HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Engineering Physics and Mathematics

Sebastian Nykopp

A Java-based Presentation System for Synchronized Multimedia

This Master's Thesis has been submitted on August 31, 1999 for official examination for the degree of Master of Science.

Supervisor:	Professor Matti Karjalainen
Instructor:	M.Sc. Martti Rahkila

HELSINKI UNIVERSITY OF TECHNOLOGY	ABSTRACT OF MASTER'S THESIS

Author:	Sebastian Nykopp		
Title of Thesis:	A Java-based Presentation System for Synchronized Multimedia		
Finnish Title:	Java-pohjainen esitysjärjestelmä synkronoidulle multimedialle		
Date:	August 31, 1999	Number of Pages: 71	
Department:	Department of Engineering Physics and Mathematics	Chair:	S-89 Acoustics and Audio Signal Processing
Supervisor:	Professor Matti Karjalainen		
Instructor:	M.Sc. Martti Rahkila		

This work addresses the problem of integrating multimedia content into Internet-based applications. An experimental system was developed for the presentation of synchronized multimedia in the World Wide Web (web) environment. The system, named SynCope, was implemented using open, web-oriented technologies such as Synchronized Multimedia Integration Language (SMIL) and the Java programming language. A key objective was evaluating the feasibility of using Java and the Java Media Framework (JMF) in a system of this nature.

In the thesis an overview of multimedia systems technology is presented, with emphasis on aspects of multimedia synchronization. These issues are explored to provide a context for the models used in the SynCope system.

SynCope provides functionality for parsing, scheduling and presenting multimedia documents specified using SMIL. It serves as a framework which can be refined and extended to suit the multimedia requirements of specialized applications. The system architecture of SynCope is presented and major design issues are described. The system is analyzed in view of the introduced general concepts. Implementation issues are documented in conjunction with developed test applications.

In conclusion the relevance and applicability of the system is assessed and possible future developments are suggested. Findings indicate that while the implemented system may be used to advantage in experimental and prototyping settings, there are still sufficiently serious issues in the underlying technology to make production use problematic.

Keywords: multimedia, synchronization, Java, hypermedia

Not borrowable until:	Library code:

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Sebastian Nykopp		
Työn nimi:	Java-pohjainen esitysjärjestelmä synkronoidulle multimedialle		
English Title:	A Java-based Presentation System for Synchronized Multimedia		
Päivämäärä:	31.8.1999	Sivumäärä:	71
Osasto:	Teknillisen fysiikan ja matematiikan osasto	Professuuri:	S-89 Akustiikka ja äänenkäsittelytekniikka
Valvoja:	Professori Matti Karjalainen		
Ohjaaja:	DI Martti Rahkila		

Tässä työssä käsitellään multimedian integrointia Internet-pohjaisiin sovelluksiin. Kehitettiin kokeellinen järjestelmä synkronoidun multimedian esittämiseen World Wide Web (web) ympäristössä. SynCope:ksi nimetty järjestelmä toteutettiin käyttäen avointa, web-pohjaista teknologiaa kuten Synchronized Multimedia Integration Language (SMIL) sekä Java-ohjelmointikieltä. Keskeinen tavoite oli selvittää Javan sekä erityisesti multimediapalveluja tarjoavan Java Media Framework (JMF)-sovelluskehyksen soveltuvuttaa tämänlaiseen järjestelmään.

Työssä esitetään katsaus multimediajärjestelmien teknologiaan, painottaen erityisesti multimedian synkronointiin liittyviä näkökohtia. Nämä asiat muodostavat viitekehyksen järjestelmän toteutuksessa käytetyille malleille.

SynCope sisältää toiminnallisuutta jolla voidaan analysoida, aikatauluttaa sekä esittää SMIL-kielellä määriteltyjä multimediadokumentteja. Se on tarkoitettu kehykseksi jota voidaan tarkentaa ja laajentaa erikoistuneiden sovellusten multimediatarpeisiin. SynCopen järjestelmäarkkitehtuuri esitetään ja keskeiset suunnittelunäkökohdat kuvaillaan. Järjestelmä analysoidaan esitettyjen yleisten käsitteiden valossa. Toteutukseen liittyviä seikkoja ja ongelmia kuvataan kehitettyjen testisovellusten yhteydessä.

Lopuksi arvioidaan järjestelmän tarkoituksenmukaisuutta ja käyttömahdollisuuksia, sekä ehdotetaan mahdollisia jatkokehityshankkeita. Todetaan että järjestelmää todennäköisesti voi käyttää hyödyksi kokeellisissa sovelluksissa sekä prototyypeissä. Toisaalta alustana käytetyssä teknologiassa on tällä hetkellä siinä määrin vakavia ongelmia että tuotantokäyttö voi olla vaikeaa.

Avainsanat: multimedia, synkronointi, Java, hypermedia

Ei lainata ennen: Sijaintipaikka:

TEKNISKA HÖGSKOLAN SAMMANDRAG AV DIPLOMARBETE

Utfört av:	Sebastian Nykopp		
Arbetets namn:	Ett Java-baserat återgivningssystem för synkroniserade multimediapresentationer		
English title:	A Java-based Presentation System for Synchronized Multimedia		
Datum:	31.8.1999	Sidantal:	71
Avdelning:	Avdelningen för teknisk fysik och matematik	Professur:	S-89 Akustik och ljudbehandlingsteknik
Övervakare:	Professor Matti Karjalainen		
Handledare:	DI Martti Rahkila		

I detta arbete behandlas integration av multimedia i Internet-baserade applikationer. Ett experimentellt system utvecklades för återgivandet av multimediapresentationer i en World Wide Web (web)-omgivning. Systemet, kallat SynCope, konstruerades på basen av öppen, web-centrerad teknologi såsom Synchronized Multimedia Integration Language (SMIL) och programmeringsspråket Java. Ett av arbetets huvudsakliga syften var att uppskatta i vilken mån Java och speciellt Java Media Framework (JMF), som erbjuder multimediaegenskaper, lämpar sig för ett system av denna typ.

Arbetet ger en överblick av grundteknologin för multimediasystem, med betoning på frågor gällande synkronisering av multimedia. Denna behandling är avsedd att bilda ett sammanhang för de modeller och lösningar som används i SynCope-systemet.

SynCope innehåller funktionalitet för att analysera, konstruera tidsscheman för och återge multimediadokument som specificerats med hjälp av SMIL-språket. Systemet utgör en stomme som används för att utveckla applikationer. Det kan vidareutvecklas till att motsvara mycket specialiserade krav beträffande multimediaegenskaper. SynCopes systemarkitektur presenteras och centrala planeringssynpunkter beskrivs. Systemet analyseras utgående från framställda allmänna begrepp. Speciella frågor och problem i systemets utförande behandlas i samband med utvecklade testapplikationer.

Slutligen uppskattas systemets ändamålsenlighet och tillämpningsmöjligheter. Det konstateras att SynCope sannolikt kan vara till nytta speciellt i experimentella applikationer och prototyper, men att den grundläggande teknologin för tillfället innehåller allvarliga problem, som kan försvåra användning i produktionssyfte. Systemets möjliga framtidsutsikter presenteras med förslag för vidareutveckling.

Nyckelord: multimedia, synkronisering, Java, hypermedia

PREFACE

This work has been carried out during the winter, spring, and summer of 1999. The original idea for the project was formulated in late 1997, but it was left to incubate for a year due to other commitments. I would like to thank the supervisor of the thesis, Professor Matti Karjalainen, for giving me the opportunity to realize the project. I would particularly like to thank the instructor of my thesis, M.Sc. Martti Rahkila, for his interest and helpful guidance throughout the work and for keeping the ambition level high. Thanks also go to my friends and colleagues at Satama Interactive, in particular Director of Technology Risto Koivula, for allowing me the time I needed to finish this work.

I also wish to thank all my friends and family for supporting me in my studies. A special thank you goes to my father, Christer Nykopp, for fueling the curiosity that lead me into the field of science and technology, and for reminding me to see my studies through. Most of all, I wish to thank my fiancée, Soile Motsakov, for her support, love, and understanding – these are the things that really matter.

Helsinki, August 31, 1999

Sebastian Nykopp

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DOM	Document Object Model
DTD	Document Type Definition
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO	International Organisation for Standardisation
ITU	International Telecommunications Union
JMF	Java Media Framework
LDU	Logical Data Unit
MCS	Multimedia Communications System
MHEG	Multimedia and Hypermedia Expert Group
MIME	Multipurpose Internet Mail Extensions
MMIS	Multimedia Information System
MP3	MPEG-1 layer 3
MPEG	Motion Picture Expert Group
MVC	Model-View-Controller
OCPN	Object Composition Petri Net
POM	Presentation Object Model
QOS	Quality of Service
RMF	Rich Music Format
RMI	Remote Method Invocation
RSVP	Resource Reservation Protocol
RTP	Real-Time Transfer Protocol
RTSP	Real-Time Streaming Protocol
SAX	Simple Application Programming Interface for XML
SGML	Standard Generalized Markup Language
SIP	Session Initiation Protocol
SMIL	Synchronized Multimedia Integration Language

SWOT	Strengths, Weaknesses, Opportunities, and Threats
TAC	Temporal Access Control
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

LIST OF SYMBOLS

$\varnothing$	no temporal relation
$\leq$	before or simultaneous to
$<$	before
$=$	simultaneous to
$>$	after
$\geq$	after or simultaneous to
$\neq$	not simultaneous to
?	any temporal relation

1 INTRODUCTION

1.1 Background

Systems for processing and presenting interactive, time-based multimedia content are a well established technology in workstation-oriented computing environments. During the past decade, research and development of multimedia systems has increasingly focused on models for *distributed* multimedia applications [1], [2], [3], [4]. Such efforts have gained further momentum as the *Internet* and the *World Wide Web* (WWW or the *web*) rapidly have become mainstream means of communication.

Since the early 1990s a wide variety of initiatives have been pursued in order to achieve integration of multimedia content with the web, which is based on the *hypertext*[1] paradigm [5]. This task poses several technical challenges, in particular related to the delivery of time-based media over the Internet communications infrastructure. While not as immediately obvious, there are many other issues to be resolved. Key among these is the development of standard means for specifying and enforcing *temporal relations* between various media objects on the Internet. This is also known as multimedia *synchronization* [6], [7].

Initial efforts for integrating multimedia into a web setting were mainly concerned with low-level issues, such as the development of media encoding algorithms ([8], [9]) and transmission protocols ([10], [11]) suitable for use in the Internet environment. In recent years increasing attention has been paid to more abstract concepts, such as the specification and authoring of composite multimedia documents.

Until recently multimedia content on the web was almost exclusively deployed using proprietary technology. In fact, a number of such applications are currently viewed as de-facto standards (RealNetworks RealPlayer and Macromedia Shockwave are typical

[1] The term hypertext describes documents, in which related textual information is non-linearly interlinked, by way of *hyperlinks*. *Hypermedia* extends this concept to allow hyperlinks between any types of media. When several types of media are used simultaneously, the term *multimedia* is used [12].

examples). This kind of proprietary technology has an important role to play in the evolution of the web; however, there is clearly a case to be made for open industry standards. In addition to enhanced interoperability, standard solutions will allow developers to spend more of their time designing applications, and less time studying variant implementations of similar concepts.

A certain trend of convergence can be seen between current models used in multimedia systems and standard technologies emerging in the Internet community. Examples of such technologies are the World Wide Web Consortium (W3C) recommendation *Synchronized Multimedia Integration Language (SMIL)* [13] and the *Java Media Framework (JMF)* [14], [15]. The former provides a specification and interchange format for multimedia presentations, whereas the latter provides multimedia system services for the Java platform. Java has been widely adopted by the Internet community as a platform and model for application development. Open, Internet-oriented technologies, such as Java and SMIL, may provide a path towards common ground for Internet-based multimedia applications. Evaluating the state and possibilities of such a convergence was a central motivation for the project described in this thesis.

1.2 Objectives, Scope and Methods

The main objective of the project described here was to develop and critically evaluate an open and extensible presentation system for synchronized, Internet-based multimedia. The system was implemented using a specific set of emerging technologies, namely SMIL, Java and JMF. The intention was to create a standards-oriented multimedia system. More than on implementing a production-level system, the focus was placed on assessing the feasibility of the task and identifying critical design and implementation issues. In this respect gaining experience and reaching conclusions about the overall technical approach was a key objective in itself.

In the case of sound signals, inconsistencies in timing or erroneous data are easily recognized by a human listener. This makes audio a particularly challenging medium with regard to multimedia synchronization. Audio was thus chosen as the primary medium to be considered in the design and testing of the presentation system.

The project consisted of three stages: a background study providing an overview of relevant multimedia technologies, a software development phase, and analysis of the developed system. The main purpose of the background study was to establish a theoretical and technical context for the system to be developed. Concepts, models, and terminology were introduced to serve as reference points in designing, and subsequently analyzing, the system. In the development phase, the concepts were applied to develop a presentation system for synchronized multimedia. The system, named SynCope, was then analyzed within the established reference framework.

Chapter 2 of this thesis provides an overview of general technology issues related to distributed multimedia systems. In Chapter 3 a more detailed view of synchronization concepts is given. The SynCope system is specified and described in Chapter 4, with test cases and analysis documented in Chapter 5. In conclusion, Chapter 6 summarizes insights gained and suggests possible paths for further work.

2 MULTIMEDIA SYSTEMS OVERVIEW

2.1 Multimedia Systems Architectures

2.1.1 Multimedia Systems Framework

Multimedia systems provide services for the integrated capture, storage, manipulation, communication, and presentation of various independent media [6]. Such systems require various forms of technical infrastructure. The framework for multimedia systems architectures presented by Buford in [2] makes a distinction between Multimedia Information Systems (MMIS) and Multimedia Communications Systems (MCS) (Figure 2.1).

The key elements of an MMIS are an Information Model for representing and describing multimedia content, and a Distributed Processing Model which provides services and an environment for multimedia application software. An MCS is characterized by a Conferencing Model which defines protocols and services for multiparty communications, and a Network Model which provides lower level abstractions in the form of a network architecture and network protocols. These four models are closely related and together provide a comprehensive description of the technical issues related to distributed multimedia systems.

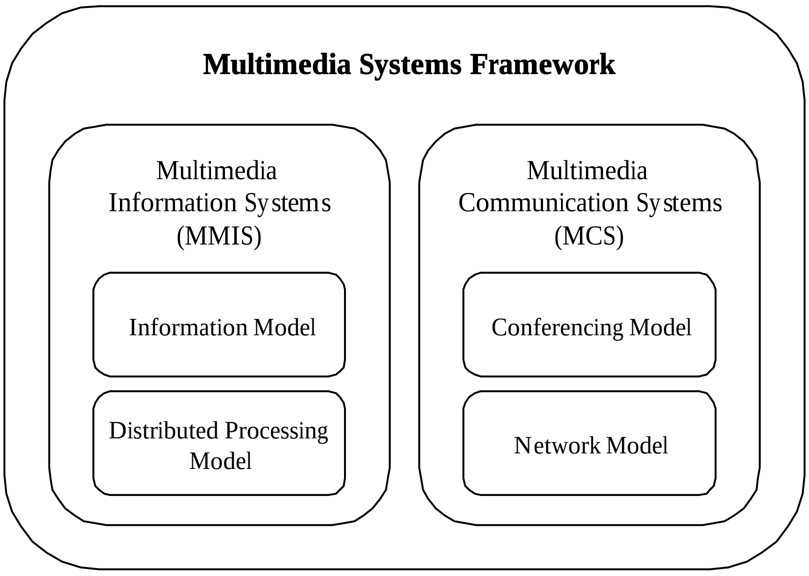

Multimedia Systems Framework
Multimedia Information Systems (MMIS)
Information Model
Distributed Processing Model
Multimedia Communication Systems (MCS)
Conferencing Model
Network Model

Figure 2.1. *The interrelated models of multimedia systems, according to [2].*

2.1.2 Quality of Service (QOS)

Any multimedia system or application is required to perform under some form of real-time constraints. The level of real-time response required varies from application to application, but the principle of operating within deadlines is shared by all. Various types of constraints and limitations in each component of a system together define what level of performance the system can provide. Multimedia applications have traditionally been based on the principle of *best-effort* service, in which the performance of an application is defined in a non-deterministic way by the resources available to it at any given time. For example, changing network conditions can severely impact the performance of a distributed multimedia application. Issues such as these are significant because they directly affect the way a user perceives the application as a whole.

To provide the user with an experience of predetermined and consistent quality, each resource used by the system must perform within specific parameters. The specification and management of these requirements is generally referred to as *Quality of Service* (QOS). QOS is a term often used to refer specifically to network communications, but here it will be used to cover any multimedia-related resource which significantly affects the experience of the end user. Because QOS is affected by every component of a system, as well as their interactions, it can be seen as one of the most important architecture-level issues to consider in multimedia systems.

The SynCope presentation system described in this thesis primarily relates to the MMIS domain, as far as implementation is concerned. However, since the system operates in the Internet environment, certain MCS issues are of considerable importance and need to be addressed. In the following sections, an overview of multimedia technology will be presented. First some general properties of multimedia systems will be examined. This will be followed by concepts more directly related to the above model, to the extent they are relevant to this thesis. Where applicable the role and impact of QOS issues will be addressed.

2.2 Media Types

Among the principal characteristics of multimedia systems is the ability to process and present various *types* of media. Each media type, such as audio, video, image or text, facilitates the communication of a specific form of information. Instances of specific media types are referred to as *media items*. This section provides an overview of common media types and their properties in the context of distributed, digital multimedia systems.

A high-level distinction is made between *time-dependent* and *time-independent* media types [16], [6]. From the perspective of a multimedia presentation system, time-dependent media are considerably more challenging technically. This is mainly due to the real-time requirements associated with delivery and playback of such media.

2.2.1 Time-Dependent Media

Time-dependent media objects are handled as a stream of logical data units (LDUs) [6] or samples, which are presented according to the implied or specified *time dependencies* of the object [17]. In the common case of *continuous* media, successive data units are presented at constant intervals. Typical examples of continuous media are audio and video streams, which are rendered by playing back audio samples or video frames at a specific rate.

Digital Audio

Audio is a crucial element in most multimedia presentations. As a digital media type, it also places high demands on the performance of multimedia systems. The term digital audio refers to the representation of audio signals in a form suitable for storage and playback in digital systems. Digital audio can be produced by *sampling* real-world audio or by performing digital *synthesis* [18], [19].

Various encoding and compression schemes are available for digital audio signals. The basic format for uncompressed digital audio, known as PCM (Pulse Code Modulation), represents an audio waveform as a series of samples characterized by a specific *sampling rate*. Each sample is numerically encoded using a specific number of bits. The sampling rate defines the capture and playback rate for the audio stream and indirectly specifies the bandwidth of the audio signal. The Nyquist theorem [18] states that a sampled signal can represent a range of frequencies equal to one half of the sampling rate. Thus, for example, a compact disc (CD) audio signal with sampling rate 44.1 kHz has a bandwidth of approximately 20 kHz[2]. CD audio is encoded using 16-bit samples, which for a stereo signal leads to a bitrate of approximately 1.4 Mbps. Telephone-quality speech has sampling rate of 8000 Hz and a bitrate of 64 kbps. [74]

Real-time transport of PCM audio requires prohibitive levels of network bandwidth, as demonstrated by the aforementioned bitrate of CD-quality audio. Various audio-specific coding schemes can be used to provide data compression or reduction and bring the network bandwidth required for audio transmission down to manageable levels. *Subband coding* [8] is based on using filter banks to split an audio signals into several frequency bands. These bands are often related to the *critical band* concept, which is used to model frequency resolution in human hearing [20]. By separately coding and assigning bits to each band and applying psychoacoustical concepts such as *masking* [20], considerable data reduction can be achieved. Audio coding which utilizes knowledge about the way sound is perceived by humans is generally referred to as *perceptual* coding.

