.getElementById("xsl0").value;
      htmlusg0= document.getElementById("usg0").value;

      htmlxsl1= document.getElementById("xsl1");
      htmlusg1= document.getElementById("usg1");

      htmlxsl2= document.getElementById("xsl2").value;
      htmlusg2= document.getElementById("usg2").value;

      htmlxsl3= document.getElementById("xsl3").value;
      htmlusg3= document.getElementById("usg3").value;

      newusg=Number(htmlusg1.value)+1;
      newxsl= htmlxsl1.value;



      text="<USAGE><TEMPLATE><NAME>";
      text= text+htmlxsl0;
      text= text +"</NAME><COUNT>"+htmlusg0+"</COUNT></TEMPLATE>";
      text= text
+"<TEMPLATE><NAME>"+newxsl+"</NAME><COUNT>"+newusg+"</COUNT></TEMPLATE
>";
      text= text
+"<TEMPLATE><NAME>"+htmlxsl2+"</NAME><COUNT>"+htmlusg2+"</COUNT></TEMP
LATE>";
      text= text +
"<TEMPLATE><NAME>"+htmlxsl3+"</NAME><COUNT>"+htmlusg3+"</COUNT></TEMPL
ATE></USAGE>";

      //read the xml file from textearea
alert(text);

xmlDoc = new ActiveXObject("Microsoft.XMLDOM");

      //load the xml into xmldom
```

```
xmlDoc.async="false";
xmlDoc.loadXML(text);


      // ..... processing the document goes here

var nodes=xmlDoc.documentElement.childNodes;


sorting(nodes);

      //.........put into order

      return true;
            }


function add2(){  //add 1 to the third template

      htmlxsl0= document.getElementById("xsl0").value;
      htmlusg0= document.getElementById("usg0").value;

      htmlxsl1= document.getElementById("xsl1").value;
      htmlusg1= document.getElementById("usg1").value;

      htmlxsl2= document.getElementById("xsl2");
      htmlusg2= document.getElementById("usg2");

      htmlxsl3= document.getElementById("xsl3").value;
      htmlusg3= document.getElementById("usg3").value;

      newusg=Number(htmlusg2.value)+1;
      newxsl= htmlxsl2.value;



      text="<USAGE><TEMPLATE><NAME>"+htmlxsl0+"</NAME><COUNT>"+htmlusg
0+"</COUNT></TEMPLATE>";
      text= text
+"<TEMPLATE><NAME>"+htmlxsl1+"</NAME><COUNT>"+htmlusg1+"</COUNT></TEMP
LATE>";
      text= text
+"<TEMPLATE><NAME>"+newxsl+"</NAME><COUNT>"+newusg+"</COUNT></TEMPLATE
>";
      text= text +
"<TEMPLATE><NAME>"+htmlxsl3+"</NAME><COUNT>"+htmlusg3+"</COUNT></TEMPL
ATE></USAGE>";

      //read the xml file from textearea
alert(text);

xmlDoc = new ActiveXObject("Microsoft.XMLDOM");

      //load the xml into xmldom
xmlDoc.async="false";
xmlDoc.loadXML(text);
```

```
      // ..... processing the document goes here

var nodes=xmlDoc.documentElement.childNodes;

sorting(nodes);

      //.........put into order


      return true;
            }

function add3(){  //add 1 to the fouth template

      htmlxsl0= document.getElementById("xsl0").value;
      htmlusg0= document.getElementById("usg0").value;

      htmlxsl1= document.getElementById("xsl1").value;
      htmlusg1= document.getElementById("usg1").value;

      htmlxsl2= document.getElementById("xsl2").value;
      htmlusg2= document.getElementById("usg2").value;

      htmlxsl3= document.getElementById("xsl3");
      htmlusg3= document.getElementById("usg3");

      newusg=Number(htmlusg3.value)+1;
      newxsl= htmlxsl3.value;



      text="<USAGE><TEMPLATE><NAME>"+htmlxsl0+"</NAME><COUNT>"+htmlusg
0+"</COUNT></TEMPLATE>";
      text= text
+"<TEMPLATE><NAME>"+htmlxsl1+"</NAME><COUNT>"+htmlusg1+"</COUNT></TEMP
LATE>";
      text= text
+"<TEMPLATE><NAME>"+htmlxsl2+"</NAME><COUNT>"+htmlusg2+"</COUNT></TEMP
LATE>";
      text= text +
"<TEMPLATE><NAME>"+newxsl+"</NAME><COUNT>"+newusg+"</COUNT></TEMPLATE>
</USAGE>";

