

Summer 2003

Event Based Retrieval From Digital Libraries Containing Data Streams

Mohamed Hamed Kholief
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/computerscience_etds

 Part of the [Databases and Information Systems Commons](#)

Recommended Citation

Kholief, Mohamed H.. "Event Based Retrieval From Digital Libraries Containing Data Streams" (2003). Doctor of Philosophy (PhD), dissertation, Computer Science, Old Dominion University, https://digitalcommons.odu.edu/computerscience_etds/109

This Dissertation is brought to you for free and open access by the Computer Science at ODU Digital Commons. It has been accepted for inclusion in Computer Science Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

EVENT BASED RETRIEVAL FROM DIGITAL LIBRARIES
CONTAINING DATA STREAMS

by

Mohamed Hamed Kholief
B.Sc. June 1991, Alexandria University, Egypt
M.Sc. February 1998, Alexandria University, Egypt

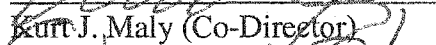
A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

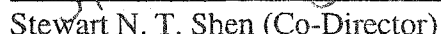
DOCTOR OF PHILOSOPHY


COMPUTER SCIENCE

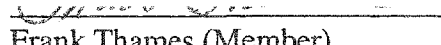
OLD DOMINION UNIVERSITY
August 2003


Approved by:


Kurt J. Maly (Co-Director)


Stewart N. T. Shen (Co-Director)


Hussein Abdel-Wahab (Member)


Frank Thames (Member)


Mohammad Zubair (Member)

ABSTRACT

EVENT BASED RETRIEVAL FROM DIGITAL LIBRARIES CONTAINING DATA STREAMS

Mohamed Hamed Kholief
Old Dominion University, 2003
Co-Directors of Advisory Committee: Dr. K. Maly
Dr. S. Shen

The objective of this research is to study the issues involved in building a digital library that contains data streams and allows event-based retrieval. "Digital Libraries are storehouses of information available through the Internet that provide ways to collect, store, and organize data and make it accessible for search, retrieval, and processing" [29]. Data streams are sources of information for applications such as news-on-demand, weather services, and scientific research, to name a few. A data stream is a sequence of data units produced over a period of time. Examples of data streams are video streams, audio stream, and sensor readings. Saving data streams in digital libraries is advantageous because of the services provided by digital libraries such as archiving, preservation, administration, and access control. Events are noteworthy occurrences that happen during data streams. Events are easier to remember than specific time instances at which they occur; hence using them for retrieval is more commensurate with human behavior and can be more efficient via direct accessing instead of scanning. The focus of this research is not only on storing data streams in a digital library and using event-based retrieval, but also on relating streams and playing them back at the same time, possibly in a

synchronized manner, to facilitate better understanding in research or other working situations.

Our approach for this research starts by considering digital libraries for: stock market, news streams, census bureau statistics, weather, sports games, and the educational environment. For each of these applications, we form categories of possible users and the basic requirements for each of them. As a result, we identify a list of design goals that we take into consideration in developing the architecture of the library. To illustrate and validate our approach we implement a medical digital library containing actual Computed Tomography (CT) scan streams. It also contains sample medical text and audio streams to show the heterogeneity of the library. Streams are displayed in a concise, yet complete, way that makes it unproblematic for users to decide whether or not to playback a stream and to set playback options. The playback interface itself is organized in a way that accommodates synchronous and asynchronous streams and enables users to control the playback of these streams. We study the performance of the specialized search and retrieval processes in comparison to traditional search and retrieval processes. We conclude with a discussion on how to adapt the library to additional stream types in addition to suggesting other future efforts in this area.

ACKNOWLEDGMENTS

First and foremost I wish to thank God without whose help I would have never been able to finish this work. Then my deepest gratitude goes to my advisors, Dr. Maly and Dr. Shen. Their untiring effort, patience, and long hours of guidance on this research deserve my greatest appreciation. I am very grateful for their criticisms and continuous suggestions that have guided me throughout the whole work. I am also very much indebted to their personal kindness, generosity, and consideration that have made this project much easier and helped me through the most difficult times. I wish to thank Dr. Hussein Abdel-Wahab, Dr. Mohammad Zubair, and Dr. Frank Thames, the members of my advising committee, for taking the time of reading this dissertation and for their insightful and helpful comments.

Thanks are due to all the members of the digital library research group. They all contributed to the development of supporting technologies. I would like to single out Hesham Anan for his great help in adapting the ARC engine to my work.

Thanks are also due to Dr. Osama Ebied whose medical knowledge has contributed a great deal to this project. He provided the actual medical data we used and identified various important points that radiologists need to find in this implementation.

I cannot even begin to thank my family for their never flagging encouragement and support through the years. Special thanks go to my wife for her emotional support, encouragement, and love. I am also grateful to Omar, Mariam, and Salma who were born in the midst of this project and who have opened to me a whole new realm of life.

Finally, I can never forget my parents, whose continuous prayers shined my way all through my life.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	vii
 Section	
1 INTRODUCTION	1
1.1 MOTIVATION	2
1.2 OBJECTIVES	5
1.3 APPROACH	8
1.4 DISSERTATION OUTLINE.....	9
 2 RELATED WORK	 11
2.1 DIGITAL LIBRARIES.....	11
2.2 SAVING DATA STREAMS IN DIGITAL LIBRARIES	17
2.3 USING EVENTS FOR RETRIEVAL	25
2.4 DISCUSSION	35
 3 CONCEPTS OF EVENT-BASED RETRIEVAL	 37
3.1 DATA STREAMS AND THEIR RELEVANT CONCEPTS	37
3.2 EVENTS AND THEIR RELEVANT CONCEPTS	40
3.3 A FORMAL MODEL FOR STREAMS AND EVENTS.....	46
3.4 RELATED STREAMS AND THEIR RELEVANT CONCEPTS	48
3.5 ROLES OF DIGITAL LIBRARY USERS AND RELEVANT CONCEPTS	50
 4 POTENTIAL APPLICATIONS AND DESIGN CONSIDERATIONS	 53
4.1 POTENTIAL APPLICATIONS	54
4.2 DESIGN CONSIDERATIONS	65
 5 STREAM BASED DIGITAL LIBRARY: ARCHITECTURE	 72
5.1 DATA STREAMS	73
5.2 EVENTS	80
5.3 RETRIEVAL ARCHITECTURE	89
5.4 GENERAL LIBRARY ARCHITECTURE – A SUMMARY	95
 6 A CASE STUDY: STREAMS IN A MEDICAL DIGITAL LIBRARY.....	 98
6.1 DATA SOURCE.....	99
6.2 DATA ORGANIZATION	104

	Page
6.3 DATA PUBLISHING.....	113
6.4 EVENT GENERATION.....	117
6.5 RELATING STREAMS	119
6.6 THE SEARCH INTERFACE.....	120
6.7 THE STREAM DISPLAY INTERFACE.....	129
6.8 THE PLAYBACK SYSTEM.....	133
6.9 TYPE EXTENSION	144
6.10 SUMMARY	146
7 CONCLUSIONS AND FUTURE WORK	149
7.1 CONCLUSIONS.....	149
7.2 FUTURE EXTENSIONS	152
REFERENCES.....	157
VITA	166

LIST OF FIGURES

Figure	Page
1. The stream-object structure.....	75
2. Stream publishing.....	79
3. Structure of the event object.....	82
4. Structure of the event-class object.	85
5. The event generation process.	87
6. Retrieval architecture.	90
7. The playback architecture.	94
8. The overall architecture of the digital library.....	97
9. Structure of the stream object in this case study.	105
10. Structure of the digital library repository.	110
11. The ER diagram of the database.	111
12. The simple search interface.....	122
13. The group-by menu.	123
14. The sort-by menu.	123
15. The search results interface.....	125
16. The event-based search interface.	126
17. The events menu.....	127
18. Results of an event-based search.....	128
19. Browsing the digital library.....	129
20. The stream display interface.	130
21. Bibliographic information of selected streams appears in a popup window.....	132

Figure	Page
22. Relationship details of selected streams appear in a popup window.	133
23. An example playback interface with two synchronous and one asynchronous stream.	134
24. One CT scan stream player.	135
25. A CT scan stream playing.	136
26. The control panel applet shows that the stream is playing backward (the middle button label is "Change to Forward").	137
27. The control panel applet shows that the player is playing faster than normal (see the slider at the bottom).	137
28. Two synchronous streams: a CT scan streams and a text stream.....	139
29. An audio stream playing.	139
30. An audio stream muted because the stream is playing backward.	140
31. An audio stream muted because the stream is playing faster than normal.	140
32. Two synchronous streams playing: audio and CT scan. A 10-seconds gap exists between the two streams as shown in the status line of the player applets.	143

SECTION 1

INTRODUCTION

Digital libraries have become a very active research area and have applications in several fields such as education, science, arts, and entertainment, to mention a few. Digital Libraries are storehouses of information available through the Internet. "They provide ways to collect, store, and organize data and make it accessible for search, retrieval, and processing" [29]. "A digital library is usually not a single entity. It is distributed and advanced technology is required to link the resources of these many entities. Digital library collections are not limited to document surrogates: they extend to digital artifacts that cannot be represented or distributed in printed formats. A digital library may contain text, images, sound and video in an integrated repository" [18].

Data streams are very important sources of information for many applications. An example is news-on-demand applications that use the video and audio streams produced by the TV broadcast. Another is weather services that use the sensor readings and images coming from different sources such as satellites and radars. A data stream is a sequence of data units produced over a period of time. An example of a data stream is a video stream where the data units are video frames and the time unit is determined by the rate at which these frames are transmitted. Another example of a data stream is an audio stream. A third one is a list of readings recorded from a sensor. Saving data streams in digital libraries is particularly advantageous because of the privileges digital libraries are endowed with such as archival, preservation, administration, and access control.

This dissertation follows the style of *IEEE transactions on networking*.

In this research we not only focus on storing data streams in a digital library but we also introduce a new means of retrieval from this kind of a digital library: *event-based retrieval*. Events are noteworthy occurrences that happen during data streams. They can be automatically deduced or can be specified by an external user. Events are easier to remember than specific time instances at which they occurred; hence using them for retrieval is more user-friendly. Relating streams and playing them back in a synchronized manner is another concept that we emphasize in this research.

1.1 MOTIVATION

The motivation for this project has first been stimulated by a distance education system developed at the Computer science Department at Old Dominion University. This system, Interactive Remote Instruction (IRI) [47], is an interactive multimedia collaborative application in the distance education arena. IRI provides a virtual classroom over a high speed Intranet in which individual computer workstations become each student's window into the virtual classroom. IRI is a multimedia, collaborative, integrated, learning environment. Video, audio, presentation tool, tool sharing are the most important features of this collaboration system. Class sessions are recorded for later playback. Playback was implemented in such a way that users can select a class, choose the date of the class, select what to play (teacher video and/or audio, students video and/or audio, and/or presentation tool), and then press a play button. The user can then move a slide bar to start the session from a random time as needed.

Many features in this system have influenced the topic of this research. The way the recorded sessions are retrieved influenced the evolution of event-based retrieval. The user has to know the exact class date and time; he has to know the streams (audio, video,

presentation) he would like to playback, and he then has to playback the whole session. Although the user can use a slide bar to forward or rewind the session, it is still done on a trial-and-error basis. Students are not likely to remember when exactly a certain issue was discussed during the class. It is believed that users tend to remember by events that occurred during the class session rather than by the time of these events. For example, a student might want to know what the teacher said when another student asked a certain question rather than after 30 minutes from the start of the session. Users can also remember by the content of the lecture, e.g. a title of a transparency, something that has been said, etc. Event-based and content-based retrieval is thus a natural, more human-friendly approach of retrieval.

The second factor in the concept evolution is that the user has to start the IRI system to be able to playback a recorded session. The session consists of a set of streams: audio, video, presentation, etc. Using IRI has the advantage of allowing a group of users to playback a recorded session together. Moreover, IRI itself is the best means of synchronizing these recorded streams, since they have been recorded through this system after all. IRI can also add other features to the playback process, like taking notes, voice chatting, etc. On the other hand, constraining the playback to be only done using IRI is limiting. All of these streams can be played back, one way or another, over the World Wide Web (WWW). Users would certainly prefer to be able to playback a recorded session from the Web, which will be more flexible for the class students since they can review their class from any location that has an Internet access.

The third factor is in the accessibility of the recordings. Only students who attended the class and people who are close to the system can playback a recorded

session since it requires the use of the IRI system. This, again, suggests that using the Web will benefit not only the class students but also the general users who might want to learn about a topic that was covered in this class. Allowing general users to access the class recordings requires some means to search for a specific topic in these recordings. Digital libraries are the ideal way to do the job.

Saving the recorded IRI streams in a digital library add the advantages of using digital libraries for archival, preservation, administration, access control of this information. Digital libraries are accessible, by definition, from the Web. They also support different search techniques. Some libraries support only a simple keyword search; others support a more advanced fielded search that enables Boolean operations between the search fields. More advanced digital libraries allow content-based retrieval from multimedia information (See Section 2). The idea is to save the recorded IRI streams in a digital library and enable event-based retrieval from the library as an effective, user friendly way of retrieval. Other search techniques, such as keyword-search and fielded search based on some bibliographic metadata are also supported.

Having event-based retrieval from a digital library that contains IRI recordings as the initial motivation for our research was then followed by a generalization. A data stream is a sequence of data units that are generated over a period of time. Video and audio are examples of multimedia streams. Another example of a data stream is the readings of a sensor that generates a reading from time to time. A third example is a stream of satellite images. Events occur in any data stream. An example of an application that uses data streams and in which event-based retrieval would be useful is a weather-related digital library. Over any specific area in the United States many weather streams

can be stored in such a digital library. Examples would be: the wind velocity over this area, the temperature, the humidity levels, the percentage of precipitation, the amount of rain or snow, and the satellite and radar maps. Furthermore, there are many streams that are indirectly related to weather that can also be saved in the digital library such as the TV and Radio weather reports. Many events could occur during the span of any of these streams, e.g. a hurricane upgrade or downgrade, a storm on track, a dangerously high temperature, etc. Researchers as well as ordinary users would be interested in correlating many of these streams based on events that occurred during the span of any of them. An example of a query would be “What was the weather report on CNN when a specific hurricane was upgraded from class 1 to class 2?” Other examples are presented in Section 4. Data streams are being collected, preserved and used for research by many agencies including NASA through its “Earth Science Enterprise” and “Digital Earth” projects [30].

1.2 OBJECTIVES

The objective of this research is to study the issues involved in building a digital library that contains data streams and allows event-based retrieval as well as other traditional retrieval approaches. To accomplish this objective, we develop a prototype for such a digital library and apply this prototype for a selected domain. This objective can further be divided into the following major components:

- Building a streams-based digital library.
- Supporting event-based retrieval.
- Developing a prototype for the digital library based on possible applications.
- Developing a digital library application of the proposed prototype.

In more detail, the objective is described in the following few sub-sections.

1.2.1 Building a streams-based digital library

The first component of the objective of this research is saving data streams in digital libraries. There are many digital libraries that hold some kind of a data stream. Video and audio digital libraries have been an important research issue for quite some time. Video-on-demand or audio-on-demand applications can also be considered a special case of a digital library with primitive retrieval mechanisms. One of our objectives is to design a library that can hold any type of a data stream, not just limited to a single type. In other words, the objective is to develop a prototype of a heterogeneous streams-based digital library. There are many issues involved in building such a digital library, e.g.:

- Data organization in the digital library.
- Specifying the required metadata.
- Specifying the services this library will provide to users.
- The characteristics of retrieval approaches from data streams.

1.2.2 Event-based retrieval

The second component of the objective is allowing event-based retrieval. Event-based retrieval is basically a new way for retrieval from digital libraries. Most digital libraries in the literature use bibliographic information as the means for retrieval. Some multimedia-based digital libraries use content-based retrieval approaches. Content-based retrieval depends on using features of video streams to retrieve these streams. Features like color, texture, shape, size, moving objects, and background are used to retrieve shots and scenes from the video stream rather than to play back the entire movie. We argue in this research that event-based retrieval is very suitable for retrieval from data streams

stored in digital libraries. Proving this argument is one of the objectives of this research. There are numerous issues involved in enabling event-based retrieval from the proposed digital library, among them:

- Creating events and inserting them in the different streams.
- The physical representation of events.
- The metadata needed to represent events.
- Using events for retrieval.
- Representing events that depend on many streams.
- Using events to synchronize the playback of multiple streams.

1.2.3 Developing a prototype

The third component of the objective is studying the possible applications of the proposed digital library and the applications' impact on the design. The goal is to develop a prototype of the digital library based on this study. There are several possible applications of the proposed digital library prototype. Issues that need particular attention are:

- Deciding on the important streams for the application.
- Determining what events should be considered.
- Tolerating the digital library design for the application.
- Deciding on what software components need to be included in the prototype.
- Supporting the extensibility in this prototype.

1.2.4 Developing a working application

An important objective of this research is to deliver a working digital library for a particular domain, which will validate our ideas. Our goal is to deliver the following components of a digital library application:

- A repository that contains few hundred data streams of text, images, and audio formats.
- A simple and an advanced search interfaces that employ event-based retrieval and retrieval based on bibliographic information and keywords.
- A player application for each supported stream type, with the corresponding streaming servers.
- A control module to control and synchronize the playback of concurrent streams whenever needed.
- An API that facilitates the extension of the digital library to support any new stream type or format.

1.3 APPROACH

Our approach for this research is as follows. We first consider some possible applications for a digital library that contains data streams and supports event-based retrieval, including a stock market digital library, a digital library containing news streams, a digital library containing census bureau statistics represented as data streams, a weather-related digital library, a sports games digital library, and an educational digital library. For each of the suggested applications, a list of the possible users is formed, e.g. there will be at least a publisher to the library, a manager, and resource discovery users. We then study the basic requirements for each of these users: what tools will be needed,

what queries will be asked, and what sort of search results will be expected. As a result of this study, we create a list of design goals that should be taken into consideration while designing the digital library system. Based on this set of design considerations and on our past experience in the digital library project at Old Dominion University [20], we develop the library architecture and introduce many concepts that are used to iteratively enhance the library design and architecture. The implementation phase of a case study application then follows. The choice of this application is largely based on the stream data availability and we intentionally keep it as simple as possible so that it does not distract us from the basic goal of studying the concepts of event-based retrieval from digital libraries containing streamed data. The research evaluation is the last phase.

1.4 DISSERTATION OUTLINE

The rest of this dissertation is organized as follows:

Section 2 presents a review of the related literature. We examine the literature on digital libraries, multimedia (especially video) digital libraries, retrieval from streams-based digital libraries, event definitions, event detection, and using events for retrieval.

In Section 3, we present the concepts introduced, or focused on, in this research. These are concepts related to events, streams, digital library design, and the services supported by the proposed prototype system.

Section 4 is a discussion of some potential applications of the proposed system and the user expectations from such a digital library. Based on these expectations we present the design considerations of the proposed digital library.

Section 5 introduces the architecture of the proposed digital library system, describing the design entities of this system.

Section 6 revolves around the digital library implementation, focusing on a case study and documenting how this library can be adopted for different applications.

Finally, Section 7 contains a summary of our work and its possible future extensions. It also focuses on the conclusions and the lessons learned during the course of this research.

SECTION 2

RELATED WORK

The title of this dissertation is *Event-based Retrieval from Digital Libraries Containing Data Streams*. This section will survey the literature related to different segments of this title. Using events for retrieval will be explored, not only in the context of data streams or digital libraries, but also in other research areas. We present digital libraries, concentrating on how they are defined in the literature and their relationship to other research areas. We then discuss saving data streams in digital libraries, concentrating on examples of digital libraries that contain video and audio streams, since this is the most common stream type stored in digital libraries. In addition, we focus on how the streams are retrieved in such libraries. The section ends with a brief discussion on the originality of our proposed work as compared to ongoing research.

2.1 DIGITAL LIBRARIES

Digital libraries have benefited from advances in such technologies as multimedia information processing, broadband multimedia computer networks, large capacity computer storage, and high performance personal computers [32]. Not only these technologies, but “inexpensive and fast mass storage devices, coupled with effective media compression algorithms” have made the storage of large volumes of multimedia information in digital form feasible and “advances in network technology have made it possible to access this information remotely. Internet technology and user interfaces provided by client software have enabled the effective presentation and browsing of multimedia and hypermedia documents” [85].

Digital Libraries are defined in [29] as “storehouses of information available through the Internet that provide ways to collect, store, and organize data and make it available for searching, retrieval, and processing.” A digital library is usually not a single entity. It is usually distributed and advanced technology is required to link the resources of these many entities. Digital library collections are not limited to document resources; they extend to digital artifacts that cannot be represented or distributed in printed formats such as sound and video [18].

2.1.1 Definition of the digital library

This section presents various definitions of the term “Digital Library” in the literature. The literature on digital libraries contains a lot of definitions of the term “digital library”, which are sometimes different. As explained in [17] “these definitions range from the digital library as a computer data repository or a collection of digital objects, to much broader definitions that consider the digital library an extension of the traditional library: that is, a library that carries out the traditional library functions of collection, preservation and access provision, while integrating, to an increasing degree, digital media and remotely accessible digital library services.” These definitions continue to change as researchers and users stretch our thinking about them. Gladney et al define the digital library in [38] as:

A DIGITAL LIBRARY is an assemblage of digital computing, storage, and communications machinery together with the content and software needed to reproduce, emulate, and extend the services provided by conventional libraries based on paper and other material means of collecting, cataloging, finding, and disseminating information. A full service digital library must accomplish all essential services of traditional libraries and also exploit the well-known advantages of digital storage, searching, and communication.

Bhargava et al detail in their definition the content types and the various technologies that should be integrated to realize the concept of a digital library [7]. They consider the digital library as:

... a facility which makes use of a large amount of data accessed over a wide area network. A digital library will contain text, images, and sound in an integrated document repository and will provide a uniform interface for users' (local/remote) access to a vast amount of varied kinds of data. The concept of a digital library can be realized by enhancing and integrating technologies in database, communication networks, user interfaces, knowledge management, and information retrievals.

A further broadening of the definition comes in Atkins' report [4]: "It was noted, however, that the concept of a 'digital library' is not merely equivalent to a digitized collection with information management tools. It is rather an environment to bring together collections, services, and people in support of the full life cycle of creation, dissemination, use, and preservation of data, information, and knowledge."

Elements identified as common to these definitions and others are:

- The digital library stores large amount of organized materials in electronic format.
- These materials in electronic format will be accessed over a wide area network, mostly the Internet. It is a goal to have universal access to digital libraries and information services.
- The digital library is not necessarily a single physical entity. It is usually distributed and technology is required to link the resources of these distributed entities. These linkages are transparent to the end users.
- Digital library collections are not limited to document resources: they extend to digital artifacts such as audio and video that cannot be represented or distributed in

printed formats. A digital library may contain text, images, sound and video in a distributed but integrated electronic repository.

- The digital library provides a uniform interface for users to access to a vast amount of varied kinds of data will be provided.
- The digital library is an environment to bring together collections, services, and people in support of the full life cycle of creation, dissemination, use, and preservation of data, information, and knowledge.
- A full service digital library must accomplish all essential services of traditional libraries and also exploit the well-known advantages of digital storage, search, and communication [18].

2.1.2 Digital libraries and its relationship with other research areas

The digital library, as a research area, brings together contributions from many existing research areas. As put by Nuernberg et al: "From a database or information retrieval perspective, digital libraries may be seen as a form of federated databases. From a hypertext perspective, the field of digital libraries seems like a particular application of hypertext technology. From a wide-area information service perspective, digital libraries could appear to be one use of the World Wide Web. From a library science perspective, digital libraries might be seen as continuing a trend toward library automation" [75].

Digital libraries as distributed databases

Many database features seem to apply to digital libraries. A digital library contains large amounts of data and provides access to this data, which needs to be updated and consistent, for a diverse group of distributed users. User interfaces will be

provided to browse, find and locate information in the digital library, user interfaces are necessary. Moreover, indexes need to be built and maintained. Data transmission and compression techniques must be used. "All of these topics, in one guise or another, and to varying degrees, are present in database system research. Many research proposals in this context are based on extending database ideas or techniques and adding a flavor of database system research" [1].

However, database technology has certain weaknesses as a base technology for implementing digital library systems. Adam et al elaborate on this point, saying that "For example, database technology uses the principle of non-interpretation of data as a basic axiom." The database system does not interpret the data that resides in it. In digital library systems, a very large number of different objects are expected to be stored. Users will have very little idea of what the digital library contains. So, users will base retrieval requests on the content of the objects. This querying by content is entirely unsupported in database systems.

Another aspect of the database technology is the role played by the "schema." The database is populated with data after the schema has been designed. "Digital libraries will collect data objects incrementally, so the schema in the digital library, if it exists, will be changing over time to accommodate for the new type of objects" [1].

Digital libraries as networked information systems

"Networked information systems", such as the World Wide Web (WWW) on the Internet, is the second main information systems technology. Documents containing multimedia data are provided to the distributed users through "a network of interconnected servers" [63]. The information servers around the network have to be

organized, searched, filtered, and navigated. All kinds of new information are rapidly being digitized and made available to the users of the WWW. This area sees explosive growth both in servers and users. Lynch, one of the leading library Information Technology experts, then raises the question of why the current WWW tools are not used instead of digital libraries. Some network specialists have claimed that the Internet is the library of the digital age. As Lynch points out, it is not entirely uncommon to think of WWW-indexes as sufficient replacements to library catalogues, and "that libraries may in the long run fade into oblivion" [63]. This kind of views is elaborately criticized by him:

One sometimes hears the Internet characterized as the world's library for the digital age. This description does not stand up under even casual examination. The Internet--and particularly its collection of multimedia resources known as the World Wide Web--was not designed to support the organized publication and retrieval of information, as libraries are. It has evolved into what might be thought of as a chaotic repository for the collective output of the world's digital "printing presses." This storehouse of information not only contains books and papers but also raw scientific data, menus, meeting minutes, advertisements, video and audio recordings, and transcripts of interactive conversations. The ephemeral mixes everywhere with works of lasting importance. [63]

The administration, documentation, security and other features that are taken seriously in any database management system are not taking any priority in the WWW. Adam, talking of the problems that are posed for the avid digital library provider and/or user because of the chaotic nature of the Internet, says that "the traditional information content is possible because of a balance between the rights of three groups of people: the authors, the publishers and the readers. The roles of these three groups and their rights in the Internet world are entirely unclear and will need to be sorted out very quickly before anything can be accomplished" [1].

Digital libraries are also distinct from simple WWW publishing because they exhibit archival properties that are not guaranteed in the WWW: [72]

- An entry in the digital library is persistent over time.
- Canonical entries are usually supported.
- There is access management in digital libraries.
- There is also a responsible point of contact.

2.2 SAVING DATA STREAMS IN DIGITAL LIBRARIES

Saving data streams in digital libraries has long been limited to saving audio and video streams. There are numerous multimedia digital libraries that have been developed in several research projects for this particular purpose. Some of these projects just preserve audio and video files and allow the retrieval and playback of these files as a whole based on some bibliographic metadata, e.g. title of the song or movie, name of a performer or an actor, etc. Examples of these projects are academic libraries that maintain video and audio collections, e.g. the EGARC digital audio library [25]. These libraries can also be identified as video-on-demand or audio-on-demand systems. Unlike such systems, some digital video libraries integrate image and video processing and understanding, speech recognition, distributed data systems, networks, and human/computer interactions in a comprehensive system. A key component of this difference is the use of content-based indexing and retrieval algorithms to enable users to interact with the video library rather than to simply play back entire movies or broadcasts. As a consequence, there has been considerable activity developing improved tools for video processing and content analysis. Two examples of these projects would be

the Informedia project at Carnegie Melon University and the VISION project at the University of Kansas [35].

The Image Information System Group at IBM is working on a project, SPIRE, which has a research agenda that involves producing progressive data representation for unstructured data (such as time series, image, video, and multidimensional data) using, for example, wavelet transformation techniques. Their major project is the Progressive Content-Based Image Retrieval in which the researchers propose a progressive framework that reorganizes the image and video data into a pyramid. Accordingly, search operations can be performed in a hierarchical fashion and thus significantly reduce the total amount of data that need to be processed [88].

2.2.1 Examples of digital libraries containing data streams

Digital video libraries

Digital video libraries are different from "video-on-demand" services or other similar projects in that "they incorporate diverse technologies in a wide-ranging system." Image and video processing and understanding, speech recognition, distributed data systems, networks, and human-computer interactions are some of these technologies. The most significant distinction is "the use of content-based retrieval algorithms to allow users to search the digital library rather than simply playing back entire movies" [36]. Consequently, there has been considerable activity developing improved tools for video processing and content analysis. There has also been important progress made developing real-time multimedia systems capable of displaying video to users on different platforms and over various types of networks [36].

Several approaches have been presented to decompose raw video into shots and scenes. A shot is a continuous roll of a camera and scenes are collections of shots which occur in a single location or are temporally unified. The problem of identifying transitions between shots (cuts) has mostly had a bottom up approach. Model-based algorithms have been developed to successfully detect fades, dissolves, and page translates edits [43]. Looking for quick changes in color histogram or image intensity is another approach [3], [70], [104].

Once shots have been identified, key frames that characterize the shot can be selected by considering the motion of objects within the shot. Here, frames that are as "still" as possible are selected [99] or the background and moving objects are explicitly identified and an image that concentrates on one or the other is selected [83]. Another related approach is to combine information from multiple frames of an image sequence to create a "salient video still" which characterizes the shot in some way [91]. These methods vary significantly in their computational complexity and effectiveness for different video sources, but each has its merits.

Although the problem of shot detection is basically solved, the problem of combining shots to obtain scenes is still very challenging. One approach used by the Princeton Deployable Video Library (PDVL) is to identify key frames in each shot and use "image-based clustering to construct a scene transition graph to visually present the relationships among shots" [100]. Users can locate scenes of interest (e.g. two person interviews) by going through a collection of graphs. The scene transition graph can be used to navigate through the video. The Algebraic Video System uses an alternative technique where shots are organized in a "hierarchical structure that allows nested

stratification" [96] (sub trees may refer to overlapping portions of the raw video). This system uses the VuSystem [56] for recording and processing video but hierarchy construction is performed manually. A model-based approach has been proposed to "parse video by an a-priori model of the video structure" [105]. Such a model is advantageous in that it reflects a strong "spatial" order within the individual frames of shots and/or strong "temporal" order across a sequence of shots. For example, as Zhang et al point out "it is necessary that all shots of the news anchorperson conform to a spatial layout" [105]. For many tasks it is highly complicated to define models for the video. In Zhang's system, on the other hand, an operator enters the text description of the video contents. This approach may be accurate but it makes production of large video collections very expensive. Automatically identifying the content of a video segment is a particularly challenging problem. Three basic approaches have been investigated for this purpose: image understanding, speech recognition, and caption processing.