For speech-oriented audio signals, higher levels of compression can be reached by considering the human speech production mechanism and common characteristics of speech signals. A basic technique in this field is *linear predictive coding*, in which a speech signal is separated into a source excitation and a filter chain, parallelling the vocal cords and vocal tract [21]. In [8], Noll provides a survey of audio coding methods useful for multimedia communications.

Both standardized and proprietary audio coding algorithms are widely deployed in Internet-based audio delivery systems. On the proprietary side RealNetworks' RealAudio line of compression formats currently have considerable support. For wideband audio, the main standards authority is the ISO Motion Picture Expert Group[3] (MPEG). The MPEG-1 (layer 3) audio coding standard [25], commonly referred to as MP3, is currently among the most widespread formats for economical storage of high-quality audio. (Typical MP3 bitrates are on the order of 100 kbps.) Speech-specific

[2] The audible range of human hearing is approximately 20-20,000 Hz. However, it is unusual for person to be able to hear over the entire range. [20]

[3] ISO-IEC/JTC1/SC29/WG11

coding algorithms have been developed particularly within the telecommunications industry. The International Telecommunication Union (ITU-T) recommendation G.723.1 [22] describes a speech coding scheme for multimedia communications in highly bandwidth-constrained environments. The recommendation introduces a dual coder with the alternative bitrates of 5.3 kbps and 6.3 kbps.

Music Formats

Digital audio can represent any sound signal at a chosen level of fidelity. However, the amount of bandwidth required for high-quality audio is still too high for real-time use in many scenarios. Furthermore, in musical settings it is often desirable to transfer data at a higher abstraction level than a sampled waveform representation of a musical performance.

The Musical Instrument Digital Interface (MIDI) [21], [23], [24] provides means for controlling synthesizers and other digital musical instruments using an event-based protocol. Instruments can be instructed when to start and stop playing specific notes via MIDI events. Other control information, such as pitch bending and instrument selection is also transferred using the MIDI protocol. Note and control data is transmitted on several parallel channels, allowing different instruments to play notes simultaneously. MIDI originated as a means for linking together and creating control chains for digital instruments, but today it is in widespread use in workstation-based multimedia scenarios. The MIDI file format provides a straightforward way of storing the events associated with a musical performance; it is widely used as an interchange format by sequencing, composition and notation software. General MIDI [21] specifies a standard instrument set, which makes it possible for MIDI files to sound similar across different synthesizers and systems. Today practically any multimedia-enabled workstation provides support for MIDI in the form of a multi-channel synthesizer implemented either in hardware or software. Current MIDI synthesizers on general-purpose multimedia workstation are typically based on *wavetable synthesis* [21]. On the whole, such synthesizers produce sound of sufficiently high quality that MIDI files can be used as a music format for multimedia presentations.

MIDI files and PCM-based audio files can be seen as two extremes; the former contains only control information and the latter only waveform data. A variety of file formats are available which combine these two approaches. The main concept of music file types such as MOD [24] and RMF[26] is combining note information with audio samples used to play the notes. These formats are not as widely known or supported as MIDI and PCM-oriented file types.

Digital Video

Digital video is characterized by successive images or video *frames*, which are presented at a specified rate (in analogue television the frame rate is 25 fps or 30 fps, depending on the country). Due to the high information content in each frame, real-time transmission of digital video typically implies considerably higher bandwidth demands than audio transmission. On the other hand, video signals contain redundancies which provide ample opportunities for employing compression algorithms.

Spatial redundancies occur within a given video frame or image, typically at the pixel or line level. For example, a single-colored surface contains spatial redundancy within connected areas of constant pixel values. *Temporal redundancies* are represented by areas which, due to lack of motion, are similar between video frames. In typical video content changes between successive frames are small compared to the data in a single frame. In such circumstances temporal redundancies can be used to provide considerable data compression. [28] Among the most important standards for video coding on the Internet today are: MPEG-1 [9], Motion JPEG [9], and H.261 (ITU-T) [27]. MPEG-1 is a generic standard for video coding in the 1-1.5 Mbps bandwidth range. It also provides facilities for audio coding (see above section on digital audio) and maintains the synchronization between related audio and video streams. Motion JPEG is extension of the JPEG still image compression standard, which allows video sequences to be coded. H.261 is a video coding standard designed for use over ISDN connections, which have a bandwidth of $p \times 64$ kbps[4].

2.2.2 Time-Independent Media

Time-independent media, typically text or images, are not intrinsically related to any points in time. The playback of such media in a multimedia presentation is controlled by

[4] *p* is an integer between 1 and 30

externally imposed time dependencies. For example, an image may be displayed for the duration of a particular audio stream, which means that the temporal endpoints of the audio stream are imposed on the image. It is normally possible to prefetch time-independent media objects completely before they are required, which makes the related presentation issues trivial in comparison with time-dependent media.

2.2.3 Media QOS

The QOS of media items is mainly related to how well an item can be presented according to its internal temporal structure, that is, its sampling rate or frame rate. The requirements on the network and presentation system become more severe as the information content or fidelity of the transmitted signal is increased. For time-independent media types this discussion does not directly apply, since the presentation of such items typically involves retrieving the complete item before presenting it.

Audio

Audio signals are extremely sensitive to transients such as inaccurate sample values or variations in the playback rate. For example, the omission of a single sample during the playback of an audio stream may introduce a sharp transient in the rendered audio waveform. In the frequency domain this translates to a brief burst of high-frequency noise[5] that is clearly audible as a “click”. If a very short portion of audio data is lost, a possible corrective measure is repeating the previous data block of similar length. This will reduce the high-frequency noise of the introduced transient by smoothing out the point of discontinuity in the waveform [29].

To illustrate the time-critical nature of audio, consider a CD-quality signal with a sampling rate of 44100 Hz. This leads to a requirement of delivering one audio sample to the output stage approximately every 23 μs. On the other hand the amount of data associated with one sample is relatively small – on the order of 16 bits.

[5] The high frequency content of the introduced transient can be explained using Fourier analysis. The sharper a transient is in the time domain, the higher the frequencies required to represent it in the frequency domain. The frequency domain representation is obtained by taking the Fourier transform of the time-domain signal. [18]

If the playback rate of a sampled audio signal deviates from the original sampling rate, this will change the frequency content and lead to audible artifacts such as pitch shifting [21]. These phenomena make interruptions in playback challenging to handle in the case of digital audio. Since corrective measures when data is late are seldom practical, most approaches concentrate on ensuring that the audio stream arrives uninterrupted at the output channel. This is normally done by buffering data close to the output channel (at the client workstation), in such a way that reasonable transport delays can be accommodated without affecting the data flow at the output [29]. When buffering fails to ensure uninterrupted playback, it is possible to employ special algorithms to stretch an audio signal in time (slow it down) without affecting the frequency content. However, this approach is processing-intensive and thus not always feasible. Furthermore, the results of such stretching may not be of satisfactory quality.

Video

Human perception of video signals is not time-critical to the same extent as for audio. In particular it is difficult for a human to detect the exact frame rate of a video signal. It is impossible, for example, to distinguish between the frame rates of 25 frames/s and 24 frames/s. This makes it possible to handle a single lost frame by reusing the previous frame, without disturbing side effects for the viewer [29]. For a massive loss of frames, the perceived video quality will naturally deteriorate and make the video appear staggered.

For 25 frames/s the interval between frames is 40ms, which should be compared to the sample interval of 23 μs for CD-quality audio. This implies that there is a time window several orders of magnitude larger for processing each video frame than for processing an audio sample. On the whole, however, the amount of data processing required in order to present a video LDU is significantly larger than the amount needed for an audio LDU.

Table 2.1 presents some of the key QOS-related properties of the audio and video media types. In summary, video requires considerable processing power but has relatively loose synchronization requirements between LDUs. Audio requires less processing but delivery needs to be extremely consistent with respect to timing between LDUs. Synchronization between different media items will be addressed in Chapter 3.

Table 2.1. *Comparison of QOS-related properties of audio and video in multimedia systems.*

	AUDIO	***VIDEO***
Interval between LDUs	10^{-6} s	10^{-3} s
Amount of data per LDU	~10 bit	~100 kbit
Perception of single lost LDU	“click” transient	negligible
Perception of several lost LDUs	disorienting pauses	discrete jump in video
Critical QOS factor	uninterrupted data flow	data processing power

2.3 Information Model

2.3.1 Information Model Properties

Multimedia information is generally organized and managed in *documents*. A document describes the components of a multimedia presentation together with information such as associations and temporal relationships between components. There is a wide variety of document models available, a situation which has arisen in response to the different requirements placed on multimedia processing. There is, however, a significant degree of overlap between many of these models. In this section a brief overview of key concepts and currently significant multimedia content models will be given.

The general issues that need to be addressed by a comprehensive multimedia information model can be summarized as follows [2]:

- synchronization
- integration of multiple media types
- composite media objects
- media object addressing
- hyperlinking
- input model for interaction

Synchronization is used to specify the temporal layout of a multimedia document; this will be covered in depth in the next chapter. A multimedia document architecture needs to allow the integration of various different media types within one logical document, and also to allow composition of media objects. An addressing scheme for media objects is an essential feature, which makes possible the use of hyperlinks. To specify a standard way of handling interactivity, an input model for interacting with the document is required.

2.3.2 Multimedia Content Models

HyTime

HyTime [30], which is an application of the ISO standard SGML [31], provides a standardized infrastructure for the representation of integrated, open hypermedia documents. In this context integrated means that all information within a document is linkable. Open implies that the link mechanism is independent of filesystem or network architectures. HyTime includes mechanisms for specifying locations in documents, associating document objects with hyperlinks and positioning document objects using a generic axes-based scheme. Each axis used to define a position can represent space, time or any other dimension. HyTime does not provide a model for interaction and, in accordance with the SGML philosophy, includes no presentation format modeling. HyTime is not directly used to represent hypermedia documents, rather it is used in conjunction with document type definitions which identify the specific elements of a given document format. HyTime is thus a meta-model used to define specific multimedia document models. [32], [30], [33]

MHEG

MHEG is an ISO standard [34] which defines a model for the processing and interchange of multimedia and hypermedia objects. It is intended as a standard for representing multimedia objects rather than documents. This is to make MHEG suitable for use in real-time presentations without requiring complex parsing, such as in the case of documents based on HyTime. As a consequence MHEG represents a nonrevisable form of multimedia, where all links and positioning are fully resolved. It is thus not a suitable format for highly interactive presentations. A typical usage scenario for MHEG would be to use a HyTime-based document model for *authoring* and converting this into MHEG objects for *presentation*. [33]

SMIL

SMIL, which is an application of XML [35], is a language for specifying simple, human-readable multimedia documents. The language is specified in a W3C recommendation [13], which differentiates it from the various proprietary multimedia formats in use on the web. The main features of SMIL are a model for specifying synchronization of media objects, hyperlink facilities and a simple layout specification format. Detailed treatment of SMIL is deferred to Chapter 4, where the language is examined in the context of the SynCope system.

MPEG-4

MPEG-4 is an emerging ISO standard which addresses the coding of aural, visual, and audiovisual media objects. Media objects can be natural or synthetic, corresponding to recordings of real world events and computer generated content, respectively. MPEG-4 describes how various media objects can be combined to form compound objects known as scenes. Standardized methods for multiplexing and synchronizing media object data are also provided, to provide for flexible transmission in heterogenous network environments. Finally, MPEG-4 defines models for interacting with audiovisual scenes generated at the client. [36]

One of the original intents of MPEG-4 was to provide very low bitrate coding for audiovisual programs [28]. The abstractions currently provided by MPEG-4, however, make it reasonable to treat the standard as a more general multimedia information model. It can be argued that MPEG-4 conceptually represents the current state of the art in representing multimedia information. However, the standard is still clearly a work in progress, particularly with regard to implementation.

2.4 Network Model

The packet-switched network architecture and data communications protocols that form the technical core of the Internet were not originally designed for real-time transport of multimedia data. In response to requirements of media-rich content on the Internet, various additions and improvements to the initial protocols have been developed. Moving from the “best effort” environment of the traditional Internet to a situation where the level of service can be explicitly addressed and managed, is still largely a work in progress. This re-engineering of the existing Internet infrastructure has proven to be a very challenging task.

Network transmission is currently the single most important factor affecting the perceived quality of Internet-based multimedia applications. As an example, private consumers of Internet services typically have a bandwidth of about 30 kbps at their disposal. For anything but the most basic media, this is a very severe limitation indeed. In this section we will examine some of the key protocols which enable the use of real-time multimedia on the Internet.

2.4.1 Basic Internet Protocols

The backbone of the Internet protocol family is formed by the protocol suite commonly referred to as TCP/IP [46]. The Internet Protocol (IP) is responsible for organizing data into packets and routing these to their destinations on the Internet. The Transmission Control Protocol (TCP) resides on top of IP and provides a reliable end-to-end transport service between Internet locations. TCP is connection-based and guarantees that packets will reach a certain destination in the order they were sent, or an error will be reported to the sender. The reliability features of TCP lead to the use of techniques such as *acknowledgement* and *retransmission* which make the protocol rather unsuitable for real-time communications. The User Datagram Protocol (UDP) is another transport service which resides on top of IP. In contrast to TCP, UDP is unreliable and connectionless. UDP transmits data packets as fast as possible to the indicated destination address; it does not retransmit lost packets or ensure correct ordering. UDP is typically used in situations were timeliness is critical and retransmitted packets are irrelevant (transmitting the current time is an example of this). Thus, UDP is the natural choice for real-time data transmission using TCP/IP. A variety of application level protocols exist on top of UDP and TCP, including the real-time protocols considered in the next section. The web is based on the Hypertext Transfer Protocol (HTTP), which is used to retrieve hypertext documents (and other objects) over the Internet. [47]

2.4.2 Real-Time Protocols

To enable the communication of real-time data in a TCP/IP context, a number of application level protocols have been introduced. Figure 2.2 presents these protocols in relation to the TCP/IP suite.

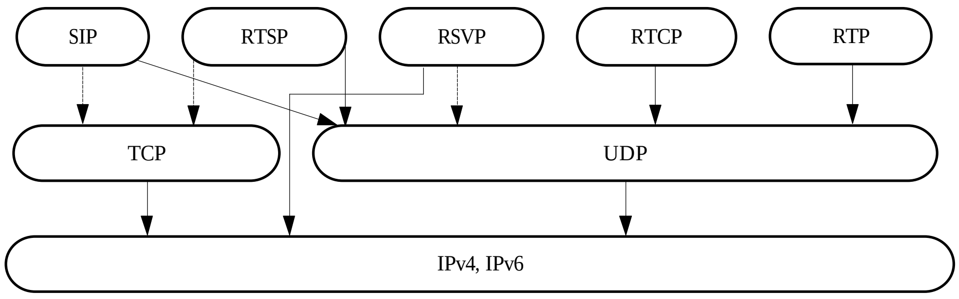


Figure 2.2. *Protocol stack for Internet multimedia, after [37].*

Real-Time Transport Protocol (RTP) and Real-Time Control Protocol (RTCP)

The Real-Time Transport Protocol (RTP) [48] handles transport issues specifically related to real-time data. RTP includes another protocol, Real-Time Control Protocol (RTCP), for managing RTP sessions. The protocols are typically used on top of UDP, but can be used with other packet-based protocols as well. Some of the main responsibilities of RTP / RTCP are:

- packet sequencing
- synchronization
- payload identification
- QOS feedback
- encryption

Packet sequencing is required to reorder packets that may have been received out of order over UDP. Synchronization of media streams is required to convey the interval between playback of successive packets in a single stream. When several streams are present, for example an audio and a video stream, they need to be synchronized to each other (lipsync). RTP packets are characterized by a specific payload, which describes the media encoding used for the stream. It may be necessary to dynamically switch payload types during a session, for example if network conditions change significantly. RTCP allows receivers to provide feedback on the quality of reception. This information may, among other things, indicate whether there is a need to change payload types in a session. [37], [75] *Resource Reservation Protocol (RSVP)*

The Resource Reservation Protocol (RSVP) [49] is used by applications to reserve specific QOS for their data streams. An application wishing to reserve resources from the network uses RSVP to transmit the corresponding request. RSVP carries the request to each network node involved in the transmission of the stream. At each node RSVP attempts to make a resource reservation by calling two decision procedures: *admission control* and *policy control*. Admission control determines whether the requested resources are available and policy control determines whether the client has permission to make the reservation. If both tests succeed, the host makes the configurations required to reserve the desired QOS. [10]

Real-time Streaming Protocol (RTSP)

Real-time Streaming Protocol (RTSP) [50] is a protocol for initiating and controlling the delivery of both stored and live multimedia streams over the Internet. The main concept of RTSP is providing a session-like abstraction for delivering one or more media streams to a single client or multicast destination. The main responsibilities of RTSP are

- initiating a session
- controlling a session while active
- signaling the end of a session

Initiating a session includes providing an identifier for the session and, for each included stream, negotiating media encoding parameters suitable for the client and current network conditions. The transport protocol and destination address is also determined during the setting up of a session. The idiomatic transport protocol in this scenario is RTP. A session may be controlled using requests such as PLAY, RECORD and PAUSE. The logical media time of the streams may be set using the PLAY request, which provides a form of random access to the stream data. When the client sends a TEARDOWN request, the media server will terminate the current session. [38]

Conferencing Protocols

Multimedia conferencing, such as Internet phone calls involving two or more parties, impose requirements specific to the management of *conferencing sessions*. In scenarios involving actively contacting and inviting parties to join a session, session management includes tasks such as locating the called party, negotiating media and communication parameters, and determining whether the called party wishes to be reached [38]. For Internet conferencing the preferred way of handling these tasks is through the use of the Session Initiation Protocol (SIP) [37]. For events that are announced beforehand, such as multicast panel discussions, the Session Description Protocol (SDP) [51] can be used to announce the nature and location of the session. [38] and [39] provide detailed descriptions of these issues.

The preceding chapter has outlined, in fairly broad strokes, the technological issues involved in the field of distributed multimedia systems. The following chapter examines in more detail the concepts related specifically to multimedia synchronization.

3 MULTIMEDIA SYNCHRONIZATION

3.1 Basic Concepts

An essential aspect of any multimedia presentation is the way time-dependent and time-independent media objects, which make up the presentation, are arranged in time. In the literature such *temporal relationships* have been extensively explored from various viewpoints, to some degree using overlapping terminology. In this text the term *synchronization* is used to refer on a general level to the specification and enforcement of temporal relations between media objects.

