INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Design and Implementation of a SMIL-based TeleLearning System

by
Beilei Huang

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

M. A. Sc.
in
Electrical Engineering

Ottawa-Carleton Institute for Electrical & Computer Engineering
School of Information and Technology Engineering (SITE)
University of Ottawa

October, 1999

Table of Contents

List of Figures

Acknowledgments

I would like first to thank my thesis supervisor, Dr. Ahmed Karmouch, for his continuous guidance and patience. Dr. Karmouch provided me a very comfortable and well-equipped environment for my work.

I would also like to thank Nabil Layaida, a W3C member, for his provision of information on the cutting-edge technology and great help.

My special thanks are due to my parents and my husband for their constant support, without which this work would not have been possible.

Abstract

One of the most obvious and important facts for TeleLearning is that it depends on telecommunication and information processing technologies to exist as a means for education. Since the learning and teaching take place at a distance, there must be some medium of communication to bridge the distance. The explosive Internet provides such a channel to convey the information nationally and internationally. Multimedia capabilities in instruction significantly enhance and expand the learning opportunities and quality for the students. Integrated video, audio, image, and text create a rich new learning environment awash with possibility and a clear potential to increase student involvement in the learning process.

For many years, various multimedia document models have been proposed and constructed by many researchers. In 1998, SMIL came out as a web-based international standard for synchronized multimedia document, recommended by W3C. We found it a very suitable model to fit in our Internet-based TeleLearning system. In this thesis, SMIL is employed as a format to structurally store an author-scheduled synchronized multimedia presentation for an information-rich course section. An algorithm to detect the temporal inconsistency within a SMIL document was proposed. A SMIL validation tool was developed, and the design and implementation of a SMIL player was discussed.

Chapter1

Introduction

1.1 Motivation

The world is changing swiftly. Learning becomes daily and lifelong. Education and training costs are rising. However, the budget for education is shrinking. Reduced federal funding and more training requirements are forcing institutes and businesses to look at creative ways to increase enrollment and provide high quality education with fewer teachers. Distance learning reveals itself as a new education model in which learning becomes dispersed both in time and location, thus many costs, such as relocation expenses, are saved.

Currently, rapid advances in communication technology are causing a dramatic increase in applications of distance learning in all levels of science education. On one hand, the demand for just-in-time training and the high cost of the traditional education model requires an educational reform that distributes the ever-changing knowledge base

to the learner who desires to learn at the lowest cost but the highest efficiency. On the other hand, the development of modern advanced technologies combining computation and communication makes this reform possible and viable. The following is a summary of some of the most important technological advances:

- The emergence of high speed networks and new storage technologies impose a new manner of processing information. They also provide valuable opportunities for the design of new applications that were not possible with previous technologies.

- The explosive growth of Internet users, along with the International Standards, and the highly developed and widely used new web-enable language – JAVA, permits information to be shared globally by non-computer specialized individuals.

- With the ever-increasing transmission technologies in telecommunication and the advent of new computer languages, direct manipulation of new types of information such as video, audio, animation and image, integrated in one single entity (a multimedia document), is becoming possible.

Consequently, considering all these evolving technologies, humankind is able to design and implement a computer-based distance learning application which allows non-sophisticated end users to be trained with a new education model.

1.2 Objectives

The main objective of this research is to design and implement a multimedia document-based TeleLearning system based on the framework of the current system MITS, which has been developing in our multimedia information and mobile agent laboratory for years. The generic architecture of MITS (Multimedia Interactive TeleLearning System) divides the system into three main components: the authoring system, the database system (including resource production), and the rendering system. MITS was first designed to use MHEG to achieve the real-time, reusable information interchange through heterogeneous platforms [WAN96]. Then, the HTML-based rendering system [ZHA98] replaced the initial design. With the availability of the new technology and Internet-based standards, it is possible to transmit and present multimedia information integrated in a single document. Our aim in this project is to make our TeleLearning system:

- ***more interactive***, so that the learners will have more levels of control on the learning order and pace rather than passively absorbing the knowledge provided by the courseware. The learning process will thus be more learner-centered;

- ***more natural and lively***, so the learner doesn't have to adjust himself/herself too much to adapt to the new education model. This is achieved by exploiting the multimedia technology; and

- ***simpler*** for both the instructor and the learner to use. The user-friendly interface is required for both the authoring stage and the rendering stage, providing an easy-to-use environment for both the author (instructor) and the learner.

1.3 Main Contributions

The main contribution of this research is the analysis and design of an algorithm to detect the temporal inconsistencies in a SMIL document. SMIL is a new Internet-based standard for presenting synchronized and integrated multimedia in a simple way. It's under developement of W3C (World Wide Web Consortium). Our project uses SMIL as one courseware document type in our TeleLearning system so as to enhance the interactivity as well as the ability to manipulate multimedia objects. Our algorithm guarantees the validity of the document before it is correctly presented.

Other major contributions include the implementation of the SMIL document validation tool and a SMIL player for playback of a SMIL document. This addition is integrated into the MITS system for playing back a course section.

1.4 Thesis Outline

This thesis is organized as follows: Chapter 2 reviews the field of TeleLearning, the evolution of the distance learning from correspondence course to TeleLearning system, and compares the advantages and disadvantages of distance learning versus the traditional education model. The architecture of MITS as well as an overview of various

applications and trends in TeleLearning are also presented. Chapter 3 describes the popular Internet-based standards: SGML, HTML, XML, and SMIL. The inability of each standard's document model to support multimedia information is also advised. Chapter 4 describes the time model of a SMIL document, analyzes the temporal inconsistencies that may occur in a SMIL document, and proposes an algorithm to detect them. The algorithm is expressed with pseudo-codes. Chapter 5 describes the implementation of the SMIL document validation tool (both syntax and semantics) and the SMIL player. Some output results are shown in snapshots. Conclusions and suggestions for future work are given in Chapter 6.

Chapter2

Overview of TeleLearning

Learning has become an important activity in our daily life, as we want to keep pace with the world. Ironically, some specific technologies are replaced so quickly that the knowledge becomes obsolete almost as soon as we master it. The reality results from many factors, including lack of the fast and efficient methods and media to convey the knowledge (authoring problem), the distribution delay (delivering problem) and less-attractive presentation view (presentation problem).

The way of teaching-and-learning changes as new technology for delivering a course through new types of media becomes available. Distance education was first practiced and accepted as the form of correspondence courses in Europe and then as the form of instructional radio and television in the middle of the century. The next evolution of distance education saw courses that were recorded and distributed via audio-video tape and CD-ROM. Currently, the most popular media are computer-based communications, including electronic mail, telephone-based audio-conferencing, video-conferencing and Internet [SHE96]. Particularly, TeleLearning, a large branch of distance learning that

allows knowledge to be stored and transferred through telecommunication technologies, has become a new technique put into practice by many post-secondary educators.

2.1 Background

There are numerous reasons why distance learning, using an advanced telecommunication technique, can overcome many drawbacks inherent in the traditional education model. However, building a network switching facility and installing the state-of-the-art communication technology are not themselves sufficient for establishing an efficient TeleLearning system. We need to know the characteristics of the strategies of teaching and learning before we go any further.

2.1.1Teaching architecture and Learning mode

Teaching and learning are inseparable. The strategy of teaching is crucial for any successful teacher, while a suitable learning strategy is a matter that any student applies in learning. As the designer and developer of a Telelearning system, we need to have a keen understanding of this process before we decide what technologies to choose and how to apply them to the system.

The six teaching architectures presented by Roger C. Schank have been widely employed by educators and CBT (Computer-Based Training) designers [SCH90]. They are:

Case-based Teaching Architecture

In this architecture, teachers are repositories of cases. They are assumed to be good storytellers. Knowledge is constructed and exploited as a case base from which the cases can be drawn when they are needed.

Incidental Learning Architecture

This architecture is based on the fact that people sometimes learn something else when they try to do or learn a special task. The important premise is that the student can be learning a great deal without noticing it when they involve themselves in something that is fun. To successfully realize this architecture, the learning environment has to be very attractive.

Cascaded Problem Sets

In many cases, we learn by solving problems. This teaching strategy requires the construction of libraries of cascades and content-based connections with internal relations between problems.

Directed Exploration of Video Base Connections

Today, many professors would like their students to report their learning result by doing some type of project, in which students have to explore the relative information with or without an assigned topic. This teaching strategy should provide three types of access modes: direct-inquiry, visual navigation and "button-based".

Simulation-based "Learn by doing"

More often than not, one learns a skill by doing. This architecture has applied to our life since we were children. This environment requires the building of at least four basic components: the simulator, a teaching program to help the student through the simulator, an input device or program that can understand what the student might ask the computer teacher, and a story-telling program activated by the teacher at appropriate moments.

Responsive Questions

Finally, a good mentor will provide on occasion good questions to his student when thinking about the new idea. This architecture is rarely used in a one-teacher-thirty-students classroom, but is actually effective. To implement this architecture, the software not only requires thought-provoking questions but also needs to anticipate the best time to ask the question.

From the student's perspective, an important variable in effectiveness is the preference of the student for a particular mode of learning, i.e., cooperative, competitive, or individual [SHE96]. Applied to an individual, learning can be broken down into: active learning or passive learning. The active learning mode is more learner-centered, while the latter tends to be teacher-centered. In the active learning mode, the learners are provided with the resources and an environment in which they can actively create their own meaning in that context, rather than to passively absorb knowledge structures created by the instructor.

2.1.2Traditional education versus distance learning

The traditional education system is referred to as chalk-and-paper work. The success of the teacher depends on how a course is organized and presented. The knowledge and skill an individual lecturer possesses has great influence on the teaching-and-learning result. Moreover, teaching usually happens in a classroom where students, numbering from tens to hundreds, gather together to listen to one lecturer presenting at a specific classroom in a scheduled duration of time. The effectiveness of the learning completely depends on the human being's state, and the teacher's role is more like an information resource provider than an instructor.

Distance learning, featured as the availability for learning at flexible time and location, relies more on communication technologies. It is fast becoming a viable adjunct to the traditional classroom model of teaching and learning. Learning effectiveness no longer relies solely on the teacher and the learner. Which technologies are chosen and how each technology is used become important issues that affect the learning environment and ultimate success.

2.1.3Advantages and disadvantages for Telelearning

TeleLearning, a form of distance learning in which more telecommunication technologies are applied, is widely recognized as the key to delivering more training to more people on more subjects with higher impact and effectiveness, in a more cost-effective way. There rise numerous advantages compared with the classroom model. Here we list the most prominent ones:

- Flexibility

In the sense of time and location for both the teacher and the student, TeleLearning provides such a perfect environment that spatial and temporal restriction may not apply for the study. This is good for both parties.

- Efficiency

With the advanced technology, such as telecommunications and multimedia technologies, distance collaborative education is allowed. A similar course in different universities can be offered by a group of instructors. It's possible that each instructor teaches the part on which he/she feels more confident with the less effort. For some graduate courses, students from diverse universities and at dispersed locations can form a virtual class instead of having one-teacher-ten-student situation in every institute.

- Reusability

With the usage of database management systems, course material for the basic courses, such as calculus, that don't update as fast as the new technology can be customized into courseware and stored into the database for multiple retrievals.

Since TeleLearning technology is still at its developing stage, we can't promise whether it can replace the old education system in the future or not. Not only the technology itself is not mature enough for implementing what the sophisticated TeleLearning system needs, but also the system designers and developers have to consider many teaching and learning strategies as we mentioned above, which they probably are not professional at. Moreover, some other important factors in education

such as evaluation, credit admittance and student administration problems should also be carefully considered, too.

2.2 TeleLearning Applications

There have been a lot of TeleLearning applications developed by organizations and institutes. In this section, we look into some typical applications under developing and find out what achievements have been done with, what trend people are working toward, and what can and should be improved.

2.2.1 Virtual classroom-based applications

A safe innovation step from traditional education model to TeleLearning is to build the system with advanced technology within the student and the teacher's familiar context. This is so called *virtual classroom* by a number of TeleLearning applications.

PERSYST is such a system developed by Bell labs/ Lucent Technologies [GIN98]. It is based on the conventional education model in which a course is taught by a live instructor using a *classroom*, augmented by other equipment and environment such as a projector and a blackboard. In PERSYST"s virtual classroom, students who simultaneously take the same lecture "join" together through their own computer, while a live instructor controls the lecture on the other side. *Blackboard* (slide presentation), *text chat* (communication through typing text messages) and *audio services* (live audio, pre-recorded lecture, and in-session audio data recording) are three basic functions in the virtual classroom. The instructor can "call on" any student to speak to the rest of the class

and "cut off" the speaking as well. One of the advantages over the physical classroom is that the student can send text comment (through text chat) or questions to the instructor at any time without waiting in a queue. This architecture is more or less similar to another application IRI (Interactive Remote Instructor) developed by Computer Science researchers at Old Dominion University [WAH96].

In Department of Electrical and Computer Engineering at University of Illinois at Urbana-Champaign, virtual classroom model has been applied and practiced in circuit analysis course since 1994 [OAK96]. The student can remotely access the networked course material asynchronously. A *PacerForum* provides an environment for student's discussion using asynchronous conferencing technique. In their approach to this specific course, it has been proved to be very successful in terms of teaching efficiency and students' learning effect.

2.2.2 Individualized network-based application

A bolder education reform is to give more learning control to the learner. Learning is not necessary to be restraint to a group. Individualized learning provides the learner more freedom to organize himself/herself and sometimes assists the learner in improving learning efficiency.

MANIC is such a system developed by University of Massachusetts [STE97]. In this system, course materials are stored as HTML documents (containing text and still GIF images), while audios are delivered over Internet using RealAudio Server to

synchronize with the text part. The student with multimedia capable machines with WWW browsers can take the course when they are ready to learn.

2.2.3 MITS

In Multimedia Information and Mobile Agent Lab at University of Ottawa, a TeleLearning system, called Multimedia Interactive TeleLearning System (MITS) has been under development since 1995. The prototype of MITS framework [WAN96] was initially designed to be built on an ATM network with five components sitting around at either client or server side: *courseware user, courseware author, on-line facilitator, media production center, and courseware database server,* shown as Figure2.1.

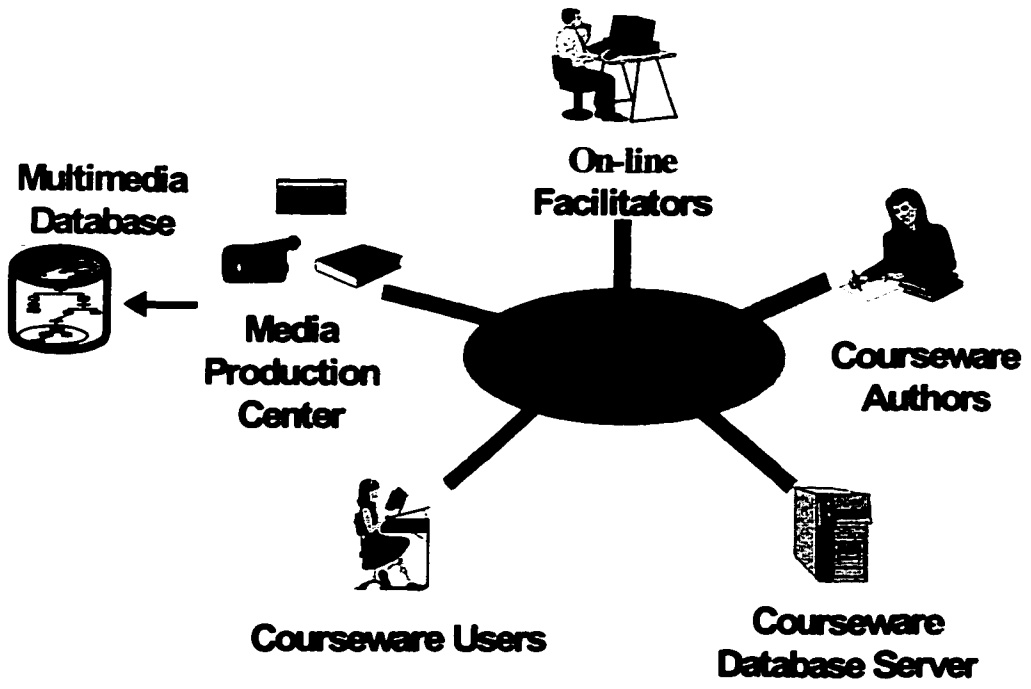


Figure 2.1 Generic architecture and components of MITS

The system architecture of MITS was based on MHEG, who acts as a container and a descriptor for multimedia presentation, to achieve the aim of real time and reusable information interchange through heterogeneous platforms. However, an MHEG engine is required to pack and unpack the MHEG objects to hide the detail of the complicated CODEC technique from the author and the courseware user. When coming to the implementation, it was found that HTML could be a better choice for delivery and presentation since it is a standard that most of the commercial Web browser can understand and interpret. Furthermore, with the explosive growth of World Wide Web (WWW), delivering course material in HTML through Internet can save us lots of effort in designing and implementing a browser for course presentation, while being able to distribute courses to more students (refer to document standard discussion on chapter3).

The Internet-based courseware rendering system MICR [ZHA98], a subsystem of MITS, was designed and implemented using JAVA language. By using HTML and JAVA, the system achieves the goal to provide an on-demand and platform-independent learning environment to users on the Internet. The courseware structure and courseware material is stored in the *courseware database* and *multimedia database* respectively. The commercial object-oriented database management system product ObjectStore (from Object Design Inc.) is adopted to handle the storage and retrieval of the courseware multimedia objects [ALS98]. Upon the receipt of the user's request on a specific course, the rendering system opens the correspondent courseware database, retrieves and assembles the course contents into an HTML file and sends it back to the user through Internet. In rendering, the playback of multimedia (especially audio and video) is managed by third party audio-player or video-player with plug-in mechanism. Figure 2.2

shows the architecture of the system which mainly contains client, server and database system.

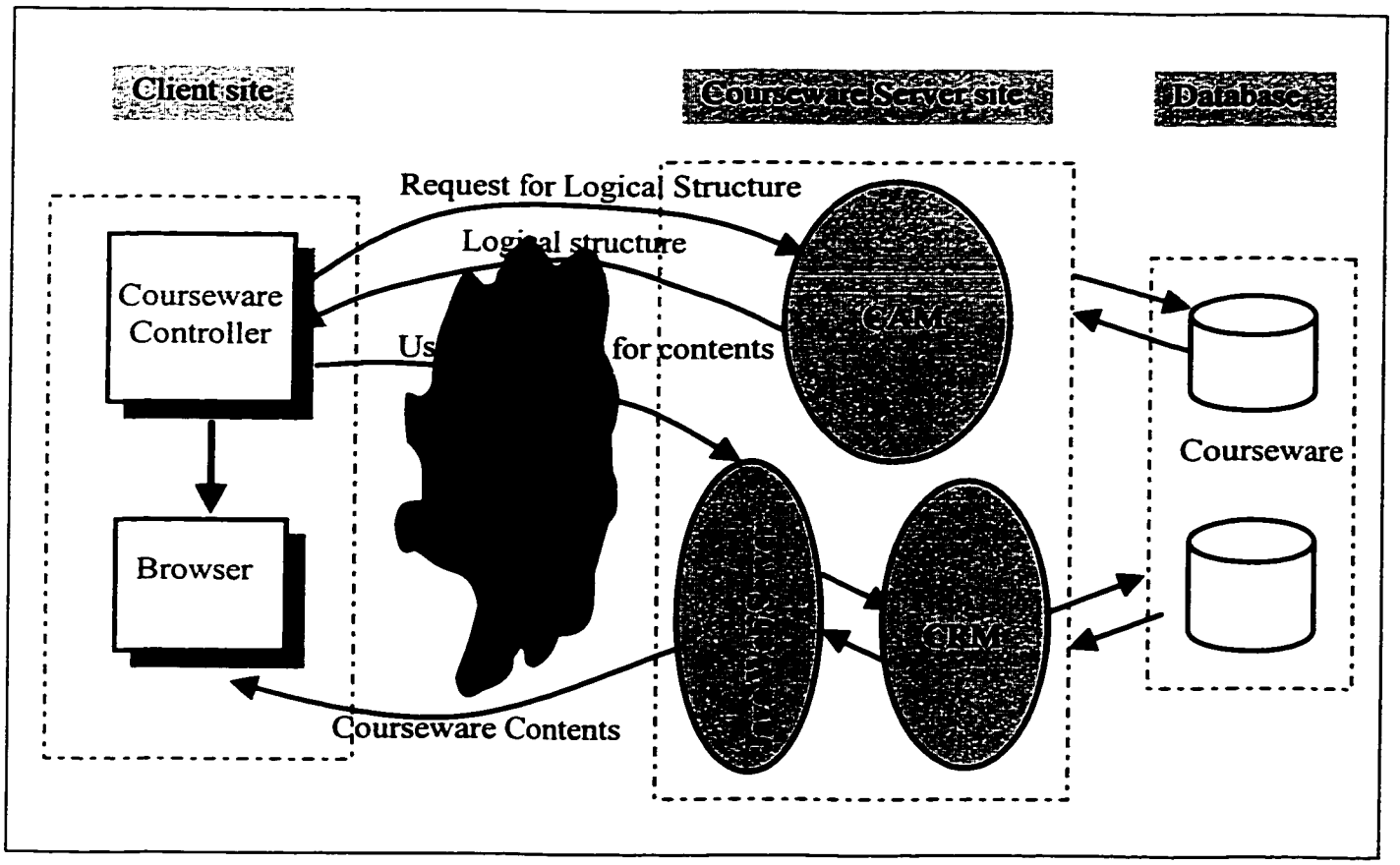


Figure 2.2 Architecture of MICR

MITS is the framework of a TeleLearning system. Its characteristics include *course on demand*, *interactive*, *integrated*, *open and scalable environment*, *courseware reusability*, *platform-independent*, etc. Many more features can be added or extended. In this thesis work, we propose and implement different presentation forms to make better use of the characteristics of the course content.