Research in computer vision over the past 20 years has had success in only limited domains [44]. For this reason, many approaches for image-based content identification have focused on feature-based schemes for classification. For example, images can be indexed using "color histograms" [90] or "combinations of shape and color features" [87]. The QBIC (Query by Image Content) project [27] explored methods to query large on-line image databases using the image contents, such as color, texture, shape, and size. Although feature-based classification is quite fast, one drawback is that very different objects may have the same characteristics (e.g. a red car and a red apple).

Moving away from pure feature-based matching, it is proven that "similarity-based image retrieval" can be also accomplished using Hidden Markov Models (HMM)

that have been trained with representative images of outdoor scenes (rivers, trees, mountains) [103]. Jacobs, however, uses "multi-resolution wavelet decompositions" for rapid image matching and retrieval. Here, a "low resolution" example (either hand drawn or scanned) is used as a query and "multi-scale matching" is used to locate the most similar image in the database [48]. More ambitious indexing that is based on "texture, shape, and appearance" has been investigated within the Photo book system [82], [78]. Although this system has had excellent success within a limited domain of images (textures and faces), the computational expense associated with computing Eigen images may restrict its use for identifying video content.

Given the difficulty of image-based content analysis, processing the audio track and closed caption information is an interesting alternative. The goal of the Informedia Digital Video Library project is to establish a large, on-line library featuring full content and knowledge-based search and retrieval of digital video, primarily for educational purposes [15], [16], [93]. Informedia's News on Demand system specifically addresses the problem of "providing efficient access to news videos, which requires complete automatic segmentation and indexing" [45]. To automatically transcribe narratives and dialogues into text files, Informedia uses the SPHINX-II speech recognition system, which is "a large-vocabulary, speaker-independent, continuous speech recognizer" [45] developed at CMU. Closed caption transcripts are also recorded whenever it is available. Speech recognition allows the developers of the "News-on-demand system" to align the closed captions with the video and to provide words for indexing where closed captions are unavailable. To process queries, Informedia employs natural language processing technologies to extract users' subject or content of interest without forcing them to

specialized syntax or complicated command forms. Users can input their queries into Informedia by either keyboards or microphones. The extensive use of AI techniques, particularly natural language processing and speech recognition in the presence of noise and background music, makes this a promising but high-risk project.

The goal of the VISION (Video Indexing for Searching over Networks) project is to demonstrate the technology necessary for a comprehensive, on-line digital video library. VISION shares many of the goals of the Informedia project. Nevertheless, much more restricted processing of the audio track is implemented to perform automated scene segmentation and content analysis. This project further constrains the problem by requiring "a system that is capable of making all news stories available within seconds of their broadcast" [36]. Audio information is particularly used to combine shots and for limited word spotting. Closed captions are recorded in the library when available and used in a full-text retrieval engine to search the video library for material of interest [34], [35], [36].

Digital music libraries

The future for digital music libraries seems bright. There is already enormous popular interest in music on the Web. Low-cost compact portable music players capable of playing high-quality music files in the MP3 format downloaded from the Internet are revolutionizing the music publishing industry. Lycos, a large search engine, has announced a search engine for MP3 files [61], and has indexed 500,000 of them. The existence of such an index is proves the significance of digital music libraries. Even more interesting is the motivation for providing the index: search engine queries for "MP3" very popular. Music is the one of the most important examples of a popular digital

library. The inclusion of digitized recordings would certainly be a tremendous asset to a digital music library. Yet, in a such a volatile and currently hotly litigated field as this, there are various important, and difficult, issues of intellectual property to be considered and resolved, particularly when dealing with commercial recordings by well-known artists. However, as Papadakis and Douligeris point out, "it is highly likely that the creators of MIDI files would readily give permission for these files to be included in a larger collection, provided that due credit was given, and that music publishers would find it in their interests to allow wider access to the works they produce, providing access to the music images was suitably controlled and the opportunity for buying paper copies directly from publishers was also available" [77].

In [6], two prototype digital music libraries are described: one small, high quality collection and this is created using optical music recognition and manual text entry; and one large, low quality collection derived from a web crawler for MIDI files. Each collection has strengths that the authors would like to reproduce in a future production-quality digital music library: on the one hand quality metadata, searching by "sung input and output in multiple formats," and on the other hand, "breadth of content" [6]. Each has stretched our current technology and demonstrated the need for further research in acquisition, searching and browsing, presentation and evaluation.

VARIATIONS is a digital music library system that provides online access to over 5,000 sound recordings from Indiana University's library collection by relying on a client-server architecture [24]. In VARIATIONS, information retrieval is facilitated through the employment of metadata. Dunn, commenting on this project, says that "each sound object is accompanied with an enhanced USMARC bibliographic record, which is

stored in an ASCII text file” [24]. The “platform-specific nature” of the corresponding architecture results in a library with limited access capability.

JUKEBOX is “an application that provides online access to a distributed collection (according to the 3-tier architectural model) of sound recordings” [31]. Although this system relies on the MP3 format for the encoding of its sound archives, access to the collection is once again limited due to the fact that decoding at the client side requires a specific hardware decoder.

The PATRON client-server architecture has been designed to provide audio (and video) on demand at the University of Surrey. Explaining the nature of this architecture, Lyon says that “it is based on a metadata repository that is structured according to the XML specifications” [62]. The system is widely available since the user interface works within a Web browser. An innovative approach to sound collections is followed at the New Zealand’s digital library of music and audio data [66]. According to this approach, no bibliographic metadata is required for the retrieval of requested files. Instead, a “melody index” system retrieves music on the basis of a few notes that are sung, hummed or otherwise entered. Related content-based music retrieval techniques are employed for querying and browsing music collections of the SMILE [67] system.

More recent systems for music delivery on the Web like Napster [71] and Gnutella [39] rely on peer-to-peer architectures for organizing their MP3 file collections and facilitating information retrieval for their users. Specifically, peer-to-peer is a type of file-sharing mechanism that allows users under the same network (i.e. Internet) to access the contents of a specified area on each other's hard drives. Such systems are widely accepted from the Web community since they depend on Web technologies and therefore

require no specific operating systems. However, the future of such systems is uncertain, since they are accused of violating copyright laws by the music industry. Consequently, the legal issues that arise when delivering music on the Web seem to be the most crucial problems in the development of effective, Web-based digital music libraries [77].

2.3 USING EVENTS FOR RETRIEVAL

In this dissertation, one major issue is the use of events for retrieval from a digital library containing data streams. The focus is on the technical issues related to the digital library design and implementation. This section describes how certain research relates to events. We show how events are defined in the computing literature in Section 2.3.1, we present work done in the area of event detection in Section 2.3.2, and we also explain that events have been used for retrieval in some applications in Section 2.3.3. The information in these sections sometimes overlaps because for systems, such as ours, we need to show the definition of the event before talking about how they are detected; or we need to describe how the events are detected before showing how they are used for retrieval.

2.3.1 *Event definitions*

Events are used in such research areas as simulation, active databases, object-oriented programming, and topic detection in news streams. Thus, the definition of an event varies depending on the context. We will give examples of different definitions below.

Microsoft defines an event in [68] as “an action recognized by an object, such as a mouse click or key press, for which you can define a response. An event can be caused by a user action or a Visual Basic statement, or it can be triggered by the system.”

In the Complex Event Detection and Monitoring System (CEDMOS) developed for DARPA by MCC, the event is defined as “an occurrence at a single point in time, either in the physical world or the virtual world inside a computer” [5]. An event is usually associated with some change of state. CEDMOS also defines a *Primitive* event as “a real world event occurring outside the event processing system.” These primitive events are the domain dependent entities that the event processing system is dealing with on the virtual level. An *event class* or *type* is described as “a syntactic definition of a collection of information associated with the occurrence of a class of events” [5]. The event type is often given an explicit name for convenience. The structure and types of information for all events in a class are the same, but the actual instance values can differ. In the same reference, an *event stream* is defined as “a contiguous sequence of events of a specific type.” An event channel is defined as “a conduit for events conforming to a single event type from an event producer to an event consumer” [5]. Event channels and event streams are also described as convenient abstractions for describing what happens to events both inside and outside the event processing system.

Active database systems (aDBS) are able to recognize specific situations (in the database and beyond) and to react to them without immediate, explicit user or application requests [33]. In the context of active databases, most of the operations that are of interest to the application take a finite amount of time for their execution as mentioned in [13]. As an example, a database operation, function, or a transaction can be viewed as an event expression. Chakravarty et al view an absolute point on the time line (corresponding to an absolute time) as “a degenerate event expression where a point is a special case of an interval.” The interest in event expressions comes from the fact that one needs to choose

a point on the time line, within the closed interval defined by an event expression, to designate the occurrence of that event. In other words, an event expression needs to be associated with a point that can be declared as an event corresponding to that event expression. An event in this context is defined as “an atomic (happens completely or not at all) occurrence” [13]. The authors distinguish between “logical events” (e.g., end-of-insert operation) and “physical/internal events” (e.g., the point in code after the last operation but before the return from insert). Both events conform to the above definition; however, logical events correspond to the specification of an event “at the conceptual level” and do not specify “the physical or internal event to which it will be mapped.” In database applications, the interest in events mostly comes from the state changes that are produced by database operations. In other words, state changes are associated with operation execution and hence event occurrences corresponding to these operations are of significance. Using parameters connected with the operations effects state changes. Hence, events are associated with important operations (i.e., event expressions) and their parameters are viewed as the parameters of the event corresponding to the operation; these operations take a limited amount of time. Or, as Chakravarty et al point out, “a seemingly instant event may be viewed as a long term one if the time granularity is sufficiently magnified.” A logical event is defined to be “conceptually atomic” (e.g., abort transaction t_1 ; insert tuple t_1 into relation R). In contrast, physical events can be viewed as occurrences that are fast compared to the granularity of the time scale chosen. The authors associate time of occurrence t_{occ} (a point on the time line) with every event instance. The time of occurrence associated with a

logical event is the same as the time of occurrence associated with the physical event corresponding to it, which is detected by the system.

The same concepts are emphasized in [33]. An event in this paper is always regarded as "a specific point in time" [21]. Because only some points in time for which a reaction is required are of interest, the authors have to specify them in some way, e.g., as an explicit time definition (at 18:00) or as the beginning or the end of a database operation. They also consider more complex ways to define events, e.g., as the point in time when the last of a set of events has occurred (complex events). Thus, an active Database System has to provide various constructs for the specification of events of interest. Most of the existing systems at the time that paper was published were restricted to "simple" event definitions, except for work in ODE [37] and Sentinel [12] that dealt with advanced event specification.

Topic Detection and Tracking systems draws a distinction between events and topics in the conventional sense. "An event identifies something (non-trivial) happening in a certain place at a certain time" [102]. For example, USAir-427 crash is an *event* but not a *topic*, and "airplane accidents" is a *topic* but not an *event*. One could think of events as instances of topics, associated with certain actions.

Our own definition of an "event" is presented in Section 3. More event definitions are presented in Section 2.3.2 and Section 2.3.3 about detecting events and using them for retrieval.

2.3.2 Event detection

A large variety of event detection and notification systems have been implemented; examples include: SIFT [101], Salamander [64], Siena [10], OpenCQ [58],

Elvin [84], Gryphon [89], NiagaraCQ [14], LeSubscribe [79], Hermes [26], and A-TOPSS [57]. A description of some of the event detection approaches is presented here.

[2] is a final report about Topic Detection and Tracking pilot study. Topic Detection and Tracking (TDT) is a DARPA-sponsored initiative to investigate the state of the art in finding and following new events in a stream of broadcast news stories. During the first portion of this study, the notion of a “topic” is modified and sharpened to be an “event”, meaning “some unique thing that occurs at some point in time.” The TDT study assumes multiple sources of information such as “various newswires and various broadcast news programs.” The information flowing from each source is supposed to be divided into a sequence of stories, which may provide information on one or more events. The general task is to identify the events discussed in these stories, in terms of the stories that talk about them. Stories that discuss unexpected events would follow the event, whereas stories on expected events could both precede and follow the event. The detection task is characterized by the lack of knowledge of the event to be detected. In such a case, Allen et al claim that “one may wish to retrospectively process a corpus of stories to identify the events discussed therein, or one may wish to identify new events as they occur, based on an on-line stream of stories.” Both of these alternatives are supported under the detection task. The retrospective detection task is defined as “the task of identifying all of the events in a corpus of stories” [2]. Their association with stories defines events. Therefore, the main task is to group the stories in the study corpus into clusters, where each cluster represents an event and where the stories in the cluster discuss the event. Each story supposedly discusses at most one event. Hence, each story is to be included in at most one cluster. The on-line new event detection task is defined as

“the task of identifying new events in a stream of stories” [2]. Each story is processed in sequence, and a decision is made whether or not a new event is discussed in the story, after processing the story but before processing any subsequent stories.; that is, a decision is made after each story is processed. The first story to discuss an event should be flagged YES. If the story does not discuss any new events, then it should be flagged NO. Further details are presented in [2].

In the same context, Yang et al describe event detection in a sequence of segmented chronologically ordered news stories. For the authors, event detection is “an unsupervised learning task, sub-divided into two forms” [102]. One is “retrospective detection,” which entails the discovery of previously unidentified events in “a chronologically ordered accumulation of documents (stories).” The other is “on-line detection,” where the goal is to identify “the onset of new events from live news feeds in real time” [102]. Talking of these forms Yang et al say that “both forms of detection intentionally lack advance knowledge of novel events, but may have access to unlabelled historical news stories for use as contrast sets.” Given that each event is usually discussed by multiple news stories, document clustering appears to be a natural approach to event discovery. The authors implement two clustering methods: a divide-and-conquer version of a Group-Average Clustering (GAC) algorithm [98], and a single-pass incremental clustering algorithm (INCR). GAC is for “agglomerative clustering, resulting in hierarchically organized document clusters. It is designed for batch processing, and has been used for retrospective detection.” INCR is designed for “sequential processing, and has been used for both retrospective detection and on-line detection. INCR results in a

non-hierarchical partition of the input collection” [102]. Technical details of both algorithms are explained in [102].

The ultimate goal of the OpenCQ research presented in [58] is to “develop a toolkit for Internet scale update monitoring with event driven evaluation and delivery.” The framework is organized around five abstract models, each of which focuses on a different dimension of concern in the design. One of these models is the event model, which characterizes distributed events and a time model. Events in this research may be defined in the time dimension (e.g., every 10 minutes or 10am everyday) or in the information space (e.g., currents at a station in the Columbia River estuary exceed 3m/s or IBM stock price drop by 10%). Events are either “primitive” or “composite.” Primitive events are further characterized as being “predefined” or “user defined.” Commenting on predefined events, Liu et al say that “the server can automatically detect predefined events.” In particular, the server polls the system environment for their occurrences. User defined primitive events must “be registered in the server by a user or a program such that the server can have information about the semantics of user defined events in order to detect their occurrences automatically” [58]. The time model revolves around the temporal and causal relationships between events and notifications and addresses “the problems of synchronizing clocks across distributed systems as well as associating times with events in distributed systems” [58]. According to this event model, an event can be uniquely characterized by the identity of the object of interest involved in the event, the identity of the operation, the identity of the invoker, and the time of occurrence of the event. An event is observable, Liu et al explain, “if some object other than the object of

interest and the invoker can detect the occurrence of the event; and it is up to the object of interest to determine which of its events can be observable.”

Haering et al propose in [41] a multi-level video event detection methodology and applies it to animal hunt detection in wildlife documentaries. The proposed multi-level approach has three levels. The first level extracts color, texture, and motion features, and detects moving object blobs. The mid-level employs a neural network to verify whether the moving object blobs belong to animals. Talking of this level, Haering et al say that it “also generates shot descriptors that combine features from the first level and contain results of mid-level, domain specific inferences made on the basis of shot features.” The shot descriptors are then used by the domain specific inference process at the third level to detect the video segments that contain hunts. The proposed approach can be applied to different domains by adapting the mid and high-level inference processes. Event based video indexing; summarizing, and browsing are among the applications of their proposed approach.

2.3.3 Retrieval using events

Several research papers such as [60], [92], and [94] claim that there is psychological evidence that we remember the temporal relationships between events. Hence, though we may not remember when we read a piece of mail, we may recall, for example, that it was on a Friday afternoon just before we left for the weekend. There is also evidence that seeing the context of a past event helps us to evoke the event itself. This means that a personal document retrieval system that is designed to work in conjunction with our own memories will be more effective if it incorporates “events and their relationships” [8]. In [8], this psychological evidence is used to justify using

personal events for retrieval. Events in that paper are defined as “things that happened in the past and are the information that would be recorded if we kept a careful personal diary.” A complete event record would include details of where we were, whom we were with and what we were doing at a time in the past. Bovey argues that people handle quite large amounts of information in the course of their working lives, and these collections of personal information can easily become unmanageable, due to their amount, without some kind of retrieval aid. A “prototype graphical event based retrieval system; an event browser and its use for personal retrieval” is described in that paper [8]. Similar projects aimed at using personal event histories for retrieval are the Memoires project [54], the Pepys project [73], and the Forget-Me-Not project [53], [74].

In [80] and [81], Pikovic et al introduce the COBRA video data model that provides a framework for using different knowledge-based methods for interpreting raw data into semantic content. They have extended the model with the object and event grammars, which aim at “formalizing descriptions of high-level concepts, as well as facilitating their extraction based on features and spatiotemporal logic” [81]. The authors also explain that the model “supports stochastic modeling of events.” They increase the number and the percentage of accurately recognizable strokes in comparison to the methods mentioned in literature. Furthermore, experimental results with regular classification of tennis strokes demonstrate that their approach is promising to realize statistics of tennis games automatically using normal TV broadcast videos. Eventually, they show the advantage of the integration of the spatio-temporal and the stochastic approach.

In [22], a system architecture for supporting event based interaction and information access is presented. The University of Leeds Virtual Science Park[®] (VSP) is an example of a virtual working system for supporting the provision of a number of services, including virtual consultancy, work based learning, consortium building and access to information. Drew et al present an architecture based on event mechanisms. According to the authors, “the extensions fall into 3 areas: provision of timely, up-to-date information based on users’ interests; integration with existing information sources; and chance interaction.” The main problems addressed in that paper are facilities for keeping the VSP information up to date and keeping users notified of relevant changes to the data. Integration with existing data sources is also important to the provision of an information space capable of supporting VSP services. The authors describe an event mechanism to “support information location, information monitoring, chance interaction and integration with existing information sources” [22]. The system is to be used in a VSP Work based learning project for supplying multimedia information and collaboration facilities to remote students to support distance learning. A World Wide Web gateway is developed to provide wide scale access to the VSP. This paper in particular concentrates on automatic monitoring of VSP information. A major user requirement is to be able to keep up to date with changes to previously discovered information. Without automatic monitoring, Drew explains, “information sources must periodically be polled by the user to determine if there has been a change in the state of the information (just as modification or deletion).” This paper also describes an “agent-based mechanism” for automating these queries and reporting the changes. A major problem with distributed virtual environments is that the authors do not give the user an awareness of the interests

or presence of other users. Such awareness is essential if the environment is to allow people to make informal contact in the same way that they would in a common area such as a coffee bar. The paper proposes an awareness mechanism based on a 'yellow pages' directory service, a "real time location broker," [22] and a monitoring agent that initiates communication between users with similar interests. The VSP directory needs to reference or incorporate existing information stored online. The authors describe a mechanism by which information sources such as the World Wide Web can be integrated into the VSP object store in order to allow seamless access and searching using the VSP tools.

2.4 DISCUSSION

The previous sections show details about the current state of the art of the research in the areas of event detection and usage for retrieval; digital libraries; and saving and retrieving data streams in digital libraries; research areas that we perceive as closely related to our work. We conclude the following from this survey:

- As straightforward as it may seem, the definition of events varies depending on the context: for example, news events are different from events in an active database system. Events may be defined as being based on specific points in time or on periods of time.
- Serious work has been made in the area of event detection, which makes it feasible to assume that one can safely use events for retrieval, knowing that there are algorithms that can be used to generate these events; or there is at least serious research in this direction.

- 'Digital libraries' is such an important research area and saving data streams in digital libraries is both valuable and feasible; there are significant current research projects that use digital libraries to save video or music streams.
- Saving data streams of types other than video or audio is not supported well in digital library research, albeit the importance of using various types of data streams as sources of information is recognized.
- Events, as far as we could find in the current literature, have not been used as a search mechanism in retrieval from a digital library containing data streams, which makes this area a ripe area for exploration and contribution.

SECTION 3

CONCEPTS OF EVENT-BASED RETRIEVAL

This section presents the different concepts that relate to the main topic of this research: event-based retrieval. It presents our definitions of the terminology used throughout this dissertation. The need for many of the concepts introduced here will become clearer in the rest of the dissertation. These concepts are also presented in [50].

3.1 DATA STREAMS AND THEIR RELEVANT CONCEPTS

3.1.1 Definition of a data stream

A data stream is a sequence of data units produced from a specific source over a period of time. Examples of data streams are video streams, audio streams, sensor readings, satellite images, and recorded monitoring information. The main criterion in determining if a list of data units constitutes a stream is the existence of the time factor. A data unit of a stream might be a video frame, an audio packet, or a sensor reading. Stream sources can be video cameras, microphones, TV broadcast, satellite sensors, radars, and messages between client applications and servers. The regularity of the stream is not an important aspect of our research, i.e. it does not matter in this research if the data units are produced periodically or not as long as the software that accesses this stream understands its nature and characteristics.

3.1.2 Stream types

A set of streams that share the same characteristics are said to be of the same stream type. Examples of a stream type would be a teacher audio, a weather sensor's

readings, or a TV news report. Characteristics that will be shared among streams of the same type are the metadata fields associated with these streams, the fields on the search form, the data access methods to be used to specify the events for this stream, and the players to be used to play this stream. A stream type is distinguished from a stream format. Many stream types can have the same format, e.g. audio format, but they are different in the way they are used depending on the application. The metadata set for a teacher audio stream would definitely be different from the metadata set for the audio stream of a TV news report. As can be seen in Section 6, technology will be used to accommodate this heterogeneity in our digital library application.

3.1.3 Stream bibliographic metadata

Different bibliographic metadata fields need to be stored with every stream type. In our application, bibliographic metadata are specifically used for resource discovery. For instance, a user may want to search the digital library for all the audio streams in which the “teacher” is Mr. Smith. The bibliographic field that is used in this search interface is “teacher” which signifies the dependence of the bibliographic metadata on the stream type, as seen in the last section. A “teacher” field would not be a bibliographic metadata field for the audio stream of TV broadcast. A decision on which metadata field to include depends solely on the stream type. Now that we are using many stream types, a design decision should be made that allows for a stream representation in the digital library that accommodates for different metadata sets based on the type. The approach used here is the object-oriented one (as can be seen in Section 5).

There are many generic metadata fields, however, that are common to all stream types. These fields are largely based on our experience with the RFC 1807 metadata

standard for digital libraries [55] and on the Dublin Core metadata set for resource discovery on the Internet [23]. Some of the metadata fields are slightly modified semantically to accommodate for the nature of the contents of our proposed library, which are data streams. Examples of these common metadata fields are the stream type, the stream format, the creator, the description, the stream source, the start time, the end time, the entry date, and the ID. A description of all the metadata fields is presented in Section 6.

3.1.4 Stream specific software

To access data streams, certain software modules are needed. Some of these modules will depend on the stream format while others will depend on both the stream type and the stream format. The following modules are typical:

- The stream display method: this module is used to display the metadata of the stream and a playback form that enables the user to playback the stream from a selected start time.
- The stream player: this module, which is implemented as a Java applet in our example application, is used to playback the stream starting from a selected start time. This module will for most streams contact a streaming server, which will access the stream data and send them to the stream player as needed.
- The data access module: This module is used to access the stream data in order to apply the event generation criteria. It is part of an abstract layered approach to deal with the event generation process. The event object, as presented in Section 3.2, contains the event information including the criteria to detect the event. The event criteria will involve accessing the stream data and apply these criteria. To access the

stream data, the stream's data access module will be used. A case in point is a video stream where an event might be about a specific person appearing in the video for the first time. In this example, advanced, content-based retrieval mechanisms are needed to specify this criterion of an event. Compare this computation to the situation where we have a text stream of wind velocities and the event in question is that the wind velocities exceeded 50 miles per hour. In this example it is just a simple numeric comparison criterion.

3.2 EVENTS AND THEIR RELEVANT CONCEPTS

3.2.1 Definition of an event

An *event* is literally a noteworthy occurrence that happens during the life period of a stream. For example, in a TV video broadcast stream, an event might be the start of a program or the appearance of an actor in the stream. It might also be a weather alert that appears on the screen. In a radio broadcast it might be the recognition of the voice of a popular singer. In a wind velocities stream it might be that the wind speed exceeded a certain limit. In a stream of radar images it may be that a hurricane is forming. In conclusion, an event is a significant occurrence in the life period of one or more streams. An occurrence is considered an event in this research only if some expert would consider it significant for some reason. These reasons may vary depending on the context of the application. A researcher might consider a temperature exceeding certain limit as an important event that has important implications (like global warming), while an ordinary user might not even notice any difference.

3.2.2 Event-based retrieval

Searching data streams in a digital library by events, besides other searching methods, gives users an extra flexibility and allows for faster and more precise retrieval. By faster retrieval we mean that users can get to the desired stream segment faster than just by playing back the whole stream or even starting from a specific estimated time. Consider an example where a student would like to review, using a recorded class audio stream, what the teacher said in reply to a specific question. One alternative is for the student to playback the whole recorded class audio stream until he reaches the required segment. Another alternative is to randomly fast-forward this audio stream to an estimated time and then start playing back from that time. Both of these approaches would waste some time until the student reaches the required segment. Consider further the case of the student not aware of when exactly that class session was. The student in this case might need to play back the audio streams of many classes before he reaches the desired information. Searching by the event of “a question was asked of the teacher” through the whole library of class sessions would be a faster and more precise. In other words, in a digital library that contains recorded audio and video streams of class sessions, a student may want to ask about what the teacher said when some other student asked a question rather than ask about what is on the audio stream after 30 minutes from the start of the class. In the context of a streams-based digital library, people would tend to be able to inquire more naturally by events rather than by time stamps on the streams. Researchers might want to use event-based retrieval on data stream collections of a digital library to study certain phenomena. For example, for weather data coming from different sources, a researcher might want to see the sensor readings or the satellite

images when a certain event instance, such as a hurricane upgrade, occurred. An ordinary user would retrieve the news announcement aired related to this hurricane upgrade rather than an audio stream segment starting when the wind speed exceeded some limit. In summary, this event-based retrieval approach can be useful for both ordinary and expert users of the digital library and is very natural for phrasing queries.

3.2.3 Event classification

Events can be atomic or composite. An atomic event spans a discrete point in time, for example: "Netscape started" or "a hurricane degree upgraded". A composite event is an event that spans duration, for example: "Netscape running" or "Mr. Smith talking". A composite event generally corresponds to an event-pair of two atomic events: a start and an end. This classification introduces many design considerations. The emphasis in the beginning is on atomic events since every event can be represented this way; note that composite events can still be represented as two atomic events. A good example to explain this point would be to consider the composite event of "Netscape running". This composite event can be looked at as two atomic events: "Netscape started" and "Netscape ended". If there are two overlapping instances of the "Netscape running" event, this means that a sequence of events like "Netscape started", "Netscape started", "Netscape ended", and "Netscape ended" will appear in the stream. If the user is interested in playing the stream segment that corresponds to only one of the "Netscape running" composite events, the application must decide on which "end" corresponds to which "start". This requires particular information to be present in both the start and end events of the composite event and requires the applications to take extra care for these special cases and to differentiate between atomic events and atomic events that are part of

event-pairs. The other alternative is to use a single representation of all events and consider that all events are composite. Atomic events would have an interval that has an identical start and end times. Applications that access the streams and events would be easier this way and a single homogeneous style will be used. More details on our design decisions follow in Section 5.

3.2.4 Event instances

Events may occur several times during the course of one stream, for example a “Hurricane Upgrade” event occurring two or three times in the same wind velocities stream, or a “Student asking a question” occurring many times in the audio stream of one class session. We will always refer to these event occurrences with specific starting times as the event instances. When the user searches the digital library for a certain event, all the instances of this event would be returned and the user can then select what instance to start playing back the stream from. These instances are generated as part of the event generation process, where the event criteria are applied to the stream (or streams) and the list of all event instances and their times are produced and inserted in the digital library metadata.

3.2.5 Event types and classes

Event types are similar in concept to stream types, described in Section 3.1.2. A set of events that share the same characteristics are said to have the same event type. Examples would be “a hurricane upgrade from class 1 to class 2”, “teacher started a software tool”, “a stock market index exceeded a certain limit by the end of the day”, and so on. An event of “hurricane Floyd upgraded from class 1 to class 2” is of the first type.

An event “Professor Smith started Netscape” is of the second type. An event “Dow Jones exceeded 10,000 points” is of the third type. The characteristics that are shared by event types are the criteria that are used to generate the event instances, the streams types on which the events are applicable, and the metadata that are shared among all events of the same type.

An *event class* contains all events of the same type and can be used to browse through events, which could be another way of retrieval in the digital library design. An event class often corresponds to an event type and will contain all the events that share that type. A relevant example would be an event class that contains all the events of the type “a teacher started a tool”. A user can browse through all events of that type to see events like “Professor Smith started Netscape” and “Professor John Doe started an XTERM”. The user can then select one event and select an instance of this event as a starting point from which to start playing back the stream.

A generalization of the concept of event classes can be an extension to our research. The event class may contain other event classes, which will provide a tree-like way of browsing. A case in point of an event class would be “someone started a tool”. This class may contain two classes: “a teacher started a tool” and “a student started a tool”. The user can descend in the tree until reaching the desired event instance and then start playing back the stream starting at the beginning time of the selected event instance.