Temporal relations between media objects can be divided into *natural* and *synthetic* relations [17]. Natural temporal relations are implicit at the time of data capture, such as the interdependence between the audio and video tracks in the recording of a motion picture sequence. Synthetic relations are explicitly specified between independently captured or generated media objects. An example of this is the construction of a motion picture by editing together various sequences.

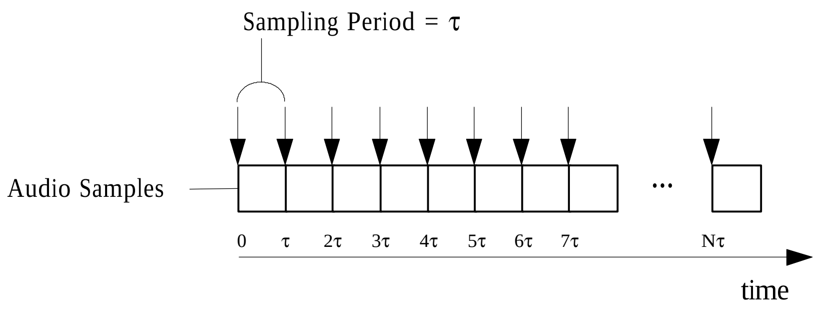


Figure 3.1. *Intramedia synchronization for a sampled audio stream.* The sampling period τ defines the temporal relationship between successive samples.

The temporal relations within a given time-dependent media object, for example the time-interval between samples in an audio stream, are known as *intramedia* synchronization (Figure 3.1). Relations between several independent media objects provide *intermedia* synchronization [7], [2]. Intermedia synchronization can be addressed at different levels of granularity, such as the *fine-grain* synchronization relating audio samples to video frames (lipsync) or *coarse-grain* synchronization specifying the playback of entire audio and video clips (Figure 3.2). Whereas coarse-grain synchronization specifies the temporal endpoints (beginning and end) for playback of a given media object, fine-grain synchronization further requires that the *media time* in synchronized objects advances along a common timeline [7].

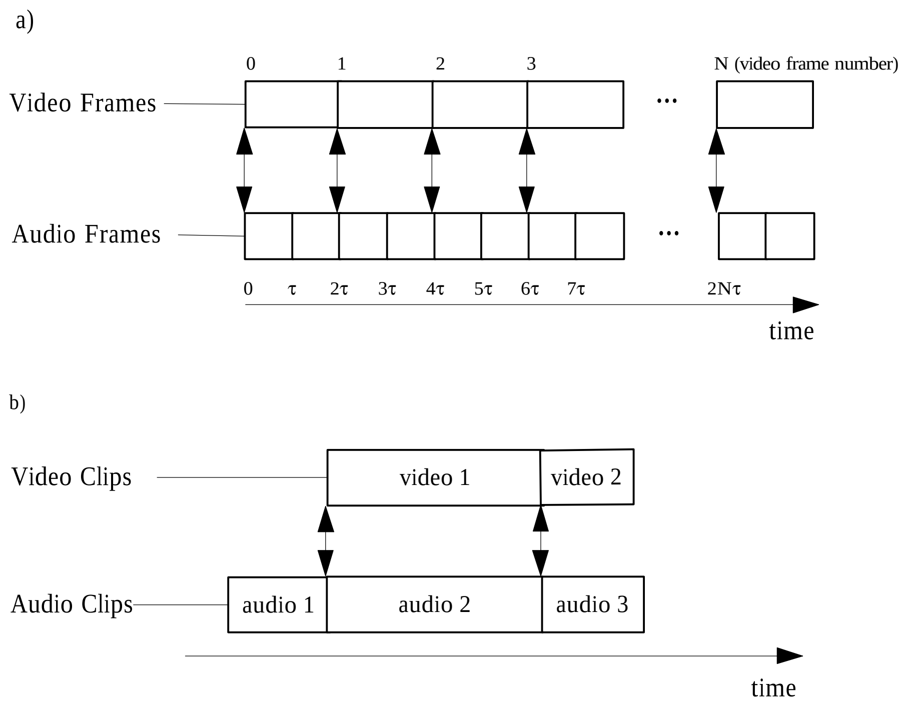


Figure 3.2. *a) Intermedia synchronization (fine-grain) between an audio and video stream. Synchronization is achieved by binding the streams to a common timebase. For each interval 2t that the timebase advances, the video stream advances by one frame and the audio stream by two frames.*

b) Intermedia synchronization (coarse-grain) between a set of audio segments and related video clips. The synchronization specifies temporal relations between the endpoints of various media objects.

3.2 Synchronization Reference Model

The concepts of multimedia synchronization may be classified using a layered model[6] introduced by [40] and further developed in [6] (Figure 3.3). The model provides four layers of abstraction through which a multimedia application can access synchronization

[6] The layer model presented here is somewhat closely related to the three-level model presented in [7], which consists of the *physical level* (media layer), *service level* (stream layer) and *human interface level* (object layer).

services. Each higher layer encapsulates the details of underlying levels, trading flexibility for a more intuitive programming model.

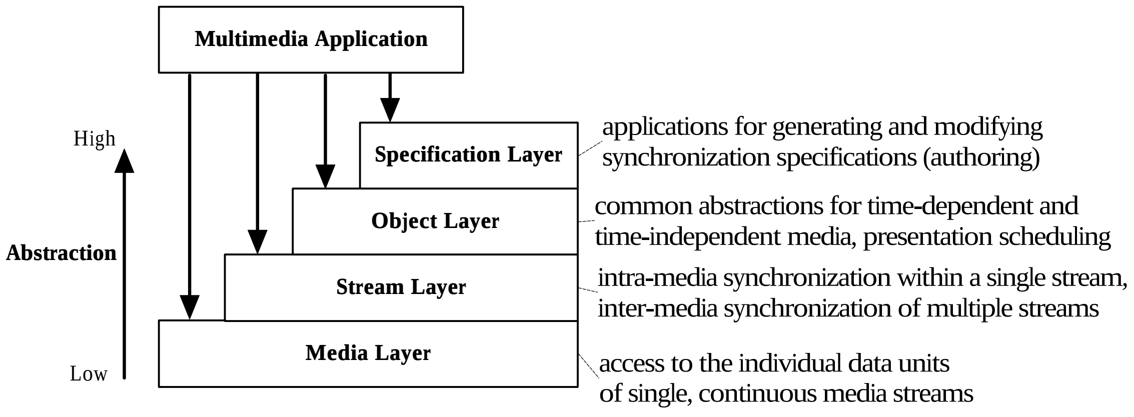

Multimedia Application
High
Abstraction
Low
Specification Layer
Object Layer
Stream Layer
Media Layer
applications for generating and modifying synchronization specifications (authoring)
common abstractions for time-dependent and time-independent media, presentation scheduling
intra-media synchronization within a single stream, inter-media synchronization of multiple streams
access to the individual data units of single, continuous media streams

Figure 3.3. *Synchronization reference model according to [6].*

3.2.1 Media Layer

The *media layer* provides a multimedia application with the capability to perform operations on single, continuous media stream. A media stream is considered to be made up of a sequence of logical data units (LDUs), such as audio samples or video frames, which are to be played back at a specified rate. At the media layer the application is responsible for the LDUs of a stream being played back at correct intervals, thus enforcing intra-media synchronization. Typically this is achieved by processing LDUs in a read-write loop, with a flow-control mechanism, such as a clock, providing the required timing. The UML [63] diagram in Figure 3.4 illustrates this point. When operating at the media layer, the synchronous presentation of time-independent media objects is the responsibility of the application.

Some media layer implementations provide support for interleaved media streams, where the LDUs of several streams are combined into one logical stream. In such cases, of which MPEG data streams are an example, the media layer also handles intermedia synchronization of the component streams.

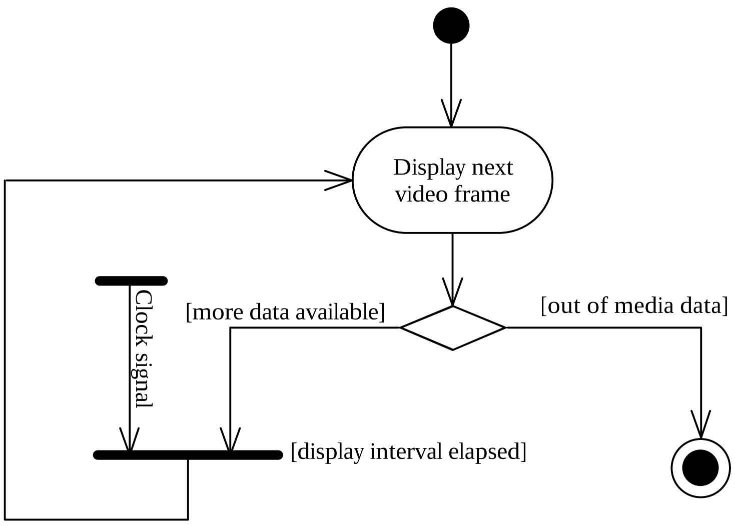


Figure 3.4. *UML activity diagram for video stream playback at the media layer.*

3.2.2 Stream Layer

The *stream layer* allows control to be exercised collectively on groups of streams, as well as on a single stream. The model at this layer is focused on providing means for the management of inter-media synchronization. Intramedia synchronization is implicit and individual LDUs are not exposed to the application.

At the stream layer, grouped streams are played back in parallel, controlled by a common clock or *time-base* [14]. The main responsibilities of an application are the grouping, starting and stopping of media streams. While the application is not required to operate in a real-time environment, stream playback is assumed to occur under real-time conditions. The handling of time-independent media is typically based on attaching *events* to specific points in media streams. Such events will trigger messages to the application at the specified times. After receiving an event message the application is responsible for taking appropriate action, such as activating or deactivating a time-independent media object.

In the case of stream layer intermedia synchronization, LDUs of the various streams may be subject to different delays. This leads to LDUs related through synchronization being presented at slightly different times. The time difference between two or more synchronized LDUs is referred to as *skew* [2], [6]. The maximum allowed skew is an important measure of QOS at the stream layer. *Lip synchronization* between related audio and video streams is a common scenario in which skew is an important parameter. [73] presents results of experiments regarding human perception of skew. In one experiment, users were asked whether they found the audio and video of a TV news speaker to be “in sync” or “out of sync” under varying skew conditions. Figure 3.5. summarizes the main findings of the experiment. It was found that for skew values of less than 80 ms in either direction (audio leading video or vice versa) most test subjects did not detect a synchronization error and found the streams to be “in sync”. Conversely, for skew over 160 ms in either direction most subjects found the data to be “out of sync” and, in general, not acceptable. In the two regions marked as “transient” the detection of synchronization errors was dependent on the distance of the speaker on the video; the closer the speaker was the easier errors were detected. A notable point was that video ahead of audio was easier to tolerate than the reverse situation. [73]

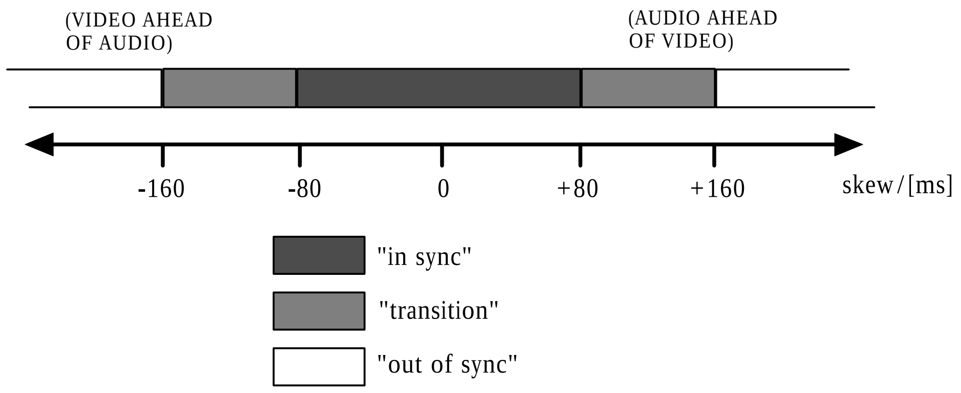


Figure 3.5. *Perception of audio/video synchronization under varying skew conditions (after [73]). For positive skew values the audio stream is ahead of the video stream and vice versa.*

3.2.3 Object Layer

The *object layer* provides a unified abstraction for any type of media, time-dependent or time-independent. The main responsibilities of the object layer are constructing and executing a complete presentation schedule, provided a synchronization specification.

The object layer encapsulates various operations required to correctly play back a composite multimedia presentation. Such operations are the preparation of media objects for playback, management of media streams using the stream layer and presentation of time-independent media objects. The management and presentation of time-independent media objects is typically implemented using system-level services not exclusively related to multimedia applications (user interface or windowing toolkits, for example).

The services provided to an application by the object layer operate on the presentation (document) level. The main functionality is included in methods for starting and stopping the presentation as well as *temporal access control* (TAC) operations such as:

- reverse
- fast-forward
- fast-backward
- midpoint suspension
- midpoint resumption
- random access
- looping
- browsing

[17]. Whether and how these operations are implemented is dependent on the media type, application and available multimedia resources. For instance, reverse playback of media coded using certain types of predictive algorithms may not be possible on workstations without specialized multimedia hardware.

3.2.4 Specification Layer

The specification layer is responsible for producing a *synchronization specification* representing a certain document. The layer contains any tools, such as authoring systems, used to create or modify synchronization specifications. Rather than postulating concrete systems the specification layer is characterized by the various temporal models (see next section) used to define synchronization relationships between media objects in a presentation.

The interface between the specification and object layers is characterized by the format in which a specification layer tool communicates a specification to the object layer. This interface is of considerable importance due to its relation with multimedia interchange standards such as MHEG ([41], [33]), HyTime ([32], [30], [33]) and SMIL ([13], [42]).

These are all examples of interface formats used for communication between the specification and object layers.

3.3 Temporal Models for Synchronization

Various models are available for the specification of synthetic temporal relations between media objects. Each model provides a different set of abstractions, with varying properties, strengths and weaknesses. The requirements placed on a temporal model for multimedia are manifold; on one hand, the model should provide a comprehensive way of formally specifying any relevant scenario. On the other hand, the model should provide an abstraction which is intuitive and cohesive enough to be a useful tool for multimedia authors.

In [6], Steinmetz provides a comprehensive summary and categorization of temporal models used in synchronization specification methods. A less detailed, but quite common approach uses two main categories for classification: models based on *temporal instants* and *temporal intervals*, respectively [7], [16]. Favoring clarity over completeness, the latter approach will be used here to describe key concepts of temporal models. As a concrete example, the SMIL Time Model [13] will be examined in the context of the general classification.

3.3.1 Instant-Based Models

Temporal instants are moments of zero extent in time [17]. Instants can be considered as points on an axis representing a one-dimensional time space. For any two temporal instants, represented by a and b, there are three possible temporal relations:

- $a < b$ (a before b)
- $a = b$ (a simultaneous to b)
- $a > b$ (a after b)

These three relations ($<$, $=$, $>$) are the *basic relations* used by instant-based models. Instants which have not yet occurred can furthermore be characterized by *indefinite relations*, formed as disjunctions of the three basic relations. Such relations can be used to express that the exact order in which instants will occur is not known. There are $2^3 = 8$ indefinite relations: $\emptyset$, $\leq$, $<$, $=$, $>$, $\geq$, $\neq$, ?, where ? represents any basic temporal relation.

∅ represents the empty set. It should be noted that the basic relations constitute a subset of the indefinite relations. [16], [17].

Applying the above instant-based temporal relations to the domain of multimedia makes possible certain simplifications and modifications. For the presentation of multimedia, the difference between the relations < and ≤ can be disregarded, since < can represent an arbitrarily small displacement in time. For compactness, < can thus be used to the exclusion of ≤. By the same reasoning the relations ≥ and ≠ can be replaced by > and ?, respectively. ∅ implies that no temporal relation exists between two points, a situation which does not naturally occur in multimedia presentations. Thus, for instant-based specification of temporal relations in a multimedia context, the set of relations {<, =, >, ?} is sufficient [16].

Timeline

A common model employing instant-based temporal intervals is the *timeline* [16], [17], in which *events*, representing the temporal boundaries of media objects, are ordered along a time axis (see Figure 3.6). Any two events are related by one of the instant relations <, =, >. Owing to the fact that all events are ordered along the time axis, the '?' relation cannot be used in the timeline model. This restriction makes the model somewhat inflexible. On the other hand, the timeline approach is quite intuitive and easy to use in authoring situations. QuickTime[43] is an example of a multimedia architecture based on the time-line approach. The HyTime standard uses a generalization of the time-line model, in which any number of *virtual time axes* can be defined [30], [32].