Firstly, if the material components of a course contain most of the content in plain text or image type, HTML file is generated and presented through Web browser. Secondly, if a course resource contains continuous media such as audio, animation and video, and the synchronization issue is not necessary, we can still use HTML page to render the course, while the audio and video part will be handled by the plug-in media player, such as RealPlayer (for streaming media) or LiveAudio (for audio file with suffix such as "au", "wav", etc). Thirdly, if a course presentation contains rich visualized media and few plain texts, a multimedia-synchronized presentation will be rendered. As of June 1998, a new exciting multimedia language, called SMIL (detail will be discussed in chapter3) was announced to be a W3C recommendation. This brings us a real opportunity to present a synchronized multimedia scenario to the end user through Internet. The presentation page is assembled behind the Java Web Server at the server side. Before the page is generated, a course filter examines whether a SMIL document is attached to the specific course and decides the way of presenting the multimedia components. Either a plug-in is used by embedding the object, or the SMIL rendering applet is included in the dynamically-generated HTML page. Figure 2.3 shows the mechanism of the server architecture.

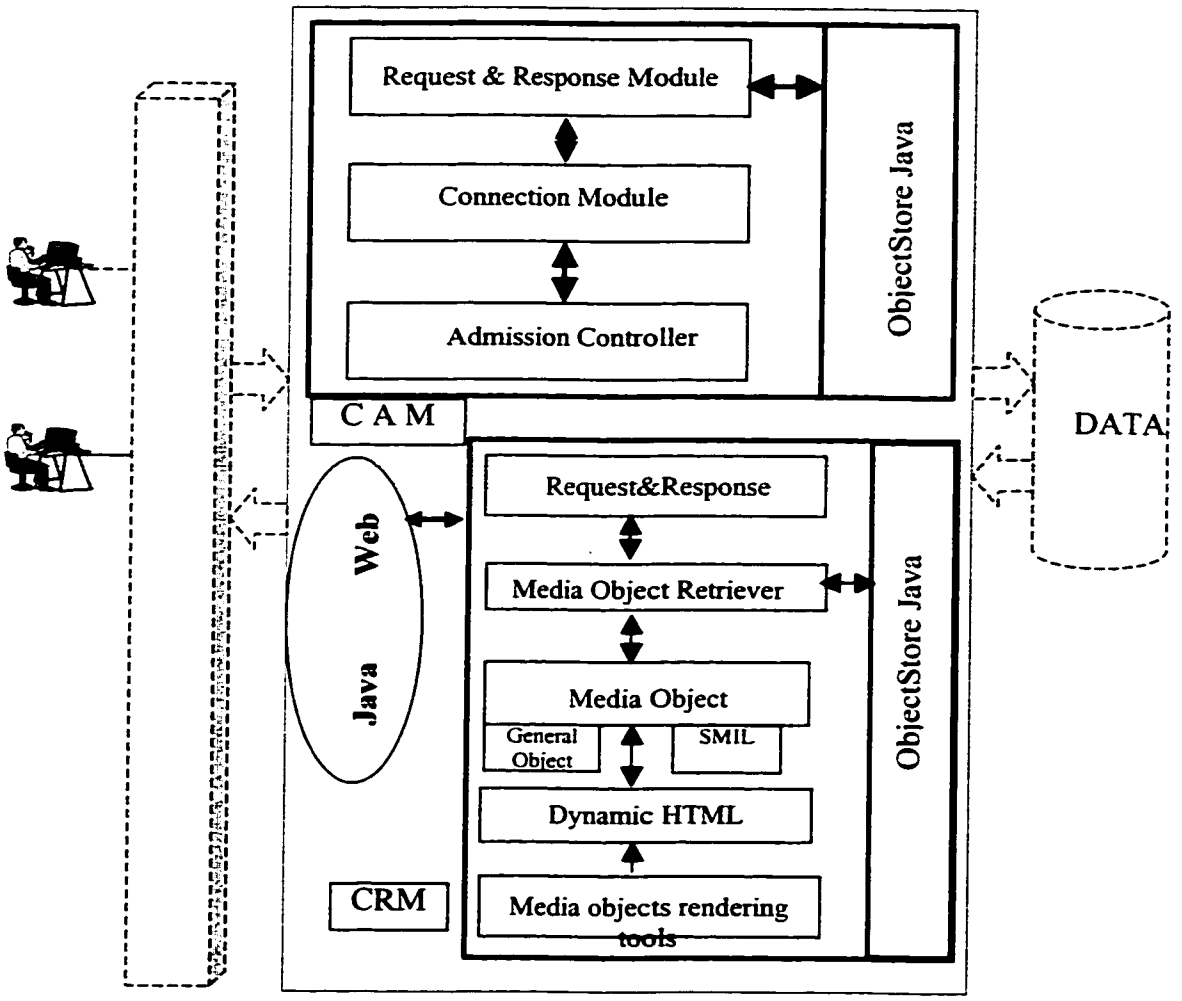


Figure 2.3 Server system architecture

2.3 Trends in TeleLearning

2.3.1Web-based

More and more TeleLearning applications present their courseware through World Wide Web (WWW) due to its rapidly increasing popularity. CommerceNet and Nielsen Media Research [COM98] have, since 1995, periodically conducted telephone surveys with individuals over the age of 16 in the U.S. and Canada to assess interactive usage, interests, and on-line activities. According to their survey statistics, the population of WWW users reached 79 million in June 1998, up from 58 millions in September 1997. Regarding the types of uses of the Internet, the result of the survey indicates that the Internet is widely used for informational retrieval purposes by students, educators, and business people.

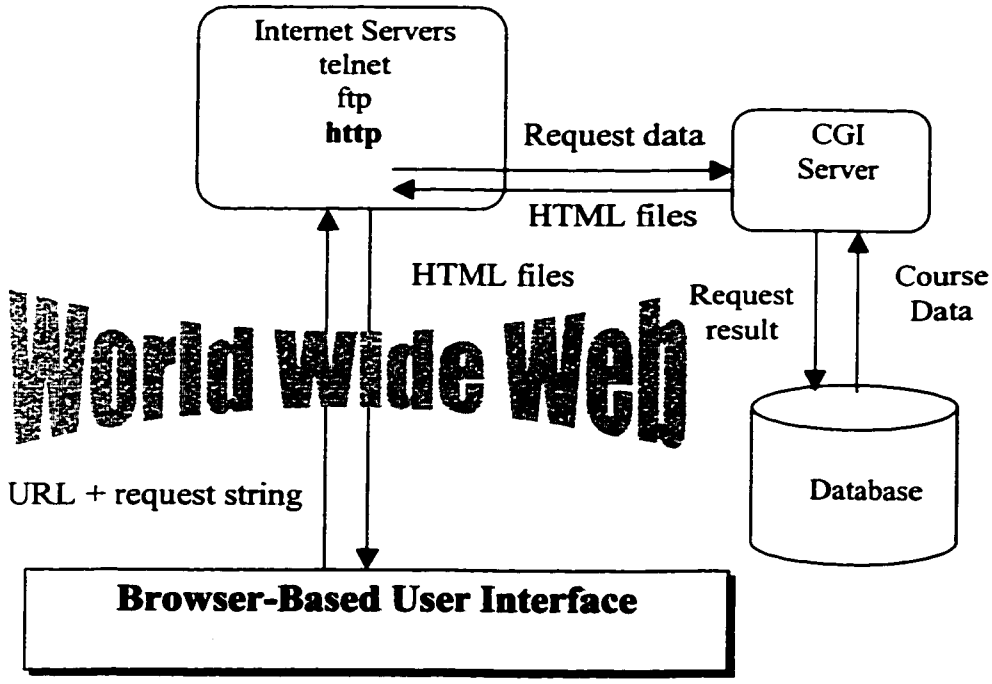


Figure 2.4 A generic intuitive architecture for web-based application

A number of projects have been undertaken to put course materials on-line for student access using WWW technology. These include efforts at University of

Massachusetts [STE97], University of Illinois [OAK96], Simon Fraser University, and many more. In these projects, a homepage is typically an entrance point where the context of the course starts. The hyper-link within the page can guide the user to the other page where related information is embedded. Any user with a browser such as Internet Explorer or Netscape can read the course material prepared in the HTML files that the web browser can interpret and present. A generic intuitive architecture for web-based application is shown as Figure2.4.

2.3.2 Interactivity

Interactivity is another important issue and trend in TeleLearning. The first form of Distance Learning, *correspondence course,* could not be popular somehow because of lack of interaction. Learning became an isolated activity and students gradually felt bored and left.

In conventional classroom learning model, interactions happen directly between human beings, that is, between the student and the teacher (either in class or after class), and among the students. In TeleLearning environment, these face-to-face interactions change to distant interactions via telecommunication technologies. If we consider learners, instructors, and courseware as three main interacting parties in the TeleLearning environment, three types of interaction may happen. That is, the interaction between the student and the instructor, between the student and the student, and between the student and the courseware. The interaction scenario is shown as Figure2.5.

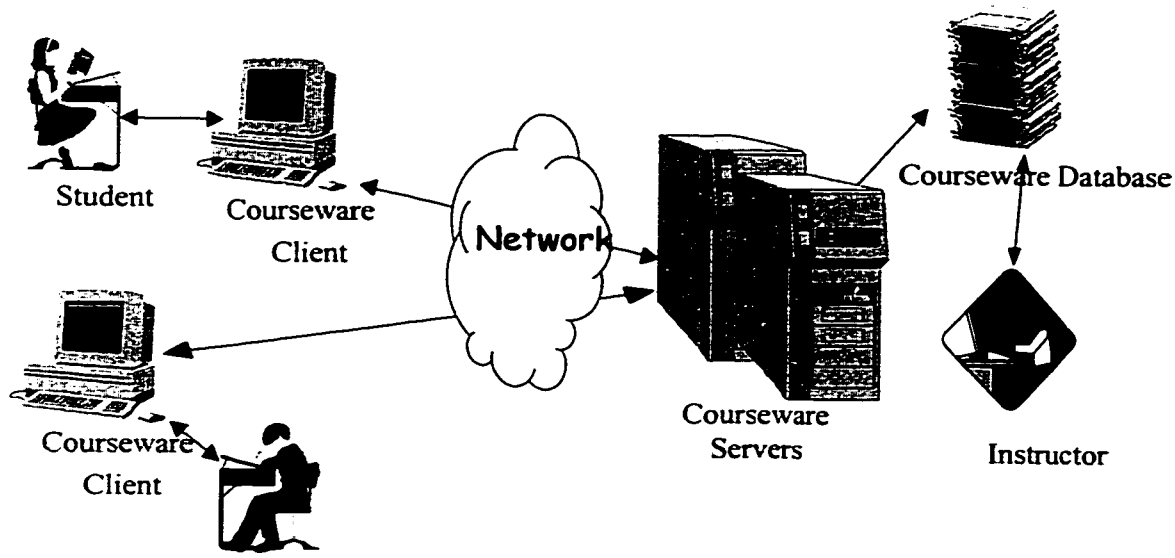


Figure2.5 Three types of interaction in the Telelearning environment

Technologies such as audio-conference, videoconference, white board (allowing communication through drawing), and chat room (communicating through real time typing) allow the in time discussion between the student and the on-line facilitator, or instructor. The end-to-end interaction through chat room or white board usually permits users at client side send messages to each other via a real-time central server that automatically performs as the "switch" and mediates between two end-points. The mechanism of this real time chat through Internet is called Internet Relay Chat (IRC). It saves the user the cost of long distance call by telephone while the quality is a little bit worse in terms of delay and clearness due to the telecommunication technology limitation, such as bandwidth and modem speed, etc.

In our project, we enhance the interactivity between the student and the courseware by using SMIL, in which the potentially interactive and synchronized

multimedia scenario is prepared by the course author in a simply way. The foreseeable interaction between the student and the instructor can be embedded to the SMIL scenario and thus shorten the distance of the interacting objects.

In summary, the interactivity in a distance learning system takes many forms; it is neither just limited to audio and video, nor solely to instructor-student interactions [SHE96]. The essential is that it must reflect the connectivity the learner feels with the distance teacher, facilitator and the peer.

2.3.3 Multimedia ability

Multimedia is another trend in designing a TeleLearning system. Grace [AUG95] has done experiment on how the multimedia can affect the effectiveness of learning and concludes that the way information is represented and organized in multimedia is very likely to facilitate and enhance the internalization, retention and recall of knowledge. Specially, Multimedia combined with interactivity, where an appropriate level of flexibility is designed for the user to navigate for information according to their style of learning and cognitive ability, can potentially reduce the gaps between the performance of the students by using another mode of knowledge delivery mechanism for students who perform relatively less well under the traditional teaching paradigm.

Multimedia refers to the integration of multiple media types of data, such as text, image, video and audio. Based on their characteristics of intrinsic temporal behavior, they can be classified into discrete media (such as image and text whose intrinsic duration is zero) and continuous media (such as video and audio). The integration of the different

media may involve the synchronization issue between the media. However, people sometimes simply take the combination (without synchronization) of the multiple media as "multimedia". In this thesis, we refer this type of multimedia as *static multimedia*, while the other is referred to as *dynamic multimedia*.

With the development and utilization of multimedia technology, large information is allowed to be organized and represented in a wide variety of ways. Not only does it provide designers the ability to organize information based on its physical format, i.e., sound clips, still images, texts, and so on, but also makes possible to organize information based on the learner's plan to use information. This is to offer an easy-to-use authoring tool for teachers and then deliver the course contents to students based on their preference of presentation and select-and-reject the contents they need.

Chapter 3

Structured Documents For the Web

3.1 Document types

Web is originally designed to pool and share information [CRA98]. It thus benefits information-rich areas such as distance learning, which allows people in the dispersed locations to share knowledge whenever they want. To publish courseware over Internet, choosing a suitable document type will assist the better manipulation of the content of the document source, and make the courseware more efficient and interactive. Before we look into the characteristics of each document type, let us clarify some relative confusing terms first:

Hypertext

Hypertext is a metaphor for presenting information in which text, image, and actions are linked together in a complex but non-sequential web association. It allows the user to browse through the relative topics regardless of the presented order.

Hypermedia

Hypermedia is a similar term to hypertext, but emphasizing more on none-text components such as video and animation.

Hyperlink

Hyperlink is an information structure that represents the relationship between two or more objects [HYT92].

3.1.1 SGML document

SGML (Standard Generalized Markup Language, ISO 8879) is an international standard designed to describe the structure and contents of types of electronic documents. It's the basis of several Internet-based standards including HTML and XML. Many organizations, especially those who demand special and complex requirements for document management use SGML relying on its flexibility and power. Compared with other document standards, such as Adobe Acrobat and Microsoft RTF, SGML provides the author the broader choice to express their document organization. That is, instead of focusing on the layout of the document, SGML concentrates on the *structure* of the electronic documents. The structure is defined as a set of elements and their relationship in the SGML document. For example, a title, an author, an abstract (or a short description), paragraphs and sections are typical elements of a general report. These

elements respect a hierarchy where the root element can be the section element which is made up of the simpler elements such as a title, author(s), an abstract and paragraphs. Usually, an element contains the start tag <element>, the content and the end tag </element>.

The advantage of the feature-rich SGML also results in the disability of widely application due to its complexity. In only allows sophisticated people or organizations to use.

3.1.2 HTML document

HTML (HyperText Markup Language) is a standard markup language used to create and process hypertext on the World Wide Web. Its structure is defined in SGML syntax with a finite set of hyper tags. An HTML document is not only easy to construct but also "portable". Most of web browsers, such as Microsoft Internet Explorer and Netscape navigator, can understand and present the content of the HTML document very well. As Internet has become more and more popular, many universities and organizations have rendered their distance learning courses in HTML through web. With the same motivation, the first version of our MITS (Multimedia Interactive TeleLearning System) was designed and implemented to present the *static* multimedia TeleLearing courses by dynamically generating HTML pages upon the different request from the end user [ZHA98].

HTML has great advantages in allowing *static* multimedia contents to be easily expressed and then presented through Web. However, it lacks the ability to allow the

dynamic multimedia to be expressed in a simple way. In another word, people can not synchronize multimedia contents in time at the authoring stage. A recently proposed Web document type, called SMIL (Synchronized Multimedia Integrated Language, pronounced "smile"), solved this problem. Furthermore, HTML provides a set of predefined tags to assist the understanding of the content layout. For example, "<p>" means a paragraph, "<table>" means a table. But it doesn't support the identification of what the content is. It may lose some information when retrieving the content from a database, in which contents are stored along with their identifications such as "course author", "course title", or "creation date". In HTML, these identifications are stripped off when presenting their real content. Often, these contents will be wrapped in a <p> element in the HTML document. For the program that wants to search a specific information item, such as "course author" in HTML documents, it becomes impossible. Using XML (eXtensible Markup Language) instead can solve this problem.

HTML is thus more suitable for the project in which document structure is not an important issue. Also, HTML itself does not support multimedia. It allows the *combination* of the text and image, which can bear hyperlink to the other HTML document. However, dynamic media such as video, animation and audio are not handled by HTML. Instead, it supports a mechanism that allows *plug-in* to handle an embedded object, such as a video. The rendering of such an object is not synchronized with other layout objects.

3.1.3 XML document

XML is an *extensible* markup language because its format is not fixed as HTML. Actually, XML itself is not a single markup language. It's a metalanguage that allows you to define your own customized markup language for many classes of documents. It is written in SGML but designed to ease the complexity of SGML. Contrary to HTML, which is an application of SGML, XML is an abbreviated version of SGML, in the sense that it omits the more complex and less used parts of SGML in return for the benefits of being easier to write and understand documents, and better suited to delivery and interoperability over the web. In another word, it removes the constraint of HTML which is dependent on a single, inflexible document type, as well as the complexity of SGML whose syntax allows powerful but hard-to-program options. XML is a project of World Wide Web Consortium (W3C). XML 1.0 specification was accepted by W3C as recommendation on February 10, 1998 [XML98]. It provides a never-better solution for storage and retrieval of structural information.

3.1.4 SMIL document

SMIL, pronounced as "smile", is the abbreviation of Synchronized Multimedia Integration Language. It allows the author to write a synchronized multimedia document without learning to use the sophisticated authoring tool. SMIL documents are XML syntax-based documents with pre-defined SMIL DTD. It's a format representing hypermedia on the web. It incorporates the hypermedia principles such as spatial layout, temporal composition, synchronization and navigational hyperlinking [LLO98]. The most valuable contribution of SMIL is its time model for synchronization multimedia

elements. On the Web, media synchronization is currently expressed by using a script language such as JavaScript or VBScript. Script languages have a number of well-known disadvantages. Script-based contents are hard to produce and maintain. Moreover, it's hard to build conversion tools and automated tools for Script language. SMIL comes out as a simple but powerful language that allows people to create attractive synchronized hypermedia with a plain text editor. It has the following characteristics according to W3C announcement [W3C97]:

i. The presentation is composed from several components that are accessible via URLs, e.g. files stored on a Web server. These components have different media types, such as audio, video, image or text.

ii. The begin and end times of different components are specified relative to events in other media components. For example, in a slide show, a particular slide is displayed when the narrator in the audio starts talking about it.

iii. Familiar looking control buttons such as stop, fast-forward and rewind allow the user to interrupt the presentation and to move forwards or backwards to another point in the presentation.

iv. Additional functions are "random access", i.e. the presentation can be started anywhere, and "slow motion", i.e. the presentation is played slower than at its original speed.

v. The user can follow hyper-links embedded in the presentation.

