An Authoring Tool for Structuring and Annotating On-line Educational Courses

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

Master of Science
in
Computer Science

at Massey University, Palmerston North,
New Zealand.

Yang Wang
2002

Abstract

This thesis studies the design and prototype implementation of a new web-based course authoring system for the Technology Integrated Learning Environment (TILE) project. The TILE authoring system edits the course structure and allows the author to annotate the course structure with meta-data. It makes extensive use of XML technology to communicate structured data across the Internet, as well as for both local and web-side databases. The Authoring tool is designed to support development by multiple authors and has check-in and check - out, as well as version control facilities. It also provides an interface for adopting other multimedia tools such as AudioGraph. The tool has an easy-to-use graphical user interface.

The technical problems that have been solved in this project include issues such as cross-platform support, drag and drop functionality using JDK1.1.8, etc. System environments, such as relational database set up, XML database set up, Java swing set up in Mac also have been discussed.

The authoring system interface analysis, database analysis and function analysis have been completed for the complete the system as specified. An intermediate system, designed to a reduced specification, has been implemented as a prototype and details of this system, which can work independently of the TILE delivery system, are included. The Full TILE authoring system including InstantDB database access also has been partially implemented. The prototype application has also has been tested on the PC platform.

Acknowledgement

Over the last two years, I have experienced so much encouragement and support from the people, and work. I have never been so grateful about any job I have done so far. It is hard work and it means a lot to me.

I have to thank my supervisor, Professor Chris Jesshope. Thank you for your inspiration, encouragement, and support, without your help, my work wouldn't be that easy and enjoyable. The confidence and knowledge I have gained are priceless.

I also want to thank all my colleagues in TILE project, especially Jenny Zhang. Jenny has done a wonderful job and has given me a great deal of support for my research. Thanks also go to the whole group; they are great people and make a great team.

Yang Wang
Master of Science (Computer Science) Candidate,

Massey University

Computer Science,

Institute of Information Science and Technology

Massey University

Palmerston North

New Zealand

Table Of Content

List Of Figure

List Of Table

Chapter 1. Introduction to Web Based Distance Education

1.1. An overview of Distance Education

1.1.1 What is Distance Education?

Distance Education is instructional delivery that does not constrain the student to be physically present in the same location as the instructor. Historically, Distance Education meant correspondence study. Today, audio, video, and computer technologies are the more common delivery modes [10]. At its basic level, distance education implies that a certain distance exists between the teachers and students, and technology is used to bridge the instructional gap.

Today, we don't need to go to school and sit in the classroom to gain knowledge, because there are many alternative ways that provide us with choice in the best and most suitable way to gain our knowledge. Distance Education is not new for us. It gives those people who do not have time to go to school and sit in class a great chance to update their knowledge, skills, and to refresh information about their employment or even their leisure activities. People think that a comparison between Distance Education and traditional education is just like two ways that have a different path, but the same destination. Actually we know today that distance education has more flexibility and freedom. It is maybe more suitable for today's education environment, because it brings out a large potential education market; it provides more opportunities to people [15].

Specifically, distance education involves a complex and hierarchical system of interrelated sub-systems. Each part has its own internal complexities, but in general each affect the other parts and are affected by the other parts [14]:

- Hardware and software technologies are the base of this hierarchy. Other sub-systems include
- Means of telecommunications, which put the student and the teacher in contact with each other,
- The instructional and learning subsystems, which are usually defined in academic programs and courses,
- The management system, which keeps the entire enterprise together,
- The social system which provides funding, and regulates the operation of the entire enterprise and,
- The international systems, such as the World Wide Web, which allows people in different countries engage in teaching and learning at a global level.

To design a successful distance education system, we have to begin with a careful plan and to fully understand the course requirements and students' needs. We must also know the key players in the distance education enterprise.

These players include [26]:

- Students - The primary role of the student is to learn.
- Faculty - The success of any distance education effort rests squarely on the shoulders of the faculty.
- Facilitators - The instructor often finds it beneficial to rely on a site facilitator to act as a bridge between the students and the instructor.
- Support Staff - Support personnel are truly the glue that keeps the distance education effort together and on track.
- Administrators - They maintain an academic focus, realizing that meeting the instructional needs of distant students is their ultimate responsibility.

In fact, a successful distance education system really relies on the consistent and integrated efforts of students, faculty, facilitators, support staff, and administrators [26]. In following section we discuss why today's education system needs distance education.

1.1.2 Why Distance Education?

Why do we need distance education today? What's the different between traditional education and distance education? Is it really just people who have the same destination going with a different way?

No wonder the technology is changing so fast today, with our rapidly changing technological base, gaining knowledge has become a task of *lifelong learning*. The learning environment and learning technology has developed according to the society's requirements. People required a new way to gain knowledge without going to school and that could be done anytime and anywhere. These requirements give a major challenge to traditional education.

First, we cannot deny there are certain fundamental problems that have been solved by distance education. These kinds of programs can provide adults with a second chance at a college education, reach those people with limited time, distance or physical disability, and update the knowledge base of workers at their places of employment [26]. The problems that have been solved by distance education we can find out in the Diana Oblinger's paper [15], they are:

- Expanding Access

Distance education improves access to education, reducing the barriers related to geography, economics, time constraints, and physical or leaning disabilities. Also as we all know, the Internet is a very popular medium to achieve this.

- Alleviating capacity constraints.

We would say distance education is more focused on "student centered" learning; the students select learning space, time, and location. They also can revisit the learning materials as they desired, initiate the communication. Also distance education is providing adequate resource and information for students.

- Capitalizing on emerging market opportunities.

As gaining knowledge is becoming lifelong learning, the demand of higher education among the people who is out of the traditional educational age range is increasing, like working adults and students who might seek further education. This shows us a potential market, maybe more lucrative than traditional markets.

- Serving as a catalyst for institutional transformation.

"Higher education institutions are being challenged to adapt rapidly to an increasingly competitive environment. Distance education can catalyse institutional transformation." [15]

1.2. Web -Based Learning

1.2.1 Information Superhighway and World Wide Web

"The Internet is perhaps the most transformative technology in history, reshaping business, media, entertainment, and society in astonishing ways. But for all its power, it is just now being tapped to transform education" [12]. The Internet [18] brings us a revolution of Education. It gives the possibility of learning of all kinds, all levels, for men, women and children. It is a fresh way of teaching and learning; it connects the people, communities and resources; it extends the learning day and learning spaces. The cutting-edge technology, such as the World Wide Web and online conferencing systems, enable universities to provide a open learning environment for students 24 hours a day and 7 days a week. It is quickly becoming the one facet most commonly used for delivery of principal course content.

The World Wide Web is one of the fastest growing information resources. The Web provides a graphical friendly user interface and enables the display of rich graphical images, pictures, full motion video, and sound clips [18]. The educator can use the Web to build an educational home page, which can cover information about the virtues of a class including the syllabus, exercises, literature references, and instructor's biography. "The instructor can also provide links to information on the WWW that would be useful to students in the class (e.g., research data on agricultural markets, global climate change, or space missions)"[27]. "Use of the Web for delivery of distance learning is finding an audience in the current *just-in-time* education environment, where customized programs and convenient professional development opportunities are valued by today's lifelong learners"[20]. Also the students in traditional facilities -base courses are seeking the convenience to access their resources, information and communication via the Internet.

There are many studies about web-based learning point a common benefit that web based learning involved more active participation by students. Students take responsibility for their studying, and great equity of participation. To design a quality web-based education system is a costly, formidable task. "It requires division of labour, integration of different technologies, professional managements and political governance" [34].

As we all know, the Internet covers all the things that any person might need in their ordinary day, because the type and extent of information found on Internet sites is so diverse. The Internet also means that anyone can become a publisher of information, even

if it is only publishing their family photograph album. Whatever that is published, then available to the whole Internet community. They are seven basic types of Internet sites listed in [28], these are:

- Personal
- Commercial
- Archive/References
- Current/News
- Informational
- Persuasive / Propaganda
- Educational

Given proper selection, this provides a valuable resource for educational purposes.

Compared with traditional education, on-line learning presents some similar points with face-to face education. As a computer is involved, the learning environment has become socialized. Not only do students learn independently, but they also learn interactively and collaboratively with peer groups. Harasim [13] stated five important characteristics for online education:

- Many to many communication
- Place independence
- Time independence (that is time-flexible not atemporal)
- Text-based communication, and
- Computer mediated interaction.

These five points clearly illustrate that today's web-based education environment takes the significant advantages from the traditional education, and use these features to construct a new education environment. But as the technology grow rapidly, especially for online education, there is another important online education feature has been widely used today. It is multimedia communication. It combines video, audio, image, and text together, delivery courses in a rich communicational way. Multimedia communication makes the way of delivering course is much close to the conventional face-to-face class.

1.2.2 The challenges of Web-based learning and potential benefits

"Although using the web to deliver instruction provides many benefits to instructors and students, substantial challenges persist which must be overcome before high-quality learning experiences can be offered" [17]. The tool constructors have to understand how the new technology affects both the educator and the learner, how these new technologies can be used to their maximum advantage and ways in which to compensate for their limitation.

The challenges for the educator are [17]:

- High cost of delivery system technology

- Bandwidth and the limitations of end user technology
- Dealing with authoring for delivery in HTML
- Moving to a learner-centered instructional model (see Figure 1.1)
- The need to learn and utilize new skills
- Misconceptions amongst educators
- Challenge of developing Hypermedia
- Challenge of making instructional content interactive

Nowadays people are working on these challenges, some problems have been solved, and some are still addressing. Like dealing with Bandwidth and the limitations of end user technology and dealing with authoring for delivery in HTML that listed in the challenges, the people developed AudioGraph [75-79] has tried to overcome these problems. They are offering low bandwidth delivery of multimedia and also by simplifying the authoring model to make authoring multimedia HTML pages accessible to the non-expert. More recently they have tried to tackle the last point [80] that is making instructional content interactive. I am sure there are a lot of people work on the problems in the distance education field today, to make the technology is more useful and stable.

The challenges for the learner are [17]:

- Student Access to requisite technology
- Loss of face-to-face interaction with the instructor
- Psychological pitfalls – In the conventional classroom, students and lectures have rich communication, which transmitted by facial expressions, posture, gaze and gestures, voice volume, inflection and tone. Online distance education is mainly delivered by text and images, merely transfer this kind of information.

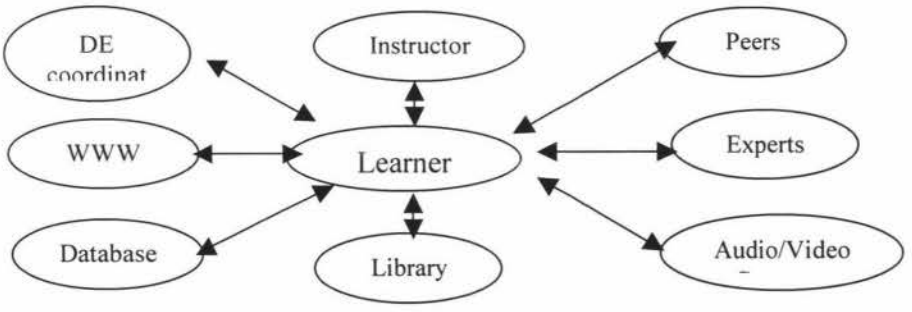

Figure 1.1 Learner – centred instructional model [19]

Studying these challenges can help developers to understand the base requirements for web-based learning delivery and authoring systems, and increasing their understanding of the principles and practices that support universal design.

It has also been found that there are a number of potential benefits that web-based learning could bring to us with its wide spread adoption. These benefits include [20]:

- Flexibility in the pace of learning,
- Material can be presented in redundant, reinforcing or alternative formats e.g., speech, print, graphics, etc.,
- Student can clarify, rehearse and review supporting materials without interrupting the flow of the learning session for classmates,
- Issues of distance, transportation and physical accessibility are reduced,
- Equal participation for students who use augmentative or alternative communication methods, as the method and rate of communication is transparent to fellow classmates,
- Greater opportunities for peer interaction and collaboration, and for student-instructor interaction and assistance,
- Material can be adapted to various learning styles [73, 74]

Other potential advantages include encouraging the development of technical skills and sophistication in learning, for example: learning to search, evaluate, and synthesize information; and learning basic computer skills, which are a prerequisite for on-line learning. Of course these benefits cannot conceal certain pitfalls, e.g. the redundant, reinforcing or alternative formats of material would raise the cost of development. According to Noriko Hara's work [35], a web-based distance education course would frustrate students as they might feel being isolation in a virtual classroom that would not be present in a face-to-face environment.

1.3. A Comparison of current software tools

There are large numbers of Web Authoring tools that have been developed since Web Based Distance Education has become a topic of great interest. The varieties of web-based learning are usually defined by the technology used, or by the approach to learning that the technology supports [21]:

- Self-Instructional and Instructor led/Collaborative
- Synchronous Learning and Asynchronous Learning
- Built in Authoring, Third Party Authoring and Content Assembly
- Rich media and Lean Media
- Low interaction and High Interaction
- Course Delivery, Course Management (CMI) and Learning Content Management
- Learning and Performance Support

We see the current generation of general WWW tools and servers was designed for browsing and information retrieval, and not as components of an active learning system. Therefore, they currently lack a number of features that an advanced educational environment requires [22]. Through the study of various existing software tools for

educational delivery, which are now appearing on the market, we can try to find what else we need to add to this newborn clutch of applications if we decide to develop a better web-based education system. Examples we have studied include LearningSpace, TopClass, Web Course in a Box, and WebCT, etc. We also have to know what general features the software should provide to a user and indeed who those users are, because we must satisfy a range of different users. These include the administrators, the teachers and the students. Good web-based education software should possess a number of the following features. They includes [23]:

- Ease of use by faculty
- Ease of use by student (intuitive interface)
- Include various media (text, graphics, video, audio)
- Support alternate character sets (mathematics, foreign languages)
- Various communication models (one to many; one to one; many to many)
- Threaded discussions
- Full text search
- HTML links within courseware
- Application links within courseware
- Student tracking
- Student registration
- Quizzes and online testing
- Automatic student reporting
- Tracking of time/hits/etc. per student
- Free client programs for students
- Ability to access remotely (faculty and students)
- Cross-platform delivery
- Ease of updates/revisions
- Security and password access
- Real-time communication (chat, videoconferencing)
- Online help and phone help (800 line)
- Time limitations feature (set display for 2 weeks, etc).

This is a daunting list of features and many of these have to taken in account when designing a web based education system. The TILE [1] project, on the other hand, which this project contributes towards, has considered different aspects of the problems in on-line education. TILE project is developing an integrated system for managing, authoring, and publishing on-line education. It looks at the different responsibilities, and provides architecture and procedures to solve some of the outstanding issues not listed above. TILE has considered four main issues, these are:

- Flexibility of use for students and staff
- The reuse of educational material in different courses
- The problems of the students' use of bandwidth
- Finally the system scalability

The TILE system includes basic two components: a course delivery system and an authoring system. This project has been investigating the latter, the authoring system. It should be emphasised that the authoring system is not for basic content production, as it is assumed that the learning objects are produced by other tools, such as HTML editors or multimedia authoring tools. This system is for authoring the content's structure, the prerequisites, the meta data and for ensuring that the system integrates with other commercial tools for content production.

Before we talk in detail about the TILE course delivery system, we need to know something about existing delivery systems. There are various types of software for course delivery, basically there are three types of system that have evolved into courseware environments, these are [30]:

Groupware

Groupware, also know as "computer-supported cooperative work (CSCW) can link people on different computers using the same software program (such as Lotus Notes) to perform a variety of functions" [30]. Usually groupware supports the following functions:

1. Face-to-face meeting facilitation
2. Group decision support
3. Computer-based telephony extensions
4. Presentation support
5. Project management
6. Calendar management
7. Group-authoring
8. Computer-supported face-to-face meetings
9. Screen sharing
10. Computer conferencing
11. Text filtering
12. Computer-supported audio/video teleconferencing
13. Group memory management
14. Spontaneous interaction
15. Comprehensive workgroup support
16. Nonhuman meeting participates (using intelligent agents)

Listserves

A listserve is a system that allows a group of people to discuss issues in a common environment, usually by email. The listserve software will provide functions to organise and browse a threaded discussion on a given topic. "The listserve can be supplemented by electronic mail, the World Wide Web, and the telephone --- as well as ... audio, multimedia, 3D models, form-based surveys, videoconferencing, etc" [30].

Multi-user environments

"A Multi-User Environment is a real-time, text-based communication; it's similar to Internet Relay Chat, except that it takes place in an imaginative context described via text and participants are usually playing some sort of role" [30].

We can see that these different classes of software have points in common, namely that they all provide interaction in an on-line mode. Student accesses the study material through the Internet or an intranet. Although the Internet or intranets are very popular today, we have to consider the situation, where a student is outside the range of a network connection. What can they do without a connection in order to get the resources they require? That question was a leading one in the design of the TILE [1] project delivery system. This new generation of delivery system does provide an on-line mode but it also covers the shortcoming of a lack of a network connection. In TILE the course delivery system provides both on-line and off-line modes. Moreover, in the offline mode it is possible to guide and track what the student is doing as the server functionality is distributed to the student's computer. It therefore has more flexibility and convenience as students can be monitored even when they are off-line. We will give more details about TILE [1] course delivery system in Chapter 2.

It must be emphasised here that learning is a bi-directional process in reality; the student is placed in a role of learning, and the teacher in the role of the educator, actually some times these roles are reversed. What must the teacher to support this role and what are the requirements of the authoring tools? These tools must provide a number of significant services for teachers to create on-line and manage the courses [70]. Some of these services are:

- Creating content of some description
- Creating a structure for the course
- Providing prerequisites that guide the student through the material
- Searching for and incorporating existing learning objects
- Providing a means by which courseware can be imported and exported for use from or in other delivery systems [71, 72].

In addition to these requirements the tool must be easy to use and the courseware must be easy to maintain. This authoring system must combine together these useful functions and provide a multi-service for teachers.

It is unlikely that any authoring tool can provide everything that a teacher will require. "Web-based educational systems, like other computer-based education (CBE) systems, must provide certain basic instructional functionalities" [29]. Usually the authoring tools can be classified into the following categories, according to what material they create:

- Voice recorder
- Text editor
- Image editor
- Video recorder
- Special purpose editor (e.g. XML editor, java script editor, mathematical equation editor, etc)
- Stand alone or on-line
- Data management tool

Some of these tools are very familiar for us, like Microsoft word, PowerPoint, Excel, QuickTime, PhotoShop, PowerBuilder, etc. But there are still plenty more authoring tools that provide excellent functions for teachers, e.g. ClassMaster 3.0 [24], AudioGraph [2], Mathematics TestBuilder [24], Site Central [25], etc. These authoring tools focus on a different aspect of authoring function, the content production or the creation of learning objects. We will not give a detailed comparison of these tools, as the project is concerned only with the last category above, that of a data management tool. Such a system must allow content created by the above tools to be integrated into a course but must also provide a means of navigating it, possibly doing this adaptively, depending on the student's preferences and knowledge. Because this information is held in a database, the authoring tool is used to manage the relationships between learning material, just like a database management tool.

1.4. Work related to this project

A lot of research has been completed in this project that is related to the TILE project. In particular, this includes:

- TILE Authoring Interface analysis and design
- TILE Authoring Interface functionality
- Java cross-platform research
- Set up JDK 1.1.8 on Mac
- Studying the use of an XML database to describe the meta data and the possibility of using this document as a replacement for a relational database on the users computer. This avoids having to install database software.
- Analysis and design of the database schema for the TILE authoring system
- Set up relational database on the client side for the full TILE authoring system
- Design the system framework and Intermediate authoring system
- Complete a prototype implementation of the intermediate TILE authoring system

This research, design and implementation work will be discussed in the following chapters.

Chapter 2. An Overview TILE project in relation to this thesis

2.1. Background to the project

The goal of this Master's project is to design and prototype an authoring interface for an on-line learning system. The authoring tool would provide a means to interface to the database of the course development and delivery servers of the TILE system [1], see also figure 2.2, which is currently being developed in a NERF-funded project at Massey University. The material authored would then be published to the TILE delivery server and delivered to students by means of the TILE clients. The requirement was to create interface to the database on the server, create and edit the dynamic course structure and also interface to various tools, such as AudioGraph multimedia authoring tool [2] for generating the on-line content. The interface in the authoring interface system will invoke the different multimedia tools, using them to produce the course content, for example using AudioGraph. A user might use it to generate course content, and then save these files locally. The user will then have to locate the files for the authoring (possibly as a URL). It is not necessary to save the files in a fixed path. All relative information about course structure and the information about these newly created mediate files will store in the databases. The authoring interface system will generate the dynamic structure of the pages that display the content, which is held in a relational database.

The TILE Authoring interface is one component of TILE project. It synthesizes or integrates content created by the text, graphics, audio and multimedia editing tools into a single presentation (for example a complete on-line course) within the TILE system. The TILE Authoring interface provides functions that can help authors to structure and annotate course media for publishing on line. The whole idea of the TILE project is to provide a flexible system for developing and presenting information for distance learning. The two major components of the TILE project are the Course Delivery system and the Authoring Interface system, a prototype of which is described in this report.

2.2. Technology Integrated Learning Environments (TILE)

2.2.1 An overview of TILE

TILE stands for Technology Integrated Learning Environments. It is a project funded under the New Zealand Government's New Economy Research Fund (NERF) [1]. Tile project is currently developing an integrated system for the management, authoring, delivery and monitoring of on-line education. The aim of this project is to integrate content into a course from a range of other tools (e.g. Multimedia Authoring tools, such as AudioGraph [2]. Web editing tools, such as FrontPage, Netscape Composer; On-line testing tools, such as StyleQuiz Version 1.0 [62], Content Authoring Tool (C.A.T.) [63], etc.). The outcomes of the TILE project will be commercialised and delivered to the distance education market and will contribute to the New Zealand distance education market and also provide competition in the global distance education market.