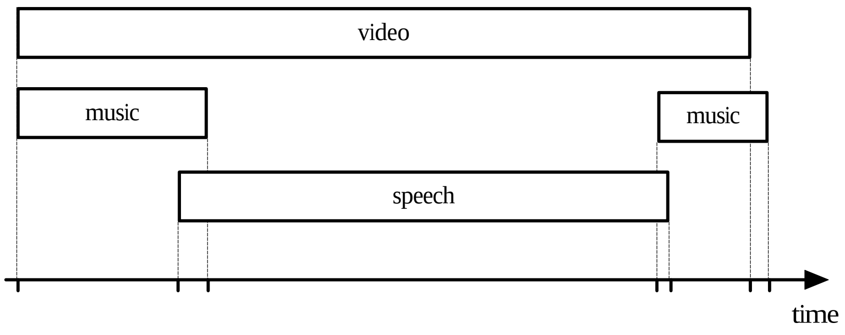


Figure 3.6. *Multimedia presentation specified using a timeline representation. The events starting and stopping playback of media objects are represented on the time axis. [17].*

Temporal Point Net

Temporal point nets are an instant-based approach to defining temporal relations between media objects (Figure 3.7). The relations are based on events which establish temporal equalities (=) and inequalities (<, >) between instants in a presentation. In contrast to the timeline approach the relation ? may also be used to specify an unrestricted temporal relation between two instants. This makes the temporal net a more flexible model than the timeline. [44], [16]

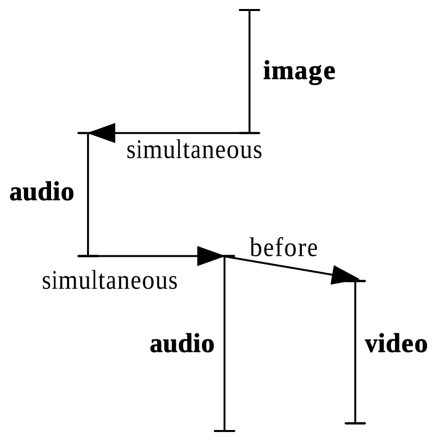


Figure 3.7. *Temporal point net defining multimedia synchronization.*

3.3.2 Interval-Based Models

Temporal intervals are defined by their endpoints (here represented by *a* and *b*). The length of such a temporal interval is *b-a*. In his widely acknowledged paper [45], Allen introduces the thirteen basic relations that can exist between temporal intervals. Fig 3.8. depicts seven of these relations, the remaining six can be constructed by taking the inverse of each relation except *equals*. For example, the inverse relation of *before* is *after* (or $before^{-1}$), where *a* before *b* is the same as *b* after *a*. In analogy to the instant-based situation, there exist $2^{13} = 8192$ indefinite interval relations

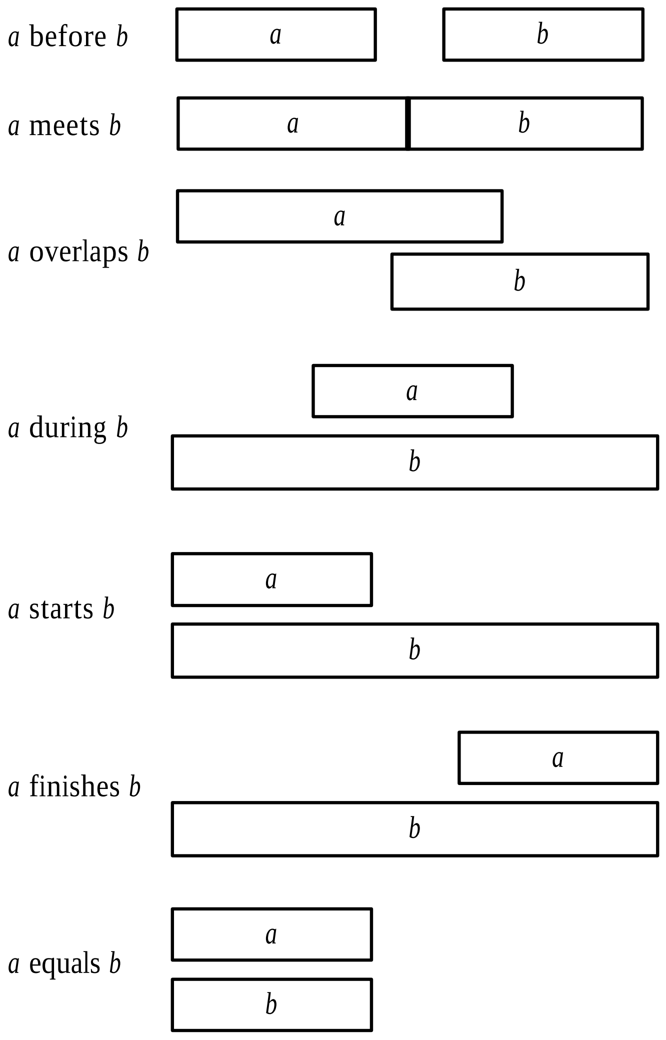


Figure 3.8. *Basic interval relations (after [45], [17]).*

Object Composition Petri Net

Object Composition Petri Nets (OCPN) [7] can be used to specify and represent interval-based temporal relations. Figure 3.9 presents an example OCPN which specifies the temporal constraints for a slide show with synchronized audio. Each *place* (circle) in the net has a duration and represents the playout of a media item or a delay. *Transitions* (arrow) represent synchronization conditions. For example, in Figure 3.9 the vertical bar at the left specifies that the playout of image1 and audio1 should start simultaneously. In terms of the interval relations in Figure 3.8, this implies the relation image1 *starts* audio1. Another interval relation present in the figure is image1 *meets* image2, expressing sequential playback of the items.

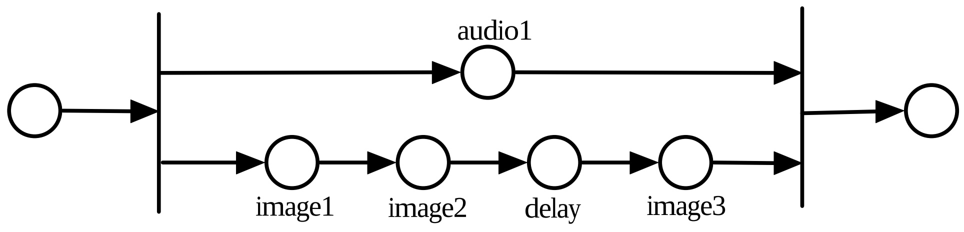


Figure 3.9. *OCPN representation of a slide show with a series of images synchronized to audio. There is a delay between the second and third images.*

3.3.3 The SMIL Time Model

The temporal relations in a SMIL document are defined using the SMIL Time Model [13], which in the following is described on a general level. The model is based on a subset of the thirteen interval relations defined in the previous section, namely the *meets* and *equals* relations. These relations represent sequential and parallel playback of media items, respectively, and are represented by the SMIL *synchronization elements* `<seq>` and `<par>`. The following SMIL fragment

```
<par>
    <audio/>
    <seq>
        <video/>
        <video/>
    </seq>
</par>
```

can be translated to a timeline structure as depicted in Figure 3.10. This type of temporal specification approach is also known as a *hierarchical control flow-based specification* [6], as the synchronization behavior can seen as a flow of control through a hierarchical *tree* structure.

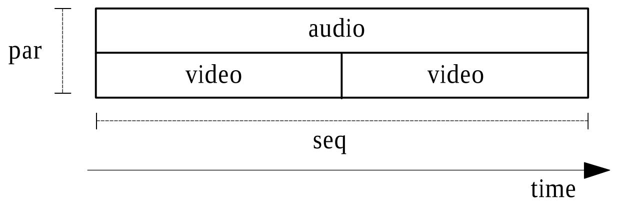


Figure 3.10. *Timeline interpretation of SMIL fragment.*

The SMIL Time Model is not limited to a basic hierarchical specification, however. The model is made more expressive by the *synchronization attributes*: `begin`, `dur`, and `end`, which can be specified for each synchronization element. The `dur` attribute specifies an explicit duration for a synchronization element, whereas `begin` and `end` are used to specify synchronization behavior for the endpoints of an element. Endpoint synchronization can be used, for example, to synchronize the beginning or end of a synchronization element to some event in another synchronization element. An example of such an event is the media time of an element reaching 5 seconds. The addition of endpoint-based synchronization attributes makes the SMIL Time Model something of a hybrid between an interval-based and an instant-based model. A full description of the SMIL Time Model is given in [13] and elaboration on some related issues is provided by [42].

3.4 Synchronization in a Distributed Environment

Operating in a distributed environment brings some added complexity to the domain of multimedia synchronization. The issues raised are mainly consequences of the synchronization information being distributed across a network. [6] presents an overview of the subject matter. Only one specific topic will be addressed here, as it directly relates to the implementation of the SynCope system.

As previously stated, a multimedia presentation is associated with a synchronization specification for describing temporal behavior. In a distributed environment it is necessary to decide how the synchronization information is transported to the presentation system. Three main approaches are identified:

1. use of a separate synchronization channel
2. multiplexing of media streams to provide synchronization
3. delivery of complete specification before start of playback

The first alternative is mainly useful in “live” broadcast situations, where the synchronization information is not available beforehand. The second alternative is used, for example, by MPEG. Multiplexing is straightforward for the presentation system to handle, since there is no external synchronization information. The main drawback is that QOS negotiations regarding media coding becomes more complicated, as media objects must be addressed in combination. The third approach, which is used by the SynCope system, is typically possible for synthetic synchronization relationships (such as authored multimedia documents). SynCope obtains synchronization information for presentations by parsing SMIL files.

4 The SynCope Presentation System

The SynCope presentation system provides an object model and implementation to serve as an extensible framework for the development of Java-based multimedia applications. It includes services for the presentation of composite multimedia documents, from parsing documents to scheduling and playback of individual media objects. The system is designed to work in a distributed environment, specifically the Internet. A simple programming interface enables applications to use multimedia presentations transparently as Java user interface elements. Applications requiring more advanced features can extend the SynCope system, for example to include new media types or user interaction models.

4.1 System Requirements

In this section the requirements for the SynCope system will be specified, with regard to functionality, design guidelines and additional constraints related to the use of specific base technologies.

4.1.1 Multimedia Information Model

The requirements for a multimedia presentation system are to a large extent determined by the type of documents it is intended to process. Multimedia document formats provide an information model [2] for specifying the content, synchronization, layout and behavior of multimedia presentations. In terms of the synchronization reference model (see Section 3.2), the information model acts as an interface between the specification and object layers. Implementing a presentation system for documents with a specific information model involves transforming the elements of that model into concrete system requirements.

For the SynCope system, the Synchronized Multimedia Integration Language (SMIL) was chosen as the primary document specification and interchange format. The main reasons for using SMIL as the base-level specification format are:

- SMIL has standard status as a World Wide Web Consortium (W3C) recommendation [13].
- SMIL is an application of the Extensible Markup Language (XML) [35], which is also a W3C recommendation and already widely deployed on the Internet. This

enables general-purpose tools (such as parsers) for XML processing to be used with SMIL documents.

- due to the markup-based format of SMIL, documents are human-readable and can be produced using a wide variety of tools, ranging from advanced authoring environments to simple text editors. Well-known web techniques, as well as more recent XML techniques, for dynamic generation and personalization of documents can also be employed to generate SMIL presentations. This widens the range of possible application scenarios.

The SMIL Information Model

The core of the SMIL information model is the time model described in section 3.3. Other key features include facilities for specifying document layout, hyperlinking and providing alternative content based on various user and environment attributes.

SMIL basic layout language provides a default format for specifying presentation layouts for SMIL documents. SMIL viewers may also support other layout languages, such as Cascading Style Sheets (CSS). SMIL basic layout is consistent with the visual rendering model of the Cascading Style Sheets 2 specification [52].

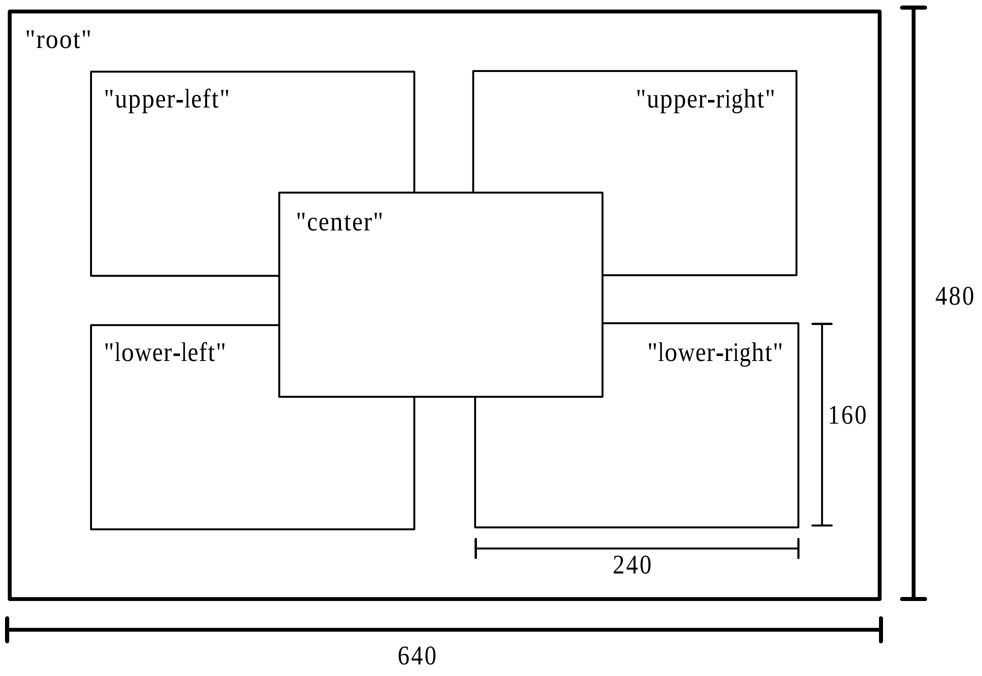


Figure 4.1. *Example of a SMIL presentation layout, as specified using the SMIL Basic Layout Language.*

Figure 4.1 shows a conceptual example of a layout specified using the SMIL basic layout language. A sample SMIL file including a full layout specification can be found in Appendix A. It should be mentioned that SMIL basic layout in its current form has no provision for *dynamic layouts*, that is, layouts that may change over time. The main implication of this is that layouts can change only through replacement. Transitional scenarios, such as panning media objects across the screen, are not possible as of now.

The hyperlinking features of SMIL are implemented by two elements providing slightly different link semantics. The SMIL `a` element is similar to the `A` element in HTML4.0 [53], and is used to define a uni-directional link from a media object in the source document to a destination resource. The destination resource can be a SMIL document, a specific element within a SMIL document or some other type of addressable web resource. Additionally, SMIL defines the `anchor` hyperlink element. This has semantics similar to image maps in HTML 4.0, with the addition of temporal partitioning. That is, an `anchor` element allows a link destination to be associated with a given spatial and/or temporal subpart of the source element. The spatial partitioning is specified using the `coords` attribute, whereas temporal subparts are specified by `begin` and `end` attributes. This distinction is illustrated in Fig 4.2.

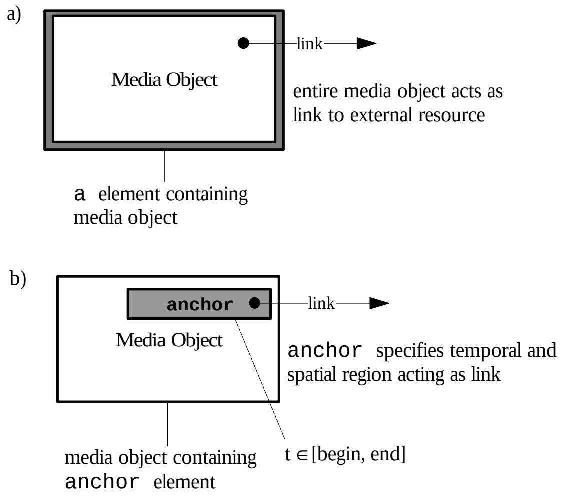

a)
Media Object
link
entire media object acts as link to external resource
a element containing media object
b)
anchor
Media Object
link
anchor specifies temporal and spatial region acting as link
media object containing anchor element
t ∈ [begin, end]

Figure 4.2. a) *Hyperlink as defined by an* `a` *element.* b) *Hyperlink as defined by an* `anchor` *element.*

SMIL `switch` elements can be used to specify alternative media elements. The element to be rendered will be selected subject to the values of one or more test attributes. The main test attributes available to qualify media objects are network bandwidth, screen dimensions, color resolution and language. For a full description of these and other test attributes, see [13]. The following SMIL fragment, consisting of a simple `switch` element, uses the `system-language` test attribute to select an audio element to render:

```
<switch>
       <audio src=”audio_swedish.mp3” system-language=”se” />
       <audio src=”audio_finnish.mp3” system-language=”fi” />
</switch>
```

4.1.2 Multimedia Distributed Processing Model

The object-oriented software paradigm is well suited to the design and implementation of distributed multimedia systems, as has been extensively documented in the literature [54], [2], [3], [1], [55]. An integral part of the synchronization reference model (see section 3.2) is the object layer, which gives an object-oriented view of multimedia synchronization issues. The base technologies underlying SynCope are primarily object-oriented by nature. Java provides a comprehensive model for object-oriented programming and widely used for object-oriented software engineering [56], [57]. XML, and by inheritance SMIL, is a language which lends itself well to be processed in an object-oriented manner. The Document Object Model (DOM), which is a W3C recommendation [58], specifies standard interfaces for such processing.

Among the particular processing requirements for multimedia systems are real-time system services, continuous media system services and distributed object management [2]. In SynCope, the Java 2 platform [59] is used both as the implementation language and as the provider of system services. The main reasons for using Java as a base platform can be summarized as follows:

- Java is rapidly becoming a de-facto standard for application development, particularly in distributed environments such as the Internet. It provides a highly familiar paradigm for developers of object-oriented Internet applications.

- Implementations of standard Internet technologies, such as XML parsers and networking protocols, are readily available for the Java platform. Java also provides ubiquitous support for distributed object services, both Java-specific (RMI [60]) and language-independent (CORBA [61]).
- The Java Media Framework (JMF) is an extension which provides media stream services for the Java platform. Despite being available since 1997, JMF has not been extensively used in real-world applications. A central aspect in developing SynCope was evaluating the applicability of JMF for demanding multimedia applications, and to identify specific limitations.

4.1.3 System Requirements

The primary functional system requirements of SynCope follow from the information model defined by SMIL. Most of these requirements are inferred from the descriptions included in the SMIL specification [13]. Some specific features typically expected in multimedia systems (playback controls, for example) are also included in the requirements. The more general objective of producing a flexible and extensible system imply a further set of design constraints. This section summarizes the key requirements of SynCope.

Functional Requirements

As part of its basic functionality the SynCope system should:

- provide the functionality required to display multimedia presentations specified using SMIL 1.0 [13]. As a secondary requirement, it should be possible to generate alternative views, such as timeline diagrams, of documents.
- provide the following basic temporal access control (TAC) operations [17] for the playback of presentation documents: *start*, *stop*, *pause*, *random access*
- allow for fine-grain media stream synchronization within the capabilities of JMF

Design Constraints

By further design constraints the system should also:

- allow for extensibility and evolution, in particular with regard to media formats, document and synchronization specification formats, user interaction models

- be designed specifically for the Java environment, favoring the utilization of Java’s possibilities over full generality

In many ways the SynCope system addresses similar issues as the ISO standard PREMO (Presentation Environments for Multimedia Objects) [3]. The main differences between SynCope and PREMO are related to scope and level of abstraction. PREMO specifies a very general middleware layer for distributed multimedia presentation systems, without committing itself to any particular document architecture, media type or interaction model. SynCope, in its current incarnation, provides multimedia presentation services for the specific technologies Java, JMF and SMIL. Thus, SynCope exists at a lower level of abstraction, as an instance of the type of presentation environment that could be implemented using PREMO. A mapping of SynCope model elements to the higher level concepts of PREMO, while a distinct possibility, is beyond the scope of this project.

4.2 Technical Architecture

4.2.1 Context

The SynCope system can be described as an application framework, which uses a set of base technologies to provide domain-specific services for synchronized multimedia applications. Figure 4.3 illustrates the architectural context of the system. The SynCope presentation system forms the client part of a distributed client-server architecture. The other main components are the network, which is based on standard Internet protocols, and servers providing multimedia documents and individual media objects. The basic case of client-server communication in the SynCope framework is the delivery of finite media objects as requested by the client, using HTTP. The architecture can be made to accommodate more sophisticated protocols through available extension mechanisms. For the case of live media streams RTP is the entry level transport protocol, as it is supported by JMF. It should be noted that the RTP protocol is mainly intended for use in conferencing scenarios, rather than in the playback of authored presentations.