SMIL specification1.0 [SMI98] was announced to be W3C recommendation on June15, 1998.

3.2 Related concepts for web documents

3.2.1 Document DTD

A DTD (Document Type Definition) is a formal specification for the structure of an SGML document. It defines a document as a set of elements which follow a given structure including the element name, minimization rules and content model. To build an SGML application, we declare the document structure in a DTD and then mark it to the document. Different documents with the same DTD will have the same document structure. An SGML document with a specific SGML-syntax based DTD is an application of SGML. Both HTML and SMIL document have their own predefined standard DTD called HTML DTD and SMIL DTD respectively. XML, however, allows the user to design their self-defined DTD to construct the document structure. XML DTDs are subsets of SGML DTDS.

3.2.2 Valid and well-formed document

The concepts of valid and well-formed come with XML. An XML document is considered *valid* if there is a DTD associated with it and the document complies with the DTD. An XML document is considered *well-formed* if all the elements are cleanly nested. That is, "if the document contains one or more elements, there is precisely one element (the root or document element) for which neither the start nor end tag is inside any other elements, and all other tags nest within each other correctly" [Holzner1998]. XML documents are well-formed but not necessarily valid. However, SMIL documents

conform to XML syntax based DTD and thus should be not only well-formed but also valid.

3.2.3 Document Object Model (DOM)

The Document Object Model is a platform- and language- neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of the parsed document. It provides a standard set of objects for representing HTML and XML documents, a standard model of how these objects can be combined, and a standard interface for accessing and manipulating them. Until July 20, 1998, W3C has published several versions of working draft of DOM Level One specification 1.0. The specification defines a set of DOM objects and interfaces that are sufficient for representing the content of parsed HTML and XML documents without loss of significant information. On the other hand, these objects and interfaces are also sufficient to construct an entirely new document instance programmatically that is identical to the parse form of a given HTML or XML document. The primary API types in DOM include *Node,* the base type of most objects in DOM, *Element,* the elements in the HTML or XML document, and *Document,* the root node of a document. No matter when constructing or deconstructing an XML document, *Element* is the vast majority of node types that authors or programmers will encounter. Elements can contain elements as child nodes. They contain all the contents between the start tag and the end tag of an element. Element objects have a list of Attribute objects which represent the combination of the attributes explicitly specified in the document, as well as those defined in the DTD which

have default values. The terminology will be used in later chapters when we work more closed with SMIL.

The latest version of DOM specification 1.0 is public on July20, 1998. It includes requirements, Level one core, HTML and XML [DOM98].

3.3 Works on SMIL

As we discuss in the previous chapter, the trend for a TeleLearning project is web-based, interactive, multimedia-capability and more. SMIL is thus a very competitive candidate to achieve these characteristics in our decision of document type. By using SMIL, the resource of multiple media can be stored in the distributed database system. A SMIL player, which is able to interpret the valid SMIL document and present it, will take care of synchronization between the multimedia. A VCR-like interactive user interface can be built on the SMIL document presentation. However, a validation tool is required to examine the validity of the SMIL document at the authoring stage.

Currently, SMIL syntax validation service is available at W3C site (http://www.w3.org/AudioVideo/). There is not a complete SMIL player that can present all the SMIL specification. Yet, some companies have implemented some kind of SMIL player by the time of this writing. In this section, we will address the state-of-the-art both of validation tool and SMIL player, and address our work.

3.3.1Validation tool

At the time being, the SMIL validation service on W3C public site is provided by CWI, the National Research Institute for Mathematics and Computer Science in the Netherlands [CWI98]. Unfortunately, the "validator" only validates the syntactical validity. No complete SMIL validator is published.

In our lab, we developed a SMIL validator which checks the syntax error as well as the semantics errors of a SMIL document. The latter problem is a special problem occurring in time-based hypermedia document and does not necessarily happen in XML documents. It's caused by the time model of the SMIL specification. When we found this problem, there had been no published research work (or paper) of analysis on the temporal inconsistency of SMIL documents. In our thesis work, we first analyze the possible inconsistency that may happen in a SMIL document, and then design the algorithm to detect them. Our validation tool implements both syntax validation and semantics validation.

3.3.2 SMIL player

3.3.2.1 Products available from other parties

More work has been done on the SMIL player than the validation tool. Four SMIL players are hyperlinked on W3C official site at the time of my writing. HPAS (Hypermedia Presenting and Authoring System) is a system implemented by former DEC

(merged into Compaq later) people. In March 1998, HPAS was claimed to be the first implementation of SMIL player on the W3C event calendar. However, it could only present SMIL documents within Netscape on Windows platform. The performance was not good for remote access. We downloaded the Applet to our local machine and tested it. The result was not satisfactory. Details about it can be found at the URL [HPA98].

In July 1998, RealNetwork Inc. made Beta SMIL implementation G2 RealSystem™ available [RLN98]. To play back the SMIL presentation is one of the features of the product. The player is specialized on transferring the streaming type of media, such as RealAudio and RealVideo. The synchronization is very good too. The RealPlayer is independent of browser. It designed and implemented its own browser to play back the multimedia synchronization document.

At about the same time, CWI offered a *feature*s release of a SMIL player called GRiNs to offer the user to evaluate [GRI98]. As a `features' release, GRiNs have tried to support as many data types as possible, given the inherent incompatibility of machines and systems across the Web. The release thus often makes use of under-optimized third-party libraries to actually render the presentation's content. RealSystem G2 player is on the other hand called "performance" release.

3.3.2.2 Our work

In our Multimedia Information and Mobile Agent lab, we designed and implemented a SMIL player which can play back SMIL documents using a popular technique: Web-enabled Java language. The purpose of this implementation is somewhat

like that of GRiNs. We want to give our user a feeling of how SMIL presentation works. We had to develop our own SMIL player because at the time we started this project, only HPAS was released with bad performance. Moreover, we could design and implement more features on SMIL presentation to suit our TeleLearning project. For example, we can implement more user controls on a SMIL scenario. In our player, we developed two-level user controls on the SMIL presentation. We'll discuss about it in Chapter 5.

Chapter 4

Analysis and Algorithm For SMIL Document Validation

4.1 Importance of validation

As stated in the previous chapter, SMIL documents are XML documents with specific DTD. Most importantly, they define the time structure of the synchronization elements of the multimedia document. They must be valid before they are processed by a SMIL processor and before any SMIL player is able to correctly present the semantics meanings contained within the document. This validity includes two aspects: syntax and semantics.

4.1.1 Syntax validation

Syntax validation checks whether the document is well-formed, as well as whether it conforms with SMIL DTD. In SMIL documents, elements are nested or

contained within other elements. A missing end tag may cause the uncertainty of the document's logical structure and sometimes, cause confusion of the temporal relationship. In the following example, figure 4.1, the end tag of "seq" element is missing. If a SMIL player is *intelligent* enough to *guess* the scenario the author wishes to present, then three different presentations may match the document specification. It would do so on the basis of the location of the inserted "seq" end tag.

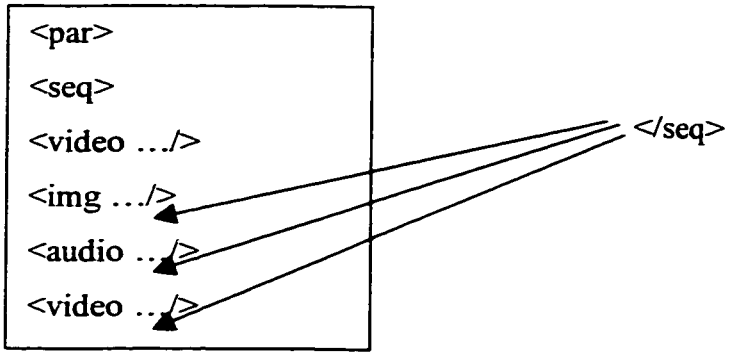


Figure4.1 An example of an invalid specification for SMIL scenario

4.1.2 Semantics validation

Multimedia data can have natural or *intrinsic* time dependencies (e.g. audio and video are recorded simultaneously and have an intrinsic none-zero duration). Static media data, such as image or text don't have time dependencies, or in other words, their intrinsic duration is zero. However, they can have *synthetic* temporal behavior by simply repeating the static data until the end instant that the author expects.

SMIL allows the author to explicitly describe the temporal behavior of the presentation through synchronization elements and element attributes such as begin, dur,

end, and endsync, etc. These specifications provide the author with a variety of expressions for playback time assignment of a media object. However, some temporal inconsistency can occur if the timeline of the presentation is not well scheduled or is improperly expressed. It is very important to check the semantic correctness before the multimedia document is played back. Though a SMIL player may be able to play back the SMIL presentation without validating the document, it can cause a viewer misunderstanding if the document is not a valid one. A simple example: a synchronization element "video" is explicitly assigned the value of its attributes begin, dur, and end as "0s", "20s", "10s" respectively. These three values are obviously conflicting because they don't satisfy the constraint equation: <begin-value> + <dur-value> = <end-value>. A SMIL player can play back the element by considering any two of the values, depending on the player's "understanding". This then results in different scenarios played back by different players. A semantics validation before the playback of the document will point out the error information to the author for further clarification or correction, and thus guarantee that the author's work will be correctly presented later.

4.2 Structure of SMIL documents

The SMIL document has a tree-like structure with element "smil" as the root. It can contain two children: "body" and "head". The "head" element contains information that is not related to the temporal behavior, such as spatial information, while the "body" element contains information that is related to the temporal and linking behavior of the document. We will call the sub-tree starting from the "head" element a *spatial sub-tree,* and the sub-tree starting from the "body" element a *temporal sub-tree.*

The "body" element can further contain the *synchronization* elements, "*switch*" element, and "*a*" element. The content and attributes of these elements define the temporal behavior of the document. The synchronization elements contain the temporal attributes, such as "begin", "end" and "dur", which allow the author to express the desired timeline of the multimedia presentation. They include "par" elements, "seq" elements, and media object elements. The "*par*" and "*seq*" elements can contain the *synchronization* elements, "*a*" element and "*switch*" element as their children, while *media object* elements can have only "*anchor*" elements as their children.

Definition: A ***two-level tree*** is a tree in which there exists only one root at the parent level and its leaves at the children level.

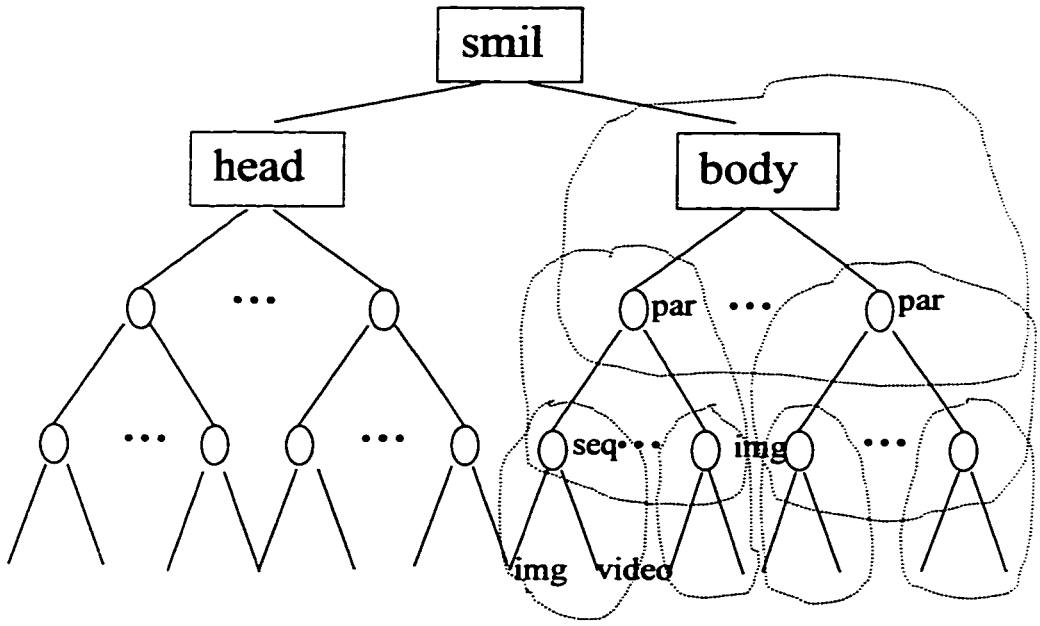


Figure 4.2. SMIL document structure

In a SMIL tree (an example is shown in Figure 4.2), element "par" and "seq" are tree nodes, and media object elements such as "video", "audio", "image", "text", etc. are leaves if they don't contain an "anchor" element. Observe that a tree contains a set of two-level trees (circled in the dot curve in Figure 4.2). In the temporal sub-tree, the parent node (root of the two-level tree) can be a "body" element, "par" element, a "seq" element, or a media object element that contains child elements, and leaves can be any synchronization elements.

4.3 Modeling temporal structure of multimedia document

4.3.1 Overview of temporal model

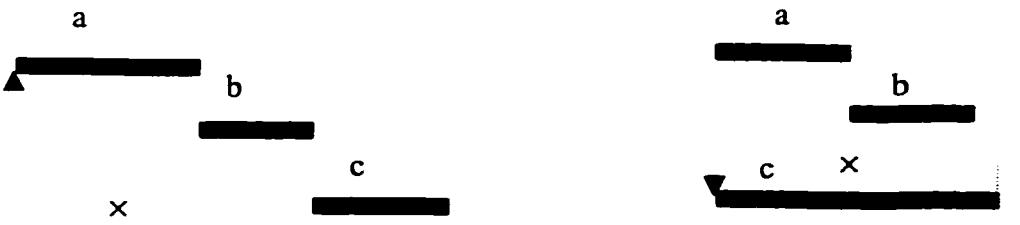
Multimedia system allows continuous and discrete media to be integrated together. Continuous media, such as video and audio are time-dependent, while time-independent media, such as image and text can be artificially assigned duration to synchronize with other media. To visualize a multimedia scenario, not only the spatial layout has to be designed but also the time interrelation between media objects has to be scheduled. It's hard to neglect the temporal behavior when we talk about multimedia. Therefore, the representation of temporal interrelations between media items is a very important issue in a multimedia presentation system. To assure the correct operation on the temporal interrelations requires two-fold works. Firstly, a good temporal model is needed to allow the author to express the relations on the related media items. Secondly, a rendering system can correctly interpret such relations. Many works have been done on the first issue, but few works have been done on the second problem, particularly with

regards to the consistency issues on the temporal relations between the media items. There is no one-size-fits-all model to represent various multimedia systems in different fields. A good temporal model for multimedia should provide simple yet powerful expressiveness to the author to express the interrelations between media.

4.3.1.1 Existing Temporal Model

The existing temporal models can be classified into two types [WAH94]: point-based and interval-based. In the point-based model, points are the atomic events and *before*(<), *simultaneous to*(=) and *after*(>) are the three basic temporal relations between two past events. When applying to future events, 8 indefinite relations may happen based on the disjunction of the three basic relations ($2^3 = 8$). The point-based model suits well for the events of known duration, but falls weak when duration is unknown. In interval-based model, atomic media events are considered as intervals instead of instances. The 13 relations [ALL83] can be presented as 7 cases: "meet", "before", "after", "overlay", "during", "equals" and "finishes", because 12 of them are inverses (except "equal"). For example, "before" is the inverse of "after", and "meet" is inverse of "is met by". This model assumes all the events last an amount of time. The instantaneous event can always be decomposed into a series events with very small intervals. Conceptually, these interval relations can be categorized into "sequential" (such as "meet" and "after") or "parallel" (such as "equal" and "during", etc.) relations. The pure interval model is incapable of dealing with metric information such as temporal distance, for instance, "a during b, and *a is 3 seconds later than b*". To solve this expression problem, some interval-based models take endpoints into account, which becomes a hybrid temporal model.

Currently, some temporal models proposed for multimedia are point-based, others are interval-based or hybrid. Typical examples for point-based model are *time-line* model and HyTime[HYT92]. In time-line model, media items are placed as discrete events in order on a time axis which functions as a common temporal measurement rule. This model is very intuitive and easy to use. However, when the temporal relation of two events are indefinite, we have no idea what order they'll be arranged in the time-line. [HIR95] somehow solved this problem by adding a bend end edge to express the indefinite start or end point. The recently developed interval-based models, such as interval-based conceptual model [LIT93] and temporal composition model [LAY95] are all based on Allen's general relational model for representing knowledge in time [ALL83]. These models solve the uncertainty problems in the point-based model, i.e., it's more powerful in expressing future events without knowing precise duration. However, it introduces inconsistency when applied to composite multimedia [LAY95] [LAY96]. When duration of the interval is undetermined, the inconsistency is *qualitative* such as the example in figure 4.3(a), in which "interval a meets interval b", and "interval b meets interval c" is consistent if further constraint "interval c starts interval a" is not given. *Quantitative* inconsistency may occur when duration of the interval is specified such as the example in figure 4.3(b), in which "interval a with duration 30 seconds meets interval b" and "interval a starts interval c" makes the qualitative consistent specification "interval c before interval b" inconsistent if interval c is further assigned duration 60 seconds.



(a). Qualitative inconsistency (b). Quantitative inconsistency

Figure4.3. Temporal inconsistency examples

4.3.1.2 Existing Temporal Inconsistency checking algorithm

Very little research has been done on the temporal inconsistent checking. People paid more attention on the power of expressiveness on the possible temporal relation, but avoided the consistency verification. In most of existing model, temporal inconsistency is either mentioned without going into consistency checking, or remained untouched. This problem is solved in [LAY96] for general multimedia documents. In their algorithm, a temporal constraint network represented in DAG (Directed Acyclic Graph) is built to denote the document by extracting the symbolic temporal relations defined by the author. The qualitative inconsistency is detected by checking whether the DAG contains a circuit. A circuit implies qualitative inconsistency because a future instance can not be the same as the past instance. The quantitative inconsistency is detected by the algorithm that basically compares two endpoint instances of two comparable parallel chains. The duration of the interval is adjusted according to their nature: free or contingent? This algorithm considers the temporal inconsistency caused by both deterministic objects (including discrete and continuous media objects with definite duration) and indeterministic objects (such as a user event, or an interval with indefinite duration).

4.3.2 Modeling SMIL temporal behavior

4.3.2.1 SMIL time model

SMIL time model is a hybrid temporal model, which is based on both event happening instance (such as for discrete media) and time-lasting event (such as a video, or a duration-assigned image). It provides high-level specification for temporal relation among a number of multimedia objects. In SMIL, multimedia information is encapsulated as an element with a set of attributes defining both spatial and temporal information. A SMIL document can be modeled as a tree, in which the temporal sub-tree contains temporal and linking behavior of the media objects. Any sub-tree of the temporal sub-tree can be viewed as a composite multimedia, of which temporal relations are constrained by the element and the characteristics of their common parent (whether it is sequential or parallel). That is, besides six basic types of media objects: video, audio, animation, text, image and textstream, conceptual elements "seq" and "par" are added to the element set that allows the author to specify the temporal relation in a higher level. Semantically, "seq" element contains "meets" and "before" relations in interval-based model, and "par" element contains all the others that are interrelated in parallel. Below these two elements, a composite multimedia is formed. The relation between the media objects in the composite multimedia is defined by the endpoint's relative instance, the duration of media object, as well as the constrained superior relation, i.e. sequential or parallel.

4.3.2.2 Hypergraph model for analyzing temporal relations of SMIL synchronization elements

In the SMIL tree, the upper node *contains* the lower neighbor nodes. The tree structure can only express the logical structure of the multimedia document. Many nodes contain not only static information, such as "author", "copyright", "id", etc. but also dynamic information, such as temporal and linking information. Such a tree graph is not capable of conveying the rich context that the node contains. In another word, it's hard to detect the duration error that the node may contain in such a tree graph. Furthermore, since it's a hybrid model in which temporal nodes contain both duration and endpoints time information, we can take advantage of it; and use hyper-graph to represent the temporal relation between any synchronization elements.