Tile project has five objectives, which can be found on-line in [1], they are:

- Objective 1: Define server system
- Objective 2: Further Develop Multimedia Editing Clients
- Objective 3: Knowledge Representation and Free-Form Querying
- Objective 4: Adaptation in the Delivery of Self-Learning Modules
- Objective 5: Research Underpinning Phase 2 of the Project

To give some background to the project, we are going to give a little bit more detail about each objective to understand how the authoring interface tool fits in to the bigger picture:

Objective 1 provides the framework for integrating all of the other tools being developed within the project. Meta-schemas will be used to define all component interactions within the system. In this objective, the whole project's feasibility and the base-level software's suitability have been examined. The database has been specified for the learning content and solutions to potential problems have also been studied. Objective 1 divided the whole framework into two primary parts, one of which, is the course delivery system and the other is the authoring system. This project is concerned with the latter. We will give a more detailed description of them later this chapter.

Objective 2 will further develop the AudioGraph, low-bandwidth, multimedia authoring and playback tools. In this objective, AudioGraph will have more functions added and it will be integrated into the server framework, which is being developed in the objective 1. The interface for this integration will be provided by the authoring interface tool, being prototyped in this thesis. Currently the Audiograph has been fully developed as a Macintosh application and a Windows PC version also is now available via the web [http://www.nzedsoft.com].

Object 3 will develop annotation and query clients to allow the multimedia to be indexed by the author and searched by students, using a free-format, and restricted-natural-language interface [1]. To achieve this two languages have been developed, a Flexible System Coding Language (FSCL) and a query language FSQL. This work must eventually be integrated within the authoring interface client but this is not being considered in the prototype studied here.

Objective 4 uses the Hypertext Transfer Protocol (HTTP) to identify the means and level of adaptation in the delivery of the multimedia. Adaptation requires development of a student model, which helps the mental process, analyses the interaction between the students and system, and provides a better learning environment. Again there will be a need to integrate this work into the authoring system but again it is outside the scope of this master's project.

Objective 5 is doing the research for two further tools, which are being developed in this project. One is "a high-level authoring tool for curriculum planners to provide all the necessary administrative support for managing instructional material and activities" [1].

The other is "an authoring tool to support a generic problem-solving approach to learning" [1].

In this thesis therefore, we focus on the integration of content from other authoring tools into the schema, which defines the dynamic content of the web-based course delivery. It studies different aspects of multimedia authoring tools, and specifies the detailed techniques and issues involved in integrating them into the TILE system via a Java application.

2.2.2 An overview of the TILE Course Delivery System

The TILE Course Delivery System provides a number of flexible delivery modes. Users can access this system no matter whether they are on-line or off-line. The functions in this system are:

- Providing two basic modes of delivery – on-line and off-line
- Users can browse all course materials regardless of their location
- Users can query the course material by either keyword or natural language
- Users can take notes when they browse the course materials
- A synchronization mechanism will always provides users with up-to-date course materials, even if they use off-line browsing and will provide lecturers with up to date information on the students' progress and knowledge based on the student model
- Implementing an adaptive, self-learning student module in this system, which provides dynamic messaging and feedback.

Figure 2.1 shows an overview of the delivery system process

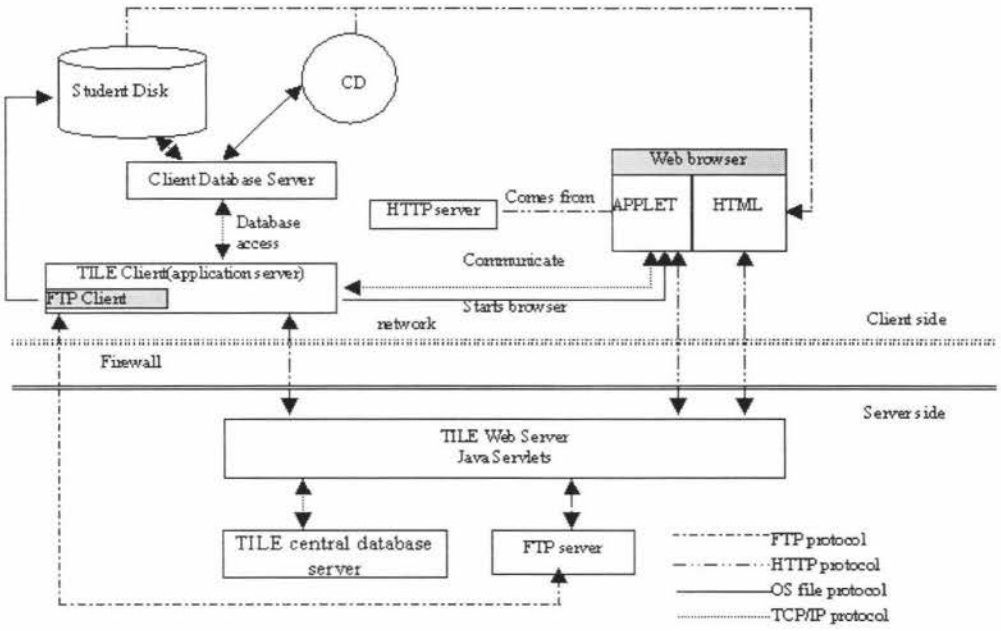

Figure 2.1 The Tile Delivery System Client –Server Architecture [11]

Figure 2.1 shows the delivery system architecture, which is part of the TILE system. It can be seen that as well as the normal centralised server on the educational provider's site, there is also a database and server as a part of the TILE client, which runs on the student's own computer. The main server is based on servlet technology and the distributed server is an application on the student's computer. The TILE does the job for providing course structures to students, regardless of the mode of operation or the connection that the student has. Also it does the job of storing course information when lecturers send their finished course to server.

2.2.3 Overview of the TILE Authoring Interface System

The TILE Authoring interface system is just one component of TILE project [1]. The aim of this system is to provide a flexible and convenient way for distance education content to be integrated into the TILE framework. Text, graphics, and multimedia editing tools have to be integrated into this system. The TILE Authoring system also provides basic functions that can help authors to structure and annotate course media for publishing on line, for example:

- Creating new course structure
- Editing exiting course structure
- Publishing course structure
- Browsing existing course structure
- Check-in and check-out facilities for multiple-author control
- Maintaining access rights and capabilities to various authors
- Searching the course structure

These basic functions form the complete TILE Authoring system. However for this project, these have been simplified and specified as the TILE Intermediate Authoring system. This has been done to decouple this work from other development proceeding concurrently by other team members on this project. It should be noted that it also provides a means of using the work without institutional support for the complete TILE system. For example a single lecturer would be able to author a course and deliver it using a standard web-server to a standard web browser. Of course this would not provide the flexibility of delivery or tracking that would be provided by the complete TILE system.

A high-level architecture of the TILE system complete with authoring interface is given in Figure 2.2. This shows the distribution of databases and authoring and delivery functions. Two specifications for different versions of the authoring systems will be described later in this chapter.

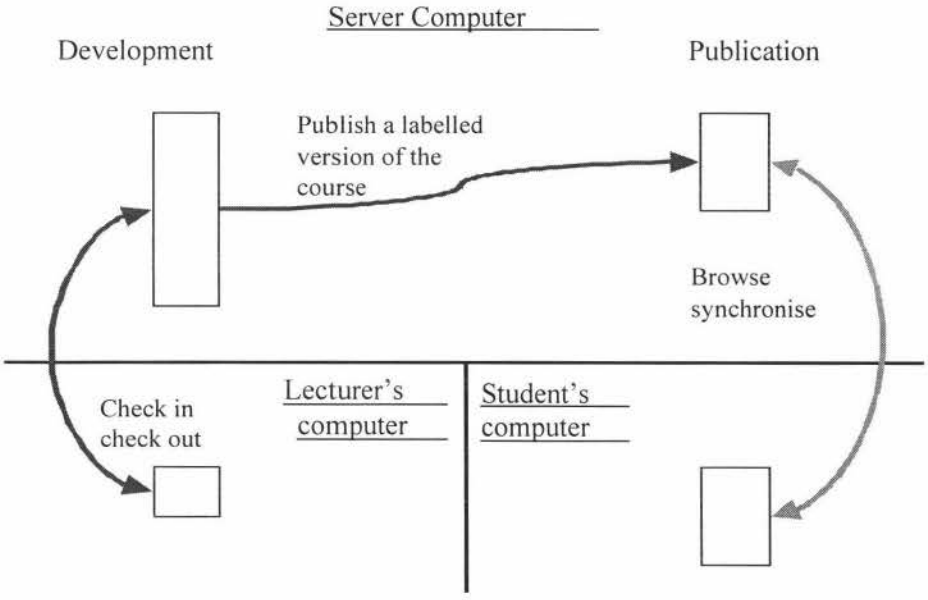

Figure 2.2This diagram shows the different instances of the database required for different stages of on-line education development and delivery, and the processes involved in updating these databases [35]. The Authoring Interface system manages the left hand side of this diagram.

2.2.4 TILE Authoring Interface System requirements

In the Authoring Interface system, our goal is to provide some powerful functions in order to help authors in developing their on-line courses. It should be noted that the multimedia content is not created by this authoring system interface, instead the content will be created by other tools such as the AudioGraph Recorder [2]. The functions implemented by this authoring interface are:

- Creating structure in the TILE database and linking the content together. Editing that structure, so that, for example, new structures can be superimposed onto existing material in order to reuse that material.
- Creating precedence between components, so that, for example, the student is aware what the prerequisites of a given module are.
- Annotating structure with keywords and other meta-data in order for the student to be able to locate material to study and for the developer to locate material for reuse.
- The system will also provide an interface to the various authoring tools used to provide content and provide a means of checking media into and out of the TILE database and learning object repository, from which it can be published to the students.

To achieve these functions, we have to consider the architecture of the authoring system first and define the interfaces between its sub-components. The authoring interface system is actually in two packages, the local application package, which the users interact with and which sends request to the development database, and the local database package, on the lecturer's computer, which receives requests, processes data, and then sends information back to the author if necessary.

So first let us look at exactly what tasks the system needs to perform. Its functions include:

- Creating new course structure and editing existing course structure in the TILE development database. The application can communicate with the local database through instantDB or XML files.
- The local database is updated every time an action is taken by the author.
- Delete course structure when it is no longer needed. Again the database is updated automatically according to changes.
- Managing the version control on the development database server. This means checking data into and out of this database and the central repository for learning objects.
- Publishing new course material on line for delivery using the Course Delivery system. This happens when the author finishes a new course and wishes to publish it. The content is moved from the development to the delivery database (See figure 3). The TILE database will keep track of version information concerning the courses that have been published.
- Launch tools such as the AudioGraph recorder [2] or web-editing tools etc., when the authors need to create new media and keep track of the files in order top provide links within the TILE delivery system.
- Authors can share existing media by editing structure, which references the media. The Authoring Interface system will provide drag and drop functionality in order to edit structure and precedence

As indicated above, the Authoring Interface system will involve implementing a version control system for keeping track of changes in the course material. Version control deals with organizing projects and project components, tracking changes, and supporting parallel development [32].

Other functions provided by the Authoring Interface system will be a login Interface to identify and validate the user's identity. Different users will have different access rights. For example, course controllers have right to edit the courses they control and to delegate that authority to other authors. Also the Authoring Interface system must provide tracking change functions, which records any changes that may have been made to material that has been checked into the central database.

It was considered important to have a working knowledge of the full TILE Authoring Interface system, even though this project produced only a prototype of the Intermediate TILE Authoring Interface system, described in the next section. Early work on this project was targeted to this full system but was redirected in order stabalise the project's specifications. A number of different sub-systems and components that were used in designing and implementing this system are described in the Chapter 3.

2.3. Intermediate TILE Authoring Interface System

2.3.1 an overview of the Intermediate System

In order to investigate the functionality of the TILE Authoring Interface system, without impacting the development of the delivery system, we have designed an intermediate system that will enable us to demonstrate the system's functionality in a stand-alone mode, i.e. in the absence of the full TILE system. The intermediate system can be deployed without the existing TILE authoring and delivery clients and can be served using a conventional web server. This intermediate system derives functionality from the TILE Authoring Interface system project, as a component of the whole system. The Intermediate system is a simplified system, designed to be deployed to individuals developing courseware independently, rather than as an institutional users. It does this by eliminating the central database and the check-in and check out functions from the TILE project server. Instead of the TILE delivery system, an XML database, and a standard web server will be used for the course delivery. This intermediate system focuses on the individual user who wants to create a course of lectures or other materials and to publish that online.

This approach also allows us to tackle a smaller problem initially, and to solve some of the problems that are common to the full Authoring Interface, which was simplified to provide this intermediate, stand-alone solution. Figure 2.3 depicts the various components of TILE Intermediated System. This should be easily modified to provide an implementation of the full system.

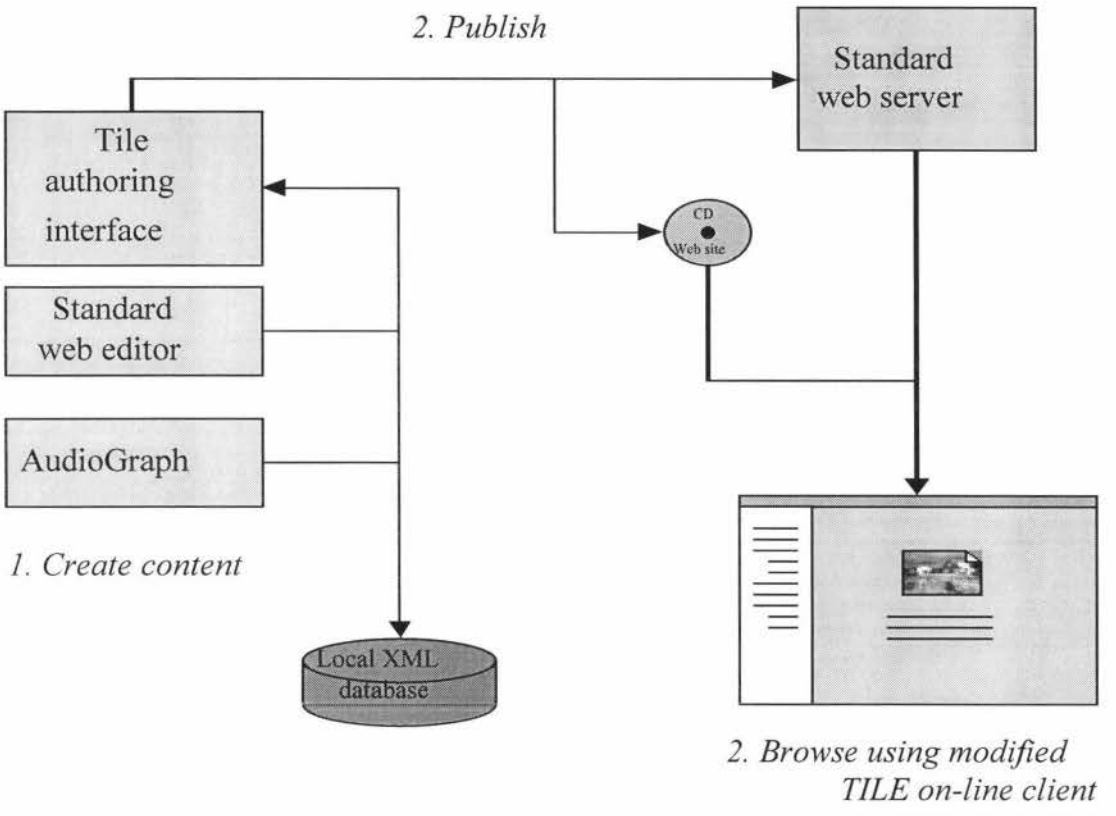

Figure 2.3 Intermediate TILE System using the Authoring Interface tool.

2.3.2 TILE Intermediate Authoring Interface System requirements

Our goals in the intermediate system are to provide the basic functions for an author to develop on-line course structure and publish it via a standard web server. This system is in essence a subset of the authoring interface system; it inherits some functions from that although the client will be significantly simplified. The authoring functions in the intermediate system are:

- Creating new course structure and linking the content together. Editing structure, so that, for example, new structures can superimpose on existing material in order to reuse that material. The new courses and edited courses will be stored in the local XML documents.
- Delete course structure when no longer needed. Again the XML database is updated automatically according to changes.
- Open existing course structure according to the XML database. Users may then further edit it and save this in XML database.
- Launch tools such as the AudioGraph recorder [2] or web-editing tools etc., when the authors need to create and incorporate new media.

- Publishing course materials on-line. This happens when the user finishes developing their courses and decided to publish them on-line.

This system is simpler than the original authoring system, but still provides the functions needed for an author to develop on-line course material.

Chapter 3. TILE Authoring System Environment

In this chapter we look at some of the underlying technology that has been used in this project and ask the question why it has been chosen for our requirements.

3.1. JAVA™

3.1.1 JAVA™ overview

This project has been written in the Java™ programming language and integrates XML™, SQL™ and JDBC™. First of all we look at JAVA™. The JAVA™ programming language is a general-purpose, concurrent, class-based, object-oriented language [3]. Nowadays, Java™ has become the most popular computer programming language in the development of network-based applications. More and more applications and systems are being developed in Java™ because of its cross-platform abilities. It provides powerful functions to programmers via a number of APIs to write various applications, which cooperate with other technologies, for example: databases, HTML, XML, TCP/IP, mobile information technology and so on.

3.1.2 Why JAVA™?

The main reason we have chosen to use JAVA™ is mainly for its cross-platform development capabilities. "Run anywhere" is the best description for the JAVA™ language. Looking back to Chapter 1, we can find what features were listed that contribute to good software development, there we mentioned cross-platform operation. That is the primary reason we have considered JAVA™ as the development language. Also our aim is to develop an application, which helps users to create on-line lectures. JAVA™ has extensive libraries for coping with the TCP/IP protocol and from this point it makes connecting to a network much easier [4]. We look at some major characteristics of JAVA™ below:

- *Simple*: When we build a system, we want it to be programmed easily, without a lot of detailed training. These days most programmers use object-oriented programming C++. Java™ was designed to resemble C++ as closely as possible, in order to make the system more comprehensible. Java™ however, omits many poorly understood and confusing features in C++, and has also added automatic garbage collection. Another aspect of its simplicity is in being a small language. Software constructed in Java™ can run stand-alone in a small machine [43] and this is important in developing embedded systems.
- *Object-Oriented*: Java™ is an object-oriented language. Object-oriented design focuses on the data or object design and the interfaces to it. The object-oriented facilities of Java are essentially those of C++, with extensions from Objective C for more dynamic method resolution [43].
- *Network-Aware:* As we mentioned before, Java™ has extensive libraries, which can help with developing TCP/IP applications, such as methods for Http and Ftp protocols, which are two of the most commonly used on-line protocols. "Java

applications can open and access objects across a network via URLs with the same ease that programmers are used to when accessing a local file system [43]".

- *Robust:* Java puts a lot of emphasis on early type checking for possible problems, later dynamic (runtime) checking, and eliminating situations that are error prone [43]. The biggest different between Java™ and C/C++ is that Java has a pointer model, which can eliminate memory overwriting and data corrupting. Instead of providing pointer arithmetic, Java™ has true arrays that perform subscript checking.
- *Secure*: Java is designed for use in networked/distributed environments. Java is able to construct "virus-free, tamper-free systems; its authentication techniques are based on public-key encryption [43]".
- *Architecture Neutral:* Java™ can execute anywhere on the network. To do this, Java™ generates byte code instructions that have nothing to do with particular computer architecture. With Java, it is easy to interpret this byte code on any machine and it is easy to translate it into native machine code on the fly (This process is called just-in-time compilation).
- *Interpreted:* Java™ byte codes are translated on the fly, which means more compile-time information can be carried over and be available at run time.
- *Multithreaded:* C and C++ are single-threaded programming languages and must rely on the underlying system to write code to deal concurrency. Java, on the other hand, "has a sophisticated set of synchronization primitives that are based on the widely used monitor and condition variable paradigm introduced by C. A. R. Hoare [43]". The Java library provides a thread class that supports methods to start threads, run threads, stop threads and check on a thread's status. Java libraries are threads-safe. Thread safe means a given library function is implemented in such a manner that it can be executed by multiple concurrent threads of execution [64].
- *Dynamic:* "In a number of ways, Java is a more dynamic language than C or C++. It was designed to adapt to an evolving environment [43]". In Java™, libraries can freely add new methods and instance variables without any effect on their clients. Classes have a runtime representation, there is a class named Class, instances of which contain runtime class definitions [43]. Also classes are linked in as required and can be downloaded from across networks. Incoming code is verified before being passed to the interpreter for execution.
- We have also considered some of the potential disadvantages of using the java language, which are mainly concerned with performance. Java performance is poor compared with C++, because the Java byte code needs to be interpreted and run by Java virtual machine. However, speed is not the major issue for TILE project, and also java performance will not be a problem in the future as computer architecture continues to be developed, with significant performance increases with each new generation. JVM is needed to run a java application, but the Java Virtual Machine (JVM) can be downloaded from web site for free.

Based on the above features of Java, we have decided that Java is the best choice for us to develop the authoring interface system.

3.1.3 AWT

AWT stands for Abstract Window Toolkit. It was developed before the Swing Package, as part of the Java Foundation Classes (JFC). JFC is the standard API, which provides general systems functions and a graphical user interface for programmers. The AWT is a portable GUI library. It runs on Solaris, Windows 95/NT and Mac OS System 7.X and above. It can be used for stand-alone applications or applets [59]. AWT connects the application to the host system's native GUI. The following diagram shows the GUI hierarchy.