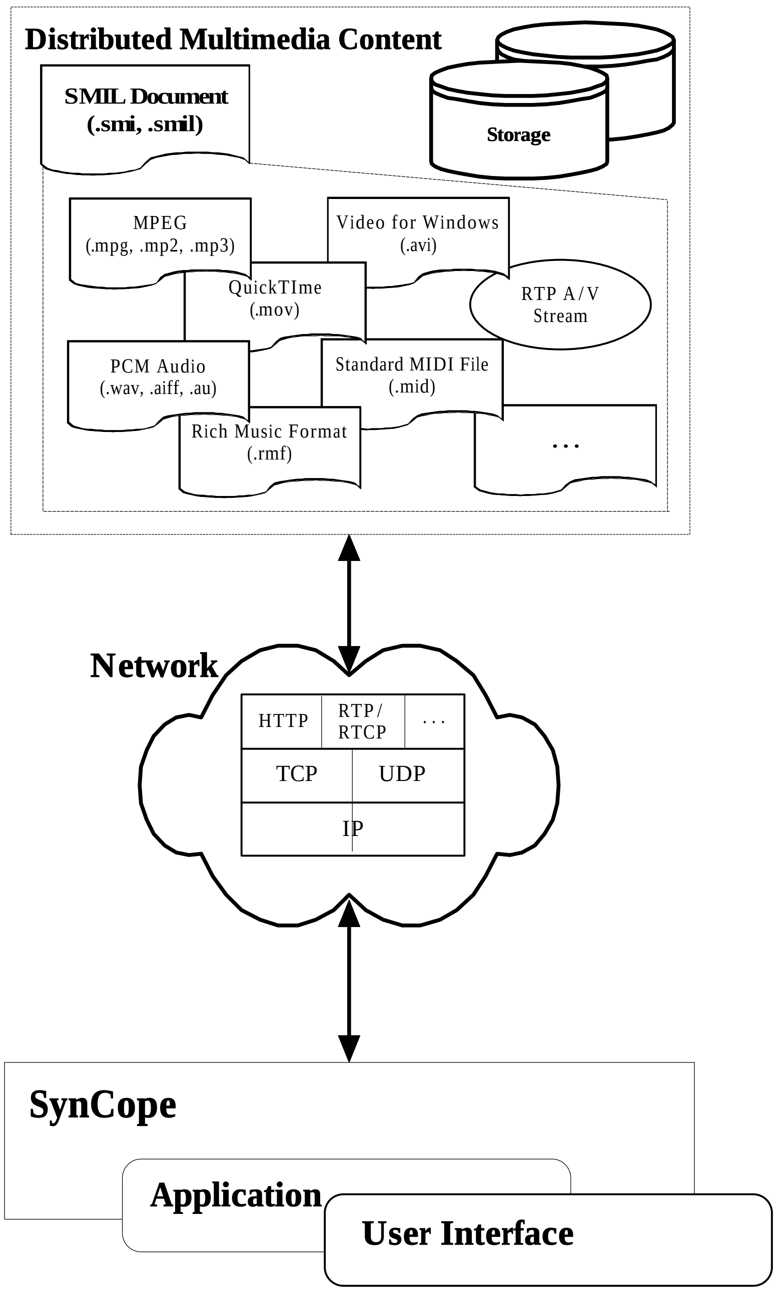


Figure 4.3. *The architectural context of SynCope.*

The Java-based implementation environment of SynCope can be viewed as layered architecture, in which each layer provides programming abstractions of a higher level than the one below. Figure 4.4. gives a broad view of the package hierarchy involved.

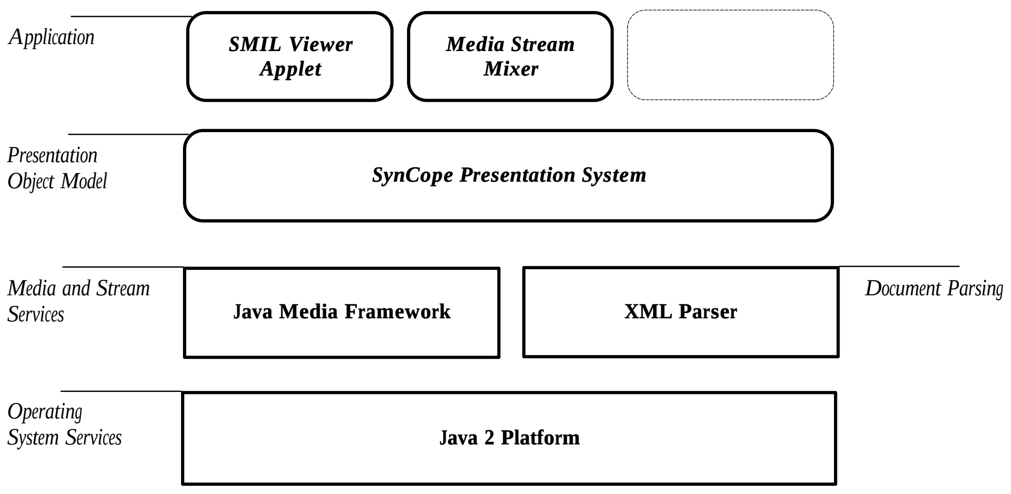


Figure 4.4. *Package hierarchy in SynCope.*

Java 2 Platform

The Java 2 platform [59] provides operating system services for SynCope. It serves as an abstraction between the upper layers in the architecture and the concrete operating system, providing a certain degree of operating system-independence.

In addition to the programming language and general object model, SynCope uses the following key services provided by Java 2

- filesystem and networking facilities (Internet-based) are used for communication and data transfer
- threads are used to perform a variety of asynchronous tasks. The most important scenario is event dispatching and enforcement of the playback schedule.
- the Swing user interface toolkit [62] is used to construct the user interface specified by the SMIL document

Java Media Framework (JMF)

SynCope makes use of media services provided by the Java Media Framework (JMF) [14]. JMF provides means for controlling individual media streams in a way that is independent of delivery mechanisms, transport protocols, media types and encoding formats. It also contains the infrastructure needed for extending the framework with new protocols and media types.

To provide support for advanced media formats, local installation of platform-specific JMF extension libraries is required on client machines. An all-Java version of JMF, which can be used in any Java-enabled environment without separate installation, is also available. This version, which is based on Java applications dynamically loading JMF classes from a network server, imposes considerable restrictions on the media formats that can be used. Some performance penalties are also to be expected when using the all-Java version. SynCope was developed based on JMF version 1.1.

Figure 4.5 presents the key Java interfaces defined by JMF in Unified Modeling Language (UML) notation [63].

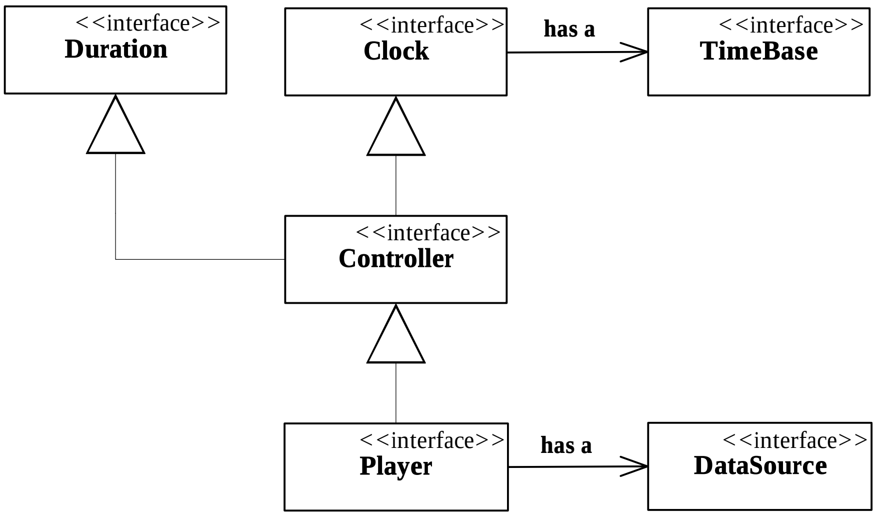


Figure 4.5. *Java Media Framework (JMF) interface hierarchy.*

The `Clock` interface defines the basic timing and synchronization operations available for the control of media object playback in the JMF. A `Clock` has a `TimeBase` object which defines the rate at which time advances in the clock. The JMF time model is based on the concepts of *time-base time* and *media time*. The time-base time represents the flow of real-time, independent of any media objects. It cannot be stopped, reset or otherwise controlled. The media time of a media object (which has a `Clock`) represents the current point in time within the media stream of the object. Methods are defined for starting, stopping, setting and defining scaling for the media time of a `Clock`. Figure 4.6 shows the conceptual relationship between media time and time-base time. [14].

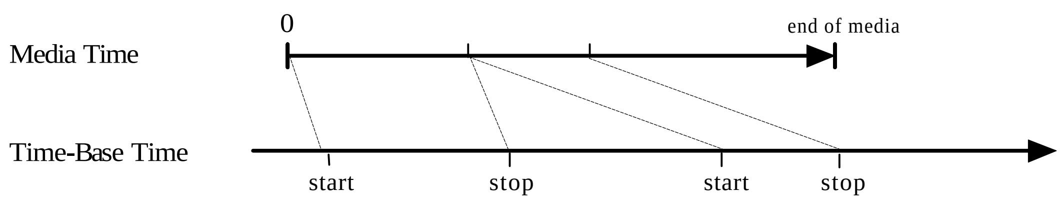


Figure 4.6. *Relationship between media time and time-base time in the JMF* [13]. *The media time can be controlled using* `Clock` *methods for starting, stopping and setting the time value. In the started state, the media time is propagated according to the time-base time. The time-base time flows without interruptions.*

The JMF `Controller` interface (extends `Clock`) defines a state model for controlling the transitioning of a media object through various resource allocation states. The interface also defines methods for registering event listeners, which will be notified through events of state changes or other occurences in the `Controller`. The `Player` interface (extends `Controller`) adds functionality related to concrete media streams. A `Player` has a `DataSource`, which represents the location and delivery protocol associated with a specific media object. `Player` also provides access to user interface components representing the media object in question. [14].

In JMF the `Player` instance required to present a specific media object is determined in part based on the MIME [65] content type of the media. JMF uses a form of the Factory Method design pattern [64] to decouple the creation of `Player` objects from the concrete `Player` classes. This allows `Player` objects to be created transparently based on the properties of the media streams they are to present. The scheme makes it easy for developers to add new media types and corresponding `Player` implementations to the repertoire of JMF, without requiring changes to existing code.

Simple API for XML

Content information is extracted from SMIL files using an XML parser, which adheres to the Simple API for XML (SAX) specification [66] for event-driven XML processing. SAX was developed as a community effort by members of the XML-DEV Internet mailing list and is currently widely supported by XML parsers.

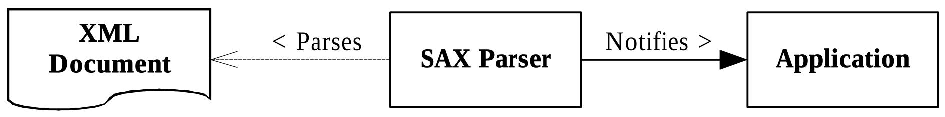


Figure 4.7. *Event-driven parsing of XML documents using a SAX-compliant parser.*

A SAX-compliant XML parser generates events to demarcate elements (represented by markup tags) in an XML document and communicate information about element attributes (Figure 4.7). An application receiving such events can use them to construct an internal representation of the document content.

4.2.2 Architecture

Top-Level View

The top-level view of the internal architecture of SynCope is based on a variant of the Model-View-Controller (MVC) architecture pattern (Figure 4.8) [67].

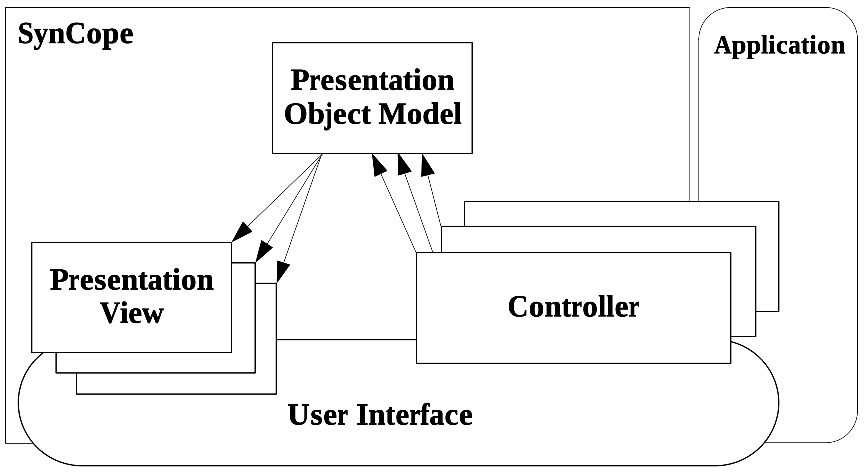


Figure 4.8. *Model-View-Controller pattern in SynCope architecture.*

The *model* is at the heart of the MVC pattern, constituting an object-oriented realization of the entities and behavior required of a system. In SynCope this role is played by the Presentation Object Model. A *view* provides an external representation of the state of a model. The basic view in SynCope is a window or other similar surface, based on the specified layout, in which the elements of a presentation are rendered. *Controllers* allow users or other systems to provide input which may affect the model's state. For SynCope, the main controllers are user interface elements for starting, stopping and setting the media time of a presentation. The type of controller referred to in the MVC pattern is quite different from the JMF controller concept; the two should not be confused.

Presentation Object Model

The Presentation Object Model (POM) is constructed from a SMIL document, using the output from an XML parser. Parser events are sent to a `SMILBuilder` object, which uses the information to construct an object representation of the document. This scenario is known as the Builder design pattern [65], because the object in question (`SMILBuilder`) knows how to "build" a SMIL document object. By employing this pattern, the use of XML is encapsulated and hidden from the application. This decoupling becomes relevant if the need arises to use some other data format than XML as a basis for SMIL presentation objects. Such modifications can be achieved by substituting another Builder object, without changing any other application or system code. For example, one could implement a Builder which constructs a presentation by combining a template SMIL file with data from a relational database.

The SynCope subsystem which realizes the POM is responsible for maintaining a `SMILDocument` object, which is a direct reflection of the SMIL document structure. The most crucial responsibility of the presentation model is resolving and maintaining the temporal relations between presentation elements. The POM consists of two parts, a *static model*, which corresponds directly to the XML representation of a SMIL document, and a *dynamic model*, which is responsible for controlling media objects and dispatching events during runtime. Figure 4.9 shows how these responsibilities are divided among the main Java packages of SynCope.

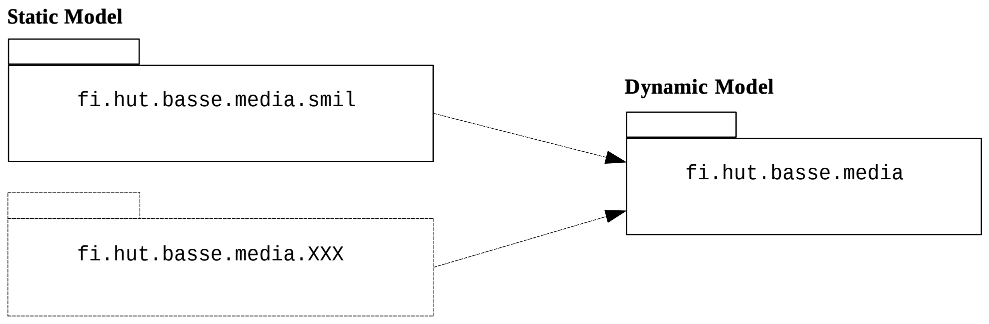


Figure 4.9. *Key Java packages in SynCope.*

It should be noted that the dynamic model is not dependent on the implementation of the static model, whereas a dependency exists in the opposite direction. This separation makes it possible to introduce static models based on other document types, without making changes to the run-time implementation. However, if the introduced information model contains synchronization relationships that cannot be expressed using SMIL, new features will have to be added to the dynamic model.

Figure 4.10 summarizes the types of objects processed in the two models. The static model is mainly focused on maintaining a structural representation of the multimedia presentation. Most of the objects in the static model have no time dependencies. The dynamic model is mainly concerned with objects that do have time dependencies, such as controllers used to manage media playback and event dispatchers that perform tasks at specific times. Objects in the dynamic model are typically *active*, that is they have an associated thread of control.

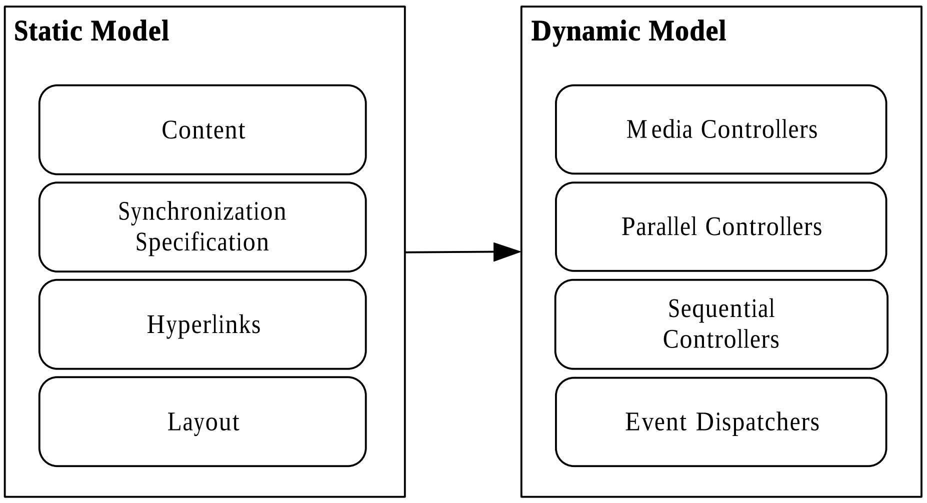


Figure 4.10. *Conceptual view of the objects processed by the static and dynamic models, respectively.*

Static Model

The static model in SynCope is based on the abstract base class `SMILElement`, which defines attributes and methods common to all classes representing some element defined by the SMIL specification [13]. In fact each element specified in the SMIL Document Type Definition (DTD) is represented by a corresponding class in the SynCope system. The static object structure of a presentation is implemented using the

Composite design pattern, which describes containment hierarchies for objects derived from a common base class. As shown in Figure 4.11 the implication is that any *SMILElement* can have any number of children and at most one parent (all of which are also *SMILElement*s). This construct is used to build a tree structure of SMIL element objects. *SMILElement* defines methods for traversing the tree to extract information about the presentation. This can be utilized by other objects, for example to generate alternative views of the synchronization specification defined for the presentation.

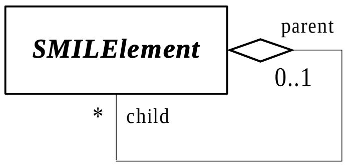


Figure 4.11. *Use of the Composite design pattern in the static model.*

Fig 4.12 depicts the key classes involved in defining the synchronization properties of a presentation. For any presentation tree, the root object is a SMILDocument and has no parent elements. SMILDocument also provides the methods through which applications control a presentation. SeqElement and ParElement define temporal relations for playback of their child elements, based on sequential and parallel execution, respectively. From a synchronization standpoint, BodyElement is equivalent to SeqElement. MediaElement contains the synchronization attributes for a specific media item. A full class diagram of the static model is provided in Appendix B.

Dynamic Model

The main responsibility of the dynamic model is enforcing the synchronization relationships defined by the static model. This is handled by various Controllers, as shown in Figure 4.13. The class AbstractController provides functionality common to all controllers implemented by SynCope.