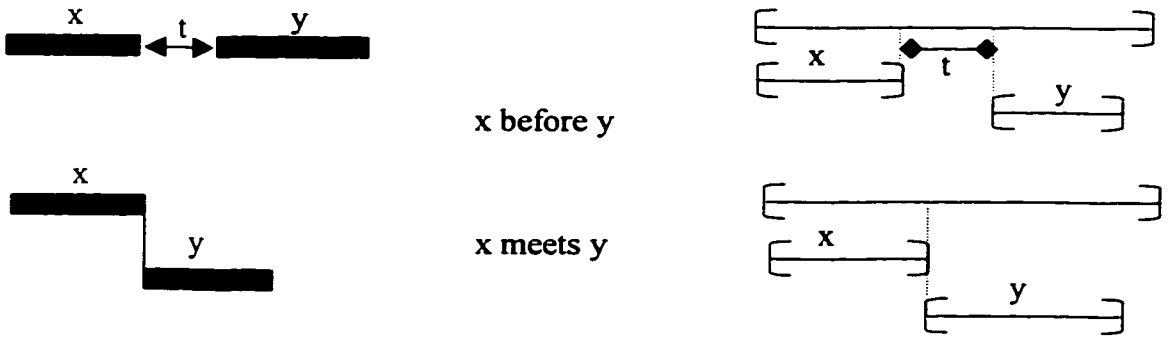
A *hypergraph* H = (E,V) is an extension of graph, in which the edge E is replaced by the *hyperedge*, that is, each hyperedge can be connected to more than two vertices V. A SMIL temporal element (node) can contain changing information I = f(t); however, from the viewpoint of the author, the temporal behavior of such an element can be defined as a begin point, and an end point (duration is the time difference between them), without bothering himself about how the contents change in the duration. Therefore, instead of using single node to represent a temporal element, we use two vertices[1] to represent the begin point and end point, and a hyperedge representing duration between them. Furthermore, the connection that implicates the equal temporal instances between

[1]Vertex is used hereafter in hypergraph in order to distinguish the "node" in the tree.

two neighbor elements is represented by a dashed-line, which connects any point of the referee element's hyperedge to the vertex of the element that receives the *element event*. The syntax and semantics of element event is defined in the SMIL [SMI98] specification 4.2.1. The connection that implicates a synchronization event generated by the "endsync" attribute of a "par" element is represented by a solid-line arrow pointing from the end vertex of "par" element to the end vertex of the element that stops "par" element. To be more distinguishable, the begin point is represented by a left bracket, and the end point is represented by a right bracket.

The seven types of temporal relation [ALL83] between two elements can hence be expressed as the right side graph in the following figure:

For sequential relation:



For parallel relation:

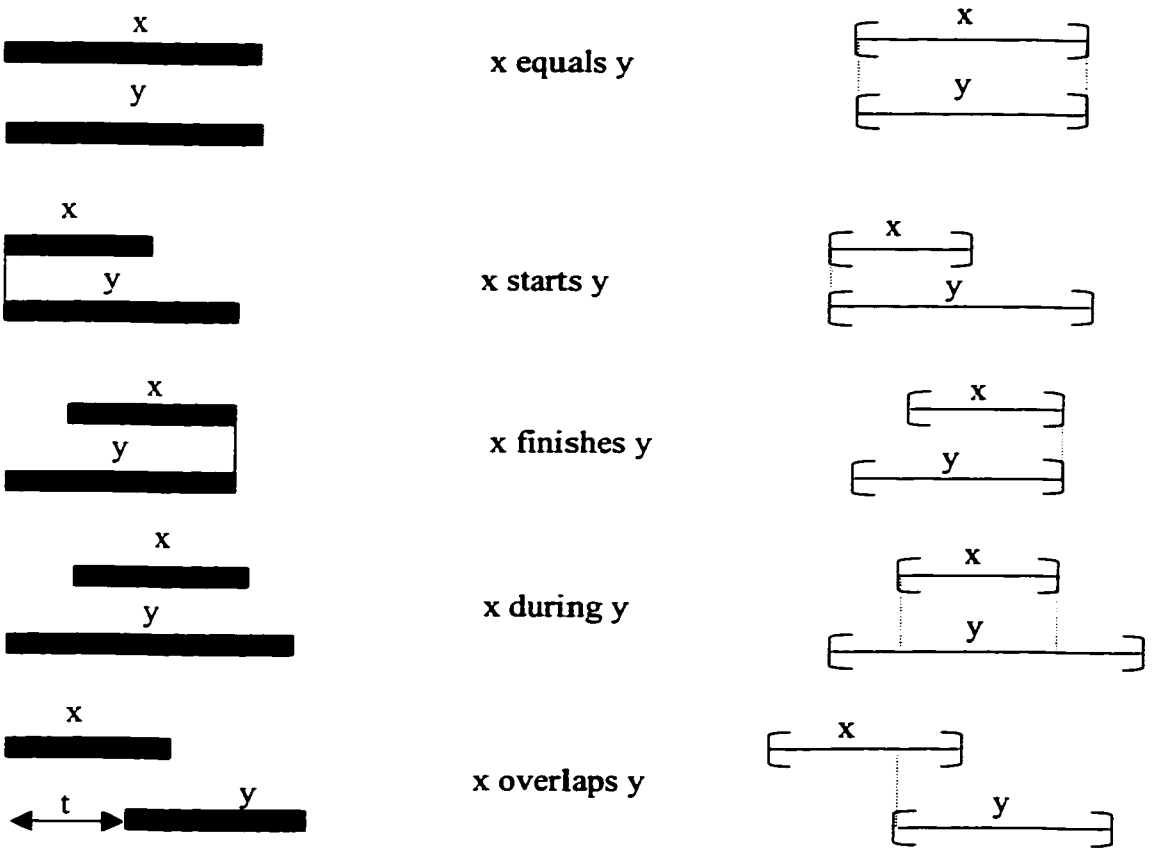


Figure 4.4 The hyper-graph representation for Allen's 7 types of temporal relation

4.4 Semantics inconsistency analysis

SMIL specification 1.0 allows the author to define the user's interaction by a hyperlink. The linking behavior is associated with a whole specific synchronized element, the spatial subpart of the visual media object element or the temporal subpart of the visual media object element. Although a hyperlink is activated by the end-user in unexpected future time, it does not affect the temporal relation among the media objects of the whole presentation. It's different from the indeterminstic case that is prompted in

[HIR95]. This characteristic reduces some inconsistency cases caused by *indeterminism*. The high-level interactive abilities (such as pause, forward, and rewind, etc.) can be extended by the SMIL player. However, our consistency checking will work on what the author specifies at the authoring stage, without considering the indeterministic user interaction on higher-level above SMIL documents. The unexpected latency caused at rendering stage, such as network delay, will not be in our view of examination.

4.4.1 Consistency requirements

Conceptually, the interval relations among the media objects can be either sequential or parallel. Furthermore, a parent node always defines the general rules or behavioral regulation that all its children should obey.

In the SMIL temporal sub-tree, a parent node decides the relation among all its direct children, that is, either sequential or parallel (excluding switch element and hyperlinking elements). Regardless of which relation the parent element implies, the basic temporal consistency requirement for a two-level tree is that the begin of any child can never be earlier than the begin of its parent, while the end of any child can never be later than the end of its parent. Additionally, an event source can not be virtual. Any instant earlier than the begin point of an event or later than the end point of the event is considered "virtual instant".

4.4.2 Assumptions

To analyze the potential temporal inconsistency in the SMIL time model, we assume that the author provides enough temporal information on each synchronization element. That is, at least any two of the begin, end and duration values are assigned to the element's attributes so that the other value can be obtained through the following equation:

<begin-value> + <dur-value>= <end-value>

If the attribute "repeat" is assigned a value, the equation above should be modified as:

<begin-value> + <dur-value>*<repeat-value> = <end-value>

Apart from the "end" attribute, the effective end-value of a "par" element can also be obtained by the value of the attribute "endsync" whose value is determined by the children and can be "first", "last" or "ref-id". The effective end-value of media objects can be affected by the value of attribute "fill". Moreover, if a media object's duration is not assigned, it can always be obtained by its implicit duration. That is, for discrete media object, its implicit duration is 0, while for continuous media object, we can use some tool to get its intrinsic duration that is determined when it's recorded. A discrete media object without explicit duration will never cause temporal inconsistency, but we consider it as an error at the authoring stage because its implicit meaning is "never be shown" if its parent implies the *sequential* relation.

We also assume that the hardware supports multi-channel for sound. Thus, it's not an error to have multiple audio files in parallel.

4.4.3 Error classification

Most of semantics errors in SMIL are caused by temporal inconsistency. According to the different conflicting types, the inconsistency can be classified into pure temporal inconsistency, linking-temporal inconsistency or spatial-temporal inconsistency. We'll discuss possible inconsistencies in all these three categories respectively.

4.4.3.1 Pure temporal inconsistency

The SMIL tree structure is propagated from the root to the leaves or from parent to children. A SMIL tree can be reconstructed into multiple two-level trees. Therefore, we can examine the temporal consistency of the entire document by checking the two-level trees. In this section, we consider the pure temporal inconsistency without linking or spatial conflicting. In another word, we only look into two types of two-level trees depending on the element type of the parent node ("par" or "seq"). According the SMIL specification, "body" element implies sequential relation between their children. It's hence considered as "seq" type in our analysis.

As defined in SMIL specification, the begin value and end value can be two types of values: *delay-value* and *event-value*, and the dur value can be a *clock value* or the string "*indefinite*". According to their semantics definitions, different types of values may cause different temporal inconsistencies.

Case 1: Node "par" and its children:

Case 1.1 Consider that the delay-value type is the only type used to express the begin, dur or end value:

The delay-value of "par" element's children defines a delay from the effective begin of "par".

i. Quantitative error: We check the basic quantitative consistence in the each element including the parent "par" element. That is, the begin, end and dur value should satisfy the equation: <begin-value> + <dur-value> = <end-value>, if all three values are assigned by the author. Otherwise, the unassigned value is calculated based on the equation. If the obtained value happens to be a negative value, the error happens.

ii. Late-begin warning: At least one of all the children's begin instant is equal to "0", which means the child element will begin at the same time as the parent "par". Otherwise, there will be a period of time when there is nothing to show.

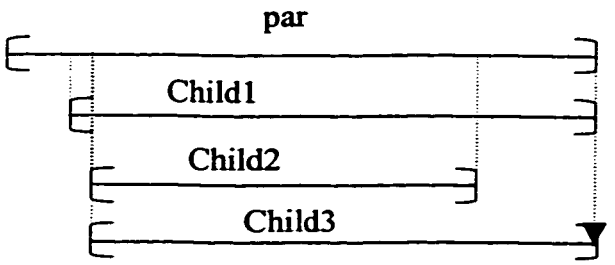


Figure 4.5 Hypergraph for an example of late-begin warning

iii. Early-end warning: At least one of all the children's end instants is equal to the end of its parent "par". Otherwise, there will be a period of time there is nothing to be shown.

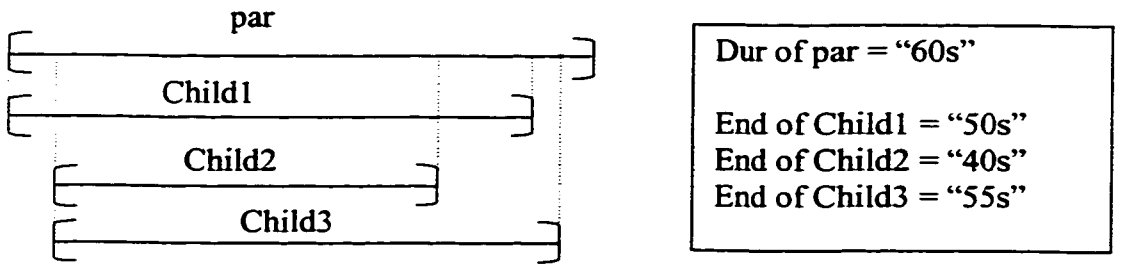


Figure 4.6 Hypergraph for an example of early-end warning

iv. Begin error: The begin instant of any child is later than the end of the parent "par". In this case, the child element will never be shown.

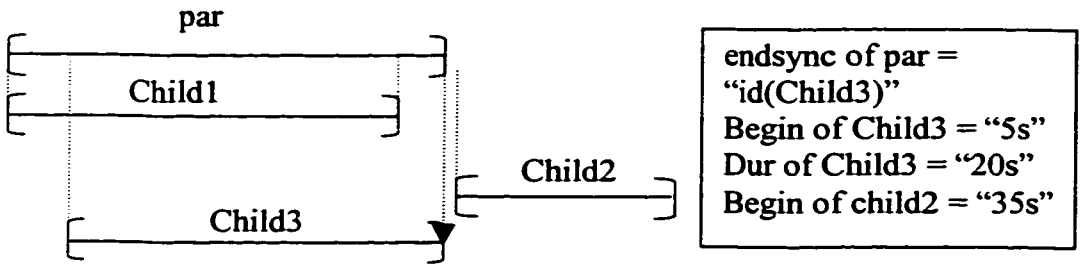


Figure 4.7 Hypergraph for an example of begin error

v. End warning: The end instant of any child is later than the end of its parent "par". In this case, the extension part will not be shown. This can be the author's intention. However, this type of warning may cause an error when Child2 has a child element

that will begin after the extension part of Child2. Consequently, the entire child element of the Child2 will never be shown.

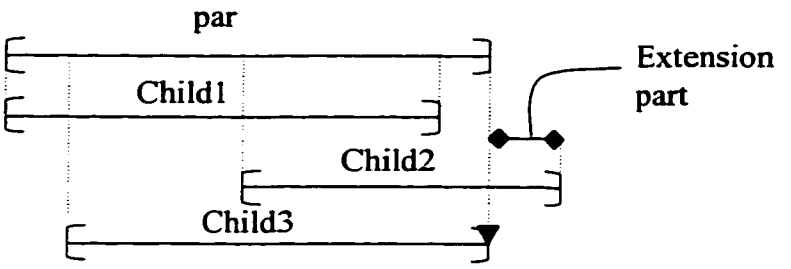


Figure 4.8 Hypergraph for an example of end warning

vi. Begin-end mixed up error: If the element is assigned begin and end value, and the end point is earlier than its begin point, this element is impossible to be handled.

Case 1.2 Consider that the event-value type is used to express the begin or end value.

According to SMIL specification [SMI98], the element-event value has the following syntax:

Element-event	::= "id("Event-source") ("Event")"
Event-source	::= id-value
Event	::= "begin" | Clock-val | "end"

When the event is a "Clock-val" event, the clock gives the elapsed time since the effective begin of element "par" or "seq". For the media object elements, the semantics are implementation-dependent. The clock may either give presentation time elapsed since the effective begin, or it may give the media time of the object. The latter may differ from the presentation time that elapsed since the object's display was started, e.g., due to

rendering or network delays. In our semantics validation, we assume the former semantics for the media object elements because the latter is not deterministic at the authoring stage.

The following semantic errors can happen when event-value type is used to express the temporal attributes:

i. Out-of-scope error: the event-source must be "in scope". The set of "in scope" elements S is determined as the sum of all children, excluding element "a" and element "switch", and all "a" elements' children except "switch" element. If any event-source is not in this scope, an out-of-scope error happens.

ii. Clock-value virtual event error: If the event is a clock-value event, and the clock value is greater than the duration of the event-source, the element can never be shown because of linking lost when this happens to "begin" attribute. If this happens to the "end" attribute, the end point can not be determined based on the semantic definition on "element event-value" of SMIL specification. In both cases, an error happens.

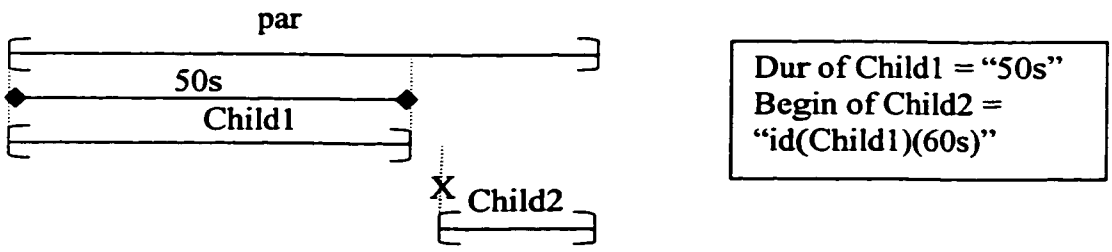
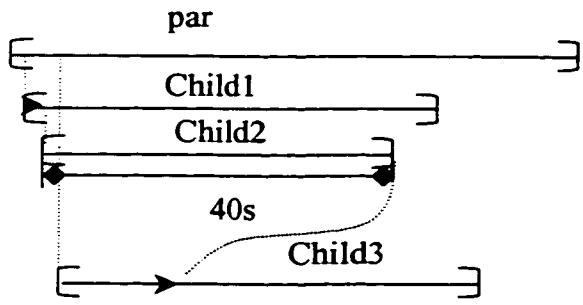


Figure 4.9 Hypergraph for an example of clock-value virtual event error

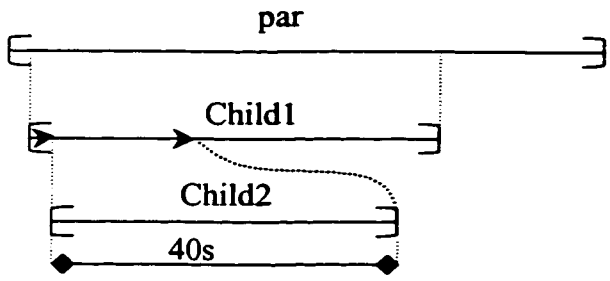
iii. Clock-value quantitative error: This type of error happens only when begin, dur and end value of an element are all explicitly assigned. The rule is that: <begin-value> + <dur-value> = <end-value>. Since the begin and end value may have different types (event value or clock value), they have to be converted to the same type and synchronized with the same clock to be operated in the constraint equation. The following scenarios show the different cases that the event-value types are used. In essential, they all belong to *quantitative* inconsistency discussed previously. Quantitative inconsistency is temporal inconsistency with respect to duration.



Begin of Child1 = "3s"
Begin of Child2 = "id(**Child1**)(3s)"
End of Child2 = "id(**Child3**)(10s)"
Dur of Child2 = "40s"
Begin of Child3 = "8s"

Observe that: (Synchronize with par)
(3s + 3s) + 40s ≠ (8s + 10s)

Scenario 1: The begin and end of the element Child2 refers to different elements



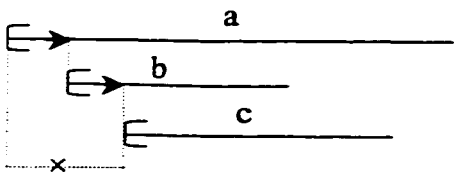
Begin of Child2 = "id(**Child1**)(1s)"
dur of Child2 = "40s"
End of Child2 = "id(**Child1**)(21s)"

Observe that: (Synchronize with Child1)
1s + 21s ≠ 40s

Scenario 2: Both the begin and end of the element Child2 refers to the same element

Figure 4.10 Scenarios of clock-value quantitative error

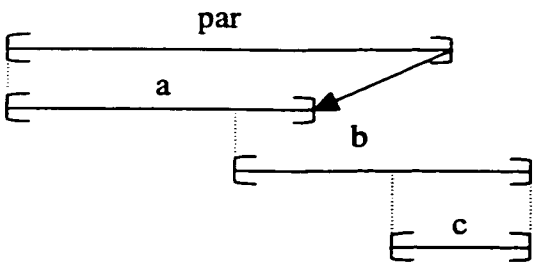
iv. Qualitative error: If the begins of the elements are referred to each other, and introduce a backward pointing on the timeline, it's a qualitative error because the future instant can not be equal to the instant in the past.



Element b: begin = "id(a)(3s)";
Element c: begin = "id(b)(4s)";
Element a: begin = "id(c)(begin)";

Figure 4.11 Hypergraph for an example of qualitative error

v. Begin error: If the begin of the child element is later than the end of the parent, the child element will never be shown. In this example, element c will never be shown.



Par: endsync = "first"
a: begin = "0s", dur = "60s"
b: begin = "id(a)(40s)"
c: begin = "id(b)(30s)" end = "id(b)(end)"
Observe that:
Begin of c is later than "par".

Figure4.12 Hypergraph for an example of begin error

Case 2: Node "seq" and its children:

Case 2.1 Only the delay value is used as the type to express the value of begin, dur or end.

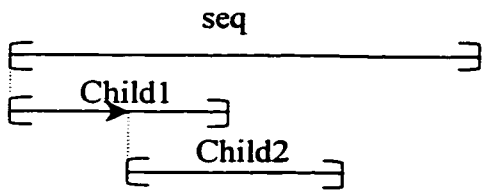
i. Quantitative error: This is the same error as case 1.1 (i).

ii. Ending warning: Check the end value of each child element, if any of them is later than the end of the parent "Seq", then it's an ending error.

iii. No show error: When the "dur" or "end" value is assigned to the parent "seq", check the begin value of each child. If any of the children starts later than the end of the parent, this child element will never be shown.