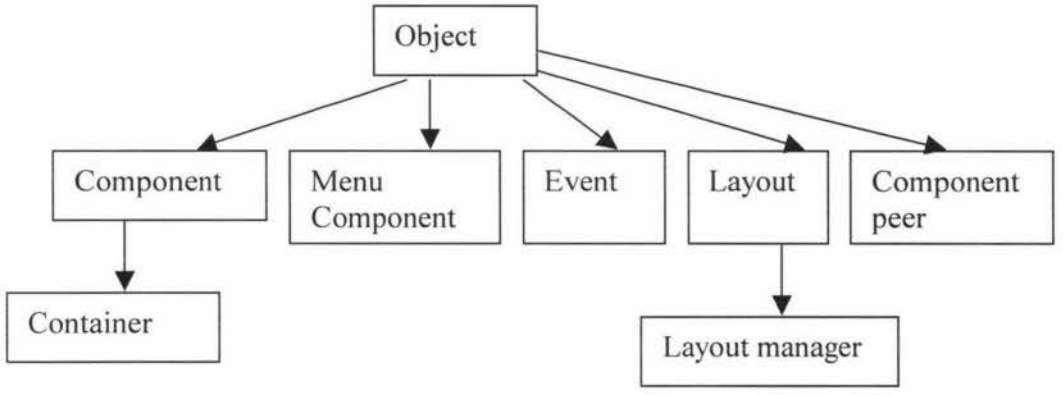

Figure 3.1 ATW GUI hierarchy diagram [65]

The AWT provides many classes for programmers to use. The four basic classes are:

- *Containers:* e.g. Window, Frame, Dialog, and Panel. They can contain components, also containers are components as well, which means that containers can be added to a container. Event handling usually occurs to the components that are added in the containers. All of the methods of components can be used in containers.
- *Components:* e.g. Textfield, Button, Label. Components are added to containers. Generally users interact with components. They provide the objects with which the user interacts, the windows, buttons, etc, how to access features and functions and how to enter text, and so on.
- *Event:* the event class defines various types of events that can occur. These include a user's action with the mouse, keyboard, etc.
- Layout is described in the LayoutManager class, which manages how Components are "laid out" within a Container.

The AWT is targeted at providing major quality improvements in the user interface, while introducing the beginnings of a richer infrastructure for larger-scale GUI development [58]. The AWT is a Java package and can be used by importing java.awt.* via the import keyword.

3.1.4 JFC/SWING GUI Components

JFC stands for Java Foundation Classes. JFC/Swing extends the original Abstract Window Toolkit (AWT), but has not replaced it. JFC adds a set of graphic user interfaces class libraries. The Swing package was included in the JAVA2 run-time environment, but a separate Swing package can be downloaded to add this functionality to JDK1.1. With Swing, you can develop lean and efficient GUI components that have precisely the *look and feel* that you specify. Below we look at what features the Swing package can provide [44]:

- Swing is lightweight and is not built on the GUI of the native operating system.
- Much bigger set of built-in controls than AWT. Swing provides a lager range of component controls, like Trees, image buttons, tabbed panes, sliders, toolbars, colour choosers, tables, etc.
- Much more customisable. The border text alignment can be changed by users. Or images can be added to almost any control. Internal representation and visual appearance can be separated.
- It has a *Pluggable* look and feel. The user can change the look and feel of the application or applet at runtime, as it provides functions to support a user defined look and feel.
- Many miscellaneous new features. For example it has double-buffering built in, tool tips, keyboard accelerators, custom cursors, etc.

Swing components are lightweight, just like the AWT components. But Swing components contain far more functionality than the AWT toolkit did. Also Swing components provide many new features and capabilities, which AWT did not have. As indicated above, the major new feature of Swing is the *Pluggable* look and feel. Swing components extend the lightweight UI Framework, which became part of the Java AWT with the introduction of JDK 1.1 [45].

3.2 XML

3.2.1 XML Overview

"Extensible Mark-up Language (XML) is a meta-mark-up language that provides a format for describing structured data" [5]. It was developed by the W3c (the World Wide Web Consortium), and has been shaped by the experience of previous mark-up languages [9]. It is much like HTML in principle, but there are major differences [36]. XML and HTML were designed with different goals:

- XML was designed to describe data and to focus on what data is.
- HTML was designed to display data and to focus on how data looks.

3.2.2 Why XML

The purpose of using XML in the TILE system is to store lecture data, represents that data in a tree structure and send that data between components of the system. The problem is that the structure is user-specified. As already indicated, the purpose of XML is to structure, store and communicate information. Also the XML tags are not predefined, we must "invent" our own tags. If we look at the syntax of XML, we will find that, unlike HTML, XML tags tell us what the data means, rather than how to display it. You are free to use any XML tags that make sense for a given application. This is just what we need to solve the user-specified structure problem.

The Authoring interfaces as well as the TILE system is designed for the users to create online lectures as a structured document, we consider that XML is the best choice for data interchange on the Web. There are two reasons for this: firstly, XML is used to exchange data. In the real world, different computer systems and databases have different data formats, also the developers of on-line client/server applications have to spend a significant time on data exchange over the Internet. Converting data to XML therefore, can reduce the format complexity and development time as XML can be read by many different systems. Secondly, XML was designed as a way to structure and store data. With XML, data can be stored in a plain text format and hence can be embedded in the html protocol and passed through firewalls. Applications can be written to store and retrieve information from the store, and generic applications can be used to display or parse the data. If we compare XML with traditional databases, XML is easy to install, is easily processed and most importantly, in our case, it comprises plain text, which is easy to edit and communicate. XML is simple, and very flexible.

There are also other advantages in us choosing XML. For example, it is an emerging standard and a lot of developments, especially in B2B applications (Business To Business) will be using XML in the near future. We have specified the learning object models and represented these learning object models as XML documents. With XML, we can make data available to more than just standard HTML browsers. Finally there are other new languages being created, which are based on XML, e.g. WAP (Wireless Application Protocol) and WML (Wireless Mark-up Language).

So what do XML documents looks like? Here is an example of an XML document:

```
<?XML version="1.0"?>
<COURSE>
<COURSENUMBER> 159703 </COURSENUMBER>
<COURSENAME> Advance computer system </COURSENAME>
<POINT>12.5 </POINT>
<MODEL> internal</MODEL>
<SECTION/>
</COURSE>
```

A few things need to be pointed out in the above document:

- The XML document starts with the processing instruction <?XML version ="1.0"?>, this is the XML declaration.
- There is no document type declaration (DTD) in this example, because it is not compulsory. We will talk about the DTD later in this section.
- Empty element, <SECTION/> is an empty element in this example. It has a trailing slash at the end of the brackets, which indicates to a program processing the XML document that the element is empty and no matching end-tag should be sought. It is equivalent to <SECTION> </SECTION>.

The reasons why XML is important can be summarised using the following points from [31]:

- Plain Text
- Data Identification
- Stylability
- Inline Reusability
- Link ability
- Easily Processed
- Hierarchical

We will talk more specifically about XML in the following two sections. Also we give in Appendix A the XML database schema specification defined for this project.

3.2.3 DOM or SAX

The *Document Object Model* (DOM) has been established primarily to specify how future Web browsers and embedded scripts should access HTML and XML documents [9]. The W3c has developed the DOM and it defines the way in which an XML document can be accessed and manipulated. With XML and DOM, we can create an XML document, navigate its structure, and add, modify, or delete its elements. "The DOM originated as a specification to allow JavaScript scripts and Java programs to be portable among Web browsers [37]". XML processing usually uses a program called an XML parser to load the XML documents into the computer's memory, where the XML document's information can be retrieved and manipulated by accessing the DOM. In the DOM, a document has a logical structure, which is much like a tree, more precisely it is more like a "forest", which means that a document can contain more than one tree. So the tree view has been widely used in representing XML documents. Generally, a document contains zero or one doctype node, one root element node, zero or more comments or processing instructions. The root element serves as the root of the element tree for the document. It is called the "*Document Object Model*" because the documents are modelled using objects, the model encompasses not only the structure of a document, but also the behaviour of a document and the objects of which it is composed, so the nodes in DOM documents are not representing a data structure, but a functioned and identified object. As an object model, the DOM identifies [37]:

- The interfaces and objects used to represent and manipulate a document

- The semantics of these interfaces and objects - including both behaviour and attributes
- The relationships and collaborations among these interfaces and objects

Let us try to define what the DOM really is and how it may be used:

- DOM specifies how objects may be represented in XML, how XML and HTML documents are represented as objects, so that they may be used in object-oriented programs.
- This is possible because the DOM is a set of interfaces and objects designed for managing HTML and XML documents [37].
- The DOM is simply an API (Application Programming Interface) to XML.

For example, DOM can be used for creating templates, where the document structure and the way in which that structure is displayed can be kept separate, DOM treats XML document as a tree. In this case, the XML document structure is the same but the DOM defines the way the document is displayed and accessed.

SAX is another API for dealing with XML documents "SAX (the Simple API for XML) is a standard API for event-driven processing of XML data, allowing parsers to deliver information to applications in digestible chunks" [9]. SAX is not a W3C recommendation, but was created by members of the xml-dev mailing list led by David Megginson [39]. Most programmers probably use the Document Object Model to manipulate XML document, because with DOM, they can do pretty much do anything they want to do with XML document. So the question that has to be asked is what are the advantages of using SAX and why is another standard required? The answer is performance! SAX provides speed and simplicity. "The Simple API for XML (SAX) is an industry-standard API intended for high-performance XML document processing [39]". Another advantage is that you don't have the whole document resident in memory at any one time, which matters if you are processing really large documents [40]. SAX is not suitable for modifying the document's structure in a complicated way, for example we could not re-order a book's chapters by using SAX, but we might use it when we want to change the name of elements or attributes in the content. Compared to the tree-based API - DOM, the Event-based API (SAX) provides a simpler, lower-level access to an XML document [42]. In summary:

- It can parse documents much larger than available system memory
- Using a call back event handler, we can create our own data structure.

So SAX and DOM are appropriate for different situations and we must ask which one should we should be using to produce our XML data structure. This depends on what we need when creating an XML application. In this project, we use DOM to access the XML document rather than SAX, because SAX is a bit harder to visualise, and cannot "back up" to an earlier part of the document, or rearrange it. Also we do not have a large document or a complex structure. The structure we have in this project is the course

structure, which is actually a tree perhaps with some other relationships superimposed upon it. Because DOM represents the XML structure as a tree, it is more suitable for course structures in this project.

The Parser we have used in this project is the Xerces Java Parser 1.2.3, which supports the XML 1.0 recommendation. Also we have chosen this Parser because it is written in pure JAVA, can hence be used anywhere. It contains advanced parser functionality, such as XML Schema, DOM Level 2 version 1.0, and SAX Version 2, in addition to supporting the industry standard DOM Level 1 and SAX version 1 APIs [38].

3.2.4 DTD

The Document Type Definition is actually a part of the XML specification, rather than a separate entity. The DTD specifies the kinds of tags that can be included in the XML document. But it is optional for an XML file – you can write XML document without it. Therefore, why should we use DTD?

- Using a DTD enables the parser to validate the XML structure, to see whether the XML document you are reading is valid or not;
- Publishing a DTD will allow different people to use a common DTD to exchange data in a particular format for a specific application;
- In essence, "Each of XML files can carry a description of its own format with it [41]".

The DTD can exist as a component of the XML document, as a part of XML prolog, or it can be a separate entity. If the DTD is included in the XML document, it should be wrapped in a DOCTYPE definition with the following syntax: <!DOCTYPE root-element [element-declarations]>. If the DTD is as a separate entity to XML document, it should be wrapped in a DOCTYPE definition with the following syntax: <!DOCTYPE root-element SYSTEM "filename">.

In the Appendix B, we give the DTD defined for this project.

3.3. SQL

3.3.1 Introduction to SQL

SQL stands for Structured Query Language. It is the language, which communicates with relational databases. SQL is an ANSI standard, which many databases can understand. So an SQL application is (theoretically) independent of the database engine and using it, any database can inter-operate if they adhere to this standard. In a relational database, all data is represented in a table format. These tables are separated but equal [8]. What we want to do is to handle all of the communications with database. So SQL is used for data manipulation, data definition, and data administration [8]. To use SQL statements or commands to handle database management and communication is very easy and efficient.

3.3.2 What does SQL do

The most important feature of SQL is that it provides a uniform and high-level access to relational databases. SQL allows users to access data in relational database management systems, such as Oracle, Sybase, Informix, Microsoft SQL Server, Access, Instant DB, and others, by allowing users to describe the data the user wishes to see. SQL also allows users to define the data in a database, and manipulate that data [6]. More generally, SQL is data sub-language in which you can write SQL commands embedded within some other language such as C, C++ or Java.

In the complete TILE system, we use a database to store the data and transfer data through the server. There are two database implementations used in our system: they are Instant DB and MySQL. Both work with SQL. Instant DB is small Java database, which is easy to install and easy to access. Instant DB is resident on the client side, and is used to store local data. MySQL is slightly larger and resides on the server side.

There are main two categories of statements within SQL: the data definition language (DDL) statements, and the data manipulation language, (DML) statements.

The data definition language (DLL) statements are fundamental to SQL, and include the following:

- CREATE TABLE statement,
- DROP TABLE statement,
- ALTER statement,
- GRANT statement,
- REVOKE statement.
- CREATE INDEX statement
- DROP INDEX statement

The data manipulation language (DML) statements are used to manipulate data in the database, and include:

- SELECT <table>.<column>, <table>.<column> FROM <table1> <table1alias>, <table2< <table2alias> WHERE <condition>. This statement does the job that selecting the data from of which tables they belong to. WHERE <condition> is part of the statement that is telling the database what criteria the data need to meet.
- UPDATE <table> SET <column> = <value> WHERE <selection> = <value>. Update statement is to tell the database some information within it needs to be updated. The information that meet the selection condition is selected from the database will be updated to the given new value.
- DELETE FROM <table> WHERE <column> = <value>. Delete statement is to delete specific data from the database tables. WHERE <column>=<value> is to tell database what exact data needs to be deleted. On other words it is to delete records that match the criterion

SQL also has a lot of built-in functions for counts and calculation, known as Aggregate functions. They are:

- MIN returns the smallest value in a given column
- MAX returns the largest value in a given column
- SUM returns the sum of the numeric values in a given column
- AVG returns the average value of a given column
- COUNT returns the total number of values in a given column
- COUNT (*) returns the number of rows in a table

SQL is a very efficient language because its high-level code is much more compact than other languages. For example, we write one SQL statement as follows: SELECT * FROM books. This simple code would return a list of all books. Conversely, other programming language would require something like the following:

Open Database
Read data
Print data
Repeat Read & Print until end of table
Close database

SQL is faster than other language because it is optimised to find data based upon the database structure.

3.4. JDBC™

3.4.1 JDBC™ Overview

JDBC™, Java Database connectivity, is a bridge, which connects Java programs and standard SQL databases. Basically it is the Java API for communicating with databases. JDBC specifies the interface to connect to the database, and then it executes SQL commands and queries, and interprets the results. The JDBC interface is both database-independent and platform-independent. JDBC can be compared with ODBC (Open Database Connectivity), as they have the same basic mission, to provide database access through a standard interface. An ODBC API is written in C rather than Java, but we can also access ODBC through JDBC. The Figure 3.2 shows the architecture of JDBC.

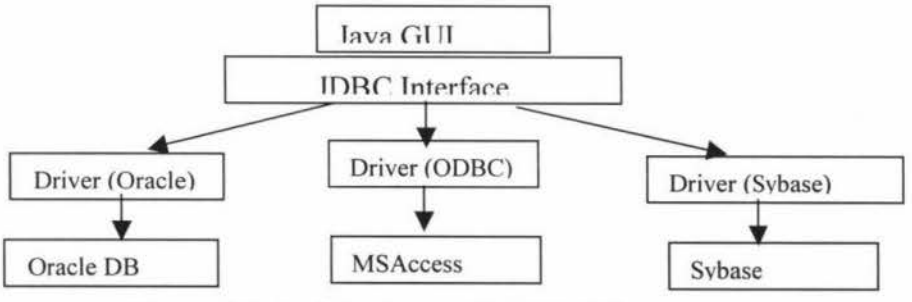

Figure 3.2 Architecture of JDBC [51]

3.4.2 JDBC Usage

There are two stages in developing a JDBC application.

1. Database access.

The JDBC API cannot talk to the database directly by itself. That is because "JDBC only defines a database-independent interface and a collection of helper classes for handling results, errors, and the like [50]". In fact, JDBC accesses the database through the JDBC driver, which is usually provided by the database vendor or a third-party provider. We can find the JDBC API in the package java.sql. It makes it possible to develop database applications using Java, which are independent of the database engine used, provided that the database has a JDBC driver. A Java application built on top of the JDBC API goes through three different phases [52]:

1. Open a connection to a database;
2. Create statement objects, through which it passes SQL statements to the underlying DBMS;
3. Retrieve the results.

2. Specify JDBC drivers.

JDBC drivers are available for most popular databases. There are basically four types of JDBC technology drivers, which are [49]:

> Type 1. JDBC-ODBC Bridge Driver. It provides JDBC API access via one or more ODBC drivers. To use this type of driver, some ODBC native code and in many cases native database client code must be loaded on each client machine. If the automatic installation and downloading of Java technology application is not important, this kind of driver can be considered.
>
> Type 2. A native-API partly Java technology-enabled driver. It converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. It is like the JDBC-ODBC bridge driver and needs some native code to be loaded on the client machine.
>
> Type 3. A net-protocol fully Java technology-enabled driver. It translates JDBC API calls into a DBMS-independent net protocol that is then translated to a DBMS protocol by a server. This net server middleware is able to connect all of its Java technology-based clients to many different databases.
>
> Type 4. A native-protocol fully Java technology-enabled driver. It converts JDBC technology calls into the network protocol used by the DBMSs directly. This allows a direct call from the client machine to the DBMS server and is a practical solution for Intranet access.

Once we have JDBC driver installed, what we need to do is to establish the connection from the application to the SQL databases. No matter whatever SQL databases you use,

the connection is a two steps process [54]:

1. Load the JDBC driver.
2. Establish the connection.

For the programmer, JDBC is implemented in a simple java package, called java.sql.*. The application needs to import the java package by using importing java.sql.* command. The connection process to connect InstabtDB to our project is as follows.

In our case, the Instant DB vendor, Instant Computer Solutions, provides the JDBC™ driver that we need to use in accessing this database, which is a type 4 driver. It supports JDBC API version 1.X. We see in the class.forName() statements, driver names will vary. InstantDB vendor provides it own JDBC driver:

DRIVER_NAME="org.enhydra.instantdb.jdbc.idbDriver".

The syntax of the DriverManager.getConnection () method used in connecting with the database is as follows:

DriverManager.getConnection ("jdbc:idb:tileClient.prp").

The tileClient.prp is the name of the database, which is resident on the local machine. So far we have just established a connection with this database and this connection process is nearly identical for all SQL databases [7].

3.5. TCP/IP

3.5.1 Definition of TCP/IP

The U.S. Department of Defense Advanced Research Projects Agency (DARPA) developed the TCP and IP in 1969. The purpose was to connect number different networks designed by different vendors into a network of networks (the *Internet*). TCP/IP stands for Transmission Control Protocol/Internet protocol. It is a collection of protocols and their implementation in software that can connect dissimilar computers and transfer information between them. It is an industry-standard suit of protocols, which provides communication in any heterogeneous environment. Each component, be it an end system device like a computer, a router or a switch must implement that part of the protocol stack appropriate to it and the level at which it communicates. For a computer this must include the application layer, which in this project are the TILE clients and servers. Because of TCP/IP's popularity, it has become the de facto standard for internetworking. The TCP/IP protocol and its implementation is divided into the following layers [66]:

- Application
- Transport
- Network
- Link

- Physical

The physical and link layers define the means by which data is communicated between nodes, which is normally defined by the transport's frame, such as ethernet. On top of this we have the network layer for which the IP protocol is responsible for. This defines the transport between hosts as datagrams. This is a connectionless service and is unreliable. TCP provides the transport service, a connection oriented service implemented on top of IP. TCP provides some guarantees about delivery and performs some speed matching. The application layer defines the messages and responses that are passed between hosts, in our case the clients and servers. In this section we will talk about TCP and IP respectively.

3.5.2 How does TCP/IP work

First of all we look at the goals for the design of TCP/IP. They are also the reasons why TCP/IP is so heavily used today [16]:

- Good failure recovery
- Ability to connect new networks without disrupting existing services
- Ability to handle high error rates
- Independence from a particular vendor or type of network
- Very little data overhead

Because the TCP/IP model was developed before the OSI model, it doesn't fit the OSI 7 layers model, the TCP/IP model only has the 4 layers defined above. Link layer, Internet layer, Transport layer and Application layer. The major differences between TCP/IP and OSI are:

- The application layer in TCP/IP handles the responsibilities of layers 5,6, and 7 in the OSI model.
- The transport layer in collection of protocols normally refereed to as TCP/IP does not always guarantee reliable delivery of packets as the transport layer in the OSI model does. TCP/IP offers an option called UDP that does not guarantee reliable packet delivery. UDP is used for many transient services, such as domain name service, as well as for real-time multimedia data.

IP, also known as Internet Protocol, is the network layer of the Internet that acts as a carrier for transport-level protocols. It handles all network-level addresses. IP is responsible for moving data packets from the host to host. IP forwards each packet based on a four bytes destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations, including Internet service providers (ISPs). The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world. IP is a connectionless and unreliable protocol. It gives a best-effort delivery service of datagrams across the Internet. We say IP is a connectionless and unreliable

protocol because it doesn't make sure the data packets are reliably sent and reassembled in the order in which they were sent.