3.2.6 Event specific software

As with streams, events also require many specific software modules to access them. Of these modules:

- *The event display method*: this is needed to show the event metadata information and to allow the user to playback certain instances of the event. This event display method is also very important for event classes where users would browse through all events of the same type and select an event to display, as we explained in the last section.
- *The event criteria module*: this module is required to generate the events from input streams. The criteria for an event depend on the stream type, as well as the event type. For example, the criteria for the event “a student asked a question in a recorded interactive class session” would be that the student clicked a specific “interrupt” button. The criteria for a similar event in a recorded audio stream of a class would be a voice recognition algorithm.

3.2.7 Event generation

Domain or field experts are required to define and specify event instances. Software tools will be needed to help these experts to specify event criteria and apply these criteria to streams to create the event indexes for digital libraries. These tools invoke the event criteria modules, described in the last section, which in turn invoke the streams’ data access modules, described in Section 3.1.4.

Event generation may be *static* or *dynamic*. Static event generation is defined as the process of applying event generation modules on pre-recorded streams while dynamic event generation is defined as applying event generation modules in real time, at the same time as the streams are generated by their sources. Dynamic event generators are much like software agents that intercept the generated data streams and check if they satisfy the event criteria and then produce the event instances. The focus in this project is on static

event generation, partly because we do not have access to data streams while they are generated. Dynamic event generation is a potential future extension for this research.

3.3 A FORMAL MODEL FOR STREAMS AND EVENTS

We call the data unit in any stream a *frame*. A frame is a tuple that have *time* and *data* fields. Examples of the data field might be an audio packet, a video frame, or a sensor reading. Formally:

$$f = (t, d);$$

Where f is the frame, t is the time, and d is the data. A frame can also be represented as a function of the time:

$$f(t)=d.$$

A stream s is a sequence of frames:

$$s = \{f(t_0), f(t_1), f(t_2)...\}.$$

We define an event as a tuple that contain the event name and a set of times at which this event happened:

$$E = (name, T)$$

Where $T = \{t_i, t_j...\}$. Example: $E = (\text{"Hurricane Floyd upgraded to class 2"}, \{\text{September 1, 1999, 12:35pm; September 4, 1999, 1:15am...}\})$ (These are not accurate data. They are just used for explanation purposes). The name here is "Hurricane Floyd upgraded to class 2" and the set of times is $T = \{\text{September 1, 1999, 12:35pm; September 4, 1999, 1:15am...}\}$. We assume that the streamed data in the example is a stream of wind-speed readings captured by sensors every minute. The frames of this stream are the wind-speed readings.

We define the criteria of an event E as C_E . These criteria vary depending on the stream nature and the event to be described. For an event like the one in the hurricane upgrade example above, the criteria could be that the wind speed exceeded a certain limit for a specific time period from that point of time. For a multimedia stream, and for an event like “the president appeared in the scene”, the criteria will be more sophisticated requiring content-based retrieval approaches and special sets of metadata. Often in defining events, the criterion C_E is that some situation must stay true for a certain period of time. Depending on the event, C_E must stay true for a period of time either immediately before (e.g.: for stock market crash) or after (e.g.: for hurricane upgrade) a specific time in order to accept that the event occurs at that time. For example, if the wind exceeded a certain limit for a couple of minutes and then calmed down again, one cannot consider this as a hurricane upgrade. The wind speed must stay above that limit for some time period, based on expert definition, for accepting a hurricane upgrade event.

The set of times T of an event E can now be defined as: the set of times t_i of the set of frames $f(t_i)$ which satisfy the criteria C_E while all $f(t_{i-1})$ do not satisfy C_E . Formally:

$$T = \{t_i \mid C_E \text{ is not satisfied for } f(t_{i-1}) \text{ and is satisfied for } f(t_i)\}$$

A generalization for multi-stream systems is possible. Consider the set of all the streams in the digital library S . S can be defined as:

$$S = \{s_1, s_2, s_3, \dots\}.$$

Not all the streams are related to all the events. S_E is the set of streams that are related to the event E .

$$S_E \subseteq S.$$

E can be generalized to include the set of related streams.

$$E = (name, T, S_E).$$

C_E is now a function of all the related streams. So the criteria now can be represented as:

$$C_E(S_E).$$

The definition of T now should be modified. We first define $F_{SE}(t)$ as the set of frames from all the related streams that happens at time t . There is no way to guarantee that all the streams have the same frequency, thus $f(t)$ might not exist for some streams. So, we modify the definition of $F_{SE}(t)$ to be the set of frames $f(t_m)$ from all related streams at time t_m where t_m is the modulated (or synchronized) time. More precisely, we have:

If $f(t)$ exists, then $f(t_m) = f(t)$,

else if $f(t \pm \delta)$ exists (δ is a tolerance value decided by domain experts)

then $f(t_m) = f(t \pm \delta)$

else $f(t_m) = NULL$.

We assume that C_E , the event criteria, handle the situation where frames of some related streams are absent at time t_m . We further have:

$$T = \{t \mid C_E(S_E) \text{ is not satisfied for } F_{SE}(t_{m-1}) \text{ and satisfied for } F_{SE}(t_m)\}$$

3.4 RELATED STREAMS AND THEIR RELEVANT CONCEPTS

3.4.1 Definition of related streams

Event instances would normally be used not only to retrieve a segment of the stream at which this event occurred, but also to correlate this stream to other streams that relate to the same subject. We refer to other streams that the user might want to retrieve whenever he retrieves a stream as related streams. An example of two related streams is the audio and video streams for a class session. Another example is the TV weather

report and the wind velocity stream for a specific location. A user may choose to play segments of these related streams at the same time, possibly in a synchronous manner, to be able to understand the issue at hand better by correlating them.

3.4.2 Classification of related streams

A data stream might relate to another stream, to an event that occurs during another stream, or even to one instance of an event that occurs during another stream, based on some expert's definition. These related streams are called stream-related, event-related, or instance-related, respectively. An example of two streams related to each other would be the audio and video streams of a class session. An example of event related streams is the text or audio stream of a description of that event. An example of an instance-related stream is the CNN broadcast when an upgrade occurs in a specific hurricane. The original stream would be a wind velocity stream and the general event would be a "Hurricane upgrade".

3.4.3 Concurrent and non-concurrent streams

Related streams might be concurrent or non-concurrent. Concurrent streams can be synchronized while they are played back, e.g. the video and audio streams of the same class session. Non-concurrent streams can still be played back simultaneously but obviously without synchronization. An example of a non-concurrent related stream that users might need to playback at the same time they play the original stream is the news broadcast about a hurricane upgrade. The news broadcast would, almost certainly, not happen at the same time of the hurricane upgrade, although still related and would also play at a different time scale than the wind velocity stream.

3.4.4 Representing related streams

Related streams are represented as any other stream: as objects that encapsulate the stream data, metadata, and software modules. The way to specify that a stream is related to another is by including a reference to this stream in the metadata of the original one. The same applies to streams related to events; a reference of the stream would be included in the event object. Similarly, references would be included in the event object for streams that relate to the event instances.

Determining if a stream is related to another, to an event, or to an event instance is the role of a domain expert as illustrated in Section 3.5.1 below. After determining this relationship, a stream publishing tool will be used to add the reference to any related stream in the metadata of the original stream, or both. The event generation tool can also be used similarly to add references in the event objects for event-related streams or instance-related streams. Implementation details follow in Section 6.

3.5 ROLES OF DIGITAL LIBRARY USERS AND RELEVANT CONCEPTS

There are many possible users for the proposed digital library. The goal, of course, of any digital library is to serve an end user, who we call a resource discovery user. In the process of achieving this goal other users will be involved in preparing the digital library material. The system developers and administrators constitute two basic roles that will always be needed to do any needed programming assignments and administrative operations. The definitions given in the following sections of the roles of these users reflect the nature of this digital library as one that contains data streams and supports retrieval using events.

3.5.1 Domain experts

A domain expert is a user who is familiar with the subject of the streams of the specific digital library application. For example, in a weather-related digital library, the domain expert would be an expert on weather issues. In an educational digital library, the domain expert would probably be a teacher who has knowledge about the possible user requirements as to what streams will be needed to understand the class material and what events would normally happen during the class. The domain expert will be the analyst who knows what is needed in the system. Some functions the domain expert has to perform are:

- Specify the related streams and the criteria to decide if some streams are related or not.
- Specify the event criteria to be applied to streams to generate event instances. An *event generator* software module will use these criteria to generate the event tables.
- Specify the stream and event metadata.
- Suggest the search fields in an advanced search interface.

3.5.2 Publishers

A publisher is responsible for inserting the stream data into the digital library. This publisher would specify the metadata of the stream and, typically, call the indexing services of the library to create the necessary indexes. A publishing tool will be needed to help doing this task.

3.5.3 Resource discovery users

A resource discovery user is the end user who searches the digital library to retrieve and playback a stream or more based on the events that occurred during those streams as well as other stream metadata. A simple search interface is needed to support regular users while an advanced interface is needed to help improve precision and recall. Due to the nature of this digital library application, a search interface that concentrates on events and on timing information is also needed.

SECTION 4

POTENTIAL APPLICATIONS AND DESIGN CONSIDERATIONS

This section presents seven potential applications of a digital library that contains data streams and supports event-based retrieval. For each of these applications, the focus is generally on presenting the following:

- Description of the possible application.
- A tentative list of the streams to be saved in the digital library for the corresponding application.
- Events that could occur during these streams.
- Users of the application.
- Queries that resource discovery users of the digital library would want to use and the effects of these queries on the system design.

The potential applications to be discussed in this section are:

- An educational digital library that contains data streams of an interactive distance-education system.
- A weather-related digital library that contains weather-related streams such as wind velocities, temperature, pressure, etc.
- A digital library that contains census bureau statistics represented as data streams.
- A digital library that contains news-streams coming from different sources.
- A stock market digital library that contains the stock prices and index values, among other streams, and that can be used to study the effects of different factors on the price fluctuations and the market behavior.

- A digital library that contains the multimedia streams related to sports games.
- A medical digital library that contains radiology streams.

Based on investigating these applications; the significant issues that should be taken into consideration when designing the digital library system are outlined in Section 4.2 “design considerations”. These design considerations are used as guidelines in designing the architecture of the digital library. A summary of this analysis is presented in [51] and [50] as well.

4.1 POTENTIAL APPLICATIONS

This section presents seven potential applications of the proposed digital library system. These applications are carefully selected to cover different popular disciplines (education, weather, news, sports, business, statistics, and health). Each of these applications has a potential of attracting a significant audience that are interested in its discipline. For each of them, the emphasis of the discussion is on: the data streams to be stored in the digital library, possible events that could occur during these streams, potential users of the application, and possible search queries these users would want to use the digital library to obtain answers.

4.1.1 *Educational digital library*

IRI, Interactive Remote Instruction [47], is a distance education system that is being developed at Old Dominion University. IRI is an interactive multimedia collaboration system that provides a virtual classroom over a high speed Intranet in which individual computer workstations become each student’s window into the virtual classroom. IRI is a multimedia, collaborative, integrated, learning environment that

solves both learning paradigm and technical problems in a cost-effective way. In IRI, the teacher's video and audio streams are broadcast to all the students' workstations. The students can broadcast their video and audio streams to the rest of the class. A limited number of channels exist for video transmissions. The teacher or any student can start any tool on the desktop and share its view with the rest of the class. There is a control panel that contains buttons to help in the class operation or to start some tools [47], [28].

As can be seen above, many data streams are used in IRI: teacher video stream, students video stream, teacher audio stream, students audio stream, a shared room view which is used to share the view of specific application windows among all the users, and a stream of all the buttons pressed during any session, e.g. times when specific tool such as Netscape is started.

One very important aspect in any educational system is the ability of the students to review the class material at their convenience. An important piece of the class material would of course be the class presentations and discussions. Recording these presentations and discussions would then be an enhancement in any educational system. More important, however, is the retrieval of these recordings. One method of retrieval is through the later playback of the recorded sessions by the students. Complete playback of the session is not always required. Often, students will need to see only some part of the lecture which they have missed or which they want to emphasize. Hence, another approach is to allow the students to playback starting from a certain time of the lecture. This approach seems better but it is still not complete since it is unusual for a student to remember when a certain issue had happened during the class, e.g. it is unlikely that a student will remember that the teacher talked about a certain issue after 30 minutes from

the start of the class. Rather, a student may remember a particular point by thinking of an event associated with this point during the class, e.g. a question by some other student or a tool that has been used by the teacher. The student can also remember by the content of the lecture, e.g. a title of a transparency, something that has been said, etc. Thus, supporting search techniques, based on both the contents of the lecture and the events that occurred during the class, is indeed very important.

In summary, saving the streams described above in a digital library would be useful for the later review or study of the class material. Event-based retrieval from such a digital library can be used to playback some of the saved streams when a certain event occurs. Examples of some of the events that could occur in these streams: class starts, class session in progress, student "John" speaks, teacher starts a specific application; someone joins the session, etc.

Potential users of this proposed digital library would, of course, be the students to review the class material; the teachers to prepare for other lectures and to evaluate their performance and their reaction to different situations; in addition to the general public to learn about a specific topic that was covered in the recorded course. Education researchers would also be interested in studying how useful such a distance education system is and to compare it to the traditional education methods. They could do that by studying how the teacher presented his lecture, how students received it, what types of questions were asked, what was the reaction to these questions, and various teaching situations that could happen in class and the reaction of the teacher and the students to them.

To clarify how the digital library can be used, some of the possible queries that a potential end user might be interested in finding their answers in the digital library are presented. These queries, which show the validity of using events as a means for retrieval, are:

- “When the teacher started Netscape, what did he say?” Many students would be interested in this kind of query to review particular sections of the lecture instead of playing back the whole thing or trying to randomly start the playback from an estimated time. The query here involves the event: “the teacher started Netscape” and the user expects a list of all the times at which instances of this event occurred. The user would also need a reference to the teacher’s audio stream so that it can be played back, starting from any of these event instances’ times, according to the query “what did he say?” A little generalization would be to list all the streams that relate to this event in case the student needed to playback any other stream for a better understanding of the lecture, e.g. the teacher’s video or the shared room view. The user can then select which streams he would like to play back starting from which time.
- “Return all sessions in which Dr. Smith is the teacher”. A user would expect that all the streams that have Dr. Smith specified as the teacher in their metadata to be returned. Retrieval using bibliographic data is the most common way of retrieval in digital libraries and a streams-based digital library is no exception.
- “When any student started speaking, what was the running application, who was the teacher, what is that student’s name, etc?” The expected results are similar to those of the first query, except for the type of the event in the query. The system should

accommodate here for the fact that the query is about any student and not about a specific one.

- “Return all streams in which Netscape was executing and the teacher opened an XTERM”. The results here are the intersection of two composite events that might have occurred simultaneously.

From these examples, one can realize the importance of using events as a means for retrieval from digital libraries containing data streams. Some important issues have also arisen from the above examples, e.g. the retrieval of more than one stream and the simultaneous playback. These issues will be discussed in detail in the “design considerations” section.

4.1.2 Weather-related digital library

Federal agencies collect and preserve data streams from sources such as satellites, sensors, radars, to name a few. These data are very useful in studying natural phenomena like earthquakes, hurricanes, ozone layer, rain forests, and tidal waves. It is sometimes important to correlate the data coming from these sources to have a better view of the phenomenon. For example, hurricane data can be gathered from streams coming from different sources including satellites, radars, and reports. Many events may be defined to occur during these streams. For instance, a tornado is spawned, a tornado's eye hits the ground, a hurricane grade is upgraded or downgraded, the wind exceeds certain speed, and so on. Using the data coming from different sources, the student or researcher can view the satellite image, move to the news reports at the time of the event, and then see the pictures taken by amateurs and radar pictures, all by searching these streams for an instance of that certain event.

A specific application of the data coming from these sources is a weather-related digital library. In any specific area in the United States the following weather streams are collected and preserved: the wind velocity over this area, the temperature, the humidity levels, the percentage of precipitation, the amount of rain or snow, and the satellite and radar maps. There can also be streams that are specific to hurricanes or tropical storms in which the wind speeds and the locations of the hurricane eye are specified. Furthermore, there are many streams that are indirectly related to weather that can also be saved in the digital library such as TV and Radio weather reports.

Some useful weather events may be created, such as when the temperature exceeds certain limit; the wind speed exceeds certain limit, a hurricane is upgraded or downgraded, the hurricane eye hits ground, etc. The potential users of a weather-related digital library are weather researchers and experts, meteorologists, and the general public who want to know about the weather. A user uses such a digital library to answer queries similar to the following:

- “Display the wind stream starting from the time at which a hurricane upgrade was reported over this area.” The event here is the hurricane upgrade and it can be obtained from the hurricane streams: wind speeds and hurricane locations. The stream to be played back is the wind speed stream over a specific area. The user might need to playback other streams that may relate to this event like the radar maps, satellite images over this area, etc.
- “Display the radar maps and satellite images during a snowstorm”. The event here is a composite one: “a snow storm” (not just the beginning or end of it).

- “Display the TV weather reports that relate to a specific event (e.g. a hurricane upgrade)”.
- “Display satellite images, radar maps, wind speeds, etc. when the hurricane eye hits the ground”.

Again from the above queries, we will present their implications on the design in the design considerations section.

4.1.3 Census bureau statistics

The census bureau [11] is one of the bureaus of the Department of Commerce. The Census Bureau's long-standing purpose has been to provide official statistics required for effective governing. It has a center for international surveys. The census bureau conducts many surveys, provides huge quantities of data, and produces various useful statistics regularly. These survey results provide a significant amount of valuable information for different types of applications. Population statistics, health statistics, trade statistics, and crime statistics are just few examples. These statistics are repeated in a somewhat regular manner that they can be represented as data streams. Examples of these streams are national or local population statistics, health statistics, economic statistics, education statistics, crime statistics, and even international trade statistics between the United States and other countries. These streams can be saved in a digital library to be used and correlated among themselves or to other streams (like the weather or the news) by researchers or by the public.

Some of the useful events that could be recognized during the lives of these streams are: when a city's population exceeds one million, when the amount of oil

imports exceeds X, when the trade volume between US and China exceeded X dollars, etc.

A list of some of the possible queries a user might want to use such a digital library to answer follows:

- “When the trade volume between the US and another country exceeds a certain limit, what is the Gross Domestic Product?”
- “When the crime rates in a specific city exceed a certain limit, what is the population, the size of the police force, etc?”

In this application, special considerations should be given to how the streams will be played back and how will they be correlated together since the time intervals are usually extraordinarily wide.

4.1.4 Digital library containing news streams

News streams are usually rich in different media formats ranging from the text of the news piece to video and audio streams. They always come from different sources: TV and radio channels, news wires, and newspapers. Events such as “John F. Kennedy’s assassination”, “Dow exceeding 10,000 points”, and “Clinton admitting relationship” are important events and people might want to see what different news media said about them. It is also possible to correlate these events to other streams, like the stock market indexes (all of the examples actually may have effect on the stock market). Another example is to correlate “JFK Jr. disappearance” with the weather streams. There are limitless correlations that people may want to do provided that all the necessary data streams, particularly with searchable events in them, are accessible. A digital library that contains news streams can give extra flexibility to users with time constraints who cannot

follow the news on time. This has been applied before in News-on-demand applications where the most studied way of retrieval is content-based retrieval (please review section 2.3). Using events would make the digital library even more useful since it would allow these events to correlate news pieces coming from different sources, for a more complete coverage of the news story.

Studying the streams and queries that relate to this proposed digital library raises many important issues. The interoperability between digital libraries containing data streams is one of these important issues, since events in the news streams might be needed to playback streams in other digital libraries, e.g. the stock market digital library presented in the next section. Another issue is the one-time event like the disappearance of JFK junior. The events in our research usually have many instances based on criteria that can usually be automated. Such single-instance events raise few more issues like: who is entitled to specify them, how will they be represented, etc.

4.1.5 Stock market digital library

Stock markets in the United States and around the world have thousands of stocks that are dealt with. There are stock indices developed to measure the market state and activity. The stock and index values keep changing around the clock and these values are closely monitored by: ordinary stock holders, stock brokers, companies whose stocks are in the market, governments, economic researchers and students. Since these values are changing over time, they can constitute data streams.

Saving these streams in a digital library would be useful for all those kinds of users described above. Any user could play back a stream and follow the pattern of

change of any stock or index value. This library can basically be used to study the effects of different factors on the price fluctuations and the market behavior.

Examples of the events that could happen during these streams are: the stock value exceeded a certain value or went below a certain value; a market crash, DOW Jones increased while NASDAQ decreased, and an index maintained its value over a certain number for a specified period of time. Market experts are needed to specify the useful events that could happen in such a digital library.

An extension to the system, which is beyond the scope of this research, is to let the user specify some queries based on these events as stored agents. When the event occurs the agent could warn the user so that he can take a specific action. For example, one of these agents could be a query involving an industry index going below a certain limit. The agent would then warn the user to sell or take any action if this event occurred.

The stock market is usually volatile and many external factors can affect its performance. An example is the terrorist attack on the World Trade Center towers in New York. Stocks went down because of this attack. The same happens with any political upheaval. This again raises the important issue of interoperability between digital libraries containing streamed data and how one can retrieve streams from one of them based on the events happening in another.

4.1.6 Sports games digital library

There are many streams that can be recorded from sports games. These streams come from many sources, mainly media sources such as TV cameras or radio transmission. There is a huge audience for these games whatever they are, football, basketball, baseball, or some other sport. There is a great interest in watching, analyzing

and studying these games. Saving them in a digital library would be useful for ordinary audience, basically through TV sports programs, to playback parts of famous games, e.g. the NBA finals in a specific year. Saving them would also be useful for coaches and trainers to study some techniques and plans to build on in their work.

The contents of such a digital library will mainly be the video and audio streams of the TV broadcast (or the video recording of the game), usually coming from different cameras and microphones, the newspapers commentaries, and other TV programs and news segments that deal with sports news. Examples of events that can occur during these recorded sports streams: A scored goal, a player gets removed from the game, or a fight among players.

One interesting query that an end-user would use such a digital library to answer is: "Play the video stream starting from the time the first goal was scored". In this query, the event is "first goal was scored" and the stream to be retrieved is the video stream of the game recorded. The identification of the starting time of this event instance should be based on the type of the game, the overall situation before scoring, which is a complicated decision. It would have to be done by a human expert, whose decision on the starting of this event instance is more likely to be acceptable to most potential viewers. Naturally the user would also like to play the audio stream. Furthermore, the user might want to see the goal from different angles, so multiple video streams coming from different cameras would also be required.

4.1.7 Medical digital library

A digital library that contains medical streams; namely Computed Tomography (CT) scans, medical text, and related audio streams is described in detail in Section 6. This digital library is implemented and it serves as a case study for the proposed system.

4.2 DESIGN CONSIDERATIONS

The last few sections presented examples of some applications of the proposed digital library. In order to design such a digital library; our approach was to think from the user's viewpoint. We compiled a list of possible queries a user might want to pose to the digital library. From these queries, like those mentioned in Section 4.1, several issues were realized. In this section we list the most important of these design considerations along with their resolution. In the next section the design considerations listed here are used to specify what entities are needed in the library architecture and what software modules are needed to enable the library.

There should be multiple modes for retrieval

This research is mainly about using events for retrieval. However, as shown in the last few sections, searching stored data stream segments in digital libraries by event instances that occurred during the generation of the corresponding stored streams is not the only way for retrieval of data streams. Users will often want to use bibliographic data, e.g. title, creator, or type for retrieval as well. For example, in the educational digital library described above, one of the examples was "Return all sessions in which Dr. Smith is the teacher". In this example the resource discovery user uses a bibliographic item, the teacher's name, as the means of retrieval. The issue that should be taken into

consideration in the digital library design is the need to keep bibliographic metadata as well as other event metadata and allow the user to use these bibliographic metadata for retrieval.

An event usually occurs more than once in a data stream

Multiple instances of an event may occur during the generation of a data stream. In the educational digital library example, the event of “a student asked a question” would normally happen many times in the span of a stream. A user would want to select a specific instance of such an event to playback the class streams. Another example from the weather-related digital library would be retrieving any stream when a “hurricane upgrade” occurred. A hurricane upgrade would occur multiple times in the course of one wind velocity recording stream. A user will need to be able to choose a specific event instance starting time to start playing some corresponding data stream segment. The design issues that need resolution are how to represent all the event instances and how to allow the users to select only one instance from which to start the stream playback.

Some events occur only once (single-instance events)

Although most events would occur many times in the course of any stream, there are still some events that would happen only once. An event like “Twin Towers are destroyed” only happened once. Similarly, “Clinton admitted relationship”, “JFK junior disappeared”, and “Barry Bonds sits home run record”. This type of events raises many issues that should be taken into consideration in the digital library design. One issue is regarding who will be entitled to specify that a particular happening is important enough to be saved in the digital library. Another issue is how this kind of events will be

generated. Obviously, it is very difficult to automate the generation of such events; so manual event generation is necessary in these cases. A third issue is the representation of such events; especially if there are many streams that are related to the event (this last issue is actually relevant to multiple-instance events as well).

There maybe a very large number of events to browse through in a digital library

In any one of the digital libraries mentioned in Section 4.1, there might be a very large number of events to be used in the retrieval process. This opens the very important issue of specifying the event to use in the search. Of course, one possible way is to use keywords to retrieve some events based on their titles or other bibliographic metadata, but this will not be helpful in browsing. One other way is to use some hierarchy in which events that have some common characteristics are clustered together.

Events may have duration (atomic and composite events)

Our handling of events when this research was started focused on events that occur at a discrete point in time, or what we call atomic events, much like the definition of an event in simulation theories. Examples of these events are the “teacher started tool”, “student started asking”, “a hurricane upgrade”, “DOW exceeds 10,000 points”, etc. In the examples above, however, one could see some other types of events which are of interest to many users and which have a period, not just a point in time. For example, “while Netscape was executing, what other streams were started by the teacher?”, or “show the radar maps or satellite images during a hurricane”. In both examples the event, “Netscape executing” or “there is a hurricane”, lasts for a period of time not just a point in time. We call these events composite events. One issue that should be taken into

consideration while designing the digital library is how to represent the events in the digital library in such a way that would accommodate for both atomic and composite events. A decision should be made whether to represent all events as composite events (with atomic events having same start and end times), or to represent all events as atomic events (with composite events represented as two atomic events, one for the start and the other for the end). Another issue is how to use different types of events in the retrieval system. A third is how the streams' playback will look like in the case of composite events. The resolution of all these issues is described in Section 5.

Issues with event generation

Another important issue is how events will be generated and who is responsible for the event generation. It is apparent that in the current state of the art, manual definition of events and event instances is necessary for many types of events due to their complexity. Furthermore, for the same reason, not everybody should be allowed to define what an event or event instance is, only qualified domain or field experts should be allowed to do this. Software tools will be needed to help these experts to specify event criteria and apply these criteria to streams to create the event instances for digital libraries. The help of the programmers will certainly be needed to create the necessary programs to be used by these experts.

Many events can be generated automatically using the proper event-generation software tools, especially those occurring in text streams. The issues here are how these tools will function and how they will use event criteria for different types of events. Another issue is that event criteria will depend on both the event and stream types and separate modules will be needed depending on those types. The criteria may range from

simple comparisons to, maybe, content-based comparison algorithms. As every stream type will have its own player application, it'll have its own data access modules to be used with the event criteria to generate the event instances.

Single-instance events, as discussed above, will need to be specified manually. There should be an appropriate software tool that plays back the streams and allows the experts to manually specify the start and end of any of such event.

Accommodation for related streams

A data stream might relate to another stream, to an event that occurs during another stream, or even to one instance of an event that occurs during another stream, based on some expert definition. An example of two streams related to each other would be the audio and video streams of a class session. An example of streams that relate to events is the video stream of a specific person whenever this person starts talking in a class session's audio stream. An example of a stream that relates to an event instance is the CNN broadcast when an upgrade occurred in a specific hurricane (see Section 4.1.2). The original stream in the last example would be a wind velocity data stream and the general event would be just a "Hurricane upgrade". As was shown in the last few sections, many users would sometimes want to retrieve the related streams when they retrieve a stream, event, or event instance that relates to them.

A user may choose to play several stream segments, which may be of different types, in a synchronous manner to be able to better understand the issue at hand. For the example of the audio and video streams of the teacher in any class session, it would be very normal for any user to choose to playback both streams at the same time.

Related streams might be concurrent or non-concurrent. Concurrent streams can be synchronized while they are played back, e.g. the video and audio streams of the same class session. Non-concurrent streams can still be played back simultaneously but obviously without synchronization. An example of a non-concurrent related stream that users might need to play back at the same time they play the original stream is the news broadcast about a hurricane upgrade. The news broadcast would, almost certainly, not happen at the same time of the hurricane upgrade, although still related. A controlling application will be needed to play back these streams at the appropriate rate.

Ancillary issues are:

- Who will define whether a stream is related or not and how?
- How will they be represented with regard to the original stream, event, or event instance?
- How will they be retrieved? What will appear on the search interface?
- How will they be played back? Separate browsers or same browser? Can they be simultaneous or not and how to know that?

The system should support many stream formats

There are various stream formats to be saved in the same library and played back by users. In the educational digital library above we showed many examples: teacher video and audio, students' video and audio, images, text, etc. There have to be a specific player for each stream format. Important design issues are how to make the library extensible to new formats and how to ensure that the appropriate player is used with every stream.

There will be many stream types

Even if some streams share the same format, they might still be different in many other characteristics. A teacher's video and the students' video streams would be slightly different in terms of the metadata to be assigned to each. A teacher's video stream and the TV video stream are totally different in this regard. Distinguishing stream types and accommodating for different stream types in the same digital library is a very important design issue.

SECTION 5

STREAM BASED DIGITAL LIBRARY: ARCHITECTURE

Section 4 presented seven potential applications of the digital library system. For each of these seven applications, a list of requirements in its corresponding digital library system was assembled based on the queries that the end user might need the digital library to answer. These requirements were presented in details in a “design considerations” section by the end of the section.

In this section these considerations are used to design the architecture of a digital library that contains data streams and enables event-based retrieval. The major aspects of the architecture that are discussed in this section relate to the contents of the library and how they are organized, the metadata used to enable their retrieval, the software modules needed to allow for the access of the contents of the digital library, and the interaction between the various users of the digital library and its utilities.