Case 2.2 Event-value is used as the type to express the value of begin , dur or end.

i. Overlapping error: If the begin of any child is in between the begin and the end of another child element, then the overlapping error happens. If this is truly the case that the author hopes to present, then a "par" tag should be used instead of "seq". According to the SMIL semantics definition of the "begin" attribute of the "seq" element, this type of error may probably happen when the author chooses the event-value type to assign value to "begin" attribute. It's my opinion that the author is recommended **NOT** to use event-value type when specifying "begin" attribute of "seq", or the semantics definition should change.



Begin of Child1 = "0s"
End of Child1 = "20s"
Begin of Child2 = "id(Child1)(10s)"

Observe that:
0s < 10s < 20s

Figure4.13 Hypergraph for an example of overlapping error

ii. Out-of-scope error: The element generating the event must be "in scope". The set of "in scope" elements S is different from what is defined on the par element and its children, because the children of element "seq" should be in a temporal order. Therefore, the set S is all the child synchronization elements that appear earlier than

the child element that receives the event. If the event-source is not in the scope, an out-of-scope error happens.

iii. Quantitative error: When the begin, end and dur values are all assigned, check the quantitative consistency, that is, whether they satisfies: <begin-value> + <dur-value> = <end-value>

4.4.3.2 Linking-temporal inconsistency (empty association)

As stated before, media object elements can have anchor elements as their children. Anchor element divides the specific media object into spatial or temporal subparts with different id so that each subpart can be associated with a hyperlink. It's an error when the begin of the anchor element is later than the end of the media object element because this link can never be activated. In the following example, the temporal subpart "picture" has an empty association with video element.

```
<video src= "http://deneb.genie.uottawa.ca/Video" begin = "0s" end = "100s">
  <anchor href= "http://deneb.genie.uottawa.ca/Video" id = "video" begin = "0s" end = "5s">
  <anchor href= "http://deneb.genie.uottawa.ca/fancyPicture"  id = "picture" begin
= "150s" end = "200s">
</video>
```

4.4.3.3 Spatial-temporal inconsistency

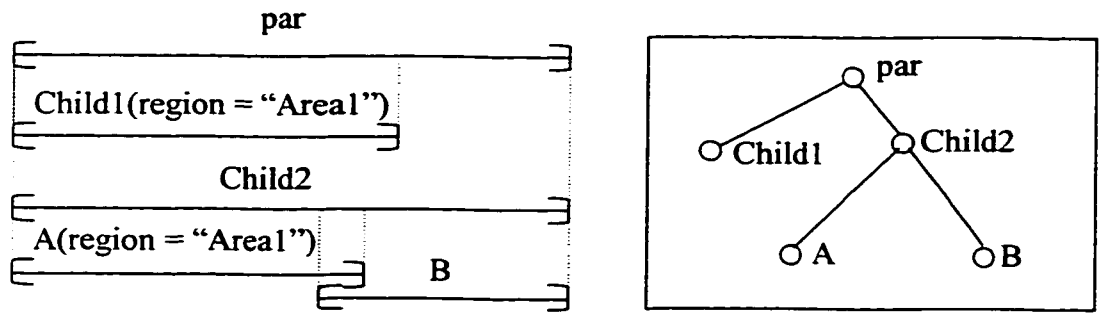
In SMIL document, spatial information is defined in the *region element* with an *id* in *spatial sub-tree,* while the synchronization element specifies its display area through *region attribute* whose value is an id defined in the region element in the spatial sub-tree. There will not be a spatial conflict in the sequentially displayed media objects. But when

the media objects are scheduled to be displayed in parallel, if two or more media objects are expected to display at the same time period on same display areas i.e., with same region value, only one media object is visualized. In another word, the same spatial region of the same window can only display exactly one media object at the same time duration, that is, all the other media objects that are supposed to be shown at the same time are shadowed. Hereafter, we will refer this case as a *spatial competition error*.

The following example gives the typical scenario. In this example, the author expects the text to be shown simultaneously with the video yet using the same display area. Unfortunately, only one of them can be visualized at the same time.

```
<par>
<video id = "video" src= "http://deneb.genie.uottawa.ca/niceVideo.mpg" region= "area1"/>
<text src= "http://dened.genie.uottawa.ca/niceText.txt" region= "area1" begin=
"id(video)(begin)" end= "id(video)(end)"/>
</par>
```

In the example above, two competing media objects are direct children of the same parent. It's easy to be detected in the two-level tree. In some case, two media objects in different two-level trees may compete for the same display region at the same time, too. The following figure 4.14 shows the scenario. In this example, Child1 and Child2 are direct children of element "par". Element A and element B are direct children of Child2. Indirectly, Child1 has the relation "start" with A and is expected to be displayed in the same region "Area1" as A during the same time period. This is impossible.



Temporal relation in hypergraph　　Logical structure in tree

Figure4.14 A scenario of spatial-temporal inconsistency

4.5 Temporal inconsistency checking algorithm

To check the temporal inconsistency between the synchronization elements in the SMIL document, we first construct the SMIL tree from the document, and work on the sub-tree starting from node "body".

4.5.1 Pure temporal inconsistency checking algorithm

This algorithm will check the pure temporal inconsistency in the temporal sub-tree. Since media object elements can only contain "anchor" elements as their children, we ignore the two-level tree that the parent node is a media object element. In section 4.5.1.1, we only consider the case that the parent node is either "par" element or "seq" element. The case when "switch" element or "a" element is met as a parent node will be discussed in the algorithm modification as of section 4.5.1.2.

The basic idea for the algorithm is to construct every deterministic two-level tree, and check the consistency within the tree. If a tree is not deterministic, we traverse to the

children of the undetermined element until a determined tree is found and check. Then we go back through the traversal road. Most of the errors mentioned in the section 4.4.3 happened within the two-level tree. However, there are some cases that the temporal relation is consistent within individual two-level tree, but it is actually not consistent when combine them together. Case 1.1 (v) in section 4.4.3.1 discussed such a scenario. Our algorithm can detect such an error.

4.5.1.1 Pure temporal semantics error detecting algorithm

Stage1: Construct two-level trees

In this stage, we construct the two-level trees from the temporal sub-tree starting from element "body". We call this tree the original tree. Initially, none of the nodes in the tree are marked "parent". Once the temporal consistency is checked on the tree, the parent node is marked "parent" and avoids being checked again.

1. **Traverse** from the root ("body" node), left to right and top to bottom.
2. **Copy** the first *non-parent* node and all its direct children to form a two-level tree.
3. **Index** the child node from 0 to i.

Figure 4.15 gives an example of constructing all the two-level trees from the temporal sub-tree.

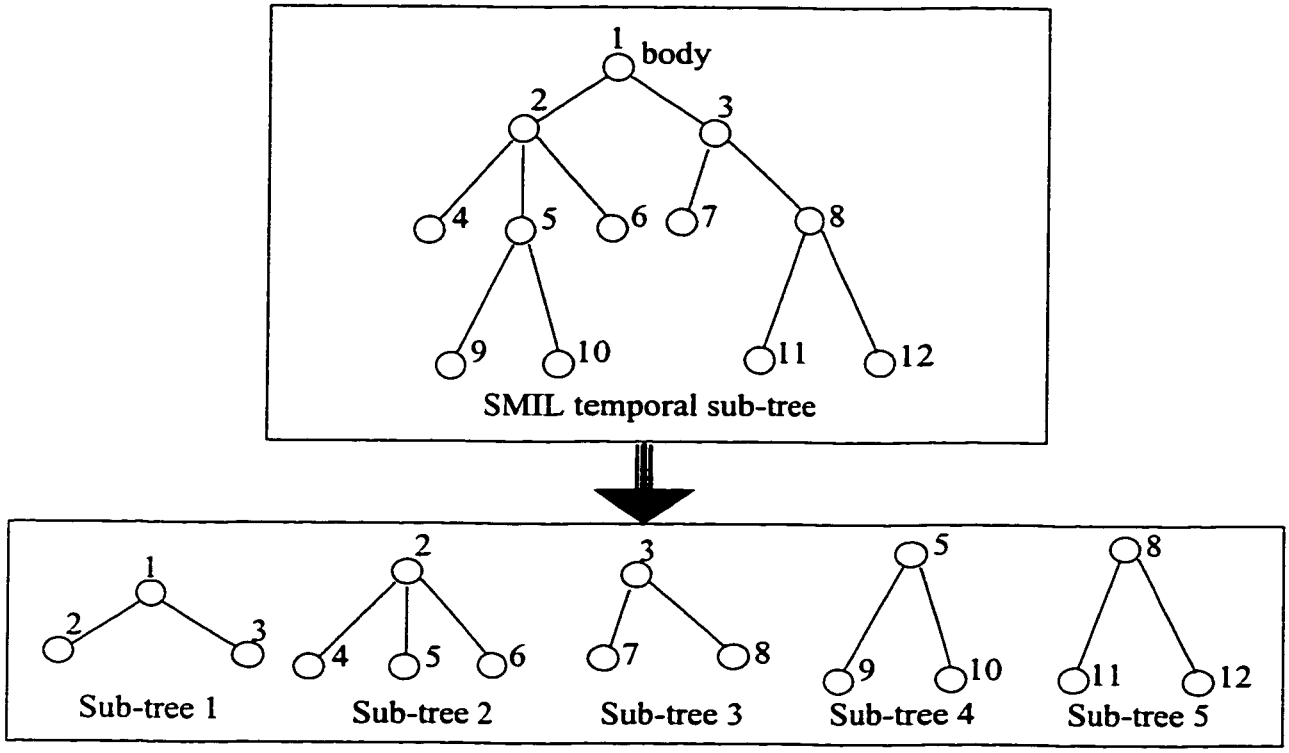


Figure 4.15. Construction of two-level trees from a SMIL temporal sub-tree

Stage 2: Determine the temporal values of each element in the two-level tree

Definition1: If the duration of the node can be determined based on the assigned values of its attributes, this node is called ***deterministic node.***

Definition2: If the two-level-tree contains only deterministic nodes, this tree is called ***deterministic tree.*** Otherwise, if any node can not decide its duration within the two-level tree, the tree is ***non-deterministic.***

Notice that we can always decide a child node's begin with respect to the parent node's begin. If a child node's begin is not explicitly assigned, its effective begin can be

decided by its implicit begin[2]. Therefore, if at least one of the dur or end value can be decided, all the begin, dur and end value of the node is deterministic based on the constraint relation among them. An interesting fact is that, if all the child nodes are media object elements, this two-level tree is **deterministic**. This is because the duration of the media object element can be obtained either by the explicitly assigned value or their implicit duration value since intrinsic duration can always be obtained with some tool.

However, if either "dur" or "end" value is not assigned to a *discrete* object, such as image, text or textstream, we consider it as an error since it's not going to be shown although it won't introduce temporal inconsistency.

Step1: Check whether the tree is deterministic or not. If it is deterministic, go to **stage 3**, otherwise, continue with Step2.

DO j = 1, i

/* for each child element C_j, identify element types:

If it belongs to media objects, **then** label it as "leaf".

Else if it is "par" or "seq", **then**

Search attributes "begin", "dur" and "end" of C_j:

If less than two of them is found, **then**

Label the tree "**non-deterministic**"

Break /* continue with Step2 */

Label the tree "**deterministic**".

Step2: Construct the two-level tree with the non-deterministic node as parent.

---

[2] As defined in SMIL specification, the *implicit begin* of the first child of the "body" element is when the document starts playing; the implicit begin of a child of a "par" element is equal to the effective begin of the "par" element; the implicit begin of the first child of a "seq" element is equal to the effective begin of the "seq" element; and the implicit begin of any other child of a "seq" element is equal to the desired end time of the previous child of the "seq" element.

1. **Get** the child nodes of C_j and **construct** the two-level tree.
2. **Goto** Step1 of this stage.

Stage 3: Testing temporal consistency in a deterministic two-level tree

Due to the characteristics of the synchronization elements, different type of elements may have some auxiliary attributes for the author to describe the temporal behavior. The effective temporal behavior of the element will not only depend on the basic temporal attributes, such as "begin", "dur", or "end", but also rely on the auxiliary attributes such as "repeat", "endsync", or "fill". The following table 1 shows what the temporal attributes that different synchronization elements may have.

Attribute / Element	begin	dur	end	repeat	endsync	fill
par	yes	yes	yes	yes	yes	no
seq	yes	yes	yes	yes	no	no
Media object	yes	yes	yes	yes	no	yes

Table 1. Valid temporal attributes for synchronization elements

Since it is impossible to compare two values in different expression types, we first convert the temporal values to a common clock after extracting all the temporal attributes and the corresponding values from each element. The common clock for the nodes is the begin clock of the parent node.

Suppose the two-level tree has i leaves (child elements). Let Begin(P), Dur(P), and End(P) stand for the begin, duration, and end of the parent element, and Begin(C_j),

Dur(C_j) and End(C_j) stand for the **synchronized** begin, duration and end of the jth child node. Initially, these values are equal to the first value that is available in the following expression:

```
Begin(P) = <begin-value> Else "0"(if <begin-value> is not assigned)
Dur(P) = <dur-value>*<repeat-value> Else null /*default value of repeat is "1" */
End(P) = Sync(<end-value>) Else null
Begin(Cj) = Sync(<begin-value>) Else Sync(<implicit-begin value>)
Dur(Cj) = <dur-value>*<repeat-value> Else (<intrinsic value>) Else null
End(Cj) = Sync(<end-value>) Else null
```

The function Sync(<any-value>) converts the <any-value>'s clock to the common clock, i.e., the clock of the parent's begin clock. The algorithm for calculating Sync(<any-value>) is as follows:

```
If <any-value> is a delay-value type
        If parent node is "par" element OR the first child of "seq" element
                Sync(<any-value>) = Begin(P) + <any-value>
        Else if parent node is "seq" element
                Sync(<any-value> of Cj) = Sync(<end-value> of Cj-1)
Else /*<any-value> is an event-value type
        DO j = 1, I
                If Id(<any-value>) = = Id(Cj)
                        If Event(<any-value>) = = "begin"
                                Sync(<any-value>) = Sync(<begin-value> of Cj)
                        Else if Event(<any-value>) = = "end"
                                Sync(<any-value>) = Sync(<end-value> of Cj)
                        Else /* event of the element-event value is a clock value */
                                If Clock-val(<any-value>) > Dur(Cj)
                                        goto Stage5 (Error 3)
                                Else
                                        Sync(<any-value>) = Sync(<begin-value> of Cj)
                                                + Clock-val(<any-value>)
                If Id(Cj) not found, goto Stage5 (Error4)
```

According to the property of the parent node, we have different procedure to obtain the temporal attributes of each element.

Case 1: Parent node is "par"

If Dur(P) != null **AND** End(P) != null
If Begin(P) + Dur(P) != End(P) **goto** Stage5 (Error1)
If Dur(P) != null **then** End(P) = Begin(P) + Dur(P)
Else if End(P) != null **then** Dur(P) = End(P) – Begin(P)
First_Begin = End(P), Last_End = Begin(P)
Do j = 1, i
If (Dur(C_j) = = Sync(<dur-value>) **and** End(C_j) = = Sync(<end-value>)
If Begin(C_j) + Dur(C_j) != End(C_j) **goto** Stage5 (Error1)
If Begin(C_j) > End(P)
Goto Stage4 (Error2)
If End(C_j) = = Sync(<end-value>)
Dur(C_j) = End(C_j) – Begin(C_j)
If Dur(C_j) < 0, **goto** Stage5 (Error6)
Else End(C_j) = Begin(C_j) + Dur(C_j)
If Begin(C_j) = = [Begin(P), End(P)]
First_Begin = **Min**(First_Begin, Begin(C_j))
Last_End = **Max**(Last_End, End(C_j))
If End(C_j) > End(P)
End(C_j) = End(P), Dur(C_j) = End(C_j) – Begin(C_j)
Copy the Dur(C_j) to the <dur-value> of the same element in the original tree
goto Stage4 (Warning 1)
If first_Begin > Begin(P) goto Stage4 (Warning 2)
If last_End < End(P) goto Stage4 (Warning 3)
Else /*Neither dur-value nor end-value is assigned */
ID = Id(<endsync-value>)
DO j = 0, i
If Id(C_j) = = ID
End(P) = End(C_j)
Recursively invoke the algorithm on Stage3
Mark the parent node in the original tree "parent"
Goto Stage1 /* no error detected in this tree */

For example, in a subpart of the SMIL document, we have a scenario as follows:

```
<par begin="3s" endsync="last">
        <audio src="http://deneb.genie.uottawa.ca/audio.au" id="au" begin="0s" end="15s"/ >
        <video src="http://deneb.genie.uottawa.ca/video.mpg"
begin="id(au)(begin)" end="20s"/>
</par>
```

In this two-level tree, we calculate the temporal attributes as the following procedure:

Begin(P) = 3s
End(P) = "last" = Max(End(C_1), End(C_2))
End(C_1) = (3s + 15s) = 18s
End(C_2) = (3s + 20s) = 23s
End(P) = Max(18s, 23s) = 23s
Dur(P) = End(P) – Begin(P) = 20s
Begin(C_1) = (3s + 0s) = 3s
Dur(C_1) = Sync("15s") – Begin(C_1) = (3s +15s) – 3s = 15s
Begin(C_2) = Begin(C_1) = 3s
Dur(C_2) = Sync("20s") –Begin(C_2) = (3s + 20s) – 3s = 20s
No error is detected in this tree.

Case 2: Parent node is a "seq"

If Dur(P) != null **AND** End(P) != null
If Begin(P) + Dur(P) != End(P) **goto** Stage5 (Error1)
If Dur(P) != null **then** End(P) = Begin(P) + Dur(P)
Else if End(P) != null **then** Dur(P) = End(P) – Begin(P)
DO j = 1, I
    If (Dur(C_j) = = Sync(<dur-value>) **and** End(C_j) = = Sync(<end-value>)
    If Begin(C_j) + Dur(C_j) != End(C_j) **goto** Stage5 (Error1)
  DO k = 1, j
If Begin(C_j) < End(C_k) **goto** Stage5(Error5)
    If End(C_j) > End(P)
        End(C_j) = End(P), copy this value to the original tree;
        goto Stage4(Warning 1)
    If Begin(C_j) > End(P) **goto** Stage5 (Error2)
Else /* Neither the dur nor the end attributes of "seq" is assigned */
  End(P) = End(C_i)
  Dur(P) = End(P) – Begin(P)
  Recursively invoke the algorithm on Stage3
Mark the parent node in the original tree "parent"
Goto Stage1 /* No error is detected in this tree */

Stage4: Warning reporting

The warning message is given without stopping the error checking procedure.

1. Warning messages:
Warning1: Part of the element won't be shown. This may cause the child of the element not to be shown as error2.

Warning2: None of the child elements of "par" starts at the beginning of "par". There will be a period of time there is nothing to be shown.

Warning3: All the child elements of "par" finish earlier than "par". There will be a period of time when there is nothing to be shown.

2. Go back to where it invokes the warning.

Stage5: Error reporting in category:

Report the element that causes the error and give the corresponding error messages:

Error1: The begin, duration and end of the element can not satisfy the constraint condition to which they are confined.

Error2: The begin of the element is later than the end of the parent which means this element will not be shown.

Error3: The event is pointing to the virtual element because the clock value is greater than the effective duration of the element generating the event.

Error4: The event id is out of scope, the author is advised to check whether the value id is within the "scope" defined in the SMIL specification.

Error5: The element is a child of "seq" element, but it overlaps with other child element. Either the temporal value is incorrectly assigned by the author, or the author is advised to use "par" element as parent.

Error6: The begin and end values of the element must have been mixed up. It doesn't make sense that the element is expected to finish before it starts.

Done!

4.5.1.2 Algorithm modification

Last section discusses the algorithm ignoring the elements "switch" and "a". We do the following modification if they occur in the SMIL document:

1. **If** an "a" element is accounted, we **mark** it "parent" and **replace** it with its child node when constructing the two-level tree at stage 1.
2. The "switch" element allows an author to specify a set of alternative elements from which only one acceptable element should be chosen. Therefore, at the stage1 of the algorithm, **if** a "switch" element is accounted as a child node, we **mark** it "parent"

and replace it with its direct child node one by one, and go over one cycle each time the node is replaced.