IP supports the following functions [56]:

- Data encapsulation and header formatting
- Data routing across the internet work
- Exchanging data across protocol boundaries with other protocols
- Fragmentation and reassembling (for example the maximum size of the data frame for Ethernet is 1514 bytes)

TCP is stand for Transmission Control Protocol, it is transport level protocol. It is responsible for verifying the correct delivery of data between hosts, for example from client to server. Data can be lost in the intermediate network but TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received or it is determined that it cannot be. It is a connection-oriented protocol, and runs on top of IP. Since TCP is a connection-oriented protocol, the connection has to be established between a socket on the server side and a socket on the client side, this connection protocol is known as handshake. This establishes resources, such as buffers, on the end-points only. It does not however, establish a connection in the telephone system sense as no physical circuit is established between hosts. Establishing the connection serves only to reserve resources at the hosts, not in the network itself.

Once the connection has been set up, either side can send or receive data from the other side. Because it is a connection-oriented protocol, TCP can provide guaranteed delivery. It makes sure that data packets have arrived and delivers them in the sequence in which they were transmitted. In order to provide guaranteed and ordered delivery, TCP uses various mechanisms, such as sequence numbers, acknowledgments, 3-way handshakes and timers. There is another protocol at the transport layer and that is UDP (The User Datagram Protocol). UDP is a connectionless protocol; it doesn't guarantee end-to-end delivery. It is usually used for delivering real time video or voice data. The reason of why it is faster than TCP is because it doesn't care whether the packets are lost or not. We are not going to give details about UDP here, as it is not relevant to this project.

TCP has the following characteristics [57]:

- Is fully reliable
- Is connection oriented
- Is acknowledged
- Is data stream-oriented
- Support data fragmentation and reassembly (like IP)

Each has well known ports which are reserved for these services. The primary advantage of TCP/IP is its interoperability. Most of today's networks support TCP/IP as a protocol. TCP/IP also supports routing, accessing the Internet and all of its resources.

3.6. Database Management System

The database has a key role in the whole project. Before we talk about the database used in this project, we have to understand what a database is. "A database is a collection of related data. It can also be viewed as a collection of related tables [46]". Generally there are two types of database: file orientated and relational. File databases, also called flat file databases, look like a word document with each line representing a piece of information and each file is independent of other files. A relational database can easily tie together data from more than one table (the relational join operation) whereas flat files are largely independent of one another [53].

We initially decided to use a relational database on both the client and the server side in the TILE project. At the beginning of development, there were two database systems involved in this project, which were suitable candidates for these requirements, one was InstantDB, a Java database and another was MySQL. As developments continued, we decided to use an XML document to hold the structured data for both transmission across the Internet and, in the intermediate system, for the client-side database system. InstantDB was still to be used as the server side database. In the following discussion we will talk about these two relational databases and also how we use XML document as a database.

3.6.1 InstantDB

InstantDB is a Relational Database Management System (RDBMS), written in the Java language. InstantDB is a small, efficient and easy installed database management system; its features include joins, transactions, triggers, sub selects, table aliasing, XA protocol, and etc. InstantDB is accessed via its own JDBC 2.0™ driver by using standard SQL and Sun's JDBC™ API [47].

Jenny Zhang has specified the client database selection for the TILE system, to which this authoring system interfaces, in her Masters thesis [67]. She gives a comparison between InstantDB and Hypersonic SQL databases. InstantDB is compatible with Java and it runs under JDK 1.1 or later JVMs. Since InstantDB is written in pure Java, so it is suitable for our goal of cross-platform use. InstantDB supports multiple threads and locking, but the locking function is only supported at the table level and not at the row level. As all locks are exclusive, there is no possibility of sharing a table for read access and in effect, a SELECT statement on a table will lock that table against all other SELECTs until the transaction completes [67]. This is not considered a problem on the client system however because it is only satisfying requests from the local user, who is unlikely to wish to make concurrent requests.

Hypersonic SQL is an Open Source Java Database. It supports standard SQL syntax and has a JDBC interface. Hypersonic SQL is free to use and re-distribute. It is a compact, in-memory database, which supports both standalone and Client/Server modes. It can also be used in applets [67]. Hypersonic SQL does not currently support multi threads.

In this project we chose to use InstantDB, which is resident in the local user machine and communicates with the server side, where MySQL is used.

3.6.2 MySQL

MySQL, is the most popular Open Source SQL database. It is "a multi-user SQL relational database server. It can run on most Unix platforms, windows and OS/2 [48, 60]".

In this project, MySQL is used on the server side, and is responsible for the communication to the local database (in XML). In this case, MySQl was selected as sever side database, because not only its stability, but also MySQL provides many features to users [61], such as:

- Being able to work on many different platforms,
- Fully multi-threaded using kernel threads,
- ODBC (Open-Database-Connectivity) support for Win32 (with source),
- A very fast thread-based memory allocation system,
- No memory leaks, etc.

The comparison concerning server-side database selection can be found in "A Feasibility Study for the Design of a Web-based Course Delivery System"[67]. It describes three frequently used databases: MySQL, PostgreSQL and Interbase, from which MySQL was selected for the TILE project.

3.6.3 XML document

On the local machine, we have used an XML document instead of an SQL Database. This is because the local database is arranged as a number of projects, each of which is transmitted across the Internet as an XML file when checked-out. The XML is embedded in the http protocol. We can use this XML file structure for local data storage, eliminating the need for a database server on the lecturer's computer. XML actually provides us with more flexibility to access data and to exchange data though the network.

A Java application program is implemented to enable the TILE authoring system to generate the XML document according to the databases we defined before. In order to represent the entire database in XML document, we have defined a DTD schema.

The following XML document is an example of a database table using some of the tags defined in the schema.

```
<course>
   <courseID> 159703 </courseID>
   <courseName> advance computer architecture </courseName>
    <Prerequisite>None </Prerequisite>
    <keyword> Architecture  </keyword>
    <Staff> Chris Jesshope </Staff>
```

```
    <StaffID> 123456 </StaffID>
    <section>
        <sectionID>1 </sectionID>
        <sectionName>section 1 </sectionName>
        <Prerequisite>None </Prerequisite>
        <keyword> Computer network </keyword>
        <parentID> 0 </parented>
        <type> bottom </type>
        <Objet>
                <object ID>  1 </objected>
                <objectName> picture of the network </objectName>
                < URL>http://www.nzedsoft.co.nz/lecture/159703 </URL>
                <timeCreated> 13/02/2002 </timeCreated>
        </Object>
        </section>
</course>
```

The full-defined XML document sees Appendix A.

Chapter 4 TILE Authoring System Design and implementation

4.1. TILE Authoring System Interface design

The complete TILE system is quite a large and complex system and this project had to be integrated with that work, which was on going during the work described here. When this project first started, the initial specifications for the TILE Authoring system were to use relational databases on both client and server side. Also we planned use the TILE server, which was then in development, as the standard server to download and upload the course information for the authoring tool. In this chapter we will describe the original plans for the complete TILE Authoring System. A significant amount of development work was performed on this full system before it was agreed that it would be cleaner in this project to develop the Intermediate system, already specified in Section 2.3 and described in detail in Chapter 5. The Intermediate system is a subset of this and it was considered easier to demonstrate the project on this independent system.

The main Authoring System is a JAVA application, which presents the user with five main views on the on-line course structure to be delivered using the TILE course delivery system [67]. The main view is the structure display, which is always present. The other four are: data view, action view, media view and filter view. In those views, the users will eventually choose, or will be taken to, a particular piece of information to be viewed. The application acts like a source code control system, in that structure is checked-out by authors, this information is downloaded from the central database and stored on the user's local machine in a local database. The local database keeps the link between the checked-out material on the local machine and the data in central database. While the application is running, the lecturer can connect to the local database via JDBC and also connect the central database if required.

4.1.1 System Interface General Browse

The User Interface of the Authoring Interface System is shown in figure 4.1 below. It shows the *Structure Display* as a tree that can be dynamically opened and closed in the left-hand side of the window. This window also contains the general controls. Figure 4.1 also shows the *Data window* in the right-hand sub-window, one of the four different tabbed displays. Users would add new nodes to the tree structure, or delete any nodes from tree structure in this view. The node represents a course unit or section and contains meta-data for that node. All this information will be display the information area, which is the right upper area and the user can input, edit and delete information for a selected node.

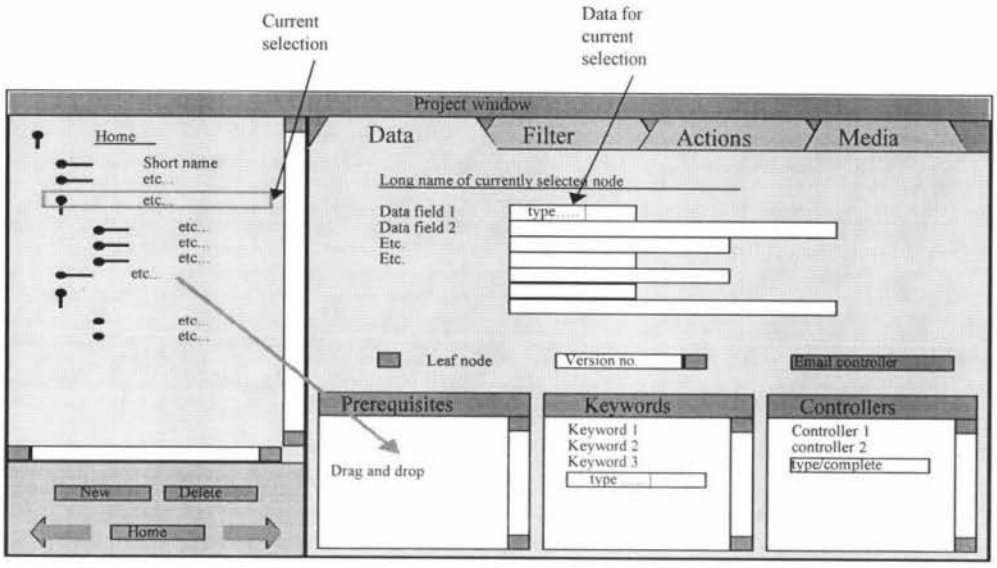

Figure 4.1 Authoring Interface Application, showing data view

The application allows one window to be open for each project and supports cut, copy and paste operations on structure between windows. The general rule for cutting or copying structure is that if a node is copied, it and all of its sub-nodes are also copied.

4.1.2 The Data View

The data view is also shown in figure 4.1. It provides the user with the opportunity to create data or edit data held in the database for that node (including staff details, when a controller is selected). There will always be a current selection in the structure view and the data view displays the data for that node. Each field can be edited simply by typing into the appropriate field or by selecting data from a menu.

There are three additional controls on the data view. The first is a check box, which is checked when the node is a leaf node. A leaf node is one that is, or is going to be, associated with a learning object. The second control is only enabled when browsing on the central server. This is a pull-down menu that allows any version of the node stored in the development database to be selected. Note that an earlier version of the same node may have different substructure to the latest version. The final control is a button, which allows an email message to be sent to the selected controller for that node. Email may be sent either by the application or by using the user's normal email client, in the latter case the client should be opened with new message containing the controllers Email address.

4.1.3 The Filter view

The filter view provides a means of setting search criteria and selecting a point in the structure from any matches obtained from the search. This view is shown in Figure 4.2. We decided to provide more detailed information to help the user search for target information. We provide search types like course, section, or object, provide staff

information such like staff Id, name, etc. Also we provide key words, which might be relevant to any area. The major problem is how we would display the results produced?

If the user is only searching for course information by keyword, a search box could be included in the tree structure. Otherwise we need a display area to show all the information. Since the user not only searches for course information, but also staff information and other information as well, the tree structure display is not large enough to display all information we might need. Also we have to consider how to display the results. There may be many matches to different courses or sections in a course. It was decided therefore that the best way to display information is to have a filter tab, which contains both the criteria being searched on and a display area, which can be a scroll panel, to display the search results in text format. The result panel shows a list of successful matches. These represent the path name to reach the node in the tree that matches the data. Selecting a match in this window creates a new home in the tree display and the data display can be used to view that node. Note that any structure shown includes only the nodes that match the filter and any intermediate nodes required to reach them.

Each search criterion is arranged as a row of three selectors/fields, three such rows are shown. In the first position the user selects the field to be matched, such as keyword, date etc. In the second the user selects the relationship, e.g. for dates this might be: *is, is before, is after*. In the final field the user types or sets the data to be matched.

The *More* button will bring up a new set of selectors/fields for setting an additional criterion. The *Fewer* button removes the last criterion displayed but is only active if there is more than one criterion. Initially one row is displayed. The *Any/all* selector is also only shown when there is a more than one search criterion and it allows the user to select the way in which criteria are combined, i.e. if *any* of the criteria are to be matched or if *all* of the criteria are to be matched.

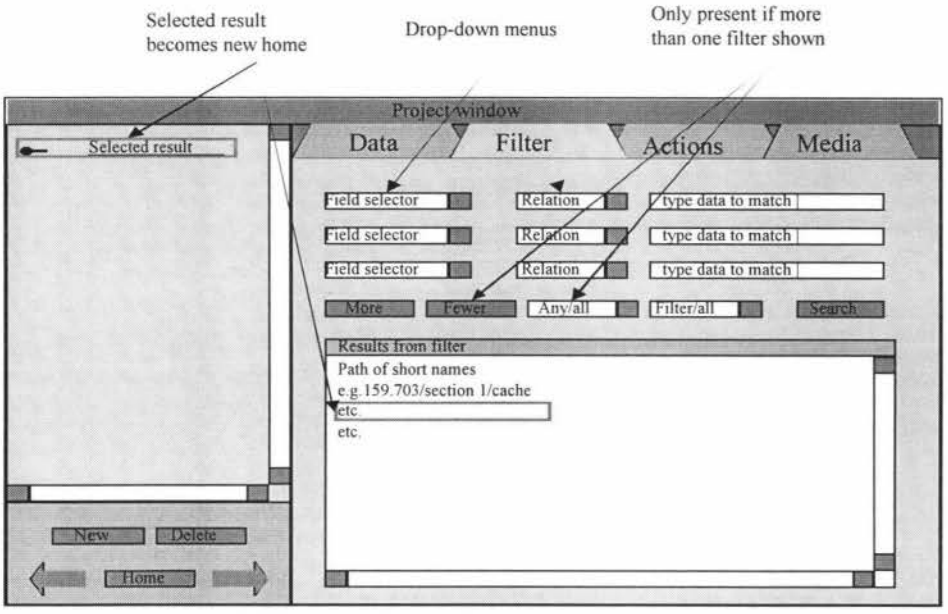

Figure 4.2 The Authoring Interface Application, showing the Filter view.

4.1.4 The Action View

The actions view provides access to the commonly used functions in the Authoring Interface tool. This is shown in figure 4.3. The dialogue area is specific to each function and will be defined as we proceed. The page has the following actions, each associated with a button or selector:
Login server, new project, open project, set home, check-in, check-out, version purge and quit.

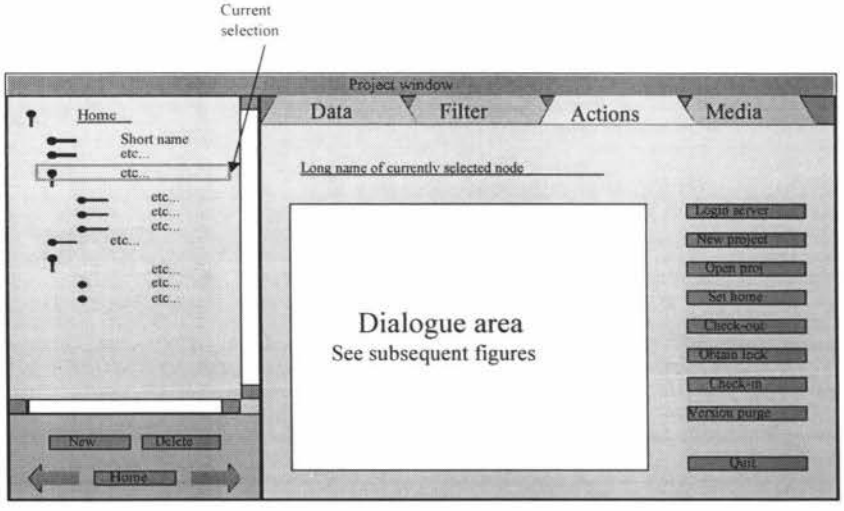

Figure 4.3 The Authoring Interface Application, showing the Action view.

4.1.5 The Media View

The media view is the last view. It provides a window to display the data about and actions that can be performed on the Learning Objects. This view is only enabled when the currently selected node is a leaf node. Leaf nodes are double, in that a structure node from the course section table is linked to a node in the Learning Object table. The node in the section table is course specific, whereas the Learning Object entry may be shared among many courses. Both have keywords, for example, and those in the section table may not be exactly the same as those stored in the learning object node. The default, action, when a structure node is linked to a learning object node, is that the keywords would be copied from the learning object. The *keywords* window can then be modified by adding or deleting keywords as required. This duality is the reason for the media view. When the currently selected node is a leaf node, the two views, Data view and Media view, display data from the linked structure node and learning object node respectively.

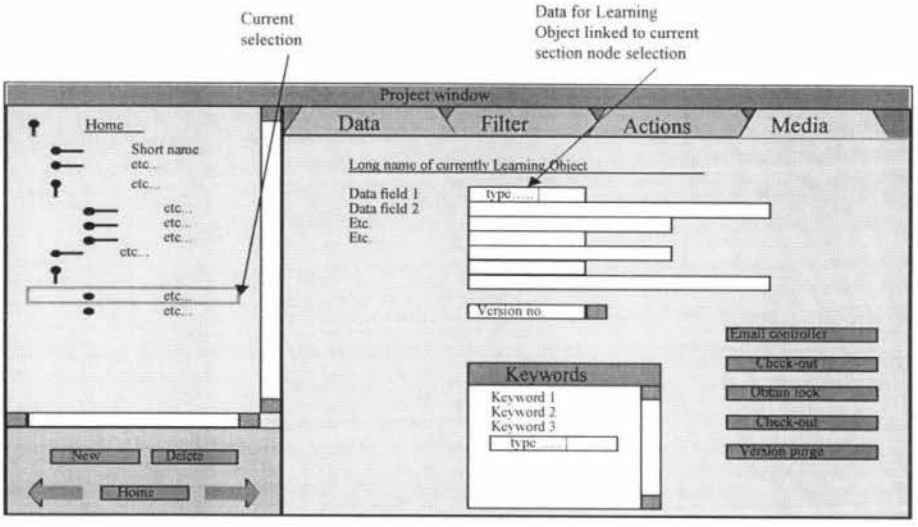

Figure 4.4 Authoring Interface Application, showing Media view

Note that the learning object must be checked out separately. Otherwise the functions are explained elsewhere.

4.2. Database Logical Design

4.2.1 Location of database - local or central

In the Authoring system, we have considered the issues concerning the location of the database. Where we put the database is an important issue for the authoring system. There are two solutions here: use a central database only, or use both local and central databases. We consider these two situations both are possible for this system. What we have to study is which way is more reasonable and flexible.

The local database is running at the lecturer's local machine. So it doesn't have problem with network server and traffic. The only thing that has to be considered is the synchronisation with the main database when the local database has been changed. Every time the local database is changed, the system eventually has to update the central database. So we can see, synchronisation will make some unnecessary transactions between these two databases, and also give rise to data transaction traffic. Almost all of the tables in these two databases are the same, so once the local database has been updated, the central database has to be updated synchronously, obviously the actions have to be duplicated. But it seems that we do need a local database sometimes. Firstly when the server is down or the lecturer does not have access to the internet. In this case, all the data can be stored in the local database. The central database can be updated the first time after the server is reconnected. Secondly because we are concerned about concurrent contribution towards course design. If more than one staff member wishes to make changes to a course, the materials have been checked out to make a copy to the local machine. It better has local database to store this data. Once the material has been finished, the staff member can copy it back to central database by checking it in.

Let's turn to look at the case for a central database. We assumed there is only one central database in the system. In this case, all the functions have to be put in the central database. But because the client side is running a local application, instead of a local database, we could use files to store these data. An application can generate a file on the local machine and store those data temporarily. Then after updating central database, these local files will be deleted. However, in this case we have no control or record of what any staff member is working on.

In summary therefore it is better to have a local database but for that database to be governed by checking out and checking in information to and from the central database, just as happens in a code store control system, like CVS.

4.2.2 Database Access Package (InstantDB and MySQL)

General Specification

There are 4 versions of **section** table in the authoring system. The development section table is on the server side. When developers have changed course material, this table will track those changes and update the central database. The lecturer's section table, it is on the client side. When an author is checking-out the material or complete new material on the local machine, then the lecturer's section table tracks and store the changes of every version number. The delivery section table, it is in the Course Delivery System. Eventually these materials will be published to students. The students usually browse the latest version on line, or download that material to their local machine.

Each staff member has an account to access the Authoring system. The password and user name is unique in a separate *staffPassword* table. The user name and password will be referenced from *saffPassword* table. Once entered correctly, the staff member will be allowed to login to the Authoring system. The staff member might be a contributor, who

can contribute to the complete course, or maybe just a section of it, for example, he or she may be an author, an editor or a course or section controller. A learning object also has a separate object contributor, because the object contributor might be one of the staff members or someone who comes from another organization.

The important thing in this system is the version control system. Not only tracking when a member of staff checks out and checks in, but also tracking any changes that have been made after a staff member checks in their material. **A change** table makes a record of any changes to the material, it includes the modification time, type, who modified it, course ID, section ID, version and changeID. The **locked** table makes a record of who locked the section or course, only controllers have the rights to lock the materials they given when they checking out. Locking is the mechanism for concurrency control. Only one member of staff may hold a lock on a section (and all of its sub-sections) at any one time. After that person checks in and releases the lock, another person can have the right to lock the table. In order to record every change that has been made within course or section, we consider both the **course** table and the **section** table and track the changes.