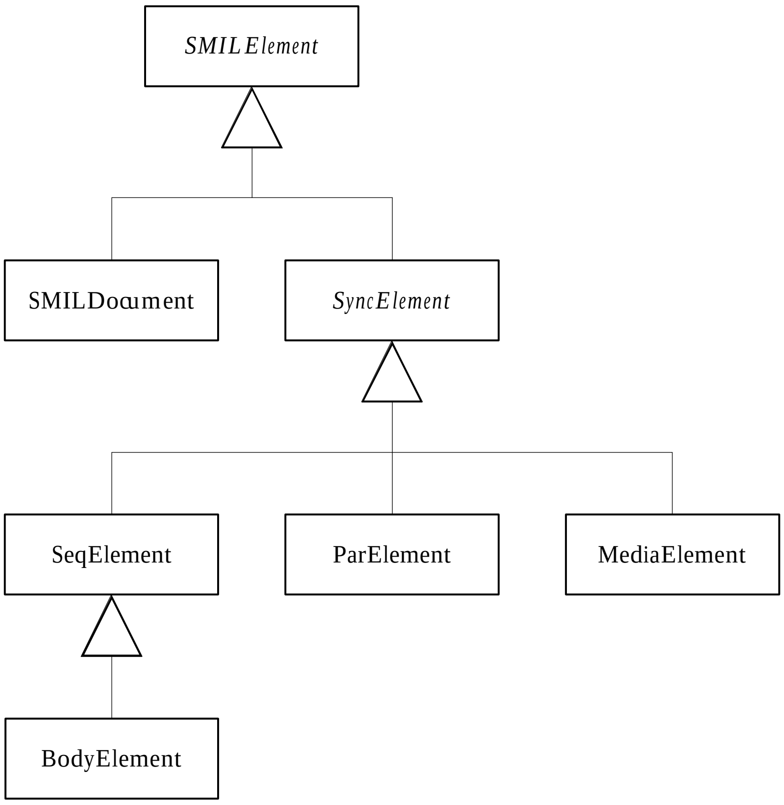


Figure 4.12. *Synchronization-related classes in the static model class hierarchy.*

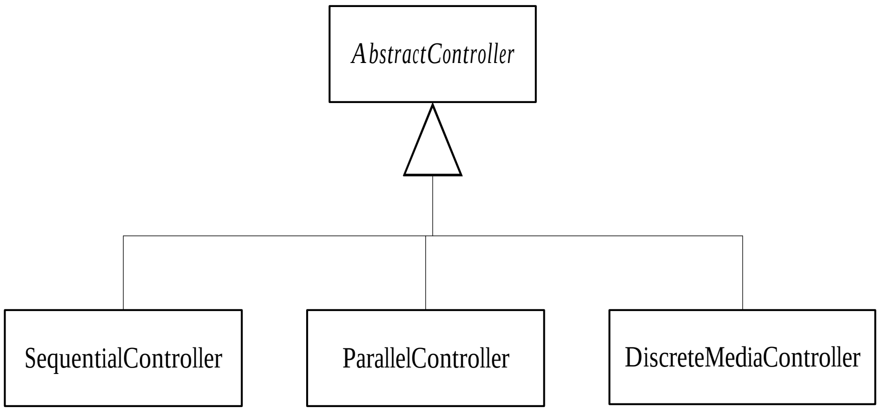


Figure 4.13. *Hierarchy of SynCope controllers.*

Sequential and parallel synchronization is implemented by the classes `SequentialController` and `ParallelController`. Both of these mirror the tree structure in the SMIL Time Model and are used to control other controller objects, known as *children*. For instance, the children of a `SequentialController` might be three controllers each representing an audio clip. As the parent controller of these clips, the `SequentialController` is responsible for the clips being played out in sequence. Time-independent media is integrated into the common model through the `DiscreteMediaController` class, which manages the lifespan of time-independent media items. A more complete class diagram of the dynamic model is provided in Appendix C.

Scheduling

The SynCope system performs scheduling in both the static and dynamic models. This situation is conceptually similar to the *compiletime* and *runtime temporal formatters* presented in [44]. As a presentation object is constructed from a SMIL document and initialized, the static scheduling attempts to determine an absolute schedule based on the SMIL Time Model. The static schedule can be fully constructed only if the duration of each media object can be calculated, either based on the intrinsic duration of the object or on timing attributes explicitly specified in the SMIL document.

The static schedule is merely an estimate of how playback of the presentation will progress. Particularly in a distributed environment it is quite normal for delays to occur in the communication and playback of media objects. This uncertainty implies that the absolute schedule can not be treated as a timeline, with each event fixed in time. To address this, as the static schedule is computed, a corresponding structure is created in the dynamic model. This part of the schedule is event-based with the principle that each controller dispatches events to its children. These events are dispatched according to the progress of the presentation. In the case of a `SequentialController`, an event to start the next child controller will never be sent before an end notification has been received from the previous child. The message flows associated with this scheme are shown in Figure 4.14 for the sequential case and Figure 4.15 for the parallel case.

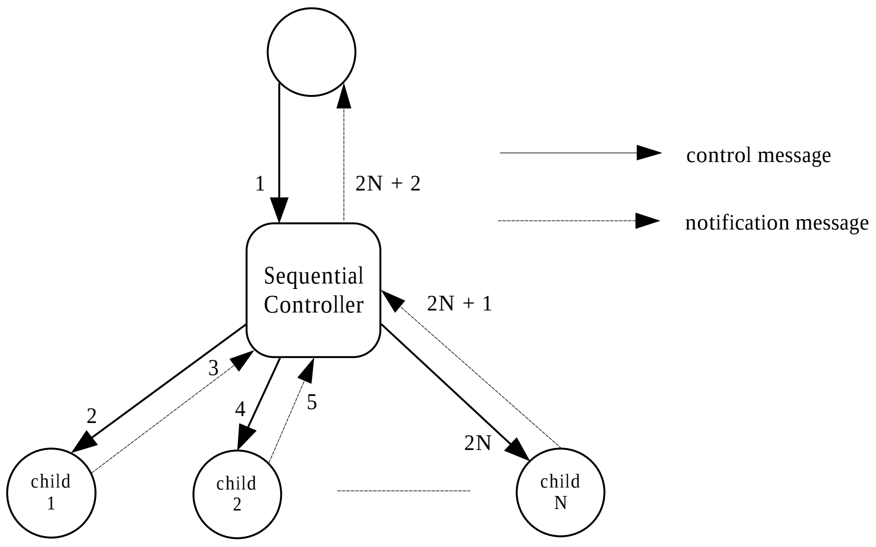


Figure 4.14. *Synchonization messages in a SequentialController.*

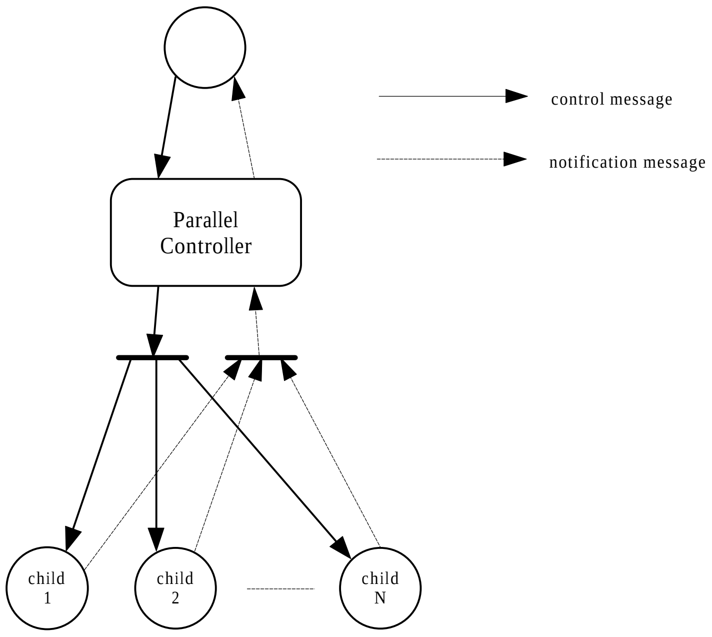


Figure 4.15. *Synchronization messages in a ParallelController.*

User Interface

A presentation can have various views associated with it. The normal view is obtained by rendering a presentation according to temporal and spatial constraints specified in the document. Another possible view is a graphical representation of the presentation's synchronization specification – such as a timeline. Presentation editors might provide a variety of different views.

Controllers and views together form the user interface for a specific multimedia application. The available controllers are dependent on the current presentation view. For example, when a presentation is rendered in the standard view, there will typically be controllers for at least starting, pausing and stopping playback of the presentation. Within the presentation, hyperlinks constitute another form of controller, as do other types of interactive user interface elements. As implied in Figure 4.8, the responsibilities regarding controllers is shared between the application and the SynCope system.

5 SYNCOPE IMPLEMENTATION

The previous chapter examined the requirements, architecture and design of the SynCope system. In this chapter the designed system will be classified in terms of the synchronization reference model presented in Section 3.2. On a more concrete level, applications implemented to test the system are presented, followed by a description of the discovered implementation issues.

5.1 Classification of SynCope

The key characteristics of the SynCope system are related to multimedia synchronization issues. In this section the system is classified according to the above mentioned synchronization reference model. In particular, the roles of various base technologies on which SynCope builds are put into the context of the reference model. Figure 5.1 presents a summary of how the various elements of the SynCope system relate to the reference model.

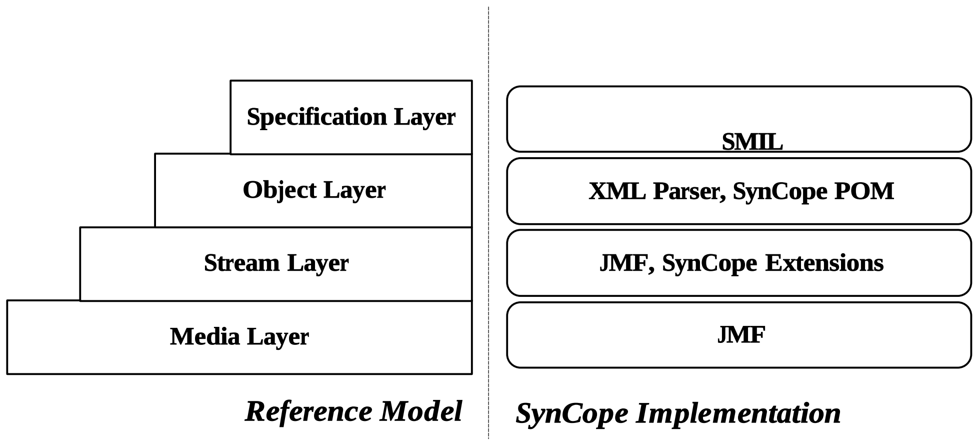


Figure 5.1. *Relationship between synchronization reference model and SynCope implementation elements.*

5.1.1 SynCope Specification Layer

SynCope does not provide services explicitly belonging to the specification layer, only the interface between the specification and object layers is defined. The specification layer is responsible for producing a SMIL document, which contains the synchronization specification for a presentation. The methods and tools for producing the document are purposely not specified, retaining a variety of options such as basic text editors, visual authoring tools or dynamic generation. The SMIL document acts as input to the object layer.

5.1.2 SynCope Object Layer

The main part of the SynCope system implementation is concerned with providing object layer services. Input SMIL documents are analyzed using an XML parser in order to build a structural object representation of the synchronization specification. The object structure mirrors the SMIL document structure and provides an implicit schedule for the presentation. Detailed scheduling is handled by the Presentation Object Model as specified in Section 4.2.2.

The presentation of a document is managed at the object layer through close communication with stream layer entities. Scheduling constraints are mapped to stream layer method invocations, which control the playback and synchronization of media streams.

5.1.3 SynCope Stream Layer

The basic stream layer implementation in SynCope is provided by JMF. SynCope adds some extensions to JMF, in order to achieve a consolidated programming model. That is, the JMF abstractions for time-dependent media items are extended to cover composite media streams with sequential or parallel playback, as well as time-independent media items. With these extensions an image, for example, can be treated as any other media stream. These extensions are defined in the dynamic part of the POM.

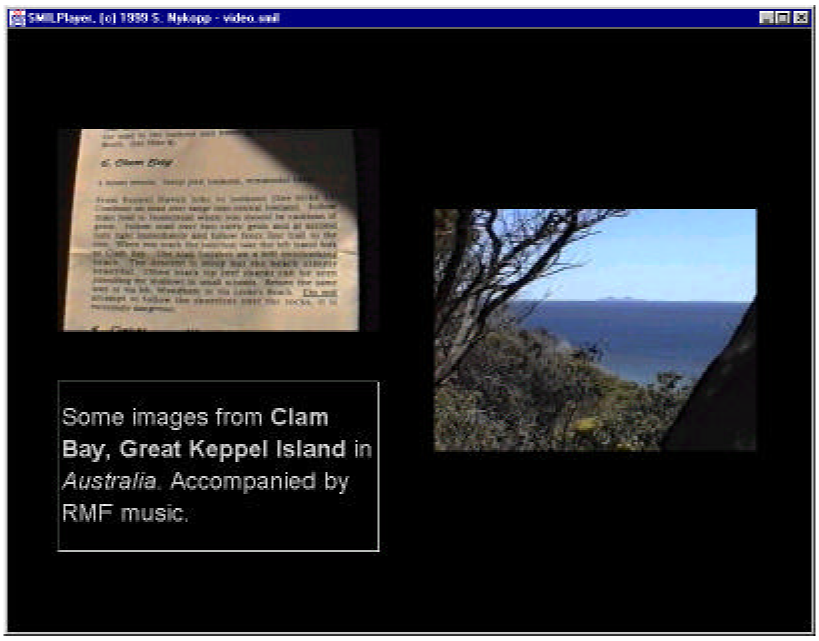
5.1.4 SynCope Media Layer

The media layer implementation in SynCope is provided by JMF in a way that provides no direct programmatic control. The current version of JMF is designed to provide a platform-independent multimedia framework with stream layer abstractions as the lowest level of access. The main implication of this is that JMF applications, such as SynCope, do not have access to individual LDUs. On the other hand, it has been announced that it will be possible to access individual LDUs, in upcoming versions of JMF.

5.2 Test Cases

5.2.1 SMIL Presentation Viewer

The most comprehensive test application developed for the SynCope system is a basic viewer for SMIL presentations. Most of the functional requirements of SynCope are


SMILPlayer, (c) 1999 S. Nykopp - video.smil

Some images from **Clam Bay, Great Keppel Island** in *Australia.* Accompanied by RMF music.

Figure 5.2. *Screenshot of SMIL presentation displayed using SynCope.*

demonstrated by this test case. Figure 5.2 shows a typical screenshot of the viewer application. The corresponding SMIL file is presented in Appendix A.

5.2.2 Audio Stream Mixer

In order to examine fine-grain synchronization and, in particular, the performance of JMF in such scenarios, a mixer for audio streams was defined as a test case. Figure 5.3 shows the control panel for this application in the case of mixing a MIDI and a PCM stream. The stream mixer uses JMF to play back streams synchronized to each other. As will be described in the next section, serious JMF performance issues were noted in relation to this type of scenario.

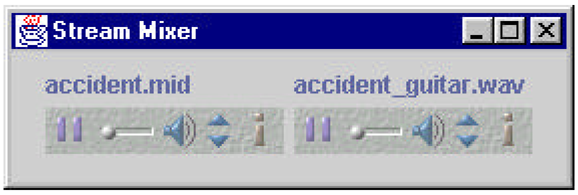


Figure 5.3. *Control panel for stream mixer with one MIDI and one PCM stream. Each stream has a panel with temporal access and volume controls.*

5.3 Implementation Issues

In the implementation of SynCope a number of issues remain unresolved. The most critical issues are related to the lack of real-time capabilities in the Java platform.

5.3.1 Real-Time Issues

Java does not provide facilities for associating hard deadlines with specific processing tasks. This follows from the way thread[7] scheduling is implemented in the Java virtual machine. A thread can be made to sleep or wait for a given number of milliseconds, but there are no guarantees that it will be activated again immediately after the specified interval. The event-based scheduling used in SynCope is implemented on these terms and, as a consequence, provides only a best-effort service. If the virtual machine does not activate the event dispatcher thread at the appropriate time, events may be dispatched too late. One cause for such disruptions is the Java garbage collector[8], which can cause user threads to interrupted while the garbage collector releases unused memory resources.

There are initiatives for adding real-time capabilities to Java [68], but currently the Java platform implements no such features.

---

[7] A thread is an abstraction used in multi-tasking systems to describe a sequential execution path within a program. Different threads are allotted time to run by a scheduler, which in the case of Java is represented by the Virtual Machine (Java execution environment).

[8] Garbage collection is the task of locating objects no longer in use by an application and freeing the memory and other resources used by these objects.

5.3.2 JMF Issues

The Java Media Framework imposed certain limitations on the implementation of SynCope, which significantly affect the performance and applicability of the system. Many major issues were consequences of the real-time problems described above.

A shortcoming of JMF, as an element of a comprehensive multimedia platform, is the lack of abstractions and facilities for handling QOS issues with regard to inter- and intrastream synchronization. The JMF API includes neither methods for specifying QOS requirements, nor methods for reliably recording metrics of QOS parameters describing the "current situation". This is made more problematic by the fact that JMF in some cases reports the media time of objects incorrectly. In the stream mixer test case it was found that a MIDI stream and a PCM stream, which to the listener clearly were out of synchronization, sometimes reported identical media times. JMF thus reported that the streams were synchronized when they were not. This apparent unreliability of the media time reporting mechanism in JMF makes it difficult to monitor whether streams are synchronized or not.

Starting synchronous playback of media streams is based on JMF knowing the *startup latency* of each media object. That is, JMF calculates the maximum time it will take to start all involved threads in order to determine a common instant when the threads will be started. This mechanism, however, proved somewhat unreliable in the current JMF implementation, as the startup latency for most media objects could not be determined by JMF. For such objects the startup latency must be guessed, which introduces an uncertainty in synchronously starting media streams.

JMF provides a method for using one media controller to control other media controllers, in a scheme similar to the *clock hierarchies* presented in [69]. In a clock hierarchy control messages, such as start and stop, are propagated along a tree-like containment structure. This feature is intended to automatically handle synchronous starting and stopping of several streams using one controller. However, this feature was unreliable and in tests streams synchronized using this method exhibited various levels of skew. The reason for this problem is likely to be related to the above mentioned latency calculation issue.

In summary it can be said that SynCope can only provide best-effort service with regard to synchronization, and is subject to some quite severe limitations. Many of these are due to JMF issues, whereas others are due to specific features of the Java platform. On the other hand, it should be said that the interfaces of JMF provide a fairly comprehensive set of abstractions. As Java and JMF implementations become more efficient, the applicability of these technologies to demanding multimedia scenarios is likely to increase.

6 Analysis, Conclusions, and Future Developments

6.1 Analysis and Conclusions

In this section the SynCope presentation system will be examined in the context of current trends within Internet-based multimedia, in order to provide some perspective on the issues involved. Based on this, potential strengths, weaknesses, opportunities, and threats are identified. The purpose of the analysis is to draw conclusions about possible uses of the system and determine whether a case can be made for further development of the system. To focus the discussion the main areas of interest in the analysis are identified as follows:

- potential user base
- presentation format qualities
- extensibility, openness, and programmability

6.1.1 SynCope and Current Trends

Potential User Base

In the field of WWW-based multimedia most real-world applications today operate in a browser environment, as provided by products such as Netscape Navigator and Microsoft Internet Explorer. Multimedia content is usually viewed using either a “plug-in” or a stand-alone application launched by the browser. Plug-ins can be used to embed multimedia content or applications within a web page, whereas stand-alone applications normally are represented by a separate window on the workstation desktop.