The section is organized as follows: Section 5.1 starts with the data stream organization in the digital library, explaining why they are organized in objects not in just database-style tables, the structure of the stream object, and how these data streams are published into the digital library. Section 5.2 presents the organization of events. Again, questions like why they are represented as objects, how they will be generated, and how they will be retrieved are answered. In Section 5.3, the retrieval architecture of the digital library is presented. This section revolves basically around what should be expected by the resource discovery user when she tries to retrieve and playback a stream stored in this

digital library, ranging from the search interface to the stream playback process. In Section 5.4, a summary of the general library architecture is presented.

5.1 DATA STREAMS

In this section, the organization of data streams in the digital library is presented. The major issue in here is to save data streams in a digital library in such a way that they can be retrieved and played back efficiently. We have discussed many design considerations that would directly affect the data organization in Section 4. We summarize in here some important observations and present the conclusions that directly relate to the data stream organization:

- The data streams of interest come from different sources, such as satellites, radars, TV broadcast, and text tables. The basic assumption is that these streams will be saved in files in some popular format. We further assume that the data stream generated from a source is saved in one or more files.
- There are usually many stream formats to be included in the same library, for example, text, audio, video, and sequences of numbers. In the example of the educational library presented in Section 4, video format (for the teacher and the students' video streams), audio format (for their audio stream), text (for the stream representing button clicks) are all stream formats that should be handled.
- In a stream-based digital library, there will typically be a set of different stream types. Data streams sharing the same format may be different in the corresponding bibliographic metadata. A teacher's video and the students' video streams would be slightly different in terms of the metadata to be assigned to each.

- Usually, when a user retrieves a stream for playback, he may desire to playback another related stream. Again, as explained in Section 4, a student would normally wish to play the teacher audio whenever he retrieves and plays the teacher video.

Many streams of different types and formats will be saved in the same digital library. A different player will be needed to playback any of these streams. Separate modules will also be needed to query any of these stream types (to generate the events in case the event generation can be automated). It is more flexible to encapsulate the stream data files, metadata files, and all the software modules needed to access this stream in one stream object. In the object-oriented approach, the digital library can be extended easily to accept new types with minimum changes in the code. The stream can be moved around, maybe to other libraries or for personal use, easily with all its required metadata and supportive code without a lot of hassle. Based on these realizations, one could conclude that the best way to organize the data streams in this digital library is by using an object-oriented approach [65].

5.1.1 Structure of the stream object

Each stream object would encapsulate the stream data, the corresponding metadata, and the software modules that are needed to access the object. The structure of the object is shown in Fig. 1. The description of the object contents is as follows:

The stream data

These are the data stream files. As mentioned before, there may be one or more files coming out of each stream source. The stream object will either contain these files or

contain references to their actual locations. Provisions must be made to ensure that the stream playback software will be able to access the stream data wherever they are.

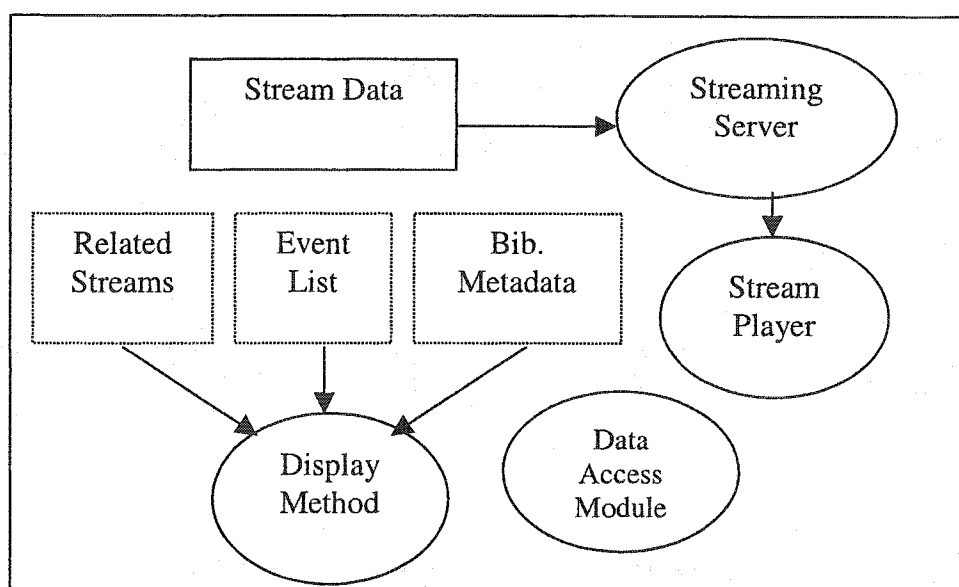


Fig. 1. The stream-object structure. Rectangles indicate the stream data or metadata, and the ovals indicate software modules.

The stream metadata

The stream metadata are data that facilitate the retrieval and playback of the desired stream segments. These metadata consists of the bibliographic information of the stream, an event list, and a related-streams list.

The stream bibliographic information is needed to retrieve the stream based on some textual, non-temporal information of this specific stream, e.g. the stream title, creator, description, start time, end time, type, or format. The stream type is one important field among these self-explanatory fields. A set of streams that are of the same stream type will also share many characteristics such as the metadata fields associated with these streams, the criteria to be used to specify the events for this stream, and the

players to be used to play this stream. The details of the bibliographic data to be used with some stream types are presented in Section 6, which is about implementation details.

The event list is a list of all significant events that have been recognized to have occurred during this stream. As will be seen in the next section, events will also be represented as objects, so the event list will be a list of references to the corresponding event objects. The minimum event information of the event name will be needed to identify the event. Having only minimum information means that the object size is smaller, hence its retrieval is faster. On the other hand, not having a complete event information adds another step in the retrieval process for the user to see this information if she so wants, a compromise we are willing to take since most of the time the interest is in the retrieval and playback of data streams. Whenever a stream object is to be displayed, this list of events will also be displayed and the user can select an instance of any of these events and start playing back the stream starting at the time of this event-instance. Creating an index on event names just like any other bibliographic information will enable retrieving the streams in which this event occurred.

The third major item in the stream metadata is the list of other stream objects that are related to this stream. Just as we decided for the event list, the related-streams list will be a list of references to the corresponding stream objects. Whenever a stream object is to be displayed, the list of related streams will also be displayed and the user can then select other streams to be played back simultaneous to the original stream.

The stream-specific software modules

For every stream type, certain software modules will be needed to access the stream objects of this type. The stream object will contain the following software modules, or the references to them:

- The stream display method: This module displays the bibliographic record of the stream as well as a playback form. The playback form lists the event list and the related-streams list and allows the user to select an event instance and some related streams to play back the stream and the selected related streams starting from the selected event instance time.
- The streaming server: This module is used to send the stream data to the stream player starting from a specific start time to an optional end time.
- The stream player: This module is a client side module that receives the stream data from the streaming server and displays it to the user.
- The data access module: This module is used by the event generation application to create the events for this particular stream type. This module is necessary because of the difference in specifying the event criteria from one stream type to another. An example is a video stream where an event might be about a specific person appearing in the video for the first time. In this example advanced content-based retrieval mechanisms are needed to specify the criteria of an event. Compare this to a text stream of wind velocities where the event might be that the wind velocities exceeded 50 miles per hour. In this example it is just a simple numeric comparison criterion. Having this module in a stream object means simply to let the stream specify how it will be queried.

5.1.2 Stream publishing

In this section, we talk about stream publishing concepts as part of our digital library design. As explained above, stream data will come from different sources and will be stored in the digital library as stream objects. Some tools are needed to create these objects. These tools as well as any related concepts are described below.

The stream template

The stream template, as the name implies, is a sample stream object that does not contain any values for data items or any metadata about any specific stream. It contains all the necessary software modules that are related to a particular stream type. A stream template also contains information about the bibliographic metadata fields of streams of this type. As explained above, depending on the stream type, the stream metadata may vary as well as some software modules, e.g. the stream player and the data access module. Having a stream template for each stream type simplifies the publishing process. All that is needed for a user to publish a stream, is to know the stream type, use the appropriate template, and “fill in” the missing parts, mainly the data and some specific metadata. Stream templates are created by the digital library administrators and are kept in a specific repository to be used by the publishing tool. A mapping table is needed to map from the stream type to the appropriate stream template. These templates should be modifiable and one should be able to duplicate them so that other templates can be created from them, given that many of these stream templates will differ only slightly. This minor difference would come from the fact that many streams that have different types would still have the same format, e.g. TV audio stream and the teacher audio

stream. The difference will basically be in the bibliographic metadata not in the player software.

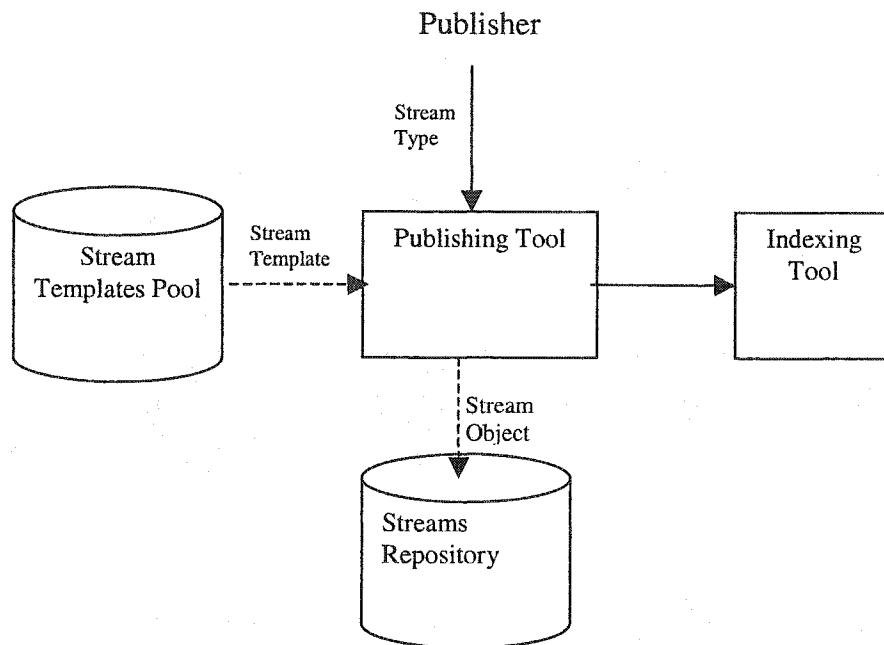


Fig. 2. Stream publishing. The (solid rectangle) indicates a software tool, the (cylinders) indicate the data repositories, and the (dashed lines) indicate data transfer.

The stream publishing tool

The stream publishing process is shown in Fig. 2. The publishing tool should be used to publish the stream data into the digital library. A *publisher* is the digital library user responsible for publishing data streams using this tool, as explained earlier in Section 3. This tool should function as follows:

- The publisher specifies the stream type to this tool.
- The tool then looks up a mapping table to load the appropriate stream template.
- The publisher then uploads the stream data files into the stream object created from the template.

- The publisher uses the tool to fill in the stream bibliographic metadata.
- The tool then, optionally, calls the appropriate indexing tools to index these new metadata items. Note that re-indexing for every new record is very inefficient.
- The tool also calls, if the publisher wishes, the event generation tool.

In the above scenario, we assume that the user would publish one stream at a time. However, this is not always the case, sometimes mass publishing is needed. In mass publishing, the tool has to handle a large number of streams, usually of the same type, and automate the creation of the stream objects with no or minimal user interaction. For example, if we have all the recorded audio streams of the lectures of Professor Smith, then the tool can create all the necessary stream objects and specify the metadata for all of them without any user interaction. The only difference in the bibliographic metadata would be in the stream start and end times, which usually can be obtained automatically from the physical attributes of the stream data files.

5.2 EVENTS

This section presents the aspects of event handling in the proposed digital library architecture. In the last section we showed that event tables are inserted in every stream object as part of the stream metadata. These tables contain references to the event objects as well as some minimal event information like the event name. In this section we explain in detail why we decided to represent the events this way. As we did in the last section, we show here the design considerations that affected our design decisions with regard to events.

- Some events need to be part of the metadata of many related streams. An example would be that a student interrupted the teacher during the class session. This event

would be part of the student's video stream as well as her audio stream. If there were a tool to get the teacher's attention in the distance-education system, the event would also be part of the tool stream (a stream of all the button clicks that occurred during the class session). The reaction of the teacher to such an interruption might be important, for any reason, to some users so the event might as well be part of the teacher's audio and video streams.

- The criteria of some events may depend on more than one stream. For example, an event like "A hurricane upgrade on Norfolk" depends on both the location of the hurricane and the wind velocity over this area. The criteria in this case cannot be associated with only one stream object.
- The event criteria will differ from one event type to another, making it necessary to define a means to associate the criteria with the events satisfying them.
- With every event, there will be a lot of information that is associated with it. Related streams, time instances, event criteria are examples of these information. This makes it inefficient to save all the event information in the stream object containing this event, especially if the event is repeated in many streams.

Based on the design considerations outlined above, the object-oriented approach used for streams objects, as described in the last section, is taken a little further in our digital library design. Instead of representing events just as sets of tables that include the event name and the set of times at which this event occurred, and instead of repeating the event information in all the streams that relate to it, the event is represented as an object that contains the times of all instances of this event, references to all the streams that relate to it, and its own criteria information. We further allow the event to be part of the

search results, besides the streams at which this event occurs. It is more flexible and more efficient to have the event represented as a separate object. Minimal information about this event object, e.g. the object ID and the event name, will need to be saved in the relevant streams' objects.

5.2.1 Structure of the event object

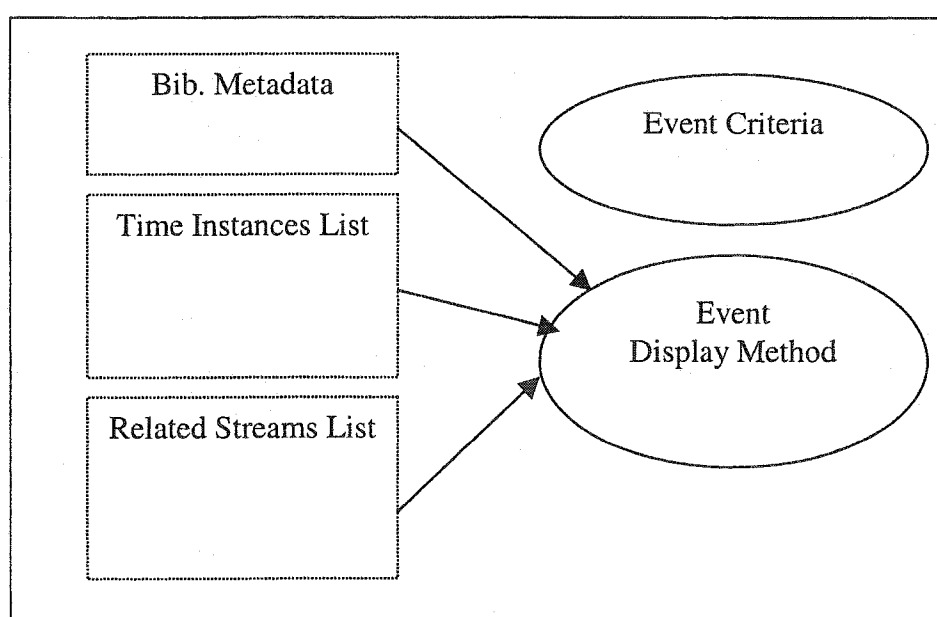


Fig. 3. Structure of the event object.

The event object will contain a list of times of all the event instances, a list of the event-related streams, and the software modules specific for event handling such as the event criteria used in event generation. To make event objects more independent and retrievable from the search interface just like stream objects, bibliographic metadata of the event will also be stored in the event object. The structure of this event object is shown in Fig. 3. The contents of the event object are described below.

The bibliographic metadata of the event

The bibliographic metadata of an event is used to describe the event information and the event object. These metadata can be used to retrieve the event directly through the search interface based on some non-temporal information. The event id, the event name or title, the event description, and the event creator are typical bibliographic fields of an event object.

The time instances list

This list is either a list of times (in an atomic event object) or a list of time-pairs (in a composite event object). Whenever an event object is retrieved, the list of time instances of the event is displayed so that the user could choose a specific instance of the event to start playing back the selected related streams starting from this instance's time.

The related streams list

This is a list of the streams that relate to this particular event. It typically includes the stream ids of the objects containing these streams. This list is vital so that when the event object is retrieved independently, the user could choose the streams he wants to playback starting from a time instance of the event. This is particularly useful with single-instance events where it is more important to browse through the event object than through the streams in which this event occurred.

The event criteria

This is a software module that is used by the event generation software to generate the instances of the event represented by this object. This module contains the instructions to generate this event. The event criteria module will depend on the criteria

modules in the stream objects involved in creating this event. For example, if the event is “student seeks teacher attention in a distance education session”, then the criterion for this event as specified in the event object would be a module whose function could be as simple as “call the data access module in the *buttons*-stream object and get all the times in which the ‘Attention’ button was pressed” (it could also be more complicated, making sure that the button was pressed by a student while the teacher was speaking). The data access module in the buttons-stream specifies how the query would actually be specified depending on the physical nature of the buttons stream. So, this module might include a search in the entries of a text table that contains the names of the buttons pressed during any session along with the times at which these buttons were pressed.

The event display method

This module is called when an event object is returned as a search result and the user wishes to see the contents of this event object. This module displays the bibliographic record of the event as well as a playback form that lists the time instances list and the related-streams list and allows the user to choose an event instance to start playing back the selected related streams from. With this module, the event object can be accessed independently from even the digital library containing it.

5.2.2 Event classes

Events also have types just like streams. An event type would affect the event metadata and its software modules, especially the criteria modules. We clarify this concept using an example. Consider the event “student X interrupted the teacher to ask a question”. The criterion for this event can be that the “attention button” was pressed

while the teacher, and not any other student, was speaking. All the events that involve a student interrupting the teacher to ask a question will have the same criterion, whatever it is student X or any other student. We can assume the event type in this example as “a student interrupted the teacher to ask a question”. A template object that depends on the event type and that contains all the common information and software modules of all the event objects of that type can be used to generate the event objects. The scenario for this process will be explained when we talk about event generation.

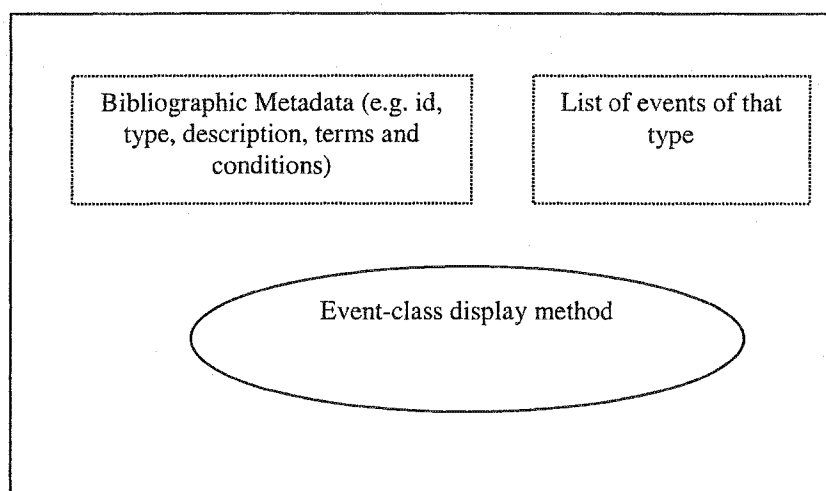


Fig. 4. Structure of the event-class object.

A simple generalization that would enhance the functionality of the system is by creating an object, the *event class*, which is retrievable itself. This event class object will contain some general descriptive information about this event type. It will also contain a list of all the events of that type. This object, as can be seen in Fig. 4, also contains its own display method that is different from those for the event object and the stream object. This display method first lists the bibliographic data of this event class object and

then displays the list of event objects that have this type. The user can then select one of these event objects for further information. In the example, the user can retrieve the class “A student interrupted the teacher” and from there he can choose a specific student and display her event object.

5.2.3 Event generation

In this section we show the architecture for event generation in the digital library. In the last two sections we talked briefly about event criteria and how they will be saved in both the event and stream objects. We also talked about event templates and how they would be used in generating events. This section gives more details about these aspects in addition to other important issues.

Basic process of event generation

The basic process of event generation is outlined in this section and shown in Fig. 5. Assuming that a new event that was never created before is being created, the process will be as follows:

Domain or field experts define and specify the event criteria and what they involve. This is basically a paper and pencil job. For example, the weather experts, who are not usually programmers, would decide what the criteria are for a hurricane upgrade, probably the wind velocity exceeding a certain limit for a certain period of time. Those experts are also expected to specify the related streams of such an event and the bibliographic metadata fields to be used with it. The domain experts would translate these event criteria into criteria modules using an event editing software tool that would allow

the domain experts to build the event criteria modules using some pre-existing building blocks prepared by the digital library developers.

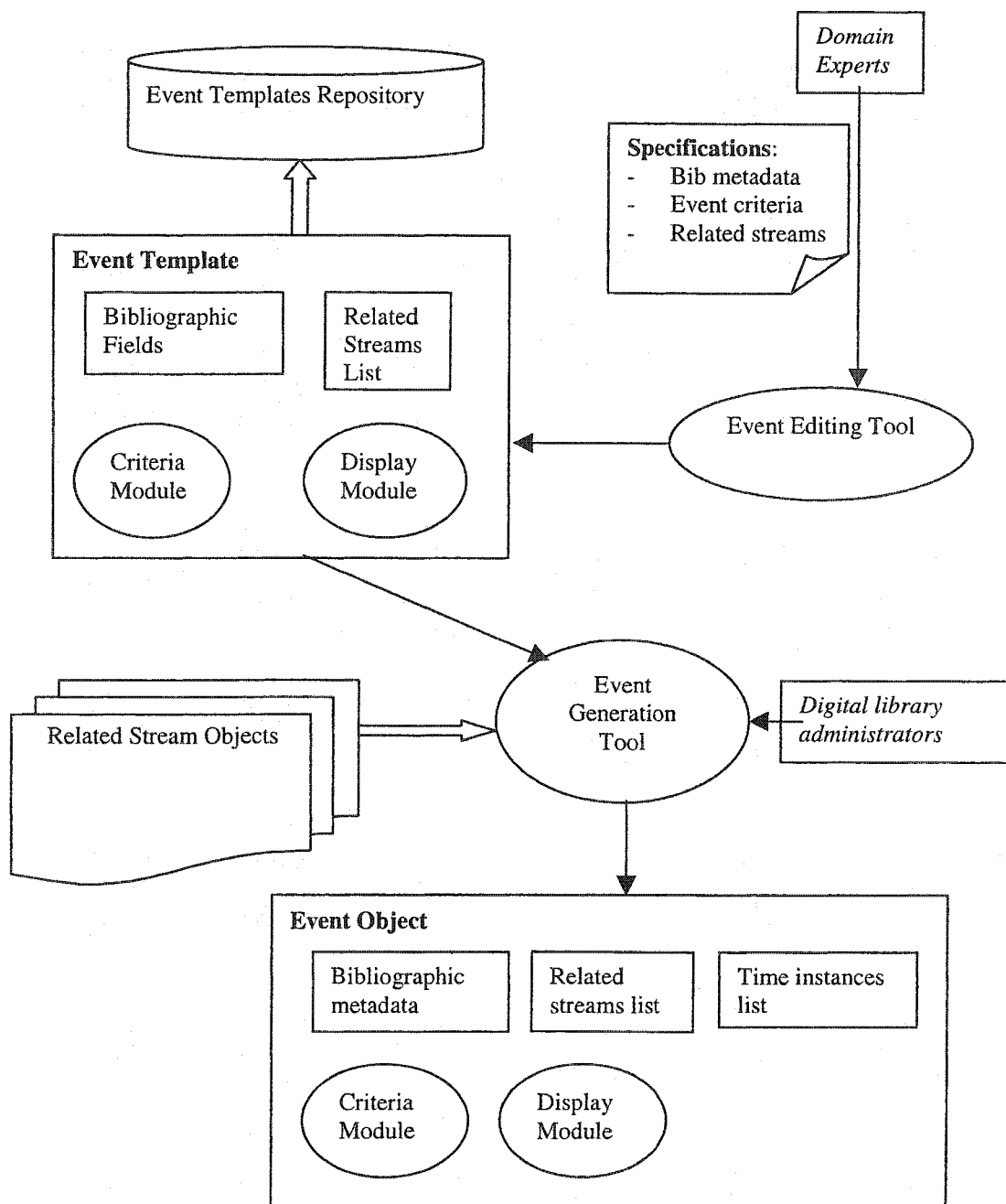


Fig. 5. The event generation process.

These modules call the stream-based data access modules whenever a stream is to be accessed, so that the event modules would be independent from the streams. This is a layered, object-oriented approach that would allow the same stream-based data access modules to be used by many events' criteria. The output of the event-editing tool is an empty event object with the necessary file structures to hold the bibliographic metadata and the related streams list based on the information provided by the domain experts and explained in the last step. The criteria module and the display module are added to the event object. The empty object just created is considered an event template to all events of the same type. This object will be stored in a templates' repository so that it can be used later whenever needed.

The digital library administrator calls an event generation tool to create new event objects using the templates created in the last step. The inputs to the event generation tool are the event template and the streams on which the event criteria depend. The output will be the complete event object, including specified bibliographic fields, their values, and a list of the time instances at which this event occurred. The event generation tool will also modify the related streams' objects to include the new event as one that occurs during those streams.

Static vs. dynamic event generation

As described in Section 3.2.7, event generation may be *static* or *dynamic*. Static event generation is defined as the process of applying event generation modules on pre-recorded streams the same as described in the last section. Dynamic event generation is defined as applying event generation modules in real time, while the streams are generated from their sources. Dynamic event generators are much like software agents

that intercept the generated data streams and check if any unit satisfies the event criteria and then update the corresponding event object.

We leave dynamic event generation as a potential long term future extension of this research.

Automatic vs. manual event generation

For some media formats, e.g. video, event generation might be very hard to automate. Consider a query about a specific person appearing in the video frame. Extensive research is still being done regarding face recognition [52]. Many cases should be taken into considerations when a software module tries to determine if a human face appeared in a video frame, e.g. brightness, different face profiles, background, etc. The point is that it is sometimes very hard to automate the event generation. An alternative is to use manual event generation. In this approach, a technician uses a software tool to browse through the stream. This tool would allow the technician to just hit a button when an instance of an event occurs and it then records the time of this instance. The drawbacks of this approach are obvious. Missing some time instances of the event will not be unexpected. The time needed to generate the event instances will be much higher than the automatic approach. Nevertheless, this approach had to be presented because of the limitations of the automatic approach with difficult, very sophisticated, or expensive event criteria modules. We decided to leave automatic event generation as a potential future extension.

5.3 RETRIEVAL ARCHITECTURE

This section presents the details for the architecture for retrieving and playing back the data streams in the proposed digital library. In Section 5.1, we present the publishing process of the data streams; Section 5.2 focuses on the event generation and representation. In this section we describe the general retrieval process as summarized in Fig. 6.

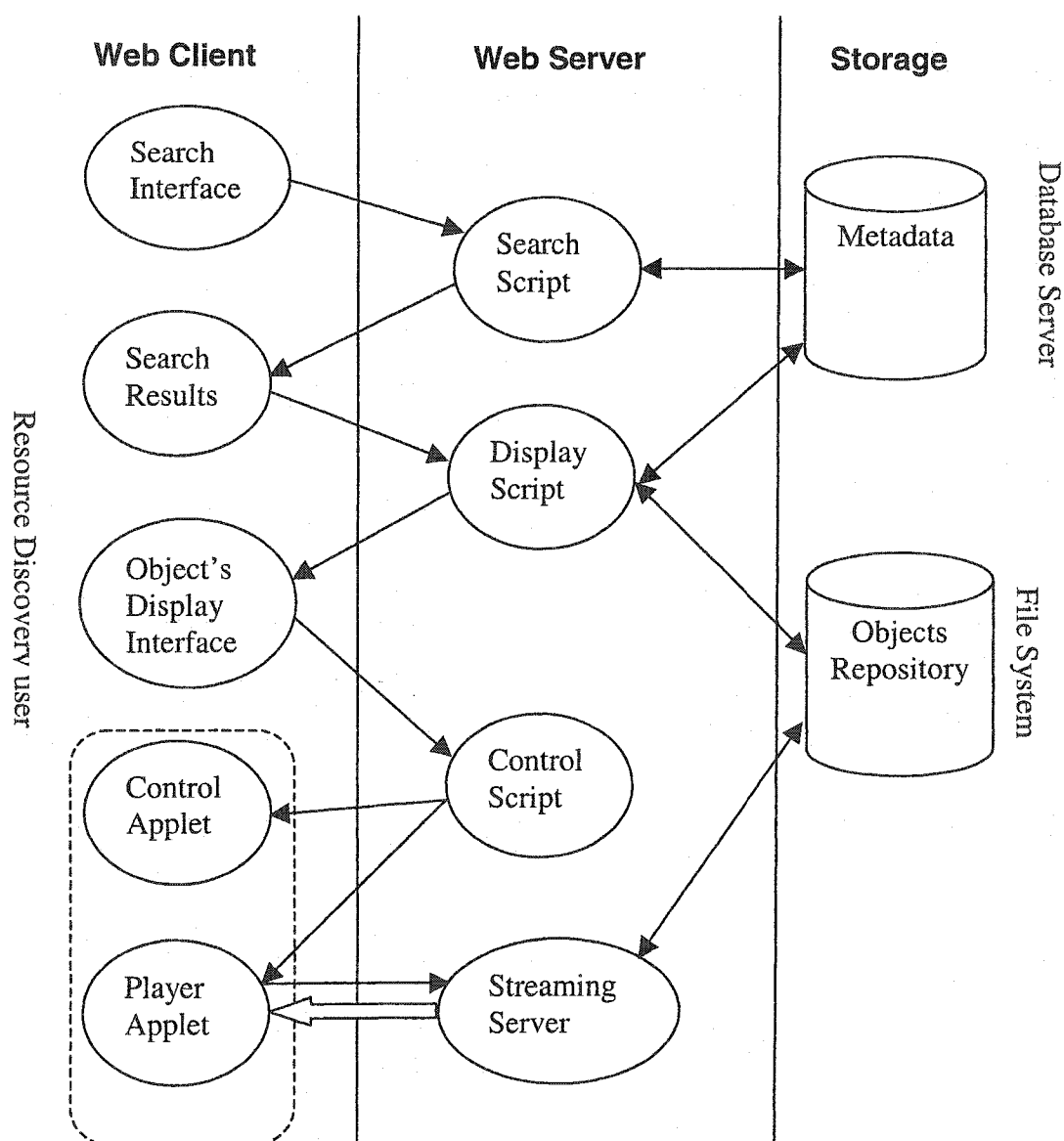


Fig. 6. Retrieval architecture.