We only use locking for checking-in anything can be checked out by someone with appropriate permissions. The locking mechanism restricts what and who can be checked-in. Only one person can lock the same material at any one time, and if a node is locked, other people will have to wait until that person has checked-in before they can also acquire the lock to check-in material. Of course they may no longer have the most up-to-date material and there may be a requirement to check -out and manually merge before checking-in concurrent changes. Only course controllers have rights to modify the course they are responsible for. Other people can check-out and copy that material, but cannot modify and check it in. The system will automatically manage the version number and assign new number to the nodes as required, when new material is added. We will give full description of this mechanism in the next section.

Let us first look at the functionality of the Authoring Interface system, before we define any modifications to the database. In the development structure table the user will be able to:

- Add a node to the existing structure;
- Update the node information, including prerequisite information;
- Delete a node;
- Change the position of a node in the structure

Let us consider each of these in turn.

Add new node

To understand the implications of adding a node into the structure table, see the example in Figure 4.5. The section table data needed to represent the addition of this new node given below. We only describe the node information pertinent to this example and that does not include all of the attributes.

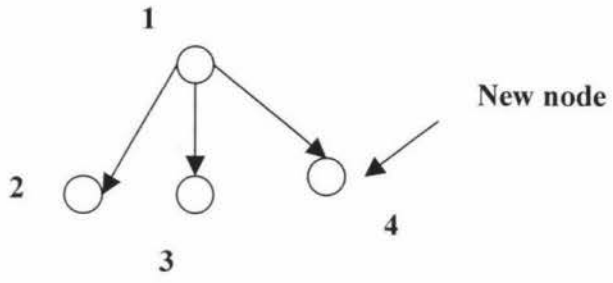

Figure 4.5 A new node is added in sequence, the numbers represent the sectionID in the structure table.

sectionID	parentSectionID	Sequence	version
1	Null	1	1
2	1	1	1
3	1	2	1
4	**1**	**3**	**1**

Table 4.1. Changes (in bold) in adding a new node to the section table.

Note that logically speaking, a node in the structure is uniquely identified by its sectionID and version number. **SectionID** and **version** are automatically calculated by the system and the user does not have to be aware of them. A **version** is associated with every node and, after any changes, the version will be increased by one. When node 4 is added in this case, the **sectionID** that is assigned 4 and its **version** is 1, because there is no previous version of the node with this **sectionID**.

We also have to make an entry in the **change** table and this is shown below

courseID	sectionID	Version	Type	staffID	Date	Sequence
159703	4	1	New	12345	12/1/2001	1

Table 4.2. Change table, which keeps track of any changes that have been made

If the new node was being added between the original two nodes (2 and 3) then additional changes are required. As the section number required for the new node is now 2, which is already being used by an existing node, then that node (and any following nodes in the general case) must first be updated. This would result in two entries in the change table with the same sequence number:

CourseID	SectionID	Version	Type	staffID	Date	Sequence
159.703	3	2	Change	12345	12/1/2001	1
159703	4	1	New	12345	12/1/2001	1

Table 4.3. Change table, which keeps track of any changes that have been made

And the following changes to section table:

sectionID	parentSectionID	Sequence	Version
1	Null	1	1
2	1	1	1
3	1	2	1
3	**1**	**3**	**2**
4	**1**	**2**	**1**

Table 4.4. Adding new node between two existing nodes.

Note that there are now two versions of node 3 in the section table. As the SectionID is generated by the system, there are two solutions to this problem. One is to create a new section Id and the other is to duplicate the section Id. We adopt the first solution as it minimises any changes to other tables and also allows us to maintain earlier versions of the structure in the same table. Because the sectionID is now no longer unique in the development section table, the primary key must be a combination of other keys. We therefore use a combination of **sectionID** and **version**.

Update nodes

The second change we must consider is in updating a node. This could be a key word change, some structure change, or a change in content, etc. Here we only discuss the general situation.

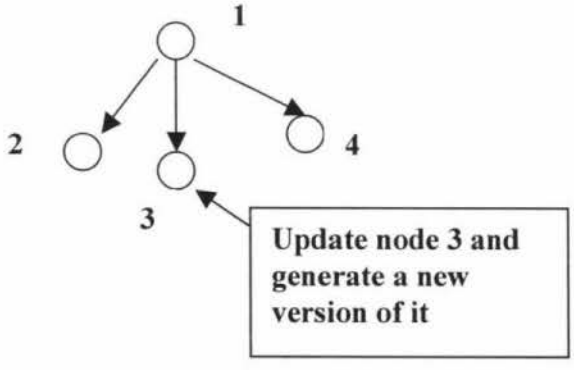

Figure 4.6 Updating a node, the numbers represent the sectionID.

It should be noted that these changes might affect other tables in the database, for example the key word or prerequisite tables. However, as we are able to maintain all versions of a node in the development section table using the sectionID and version to form a unique key, then any relationship with other tables must also use both sectionID and version to create the link. Thus provided that we also maintain both fields in the related tables, we can simply add new entries to these tables when a node has been changed. It will use the new version number. Therefore we only need to identify changes to the section table in the change table and from this we can reconstruct earlier versions of the structure.

Figure 4.6 shows an example of a fragment of structure being updated. In this example, the Node with a sectionID of 3 is being updated in some way. The resulting section table is given below. Note again there are two versions of this node in the development section table.

SectionID	parentSectionID	Sequence	Version
1	Null	1	1
2	1	1	1
3	1	2	1
3	**1**	**2**	**2**
4	1	3	1
5	3	1	1
6	3	2	1

Table 4.5. Changes to the section table in updating node 3

Deleting node

When deleting a node from structure the situation is more complex. Not only must we delete the node specified, but also any children of that node.

From Figure 4.7, we can see that there are two ways to implement the deleting of a node. One is to require the user to delete the descendants of node 3 prior to deleting node 3 itself. The other is to assume that the user wishes to delete all descendents of the node when deleting node 3. We will assume the latter but will inform the user of the consequences and request a positive confirmation prior to deleting the node and its descendents.

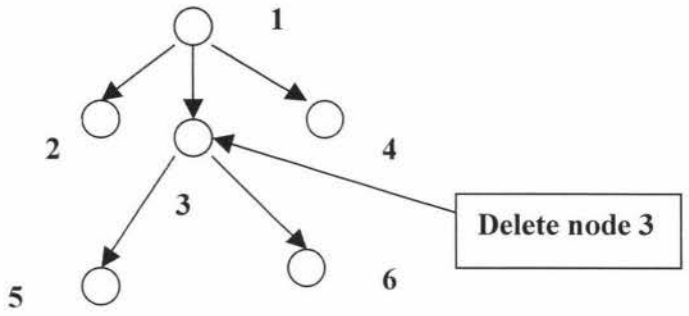

Figure 4.7 Node 3 and its descendents will be deleted

The result of the changes in figure 4.7 is all follows:

SectionID	ParentSectionID	Sequence	Version
1	Null	1	1
2	1	1	1
3	1	2	1
4	1	3	1
5	3	1	1
6	3	2	1

Table 4.6. Section table before node 3 has been deleted

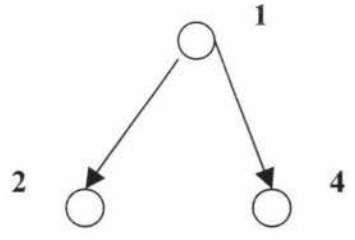

Figure 4.8 Structure after node 3 has been deleted

sectionID	ParentSectionID	Sequence	Version
1	Null	1	1
2	1	1	1
3	1	2	1
4	1	3	1
5	3	1	1
6	3	2	1
3	**Null**	**Null**	**2**
4	**1**	**2**	**2**
5	**Null**	**Null**	**2**
6	**Null**	**Null**	**2**

Table 4.7. After node 3 has been deleted

The system recognises deleted nodes when the parentSectionID and sequence are both null. Thus node 3 and all of its siblings have been given new version numbers and flagged as deleted. A re-sequencing must also be made by system, and the sequence of node 4 must become 2 in this case.

Change of position

A change of position for a node also means change sequence and possibly a change of ParentSectionID. The diagrams below show a simple change in a node's position.

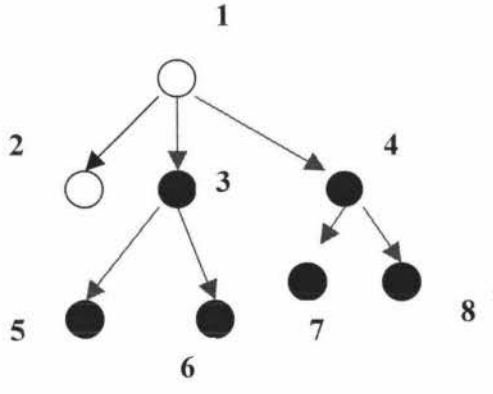

Figure 4.9 Structure before change of position

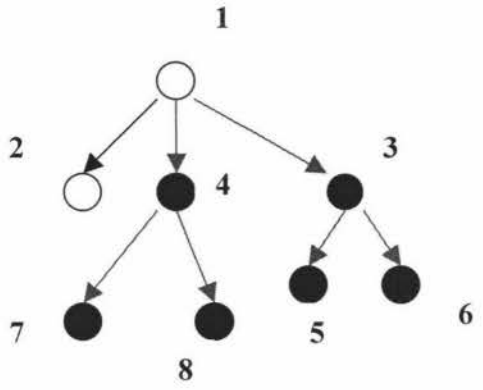

Figure 4.10 Structure after change of position

The corresponding changes to the structure table are given below: changes

sectionID	parentSectionID	Sequence	Version
1	Null	1	1
2	1	1	1
3	1	2	1
4	1	3	1
5	3	1	1
6	3	2	1
7	4	1	1
8	4	2	1

Table 4.8. Structure table before node position change

SectionID	ParentSectionID	Sequence	Version
1	Null	1	1
2	1	1	1
3	1	2	1
4	1	3	1
5	3	1	1
6	3	2	1
7	4	1	1
8	4	2	1
3	**1**	**3**	**2**
4	**1**	**2**	**2**

Table 4.9. Structure table after node position change

We do not change the version number of node 5, node 6, node7 and node 8 because if a node has no change relatively to its parents or descendents, then there is no need to

update its version. We see node 5, 6, 7, 8 have no change relatively to their parent and each other, so we do not need to create a new version for them.

4.3. TILE Authoring System function design

4.3.1 Access

Login and log off

This action allows the user to access the central development database. Login is required to authenticate the user, as some functions, e.g. checking-in, are restricted to the appropriate course controllers. When users log off, the connection with database will be cut off, and any actions taken after log off only affect the local database.

New Project

New project – this function allows the user to create a new project. A number of parameters can be set, for example the location of the project on the user's hard drive, the name of the project and perhaps others.

Open Project

Open project – this function allows the user to open an existing project. The project has to be a specific type file, which can be opened by authoring application.

Quit

Quit - the quit function quits the current project, if the current project is the only project open, the application terminates.

4.3.2 Browsing

The general controls comprise 5 buttons, which are used for moving around and editing the structure. These are:

- Add a *New* node
- *Delete* the selected node
- Go to the user's *Home*
- Go *Forward* (this refers to the next selection)
- Go *Back* (this refers to the last selection)

Adding a new node

The *New* button adds a new, empty node below the currently selected node. If any nodes already exist at this level it will be added to the end of that list. By default any new node

added is a leaf node and un-checking its leaf node box will allow further nodes to be added below it. To add a node to the same level as the currently selected node, the *New* button is used with the option/alternative key depressed. This will insert a new node immediately following the current selection. Using both a standard and an alternative click will allow nodes to be added in one action anywhere in the tree structure.

Deleting a node

The *Delete* button deletes the currently selected node in the tree structure and all nodes below it. The application (by default) will warn the user if there are any nodes below the current node that contains valid data, as these nodes will also be deleted. (N.b. any action performed should be undoable).

Forward and back browsing buttons

The *Forward* and *Back* buttons act like browser buttons. They are used for tracing and retracing the history of selections in the structure display. The application must track and keep a history of two actions:

- **Tracking the selection** - every time the user changes the selection either by selecting a node or by searching for a node, the application will keep track of it. The back button will move back through this history and the Forward button forward through the history. If the selected node is the last selection in the history, then the forward button will become disabled. If the selected node is the first selection in the history, then the back button will become disabled
- **Tracking tree status** - every time the user expands or collapses the structure tree, the application will also add these actions to the history. Thus the user will be able to trace all actions in the display window.

When moving through the history, the current selection will be shown in the centre of the structure area and the scroll position will be adjusted accordingly. The only exception to this is if the structure display is not full, in which case the scroll-bar will be disabled.

Set Home button

The user may set the *Home* location for the structure display in the Actions window for every project and also for the development database. By default this will be the root node of the project and the root node of the server database. When the user opens a project or logs into the development database, then the structure display shows the home location as the root node, with all nodes below it hidden. If user presses the Home button, then at any time they are returned to this initial status.

Version purge

Version purge – This allows the root level controller to purge all but the last n versions of nodes in the development database for a given course. This function is restricted to the root level controller, as it is considered an administrative function. This function cleans the section table, the change table and any other table that contains version information.

4.3.3 Check In and Check Out

For the Authoring Interface system, we have to specify the additions required to the database specification [1] for the checking-in and checking-out functions. Any author who has login access to the TILE system can check out any material they require. In addition they can copy that material, i.e. the structure and references to the learning objects. However, only authors who are responsible for a given course or section of a course can modify the material and check it in. Such authors will be identified as controllers of the structure they are responsible for (controllership propagates down the course structure but may not include the media objects). In order to ensure these restriction and to avoid inconsistent updates of course material and structure, a locking mechanism has to be implemented for the checking-in function. Every change to the database will be stored in a change log table, which can be used to provide an audit trail of any changes and to be able to backtrack on specific changes.

Check out

Check out - this function, which is active only when the user is browsing the development database, allows the user to check out a node and its descendants from the current selection. The dialogue for this function will allow the user to first select a project, into which the structure will be checked-out to. It will then allow the user to select a node in that project, into which the structure will be loaded. This will use a display similar to the structure display. The node selected in the project may be the root if it is empty, or any other leaf node in a non-empty project. If the node has any data, that will be overwritten by the node being checked out. The user will be given a warning if this is the case.

Check in

There are two strategies for locking the database for checking in:

- **Locking after check out** - Any one can check out a section of the given course, which can be modified. When they need to check the changes in, they will have to lock that section on the development section table and then check it in and finally release the lock. The disadvantage of this approach is that during the time between the user checking-out and checking-in, another user may also have checked out the same section, changed it and checked it in again. This is a risk that the user faces when checking out and not locking a section. The first user must now check out the modified section again before checking it in and that user is responsible for manually merging any changes before checking it in again. We can minimise any problems by informing each user of the actions of others when they have the potential to cause such problems. Thus if two users check out overlapping sections they should be

informed of this. Also when the structure is checked in again the user who has also checked out the same structure must be informed.

- **Checking out and locking at the same time** - a user can check out a section and lock it at the same time. Any other person wanting to check out the same structure will be informed of the lock and the user. They may still check it out and make and change a copy at the local machine, but cannot lock the section and check it in until the first user has released the lock. The disadvantage of this approach is that a section may be locked for a long time. Again this situation can be monitored and reminders made. Ultimately the administrator will be able to release locks, as well as the controller who owns the lock.

We adopt the former strategy in this system.

Check in- this function, which is active only when the user has a lock on the currently selected node, allows the user to check that node back into the development database. There are two possibilities, a simple update, in which the currently selected node and its descendants are returned to their original position in the database. The original position of the selected node in the development database can be determined by its **ParentSectionId**. The alternative is to insert the selected node as a new node at some point in the development database. This, for example, might be the case when borrowing material from one course to insert into another. In this case the user must be provided with a structure view of the development database in order to select the insertion point. (Like new this will require two options, insert after and insert below).

Locking Protocol for Check in and out

The protocol for checking-in, locking and checking-out is as follows:

- Anybody may check out a section of a given course.
- Only one of the course controllers for a section may lock that section.
- The lock is propagated down the tree but not to the learning objects. A separate locking mechanism is provided for the learning objects
- Only one author can hold a lock for an entry in the section table at any given time. The lock holder is stored in a locking table and any attempt to lock this entry again will result in notification by email to the person wishing to lock that entry of the identification of the person currently holding the lock.
- Only the lock holder may check in updated information to the development table.
- An author, who has changed some checked-out structure that is subsequently updated by someone else, by locking and checking-in, is responsible for checking out the new material and transferring their changes to the new structure. Warnings will be given by email to anybody who has structure checked out that is updated by somebody else checking in.
- The system will not allow a version to be checked in that has been updated since it has been checked out

So as we know, at any time, only one person can obtain the lock to a particular part of the course material in order to check it in. Other people wishing to modify the same material will have to wait for that person to first check it in and hence release the lock, before they are able to lock it and check in their changes. In order to implement these functions we have to be able to make and keep track of modifications to the section table and related changes to other tables. At some stage the modified courses will be published, i.e. made available to the students in some version. For this reason we will have to keep two versions of the section table on the TILE database server, the development version, which keeps track of any changes and the published version, which contains a clean copy of the structure that has been published.

4.3.4 Filtering

Filtering is a powerful tool for selecting from the development database. For example, all structure produced by a given lecturer could be filtered in the display. Any operations on a window that has been filtered will only selected the filtered data. Thus all material from a given lecturer might be copied or checked out.

The result of a search is a set nodes, which are displayed in the results window. These are the nodes that match any or all of the search criteria. The results window displays the path name from the project root involving all of the short node names to reach it. This will give the user a guide in selecting a result for display. The user can then select any of the paths displayed and the path selected in the results window will become the home path in the structure display.

Depending on the user's choice in setting the *Filter/all* button, one of two alternate displays will be seen in the structure area. If the user has selected the *All* option then every node below filter-set home will be displayed. If the user has set the *Filter* option for the filter, then only nodes that match the search criteria and are below the filter home will be displayed in the structure area. To avoid a disconnected structure, there may also be some nodes that do not match the search criteria but that are required to navigate to a lower node that does. Figure 4.11 illustrates this. Only the visible nodes are copied when copying or checking-out structure from a filtered view,

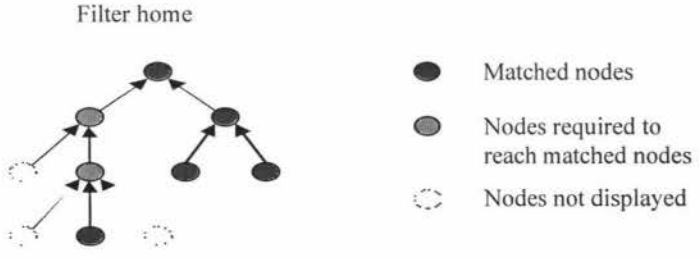

Figure 4.11 A filtered structure

After completing the specification and some of the implementation of the full system, it was decided to continue the work on the intermediate system, due to instability in the full system's specifications. At the moment, we have a worked full authoring system, with database. The functions that have been completed in the full authoring system are:

- Users can open an existing project, read information from the database
- Create new project
- Saving course structure information to the local database
- Update local database
- Different interfaces have been completed.

Further work we decided to move to intermediate system includes migrating these existing functions to the intermediate system and also develop more functions and with more flexibilities.

Chapter 5 Intermediate System Design and implementation

5.1. Intermediate system interface design and implementation

The Intermediate system defined in this chapter, is a simplified authoring system based on the original authoring system specification. The idea is that we are trying to make the application simple, efficient and able to be used in a stand-alone mode, without institutional support for the complete TILE system. The latter requires a supported server with staff and program information etc. What we want is a stand-alone Java application, which runs on the local client machine with a local XML database.

The intermediate system was designed for single users. It allows them to create a local database of a course's structure, which can be uploaded and delivered to students with a simplified version of the TILE course-delivery client. This idea to develop an intermediate system was proposed while the main authoring system was under development. In order to meet the new requirements, some changes have been made in this new interface and the functional specifications have to be simplified. These new requirements are:

- There is no needs for the central database, instead a local XML database is used to capture the course structure
- Do not need to check out, instead, the user will create a new project and open existing projects on the local machine
- Do not need to check in and instead of publishing from the main TILE server, the user will up-load the project to a suitable standard web-server.

There are two main modes with this application, instead of the 4 panels described in the main authoring system. One is the *open new* mode, which allows the user to open an empty new project; the other is *open existing*, which will open an exiting project from the location the users chooses.

5.1.1 Open existing object

Users can choose to open an existing project from any disc attached to their local machine or network environment. This project would be a specific type of file or XML type file, which contains information about the course structure. Once it is opened, the application will present the information as a tree structure, in a similar manner as described for the full authoring system. The user can either edit, or browse it (as Figure 5.1 below shows). When an existing project is opened, the tree display on the left hand panel shows the root node of that project, which is the node representing the course that this project represents. Users can add or delete nodes, select node and fill in the required information to the various meta-data fields. The OK button is used to confirm the information entered into a node and it updates the internal tree data structure, and then after editing, users can save project by selecting the save button. Also users can make changes to the currently selected node, and then save those changes to the database.

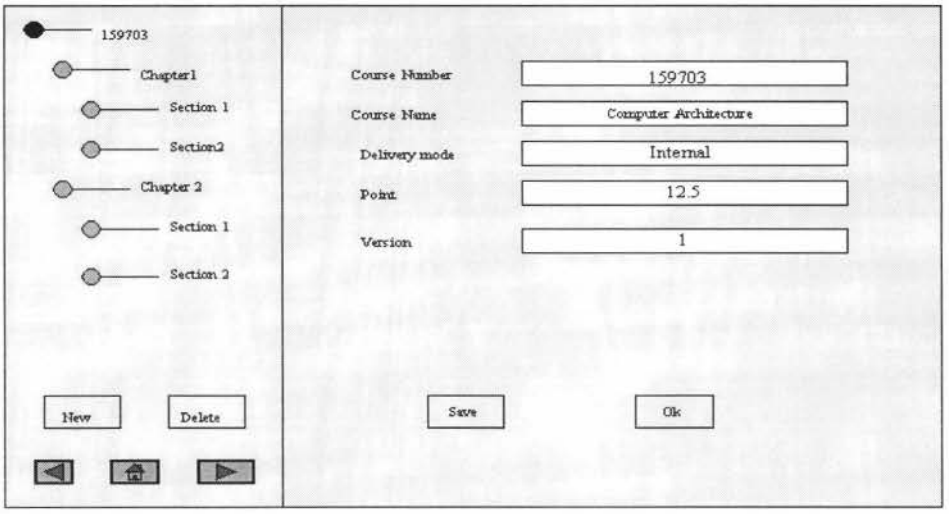

Figure 5.1 Opening an existing project, the current selected node is called "159703". On the right side panel, showing the node information.