4.5.2 Linking-temporal error checking algorithm

If the parent node of the two-level tree is a media object element:
DO j = 1, i
If Begin(C_j) > End(C_j) **report** a "begin-end" mixed-up error
If Begin(C_j) > End(P) **report** a virtual anchor error

4.5.3 Spatial-temporal competition error checking algorithm

We detect the spatial-temporal competition error depending on the region id that the synchronization element specifies. This type of error will only happen when the two media objects are in parallel relation (including "duration", "overlap", "equal", "start" and "end"). These relations can be represented in time point (begin and end) as shown in the following table2.

Relation	Expression in time point
x equals y	(Begin(x) == End(y)) & (End(x) == End(y))
x overlaps y	(Begin(x) < Begin(y)) & (End(x) >Begin(y)) & (End(x) < End(y))
x during y	(Begin(x) > Begin(y)) & (End(x) < End(y))
x starts y	Begin(x) ==Begin(y)
x finishes y	End(x) == End(y)

Table 2. Five types of parallel relation expressed in time point

In the following algorithm, we still work on the temporal sub-tree. Moreover, it is not necessary to allocate a display area for an audio; therefore, we don't consider it as an error if one of the two space-competing objects is an audio.

Stage 1: Find the set of media objects that may be in parallel.

1. **Traverse** the temporal sub-tree from top to bottom in level order.
2. **Find** the first non-parent "par" node.
3. **Add** all of its direct children to the **set**, which initially is set to null.
   3.1 **If** a child node is a "seq", "par" or "a" element, **remove** itself from the **set**, and then recursively **invoke** the algorithm on step3.
   3.2 **If** a child node is a "switch" element, **add** and **replace** one of its direct children till the last one, and recursively **invoke** the algorithm on step3.

Stage 2: Detect the Spatial-temporal competition error.

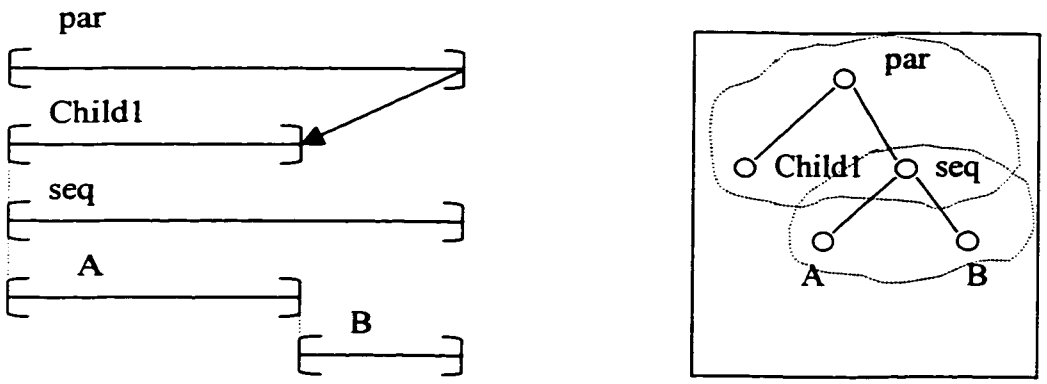
1. **Decide** the temporal relation of any two non-audio type children.
   If they are in parallel, **then compare** the region attribute values.
   If the same, **report** the error.
   Else recursively invoke the algorithm

SMIL also allows the author to specify the same display region specified with different layers in diverse region elements. Therefore, if the regions of two parallel media objects are specified with different id values but in fact physically the same display area, our algorithm won't be able to detect it. We ignore this case because this could be the purpose of the author.

4.6 Important issues in the algorithm

There are a number of interesting yet important issues in the algorithm. Firstly, the tree was traversed starting from the top of the tree instead of the leaf. This guarantees that combined with the second issue *modification*, an inconsistency error hidden in a multiple level tree will be detected with our algorithm.

Secondly, when the two-level-tree is found to be consistent, the temporal values of the children are modified. The modification step is necessary. It can be seen in the following example shown as figure4.16. It's obvious that the temporal relation is consistent within the two two-level trees respectively. When applying our algorithm, it's important to modify the child's temporal value (in this case, the seq's end is modified to the par's end, or, Child1's end). Thus the error can be detected when examining the two-level tree with "seq" as the parent.



a: hypergraph for temporal relation **b:tree for logical structure**

Figure 4.16 A scenario for temporal error detectable when combining two two-level tree

Thirdly, when the warning is given, the validation continues. However, if an error is found, the validation stops after reporting the error. This could be inefficient than to find out "all the errors" and then stop. Yet, in many cases, the modification of one element may change the relation between it and the others. Therefore, we stop our semantic checking at the first error we find.

Chapter 5

Implementation

5.1 Technology background

5.1.1 Designing methodology

We use UML Object-oriented methodology [RUM99] to express our design for our thesis work. In UML notation, the structure of the classes that represents the architecture of the system is described in the *class diagram*; the mechanisms used to regulate how objects collaborate is represented by the *object diagram*; and the dynamic semantics of a problem or its implementation can be expressed through *state transition diagram* or *interaction diagram*. Figure5.1 shows some examples of notation that will appear in the thesis.

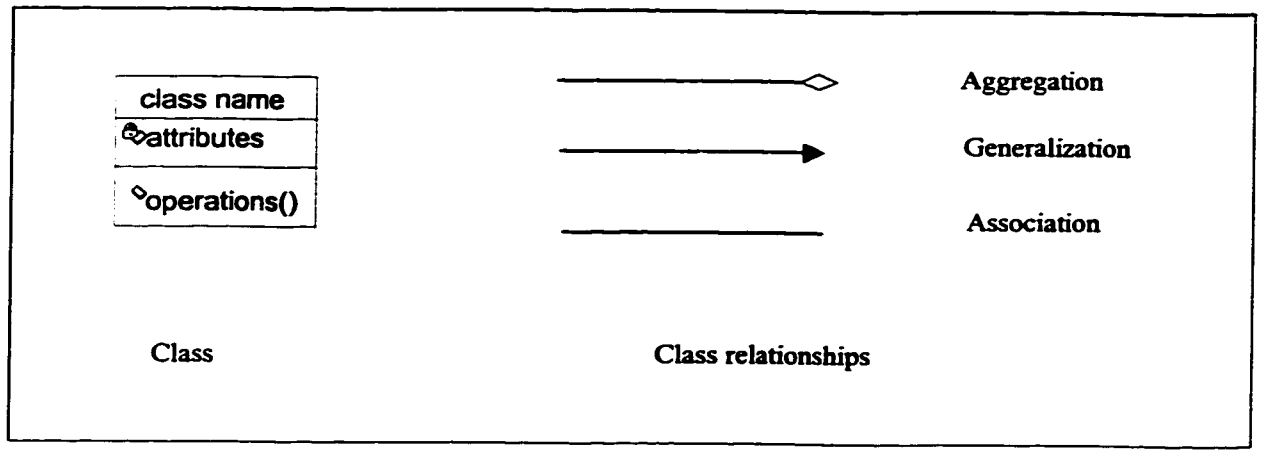


Figure5.1 Examples of UML notations on class and class relationships

5.1.2 Software used

5.1.2.1 Java language

Java is a newly developed object-oriented language. The Java language and its companion class libraries provide a portable, interpreted and high-performance development environment. Many Internet applications adopt Java as the language to achieve their goals ranging from improving web-based GUI to implementing a large Internetworking system. The obvious advantage of Java over other languages when fulfilling an Internet based system is that Java code is understandable by popular browsers, such as Internet Explorer and Netscape. This saves developers much time in developing a particular browser. Another reason why we choose Java to implement our system is that it is easy to learn yet powerful. Let's briefly inspect two cases to see how easy it is in Java to handle events and racing conditions:

Event handling:

In JDK 1.1, each UI component has a corresponding *ActionListener* to associate with. The ActionListener is an *interface* (a special type of class that is completely abstract in Java language) provided in the JDK packages. Before instantiating a listener object, the listener class implements the ActionListener *interface*, and the reaction to an event must be implemented in the appropriate method in the class. Then the listener object is added to the UI component just as simple as the following lines of codes:

```
Button startButton = new Button("Start");
ButtonListener startListener = new ButtonListener(); //ButtonListener implements
                                                     // ActionListener
startButton.addActionListener(startListener);
```

Synchronized keyword:

Race conditions can be easily solved in Java. Race conditions arise from multiple asynchronously executing threads trying to access a single object at the same time and getting the wrong result. This type of problem can be solved in Java by synchronizing threads around a condition variable or method through the use of monitors. Monitors prevent two threads from simultaneously accessing the same variable. A monitor is associated with a specific data item (a condition variable) and functions as a lock on that data. When a thread holds the monitor for some data item, other threads are locked out and can not inspect or modify the data. To functionalize this, the user just needs to mark the code segments as *synchronized*. As a result, the system associated a unique monitor with every object that has a synchronized method.

5.1.2.2 XML parser

Many organizations and companies have been working on XML parser. Some of them even provide the right of free usage for the user. Microsoft is the pioneer who offers the royalty-free package. Microsoft XML Parser version 1.8 was released in early Dec 1997 and has since been quite stable. To be competitive, Netscape created a team (mozilla.org) that is responsible for modifying and re-distributing the client source code at the end of February 1998. This team released the source code of their XML parser in April 1998, while Microsoft gave the press release on Aug 20, 1998 announcing their cooperation with DataChannel Inc. to deliver XML technology. The new version of Microsoft XML parser can be downloaded from DataChannel site [DAT98]beilei. At the time we started our work, Microsoft XML parser was the only one we could get. We then decided to build our project on it (Microsoft XML parser version 1.8), and it proved to be a good fundamental software for our project in the end.

Microsoft XML parser is an XML validating parser package written in Java provided by Microsoft Corporation. It provides packages of classes and interfaces which support accessing and manipulating the structure of the XML document. Optionally, the user is permitted to check the validity of the document with *internal* or *external* XML DTD. Once the document is parsed, the document is exposed as a tree with a set of Java codes. The nodes of the tree represent the XML elements of the document. In the Microsoft XML parser, the tree elements are objects instantiated from the Element class. From the element, the user can get both the parent element and child elements through the member functions that the Element class provides. Essential information such as

attributes' information can also be accessed through the element's function APIs. Since the SMIL document is XML syntax-based document, this package can also benefit us in manipulating the elements of the SMIL documents. Our SMIL validation tool is built by reusing the Microsoft XML parser.

5.1.2.3 Java Media Framework

The standard Java core packages don't support the operation on most continuous media type data. They only offer APIs to *play* or *stop* the audio files in AU format. The Java Media Framework (JMF) is an application-programming interface for incorporating media data types into Java applications and applets. It's specifically designed to take advantage of Java platform-independent features. In the JMF 1.0 version, APIs for media players are provided to present time-based media such as audio and video, while other features such as media capturing and CODEC (coder-decoder) architecture are not available in this version. However, it supports most of the standard media content types, including MPEG-1, MPEG-2, QuickTime (.mov), AVI, WAV, AU and MIDI, which is good enough for us to test out the synchronized playback with a SMIL presentation with any of these media types.

5.2 Implementation of SMIL validation tool

The functionality of our SMIL validation tool is mainly composed of two steps: syntax validation and semantic validation. The procedure of the validating can be expounded as the following figure 5.2. The detail of the implementation will be explained in the section 5.2.1 and section 5.2.2.

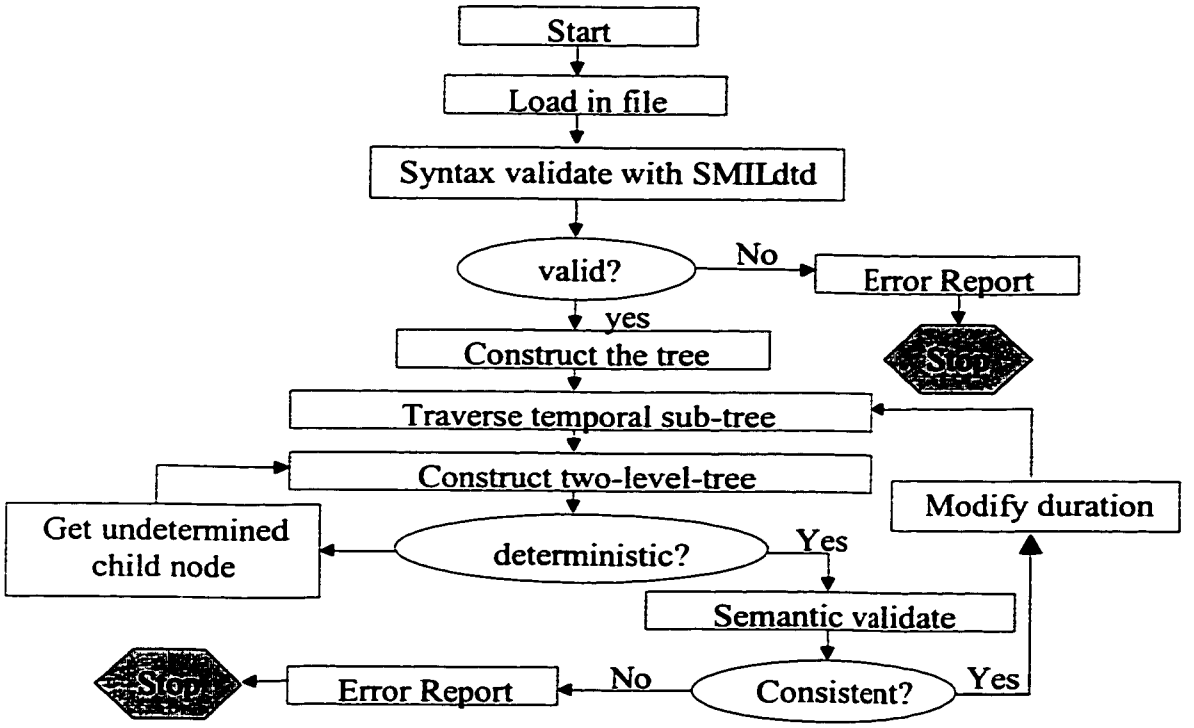


Figure5.2 Flow chart of SMIL document validation

5.2.1 Syntax validation

The XML 1.0 specification provides a way to extend the DTD using the <!DOCTYPE> element, for instance, to add a new set of entity definitions. However, our aim is to validate the documents that conform to W3C-defined DTD only.

There are two ways to check the syntactical validity of the document. One is that the DTD is defined within the document; the other, outside the document. In our validation tool, we validate the SMIL documents on the basis of the SMIL 1.0 DTD. Since it's a recommended DTD defined by W3C, we store the DTD [STD98] which is the exact copy of the one published by W3C as an individual file, and validate the SMIL document using external DTD strategy. When external DTD strategy is used, any SMIL

document to be validated has to declare the document type declaration at the beginning of the document as follows (the double quotes can be replaced by single quotes):

<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 1.0//EN" "http://www.w3.org/TR/PR-smil/SMIL10.dtd">

To ease the work of the user, our validation tool doesn't require the user to add this specific line of declaration in the SMIL document. Our validation tool first checks whether the document is a file with suffix ".smil". If it is, the program copies the document to a new file and automatically inserts the declaration at the beginning. Both the syntax and semantics validating will then operate on the new generated file.

The syntax validation can be done with a few lines of Java codes just as simple as the following by importing the XML parser packages:

```
Document d = new Document();
try{
        URL fileURL = new URL("file URL representation string"); //In our
                //program, the string is input through GUI by the user;
}catch(MalformURLException e1){}
try{
        d.load(fileURL);
}catch(ParserException e2){
        reportError(e2, System.out);
}
```

The reportError() function is provided by Document class, and the error is captured when parsing the document. In the code example above, the error is sent to the

printStream (System.out) which prints out the error messages when the syntax error is detected. We adapt it to cache the error information in an outputStream and then write it to the display area of our graphical interface. The following snapshot (Figure 5.3) shows one example output when a syntax error is found in the SMIL document.

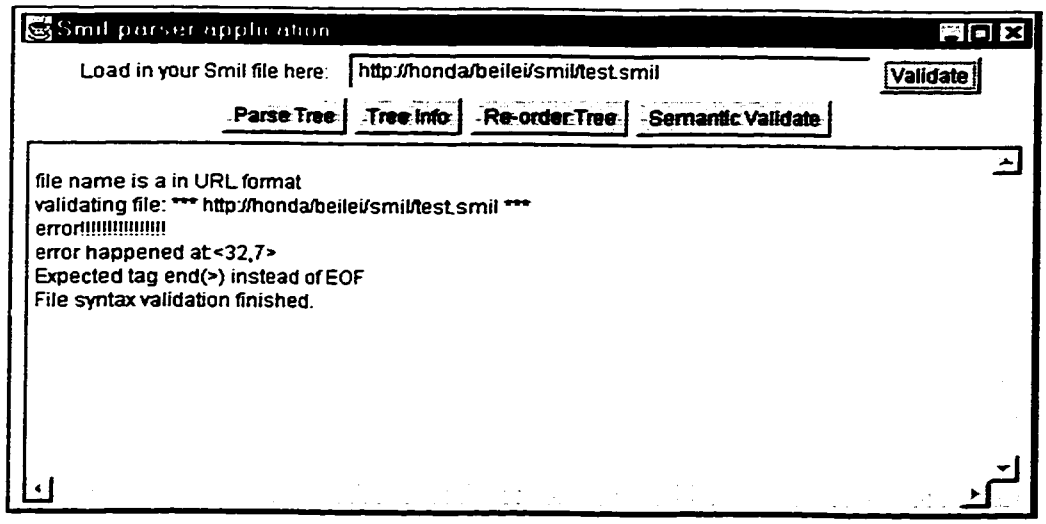


Figure5.3 A snapshot of error reporting when syntax validating

In this example, an error is found at line 32, column 7, where and end tag ">" is expected.

5.2.2 Semantics validation

Semantics validation goes on only after the document syntax is valid. Three notable classes (NodePosition, TwoLevelTree, and TimeModel) are designed and implemented to facilitate the realization of the algorithm. The responsibilities of them will be addressed in the section 5.2.2.1. Section 5.2.2.2 discusses the interactions among objects of these classes in the lifecycle of the validation. A crucial technique, *recursion,*

is highly practiced in our program. This technique optimizes the codes as well as helps solving self-contained problems. The applications of recursion used in our program will be discussed in section 5.2.2.3.

5.2.2.1 Responsibility of main classes

In our validating tool, classes can be divided into two groups based on the essential functions they run. GUI group (module) is responsible for *graphical user interface*, as well as handling the events from the user and then informing the core module, which processes the requests and sends back the results to the GUI module. User events include putting in the name of the file to be validated, asking for syntax validation, parsing tree (or, displaying the logical structure of the document), and semantics validation, etc. The processing procedure is shown as Figure 5.4.

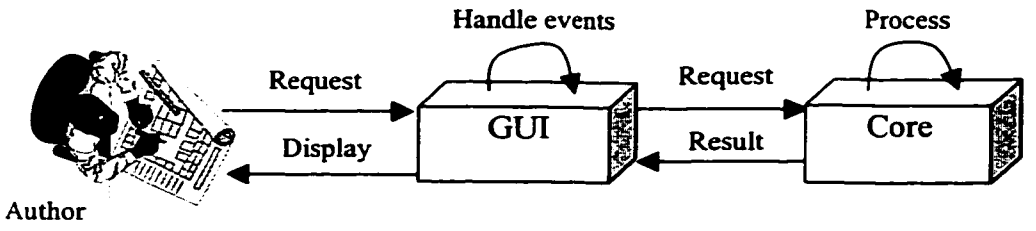


Figure5.4 Relation of the modules in validating tool

In the GUI module, class types such as Frame, TextArea, Label, Textfield, Button and their corresponding action listeners are implemented either by inheriting from the classes in the Java AWT package or by implementing the interfaces that the Java AWT package provides. In the core module, four classes are designed and implemented to

fulfill the validation task along with some functions of the classes provided by the Microsoft XML parser package.

Class SmilParser:

This class is the backbone of the program which completes methods from initiating, executing, to finally terminating the process. It is responsible for the communication among all the objects and also acts as a commander among all. The main validating functions are implemented in this class, such as validate(parameter*), doTree(parameter*), and semanticsCheck(parameter*), etc.

Class NodePosition:

This class is responsible for collecting the informations of the document elements. It provides methods to set and get the document element, and most importantly, marks the position information of the element in the tree. The position parameters include level of the tree, index among the children with the same parent, and the index of the NodePosition array. The NodePosition array stores all of the NodePosition objects that represent the information of the document elements. It is the only copy of the document elements on which the algorithm is executed. The NodePosition objects are generated at the time of parsing tree. The position information is essential prior to the validating procedure. The main member variables are listed in figure 5.5.

```
Class NodePosition
{
    Element self;        // the information of the node;
    int     level;       // level in the document tree;
    int       index;     // index of position among the children with the same
parent
    int     docIndex; // index in the one dimension NodePosition array of the
                          document tree node
    boolean leafFlag;    // indicate the element is the leaf or not.
    boolean validateFlag; // indicate whether the two-level-tree with the
                    NodePosition object as the parent has been validated or not
    .
    .
    .
}
```

Figure5.5 Main member variables of Class NodePosition

Class TwoLevelTree:

This class constructs the object with the information of the parent node of the two-level tree. The main task for this class is to provide the functionality of determining whether the two-level-tree is deterministic or not.