For multimedia client software to reach as many users as possible in the mainstream Internet audience, it is important that there be no complex downloading and installation procedures for the user to perform. One approach, already used by vendors such as Macromedia and RealNetworks, has the multimedia plug-in bundled with a leading browser product for simultaneous installation with the browser. Products distributed using this method are likely to reach a maximum audience. Another approach is specifying the download location of a plug-in in the WWW pages with content specific to that plug-in. The browser can then automatically locate the plug-in and initiate installation.

Making full use of SynCope calls for the presence or installation of two software products on the client-side workstation: a Java 2 runtime environment and JMF libraries. Java 2 is not currently included in the major browsers. However, it is available as a plug-in and bundled browser support is expected in the relatively near future. As stated in Section 4.2, a limited version of JMF can be used without any local installation procedures on the client. At present this is something of a low-end solution, however, and optimum performance requires that native libraries be locally installed. Based on the above observations, it can be said that the software requirements for using SynCope are likely to place it outside the reach of the mainstream audience for now. The most potential users are persons who are interested in using experimental Java-based technology.

Presentation Format Qualities

A considerable portion of the multimedia content available on the WWW today consists of isolated audio or video clips linked to from WWW pages. Depending on the plug-in or application used for presenting the clips, some synchronization properties may be specified using scripting languages (for example, JavaScript or VBScript). Popular formats for more robust presentations are, among others, Macromedia Flash and Macromedia Shockwave, both of which spring from the tradition of authoring multimedia using vendor-specific tools and formats. Flash is a format used for presentations based on vector graphics, whereas Shockwave is primarily a means for WWW-enabling multimedia content created using traditional Macromedia authoring products, such as Director. Java applets can also be used to present multimedia-type content. However, these applets typically provide their own formats and abstractions for defining presentations; presentation interchange is rarely addressed.

The above mentioned formats all provide the possibility for creating highly expressive presentations. However, the approaches conflict with some of the basic objectives of the Internet. The main problems are related to the reuse and refinement of content and information. Proprietary formats are typically, for most intents and purposes, opaque to any other entities than applications specifically designed to decode and present them.

This makes automatic processing of such content, in applications such as search engines, software agents, or automated conversion tools, very difficult.

SMIL, which is an application of XML, takes a different approach to the specification of multimedia presentations. It follows in the tradition of such standards as HyTime [30] (an application of SGML [31]), which uses descriptive markup to specify presentations. When compared to some currently available technologies, SMIL clearly has shortcomings as far as expressive power is concerned. Some specific issues are the lack of dynamic layouts (layouts that may change over time), the lack of facilities for specifying fine-grain media synchronization conditions, and the absence of a model for user interaction. Rousseau and Duda address some of these issues in [70]. However, the limited expressiveness of SMIL should be viewed in the context of what the language is intended to achieve.

Key requirements in the design of SMIL were simplicity and a declarative format as opposed to a scripting model. Scripting can add expressive power, but typically requires more effort in the production and maintenance of presentations. Since SMIL is based on XML, documents can automatically be processed by any application which recognizes XML data. Typical examples of such applications are content management systems, search engines, automated conversion tools, and software agents. In summary, SMIL is intended to solve a different set of problems than most of the currently employed solutions for WWW-based multimedia. The advantages and possibilities described above make SMIL highly interesting in spite of the lesser expressiveness.

Extensibility, Openness, and Programming Abstractions

For multimedia applications intended to explore new possibilities and quickly adapt to a changing technology landscape, extensibility, openness, and a robust set of programming abstractions are important features. It is in this area that the main strengths of a system such as SynCope lie, when compared to many currently popular solutions. Several different areas can be identified in which extensibility is required:

- media types
- network protocols
- document formats

SynCope addresses the two first areas through the use of JMF, which by its architectural approach decouples both media types and network protocols from applications. Integrating support for new media types and protocols is largely a straightforward matter, as long as no platform-specific native code is required. The SynCope system itself provides a framework into which the processing of new document formats can be integrated. This will typically require more effort than the two former types of extension, depending on the amount of new concepts and abstractions that need to be added to SynCope. Since Java is based on dynamically loading application code over a network, any developer can extend the system and make the added features available to users on the network. This process is transparent to the end user. For most plug-in oriented applications the addition of new features is more complex, since application code needs to be locally installed on client workstations.

The term openness is used here to describe the extent to which a given multimedia system adheres to interchange standards as well as the extent to which the system can be used from and communicate with various applications and systems. SynCope addresses the issue of openness through the technology choices documented in Section 4.2 and by exposing a considerable part of its services as application programming interfaces (APIs). The latter feature is also very much related to the issue of programming abstractions, which in SynCope are centered on the Presentation Object Model (POM), also described in Section 4.2. Programming abstractions, then, are essential to the development of new functionality and automation in multimedia applications. A system with sufficiently flexible programming interfaces can be integrated into a wide variety of applications. In this respect a beneficial characteristic of SynCope is the tight integration with the Java platform. This integration allows SynCope objects, such as presentation views, to be used transparently in any Java application. Robust programming abstractions also provide opportunities for various types of automation, for example in the case of automatic document format conversions.

6.1.2 Strengths, Weaknesses, Opportunities and Threats (SWOT)

Based on the above discussion, some conclusions can be drawn regarding the relevance of the SynCope system. Figure 6.1 presents a summary of the identified strengths, weaknesses, opportunities, and threats associated with SynCope.

Strengths	**Weaknesses**
use of Java provides platform-independence and feature-rich programming model supports the SMIL recommendation, which provides a simple, declarative format for multimedia presentations open and extensible interfaces for application developers	base technology has unresolved performance issues client-base with full Java 2/JMF support still quite limited expressiveness and flexibility issues in SMIL
Opportunities	**Threats**
achieving ubiquity through Java/JMF breakthrough in browsers innovation through synergy with Java or XML technology use in niche or domain-specific scenarios through extension and customization	performance issues in JMF remain unresolved faltering or diverging Java support in browsers SMIL recommendation retired and replaced by significantly different concepts

Figure 6.1. *SWOT-analysis of the SynCope system.*

Strengths

The main strengths of SynCope are related to the openness, extensibility and flexible programming abstractions it provides. The system can easily be integrated into Java applications. Through extension the functionality of SynCope can be modified or further developed, which provides opportunities for exploring widely different scenarios for using multimedia in Java applications.

Weaknesses

The immaturity of Java as a platform for the presentation of multimedia leads to performance issues, some of which are currently quite critical in the context of SynCope. Partially for the same reason the user base of JMF is currently quite limited. The limitations in expressiveness associated with SMIL are also something of a weakness.

Opportunities

A massive adoption of JMF, particularly in association with WWW browsers, would lead to a much larger potential audience for SynCope than is currently possible. Using the extensibility of SMIL, JMF, and SynCope together provides interesting possibilities for domain-specific multimedia applications. There are also clearly opportunities for innovation based on the synergy between Java and XML.

Threats

Perhaps the main threat against SynCope is the possibility that the performance issues of JMF remain unresolved. The level of support for Java in the major browsers is another concern, which has a direct bearing on the number of users available for SynCope. A slightly lesser threat is the possible failure of SMIL with a subsequent replacement by significantly different concepts. While not totally disastrous to the concept of SynCope, this could nonetheless make much of the current implementation obsolete.

6.1.3 Conclusions

The potential success and utility of SynCope lies mainly with how well it can be adapted to changing circumstances, such as the evolution of SMIL or widespread adoption of some other document format. The key question here is whether the design captures essential abstractions well enough to provide reusability in future scenarios. As a static system (SMIL viewer) SynCope is not particularly relevant, in particular due to the stated performance limitations.

A feasible and interesting role for SynCope is acting as a platform for prototyping and testing experimental multimedia applications. Integrated access to such the manifold features of the Java platform provides an excellent environment for testing new user interfaces, interaction types, and service distribution models. The performance constraints currently limit the possibilities of exploring scenarios requiring high-end processing power or high-precision synchronization.

6.2 Possible Future Developments for SynCope

The previous section presented an analysis regarding the relevance of SynCope as it stands at the end of this project. In this section, a more speculative approach is taken in order to present some visions on future possibilities for the system.

6.2.1 Presentation Components

One of the unimplemented concepts that emerged during this project was to extend the presentation model beyond the use of traditional media objects (such as audio and video). In this scenario any object fulfilling certain basic requirements could be integrated into a multimedia presentation. The presentation system would communicate with such a *presentation component* through an interface providing the following facilities:

- methods for obtaining user interface components (views and controllers) that represent the component
- methods for obtaining information about and controlling the flow of time in the component. In particular, mechanisms for *starting* and *stopping* the component, setting the *component time* and receiving notification on whether the flow of media time has stopped due to some event

The basic concept is that a dimension of time is added to the component. Analogously to time-dependent and time-independent media objects, presentation components could either have or not have internal time dependencies. In the latter case the view of the component would be treated as a time-independent media object. That is, it would be displayed over a period of time defined in the synchronization specification. An example of the time-dependent case is a Java component generating musical notation for a composition. A possible view of this component could be a user interface element into which musical notation is rendered, according to the internal time dependencies of the component. The notation component may, for example, present the score as a set of discrete images, each of which is displayed for a given number of seconds. The number of seconds for each image would be determined by the inner time scale of the component, as defined by a tempo attribute. The component may also contain a controller, for example in the form of a “Quit” button, which when activated would cause the component to deactivate itself. At this time, the presentation system would be notified that playback of the notation component has ended. The presentation system would then start any presentation components synchronized to the end of the notation component.

The possibilities of a scheme as described above are quite interesting, when combined with the possibilities for extending SMIL with features for describing domain-specific multimedia content. (These features could be related, for example, to the field of musical composition). SMIL elements corresponding to specific presentation components could be introduced, with attributes employed to provide initialization and configuration information to the component.

A possible approach for introducing presentation components into SynCope would be using the JavaBeans component architecture [71] as a basis, and refining this to include a well-defined interface for synchronization. Taking the concept one step further, one could define various types of interactions both between component and presentation system and among components. Such an architecture would open the door for interesting experimentations using interactive multimedia.

6.2.2 Alternative Document Formats

The use of SMIL was one of the defining parameters of this project. However, an interesting path for further development would be the addition of support for other document formats as well. MHEG is an interesting alternative among currently available formats. Including support for high-end formats, such as the upcoming MPEG-4 standard [36], in any meaningful way would most likely require substantial redesign of SynCope. Whether such a scenario is at all possible is highly dependent on the direction in which Java and its media APIs are developed.

6.2.3 Adding Voice Capabilities

Alternative ways of rendering multimedia content is an area that could potentially be addressed using SynCope as a platform. Java provides an application programming interface for speech synthesis and recognition – Java Speech API [72]. Within the W3C, a “Voice Browser” working group has been established to address issues related to voice-oriented WWW browsing [39]. Using speech as a two-way media, employing synthesis and recognition, it may in the future be possible to access WWW documents from any telephone. Investigating Java Speech API as a means to adding speech-enabled features to SynCope is a potentially very interesting area for further work. In such an effort, the work of the W3C “Voice Browser” group would serve as a good reference point.

REFERENCES

[1] Blair, G. S.; Coulson, G.; Papathomas, M.; Robin, P.; Stefani, J. B.; Horn, F.; Hazard, L. A Programming Model and System Infrastructure for Real-Time Synchronization in Distributed Multimedia Systems. *IEEE Journal on Selected Areas in Communications*, Jan., 1996. Vol. 14, Issue 1, pp. 249-263. ISSN 0733-8716.

[2] Buford, J. F. K. Architectures and Issues for Distributed Multimedia Systems. *In:* Buford J. F. K. *Multimedia Systems*. New York, New York, USA: ACM Press, 1994. pp. 45-64. ISBN 0-201-53258-1.

[3] Duke, D. J.; Herman, I. A Standard for Multimedia Middleware. *Proceedings of the 6th ACM International Conference on Multimedia '98*. Bristol, UK. pp. 381-390.

[4] Choi, S.; Chung K.; Shin, Y. Distributed Multimedia Presentation System Based on the MHEG. *Proceedings., Twelfth International Conference on Information Networking, 1998*. Jan. 21-23, 1998. pp. 403-406. ISBN 0-8186-7225-0.

[5] Halasz, F.; Schwartz, M. The Dexter Hypertext Reference Model. *Communications of the ACM*, February 1994. Vol. 37, No. 2, pp. 30-39.

[6] Blakowski, G.; Steinmetz, R. A Media Synchronization Survey: Reference Model, Specification and Case Studies. *IEEE Journal on Selected Areas in Communications*, Jan. 1996. Vol. 14, Issue 1, pp. 5-35. ISSN 0733-8716.

[7] Little, T. D. C; Ghafoor, A. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communications*, Apr., 1990. Vol. 8, Issue 3, pp. 413-427. ISSN 0733-8716.

[8] Noll, P. Digital Audio Coding for Visual Communications. *Proceedings of the IEEE*, June 1995. Vol. 83, No. 6, pp. 925-943.

[9] Schäfer, R.; Sikora, T. Digital Video Coding Standards and Their Role in Video Communications. *Proceedings of the IEEE*, June 1995. Vol. 83, No. 6, pp. 907-924.

[10] Liu, C. Multimedia over IP: RSVP, RTP, RTCP, RTSP [online]. [Ohio, USA, 1997.] Last updated: January 15, 1998. Available WWW: <http://www.cis.ohio-state.edu/~jain/cis788-97/ip_multimedia/>.

[11] Jayant, N. High Quality Networking of Audio-Visual Information. *IEEE Communications Magazine*, Sept. 1993. Vol. 31, Issue 9, pp. 84-95. ISSN 0163-6804.

[12] Buford, J. F. K. Uses of Multimedia Information. *In:* Buford J. F. K. *Multimedia Systems*. New York, New York, USA: ACM Press, 1994. pp. 1-25. ISBN 0-201-53258-1.

[13] World Wide Web Consortium (W3C) Synchronized Multimedia Working Group. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification [online]. W3C, June 15, 1998. Available WWW: <http://www.w3.org/TR/REC-smil/>.

[14] Sun Microsystems, Inc. Java Media Players [online]. Silicon Graphics, Inc.; Intel Corporation. Version 1.0.5. Mountain View, CA, USA: Sun Microsystems, Inc., May 11, 1998. Available WWW: <http://java.sun.com/products/java-media/jmf/forDevelopers/playerguide/index.html>.

[15] Sun Microsystems, Inc. Java Media Framework 1.0 Specification [online]. Sun Microsystem, Inc., [1997]. Available WWW: <http: http://java.sun.com/products/java-media/jmf/forDevelopers/ playerapi/packages.html>.

[16] Wahl, T.; Rothermel, K. Representing Time in Multimedia Systems. *Proceedings of the International Conference on Multimedia Computing and Systems, 1994,* May 15-19, 1994. pp. 538-543. ISBN 0-8186-5530-5.

[17] Little, T. D. C. Time-Based Media Representation and Delivery. *In:* Buford J. F. K. *Multimedia Systems*. New York, New York, USA: ACM Press, 1994. pp. 175-200. ISBN 0-201-53258-1.

[18] Orfanidis, S. J. Introduction to Signal Processing. USA: Prentice-Hall, 1996. 798 p. ISBN 0-13-240334-X.

[19] Atkinson, J. The Art of Digital Audio. 2nd ed. Oxford, UK: Focal Press, 1994. 685 p. ISBN 0 240 51320 7.

[20] Rossing, T. D. The Science of Sound. 2nd ed. Reading, Massachusetts, USA: Addison-Wesley Publishing Company, Inc., 1990. 686 p. ISBN 0-201-15727-6.

[21] Roads, C. The Computer Music Tutorial. Cambridge, Massachusetts, USA: The MIT Press, 1995. 1234 p. ISBN 0-252-18158-4.

[22] ITU-T. Recommendation G.723.1 - Speech coders: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s. 1996.

[23] Messick, P. Maximum MIDI: Music Applications in C++. Greenwich, CT, USA. Manning, 1998. 456 p. ISBN 1-884777-44-9.

[24] Kientzle, T. A Programmer's Guide to Sound. Reading, Massachusetts, USA: Addison-Wesley Developers Press, 1997. ISBN 0-201-41972-6.

[25] Brandenburg, K.; Stoll, G. ISO-MPEG-1 Audio: A Generic Standard for Coding and High-Quality Digital Audio. *Journal of the Audio Engineering Society*, October 1994. Vol. 42, No. 10, pp. 780-792.

[26] Robertson, T. D. Linear Sound versus HyperSound: A Fresh Look at Audio on the Internet. *103rd Convention, Audio Engineering Society*, September 26-29, 1997, New York, New York, USA.

[27] ITU-T. Recommendation H.261 - Video codec for audiovisual services at p x 64 kbit/s. 1993.

[28] Luther, A. C. Digital Video and Image Compression. *In:* Buford J. F. K. *Multimedia Systems.* New York, New York, USA: ACM Press, 1994. pp. 143-174. ISBN 0-201-53258-1.

[29] (Editors) Makki, K.; Ni, L. M.; Singhal, M.; Pissinou, N. New Algorithms and Techniques for Well-Synchronized Audio and Video Streams Communications. *Proceedings., Sixth International Conference on Computer Communications and Networks, 1997.* 22-25 Sept. 1997. pp. 214-219. ISBN 0-8186-8186-1.

[30] Goldfarb, C. F. HyTime: a standard for structured hypermedia interchange. *Computer*, Aug. 1991. Vol 24, Issue 8, pp. 81-84. ISSN 0018-9162.

[31] ISO/IEC. ISO 8879:1986(E). Information processing -- Text and Office Systems --Standard Generalized Markup Language (SGML). First edition -- 1986-10-15. [Geneva]: International Organization for Standardization, 1986.

[32] Erfle, R. HyTime as the Multimedia Document Model of Choice. *Proceedings of the International Conference on Multimedia Computing and Systems, 1994.* May 15-19, 1994. pp. 445-454. ISBN 0-8186-5530-5.

[33] Markey, B. D. HyTime and MHEG. *Compcon Spring '92. Thirty-Seventh IEEE Computer Society International Conference, Digest of Papers.* Feb. 24-28, 1992. pp. 25-40. ISBN 0-8186-2655-0.

[34] ISO. ISO/IEC JTC 1/SC29. Coded Representation of Multimedia and Hypermedia Information Objects (MHEG) Part 1, Committee Draft 13522-1. ISO/IEC, June 15, 1993.

[35] World Wide Web Consortium (W3C) XML Working Group. Extensible Markup Language (XML) 1.0. W3C, February 10, 1998. Available WWW: <http://www.w3.org/TR/1998/REC-xml-19980210>.

[36] ISO/IEC JTC1/SC29/WG11 CODING OF MOVING PICTURES AND AUDIO. Overview of the MPEG-4 Standard [online]. ISO/IEC, March 1999, Seoul, South Korea. Available WWW: <http://drogo.cselt.stet.it/mpeg/standards/mpeg-4/mpeg-4.htm>

[37] Schulzrinne, H; Rosenberg, J. Internet Telephony: Architecture and Protocols an IETF Perspective. *Computer Networks and ISDN Systems*, Vol. 31, Feb. 1999, pp. 237-255.