Figure 5.2 shows that the currently selected node is *section 2*, which is a sub node of node *chapter 1*. The information panel also show the selected node's *type*, *sequence*, and *prerequisites* and *key words*. When the node's *type* is set to bottom, then the object panel will become activated and the user can enter the object's *number*, *name*, and the *location*, which is where is the object is located on the local machine. The latter can also be a URL rather than a local address. The *time created* is the time when this object has been first created. After some or all of this information has been entered in the specified fields of the node being edited, the user can choose either save it or to click the OK button. The OK button and save button do different jobs. The OK button is for user to update the node information, for example the user might like to change the name of node 1 from "chapter 1" to "chapter 2", so after this change has been made the user has to click the OK button in order to update the information. It is not being saved to any file, it is just changing the information on the screen (and also in the internal data-structures), so that the user knows it has been changed. Changing from one node to another using the tree panel also updates the information for the node previously displayed, otherwise that information would be lost.

In order to save this change permanently, the user has to click the save button when they have finished editing.

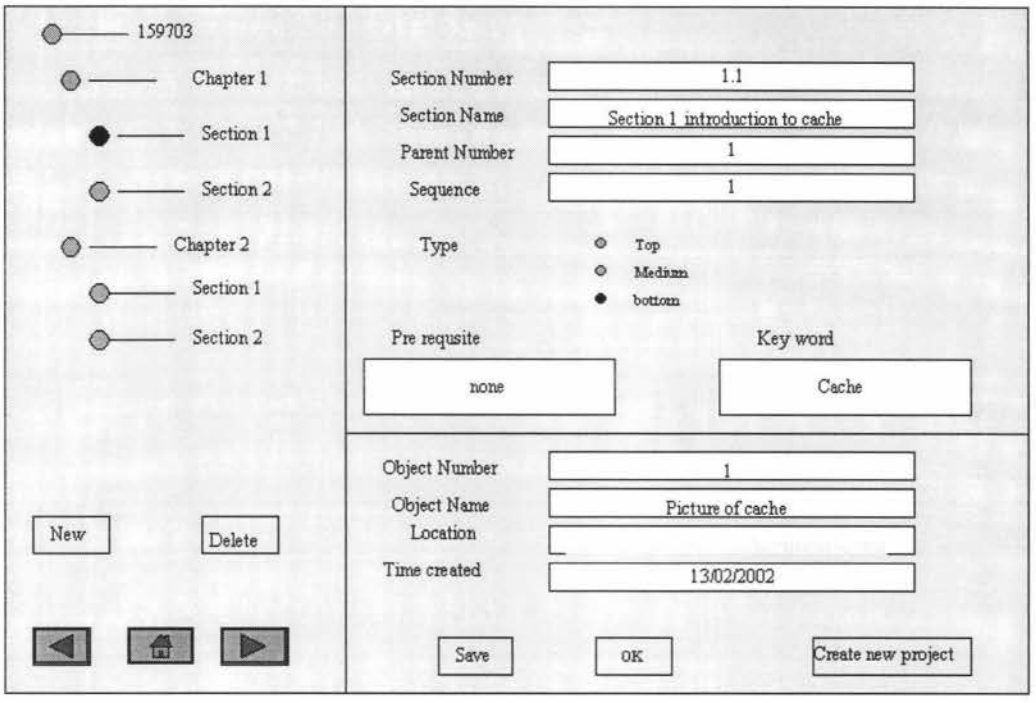

Figure 5.2 showing current selected node is a sub node of node "chapter 1". Also showing the corresponding information about current selected node. Alsoshowing the object node information that is corresponding to section 1.

5.1.2 Creating a new object

When a user chooses to create a new project, the application will require the user to fill out all the information for a new project, and the application will then guide the user to finish building a new course structure step by step.

The principle is that every new project will generate one new XML document on the local machine. The project is named by the user. Users can either create entire course structure, or a part of the course structure. For an incomplete course structure, a user can complete it later, but this must be done before publishing it. (Users can open this incomplete course structure at a later time and then edit it to complete the course structure).

Figure 5.3 is a screen shot of the implemented application, which shows that a user has created a new empty project. Currently there are no nodes that have been created. The interface starts with e-course information panel. This screen shot shows part of what has actually been implemented in this project as a Java application on the Macintosh.

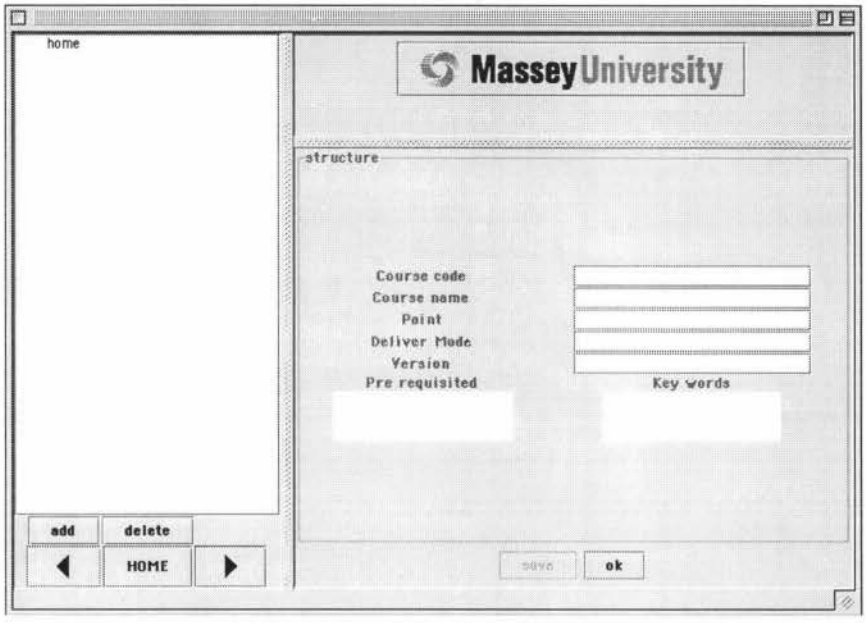

Figure 5.3 A screen shot of the intermediate application when creating a new empty project.

Figure 5.4 shows a further screen shot after user has created 4 new nodes under the home node, which are 159703, chapter 1, chapter 2 and chapter 3. The currently selected node is 159703. Users can edit the information about node 159703 in the information panel on the right hand side. Updating the node information by clicking the OK button. After final editing, the project file can be saved by clicking the save button.

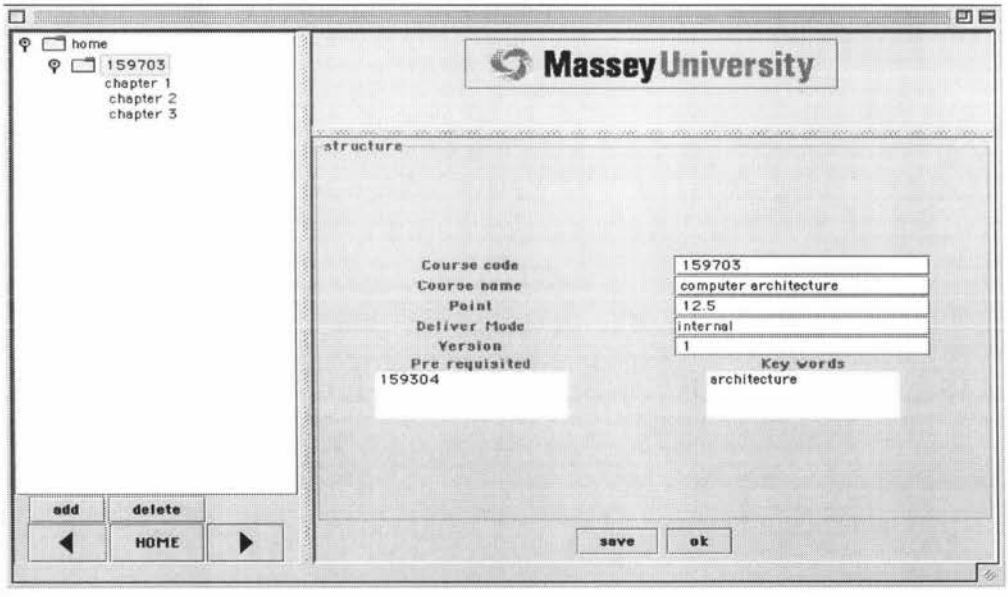

Figure 5.4 shows a screen shot of the intermediate application in which 4 nodes have been created in this project.

5.2. Intermediate system XML database design and implementation

5.2.1 Creating and publishing a course as an XML document

After a user has finished creating a new course structure, the Intermediate system will create an XML file to store information about the course. Because XML allows a hierarchical internal structure, so we can store all the required information in one XML document. It will contain course information, section information, and object information, lectures' information as well as all of the other metadata required. Below is an example of an XML document that has been generated by this project. It just shows the basic structure.

```
<course...>
    <section section 1...
       <Prerequisite>
       </Prerequisite>
       <keyword>
       </keyword>
        etc...

    </section>
    <section section 2...
       <Prerequisite>
       </Prerequisite>
       <keyword>
       </keyword>
       <Object...
           <objectID...
           </objectIDd>
           <objectName...
           </objectName>
           < URL...
           </URL>
       </Object>
       etc...

    </section>
    <keywords...
    </keywords>
    <Prerequisite...
    </Prerequisite>
    <Lecture...
    </Lecture>
```

```
        etc...
    </course>
```

This XML file stores all the information about the course. It will be sent to the server along with all of the learning objects that are not globally accessible URLs, when the user decides to publish the course. When publishing the course the application will have to create a repository for the local learning objects on the web server and upload them from the user's local discs. It will also have to modify the URLs of these learning objects in the XML document uploaded, in order for the published version to refer to this repository instead of the local disc. Universal URLs (e.g. http://www.universal.edu/) do not have to be changed as these are references to external learning objects that already an address to be served from.

The course will be browsed by the student using a modified version of the TILE client. The user will download a modified version of the TILE applet to display the course structure in a frame on the left hand side of the browser window, which is very similar to the authoring display in this application. The applet will load and display the learning objects in the main display on the right hand side. The uploaded XML file will be read by this applet to create the tree structure display and to implement the browse and search functions [67]. Figure 5.5 shows the current TILE applet. The modified applet would be identical except it would not have the functions related to the central server, such as Go-online, login etc. It would also omit the guide me tab in the applet display.

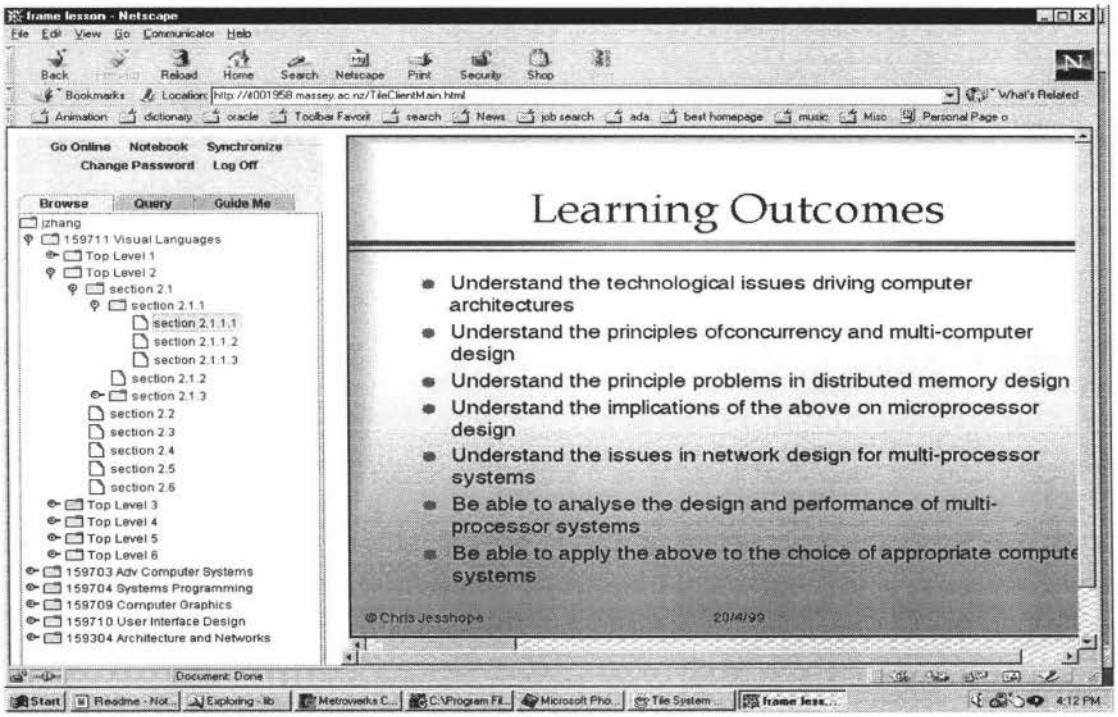

Figure 5.5. The TILE client applet, showing how the user wouldsee the course in a browser window.

5.2.2Update XML document

The publishing mechanism uploads the XML database and all of the learning objects it references to the server. The published course will not change until the user updates the server course by submitting or publishing a modified version of it. We include an ftp server in the authoring tool in order to update the course without having to exit to another application. In this way the user can edit the course structure, or the learning objects and then, when it is ready, simply publish it again by, which will ftp the new structure to the server again. The application will also create a list of files that indicate which files need to be changed on the server, which files need to be deleted and which do not need to change.

5.2.3Deleting XML document

Deleting an XML documents locally only happens when a user wants to delete whole courses. If the user deletes a course, then we have to consider whether all information relative to this course should also be deleted. There is a possibility that learning objects may be shared between different courses however, and so it is not safe to delete these. Therefore only the XML file will be deleted and any local learning object will be left for the user to delete as required. The user will be asked if they wish to update server-side course information, in which case the ftp client will also clean up the server. In this case the XML file and all of the learning objects in its repository will be deleted as it is assumed that these are not shared between courses.

5.3. Intermediate system function design and implementation

5.3.1 Browse

Figure 5.6 shows a screen shot, in which a user is browsing the course structure. The left panel shows the course structure as a tree, and the right side panel shows the information for any selected node in the tree. Users browse the course structure by unfolding the structure as appropriate and clicking on a tree node to view the metadata associated with that node. Each node in the tree structure represents a section or chapter of a course. When viewing the general view of a node, a user can decide whether to edit it or leave it as it is. Updated nodes are changed with the OK button or by selecting another node.

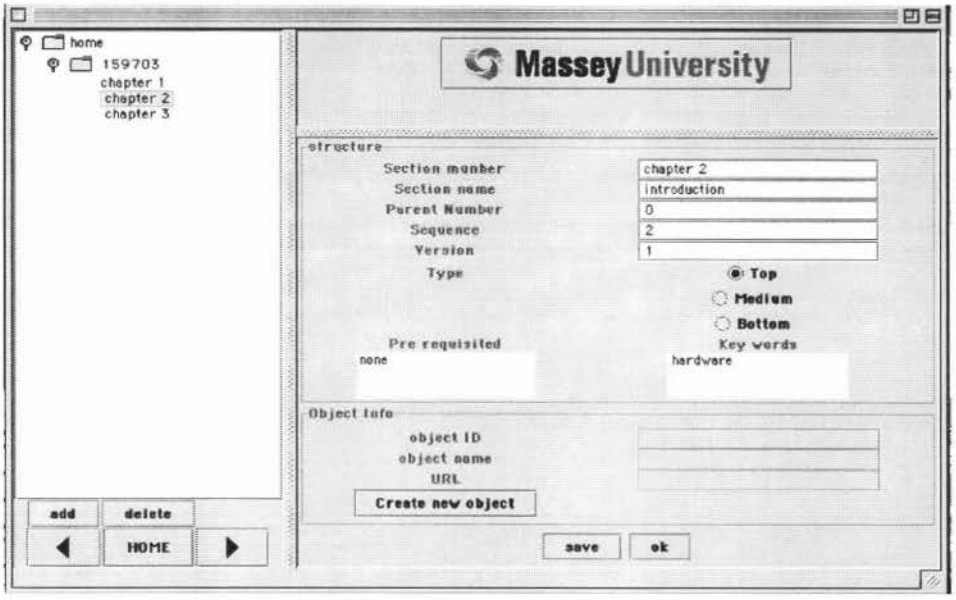

Figure 5.6 A user can browse or edit the course information. The currently selected node is chapter 2. The right-hand panel shows the information for node "chapter 2"

5.3.2 Creating new structure

Figure 5.7 is a screen shot, which shows the first panel that a user will see when they start using the application. This panel contains the main menus for the application. When a user chooses to create a new project with the application, they can select the file drop-down menu where there are various options, to create load or save a project.

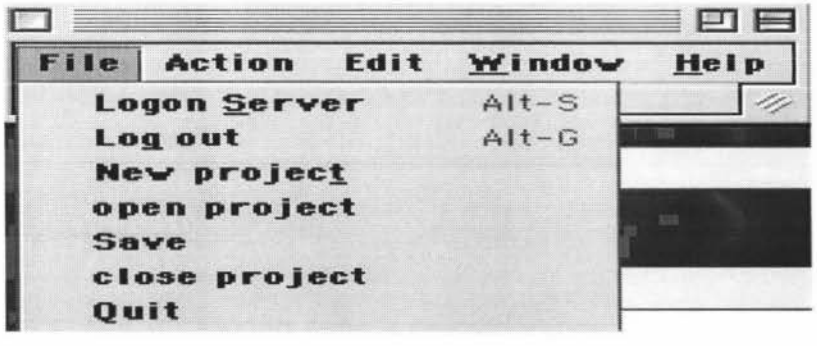

Figure 5.7 The application starts with this menu panel, which contains dropdown menus. The file menu can be used to create a new project, open existing project, etc.

Once the users selects the "New project" option. The system will show an empty project file as shown below in figure 5.8. This information can be accessed at any time by selecting the home node in the tree structure.

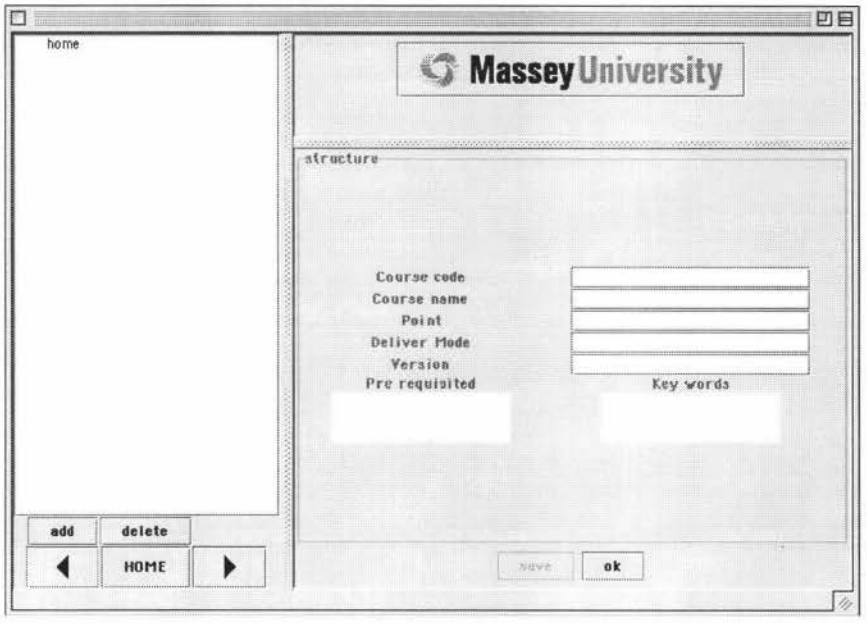

Figure 5.8 Create an empty project file

A user can use the "add" and "delete" buttons to add and delete tree nodes. For each node that the user adds to the tree structure, the information could be filled in the right hand of panel, which is information panel. The save button will not function until nodes have been added in the left hand plane. Figure 5.9 (see below) shows four nodes that have been added in the tree structure. They are node "*159703*", "*chapter 1*", "*chapter 2*" and "*chapter 3*". In this figure the user has filled out the information on the right hand panel. All we have to do is press the "ok" button or select another node in order to update the information internally after filling out the information for each node, and then after everything has been completed, the "save" button can be pressed in order to save the project to local disc. The Application will then generate an XML document for this newly created project. Although we didn't give the snap shots here to describe how to produce learning object, users can easy find there is a button down at the right hand panel. When users select the section node type is bottom, the learning object panel will become active, from this panel, users can launch any multimedia tool (e.g. Microsoft Power Point, AudioGraph, Photoshop, etc...) to produce learning object. The advantage with this function is that Authoring tool doesn't limited the type of multimedia tools, as long as users have such tool in their computer, they can use them to produce learning object.