5.3.1 Database backend

Whenever a stream is published in the digital library, its bibliographic metadata are stored in a database backend. The fields of this database are indexed so that they can be searched and the selected stream can be identified and retrieved.

Since events and event classes in this architecture are also retrievable just like streams, their bibliographic metadata are also stored and indexed in the database.

Section 5.1 revolves around and discusses the concept of stream types. One of the implications of supporting this concept in the digital library is that different streams may have different sets of bibliographic metadata. For example, the teacher video stream in the educational digital library discussed in Section 4.1 may have a bibliographic field called “teacher”. It may also have another field that is called “frames per second”. These fields depend on the stream type and format and may not exist in other streams of different types in the same digital library. The database design must accommodate this. One approach is to use a different database table for each stream or event type. Since all streams or events from the same type will have the same set of bibliographic metadata, one table will be sufficient to hold the metadata of all the objects of that type. Another alternative is to use one table and combine most of the metadata items in one field and then use some sort of text search on this field. This latter approach is more efficient in this application since there will be many stream types. Combining the XML technology [40], where field names are represented as tags, with this approach will make it rather easy to display the metadata of the retrieved object.

5.3.2 Search script

The search script is a program whose function is to display the appropriate search interface to the resource discovery user who should use this interface to specify the search terms. The search script would then read these search terms and form a query to be sent to the database backend, as introduced in the last section. The search script would then receive the search results from the database, put them in the appropriate form, and display them to the user. The search script formats the search results in such a way that allows the user to retrieve the details of the object he selects just by clicking on an element of the result page.

The Search interface

For this research, the most important item of the search interface is the support for using events for retrieval. Users may know the event they are looking for by name, or just by some keywords. The search interface should support both alternatives. Events could be listed in a menu so that the user can select those events by name. A free text area could be part of the interface so that the user can enter the words she thinks are part of the event name. In the menu option, a solution must be found to the problem of listing a very large number of events. One approach is to use “clustering” or “grouping”, where every set of events is grouped in one logical group. The user can select the group and then select from the events that belong to this group. Limiting criteria are needed whenever there are a large number of instances of the same event occurring during the span of the streams saved in the digital library. The most appropriate factor is the period during which the event instance occurred.

A more advanced search interface should enable the user to search by other bibliographic fields in the metadata of the stream or event objects. A standard set of fields such as the title, creator, subject, start time, end time; etc could be used in an advanced search interface besides the events and the keywords. Since this research is mainly interested in using events, this alternative interface will be left as part of future extensions of the system.

The Search results

When the search script receives the search terms from the search interface and forms a query that is to be sent to the database backend, a set of search results is obtained from the database. This set will contain a minimal, but sufficient, set of information about the retrieved objects and should be organized in a form such that the user could select whatever he wants for further details and for playback, if needed. The results will be organized in separate records. Each record will contain information about the object, whatever the object type is: stream, event, or event class, such as the title, the id, the start and end times, the creator, etc. A link should be associated with the record so that by clicking this link, the user can view more details about the object. This interface should also allow the user to refine the search, in case there are too many results. One way is by specifying more search terms and re-executing the search on the returned results only.

5.3.3 Object display

By clicking on the link associated with an object, a complete set of bibliographic information is displayed. By browsing through this bibliographic information, the user may choose to continue further with the playback of any selected streams. If the selected

object is a stream object then the user should be able to select one time instance of an event that occurred during the stream and start the stream playback from that time. The user should also be able to select any other related streams he might want to playback simultaneous to the original stream. If the selected object is an event object, the user should be able to choose an event instance and some related streams to play back. If the selected object is an event class, the user should be able to select one of the events belonging to this class and continue as in the case of an event object. Details of the display methods for the stream, event and event class objects are demonstrated in sections 5.1.1, 5.2.1 and 5.2.2 respectively.

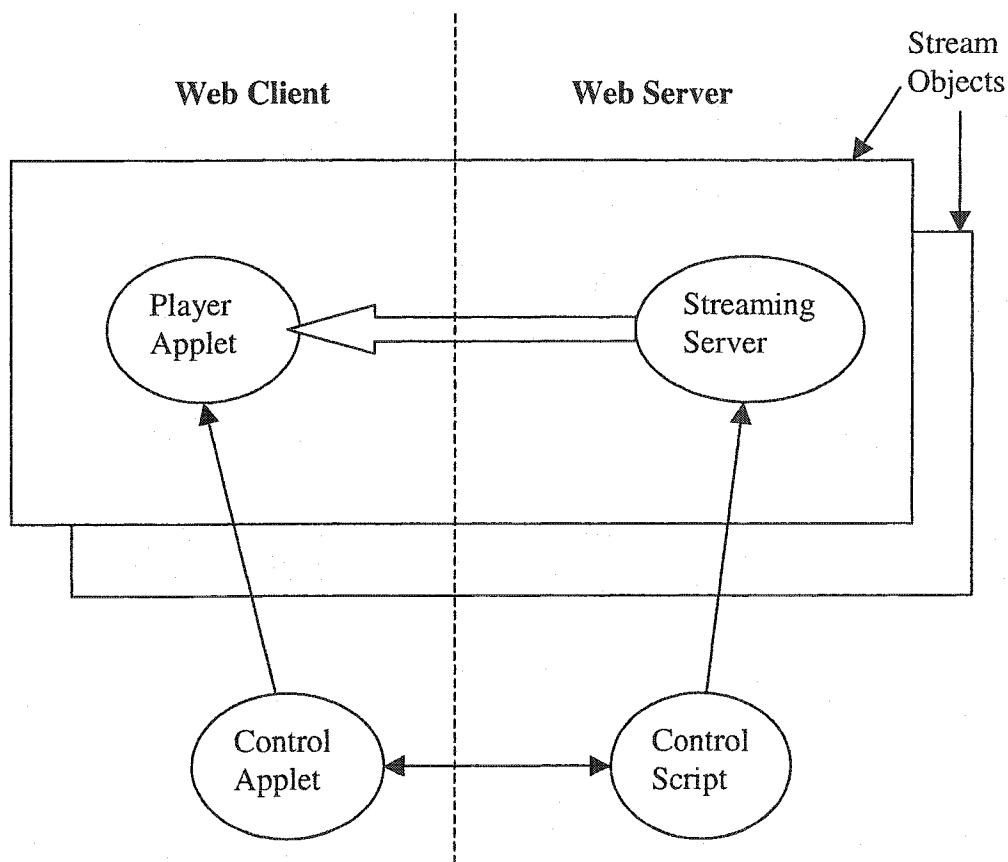


Fig. 7. The playback architecture.

5.3.4 Playback architecture

The subject of this section is how the actual stream playback occurs. Fig. 7 shows the architecture of the playback of selected streams.

As soon as the user hits the “playback” button on the playback form shown when an object is selected (see last section for details), the object display method calls a control script that is not part of any specific object. This script runs on the web server. It starts the streaming servers, described in Section 5.1.1, in the stream objects of all the selected streams. It also generates an HTML page that contains a control applet for each set of synchronous streams and a player applet for each of the selected streams. The control applet should contain buttons such as “start”, “stop”, “forward”, “rewind”, “pause”, and “resume”. Whenever one of these buttons is pressed, the appropriate command should be passed to all the player applets, which in turn should communicate with their corresponding streaming servers. The technical details of how this is implemented are shown in Section 6 “Implementation details”.

5.4 GENERAL LIBRARY ARCHITECTURE – A SUMMARY

This section shows the overall architecture of the digital library. It serves as a recap of all the details that the several last sections deal with. The overall architecture of the digital library is shown in Fig. 8. The various roles of users are shown in bold at the top and the bottom of the figure. The streams will be published into the digital library using a publishing tool, which will be used by a publisher to specify the metadata of the stream and to build the stream object. The publishing tool will call indexing services to add any new stream metadata information to the current metadata of the system. An expert user, or what we call a domain expert, will use an event generation utility to

generate the events of the streams in the digital library. The new events will be inserted in the event objects repository.

The scenario for the resource discovery user to retrieve an object from the library is as follows:

1. The user will use a search form to specify the search criteria. These search criteria will be transferred via the Web to a search script. This script will contact the database server to search the system metadata and return the search results.
2. The search results, the hits, of the search will be presented to the end user.
3. From these search results, the user will choose an object to display. The search script will get this information and look for that object in the object repositories (stream objects and event objects). The selected object will be displayed to the user. A metadata list of that object will be presented followed by a playback form. If the selected object is a stream object, then the playback form will include a list of all the events that occurred during the stream, a list of time instances of the selected event, and a list of the related streams. If the selected object is an event object, then the playback form will include a list of time instances of that event and a list of the related streams.
4. The user will specify the set of streams he wants to playback as well as the starting time from which these streams should start playing. This information will be passed to a control panel script that will start the stream servers and the stream player applets. The servers will provide the players with the stream data.

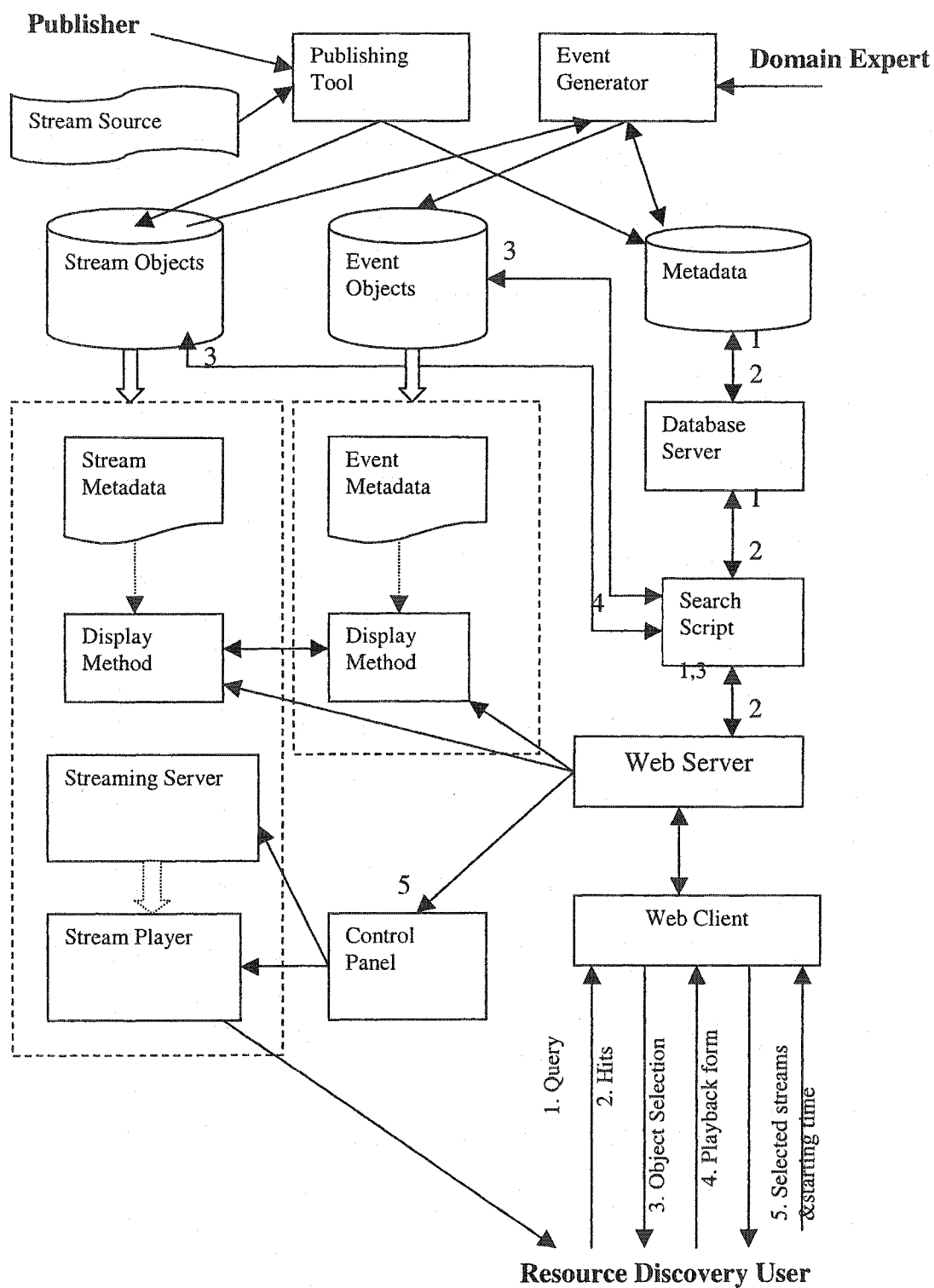


Fig. 8. The overall architecture of the digital library.

SECTION 6

A CASE STUDY: STREAMS IN A MEDICAL DIGITAL LIBRARY

The focus of Section 5 is on the architecture of any potential digital library that contains data streams and enables event-based retrieval. The major issues of this architecture discussed in that section concern the contents of the library and how they are organized, the metadata to be used to facilitate the search, retrieval and playback of this data, the software modules that are needed to enable the access of the contents of the digital library, and the interaction between the various users of the digital library and its utilities.

This section mainly revolves around a case study. We actually implement a digital library whose primary streams are Computed Tomography (CT) scans of patients who have liver tumors. Other contents of the digital library include audio streams and text descriptions of the CT scans and the events occurring during these scans. Section 6 presents the details of creating the digital library, publishing stream data and generating the metadata, the search engine, navigating the result set and the playback of the streams.

CT imaging, also known as "CAT scanning" (Computed Axial Tomography), was developed in the early to mid 1970s and is now available at tens of thousands of locations throughout the world. CT is fast, patient friendly and has the unique ability to image a combination of soft tissue, bone, and blood vessels. CT is the workhorse imaging system in most busy radiology departments and diagnostic centers. Since its invention, CT imaging has seen massive advances in technology and clinical performance. Today CT enables the diagnosis of a wider array of illness and injury than ever before [46].

Tumors are abnormal masses of tissue that form when cells begin to reproduce at an increased rate. The liver can grow both non-cancerous (benign) and cancerous (malignant) tumors. CT scanning is used in the diagnosis of both types of liver tumors.

This section covers all the details of the actual implementation of the case study. It presents the data types used in the digital library, the source of the streams used in the study, how these streams are organized in the library, the metadata used, and how the streams are published. It also covers the search and retrieval of the streams.

The section is organized as follows: Section 6.1 describes the details of the data stored in the digital library, including how they are obtained, what their raw format is, and how they are prepared for organization in the digital library. Section 6.2 describes how the data are organized in the library: both on the file server and on the database server. Section 6.3 describes data publishing, whether mass publishing or individual streams publishing. Section 6.4 presents what little we do on event generation for this DL. Section 6.5 shows how streams are related to each other. Section 6.6 describes the search interfaces: simple, event-based, and browsing. Section 6.7 deals with the stream display interface. Section 6.8 describes the stream playback interface. Section 6.9 shows that the library can accommodate new stream types by describing how streams of a new type can be added to the library. We end the section with a summary in Section 6.10. A brief summary of this implementation is presented in [49].

6.1 DATA SOURCE

The contents of the digital library are of three types: CT scans, audio descriptions, and text descriptions. In this section we specify how this data is obtained and prepared for organization in our digital library.

6.1.1 CT scan streams

The CT scans are obtained from a radiologist who is using these scans to do research on liver tumors. The system that is used to generate these scans has a specific file format for the images and special metadata architecture. Every patient is assigned a unique id number (basically his social security number unless not available) and a system folder is created for this patient. In each patient folder, a sub-folder called "ctscan" contains all the CT scans information of this patient. In the "ctscan" folder, a set of sub-folders contains all the patient visits information. Unique id numbers are assigned to each visit folder. In each visit, the patient usually goes through 4 or 5 films. The first film is called a scout film, which gives a horizontal view of the entire body and that is usually one image. The second film is a CT scan, which the patient goes through without any injections and is called a "non-contrast" scan. Injections are usually given to patients to cause contrast in the images for better identifications of organs and any existing tumors. The third film is usually taken after the injection of this contrast material and while it is still in the arteries, and that is why it is called "arterial". The fourth film is called "portal venus" and there might be a fifth called "delayed," all depending on the timing after injecting the contrast material. For each of these 5 films, there is a sub-folder in the visit folder that contains all the images of this film along with all the metadata files needed to view the film. Except for the first film, the scout film, all of these scans consist of a large number of images, usually in the range from 40 to 60 images. The images in this system are stored in two formats: Dicom (.DCM) format for the whole image in its original size and with all details stored, and a GIF format for a half size image used primarily as a thumbnail. For each of the images, there is a corresponding text file that contains

metadata information about the patient, the visit, this particular scan, and this particular image. There is redundancy in the metadata associated with each image since all the information, except for the image specific information, is repeated for all images. An example of these text metadata files for a random image is shown below. The patient's and doctor's names, plus any possible ID information are omitted:

```
ID Image Type=ORIGINAL\PRIMARY\AXIAL
ID SOP Class UID=XXXXXXXXXXXXXXX
ID SOP Instance UID=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ID Study Date=19990611
ID Series Date=19990611
ID Acquisition Date=19990611
ID Image Date=19990611
ID Study Time=150617.000000
ID Series Time=151023
ID Acquisition Time=151046
ID Image Time=151023
ID Accession Number=3990319
ID Modality=CT
ID Manufacturer=GE MEDICAL SYSTEMS
ID Institution Name=CENTRAL IMAGING SERVICES INC.
ID Referring Physician's Name=XXXXXX^XXXXX^
ID Station Name=CT6XOC0
ID Study Description=ABDOMEN W
ID Series Description=ABD. W/O CONTRAST
ID Name of Physician(s) Reading Study=PAI
ID Operator's Name=JL
ID Manufacturer Model Name=GENESIS_HISPEED_RP
PAT Patient Name=XXXXXX^XXXXXX^X^^
PAT Patient ID=XXXXXXXXXX
PAT Patient Birthdate=19260616
PAT Patient Sex=F
PAT Patient Age=072Y
PAT Patient Weight=0.000000
PAT Additional Patient History=LIVER TUMOR
ACQ Contrast/Bolus Agent=NONE
ACQ Slice Thickness=7.000000
ACQ KVP=120
ACQ Data Collection Diameter=480.000000
ACQ Software Version=07
```

ACQ Reconstruction Diameter=340.0000000000
 ACQ Distance Source-Detector=1099.3100585938
 ACQ Distance Source-Patient=630.000000
 ACQ Gantry/Detector Tilt=0.000000
 ACQ Table Height=149.300003
 ACQ Rotation Direction=CW
 ACQ Exposure Time=1000
 ACQ X-ray Tube Current=240
 ACQ Filter Type, extremity=BODY FILTER
 ACQ Focal Spot=1.2 \1.2
 ACQ Convolution Kernel=SOFT
 ACQ Patient Position=FFS
 REL Study Instance
 UID=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 REL Series Instance
 UID=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 REL Study ID=XXXXXX
 REL Series Number=2
 REL Acquisition Number=1
 REL Image Number=1
 REL Patient Orientation=L \P
 REL Image Position Patient=-170.000000\ -170.000000\1.600000
 REL Image Orientation
 (Patient)=1.000000\0.000000\0.000000\0.000000\1.000000\0.000000
 REL Frame of Reference
 UID=1.2.840.113619.2.1.1.2702911929.643.929071451.540
 REL Laterality=
 REL Position Reference Indicator=XY
 REL Slice Location=1.5999999046
 Unknown group length=20 32
 IMG Samples Per Pixel=1 1
 IMG Photometric Interpretation=MONOCHROME2
 IMG Rows=200 512
 IMG Columns=200 512
 IMG Pixel Spacing=0.6640625000\0.6640625000
 IMG Bits Allocated=10 16
 IMG Bits Stored=10 16
 IMG High Bit=f 15
 IMG Pixel Representation=1 1
 IMG Smallest Image Pixel Value=0 0
 IMG Pixel Padding Value=IMG Rescale Intercept=-1024
 IMG Rescale Slope=1
 SDY Study Priority ID=MED
 SDY Study Reason= LIVER CT SCAN LIVER CA,RUQ PAIN HT:000.0
 ROOM 5 RIGHT SIDE PAIN WT:000.0 MD PHONE #:XXXX
 NAME:XXXX, XXXXX

SDY Requesting Physician =XXXXXX^XXXXXX^
 PXL Group Length=80008 524296
 PXL Pixel Data=Data on disk

23 patients' data are used. Each patient has 4 visits. In each visit, the patient goes through 4 or 5 scans as explained above. The scout film is not used because it is only one image. The delayed scan is not used either since it does not exist for many patients. The other 3 scans exist for almost every patient. We publish these scans in the digital library. As we describe in Section 6.3.1, not all the metadata similar to the example above are used. A publishing script, as explained later in this section, is used to extract selected fields. We drop the DCM (larger) images from our consideration due to the large storage requirement which was not available for our testbed. Considering that the CT scans will be streamed to viewers over network connections with different bandwidths, a practical consideration is to use images of moderate size and available viewers like the GIF ones.

6.1.2 Audio streams

The second type of data streams included in the digital library is audio. Audio streams are used to record oral notes doctors take when examining the CT scans. The proper name for these streams are "streams of audio clips" because they are not a single audio stream that describes a CT scan stream, rather a stream of audio clips where each clip relates to one or more images of the set of CT scan images. These streams are recorded specifically for the case study and are not actual data obtained from a production system like the CT scans. The audio format that is used in this case study is the WAV format recorded using the audio recorder that is part of the WINDOWS operating system accessories.

6.1.3 Text streams

The third type of data streams included in the digital library is text. This text is used to describe some of the events occurring at some point in the other streams in the library and serves the purpose of demonstrating event-related streams in our case study. Using audio and text streams in this digital library proves the concepts of the heterogeneity and the extensibility in our implementation. Later in this section, there is a description of the steps needed to insert additional types to the library.

6.2 DATA ORGANIZATION

A good organization for the data streams and their metadata information in the digital library is essential for the performance of the whole system. In this section, we describe how these streams, their bibliographic metadata, the events information, and the related streams information are organized in the file system and the database server.

6.2.1 File system organization

Stream objects

As described in Section 5, streams are organized into stream objects; each object contains the stream data, metadata and the binaries needed to access the stream. We have slightly changed the design presented in Section 5. Instead of having the streaming server, the stream player, and the stream display module in each stream object, these programs are moved out of the object and are now common to all objects of the same type. The reasons for these design changes are:

- *Saving process space:* Instead of creating a streaming server process for every stream object - and they can be numerous - only one streaming server process will be

running which would then use the appropriate streaming module for the particular stream type.

- *Saving disk space:* Most of these streaming servers, players, and display modules would have been the same. Instead of repeating the same set of programs in every stream object, only one set that serves all streams is kept.

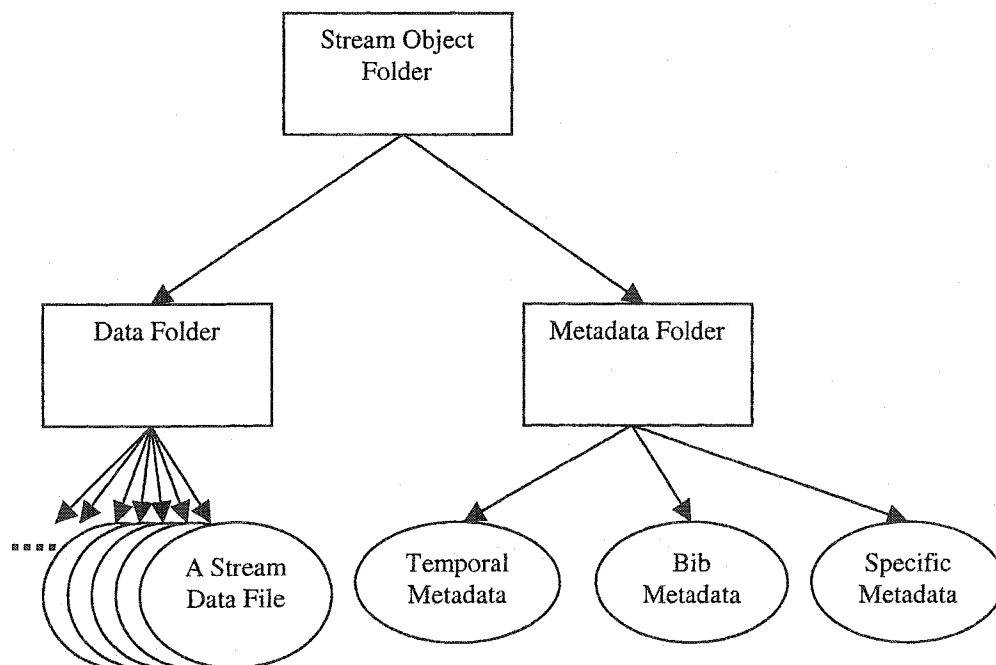


Fig. 9. Structure of the stream object in this case study.

Each of these processes is moved out of the stream object and is now common to all streams. Details for the streaming server, players, and the display method will follow in Section 6.8.

Structure of the stream object

The stream object now contains the stream data and metadata. As can be seen in Fig. 9, the stream object is a file system directory, or folder, that contains two main folders: data and metadata. The data folder contains all the data files of a particular stream, e.g. the entire image files of a CT scan stream, or all the audio clips for a stream of audio clips. The metadata folder contains the corresponding metadata files that are divided into three files: the temporal metadata file, the standard bibliographic metadata, and the specific bibliographic metadata.

The temporal metadata file

The temporal metadata file contains a listing of the timing of all the stream frames. The format of this file is simple; it consists of 2-field records. The first field is the offset in milliseconds from the start of the stream and the second one is the relative frame file path. The frame, as discussed before in Section 3.1, could be an image, a clip, a text string, etc depending on the stream type. The relative frame file path is appended to a pre-configured base URL to be used by the stream player to retrieve that frame.

The standard bibliographic metadata file

The second file in the metadata folder is the standard bibliographic metadata file, or the bib metadata file for short. This file contains general bibliographic metadata about the stream. This file is used for archival purposes, for backward compatibility, and for improving the portability of the stream object. The fields in this file are saved in the database. The metadata fields in this file are standard for all stream types. These fields are: the stream id, the entry date, the title, the creator, the start date, the end date, the

stream type, the stream format, the stream source, and the description. Most of these fields are similar to the corresponding Dublin Core fields [23]. The file format, however, follows the syntax of the RFC1807-metadata format [55]. An example of one of the files used in our implementation is shown below followed by an explanation of the bibliographic fields:

```

BIB-VERSION:: X-streams-1.0
ID:: patients2//p15.2
ENTRY:: 2002-06-06
TITLE:: CT scan stream for patient p15 for his visit on 19990608
CREATOR:: Mohamed Kholief
START_DATE:: 19990608
END_DATE:: 19990608
TYPE:: CT SCAN
FORMAT:: GIF
SOURCE:: Station name: CT1X_OC0
DESCRIPTION:: This is the CT scan stream for patient p15 for his visit on:
19990608. The study reason is: CT CHEST/ABD/PELVIS SCH W/FAITH 8-
3200--PREP 0730AM UPMC HEALTH PLAN. The study description is: CHEST
W & ABD W, and the series description is :
END:: patients2//p15.2

```

- *BIB-VERSION*: A mandatory field for RFC 1807. We will always assign the value "X-streams-1.0" to this field. The "X" indicates that this is an experimental bib file as required in RFC 1807.
- *ID*: This is the second field of any record. It is also a mandatory field. The ID field identifies the bibliographic record and is used in management of these records. Its format is "ID: XXX//YYY", where XXX in our implementation is a unique archive ID and YYY is an arbitrary ID of the stream object. The archive ID is a globally

unique URN. The reverse domain name will be used as a part of it to insure its uniqueness.

- *ENTRY*: This too is a mandatory field. It is the date of creating this bibliographic record. The format for ENTRY date in this implementation is "yyyy-mm-dd", e.g. "1999-01-02".
- *TITLE*: This is the title of the stream.
- *TYPE*: In the RFC 1807 memo, this field indicates the type of publication (summary, final project report, etc.) as assigned by the issuing organization. In this implementation we use this field to indicate the stream type, e.g. CT SCAN, audio lecture, etc.
- *CREATOR*: From the Dublin Core specifications, the creator is the person or organization primarily responsible for creating the intellectual content of the resource, for example, authors in the case of written documents, artists, photographers, or illustrators in the case of visual resources. In this application, the same definition could still work with minor semantic differences depending on the stream type: for a lecture for example, the creator is the teacher and for the CT scan images, the creator could be the radiologist.
- *DESCRIPTION*: A textual description of the content of the resource, including abstracts in the case of document-like objects or content descriptions in the case of visual resources.
- *FORMAT*: The data format of the resource is used to identify the software and possibly hardware that might be needed to display or operate the resource.
- *SOURCE*: Information about where the stream is obtained.

- *START_DATE*: The stream's start date. The format is *yyyymmdd*.
- *END_DATE*: The stream end date. The format is the same as the start date. In a future extension to this system, the start and end times will also be included.

The specific bibliographic metadata

To make the system heterogeneous, with regard to the type of the streams that could be saved in the digital library, another metadata file, the specific metadata file, is needed to save those particular metadata fields that are specific to a given type. For a medical library as the one used in this case study, users may want to retrieve the information for particular patients, particular dates of birth, etc. This information cannot be saved in the general bibliographic metadata file since it is not applicable to other stream types and so as not to inflate that file. The specific bibliographic metadata file is used to maintain these fields. The fields in this file will depend on the stream type. This file in itself is optional. A stream type may not need to use bib fields other than those used in the general bib metadata file. As with the general bib file, this file is now redundant and all its contents are also maintained in the database. The structure of this file follows the same format as with the general bib file, following the syntax of the RFC1807 metadata format. For the CT scan streams, the following specific metadata fields are used based on the information taken from a radiologist concerning his needs: patient name, birth date, age, sex, history, slice thickness, contrast, and reason.