[38] Schulzrinne, H. A Comprehensive Multimedia Control Architecture for the Internet. *Proceedings of the IEEE 7th International Workshop onNetwork and Operating System Support for Digital Audio and Video, 1997.* 19-21 May 1997. pp. 65-76. ISBN 0-7803-3799-9.

[39] World Wide Web Consortium (W3C) ”Voice Browser” Working Group. ”Voice Browser” Working Group Charter [online]. W3C, 18 January, 1999. Available WWW: <http://www.w3.org/Voice/1999/voice-wg-charter.html>.

[40] Meyer, T.; Effelsberg, W; Steinmetz, R. A Taxonomy on Multimedia Synchronization. *Proceedings of the Fourth Workshop on Future Trends of Distributed Computing Systems, 1993.* Sept. 22-24, 1993. pp. 97-103. ISBN 0-8186-4430-3.

[41] Meyer-Boudnik, T.; Effelsberg, W. MHEG Explained. *IEEE Multimedia*, Spring 1995. Vol. 2, Issue 1, pp. 26-38. ISSN 1070-986X.

[42] Hardman, L.; van Ossenbruggen, J.; Mullender, K. S.; Rutledge, L.; Bulterman, D. C. A. Do You Have the Time ? Composition and Linking in Time-based Hypermedia. *Hypertext '99. Proceedings of the 10th ACM Conference on Hypertext and Hypermedia: Returning to Our Diverse Roots.* pp. 189-196.

[43] Steinmetz, R.; Nahrstedt, K. Multimedia: Computing, Communications and Applications. Prentice-Hall, May 1995. 854 p. ISBN 0133244350.

[44] Buchanan, M. C.; Zellweger, P. T. Automatic Temporal Layout Mechanisms. *Proceedings of the Conference on Multimedia '93.* Anaheim, CA, USA. ACM, Aug. 1-6, 1993. pp. 341-350.

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

[46] Comer, D. Internetworking with TCP/IP, volume 1: Principles, Protocols and Architecture. 3rd ed. USA: Prentice-Hall, 1995. 613 p. ISBN 0-13-216987-8.

[47] Rahkila, M.; Huopaniemi, J. Real-Time Internet Audio – Problems and Solutions. 102nd Convention 1997 March 22-25, Munich, Germany. Preprint 4477 (K4).

[48] Schulzrinne, H.; Casner, S.; Frederick, R.; Jacobson, V. Request for Comments (RFC) 1889. RTP: A Transport Protocol for Real-time Applications [online]. Internet Engineering Task Force (IETF) Network Working Group, January 1996. Available WWW: <http://www.ietf.org/rfc/rfc1889.txt>

[49] Braden, R.; Zhang, L.; Berson, S.; Herzog, S.; Jamin, S. Request for Comments (RFC) 2205. Resource ReserVation Protocol (RSVP) – Version 1 Functional Specification [online]. Internet Engineering Task Force (IETF) Network Working Group, September 1997. Available WWW: <http://www.ietf.org/rfc/rfc2205.txt>

[50] Schulzrinne, H.; Rao, A.; Lanphier, R. Request for Comments (RFC) 2326. Real Time Streaming Protocol (RTSP) [online]. Internet Engineering Task Force (IETF) Network Working Group, April 1998. Available WWW: <http://www.ietf.org/rfc/rfc2326.txt>

[51] Handley, M.; Jacobson, V. Request for Comments (RFC) 2327. SDP: Session Description Protocol [online]. Internet Engineering Task Force (IETF) Network Working Group, April 1998. Available WWW: <http://www.ietf.org/rfc/rfc2327.txt>

[52] World Wide Web Consortium (W3C) Working Group on Cascading Style Sheets and Formatting Properties. Cascading Style Sheets, level 2, CSS2 Specification [online]. W3C, May 12, 1998. Available WWW: <http://www.w3.org/TR/REC-CSS2/>.

[53] World Wide Web Consortium (W3C) HyperText Markup Language (HTML) Working Group. HTML 4.0 Specification [online]. W3C, December 18, 1997. Revised on: April 24, 1998. Available WWW: <http://www.w3.org/TR/REC-html40/>.

[54] Steinmetz, R. Synchronization Properties in Multimedia Systems. *IEEE Journal on Selected Areas in Communications,* Apr., 1990. Vol. 8, Issue 3, pp. 401-412. ISSN 0733-8716.

[55] Qazi, N. U.; Woo, M.; Ghafoor, A. A Synchronization and Communication Model for Distributed Multimedia Objects. *Proceedings of the Conference on Multimedia '93.* Anaheim, CA, USA. ACM, Aug. 1-6, 1993. pp. 147-155.

[56] Scach, S. R. Software Engineering with Java. McGraw-Hill, 1997. 618 p. ISBN 0-07-115552-X

[57] Grand, M. Patterns in Java: a catalog of reusable design patterns. Volume 1. New York, NY, USA: John Wiley & Sons, Inc., 1998. 467 p. (Wiley Computer Publishing). ISBN 0-471-25839-3.

[58] World Wide Web Consortium (W3C) Document Object Model Working Group. Document Object Model (DOM) Level 1 Specification. Version 1.0 [online]. W3C, October 1, 1998. Available WWW: <http://www.w3.org/TR/REC-DOM-Level-1/>.

[59] Sun Microsystems, Inc. Java™ 2 SDK, Stnadard Edition [online]. Version 1.2.2. USA: Sun Microsystems, Inc. Available WWW: <http://java.sun.com/products/jdk/1.2/docs/index.html>.

[60] Asbury, S.; Weiner, S. R. Developing Java Enterprise Applications. USA: John Wiley & Sons, 1999. 780 p. ISBN 0-471-32756-5.

[61] Orfali, R.; Harkey, D. Client/Server Programming with Java and CORBA. 2nd ed. USA: John Wiley & Sons, 1998. 1022 p. ISBN 0-471-24578-X.

[62] Eckstein, R.; Loy, M.; Wood, D. Java Swing. USA: O'Reilly & Associates, September 1998. 1221 p. ISBN 156592455X.

[63] Fowler, M. UML Distilled. Scott, K.; Jacobson, I. Addison-Wesley Publishing Company, June 1997. ISBN 0201325632.

[64] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. 1994. Addison-Wesley Longman Inc. ISBN 0-201-63361-2

[65] Borenstein, N; Freed, N. Request for Comments (RFC) 1521. MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. Internet Engineering Task Force (IETF) Network Working Group, September 1993. Available WWW: <http://www.ietf.org/rfc/rfc1521.txt>.

[66] Microstar Software Ltd. SAX 1.0: Road Map [online]. May, 1998. Available WWW: <http://www.microstar.com/sax/roadmap.html>

[67] Krasner, G. E., Pope, S. T. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. Journal of Object-Oriented Programming, Vol. 1 (3), 1980.

[68] Nilsen, K. Adding Real-Time Capabilities to Java. *Communications of the ACM*, June 1998. Vol. 41, Issue 6, pp. 49-56.

[69] Rothermel, K.; Helbig, T. Clock Hierarchies: An Abstraction for Grouping and Controlling of Multimedia Streams. *IEEE Journal on Selected Areas in Communications*, Jan. 1996. Vol. 14, pp. 174-184. ISSN 0733-8716.

[70] Rousseau, F.; Duda, A. Synchronized Multimedia for the WWW [online]. [1997]. Available WWW: <http://delos.imag.fr/sync-mm/>

[71] Sun Microsystems, Inc. JavaBeans API Specification. Version 1.01. Mountain View, CA, USA: Sun Microsystems, Inc., 1997. 114 p. Available WWW: <http://java.sun.com/beans/docs/beans.101.pdf>

[72] Sun Microsystems, Inc. Java™ Speech API Programmer's Guide. Version 1.0. Palo Alto, CA, USA: Sun Microsystems, October 26, 1998. 156 p. Available WWW: <http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide.pdf>.

[73] Steinmetz, R. Human Perception of Jitter and Media Synchronization. *IEEE Journal on Selected Areas in Communication*, January 1996. Vol. 14, Issue 1, pp. 61-72. ISSN 0733-8716.

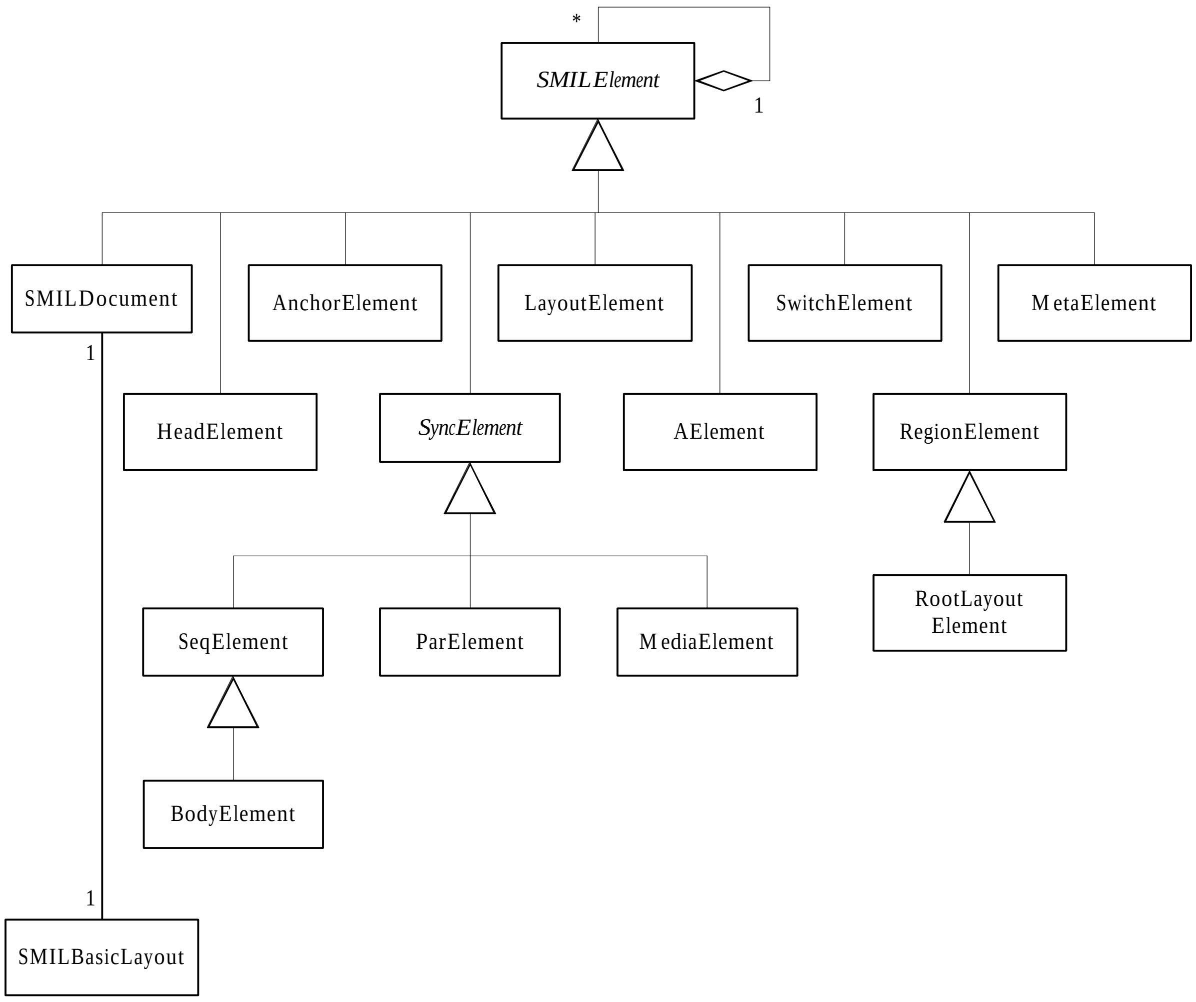
[74] Strawn, J. Digital Audio Representation and Processing. *In:* Buford, J. F. K. *Multimedia Systems.* New York, New York, USA.: ACM Press, 1994. pp. 65-107. ISBN 0-201-53258-1.

[75] Palacharla, S.; Karmouch, A.; Mahmoud, S. A. Design and Implementation of a Real-Time Multimedia Presentation System Using RTP. *Computer Software and Applications Conference, 1997. COMPSAC '97. Proceedings., The Twenty-First Annual International.* 13-15 Aug. 1997. pp. 376-381. ISBN 0-8186-8105-5.

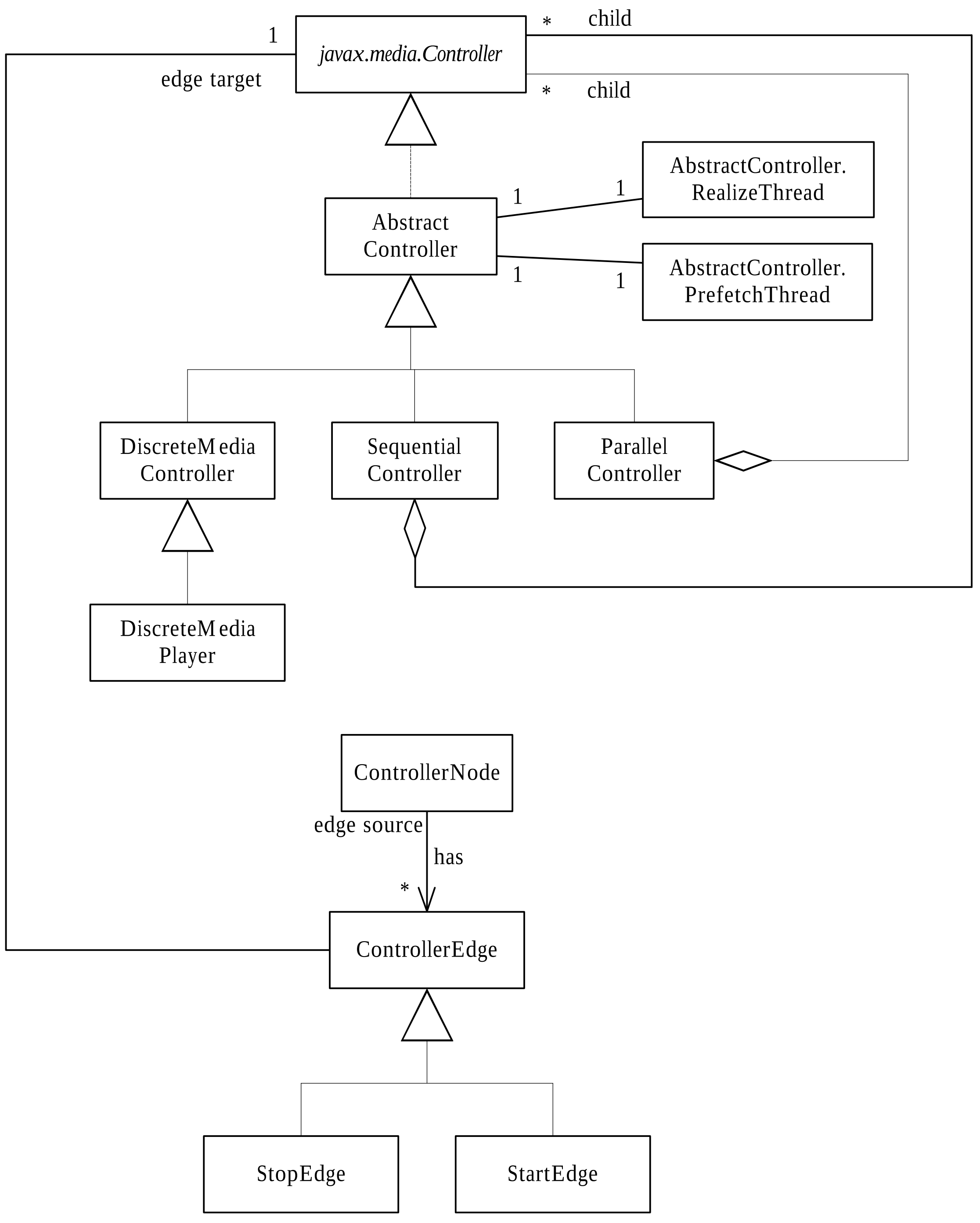
APPENDIX A: Sample SMIL Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil>
    <head id="documenthead">
        <meta name="title" content="SMIL Test document"/>
        <meta name="author" content="Sebastian Nykopp"/>
        <meta name="date" content="20.5.1999"/>

        <layout>
            <root-layout id="main" background-color="#000000" width="800"
               height="600" title="Main region"/>
            <region id="center-logo" background-color="#000000" top="260" left="300"
               width="200" height="72"/>
            <region id="2001_video" top="100" left="240" width="160" height="120"/>
            <region id="left-img" top="100" left="50" width="320" height="240"/>
            <region id="right-img" top="140" left="405" width="320" height="240"/>
            <region id="right-img2" top="180" left="425" width="320" height="240"/>
            <region id="right-img3" top="220" left="410" width="320" height="240"/>
            <region id="right-img4" top="280" left="435" width="320" height="240"/>
            <region id="center" top="150" left="240" width="320" height="240" />
            <region id="bot_left_html" top="350" left="50" width="320" height="170"/>
        </layout>

    </head>
    <body id="body">

        <!-- logo & intro -->

        <par id="intro">
            <audio src="sw/logo.wav" id="logoaudio"/>
            <img src="syncope_logo_black.jpg" begin="5s" dur="9.3s"
               region="center-logo" id="syncopelogo"/>
        </par>

        <!-- slide show with RMF music -->
        <par id="slide">
            <audio dur="40s" src="rmf/rvrboat.rmf" id="riverboat_rmf"/>
            <img src="oz/clam_map.jpg" dur="45s" region="left-img" id="map"/>
            <text src="clam.html" dur="45s" begin="0.2s" region="bot_left_html"
               id="clam_text"/>
            <seq id="imgs">
                <img src="oz/oz1.jpg" begin="10ms" dur="3s" region="right-img"
                      id="slideimg1"/>
                <img src="oz/oz2.jpg" dur="2s" region="right-img2" id="slideimg2"/>
                <img src="oz/oz3.jpg" dur="2s" region="right-img3" id="slideimg3"/>
                <img src="oz/oz4.jpg" dur="2s" region="right-img4" id="slideimg4"/>
                <img src="oz/oz5.jpg" dur="2s" region="right-img3" id="slideimg5"/>
                <img src="oz/oz6.jpg" dur="2s" region="right-img2" id="slideimg6"/>
                <img src="oz/oz7.jpg" dur="2s" region="right-img" id="slideimg7"/>
                <img src="oz/oz8.jpg" dur="2s" region="right-img2" id="slideimg8"/>
                <img src="oz/oz9.jpg" dur="6s" region="right-img3" id="slideimg9"/>
            </seq>
        </par>

        <!-- a QuickTime movie -->
        <video src="2001.mov" dur="30s" region="center" id="mov_video"/>

    </body>
</smil>
```

APPENDIX B: Class Diagram for Static Presentation Object Model



APPENDIX C: Class Diagram for Dynamic Presentation Object Model



javax.media.Controller
1
edge target
*
child
*
child
Abstract Controller
1
1
AbstractController. RealizeThread
1
1
AbstractController. PrefetchThread
DiscreteMedia Controller
Sequential Controller
Parallel Controller
DiscreteMedia Player
ControllerNode
edge source
has
*
ControllerEdge
StopEdge
StartEdge