Class TimeModel:

Class TimeModel gathers a set of member functions to manipulate the time values of the synchronization element of the SMIL document. Given a synchronization element, the class provides the function interfaces to extract the temporal values of the three basic

temporal attributes, including *begin, dur* and *end.* The three values are returned as a 1*3 dimension string array. The class also provides the function to convert the time value represented in the *String* type, such as "5 min" and "50s", into the absolute value in milliseconds in *long* type. Another essential function is the syncValue(parameter*) function which interprets the temporal value represented in element event type into the relative value (in milliseconds) with regard to the parent's begin.

5.2.2.2 Interaction diagram

The SmilParser object is the central controller and commander among all the objects. The GUI objects contain input objects (textField and button) and output objects (label and textArea). The user posts his request through input objects, such as typing the file name in the textField and clicking the button. The corresponding listener listens to the user event and transmits the information or request to the smilParser. The smilParser then processes the user request, and posts the result in the textArea. The following interaction diagram presents the main function interfaces between the object.

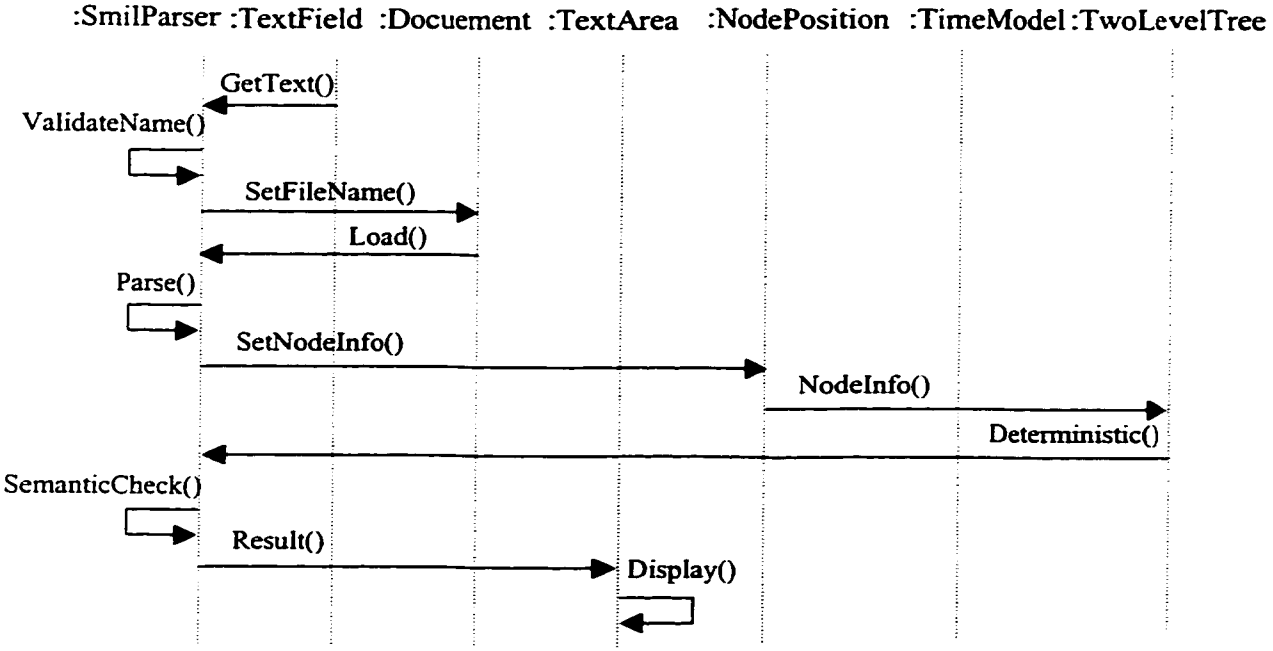


Figure 5.6 Interaction diagram for SMIL document validation

5.2.2.3 Mode of operation

As stated previously, *recursion* technique is employed in several places when validating the SMIL document. All the problems we use recursion to solve have the following characteristics: Firstly, the problem can be broken into smaller problems of the same kind that can be solved by the same method. Secondly, the sub-problem can be solved either directly (as a base case) or recursively by making a recursive call. Thirdly, the sub-problem's solution can be combined with solutions to other sub-problems to obtain the solution to the overall problem. In our program, the technique is used in the following three operations:

- doTree(int level, Element topNode)

This function implements the first stage of the semantic validation algorithm. It is called before the semantics validating starts. It's responsible to parse the document elements on the basis of its tree structure. Microsoft XML Parser provides the APIs to get the root element of the document object and get the direct children of the element object. This makes us possible to manipulate all of the tree nodes (or, elements). In our program, the tree nodes are level-ordered and stored in a NodePosition type array.

- semanticCheck()

  This function implements the ordered-node traversing at the second stage of the semantic validation algorithm. In this stage, the document tree is traversed on the level-ordered NodePosition array which was generated at the first stage. However, if the two-level tree is found to be none-determined, we will explore the branch stemming from its indeterministic child element(s) and then come back to its parent node. This recursive function contains five sub-problems in which recursive call is made. A global counter is used to count the number of validated two-level trees. It is initially assigned as the length of the temporal sub-tree node length (variable nodeLength). Every time a two-level tree is validated, the counter is decreased by one, and the nodeIndex (initially 0) is increased by one. If the counter is less than 0 or the nodeIndex exceeds nodeLength, the function returns true which indicates the end of the traversal.

- syncValue(long parentBegin, Element self)

  This is a member function of TimeModel class. It converts the temporal values of the child elements from the attribute *String* type values (either clock value or

element type value) to comparable *long* type values with regard to the parent's begin time. If the Element "self" specifies the temporal attribute value in *element event* type, the final *long* type value will depend on the synchronized value of the referee element. Therefore, it is possible and a good idea to recursively call the function itself when the element event type is met. Otherwise, if the *clock value* type is encountered, the synchronized value is obtained on the basis of the parent's begin value. Figure 5.7 shows an example of the scenario.

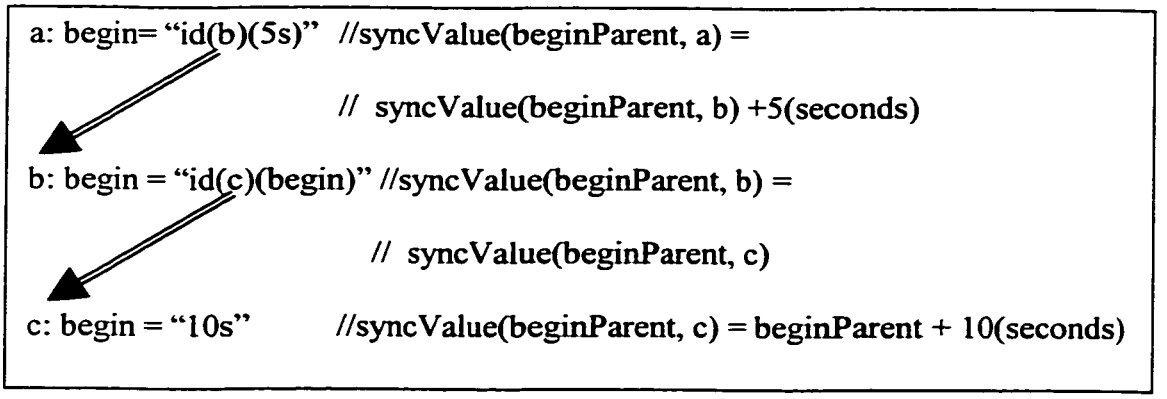


Figure 5.7 An example of recursion scenario in function syncValue()

5.2.2.4 Results

We downloaded a number of SMIL files from the web and validated them. The temporal structures of most of the SMIL files we could get are simple, and usually didn't produce temporal inconsistency within the document. But we got some warning messages. Figure 5.8 shows one scenario, in which the media object is not assigned the duration and it does no harm on the temporal consistency among others. We then validated some more complicated SMIL files written by ourselves. The errors we listed in the previous chapter were detected. Figure 5.9 shows one of the cases.

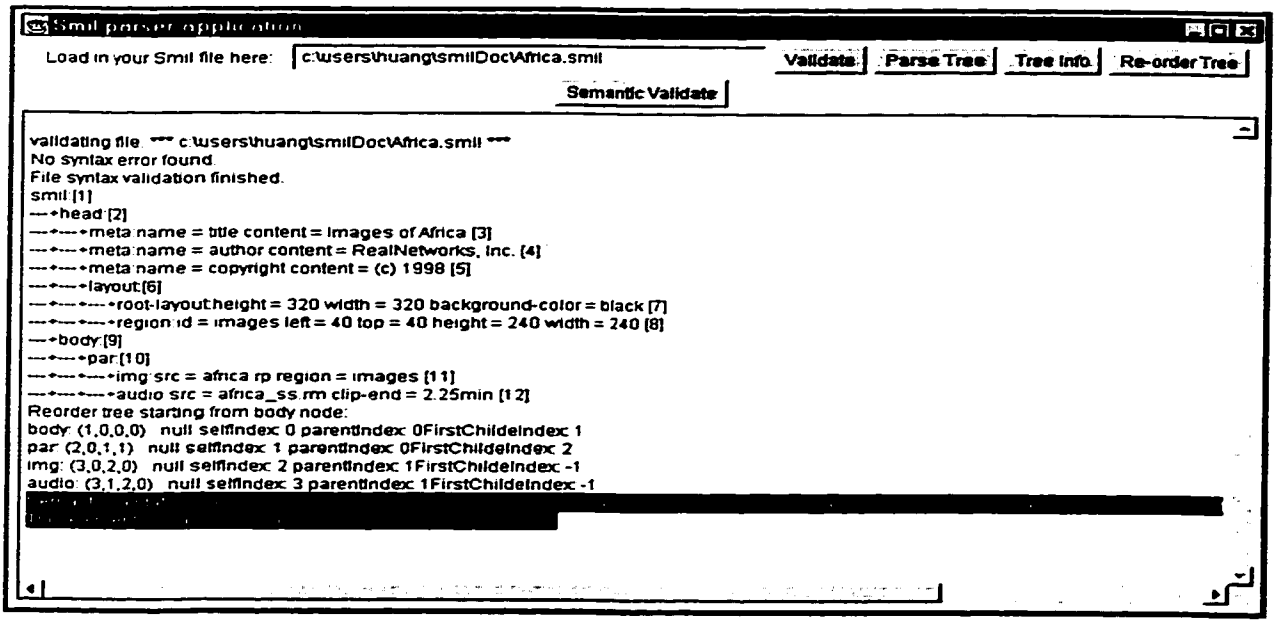


Figure 5.8 A snapshot of warning message when semantic validating

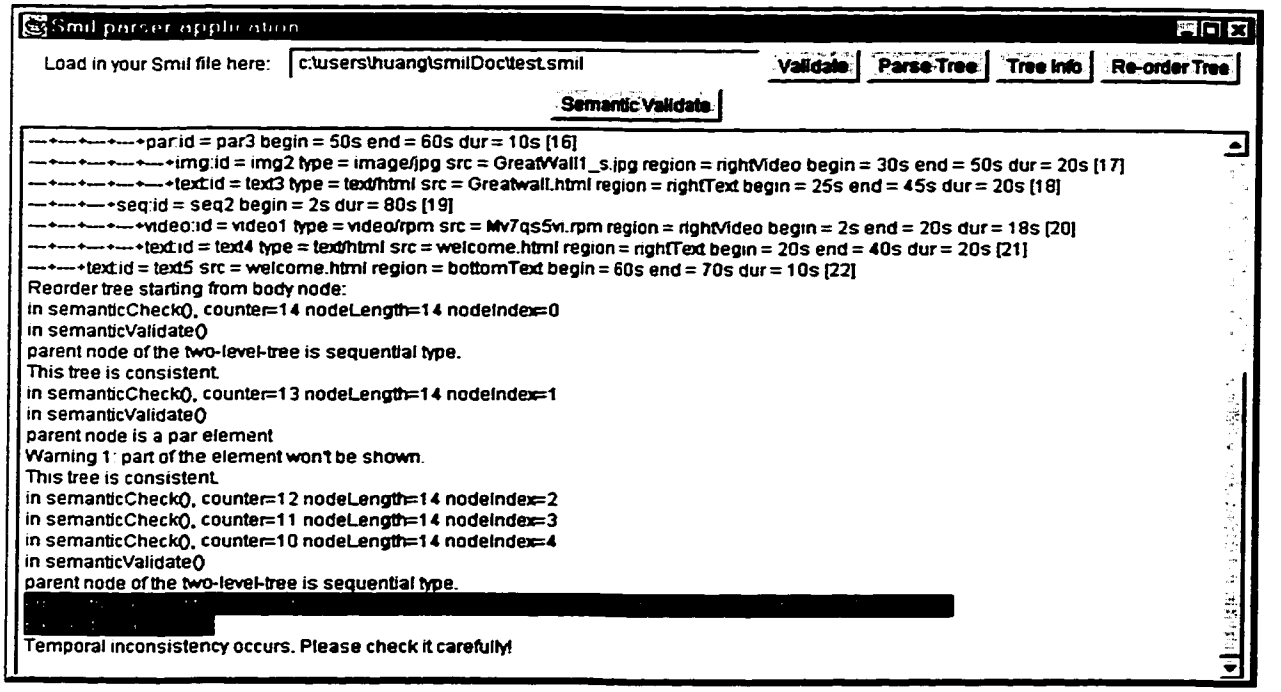


Figure 5.9 A snapshot of error detected when semantic validating

5.3 Implementation of SMIL player

To play back a SMIL presentation specified by the SMIL document, the player first needs to parse the document, extract the temporal information of the synchronization elements, schedule the synchronized scenario and finally play out through multimedia output devices. The architecture of the system is shown in figure 5.10.

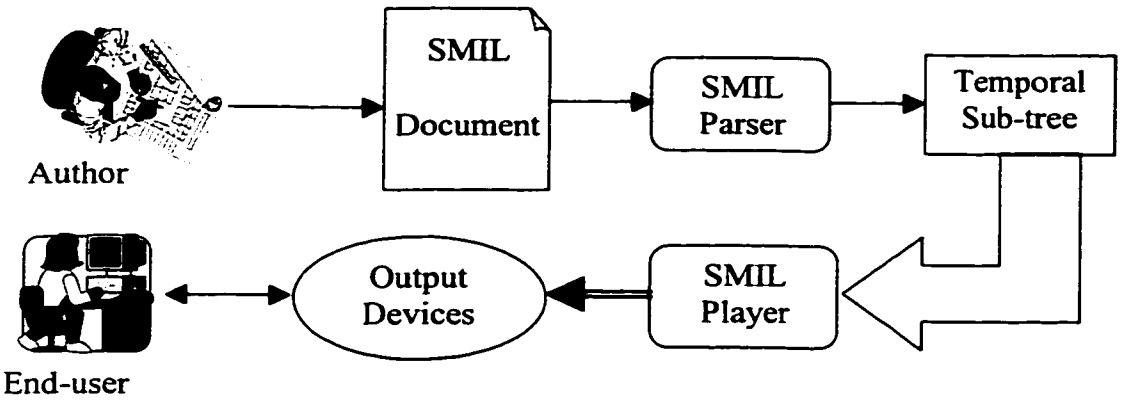


Figure 5.10 Architecture of the SMIL player system

5.3.1 SMIL parser

To play out the SMIL presentation specified in the SMIL document, we have to parse the valid document first, and extract the temporal values and relations among the synchronization elements. The playback of the multimedia sources is scheduled according to their temporal behavior specified in the document. Each synchronized presentation in parallel are viewed and marked as one task. Since the first level of the SMIL presentation constrained by "body" tag defines a sequential relation among its

direct children, the sequential tasks can be laid out in the order of task[0], task[1], ... and so on.

5.3.2 SMIL scenario presentation

We build our SMIL player demo on a runnable Java applet, named SmilPlayer applet. The playback of the synchronized multimedia elements with parallel relation is considered as an indivisible task. In the init() method, the begin and duration values of the synchronization elements are extracted from the SMIL document and passed to the member variables of the SmilPlayer object. Meanwhile, the task number is counted and the information of the different media elements within each task is stacked on the corresponding array, such as ImagePanel array and VideoPlayer array. After the player receives the start command, the start() method of the SmilPlayer is called, the "running" variable is set true, and the applet starts to orderly execute the tasks from task[0] to task[taskNo] if there is no interruption during presentation. A timer object starts counting at the time when the player starts. When the player receives the request from the user to pause, continue, fast-forward, or rewind the presentation, the timer takes a corresponding action to correct the undergoing time. The scheduler, on the other hand, monitors the timer every 50 milliseconds, compares the undergoing time with the begin and end instants of each task, and switches *off* and *on* the current and the next task. Section 5.3.2.3 will discuss about the strategy of the timer setting. The interaction ability on the special media object is completed within the corresponding media class, such as ImagePanel, VideoPlayer, AudioPlayer, and TextArea. The hyperlinking behavior is implemented by the mouseClicked() function in the MouseListener class.

5.3.2.1 Class diagram

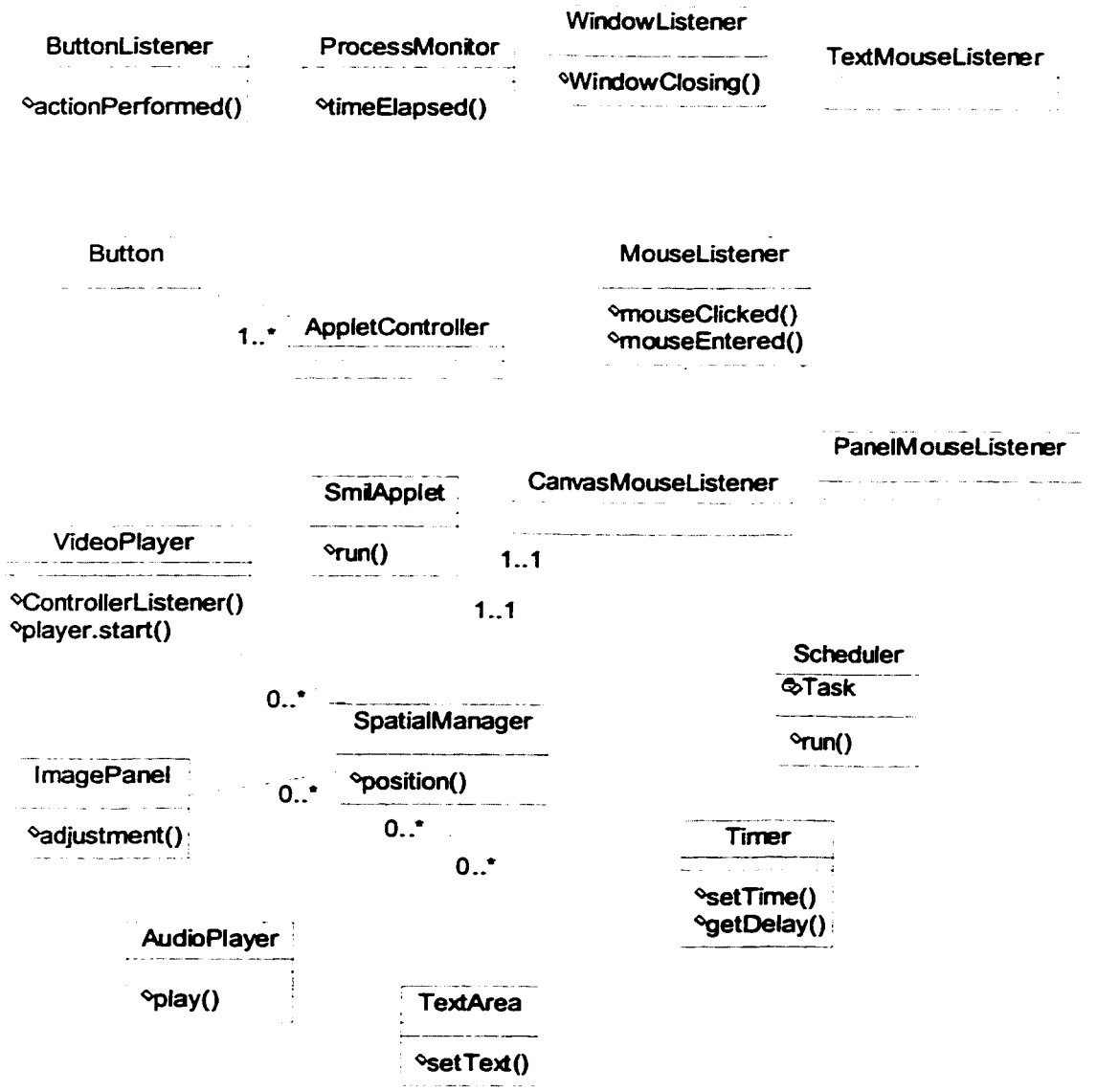


Figure5.11 Class diagram for SMIL player

The classes designed for the player can be grouped into three: GUI classes (Button, Frame, Panel, and TextField, etc.), core control class (SmilPlayer), and media layout classes (ImagePanel, VideoPlayer, AudioPlayer, and TextArea, etc). The relation among them is shown as Figure 5.11.