Stream archives

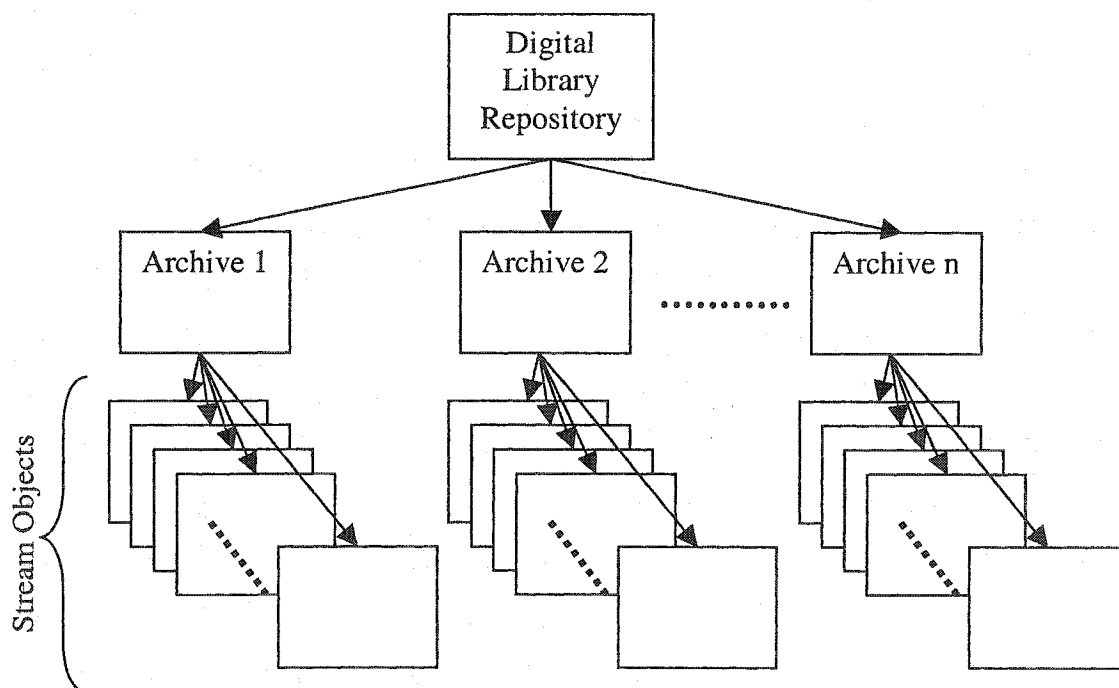


Fig. 10. Structure of the digital library repository.

The digital library may use different archives to save the stream objects as appears in records for patients from one clinic per archive. The archive name is part of the stream ID as mentioned in the last section. Streams, coming from different sources, may be saved into different archives. In this implementation, it is not necessary to save streams coming from the same source into the same archive. Fig. 10 shows the digital library repository. In the file system design, this repository folder is called “Archives”, and within this folder each archive will be saved in a separate folder. Inside each archive folder the stream objects are stored in this archive.

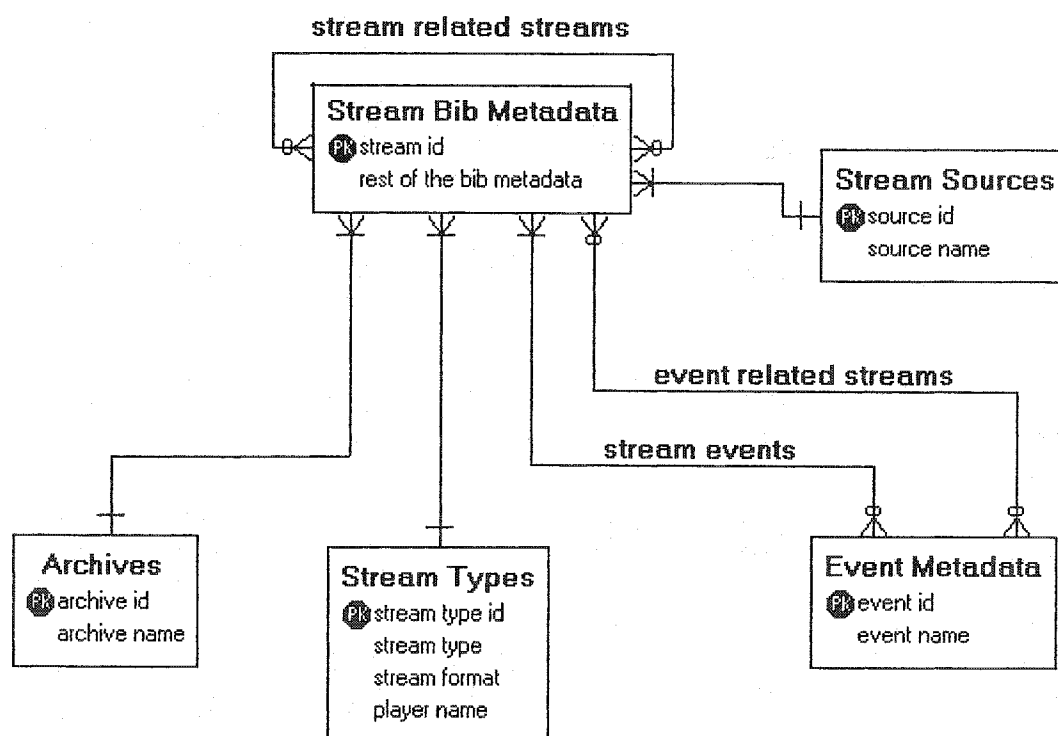


Fig. 11. The ER diagram of the database.

6.2.2 Database server organization

A database system is used to facilitate the retrieval from the digital library. Using a production database management system like Oracle [9] or MySQL [69] insures a fast and reliable search and retrieval of the stream objects.

The following information is kept in the database:

- All the stream standard and specific bibliographic metadata information, thus stream objects could be retrieved based on any bib metadata field.
- All the events information, which enables the retrieval of streams based on events that occurred during these streams, which is the main point in this research.
- Information about related streams.

The search engine uses the database to retrieve any stream object based on its bibliographic information or based on the events that occur during this stream. Playback scripts use the database to decide on the related streams that will be played back simultaneous to the retrieved stream, and to decide on the playback start time based on the time of any event instance.

A simplified entity relationship (ER) diagram is shown in Fig. 11. Because of its availability, an Oracle [9] database is used. The ER diagram is converted to tables and normalized. Some of the important tables are:

- Stream bib metadata: contains all the standard bib information, foreign keys to all referenced tables such as the types and archives tables, and a field that contains a concatenation of all the specific bib metadata fields. The primary key of this table is the stream ID.
- Event metadata: just contains the event ID and the event name in this implementation. In a future extension to the system it can also include other event bib metadata.
- Stream events: used to resolve the many-to-many relationship between the streams and events tables. This table has the stream ID and the event ID combined as the primary key. The event instance time is the only other field.
- Stream-related streams: contains the IDs of the primary and related streams. It also contains information on whether these streams are synchronous or not.
- Event-related streams: contains the event ID and the IDs of its related streams. This table will be used in a future extension since the concentration in the current implementation is on stream-related streams.

- Streams types: contains information about all the stream types supported by the digital library. For each type, the type ID, the type name, the format of streams of this type, and the class name of the Java applet used to play the streams of this type are included in this table.

6.3 DATA PUBLISHING

Stream publishing involves the tasks of creating the stream object into a specific archive folder, copying the stream data files from the stream source to the appropriate file system folder, creating the stream metadata files in the appropriate file system folder, and updating the database tables to include the metadata information of this stream.

Publishing is one of two types: mass publishing or one-at-a-time publishing. Mass publishing is used when many streams are available at once from their stream source. This kind of publishing includes the case of publishing old streams or streams used in other systems. Publishing one stream at a time is used to add individual streams to the digital library and is usually used for experimental reasons or for other digital library systems that mainly store documents (e.g. research papers).

As mentioned in Section 6.1, in this implementation, data are obtained from different sources. Streams of the text and audio types are created manually. The CT scan streams are real data obtained from an actual hospital repository. They are relatively much larger in quantity than the streams of other types. These data could not possibly be published manually in the digital library. The data used in this case study are about 23 patients only. Assuming that the digital library will be expanded to contain information about many more patients highlights the need to the automation of the publishing process.

The following subsections describe how streams of each type are published in the digital library.

6.3.1 Mass publishing of CT scan streams

One major PERL [86] script and few small scripts are used to automate the publishing of the CT scan streams. The bib metadata generation script, which is the major script, is described here. The script functionality is to:

1. Create the stream object into a specific archive folder, given as a parameter.
2. Copy the stream data files from the stream source to the appropriate file system folder, which is the data folder of the stream object created in the last step.
3. Create the stream metadata files in the appropriate file system folder, which is the metadata folder of the stream object created in step 1.
4. Update the database tables to include the metadata information of this stream.

As mentioned in step 3 above, this script is used to generate the bibliographic metadata files, whether standard or specific, from the CT scan native metadata files shown in the example in Section 6.1.1. Not all fields are necessary for our experiment. Many of the fields in that table are specific for the software used to capture and playback these streams in the hospital. An informal discussion with few radiologists further reduced the number of fields that are selected for use in the specific bibliographic metadata fields. Standard bib metadata fields are pre-decided as discussed in Section 6.2.1. The main task of the metadata generation script is to map some of the native metadata fields to the standard and specific bibliographic metadata fields that we decided to use in this system and to add these metadata fields to the appropriate database tables. The fields in the standard bib metadata file are mapped as follows:

- *ID*: is the concatenation of the archive name and an automatically generated stream ID separated by “//”.
- *ENTRY*: the system date is assigned to this field.
- *TITLE*: the phrase “CT scan stream for patient <patient name> for his visit on <ID_Series_Date>”, where <patient name> is an automatically generated virtual name of the patient (to suppress the true identity of patients) and <ID_Series_Date> is the visit date obtained from the raw metadata files for this stream.
- *CREATOR*: a randomly selected creator name from a set of pre-defined imaginary names, again to suppress any identities.
- *START_DATE*: <ID_Series_Date>, which is the visit date, obtained from the raw metadata files for this stream.
- *END_DATE*: same as the start date for this particular stream type.
- *TYPE*: CT SCAN.
- *FORMAT*: GIF
- *SOURCE*: the phrase “Station name: <ID_Station_Name>”, where <ID_Station_Name> is obtained from the raw metadata file and indicates the name assigned to the machine used to obtain this CT scan.
- *DESCRIPTION*: the phrase “This is the CT scan stream for patient <patient name> for his visit on: <ID_Series_Date>. The study reason is: <SDY_Study_Reason>. The study description is: <ID_Study_Description> and the series description is <ID_Series_Description>”, where all the fields enclosed in angled brackets are metadata fields obtained from the raw metadata files of the stream.

As for the specific bibliographic metadata fields for the streams of the CT scan type, the fields are mapped as follows:

- *PATIENT_NAME*: patient name, an automatically generated imaginary name as described before
- *BIRTH_DATE*: <PAT_Patient_Birthdate>, obtained from the raw stream metadata.
- *AGE*: <PAT_Patient_Age>
- *SEX*: <PAT_Patient_Sex>
- *HISTORY*: <PAT_Additional_Patient_History>
- *SLICE_THICKNESS*: <ACQ_Slice_Thickness>
- *CONTRAST*: <ACQ_Contrast/Bolus_Agent>
- *REASON*: <SDY_Study_Reason>

All the fields in the angled brackets are obtained from the raw metadata files of the stream. The selection of these fields and the mapping decisions are all made based on the needs of radiologists doing research in the “Liver Tumors” issue as stated by some of them who are informally interviewed.

6.3.2 Individual stream publishing for other types

The audio and text streams are created manually as mentioned earlier. They are actually created “in place,” so the publishing process itself is done manually. A future extension to this work is to create a publishing tool that would take the user through a wizard to help her publish the stream without actually having to know the underlying file system or database details. This tool will ask the user to input the necessary metadata information and then create all the necessary files transparently. It will also upload all the data files of the stream into the appropriate file system folders. An advanced feature of

this tool would be to let the user decide and input the field names of the specific bibliographic metadata fields.

6.4 EVENT GENERATION

Searching for data streams based on the events that occurred during them is the main topic in this research. Many issues relate to the use of events in retrieval. In this section, each of these issues is examined. The main goal is to describe how each of them is handled in the current implementation and how it could be handled in any future extension to this system.

6.4.1 *Event specification*

Event names are specified in the event metadata database table as described in Section 6.2.2. In this implementation, events are inserted in this table manually. In other words, direct database instructions are used to insert the event names. This table mainly has an event ID and an event name fields. The domain expert is responsible for inserting the new events in this table (via the system administrators). In a future extension to this implementation, an event editor will be used to edit and insert these events in the database without having to know explicitly the details of the underlying database design.

Based on the opinions of radiologists who work in the field of Liver Tumors, the following events are used for this particular case study. Adding new events is very easy and does not require any changes in any implemented software.

- Calcification
- Tumor started to appear
- Maximum diameter of the tumor

- Necrosis appearing
- Maximum necrosis diameter

All of the above events are related to the liver tumors research topic and is used by the doctors to monitor the development of the liver tumors patient who is being treated.

6.4.2 Event instances generation

Events may occur in many streams and may occur many times in the same stream. Each of the occurrences of the event is called an event instance. Event generation is mainly applying the event criteria to the streams and generating a list of the times at which the event occurs. This process can be automated for some stream types and can be done manually for others. Section 2 gives a few examples of cases for which automatic event-generation could work. For CT scan streams, the dominant stream type in this digital library, all suggested events as discussed in the last section require sophisticated pattern recognition algorithms in the CT scan images. Since adopting any pattern recognition algorithm would have been a huge distraction from the original research, this is left for a future extension to the system or when it is further developed for practical, non experimental, applications. The main goal of this research is to study and prove the feasibility of using events for retrieval from a digital library containing data streams. A prototype of an event editor that assumes automatic event generation is described in the next section.

For the above reasons, the event instances' times are manually inserted into the stream events database table that has three fields: the stream ID, the event ID, and the event instance time. The event instance time is the offset from the start time of the stream

in milliseconds. If the event has more than one instance in the same stream, there will be many rows in this table with the same stream ID and event ID but with a different event instance time.

6.4.3 A prototype event generation tool

In a future extension to this system, an event-generation tool will be used to automate the event generation. The main functionality of this tool will be to:

- Read the event name and the IDs of the streams affected by this event from the user.
- Apply a suitable ready-made criteria module to the stream to generate the time instances of this event.
- Read the event-related streams from the user, or apply another criteria module to decide which streams relate to this event.
- Insert the event name, time instances, and related streams in the appropriate database tables.

6.5 RELATING STREAMS

One important feature of this research is the ability to playback related streams along with the originally retrieved stream. Deciding on what the related streams are and how they are maintained in the digital library is the focus of this section.

6.5.1 Related streams representation

In this implementation, streams could be related to streams or to events that occurred during these streams. As mentioned in Section 6.2.2, two database tables are used to maintain streams that relate to other streams and streams that relate to events. The fields in the stream-related table are the original stream ID, the related stream ID, why

they are related, whether they are synchronous or not, and the time offset in milliseconds between the two streams if they are synchronous, assuming that one of them may start some time after the other. For event-related streams, the event ID and the stream ID are the two fields in the table. Event-related streams are by default non-synchronous with the streams during which this event occurred.

6.5.2 Related streams specification

An important question in the digital library implementation is how to specify the related streams. One alternative is to let a domain expert specify, manually, the streams related to each stream to be published in the library. The more realistic alternative, however, is to use a generation script that applies a certain criteria on all streams for automatic generation of related streams.

In this implementation, both ways are used. A generation script is used for one criterion and it could be generalized to accommodate for other criteria. A simple script is used to relate streams of the same patient. Manual specification is used to relate some streams by just inserting the relationship information in the appropriate database tables as described above.

6.6 THE SEARCH INTERFACE

Four different search interfaces are proposed in the digital library architecture shown in Section 5. The simple interface enables end-users to locate stream objects by using words that appear in any field of its bibliographic metadata. The event-based interface enables users to locate streams based on events that occurred during these events. The advanced interface would enable users to locate streams based on a

combination of the values of specific bibliographic fields and events that occurred within these fields. The browse interface enables users to go through all the contents of the digital library.

In this case study, the simple, the event-based, and the browse interfaces are implemented. The advanced search interface is postponed to a future extension to the system since this is an experimental system and it does not add anything to the proof of the concept behind this work.

The search engine is a modification of the Arc engine [59]. Arc is an experimental research service of Digital Library Research group at Old Dominion University. Arc is used to investigate issues in harvesting OAI (Open Archives Initiative) compliant repositories and making them accessible through a unified search interface. Arc is used because of several reasons, the most important of which is the familiarity with the code as a member of the research group and as a participant in the implementation of its predecessor, UPS [19]. Another reason is its usage of current and efficient technologies for implementation, e.g. JAVA Servlets [42], JDBC [97], JSP [42], and the Oracle database management system [9]. The third reason is that it is an implementation of the OAI [76], which could be a good extension to this research. A future extension of this system is to use more than one repository and to make all of them OAI-compliant. Implementing the same harvesting techniques used in Arc with minor modifications will enable the search engine to retrieve streams from all of those repositories.

6.6.1 Simple search

The screenshot shows a web browser window titled "Stream Based Digital Library - Microsoft Internet Explorer provided by AT&T W...". The address bar shows "http://". The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar contains Back, Forward, Home, Search, Favorites, Media, and a Go button. The Links bar shows Hotmail, Media, CITI, PNC, PhD, and CUP EMAIL.

The main content area features the SBIDL logo on the left and the text "Stream Based Digital Library" on the right. Below this, there are links for [Simple Search](#), [Event-based Search](#), [Bibliographic Search](#), [Browse](#), and [Help](#).

The "Simple Search Form" section includes a "Search for" text input field, a "Group results by" dropdown menu set to "Archive", and a "Sort results by" dropdown menu. A "Start Searching" button is located below these fields.


At the bottom, it says "Powered by  Old Dominion University".

Fig. 12. The simple search interface.

The Simple search interface allows the user to locate data streams in the digital library based on the value of *any* bibliographic field. As can be seen in Fig. 12, the end-

user specifies the value he wants to search for and the search engine will try to match this value to the values in any bibliographic field. The results will be grouped by the archive name by default. Users can use the group by menu, shown in Fig. 13, to group the results by the values of a different field. Currently the interface supports grouping by the archive name, the stream source, or the stream type.

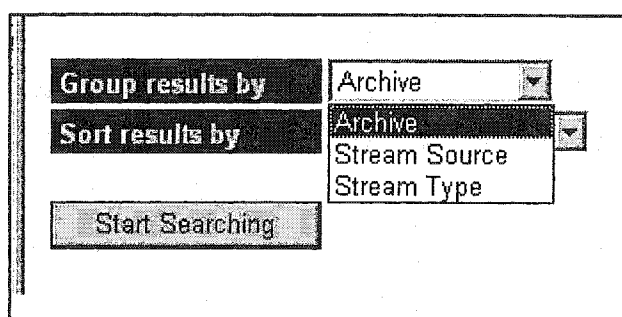


Fig. 13. The group-by menu.

The user can also sort the results in the ascending order of the values of either the title, the creator name, the stream start date, or the stream end date using the 'sort results by' menu shown in Fig. 14.

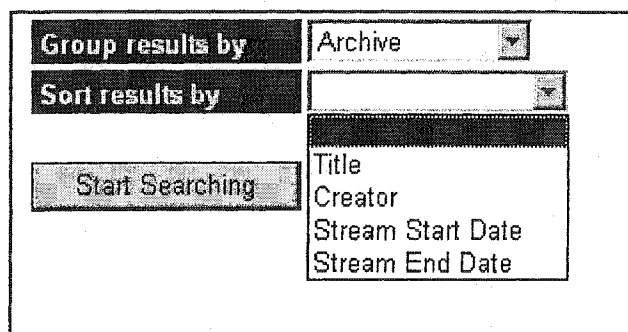


Fig. 14. The sort-by menu.

The search results are organized in the way seen in Fig. 15. The results frame is divided in two columns; the left one contains a list of all the groups that contain results, depending on the grouping field. In the figure, the results are grouped based on the stream type and all results come from one stream type, which is the CT SCAN type. Each group name is followed on the same line by the number of results (hits) from this group. The right column contains the details of the results. It starts with a search summary that specifies the search terms, how results are grouped and how they are sorted. The search summary is then followed by the search results. Results are arranged in pages. Each page contains 10 results and the user can switch between these pages by clicking on the page number. For each search result, minimal bib metadata are provided to give the user a chance to exclude irrelevant results without the need to open the stream. For each result, the stream title, the creator(s), the start date, the stream type, and the stream ID are displayed. The stream title is linked to the stream display interface to be discussed in the Section 6.7.

Simple Search Event-based Search **Bibliographic Search** Browse Help

Matches were found in these stream_types

stream_type	Hits
CT SCAN	151

Search Summary

- Search: *where upper(all_text) like '%TUMOR%' and stream_types.stream_type_id=bibmetadata.stream_type_id* Sort: *order by title*
- Group By: *stream_type* Value: *CT SCAN*

This is page 1, hits (1--10) of total 151 hits.

Results Pages: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#) [15](#) [16](#)

SEARCH RESULTS

Title	<u>Arterial CT scan stream for patient p1 for his visit on 01-Feb-1999</u>
Creators	Omar Kholief, Mariam Kholief
Start Date	1999-02-01 00:00:00.0
Type	CT SCAN
Stream ID	patients3//p1.2
Title	<u>Arterial CT scan stream for patient p1 for his visit on 16-Dec-1998</u>
Creators	Mohamed Kholief
Start Date	1998-12-16 00:00:00.0
Type	CT SCAN
Stream ID	patients3//p1.1

Fig. 15. The search results interface.

6.6.2 Event-based search

The screenshot shows a web browser window titled "Stream Based Digital Library - Microsoft Internet Explorer provided by AT&T W...". The address bar is empty. The menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar contains Back, Forward, Home, Search, Favorites, Media, and other icons. The main content area features the SBDL logo and the title "Stream Based Digital Library". Below the title are links for Simple Search, Event-based Search (highlighted), Bibliographic Search, Browse, and Help. The "Event-based Search Form" is displayed, consisting of several input fields and a search button. The "Event" field is set to "Any Event". The "Time Period From" and "To" fields are set to "Year", "Month", "Day", "HH", "MM", and "EST". The "Group results by" field is set to "Archive". The "Sort results by" field is empty. A "Start Searching" button is located below the form. At the bottom, it says "Powered by Old Dominion University".

Stream Based Digital Library

[Simple Search](#) [Event-based Search](#) [Bibliographic Search](#) [Browse](#) [Help](#)

Event-based Search Form

Event: Any Event

Time Period From: Year - Month - Day HH:MM EST

To: Year - Month - Day HH:MM EST

Group results by: Archive

Sort results by:

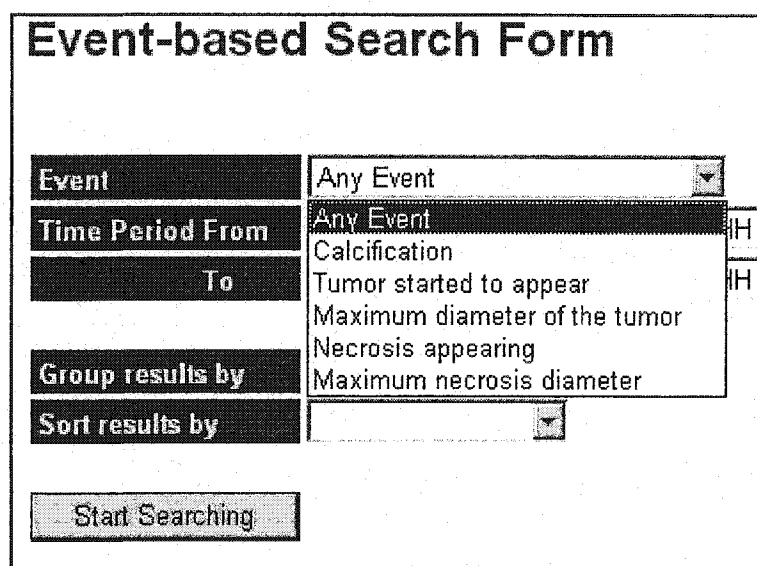
Powered by Old Dominion University

Fig. 16. The event-based search interface.

The event-based search interface is seen in Fig. 16. End-users look for events by their name using the events menu shown in Fig. 17. Currently, all events are read from

the events table in the database and displayed in one menu. In a future extension to this system, events will be grouped in tree-structured categories to speed up locating the required events. Another possible approach is to allow the user to type part of the event name and dynamically generate a menu that contains the events that include this part.

Users are also allowed to look for events that occurred during a specific time period. The “time period from” and “time period to” fields shown in Fig. 16 would enable the user to specify this period.



The image shows a software interface titled "Event-based Search Form". It contains several input fields and a button. The "Event" field has a dropdown menu currently showing "Any Event". The "Time Period From" and "To" fields also have dropdown menus, both currently showing "Any Event". The "Group results by" field has a dropdown menu showing a list of events: "Calcification", "Tumor started to appear", "Maximum diameter of the tumor", "Necrosis appearing", and "Maximum necrosis diameter". The "Sort results by" field has a dropdown menu that is currently empty. At the bottom of the form is a button labeled "Start Searching".

Event	Any Event
Time Period From	Any Event
To	Any Event
Group results by	Calcification Tumor started to appear Maximum diameter of the tumor Necrosis appearing Maximum necrosis diameter
Sort results by	
Start Searching	

Fig. 17. The events menu.

As in the simple search interface, users can specify a group-by value and an ordering value to group and sort the search results. The example in Fig. 18 shows the results of an event-based search for streams containing the event “Max. Tumor Diameter” grouped by the stream source with no sorting preferences.

Simple Search Event-based Search **Bibliographic Search** Browse Help

Matches were found in these stream_sources

stream_source	Hits
<u>Global CT Scan</u>	5
<u>Liver Tumor Patients</u>	3

Search Summary

- Search: where bibmetadata.stream_id=stream_events.stream_id and stream_events.event_id='event3' and start_date>='01-JAN-1995' and end_date<='31-DEC-2005' Sort;
- Group By: stream_source Value: Global CT Scan

SEARCH RESULTS

Title	<u>Portal Venous CT scan stream for patient p2 for his visit on 02-Feb-1999</u>
Creators	Kurt Maly
Start Date	1999-02-02 00:00:00.0
Type	CT SCAN
Stream ID	patients4/p2.1
Title	<u>Arterial CT scan stream for patient p2 for his visit on 06-Jul-1999</u>
Creators	Omar Kholief, Mariam Kholief
Start Date	1999-07-06 00:00:00.0

Fig. 18. Results of an event-based search.

6.6.3 Browsing

Browsing the digital library is just like doing a simple search with no specific search criteria. All streams are returned and grouped by archive with no sorting preferences. The results are shown in Fig. 19. The browsing capability is needed for those undetermined users and it provides a means for those users to see the contents of the library.

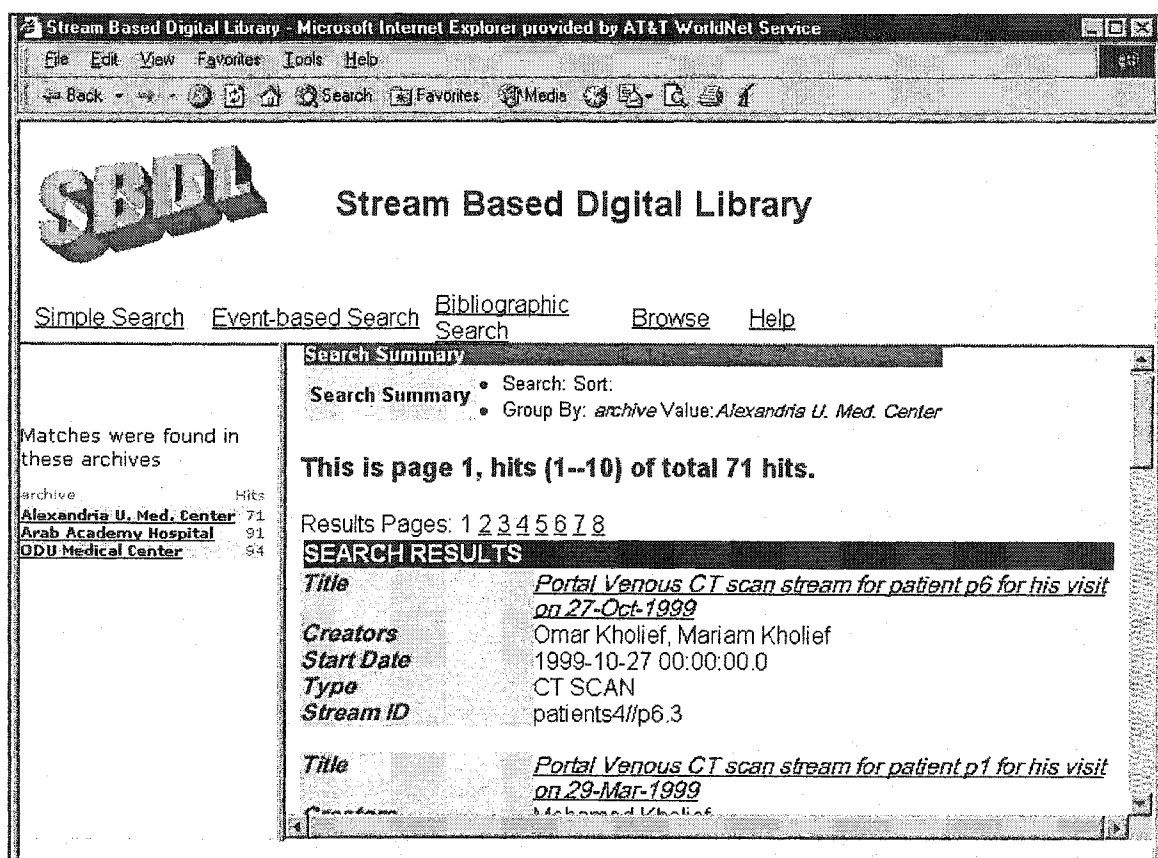


Fig. 19. Browsing the digital library.