      //read the xml file from textearea
alert(text);

xmlDoc = new ActiveXObject("Microsoft.XMLDOM");

      //load the xml into xmldom
xmlDoc.async="false";
xmlDoc.loadXML(text);

      // ..... processing the document goes here

var nodes=xmlDoc.documentElement.childNodes;

sorting(nodes);
```

```
        //.........put into order
        return true;
              }
</script>
```

Appendix E The instructions of evaluation on Prototype A

The procedures to log in the Prototype A are listed here:

1. Type http://it016600:8080/teacher_model/logon.htm into the address field of Internet Explore, evaluator will see the welcome page.
3. Key in **157736** as the default user, and then press Enter.
4. Click on the Chapter **Template Server Prototype**.

Followings are the tasks to help the evaluators get familiar with the prototype. The evaluator is expected to choose any two templates from them.

Scenario A

Now, evaluators are supposed to create the online content using the prototype. The section is called "Data Modeling".

1. In the chapter **Template Server Prototype**, click on *Modify* dropdown list to choose "*Add section*".
2. Click on the link *Create New* at the bottom.
3. Choose the template: **Bulleted list form,** press the button *Continue.*
4. Give the name **Data Modeling** to the new unit
5. Add the following content to this section.

    - *to understand why data modeling is essential to database design*
    - *to understand the basic concepts relevant to all data,*
    - *to recognise differences between a conceptual E-R model and a logical relational model*

6. Press *OK* to confirm the inputs in the message box.
7. Select the new section *Data Modelling* from the dropdown list.
8. Select the prerequisite if you need to.
9. Add the summary.

Data modelling is a technique which is widely used in database development.

10. Press the button *Submit*.

Scenario B

Now, evaluators are supposed to create the online content using the prototype. The section is called "Objects".

1. In the chapter **Template Server Prototype**, click on *Modify* dropdown list to choose "*Add section*".
2. Click on the link *Create New* at the bottom of the page.
3. Choose the template: **Paragraph with title form,** press the button *Continue.*
4. Give the name **Objects** to the new unit.
5. Add the following content to this section.

   Title: *Object Paradigm*

   Content: *Objects (Instances) are grouped into Classes based on common attributes and behaviors in the problem domain. Each Object has its own state but operations are shared. Each Object has a unique identity independent of its state.*
6. Press *OK* to confirm the inputs in the message box.
7. Select the new section *Objects* from the dropdown list.
8. Select the prerequisite if you need to.
9. Add to the Summary.

   Objects are widely used in software engineering.
10. Press the button *Submit*.

Scenario C

Now, evaluators are supposed to create the online content using the prototype. The section is called "ER Diagram".

1. In the chapter **Template Server Prototype**, click on *Modify* dropdown list to choose "*Add section*".
2. Click on the link *Create New* at the bottom of the page.
3. Choose the template: **Text with picture,** press the button *Continue.*
4. Give the name **ER Diagram** to the new section
5. Browse the server to find the file called **Scenario_C_text.doc** which contains the following content.

   The *entity types* are shown in rectangular boxes. *Relationship types* are shown diamond-shaped boxes attached to the participating entity types with straight lines. *Component attributes* of a composite attribute are attached to the oval representing the composite attribute. *Multi-valued attributes* are shown in double ovals. *Key attributes* have their names underlined. *Derived attributes* are shown in dotted ovals.

6. Browse the server to find the file called **erd0.gif**
7. Browse the server to find the file called **erd1.gif**
8. Press *OK* to confirm the inputs in the message box.
9. Select the new section *ER Diagram* from the dropdown list.
10. Select the prerequisite if you need to.
11. Add to the Summary.

    ER Diagrams are widely used in modelling the data requirement.

12. Press the button *Submit.*

Appendix F The URL of the Evaluation Form

Available At http://it016600:8080/teacher_model/Survey.htm