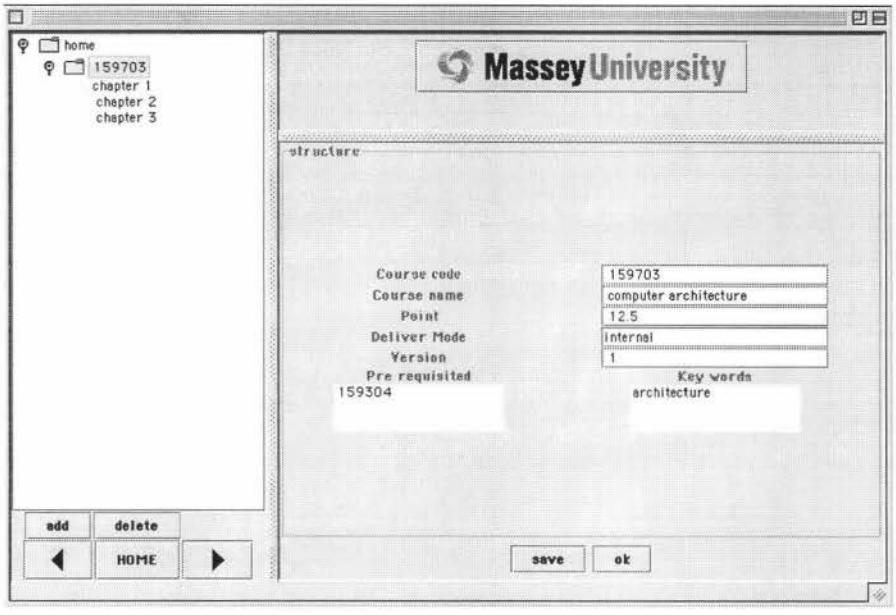

Figure 5.9 Shows a user has created a new course project and new course structure for course 159703. All information about this course can be browsed in the information panel on the right side.

Also we need a publishing item in the file menu in order to invoke the FTP function to upload the course project files to server. This function needs the user to set up connection information to server. This information should be stored as preferences for the user. The information is required to login in to the server. Then, after validating the username and password and the connection has been established, the information concerning a particular project will be FTPed to the server to be browsed by the students. The URLs that are necessary for browsing on the server will need to be remapped during this process. We will discuss this in section 5.3.5 publishing online. It would also be useful to have a preview function in the application to preview the course in the web browser locally.

5.3.3 Editing structure

Users can edit the structure for a course after they have created it. Editing includes changing the node information, adding new nodes to structure, deleting nodes from the tree structure, changing the tree structure by moving a node's position, etc. Nodes can also be copied from one project to another. Copying a node from one presentation and pasting it into another will result in the copied node and all nodes below it being copied into the new project. This is known as a deep copy.

If a user chooses to close the window and there are still some changes that have been made to the project, the system will alert users by showing them the warning window in fingure5.10 (see below). "Cancel" will bring users back to project window, so that they can save the changes. "Quit" will close all windows and it will cause losing any unsaved data.

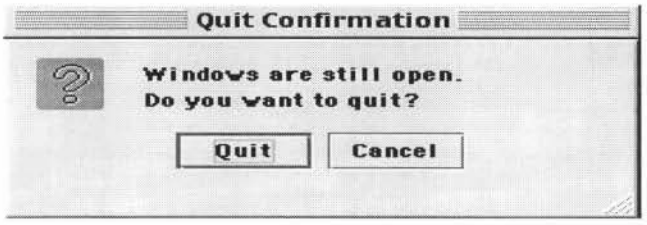

Figure 5.10 Screen shot showing the alert giving the user the option whether to quit or not.

5.3.4 Drag and Drop

In same cases it is desirable to implement functions with drag and drop for ease of use by the author. Two examples are: when copying material from one open project window to another or when specifying prerequisites within a single project. In the later case a tree node from the display on the left hand side of a project window is dragged into the prerequisites window for another node. This saves typing in the node name, possibly making a typing mistake in doing this. In order to accomplish drag and drop functionality under in JDK 1.1.8, it is necessary to download the Java Bean suite that supports drag and drop, as drag and drop functionality is not a standard supported function for JDK 1.1. This Java bean suite is rather large, like other Java packages it provides a number of beans that can be used, in this case there are two beans, one is a drag bean and the other is a drop bean. What we have to do is to use these beans and their methods. The following code show how the drag and drop beans have been used in this project.

```
/********* import the library *********/
import com.ibm.dnd.*;
import com.ibm.dnd.DragListener;
import com.ibm.dnd.DragEvent;
import com.ibm.dnd.DragBean;
import com.ibm.dnd.DDUtilities;
import com.ibm.dnd.DropListener;
import com.ibm.dnd.DropEvent;


/****** Add drag and drop listeners *******/
myDragBean.addDragListener(new MyDragListener());
myDropBean.addDropListener( new MyDropListener());
myDropBean.setComponent(prerequisiteArea);
myDragBean.setComponent(tree);
myDropBean.setComponent(tree);


/***** Class implement Drag Listener functions ******/
class MyDragListener implements DragListener {
```

```
    public void dragEnter(com.ibm.dnd.DragEvent event) {
         System.out.println("drag eneter");
         }
    public void dragOver(com.ibm.dnd.DragEvent event){
          System.out.println("drag over");
         }
    public void dragExit(com.ibm.dnd.DragEvent event ) {
         System.out.println("drag exit");
         }
  public void dragDropEndFailed(com.ibm.dnd.DragEvent  event){
         System.out.println("drag failed");
         }
     public void dragDropEndOk(com.ibm.dnd.DragEvent event){
          System.out.println("drag ok");
         }
         public void dragStart(com.ibm.dnd.DragEvent event){
           TreePath currentSelection =
                                tree.getSelectionPath();
           Object dragItem=null;
           if (currentSelection != null) {
           DefaultMutableTreeNode dragNode =
  (DefaultMutableTreeNode)tree.getLastSelectedPathComponent();
           dragItem = dragNode.getUserObject();
           }
          myDragBean.setInputObject(dragItem);
          System.out.println("drag start");
         }
       }


/********** Class implement Drop Listener functions *********/
class MyDropListener implements DropListener {
         public void dropEnd(com.ibm.dnd.DropEvent event){
            prerequisiteArea.setText(String.valueOf
                          (myDropBean.getOutputDropObject()));
            System.out.println(event.getSource());
            System.out.println("drop end "+event.getSource());
         }
         public void dragOver(com.ibm.dnd.DropEvent event){
            System.out.println("drop over");
            }
         public void dragEnter(com.ibm.dnd.DropEvent event){
              //prerequisiteArea.setText(String.valueOf
              // (myDropBean.getOutputDropObject()));

               System.out.println("drop eneter");
         }
         public void dragExit(com.ibm.dnd.DropEvent event){
               System.out.println("drop exit");
```

```
        }
    }
```

5.3.5 Publishing on-line

When a user has finished the course construction and decide to publish it to students, or whoever, they will use the publish option from the file menu. The intermediate system uses a standard web server that is doing the hosting work. Following are the various stages required to publish a course to the server.

- First the user has to supply information about the server and where on the server the course should be put, e.g. ftp address, repository directory for the XML, files, learning objects, java applet etc. The repository is a folder somewhere on the web server's that can be referenced by the XML. For example, when using an apache server, then we can create a folder, say *object-files*, somewhere in apache's *htdocs* directory, e.g. *...apache/htdocs/object-files/*
- We can now upload the raw media files that are referenced in the local XML file to this repository. However, it must be noted that the references in the XML file will also have to be updated in the version of the XML file that is uploaded to the server. The local XML file contains a local address for the media but the URL in the XML file on the server will need to contain the location where the media has been uploaded to, e.g.: *http://www.my.server/object-files /raw-media.html*
- The application creates and uploads the modified XML file, which will correspond to the course structure as created but with server-side rather than local references.
- Next the application must upload client applet, which will be used by the students to read and display the XML file.
- Finally the application must upload the home page for the course, which contains a reference to the java applet for browsing and an introduction to the course (this can be derived from the information the user sets when creating the course, e.g. course number, title, points value, course controller and email address etc.

Publishing function in this project has not been implemented yet, so it needs more work on it and also it needs to consider the security issue about data transfer across the Internet.

5.4. System environment set up

5.4.1 JDK 1.1.8

The TILE authoring system application has been written on the Macintosh computer, the operating system used is MacOS 9.0. MRJ 2.2.3 [81] is installed on the Mac OS9.0, which supports JDK1.1.8. The Development Environment that has been used to develop this project is CodeWarrior 5.0. On the Macintosh, this gives MRJ a fixed CLASSPATH, which is located under the System Folder. The path is:

```
...:Systems Folder:Extensions:MRJ Libraries:MRJClasses
```

There is nothing else we need to change in order to use this environment, however all Java archive or JAR files should be placed in the CLASSPATH location given above.

In order to implement this project a number of software components had to be downloaded and installed onto the local machine, these are described in detail in the sections below.

5.4.2 Swing Installation

For the user interface components in this project, we have used JFC1.1 but have augmented this with Java Swing version 1.1.1, which can be downloaded from Web site http://java.sun.com/products/jfc/download.archive.html#1.1.1 free of charge. After downloading and installing Swing 1.1.1, what we need to do is add an entry to the project CLASSPATH, telling the project where to look for the Swingall.jar file. On the Macintosh, under the CodeWarrior environment, we simple drag the Swingall.jar file to the CLASSPATH folder.

5.4.3 InstantDB Installation

InstantDB has been used for early work on the full authoring system design. It has been installed on the local machine. The InstanDB installation is very easy to follow. After downloading the zip file, unzip it, you will have following folders:

Classes: Holds the jar files containing the database classes
Examples: Holds various example files
Examples/SQLBuilder: Holds a JFC based database exploration tool.
Examples/Win98: Holds a start up script for Windows 98 users.
Examples/Linux: Holds a start up script for Linux users.
Functions: Holds the source code for InstantDB's SQL functions
Docs: Holds InstantDB documentation.

These should be put into the newly created folder, which can be in anywhere on the system you want it to be. It is not necessary to put these files into MRJClasses path in Mac. After installation, simply add the Classes/idb.jar, Classes/idbexmpl.jar and Classes/jta-spec1_0_1.jar files to the project source path by dragging these files to your project.

The version we used in this project is InstantDB 3.26, which can be downloaded from web site http://instantdb.tripod.com/old-site/index-9.html free of charge.

5.4.3 XML Parser Installation

The Xerces Java XML Parser is also installed in the project to support the processing of XML files. The Xerces-J-bin1.2.3.zip file can be downloaded freely from the web site at http://xml.apache.org/dist/xerces-j/old_xerces1/. After you have downloaded it, unzip the file to a newly create folder (wherever you want to create it), you will get following files in your newly created folder:

License: License for Xerces-J
Readme.html: Web page redirect to docs/html/index.html
xerces.jar: Jar file containing all the parser class files
xercesSamples.jar: Jar file containing all sample class files
data/ : Directory containing sample XML data files
docs/html/ : Directory containing documentation
docs/html/apiDocs/ : Directory containing Javadoc API for parser framework

Again you will need to add the Xerces.jar file into the project source path. With the CodeWarrior development environment on Mac, we just drag the Xerces.jar to the source path.

5.4.4 Darg and drop Java Bean installation

The drag and Drop function in this project uses the Java Bean suit, which is developed by alphaWorks [69]. The purpose of the Drag and Drop Bean Suite is to allow you to use the Drag and Drop mechanism both inside of a single Frame or between Frames of an application. This is necessary because the Mac doesn't support JDK 1.2, which first introduced drag and drop functionality. Therefore JDK1.1.8 used on MacOS 9.0 doesn't support drag and drop functionality, it has to be added with the Java bean implementation. We have therefore downloaded this bean suite to add drag and drop functionality for JDK1.1.8. The Drag and Drop Bean Suite has been written in 100% Pure Java, and it contains two bean suite, one is the drag bean and the other is the drop bean. There are two jar files and again both have to be added into the project source path. These are DND.jar and DND_Runtime.jar.

Chapter 6 Results and future development

The full TILE Authoring system is a large and complex system that will take a long time to complete. Also during the course of this project, the TILE course delivery architecture, its schema and functionality were still being developed. It was for this reason that a subset of the original specification was implemented as a prototype.

This thesis therefore studies in detail the design of this authoring system. However, in considering the prototype implementation only a subset of the full authoring system has been considered, we have called this the Intermediate Authoring system. The full system is a new web-based course authoring system for the Technology Integrated Learning Environment (TILE) project. Individuals can author and publish courses just using the intermediate system, a conventional server and a modified version of the TILE delivery applet.
The database model in this project has been fixed and is based on the Massey University course structure. The prototype implementation has prototyped a basic interface and all local functions for the authoring system. It has been implemented as a stand-alone application in Java and it is quite flexible. Although it has been created on the Macintosh, it can be installed on any system.

This project achieved some substantial results. A full intermediate authoring system has been implemented that allows the user to create new projects, load and store projects to a local XML file and add, edit nodes and their metadata quite intuitively. In implementing this system we have had to solve a number of technical problems during its development. The first of these was the conversion of the TILE database specification to use an XML document. This replaces the database and JDBC access to it that was used in the TILE framework application. We have also considered the human interface design and have decided to implement drag and drop functionality for ease of use, despite it not being supported in JDK1.1.8 on the Mac. In this project we have decided to use the lowest common denominator in order to gain cross-platform functionality. Also the XML parser used in this project is chosen from numbers of Parsers. Xerces Parser is XML parser for Java, and installation and implementation on Macintosh is quite successful. Although most of the local functionality has been completed the components that deal with publishing have only been designed at a conceptual level. Details of the work still to be undertaken to complete this project are given below.

Work was also completed on the full authoring system before switching to the intermediate system implementation. A database has been implemented using InstantDB and nodes in the authoring application were linked to this database using the TILE schema. The communication between the application and the databases was implemented using JDBC. This interface was later changed in the intermediate system, when we switched to an XML document that replaced the database. The functions are the same however. It should also be noted that in the full authoring application both interfaces are required, as we have decided that access to the local database would retain the XML

interface. The reason for this is that the TILE server will communicate with the authoring system using XML wrapped up in an http protocol. This is to allow it to pass through firewalls.

There are a number of issues that still have to be considered in using both the intermediate and the full authoring systems. Since the aim of designing and implementing this authoring system tool is to use it in various educational organization and different situations, the database model will need to be more flexible and generic in order for it to be used with different organizations with different purposes. This is one of the fundamental problems in tool and course design and is being considered by such projects as the educational modelling language, EML (http://eml.ou.nl/), which is developing conceptual models and XML bindings for describing the various actors and objects in educational delivery.

At present all of the system program functionality relies on the database schema. Any change of the database schema means changing the application program. Ideally this should be implemented more flexibly. Even in the later version after we modified the program to use an XML document instead of an SQL databases, we still needed to base the application on the database schema. To implement it more flexibly we need to derive the generic objects and methods and to customise these with the vocabulary used by a given institution. For example this authoring tool uses courses and sections. In EML these are all *learning units* and any organization will have its own vocabulary for a learning unit. The functionality will not be changed, just the mapping of the concept onto the local structure and naming convention.

Further coding of the fundamental functions of the authoring system also be completed. Until now, only the browsing of the course structure, creation of new projects and the opening and editing of existing projects has been implemented. Further functions have still to be implemented.

In the intermediate system, the only functions remaining to be implemented are the functions required to publish the course. This requires implementing an ftp server in the application. This however is very easy in Java, as Java has built-in support for networking and has a number of protocols, such as ftp available as standard. It would be desirable to have parameters concerning the locations of the published material to be stored in the project XML file. Default parameters could also be entered in a user preferences panel, because it is likely that all courses will be published to the same web server. Only the directories of the course repository will need to be customised for a project.

Finally to complete the TILE intermediate authoring system the TILE client applet would need to be modified. Modifications would be to remove all functions that imply access to the main server, such as login, go-online, guide me, etc. (see Figure 5.5). It would also be desirable to be able to define and customise the look and feel of both the home page for the course and the interface of the Java applet, so that an institution or even a department or lecturer can provide a personalised look to the on-line course.

Conclusion

Online education is a hot topic nowadays. Over the last five or so years there has been an increasing interest in this form of learning because of the potential for a low-cost education. There are many tools that have been developed for achieving good distance eduction results. These are mostly virtual learning environments (VLEs), which combine both rudimentary authoring and delivery capabilities. They do not however, consider the issues of reuse and flexible delivery. One of the main problems is that many different tools are required to create on-line courses, these include web editors, multimedia authoring tools, etc. and each does a different job. One tool does one aspect of the job but none of them does all of the work required to produce rich, interactive, on-line educational material. The aim has been to reproduce the results of, or even better, improve on what is achieved in face-to-face education. Online education involves many issues for the lecturer and student and what we want to archive is to develop a system that is more flexible, more stable, more adaptable, and more reusable.

In this thesis, we have analysed and designed a new web-based course authoring system. The authoring system creates the course structure and allows the author to annotate that with metadata and prerequisites. It combines this structure with the course material, which can be any raw media, such as text, graphics, video or any other multimedia files. These are either referenced on the web, as general information, or are produced by the author using other tools and stored on the local machine. Using this system, the combined structure and media is then published on-line in a very flexible manner, allowing the student to browse structure and search metadata in order to find their way around the material.

This work fits into a much larger project called the Technology Integrated Learning Environments (TILE) project, which is funded by the New Economy Research fund. The work outlined in this thesis provides two partial implementations (one almost complete) of the authoring tool, whose requirements are given above. The two authoring tools address different issues. The first is an authoring tool to add functionality to the TILE system, which is an institution-wide, managed learning environment (MLE). The second and almost complete one, provides a stand alone tool that can be used in conjunction with a modified version of the TILE delivery applet, in conjunction with a standard web server to deliver educational material on single courses. These tools use respectively, a relational database and an XML document to store the course structure and references to the media. The common functionality of these authoring systems is all about the browsing and editing the course structure, enabling queries to be made, creating new projects and opening existing projects, etc. Both systems will eventually provide functionality such as publishing online, logging in and logging out, adaptation in various online education situations and the integration of media produced by various multimedia tools. In this sense they are open systems.

Although there are many existing web-based, educational authoring tools, tools with the flexibility and functionality of those specified and prototyped here, are quite few. The functionality of the full authoring system requires authentication with a central server, which is provided by TILE system itself. A central server is also required to provide a repository of authored material, ready for publication or reuse. This is held in the database system, which stores all of course information and sends it to the client side as required. The functionality of the intermediate system is as a stand-alone system, which doesn't need to connect to a server besides publishing the course online.

In the implementation work, Java is chosen to be the programming language for this system because of its cross-platform compatibility. We also choose to use JDK 1.1.8 because it is the only Java version that can really be supported by any platform, especially the Macintosh Operating system.

Based on all the above analysis, a TILE authoring interface system has been designed. Other related techniques have also been studied in this thesis. A cross platform client-side database has been implemented. Methods to add the Java Swing package from Java SDK 2 to JDK1.1.8 have been found, in order to enhance interface performance. XML has been studied as a standardised means of encapsulating course content. An XML document replaces the relational database in the intermediate system and XML is used to communicate structured data between the client and the server in the full system. The whole structured database is replaced by a single XML file in the intermediate system, and this represents the course structure and is uploaded and delivered to the students by a standard web server and a modified TILE applet. The applet extracts the course structure displaying it to the student dynamically. Leaf nodes in this structure correspond to the media files and are displayed using a standard web browser's functionality.

In this thesis, the TILE authoring system and intermediate system functionality have been specified and a user interface has been designed and demonstrated. Although the demonstration is not fully functional, a complete functional design for the whole system has been given. The functional design and the partial implementation have confirmed the feasibility of this approach, for both the intermediate and full TILE authoring system. All in-principle problems have been solved.

The thesis also presents background research, both in the area of e-education as well as practical issues such as setting up a large system based on reusable components. Important information is given concerning the setting up the system environments. All the tests and their results concerning the system interface design have been discussed in detail.

The status of the work on completion of the thesis, is that the full system was evaluated and partially completed and the intermediate system application with complementary functionality has been developed to a stage where complete course structures can be entered and made persistent. This includes all metadata specified in the TILE database schemas, such as keywords, authors, prerequisites, etc. The course structure can be

created, edited and stored on the local computer. The only functionality not complete in the intermediate system is the publishing of the course data on-line. There was not sufficient time to complete this, however, we have given a full specification of what is required to complete it, which involves two tasks. The implementation of an ftp client within the authoring application, which updates a transformed XML database to the server, along with all of the raw media objects and the delivery applet. The other task is the modification of the TILE applet to access the course structure from this XML database. This will use a standard web server instead of the TILE servelet components.

In summary, the work undertaken in this thesis has defined and implemented an authoring tool for structuring and annotating on-line educational courses. All technical problems encountered during the system architecture design have been solved. Work is still continuing on this system in the TILE project.

References

[1] NZEdSoft, (2001) Tile Home page,
http://www-tile.massey.ac.nz/, Cited 12/6/01.

[2] NZEdSoft (2001) AudioGraph home page,
http://www.nzedsoft.com/audiographhomepa.html, Cited 12/6/01.

[3] James Gosling, Bill Joy Guy Steele, Gilad Bracha. "The Java Language Specification Second Edition"
http://java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#2219
Cited11/6/2001

[4]"Java Language Overview"
http://java.sun.com/docs/overviews/java/java-overview-1.htm Cited 11/6/2001

[5] "XML Version 3.0 Purpose"
http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/xmlsdk/xmls6g53.htm
Cited 11/6/2001

[6]"*Introduction to Structured Query Language Version 4.71*"
http://w3.one.net/~jhoffman/sqltut.htm Cited 11/6/2001

[7] "*JDBC 101: How to connect to an SQL database with JDBC*"
http://www.devdaily.com/java/edu/pj/pj010024/ Cited 11/6/2001

[8] Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. "*The practical SQL Handbook Using Structured Query Language*" Third Edition ISBN 0-201-44787-8

[9] Neil Bradley " *The XML companion*" Second Edition
ISBN 0-201-67486-6

[10] Virginia Steiner, DLRN Research Associate (10/10/95) "*What is Distance Education*?"
http://www.dlrn.org/library/dl/whatis.html, Cited 15/6/2001

[11] Regina Gehne, Chris Jesshope, and Zhenzi (Jenny) Zhang "*TECHNOLOGY INTEGRATED LEARNING ENVIRONMENT - A WEB_BASED DISTANCE LEARNING SYSTEM*" Accepted by IMSA 2001, Hawaii, USA.