6.7 THE STREAM DISPLAY INTERFACE

The previous section describes the search interface and how the search results are organized whether the user used the simple search, the event-based search, or the browsing capability. This section describes what goes next. In the results interface, minimal bibliographic information about the returned streams is displayed. Based on this minimal information, the user may decide to see more of the stream and play it back if desired. To reach more stream information the user should click on the stream title in the search results interface to see the stream display interface. This display interface will

open in a new browser window. An example of the stream display interface is shown in Fig. 20.

Bibliographic Information

General:		Type-specific:	
Title	Portal Venous CT scan stream for patient p2 for his visit on 02-Feb-1999	Date of Birth	19320718
Creator	Kurt Maly	Age	066Y
Stream Type	CT SCAN	Sex	F
Stream Source	Global CT Scan	History	LIVER MASS
Start Date	02-FEB-99	Slice Thickness	7.500000
End Date	02-FEB-99	Contrast	BARIUM & OPTIRAY 350
Description	This is the CT scan stream for patient p2 for his visit on: 02-Feb-1999. The study reason is: CT CHEST/ABD/PELVIS--PREP 1100AM SCH W/FAITH 8-3200--SLOT OK PER BEV MEDICARE A. The study description is: CH WOBABD W/WO, and the series description is:		
Stream ID	patients4/p2.1		
Reason	CT CHEST/ABD/PELVIS--PREP 1100AM SCH W/FAITH 8-3200--SLOT OK PER BEV MEDICARE A		

Playback Form

(Use this form to playback this stream and, if desired, any streams related to it. The playback starts from the beginning of the stream unless an event instance is specified as the starting time.)

Playback Start Time: (Select an event, and then select one instance of this event to start the playback from the time of that event instance)

Event:

Event Instances: Milliseconds from the stream start time.

Related Streams: (Select zero or more of the related streams to playback along with the original stream. Ctrl+click to select more than one stream.)

Related Synchronous Streams: (One control panel is used to control the original stream and any of these streams.)

Related Asynchronous Streams: (Each of these streams will have its own control panel and will start playing from its beginning.)

For Selected Streams, Display:

Fig. 20. The stream display interface.

This interface is divided into two sections. The upper section contains the complete stream bibliographic information and the lower section contains a playback form. The bibliographic information is divided into two columns. One of them contains the standard bibliographic information that is found in any stream object and the other contains the specific information that is optional and differs from one stream type to another. This interface is designed in such a way that automates the display of the

bibliographic information regardless of the used bib fields, which makes it easy for any future changes to the system.

The playback form is used if the user wants to playback this stream. There are two sections in this form. One section is concerned with determining the starting time for the playback and the other one is concerned with determining what other streams, if any, should be played back simultaneously. The user may need to play the stream starting from the time of an event instance. To do so, she can select an event from a menu of all the events that occurred during this stream. Since an event may occur many times during the same stream, the user can then select one time instance of the selected event from the time instances menu. In the case that the user originally is looking for a specific event using the event-based search form explained in the last section, this event will be pre-selected in the events menu and its first time instance will be pre-selected in the time instances menu.

Related streams are divided in two menus, synchronous related streams and asynchronous related streams. Synchronous streams are those streams that can playback synchronously with the original stream and the asynchronous streams are those that cannot. Both types are listed in list boxes that allow multiple selections, so the user can select as many streams as wanted to playback simultaneously with the original stream. The user can check the bibliographic information of the selected related streams by clicking on the "Bibliographic information" button. A window pops up in which the bibliographic information of all selected streams is displayed. Fig. 21 shows an example of such bib information window. As the figure shows, users are able to see further details

and a complete stream display interface of any of these selected related streams by clicking the button labeled “More details and playback form”.

The image shows a software interface with a main window and a popup window. The main window contains a form for stream playback with fields for Description, Stream ID, Playback Start Time, Event, Event Instances, and Related Streams. The popup window, titled "Bibliographic information of selected related streams", displays detailed information for two selected streams.

Main Window Fields:

- Description:** visit on: 02-Feb-1999. The study reason is: CT CHEST/ABD/PELVIS--P 3200--SLOT OK PER B description is: CH WO description is :
- Stream ID:** patients4//p2.1
- Playback Start Time:** (Select an event, Event: -Select an Event-----)
- Event Instances:** -Select Time Instance
- Related Streams:** (Select zero or more)
 - Related Synchronous Streams:** --Select Related Syn A mini-lecture about A mini audio lecture
 - Related Asynchronous Streams:** --Select Related Asy Portal Venous CT sc Portal Venous CT sc Arterial CT scan stre
- For Selected Streams, Display:**
 - ☐ Bibliographic information
 - ☐ Relationship details
 - ☐ Play

Popup Window: Bibliographic information of selected related streams

Bib informaion of the selected streams

Title	A mini-lecture about CT scans and Lever tumors
Creator	Mohamed Kholief
Stream Type	LECTURE TRANSCRIPT
Stream Source	Myself
Start Date	02-FEB-99
End Date	02-FEB-99
Description	A lecture about using CT scans in detecting Lever tumors and the progres in their treatment
Stream ID	patients4//p2.1text

More details and Playback form

Title	Portal Venous CT scan stream for patient p2 for his visit on 11-Oct-1999
Creator	Mohamed KholiefMohamed Kholief
Stream Type	CT SCAN
Stream Source	Lever Tumor Patients
Start Date	11-OCT-99

Fig. 21. Bibliographic information of selected streams appears in a popup window.

In the stream display interface shown in Fig. 20, the user is also able to see the details of the relationship between the selected related streams and the original stream by clicking on the “Relationship details” button. A window that shows the details of these relationships pops up when this button is clicked, as shown in Fig. 22. This window shows the ID and title of the original stream and a table showing the ID, title, and the relationship reason for all selected related streams.

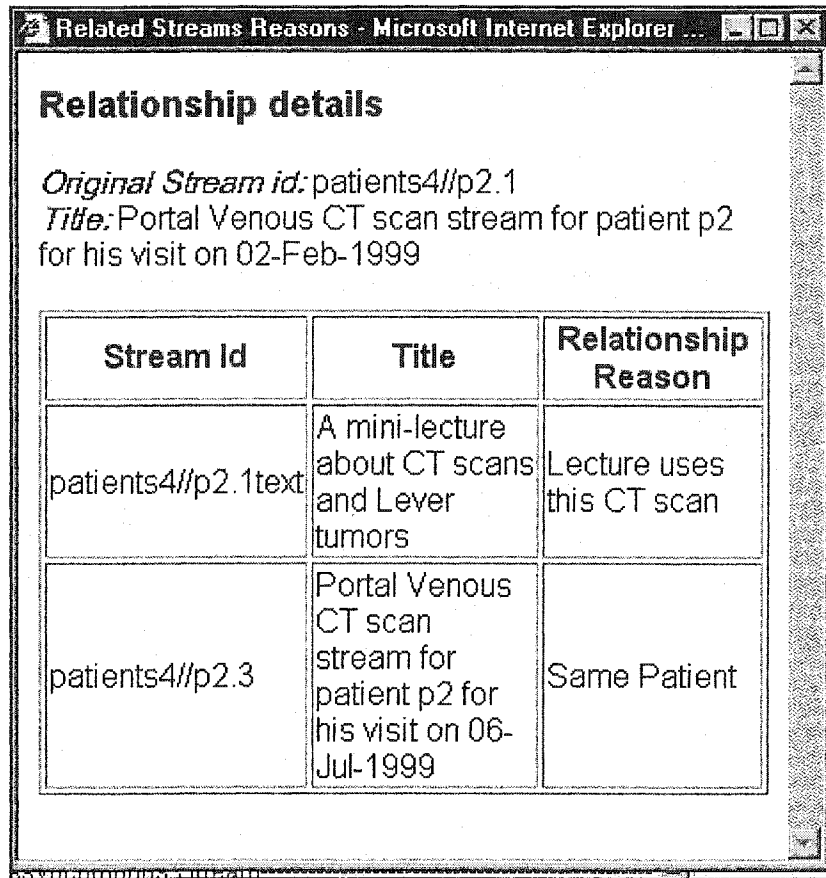


Fig. 22. Relationship details of selected streams appear in a popup window.

6.8 THE PLAYBACK SYSTEM

The last section describes the stream display interface. In this interface, the user can select the starting playback time and other related streams to be played back simultaneous to the original stream and hit the “play” button. Hitting the “play” button starts the playback interface described in this section. The stream playback involves stream player applets controlled by a control panel applet and a streaming server that sends the data of each stream to the corresponding player applet.

6.8.1 Description

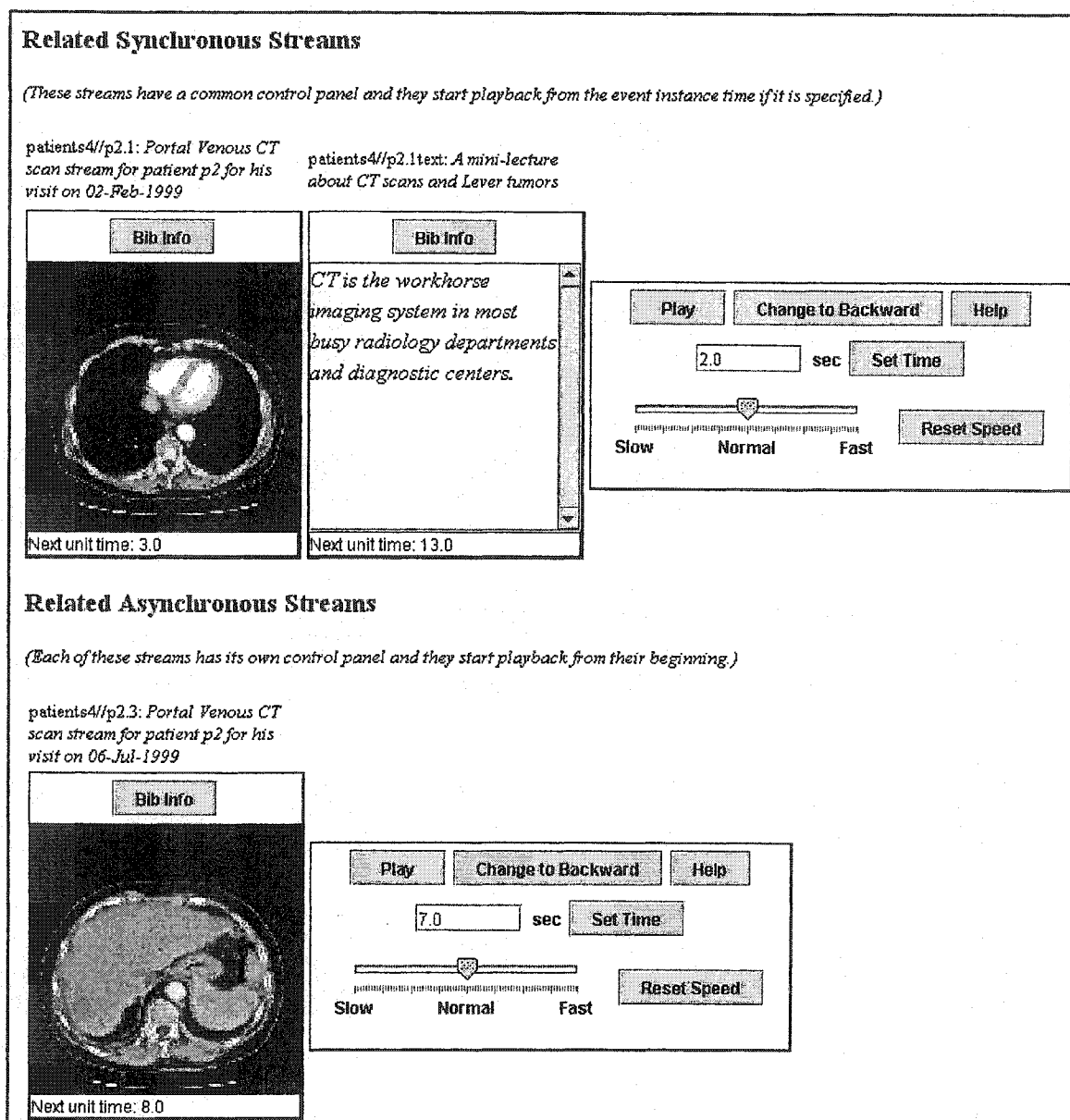


Fig. 23. An example playback interface with two synchronous and one asynchronous stream.

As can be seen in Fig. 23, the playback interface consists of two main sections: the upper section contains the players of the related synchronous streams and one control panel to manage all these players and the lower section contains the players of the related

asynchronous streams with one control panel for each. The leftmost stream player in the upper section is the player for the original stream. The ID and the title of each stream are displayed above the player of that stream. If only one stream is to be played back, the player of that stream and a control panel will be displayed in the upper section as can be seen in Fig. 24. The figure shows a CT scan stream player applet. The following sections describe the details of the stream player applets and the control panel.

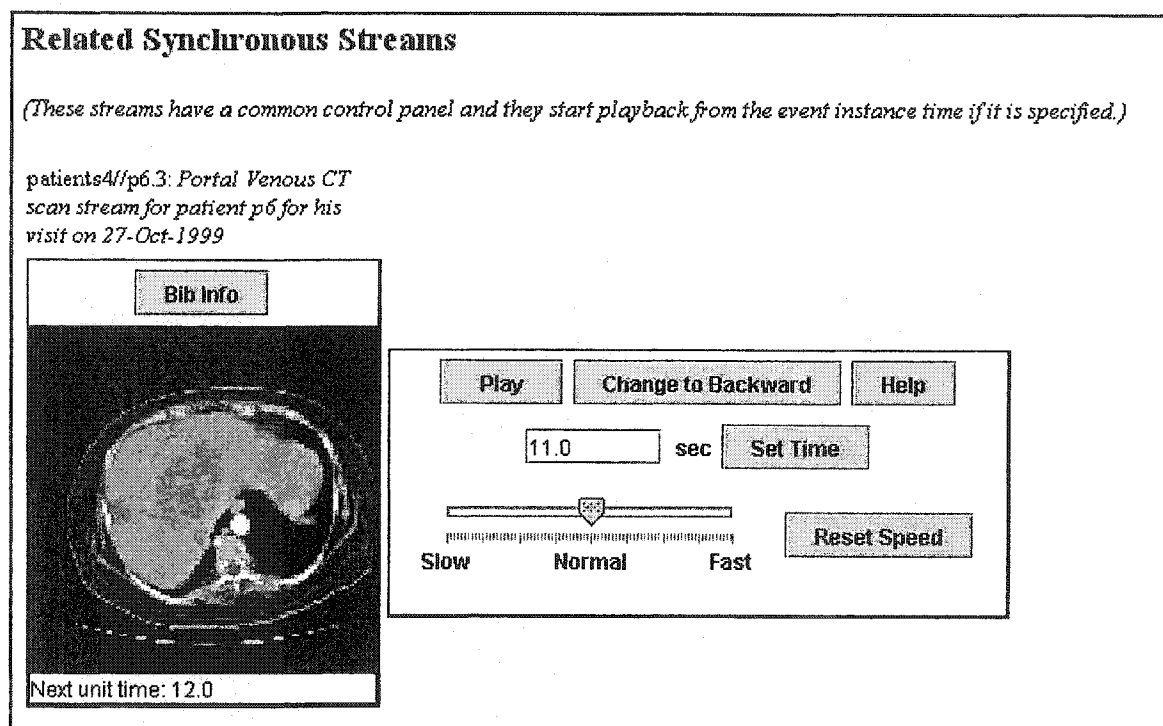


Fig. 24. One CT scan stream player.

6.8.2 The control panel applet

The control panel applet, the rightmost applet in Fig. 24, controls the behavior of the player applets. It contains the following interface elements:

- **Play:** starts the playback of the adjacent stream player(s) from the time specified in the current time text field (which shows under the play button in the figure). The label on this button is switched to stop when the stream is playing, as shown in Fig. 25.

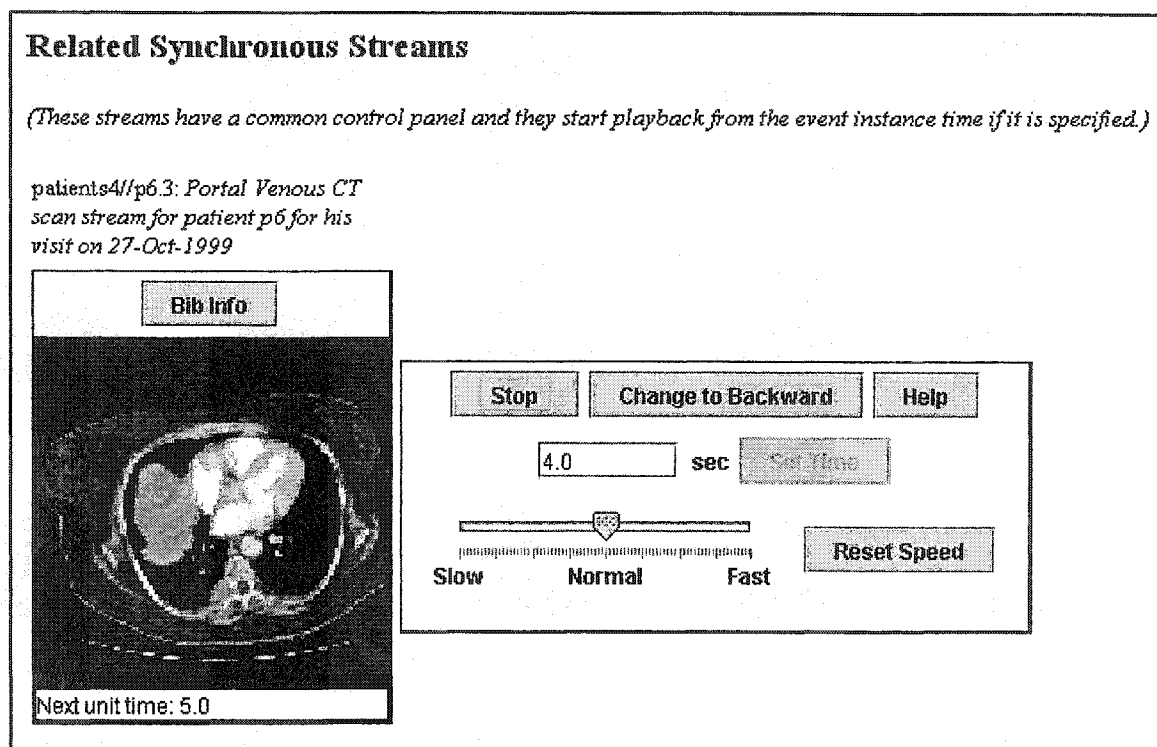


Fig. 25. A CT scan stream playing.

- **Stop:** stops the playback of the adjacent stream player(s). The current time will freeze so that if the “Play” button is clicked the stream just resumes playing back. The label on this button changes to “Play” when it is clicked.

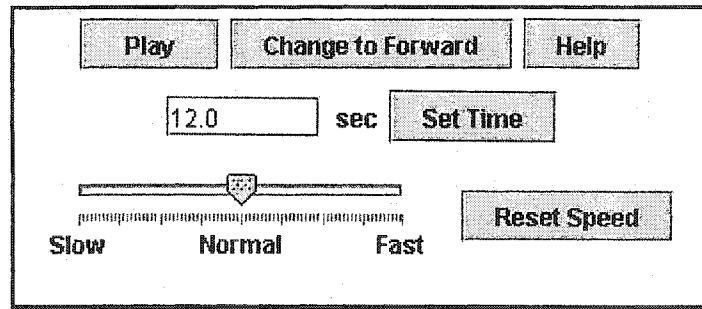


Fig. 26. The control panel applet shows that the stream is playing backward (the middle button label is “Change to Forward”).

- Change to Backward: streams start playing back in the forward direction by default. Clicking on this button will reverse the playback direction. The label changes to “Change to forward” when this button is clicked as seen in Fig. 26.

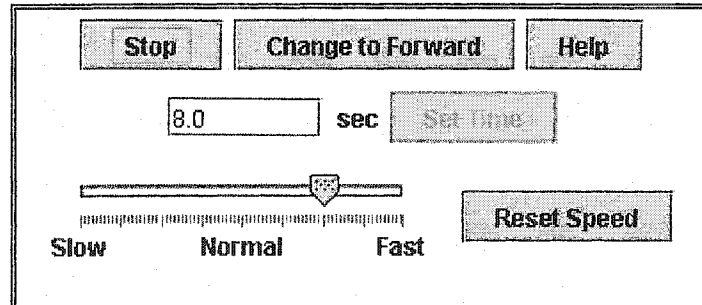


Fig. 27. The control panel applet shows that the player is playing faster than normal (see the slider at the bottom).

- Help: opens a small browser window that describes how the playback interface could be used.
- Current time text field: displays the time of the current frame of the leftmost player applet. The leftmost player is for the original stream and any other players are for

related streams. The current time is an offset from the start of the stream. The contents of this text field cannot be changed while the stream is playing. If the stream is stopped, the user can set a new time to start the playback from. (This is the text field containing 8.0 in Fig. 25).

- Set time: the user clicks this button after entering a new time offset in the current time text field. This button is inactive (dim) when the stream is playing.
- Playback speed slider: this slider bar allows the user to change the playback speed. Sliding the slider toward the “fast” label increases the speed and towards the “slow” label decreases the speed using a simple heuristic. See Fig. 27.
- Reset speed: clicking this button resets the stream playback rate to its normal speed. This button is significant since users would not normally be accurate enough to move the slider back exactly in the middle of the slider bar to reset the speed to normal. Audio streams, for example, will not play unless the stream plays at the exact normal playback rate.

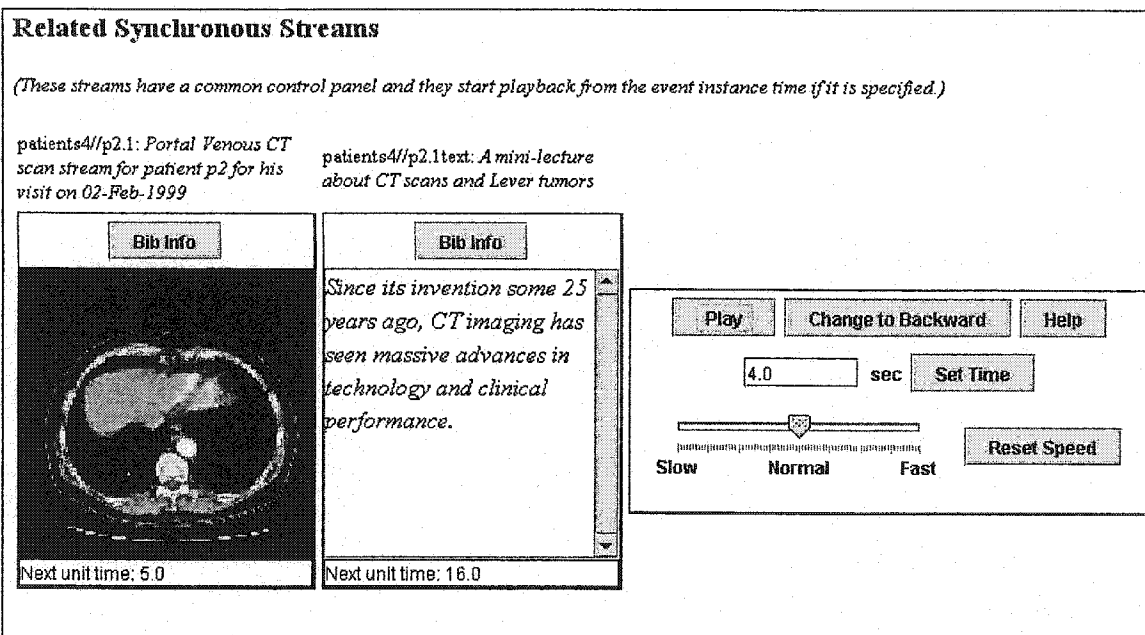


Fig. 28. Two synchronous streams: a CT scan streams and a text stream.

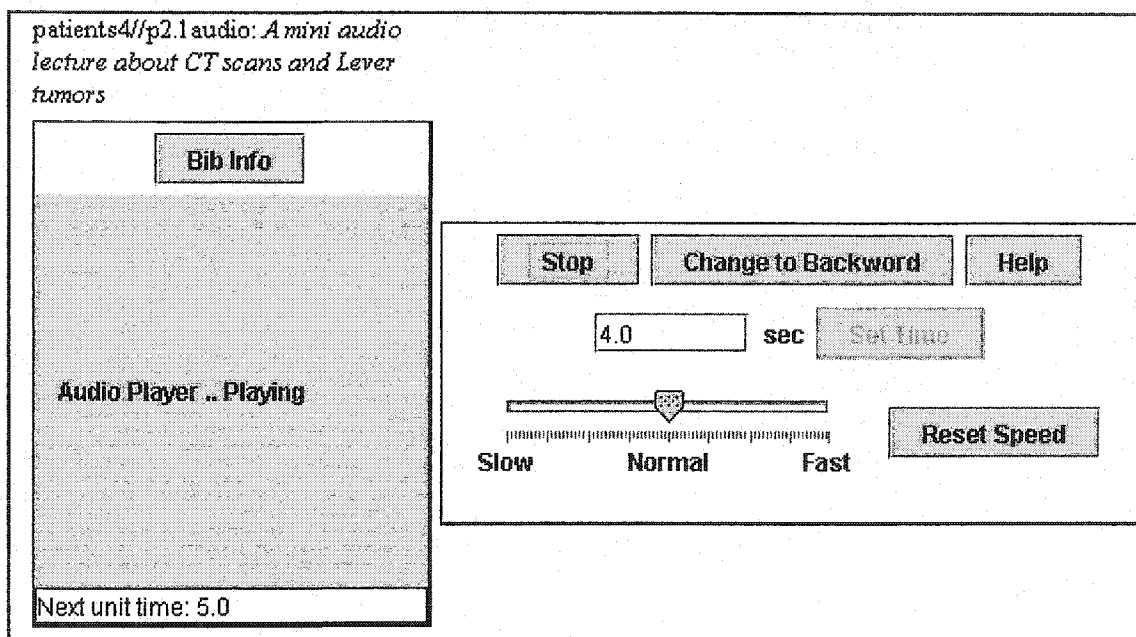


Fig. 29. An audio stream playing.

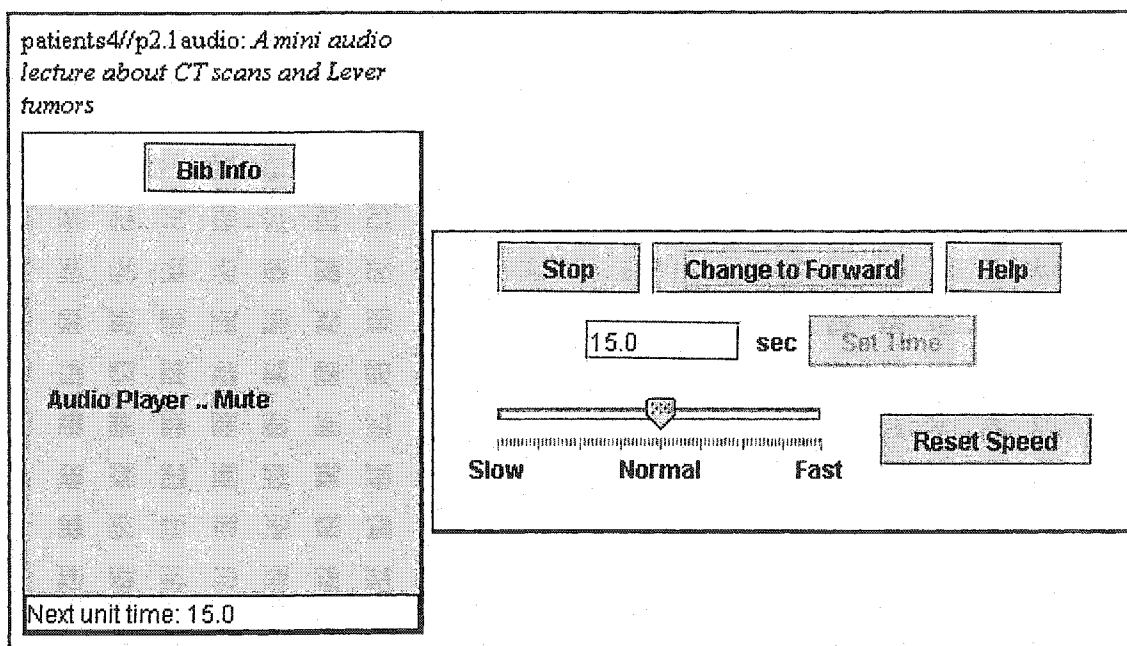


Fig. 30. An audio stream muted because the stream is playing backward.

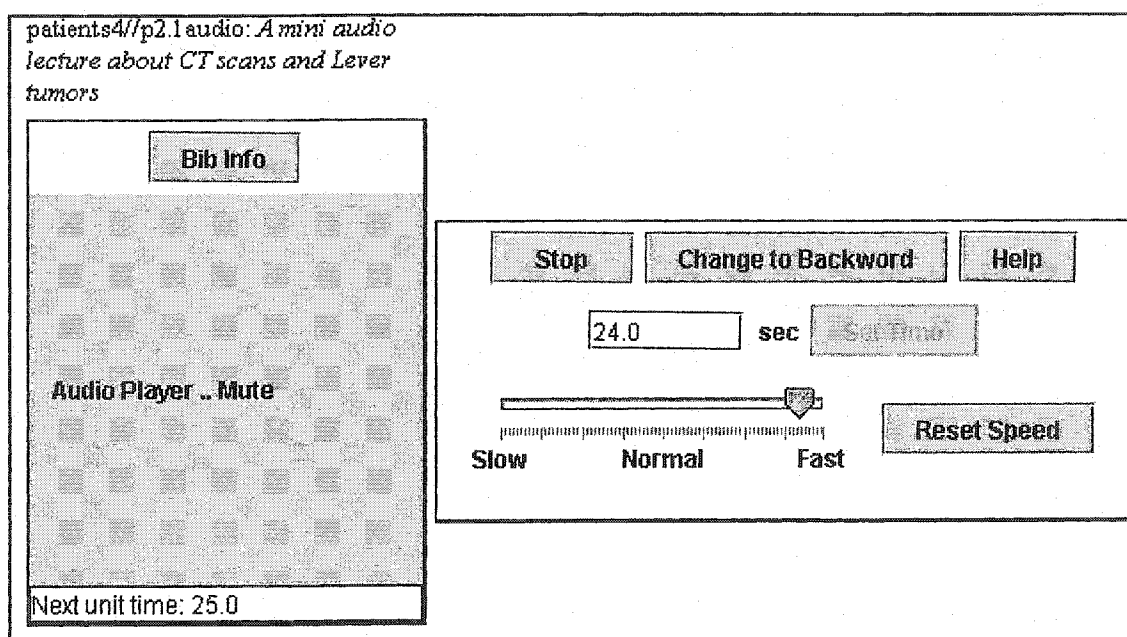


Fig. 31. An audio stream muted because the stream is playing faster than normal.