5.3.2.2 Interaction diagram

The user's interaction with the SMIL presentation is through clicking on the button of *start, pause/continue, forward, rewind, and stop.* The user's different action causes the corresponding operation on all the relative objects. Figure 5.12 shows a scenario for the "*continue*" case, which happens after "pause" action. Here we assume the SmilPlayer is performing task[i] when *pause* button is clicked. The tip here is to *correct* the start time of the timer when a user event happens. Different action invokes different start time correction scheme. For example, when the continue is requested after pausing for pauseDelay seconds:

the newStartTime = the oldStartTime + pauseDelay.

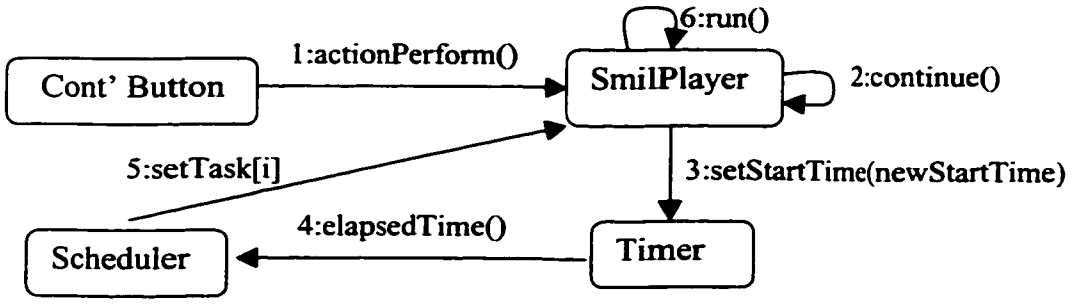


Figure5.12 Interaction diagram for SMIL player when receiving "continue" event

5.3.2.3 Time line scheduler

In our implementation, the begin and end of any multimedia component is controlled by a scheduler object. Before the player applet starts, the SmilPlayer applet acquires the begin and duration of each synchronized element from the document and stores the values into the corresponding *long* type array, such as begin[] and dur[], and passes these static values to the scheduler. As soon as the applet starts to run, the scheduler periodically gets the elapsed time from a timer object which starts to tick the time simultaneously with the applet's running. The scheduler compares the elapsed time with the begin and end value of the task in the order. It turns the corresponding task on if the elapsed time is within the begin and end value of the task and turns all the other tasks off. The task variable is the bridge between the scheduler and the SmilPlayer applet. When the applet detects that a task is on, it renders the corresponding multimedia components. Figure 5.13 shows the timeline of the SMIL presentation example.

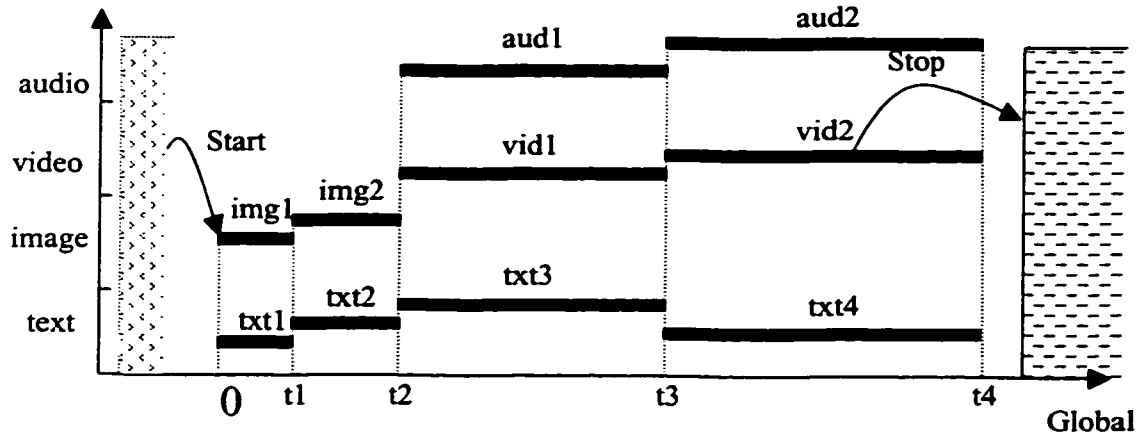


Figure 5.13 Timeline of the multimedia presentation of the example

Questions may rise such as how to handle the presentation when the user interactions such as request for "pause" or "forward" is prompted by the user? When the corresponding button is clicked, the timer changes its start time. Consequently, the elapsed time is changed and the scheduler can then decide which task should be on. In the case the "pause" button is clicked, all the tasks are off until the "resume" button is pressed. If "forward" or "rewind" button is pressed, the start time of the timer is modified by decreasing or increasing the time difference between the begin of the next task and the current time.

5.3.2.4 Characteristics on the SMIL player

Our SMIL player is an incomplete player that emphasizes on the time scheduler of the layout of the multimedia source. Based on the two distinctive characteristics of the SMIL presentation: *multimedia,* and *synchronized,* we further developed two-level user interactions to extend the W3C recommendation-based SMIL presentation. One is that each type of media is controllable by the user within the scenario. For example, the user can pause, forward, or rewind the video, increase the volume of the audio, or scroll the image if the image is bigger than its display area, etc. The other level is that the player has a VCR-like control interface beyond the SMIL scenario level. It should be pointed out that all the VCR-like controls in these two levels function differently. The within-scenario controls have the power on the corresponding media. The user interaction on this level does not influent the global time scheduler. However, the above-scenario controls have the power on the entire synchronized multimedia scenario. The global timer has to

reset or change its member variables such as startTime so that the scheduler can assign the task properly. The interaction interface is shown as figure 5.14.

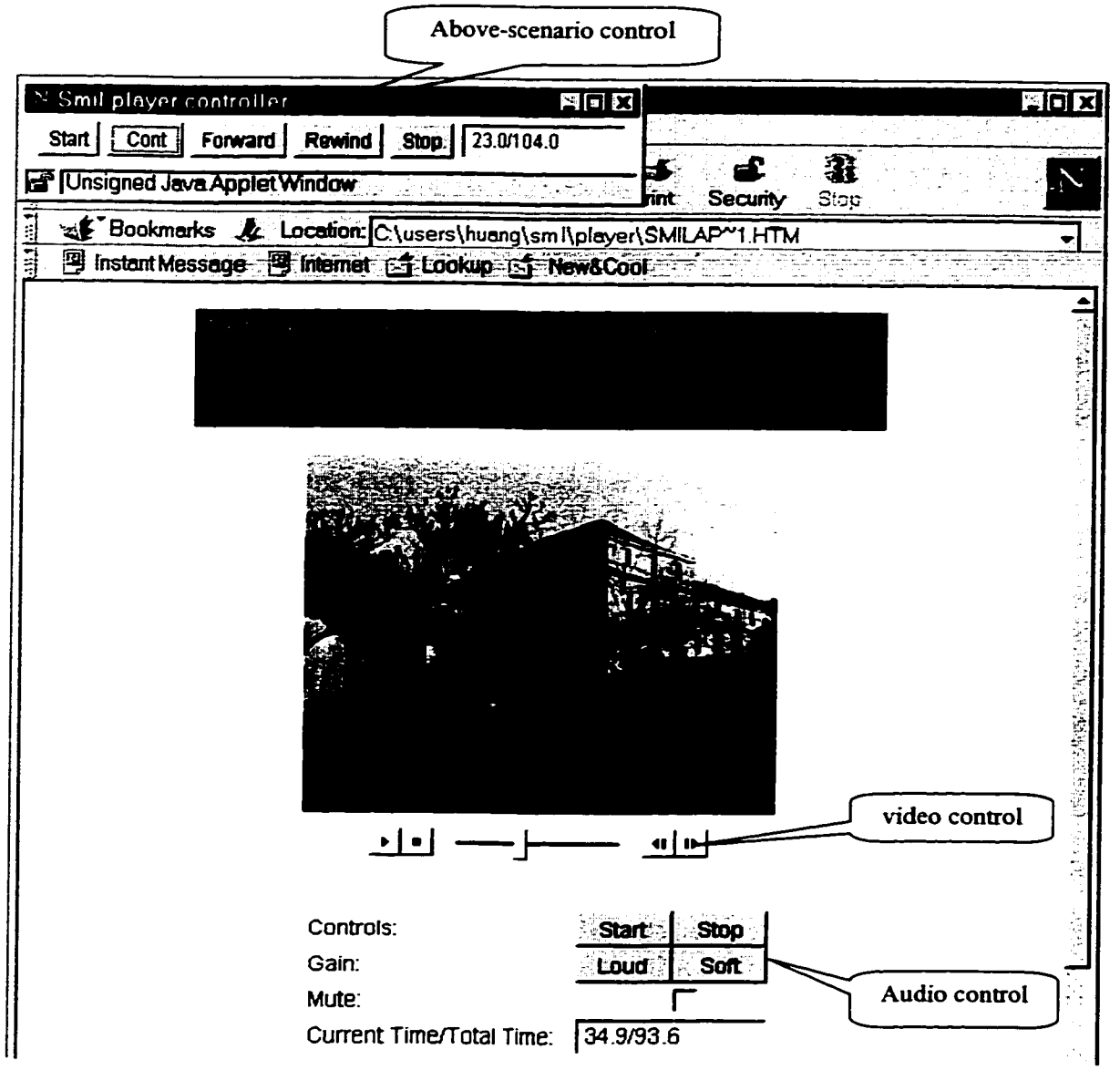


Figure 5.14 A snapshot of SMIL presentation

Chapter 6

Conclusions

6.1 Summary

The rapid evolution of communication technologies, contributed by the data networks, which enable the integration of multimedia information processing have greatly impacted our approach to TeleLearning project. The main goal of this TeleLearning project is to design and implement a practical TeleLearning system which allows the instructor to organize and store his/her courses without sophisticated computer skills, and allows the learner at the client side to retrieve and learn the course with minimum software requirement. With these goals in mind, we don't develop specialized client-side software beyond some simple JavaScript programs. Instead, we leveraged

WWW browser and plug-ins (such as RealAudio client, which is available from RealAudio as a free download) as much as possible. We also take advantage of the new powerful language (such as Java), and free software packages (such as Intel Java Media Framework and Microsoft XML parser) to implement our system. This not only minimized our own software development effort, but aided in portability as well. Companies that make browsers and plug-ins naturally want to make their products available on as many platforms as possible; and companies that develop those software packages will take responsibilities to add more convenient APIs and make the software more stable.

Web is chosen as the media to distribute and present the course because it facilitates updating contents and it is cheaper for distribution. Java is chosen as the language for implementation not just because it is "understandable" by the web, but also it is platform-independent. Java platform is incorporated into major web browsers and later on it will be built into next-generation telephone, TV set and many other consumer and business devices [JAV98]. SMIL, as a new international standard for Web, provides the Web author a simple way to express a synchronized multimedia scenario. It's expected to follow the successful steps that HTML has achieved for Web authors.

In this thesis we presented a TeleLearning system based on the most advanced on-developing standard, SMIL. We proposed an algorithm to reveal the temporal inconsistency in a SMIL document and implemented a validation tool to detect the temporal errors as well as syntax errors. Our validation tool helps the author to check the validity of the SMIL document so that any SMIL player can correctly play it back

accordingly with the same scenario as the course author expects. Our SMIL player, on the other hand, plays back the SMIL document and gives the user the sense of the SMIL presentation, although it's not a player with full functionality.

6.2 Suggestions for future work

TeleLearning is a new education model which is built on the top of the traditional education model using the modern telecommunication technologies as well as information processing technologies. Conceptually, a TeleLearning system needs to provide solutions for the three major difficulties: an easy-to-use authoring tool, the solution for storing the course material, and a powerful rendering system. This thesis proposed to apply a new web standard, SMIL, to the TeleLearning system, and thus ease the authoring difficulty. The instructor can use any familiar text editor to organize the course with SMIL. For the rendering part, companies who are members of W3C and are in favor of SMIL will work very closely to W3C and develop the player with efforts.

Currently, our SMIL player was implemented as a Java applet and the multimedia data were transferred using TCP/IP by taking the advantage of JAVA Framework API. Before a SMIL scenario is presented at the user side, the multimedia data are downloaded to the client's local machine to insure the quality of the synchronization. If there is a large size of video, the user has to wait for a long time for downloading. An improvement for this problem is to use RTP (Real-time Transfer Protocol) to transfer multimedia data. This requires the implementation of the protocol since it is not implemented in Java

Media Framework at the time that we deployed in our implementation of the SMIL player.

SMIL1.0 specification is currently a W3C recommendation. The validation tool and the player developed in this thesis were based on the recommendation version. Its further development may require modification of our tool. As of writing, three members of W3C, Microsoft, Compaq/DEC and Macromedia submitted another web-based multimedia document, called HTML + TIME (Timed Interactive Multimedia Extension for HTML), as a proposal for W3C to review [TIM98]. It builds upon SMIL recommendation to extend SMIL concepts into HTML and web browsers. The proposal emphasizes the use of *timing attributes* in contrast to *timing structure* in SMIL. It is based on HTML and hence it could be easier for the current HTML-based application to adapt to. It can also possibly become a good candidate as a document type for the Internet-based TeleLearning system.

Bibliography

[ALL83] James F. Allen, *Maintaining Knowledge about Temporal Intervals,* Communications of the ACM, Vol 26, No. 11, pp.832-843, Nov. 1983.

[ALS98] A. Al-Shammari, A.Karmouch, *Designing and Modeling a Multimedia TeleLearning Database,* CCECE 1998, pp. 605-608.

[AUG95] Grace Au, *Can Multimedia Help People Learn Fast?,* Proceedings of the IEEE International Conference on Multimedia Computing and Systems, May 1995, pp.115-122.

[COM98] CommerceNet/Nielsen Internet Demographics Survey http://www.commerce.net/research/stats/

[COU96] Courtiat, J.P. & Oliveria, R.C. *Proving Temporal Consistency in a New Multimedia Synchronization Model,* ACM Multimedia 96 Proceedings, Boston, USA, Nov. 1996, pp141-152.

[CRA98] Patricia Cravener, *Education on the Web: A Rejoinder,* Computer Network Security, IEEE computer society, September 1998, pp.107-111.

[CWI98] SMIL Syntax Validator http://dejavu.cs.vu.nl/~symm/validator/

[DAT98] The Leader in XML Presents DataChannel RIO, http://www.datachannel.com/

[DOM98] W3C working draft, *Document Object Model,* http://www.w3.org/DOM/

[GIN98] Allen Ginsberg, Peter Hodge, Teri Lindstrom, et al, *"The Little Web Schoolhouse" Using Virtual Rooms to Create a Multimedia Distance Learning Environment,* ACM Multimedia 98 – Electronic Proceedings,

http://www.acm.org/sigmm/MM98/electronic_proceedings/ginsberg/index.html

[GRI98] The GRaphical iNterface (GRiNS) to SMIL, http://www.cwi.nl/GRiNS/

[HIR95] Nael Hirzalla, Ben Falchuk, and Ahmed Karmouch, *A Temporal Model for Interactive Multimedia Scenarios*, IEEE MultiMedia, 2(3):24-31, Fall 1995.

[HOL98] Steven Holzner, *XML complete,* McGraw-Hill publisher, ISBN 0-07-913702-4.

[HPA98] Hypermedia Presentation and Authoring System, http://www.research.digital.com/SRC/HPAS/

[HYT92] HyTime. Information technology – Hypermedia/Time-based Structuring Language (HyTime). ISO/IEC DIS 10744, Aug 1992.

[JAV98] The Source for JAVA Technology, http://java.sun.com/nav/developer/index.html

[LAY95] Nabil Layaida, Cherif Keramane, *Maintaining Temporal Consistency of Multimedia Documents,* Electronic Proceedings of the ACM Workshop on Effective Abstractions in Multimedia, November 4, 1995, San Francisco, California

[LAY96] Nabil Layaida, Loay Sabry-Ismail, *Maintaining Temporal Consistency of Multimedia Documents Using Constraint Networks,* Proceedings of SPIE 2667, pp.124-135, January 1996

[LIT93] Little T.D.C., Ghafoor A, *Interval-Based Conceptual Models for Time-Dependent Multimedia Data,* IEEE Transactions on Knowledge and Data Engineering (Special Issue: Multimedia Information Systems), Vol. 5, num. 4, pp.551-563, August 1993.

[LLO98] Lloyd Rutledge, Lynda Hardman, Jacco van Ossenbruggen* and Dick C.A. Bulterman, *Structural Distinctions Between Hypermedia Storage and*

Presentation, ACM multimedia 98, http://www.acm.org/sigmm/MM98/electronic_proceedings/rutledge/

[OAK96] Burks Oakley II, *A Virtual Classroom Approach to Learning Circuit Analysis,* IEEE Transactions on Education, August 1996, Vol. 39, Issue 3. http://aln.coe.ttu.edu/ieeezips/homepage/table.htm

[RLN98] RealSystem G2, http://www.real.com/g2/index.html

[RUM99] James Rumbaugh, Ivar Jacobson, Grady Booch. The Unified ModelingLanguage Reference Manual. Addison-Wesley, Reading, Mass., 1999

[SCH90] Roger C. Schank, *Teaching Architectures,* Technical Report #3, August 1990, Institute for the Learning Sciences, Northwestern University.

[SHE96] Sherry, L. *Issues in Distance Learning,* International Journal of Distance Education, 1(4), pp.337-365.

[SMI98] W3C recommendation, *SMIL1.0 specification,*

http://www.w3.org/TR/REC-smil/.

[STD98] W3C recommendation, *SMIL1.0 DTD,* http://www.w3.org/TR/PR-smil/SMIL10.dtd

[STE97] Mia Stern, Jesse Steinberg, Hu Imm Lee, Jitendra Paddhye, James F. Kurose, *MANIC: Multimedia Asynchronous Networked Individualized Courseware,* Educational Multimedia and Hypermedia, 1997, Department of Computer Science, University of Massachusetts, http://www.aml.cs.umass.edu/publications.html

[TIM98] Timed Interactive Multimedia Extensions for HTML (HTML + TIME), http://www.w3.org/TR/1998/NOTE-HTMLplusTIME-19980918

[W3C97] W3C architecture domain, *Audio, Video and synchronized multimedia,* http://www.w3.org/AudioVideo/.

[WAH96] H. Abdel-Wahab, K. Maly, A. Youssef, et al., *The Software Architecture and Interprocess Communications of IRI: An Internet-based Interactive Distance Learning System,* Proceedings of the 5th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'96) IEEE, pp.4-9.

[WAH94] Thomas Wahl & Kurt Rothermel, *Representing Time in Multimedia systems,* Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'94),pp.538-543, Boston, MA, May 14-19, 1994.

[WAN96] R.Wang, A. Karmouch, *A Broadband Multimedia Telelearning System,* Proceedings of 5th Int. IEEE Symposium on High-Performance Distributed Computing-Multimedia and Collaborative Environments, Syracuse, August 1996.

[WEB98] Larry Bouthillier, *Synchronized Multimedia on the Web,* WEB Techniques magazine, Sept.1998, vol. 3, Issue 9. http://www.webtechiniques.com/

[XML98] W3C recommendation, *Extensible Markup Language (XML) 1.0,* http://www.w3.org/TR/1998/REC-xml-19980210.

[ZHA98] Zhongyao Zhang, Ahmed Karmouch, *Multimedia Courseware Delivery over the Internet,* CCECE June 1998, Waterloo, pp.609-612.

Publication

Beilei Huang, Ahmed Karmouch, "Enhanced Interactivity in A Multimedia System over Internet", CCECE June 1998, Waterloo, pp533-536