[12] Senator Bob Kerrey, Representative Johnny Isakson "*The Power of the Internet for Learning: MOVING FROM PROMISE TO PRACTICE*" Final Report of Web-Based Education Commission (DECEMBER 2000)
http://www.distance-educator.com/de_ezine/index3a011501.html, Cited 20/6/2001

[13] "*What is Distance Education? Defining the Concepts and Terms Which Have Characterized the Field*"
http://www.distance-educator.com/index1a101600.phtml Cited 20/6/2001

[14] "*Distance Education: Foundations and Fundamental Concepts*"
http://www.distance-educator.com/de_ezine/article.php?sid=103
Cited 20/6/2001

[15] Diana G. Oblinger "*The Nature and Purpose of Distance Education*"
http://horizon.unc.edu/TS/default.asp?show=article&id=647, Cited 26/06/2001

[16] Mark Minasi, Todd Lammle with Monica Lammle. ISBN 0-7821-2123-3 "*Mastering TCP/IP for NT server*"

[17] Christopher D. King "*The Quest for Cyber- school: The Challenge of Designing Effective, Web-Based Instructional Delivery Systems*"
http://personal.bellsouth.net/mia/c/d/cdk6164/WBDistED.html, Cited 24/06/2001

[18] Jerry Fitzgerald, Alan Dennis. "*Business data communications and networking*"5th Edition ISBN 0-471-12365-x

[19] Charlote N. "Lami" Gunawardena, PhD. Associated Professor of Distance Education and Instructional Technology, College of Education, University of New Mexico, USA "*Designing and Evaluating Web-based Distance education courses*"
http://www.lite.fae.unicamp.br/educdist/sld001.htm, Cited 25/06/2001

[20] Laurie Harrison, University of Toronto, Canada, Katharyn Foster, University of Toronto, Canada "*Accessible Web-based Distance Education: Principles and Best Practices*"
http://naweb.unb.ca/proceedings/1999/harrison/harrison.html, Cited 27/06/2001

[21] "*Web based Learning Primer*" by Tony Mark
http://www.ctt.bc.ca/landonline/primer.html, Cited 27/06/2001

[22] Elizabeth J. Gibson, Patrick W. Brewer, Ajay Dholakia, Mladen A. Vouk, Donald L. Bitzer Dept. of Computer Science, North Carolina State University, Raleigh, NC 27695-8206. "*A Comparative Analysis of Web-Based Testing and Evaluation Systems*"
http://renoir.csc.ncsu.edu/MRA/Reports/WebBasedTesting.html, Cited 28/06/2001

[23] Ann E. Barron *University of South Florida,* Chet Lyskawa University of South Florida "*A Review of Tools for Developing and Managing Online Courses*"
http://www.coe.uh.edu/insite/elec_pub/HTML1998/de_barr.htm, Cited 28/06/2001

[24] "William K.Bradford Publishing Company" home page http://www.wkbradford.com/teacher.htm, Cited 30/06/2001

[25] "SOFTWARE REVIEW" http://204.98.1.2/area_teams/software/sitectrl.htm, Cited 30/06/2001

[26] "*Guide #1Distance Education: An Overview* " by Barry Willis http://www.uidaho.edu/evo/dist1.html, Cited 03/07/2001

[27] "*Guide #11Distance Education and the WWW*" http://www.uidaho.edu/evo/dist11.html, Cited 04/07/2001

[28] Karla Embleton Educational Technology Brown Bag Series 9/30/99 "*Web-based Education Overview*" http://www.fcs.iastate.edu/computer/tips/weboverview.html, Cited 05/07/2001

[29] By Elizabeth J. Gibson (email: ejgibson@unity.ncsu.edu), Patrick W. Brewe, Ajay Dholakia, Mladen A. Vouk, Donald L. Bitzer Dept. of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 "*A Comparative Analysis of Web-Based Testing and Evaluation Systems*" http://renoir.csc.ncsu.edu/MRA/Reports/WebBasedTesting.html#bib6, Cited 06/07/2001

[30] "*Internet-Based Education: Some Guidelines*" By Hilary McLellan http://tech-head.com/i-ed.htm, Cited 06/07/2001

[31] "*1. A Quick Introduction to XML*" http://java.sun.com/xml/jaxp-1.1/docs/tutorial/overview/1_xml.html, Cited 09/07/2001

[32] "*Concurrent Versions System – the open standard for version control*" http://www.cvshome.org/, Cited 13/07/2001

[33] Charles Graham, Kursat Cagiltay, Joni Craner, Byung-Ro Lim, & Thomas M. Duffy "*Teaching in a Web Based Distance Learning Environment: An Evaluation Summary Based on Four Courses*" CRLT Technical Report No. 13-00. March 1, 2000 http://crlt.indiana.edu/publications/crlt00-13.pdf, Cited 14/07/2001

[34] Farthad Saba, Ph.D. Professior of Education Technology San Diego State University, saba@cts.com "*Distance Education: An Introducation*" http://www.distance-educator.com/portals/research_deintro.html#Quality%20Education%20at%20a%20Distance, Cited 15/07/2001

[35] Chris Jesshope 26/1/01 "*Implication of the Checkout/in development tool on the TILE database*"

[36] "*XML Basic*" represented by W3Schools

http://www.w3schools.com/xml/xml_whatis.asp, Cited 24/07/2001

[37] Editors Philippe Le Hégaret, W3C; Lauren Wood, SoftQuad Software Inc., WG Chair; Jonathan Robie, Texcel (for DOM Level 1) *"What is the Document Object Model?"*
http://www.w3.org/TR/DOM-Level-2-Core/introduction.html, Cited 25/07/2001

[38] *Xerces Java Parser download page*
http://xml.apache.org/dist/. Cited 25/07/2001

[39] Yasser Shohoud *"Use the Simple API for XML - Try SAX for XML document processing—it's faster and uses less memory than DOM"*
http://www.xmlmag.com/upload/free/features/xml/2000/05win00/yy0005/yy0005.asp, Cited 27/07/2001

[40] *"3 SAX: The Simple API for XML"*
http://py-howto.sourceforge.net/xml-howto/SAX.html, Cited 27/07/2001

[41] *"Introduction to DTD"*
http://www.w3schools.com/dtd/dtd_intro.asp, Cited 27/07/2001

[42] *"What is an Event-Based Interface?"*
http://www.megginson.com/SAX/, cited 27/07/2001

[43] *"The java language: an overview"*
http://java.sun.com/docs/overviews/java/java-overview-1.html#FOOTNOTE-1, Cited 30/07/2001

[44] *"Swing: An Overview"*
http://www.apl.jhu.edu/~hall/java/Swing-Tutorial/Swing-Tutorial-Overview.html, Cited 01/08/2001

[45] *"Front End GUI: Java Foundation Classes"*
http://www.depaul.edu/~elliott/shared/projectsarchive/DS513Winter98/green/swingFaq.html, Cited 01/08/2001

[46] *"SQL Tutorial – Introduction"*
http://www.baycongroup.com/pervasive_sql.htm, Cited 09/08/2001

[47] *"About InstantDB "*
http://www.lutris.com/products/projects/instantDB/project/aboutProject/index.html, Cited 10/08/2001

[48] *"1.1.1 What Is MySQL"*
http://www.mysql.com/doc/W/h/What-is.html, Cited10/08/2001

[49] "*JDBC DATA ACCESS APT DRIVERS* - Types of JDBC technology drivers" Available on-line http://java.sun.com/products/jdbc/driverdesc.html, Cited 05/10/2001

[50] Duane K. Fields, "Adding Database Support With JDBC - A Primer for Java Programmers" http://developer.iplanet.com/viewsource/fields_jdbc2/fields_jdbc2.html, Cited 11/08/2001

[51] Selena Sol, November 23, 1998 "*JDBC*" http://www.wdvl.com/Authoring/DB/Intro/jdbc.html, Cited 11/08/2001

[52] Qusay H. Mahmoud "*J D B C - A Persistant Storage for Java Objects*" http://www.javacats.com/US/articles/Qusay/JDBC.html, Cited 11/08/2001

[53] Countless Falls home page "*Database*" http://www.countlessfalls.com/other/databases.htm, Cited 12/08/2001

[54] "*JDBC 101: How to connect to an SQL database with JDBC*" http://www.devdaily.com/java/edu/pj/pj010024/, Cited 12/08/2001

[55] Domenico Ruggeri (CoRiTeL, www.coritel.it), 29-30 October 1998 "TCP/IP SUITE BASICS" http://www.coritel.it/coritel/documents/slides/TCP-IP%201/tsld019.htm, Cited 13/08/2001

[56] Domenico Ruggeri (CoRiTeL, www.coritel.it), 29-30 October 1998 "TCP/IP SUITE BASICS - IP (Internet Protocol) Main Features" http://www.coritel.it/coritel/documents/slides/TCP-IP%201/tsld031.htm, Cited 13/08/2001

[57] Domenico Ruggeri (CoRiTeL, www.coritel.it), 29-30 October 1998 "TCP/IP SUITE BASICS - TCP (Transmission Control Protocol)" http://www.coritel.it/coritel/documents/slides/TCP-IP%201/tsld033.htm, Cited 13/08/2001

[58] *JDKTM 1.1 - AWT Enhancements* http://java.sun.com/products/jdk/1.1/docs/guide/awt/designspec/, Cited 15/08/2001

[59] "*First sips: An Overview of the AWT*" http://www.eng.auburn.edu/~rayh/java/java/AWT.Introduction.html, Cited 15/08/2001

[60] "*Introduction to MySQL and JDBC*" http://www.ils.unc.edu/~lindgren/190/mysql-jdbc/, Cited 15/08/2001

[61] *1.1.6 The Main Features of MySQL*
http://www.mysql.com/doc/F/e/Features.html, Cited 15/08/2001

[62] *" Web Quiz Script:Online Testing Tool"*
http://www.indiawebdevelopers.com/products/quiz.asp, Cited 24/09/2001

[63] "How to master" home page
http://www.howtomaster.com/products/default.asp?displayID=cat&corp=, cited 24/09/2001

[64] James Gosling, Henry McGilton May 1996 *"The Java Language Environment A White Paper"*
http://java.sun.com/docs/white/langenv/, Cited 24/09/2001

[65] These slides are Copyright Jan Newmarch, 1995, 1996, 1997, 1998. Last modified: 10 February, 1998. *"Programming User Interfaces using the AWT (and the JFC)"*
http://jan.netcomp.monash.edu.au/java/swingtut/tut1a.html

[66] 2 Feb 1995 *"Introduction to TCP/IP"*
http://www.yale.edu/pclt/COMM/TCPIP.HTM, cited 09/10/2001

[67] ZhenZi Zhang 2001 Thesis *"A Feasibility Study for the Design of a Web-based Course Delivery System"*

[68] Jenny Zhang and Chris Jesshope (2001) *TILE project database design and Specification*, TILE project report (confidential to the TILE project).

[69] http://alphaworks.ibm.com/alphabeans, cited 26/01/2002

[70] Duncan Lennox (2001), Managing Knowledge with Learning Objects, The Role of an e-Learning Content Management System in Speeding Time to Performance, http://www.internettime.com/itimegroup/lcms/wbt_Mngknw.pdf, cited 06/02/2001

[71] Harvi Singh (2000), Achieving Interoperability in e-Learning, http://www.learningcircuits.org/mar2000/singh.html, cited 06/02/2002

[72] Robin Rover (2002), Shareable Content Object Reference Model Initiative (SCORM), http://xml.coverpages.org/scorm.html, cited 06/02/2002

[73] Kinshuk, Hong H., Albi N., Patel A., Jesshope C. Client-Server Architecture based integrated system for education at a distance, *PEG-2001 Conference*, June 23-26, 2001, Tampere, Finland.

[74] Hong H., Albi N., Kinshuk, He X., Patel A., Jesshope C. Adaptivity in Web-based Educational System. *The Tenth International World Wide Web Conference*, May 1-5, 2001, Hong Kong, China.

[75] C. R. Jesshope (1999) Web-based Teaching - Tools and Experience, *Australian Computer Science Communications*, **21,** (1), pp27-38, ISBN 981-4021-54-7, Proc Australasian Computer Science Conference, ACSC99, Auckland, Jan 1999, (Springer).

[76] C. R. Jesshope (2000) The use of streaming multi-media in microelectronic education, *Microelectronics Education,* Kluwer Academic (London), ISBN 0 7923 6456 2, pp45-48.

[77] C. R. Jesshope (2000) The use of multi-media in internal and extramural teaching, *Proc Lifelong Learning Conference*, Central University of Queensland (Brisbane, Australia), ISBN 187 6674 06 7, pp257-262.

[78] R. Gehne and C. R. Jesshope (2000) Tools for the production of small-footprint, low-bandwidth, streaming multi-media for distance education, *Proc Lifelong Learning Conference*, Central University of Queensland (Brisbane, Australia), ISBN 187 6674 06 7, pp240-244.

[79] C. R. Jesshope (2000) Using AudioGraph in On-line Teaching, *Proc Open Learning Conference*, Brisbane, Australia, pp315-320, Learning Network Queensland (Brisbane, Australia).

[80] C. R. Jesshope and Y. Q. Liu (2001) High Quality Video Delivery over Local Area Networks With Application to Teaching at a Distance, *Intl J. of Electrical Engineering Education (IJEEE),* **Vol 38(1)**, ISSN 0020-7209, pp11-25, Manchester University Press.

[81] MRJ 2.2 download page, http://www.apple.com/java, cited 14/02/2002

Appendix A XML specification

Following is XML document definition for TILE Authoring System

```
<!DOCTYPE database SYSTEM "database.dtd">

<database name="TILE">
<!----------course table ----------->
<table name="course">
        <field nullOrNot="NOT NULL">
            <fieldName> CousrseID </fieldName>
            <fieldDataType>
                <CHAR length="20"/>
            </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">
            <fieldName> CourseName</fieldName>
            <fieldDataType>
                <CHAR varying="yes"  length="20"/>
            </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">
            <fieldName> Version </fieldName>
            <fieldDataType>
                <CHAR length="20"/>
            </fieldDataType>
        </field>
        <field>
            <fieldName> Point </fieldName>
            <fieldDataType>
                <CHAR length="20"/>
            </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">
            <fieldName> DeliveryMode </fieldName>
            <fieldDataType>
                <CHAR length="20"/>
            </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">
            <fieldName> StaffID </fieldName>
            <fieldDataType>
                <CHAR length="20"/>
```

```
        </fieldDataType>
</field>
<field nullOrNot="NOT NULL">
        <fieldName> STaffName </fieldName>
        <fieldDataType>
                <CHAR length="20"/>
        </fieldDataType>
</field>
<!----------------------section table ----------------->
<table name ="section>
        <field nullOrNot="NOT NULL">>
                <fieldName> SectionID</fieldName>
                <fieldDataType>
                    <CHAR varying="yes"  length="20"/>
                </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">>
                <fieldName> SectionName</fieldName>
                <fieldDataType>
                   <CHAR length="20"/>
                </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">>
                <fieldName> ParentSectionID</fieldName>
                <fieldDataType>
                  <CHAR varying="yes"  length="20"/>
                </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">>
                <fieldName> Sequence</fieldName>
                <fieldDataType>
                  <CHAR varying="yes"  length="20"/>
                </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL"> >
                <fieldName> Type</fieldName>
                <fieldDataType>
                  <CHAR varying="yes"  length="20"/>
                </fieldDataType>
        </field>
        <field nullOrNot="NOT NULL">>
                <fieldName> Version</fieldName>
                <fieldDataType>
                  <CHAR varying="yes"  length="20"/>
                </fieldDataType>
```

```
                    </field>
                    <!------------------------ object table ---------------------->
                    <table name =”object”>
                            <field nullOrNot=”NOT NULL”>>
                                    <fieldName> ObjectID</fieldName>
                                    <fieldDataType>
                                     <CHAR length=”20”/>
                                    </fieldDataType>
                            </field>
                            <field nullOrNot=”NOT NULL”>>
                                    <fieldName> ObjectName</fieldName>
                                    <fieldDataType>
                                     <CHAR length=”20”/>
                                    </fieldDataType>
                             </field>
                              <field nullOrNot=”NOT NULL”>>
                                    <fieldName> URL</fieldName>
                                    <fieldDataType>
                                     <CHAR length=”20”/>
                                    </fieldDataType>
                             </field>
                             <field>
                                    <fieldName> CreateTime</fieldName>
                                    <fieldDataType> <DATESTAMP/>
                                    </fieldDataType>
                            </field>
                            <primaryKey>ObjectID</primaryKey>
                    </table>
                    <primaryKey> SectionID</primaryKey>
              </table>
             <primaryKey> CourseID </primaryKey>
             <foreignKey name=”staffID” referenceTable=”staff”
              referencefield=”staffID” nullOrNot=”NOT NULL” />
        </table>
</database>
```

Appendix B DTD Definition

This DTD Definition is referenced from Jenny Zhang's thesis [67], which describes the generalization of TILE XML database schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-this DTD tries to define a format to describe a database schema-->
<!ELEMENT database(table+)>

<!ELEMENT table(field+, primaryKey+, unique*,foreignKey*,check*)>

<!ELEMENT field(fieldName, fieldDataType)>
<!ELEMENT  fieldName (#CDATA)>
<!ELEMENT fieldDataType
(CHAR|NUMERIC|DECIMAL|"INTEGER"|"BIGINT"|"FLOAT'
|"DOUBLE"|"DATE"|"TIME"|"TIMESTAMP")>

<!ELEMENT primaryKey (#CDATA)>
<!ELEMENT unique(#CDATA)>
<!ELEMENT foreignKey(#PCDATA)>
<!ELEMENT check (#CDATA)>

<!ELEMENT CHAR(EMPTY)>
<!ELEMENT  NUMERIC(EMPTY)>
<!ELEMENT  DECIMAL(EMPTY)>

<!ATTRILIST database name CDATA #REQUIRED>
<!ATTRILIST table name CDATA #REQUIRED>

<!ATTLIST CHAR varying "yes" #IMPLIED>
<!ATTLIST CHAR length CDATA #REQUIRED>

<!ATTLIST NUMERIC precision CDATA #REQUIRED>
<!ATTLIST NUMERIC scale CDATA  #REQUIRED>

<!ATTLIST DECIMAL precision CDATA #REQUIRED>
<!ATTLIST DECIMAL scale CDATA  #REQUIRED>

<!ATTLIST  field nullOrNot  "NOT NULL" #IMPLIED>
<!ATTLIST field default CDATA #IMPLIED>
<!ATTLIST field unique CDATA "UNIQUE"  #IMPLIED>
<!ATTLIST field autoInc CDATA "no"  #IMPLIED>
<!ATTLIST field inputMask CDATA "no"  #IMPLIED>
<!ATTLIST field check CDATA  #IMPLIED>
```

```
<!ATTLIST foreignKey name CDATA #REQUIRED>
<!ATTLIST foreignKey referenceTable CDATA #REQUIRED>
<!ATTLIST foreignKey referenceField CDATA #REQUIRED>
<!ATTLIST foreignKey nullorNot  "NOT NULL" #IMPLIED>
<!ATTLIST foreignKey onDelete (NO ACTION|CASCADE|SET NULL|SET
DEFAULT|NO CHECK) #IMPLIED>
<!ATTLIST foreignKey onUpdate (NO ACTION|CASCADE|SET NULL|SET
DEFAULT|NO CHECK) #IMPLIED>
```

Appendix C InstantDB specification & MySQL specification

According to theses requirements, we defined the entities and their attributes of the Authoring System. Some tables have already defined in Course Delivery System [68].

Entity	attribute	Data type	constraint	Null Value	Others
changes	Staff ID CourseID SectionID Version Date Type ChangeID	CHAR(10) CHAR(15) CHAR(15) CHAR(10) TIMESTAMP VARCHAR(50) VARCHAR(10)	Primary key Primary key Primary key	No No No No No	Type means that it is creating new one or update information, etc.
Locked	LockedPersonID CourseID SectionID LockedTime	CHAR(10) VARCHAR(20) CHAR(15) TIMESTAMP	Primary key Primary key Primary key	No No No	
Course	CourseID CourseName(short) CourseName(long) Published version Points DeliveryMode MappingTable	CHAR(15) CHAR(50) CHAR(50) "yes" "no" CHAR(20) Float "internal" "external" VARCHAR(50)	Primary key	No No No No No	CourseID is generated automatically, unique and meaningless number

Development Section [35]	SectionID shortName longName ParentSectionID Sequence Type Published Version Label Lock LockPersonID	CHAR(15) VARCHAR(50) VARCHAR(50) VARCHAR(15) INT 'top' 'middle' 'bottom' 'yes' 'no' VARCHAR(20) VARCHAR(50) 'yes' 'no' CHAR(15)	Primary key	No No No No No No No No	SecionID is generated automatically, unique and meaningless number. Version number is generated by system, users can not aware of it. It is unique, maybe meaningless.
Lecture's Section [35]	SectionID ShortName LongName ParentSectionID Sequence Version Type	CHAR(15) VARCHAR(50) VARCHAR(50) VARCHAR(15) INT VARCHAR(20) 'top' 'middle' 'bottom'	Primary key	No No No No No No No	
Delivery Section [35]	SectionID ShortName LongName ParentSectionID	CHAR(15) VARCHAR(50) VARCHAR(50) VARCHAR(15)	Primary key	No No No No	
LearningObject	ObjectID ObjectName PresentationURL SourceURL DownloadFTP DownloadHTTP CreateTIME Published Version	CHAR(20) VARCHAR(50) VARCHAR(255) VARCHAR(255) VARCHAR(255) VARCHAR(255) TIMESTAMP 'yes' 'no' CHAR(50)	Primary key	No No No No No No No No	