6.8.3 Stream player structure

In this implementation, there is a common structure for all stream players regardless of the stream type. The player interface consists of three panels. The top panel contains a button “Bib Info”, which displays a bib information window, similar to that in Fig. 21, for the stream to be played. The middle panel shows the stream being played if it is visual as in CT scan and text streams. The bottom panel reflects the current status, which is mostly the time of the next frame to be played. In this implementation, the time shown is the offset in seconds from the start of the stream being played. Stream players are controlled by the control panel applet that is described in Section 6.8.2. The data units of each stream, or the frames, are only played back upon receiving a request from the control panel through a timer event. The main difference between the stream player of a certain type and of another is the way the stream frame will be played. In a CT scan stream, the frame is a GIF image that will be displayed. In a text stream, the frame is a text string that will also be displayed. In an audio stream, the frame is an audio clip that will be played. Fig. 28 and Fig. 29 illustrate the interface of the CT scan, text, and audio player applets. Once the stream player starts, it contacts the streaming server, receives the URLs of the stream frames as well as their timing information, and stores them in an array. The frames are then buffered and displayed based on the timing information and controlled by the control panel applet.

6.8.4 The streaming server

Stream player applets contact a streaming server to get the frames of the stream to be played back. The stream player is a Java application that runs on the web server at a specific port known to all the players. The streaming server waits for player applets

connections and whenever any of them connects it spawns a thread to handle the request of that player applet. The thread receives the stream ID information from the applet upon connection. It uses the stream ID to identify the stream object path in the file system. It then accesses the stream object to read the list of the stream frames' URLs and their time offset from the start of the stream. The thread then sends back all this information to the player applet and just stops.

6.8.5 Playing synchronous streams

In the stream display interface described in Section 6.7, the user can select some synchronous streams to be played back simultaneously with the original stream. Since all of these streams are synchronous, only one control panel will be needed to control all of them. Fig. 23, Fig. 28, and Fig. 32 show some examples of playing back synchronous streams. The original stream will play at the leftmost player applet. The time shown in the current time text field in the control panel represents the current time of the original stream. Synchronous streams might not have the same start time as the original stream. They may start earlier or later. The status line in each stream player applet will show the time of the next frame to be played with respect to the start time of that stream. That is why the next unit time shown on the status line may differ from one stream to another although they are synchronous. An example is the situation of a radiology professor who started recording the audio stream of his lecture at 11:30 and then started recording a CT scan 15 seconds later. Both streams are synchronous; they are recorded at the same time, but one stream started 15 seconds earlier than the other. In the playback, assuming the audio stream is the original stream and the CT scan stream is the related synchronous stream, the audio stream will start the playback first. The status line of the audio player

applet will show the offset from the start of that audio stream. The CT scan stream will not start the playback at the same time; it will start it 15 seconds later. Before the first 15 seconds are elapsed, the status line of the CT scan player applet will show that the next frame time is 1 second, i.e. the next frame will be the frame playing at a 1 second offset from the start of that CT scan stream. If the original stream is the CT scan stream and the audio stream is the related synchronous stream, starting the CT scan from the beginning means that the audio stream will start 15 seconds from its beginning. When the status line of the CT scan shows next frame starting at 1 second, the status line of the audio stream will show that the next frame will start at 15 second and the playback will start from the middle of the stream. Fig. 32 shows an example where there is a 10 seconds gap between the audio stream and a related synchronous CT scan stream.

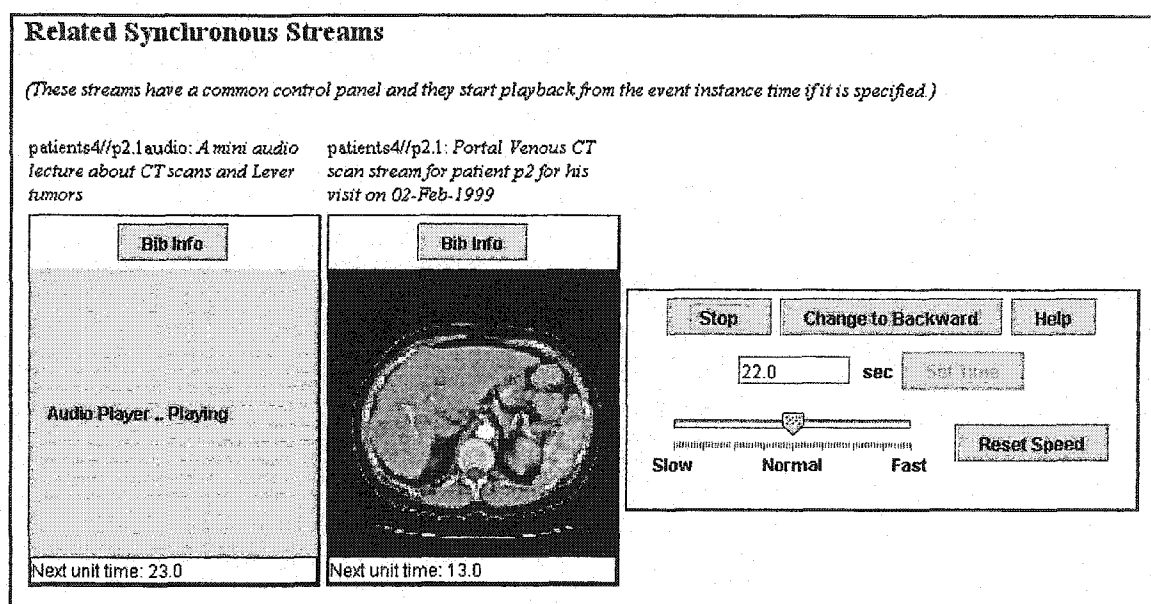


Fig. 32. Two synchronous streams playing: audio and CT scan. A 10-seconds gap exists between the two streams as shown in the status line of the player applets.

6.8.6 Playing asynchronous streams

In the stream display interface described in Section 6.7, the user can select some asynchronous streams to be played back simultaneously with the original stream. These asynchronous related streams are supposed to be recorded at different times but they are related to the original stream any way. They can be related because they all concern the same patient, same doctor, have similar events, etc. Since they are asynchronous, there is no point in trying to synchronize their playback. Each one of these streams will have its own control panel as can be seen in Fig. 23. A control panel will only control the players to its left. Some streams can be played while the others are stopped. Each stream will be completely independent from the others.

6.9 TYPE EXTENSION

One important feature of this implementation is the ability to easily support new stream types besides the three currently supported types. This ability comes from the fact that adding streams of any new type will not require any modifications to the current implementation. In this section, the steps needed to add streams of any new data type as well as players to these streams are presented.

6.9.1 Defining a new stream type in the database

All stream types are defined in a database table as described in Section 6.2.2. To add a new stream type, a new record should be added to that table. This new record will include:

- A code for this type
- A type name as it should appear in the various menus

- The format of the streams of this type
- The class name of a Java applet that will be used to play streams of this type

For example the CT scan streams are of a type that has a code “CTSCAN”, a name “CT scan”, a format “GIF”, and a player class name “ImagePlayer.class”. Specifics on how to create a new player applet are discussed in the following section.

6.9.2 Implementing a new stream player

For a new Java applet to be used to play streams in this implementation, it has to follow a certain structure and implement a specific interface. All stream players are controlled with one control applet. The control applet calls specific methods in any player applet in order to play or stop the stream, change the playback speed, or change the playback direction. In order to make sure that these methods are there in every stream player, the stream player has to implement a specific interface called “SBDLPlayer”. The source for this Java interface is shown here:

```
/**
 * All stream players in this application must implement this interface.
 * the control applet will cast all the player applets to this interface and
 * then call the methods below as needed.
 * @author Mohamed Kholief
 */
public interface SBDLPlayer
{
    int DEFAULTPORT=9911;
    void tick(); //for discrete media, e.g. text messages, image streams
    void changeDirection();
    void setCurrentTime(int newTime);
    void play(); //for continuous media, e.g. audio, video
    void stopPlayer(); //for continuous media, e.g. audio, video
    void setNormalSpeed(boolean flag); //for continuous media, e.g. audio, video
}
```

A recommended way to implement a new stream player is to start from an existing one. The current system has players for a stream of images, a stream of text strings, and a stream of audio clips.

6.9.3 Publishing streams of a new type

Publishing streams of a new type in the digital library is not much different from the process described in Section 6.3. What needs to be emphasized though is that there could be specific metadata fields for streams of the new type. These specific metadata fields are saved in two different places. A specific metadata file should be saved in the file system directory of the stream and a concatenated field of all entries in this file should be added to the database table. The specific metadata file should follow the same format described in Section 6.2.1, which is the RFC 1807-like format [55]. The fields in this file are actually not constrained; any field could actually be added and even different fields can be added to different streams from the same type. Concatenating all of these fields and inserting them in the stream bib metadata file in the database, enables the user to search and retrieve streams based on the specific metadata fields besides the standard bibliographic metadata fields.

6.10 SUMMARY

This section demonstrates a case study of a digital library containing data streams and supporting event-based retrieval. The section goes through all the aspects of the implementation. It starts with talking about what data are used in this library and how they are obtained. Three main types of streams are used, CT scans, text, and audio; with CT scan streams obtained from actual data from a hospital. The section then goes on to

describe how the data are organized in the library, showing that some data are saved in file-system directories and files while some other data are stored in a database. The organization of the file system and the schema design of the database are described. The various used metadata are described in the context of data organization. The publishing process then follows. We describe mass publishing of CT scan streams and how the metadata items are obtained from the native metadata of the hospital-saved scans. This section has also focused on individual publishing of manually generated text and audio streams. The section then continues with the event generation process and on what basis events is determined. The same is discussed with regard to relating streams, as related-streams are an integral part of the retrieval from the digital library.

Section 6.6 presents the main search interface, which the user first accesses upon using the digital library, along with many snapshots of the system. The simple search, event-based search, and browse interfaces are described in detail. The stream display interface, which is shown when the user selects one stream from the search results, is depicted next. The playback interface, which is shown if the user opts to play the stream displayed in the stream display interface, is described in great detail in Section 6.8.

The section ends with describing how streams of a new type, other than CT scans, text, or audio, can be added to this case study. The section in this way can be described as a detailed user manual for one application of the digital library system proposed in this dissertation and also a technical overview of all implementation aspects.

The implementation and use of this system demonstrates how feasible it is to create a digital library that contains data streams and supports event-based retrieval. We represent events in a way that they can be easily used to retrieve and playback stream

segments. Streams can be displayed in a concise, yet complete, way that makes it unproblematic for users to decide whether to playback a stream or not and to set playback options. The playback interface itself is organized in a way that accommodates synchronous and asynchronous streams and makes it very simple for users to control these streams.

Having used this library we saw performance comparable to such libraries as Arc even for the new search and retrieval capabilities explained above. As seen in Section 6.9, the library can be adapted to support more stream types. The underlying engine proves to work efficiently - testing at various phases of the data publishing process showed no performance differences - [59], which bodes well for the scalability of this work.

SECTION 7

CONCLUSIONS AND FUTURE WORK

This dissertation presents the concepts, applications, architecture, and implementation details for event-based retrieval from a digital library containing data streams. We introduced concepts relevant to streams, events, and related streams. We also shed light on the roles of possible users of the digital library. Our approach for this research was to identify some possible applications of the proposed digital library system, suggest the requirements of potential users of each of these applications; and based on these requirements; decide on the goals that play an integral and crucial role in designing that digital library. We developed the architecture of a prototype digital library founded on these design considerations. A case study, a digital library containing actual streams of CT scans and sample related medical text and audio, was implemented. This section presents the conclusions and possible future extensions to this research.

7.1 CONCLUSIONS

The objective of this research is to study the issues involved in building a digital library that contains data streams and allows event-based retrieval as well as other traditional retrieval approaches. This objective was divided into 4 components in Section 1.2. All the components of this research objective were successfully met. We were able to design a prototype of a heterogeneous stream-based digital library. The same library can hold streams of many data types without any modifications to the base architecture. The implementation of the case study proved the possibility of adding streams of any new type to the digital library with only minor extensions to the implementation. These were:

the addition of a new entry to the stream-types' database table and the use of a template Java-based player applet to implement the player of streams of the new type. A Java interface was used to force the developer to use the appropriate method names for the new player since the control applet will call these methods to control the player functionality.

Events were efficiently used for retrieval from the digital library. The event information was represented in the underlying architecture as well as the user interface. Users can retrieve stream segments using events just by selecting the event name from a menu, choosing the time instance from another menu, and then clicking a "play" button; a process that is as few as 3 mouse clicks if no other options needed to be set. Comparing with video- or audio-on demand libraries, where the user has to play an entire stream to find the desired segment, this is a much faster and more intuitive approach. Furthermore; the digital library design makes it very easy to insert new events and their time instances just by modifying database tables without any need to modify the code. While tools are needed to detect the events in data streams, our survey in Section 2 shows that an assortment of promising research is going on in this area.

The architecture of a prototype digital library was completed, which was an objective in itself. We presented several possible applications and studied their design considerations which facilitated the design of this prototype.

To prove our concept, a case study has been carefully explored. The medical digital library that we implemented contained real CT streams of liver tumor patients obtained from a hospital and some sample text and synchronized audio streams created manually to prove the heterogeneity of our prototype digital library. Domain experts, who

are radiologists in this case, were responsible for specifying the events important to them as they need to refer to them in their research as well as their regular daily job. They were also responsible for generating time tables of some sample events and for specifying the bibliographic metadata fields that need to be used in this digital library. Making domain experts responsible for these technical details insures that the resulting digital library is valuable to the intended user population.

The digital library implementation was based on Arc [59]; a digital library engine proven to scale well for a very large number of documents. Our library used the same engine and thus it has the same performance in retrieval. Reaching the stream object in itself was equivalent to reaching a document in Arc but using different search fields. The overall performance really depended on the Oracle database management system, which is accepted as highly efficient in real world applications, in the backend of the system.

To summarize, the implementation and use of this system demonstrated that it is feasible to create a digital library that contains data streams and supports event-based retrieval. We were able to successfully represent events such that traditional DL services run at a comparable performance with no degradation. On the other hand, the stream-based digital library realized its specific features in such a way that retrieval and playback can be accomplished in as few as 3 mouse clicks. The playback interface itself was organized in a way that accommodated synchronous and asynchronous streams and enabled users to control these streams using a set of buttons in a control panel. The ease of use of this library was informally confirmed by a group of radiologists. The underlying search engine, Arc, has proven to work efficiently with the data saved in the library, which promises the scalability of this work. The heterogeneity of the digital library is

also emphasized in every aspect of the implementation. In particular, the library can be adapted to support more stream types just by defining a new type in the stream-types database table and defining the player for this new type by implementing a pre-defined Java interface and using an existing template. Key to the specification of events and what metadata to use were the domain experts (radiologists). In this case study we did not concentrate on sophisticated editors to publish the streams and events but on the presentation to the resource discovery user. Also, domain experts were used to create the proper wording in the various event and playback menus, which, as explained above, makes the digital library more appealing to the intended user population since their terminology is what is being used. .

The first lesson learned from this study is: it is feasible to build a domain specific stream-based digital library that is efficient and scales to support a production system. Secondly, we learned that it is relatively easy to automate a number of processes. One process that can be automated but left to a future extension is the process of publishing individual streams. Another is the creation of tools to further automate the process of relating streams. We also need to formally define the processes of tailoring a stream-based digital library for a new domain and obtaining the information from the domain experts. These and other long-term future extensions are discussed in more detail in Section 7.2.

7.2 FUTURE EXTENSIONS

This section briefly goes through some potential extensions to this work.

7.2.1 Event generation tool

In a potential expansion to this system, an event-generation tool will be used to automate the event generation. The main functionality of this tool will be to:

- Read the event name and the IDs of the streams affected by this event from the user.
- Apply a suitable ready-made criteria module to the stream to generate the time instances of this event.
- Read the event-related streams from the user, or apply another criteria module to decide which streams relate to this event.
- Insert the event name, time instances, and related streams in the appropriate database tables.

7.2.2 Automatic event generation

The main goal of this research is to study and prove the feasibility of using events for retrieval from a digital library containing data streams. Focusing on this goal, we stopped short of going through automated event detection and generation algorithms after obtaining the data streams since it would have been a great distraction from the original research objectives. It is an ongoing significant research area as we presented in Section 2. We inserted events manually in the digital library. Nevertheless, automating event generation would appeal to potential users of the proposed digital library system and it is a planned long -term future extension to this research.

7.2.3 Dynamic event generation

In this research we concentrated on static event generation. Dynamic event generation is defined as applying event generation modules in real time while the streams

are generated from their sources. Dynamic event generators are much like software agents that intercept the generated data streams and check if a unit satisfies the event criteria and then produce the tables of event instances. This process may or may not require additional real-time expert manual input. A long -term future extension of this research would deal with this possibility. It is obviously very difficult in this particular domain of Liver Tumors since it is more like a Knowledge Base system that needs more than CT scans, even if very efficient pattern recognition algorithms existed, to decide if there is tumors or not

7.2.4 Advanced search interface

The advanced interface would allow users to locate streams based on a combination of the values of specific bibliographic fields and events that occurred during the stream. The browse interface enables users to go through all the contents of the digital library. The advanced search interface was postponed to a future extension to the system since this is an experimental system and it does not add anything to the proof of the concept behind this work.

7.2.5 OAI compliance

One of the reasons of using Arc as the underlying search mechanism is that it is an implementation of the Open Archives Initiative (OAI), which could be an interesting extension to this research. A future work on this system is to use more than one repository and to make all of them OAI-compliant. Implementing the same harvesting techniques used in Arc with minor modifications will enable the search engine to retrieve streams from all of those repositories.

7.2.6 Manipulating a greater number of events

Future work in this field of research includes grouping events in tree-structured categories to speed up locating the required events. Another promising approach is to allow the user to type part of the event name and dynamically generate a menu that contains the events that include this part.

7.2.7 Event-related streams

The concentration in the current implementation is on stream-related streams. In Section 3, we presented the concept of event-related streams. This can be implemented using a database table that contains the event ID and the IDs of its related streams.

7.2.8 Further automating the process of relating streams

In this work, scripts were used to automate the generation of related streams tables based on some common fields, e.g.: same patient. Relating streams based on other factors, such as the existence of similar events, was done manually. A future extension is to use a user-friendly tool that lets the user specify the relationship criteria and generates the tables accordingly.

7.2.9 Publishing tool

A possible future extension to this work is to create a publishing tool that would take the user through a wizard to help him/her publish the stream without actually having to know the underlying file system or database details. This tool would ask the user to input the necessary metadata information and then create all the necessary files transparently. It would also upload all the data files of the stream into the appropriate file

system folders. An advanced feature of this tool would be to let the user decide and input the field names of the specific bibliographic metadata fields.

7.2.10 More applications

We plan to use our digital library prototype for other applications. As presented in Section 4, there are many possible applications to this research: an educational digital library, a weather-related digital library, a stock market digital library, and a census-bureau statistics digital library, to name a few. Another promising potential application is a digital library containing streams related to information about security. Today, with heightened security measures, there is a need for sophisticated approaches to monitor security streams such as security cameras' output and airport arrival data for potential events. For example, the appearance of the face of a fugitive in a security camera's output stream would be an event. This raises some interesting research issues some of which are: the need to preserve these streams in a digital library in the first place, the detection of events, dynamic event generation, digital library security and access control (not anybody can access such a library of course), and interoperability. Using the digital library prototype for the above applications will be beneficial in the continued refinement of this prototype.

REFERENCES

- [1] Nabil R. Adam, Milton Halem, and Shamim Naqvi, "Promising research directions in digital libraries," in *Proc. of the Digital Libraries Workshop DL'94*, May 1994, pp. 21-30.
- [2] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic detection and tracking pilot study: Final Report," in *Proc. of the Broadcast News Transcription and Understanding Workshop*, 1998, pp. 194-218.
- [3] F. Arman, A. Hsu, "Image processing on compressed data for large video databases," in *Proc. of ACM Multimedia'93*, 1993, pp. 267-272.
- [4] Daniel E. Atkins. (1997) Report of the Santa Fe planning workshop on distributed knowledge work environments: Digital Libraries. [Online]. Available: <http://www.si.umich.edu/SantaFe/report.html>.
- [5] Donald Baker, Anthony R. Cassandra, and Mosfeq Rashid, "CEDMOS: complex event detection and monitoring system," MCC Tech. Rep. MCC-CEDMOS-002-99, 1999.
- [6] D. Bainbridge, C.G. Nevill-Manning, I.H. Witten, L.A. Smith, and R.J. McNab, "Towards a digital library of popular music," in *Proc. of Digital Libraries 1999*, May 1999, pp. 161-169.
- [7] Bharat Bhargava, Melliya Annamalai, Shalab Goel, Shunge Li, Evaggelia Pitoura, Aidong Zhang, Yongguang Zhang, "DL-Raid: an environment for supporting digital library services," in *Proc. Digital Libraries Workshop DL'94*, May 1994, pp. 281-300.
- [8] John Bovey, "Event based personal retrieval," *Journal of Information Science*, vol. 22, no. 5, pp. 357-366, May 1996.
- [9] Lynnwood Brown, *The Complete Oracle DBA Training Course*. Upper Saddle River, NJ: Prentice Hall, 2000.
- [10] A. Carzaniga, "Architectures for an event notification service scalable to wide-area networks," PhD dissertation, Politecnico di Milano, Milano, Italy, 1998.
- [11] Census Bureau Home Page. [Online]. Available: <http://www.census.gov>.
- [12] S. Chakravarthy and D. Mishra, "An event specification language (Snoop) for active databases and its detection," University of Florida, Tech. Rep. UF-CIS-TR-91-23, Sept. 91.

- [13] S. Chakravarthy and D. Mishra, "Snoop: an expressive event specification language for active databases," *Knowledge and Data Engineering Journal*, vol. 14, pp. 1-26, 1994.
- [14] J. Chen, D. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: a scalable continuous query system for internet databases," in *Proc. of the ACM SIGMOD Conf. on Management of Data*, 2000.
- [15] M. Christel, "Informedia Digital Video Library," in *Proc. of the ACM Multimedia'94*, 1994, pp. 480-481.
- [16] ———, "Informedia Digital Video Library," *Communications of ACM*, vol. 38, no. 4, pp. 57-58, 1995.
- [17] Gary Cleveland, "The Challenge of the Digital Library," *National Library News*, vol. 28, no. 5, pp. 5, May 1996. Available: <http://www.nlc-bnc.ca/pubs/nl-news/1996/may96e/2805e-07.htm>.
- [18] Definition and Purposes of a Digital Library, Association of Research Libraries. (1995). [Online]. Available: <http://sunsite.berkeley.edu/ARL/definition.html>.
- [19] H. V. de Sompel, T. Krichel, M. L. Nelson, P. Hochstenbach, V. M. Lyapunov, K. Maly, M. Zubair, M. Kholief, X. Liu, and H. O'Connell. (2000, February). The UPS prototype: an experimental end-user service across E-Print archives. *D-Lib Magazine*. [Online]. 6(2). Available: <http://www.dlib.org>.
- [20] The Digital Library Research Group at Old Dominion University. [Online]. Available: <http://dlib.cs.odu.edu>.
- [21] K. R. Dittrich and S. Gatzui, "Time Issues in Active Database Systems," in *Proc. Intl. Workshop on an Infrastructure for Temporal Databases*, June 93.
- [22] R. S. Drew, D. Morris, P. M. Dew, and C. Leigh, "System architecture for supporting event-based interaction and information access," in *Proc. Multimedia Computing and Networking*, Jan. 1996, pp. 148-160.
- [23] Dublin Core Metadata Initiative. [Online]. Available: <http://dublincore.org/>
- [24] J. V. Dunn and C. A. Mayers, "VARIATIONS: a digital library system at Indiana University," in *Proc. of the fourth ACM Conference on Digital Libraries DL'99*, 1999, pp. 12-19.
- [25] EGARC MP3 Audio Library. [Online]. Available: <http://www.ku.edu/~egarc/diglib/index.htm>

- [26] D. Faensen, L. Faulstich, H. Schweppe, A. Hinze, and A. Steidinger, "Hermes – a notification service for digital libraries," in *Proc. ACM/IEEE Joint Conference on Digital Libraries 2001*, June 2001, pp. 373-380.
- [27] C. Faloutsos, "Efficient and effective querying by image content", *Journal of Intelligent Information Systems*, vol. 3, pp. 231-262, 1994.
- [28] W. Farag, H. Abdel-Wahab, K. Maly, C. M. Overstreet, C. Wild, A. Gupta, A. Abdel-Hamid, S. Ghanem, and B. Koodallur, "Recording and replay support for asynchronous learning modes within a distance learning system," in *Proc. of Networked Learning in a Global Environment: Challenges and Solutions for Virtual Education (NL'2002)*, May 2002.
- [29] Nattha Flanagan, "Use digital library resources to enhance undergraduate education". [Online]. Available: <http://www.edcenter.sdsu.edu/repository/navDigLib.shtml>.
- [30] NASA's Earth Science Enterprise. [Online]. Available: <http://www.earth.nasa.gov/>.
- [31] E. Fonss-Jorgensen. (1997, April) JUKEBOX: Final report. Aarhus, Denmark: State and University Library, Tech. Rep. LIB-JUKEBOX/4-1049. Edited report no. 2. [Online]. Available: <http://www.sb.aau.dk/Jukebox/finalrep.html>.
- [32] H. Fujisawa, Yusuka Mishina, Minoru Ashizawa, and Kanji Kato, "Multimedia digital library systems for the global information network," *Hitachi review*, vol 44, no. 5, pp. 273-280, Oct. 95.
- [33] S. Gatzju and K. R. Dittrich, "Events in an active object-oriented database system," in *Proc. of RIDS 1993*, 1993, pp. 23-39.
- [34] S. Gauch and R. Aust, "The digital video library system: vision and design," in *Proc. of Digital Libraries '94*, June 1994, pp. 47-52.
- [35] S. Gauch, John. M. Gauch, and Kok Meng Pua, "The VISION digital video library project," *The Encyclopedia Microcomputers*, vol. 28, no. 7, pp. 365-379, 2002.
- [36] S. Gauch, W. Li, and J. Gauch, "The VISION digital video library system," *Information Processing & Management*, vol. 33, no. 4, pp. 413-426, 1997.
- [37] N. H. Gehani, H.V. Jagadish, and O. Shmueli, "Event specification in an active object-oriented database," in *Proc. Intl. Conf. on Management of Data (SIGMOD)*, June 92, pp. 81-90.

- [38] H. M. Gladney, E. A. Fox, Z. Ahmed, R. Ashany, N. J. Belkin, and M. Zemankova, "Digital library: gross structure and requirements: report from a March 1994 workshop," in *Proc. of the First Annual Conference on the Theory and Practice of Digital Libraries*, June 1994.
- [39] The Gnutella file sharing system. [Online]. Available: <http://www.gnutella.com>.
- [40] C. F. Goldfarb and P. Prescod, *XML Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.
- [41] N. Haering, Richard J. Qian, and M. I. Sezan, "A semantic event-detection approach and its application to detecting hunts in wildlife video," *IEEE transactions on circuits and systems for video technology*, vol. 10, no. 6, Sept. 2000.
- [42] M. Hall, *Core SERVLETS and JAVASERVER PAGES™*. Upper Saddle River, NJ: Prentice Hall, May 2000.
- [43] R. J. Hampapur and T. Weymouth, "Digital Video Segmentation", in *Proc. ACM Multimedia 94*, 1994, pp. 357-364.
- [44] R. M. Haralick and L.G. Shapiro, *Computer and Robot Vision*. Reading, MA: AddisonWesley, 1992.
- [45] J. Hauptmann, M. Witbrock, "Informedia: news-on-demand multimedia information acquisition and retrieval, intelligent multimedia information retrieval," Mark T. Maybury, Ed., AAAI Press, pp. 213-239, 1997.
- [46] Imaginis: Computed Tomography Imaging (CT Scan, CAT Scan). [Online]. Available: <http://imaginis.com/ct-scan/>.
- [47] Interactive Remote Instruction, distance education over the Internet. [Online]. Available: <http://www.cs.odu.edu/~tele/iri/index.html>.
- [48] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast multiresolution image querying," in *Proc. ACM Computer Graphics (SIGGRAPH '95)*, 1995, pp. 277-286.
- [49] M. Kholief, K. Maly, and S. Shen, "A medical streams-based digital library architecture," in *Proc. JCDL'03*, May 2003, pp. 231-233.
- [50] M. Kholief, S. Shen, and K. Maly, "Architecture for event-based retrieval from data streams in digital libraries," in *Proc. ECDL'2001*, Sept. 2001, pp. 300-312.
- [51] M. Kholief, S. Shen, and K. Maly, "Event-based retrieval from digital libraries containing streamed data," in *Proc. ITCC 2000*, Mar. 2000, pp. 234-238.

- [52] V. Krueger and S. Zhou, "Exemplar-based face recognition from video," in *Proc. of fifth IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002, pp. 182-188.
- [53] M. G. Lamming, P. J. Brown, K. Carter, M. Eldridge, G. Flynn, G. Louie, P. Robinson, and A. Sellen, "The design of a human memory prosthesis," *Computer Journal*, vol 37, no. 3, pp. 153-163, 1994.
- [54] M. Lansdale and E. Edmunds, "Using memory for events in the design of a personal filing system," *International Journal of Man-Machine Studies*, vol. 46, pp. 97-126, 1992.
- [55] R. Lasher and D. Cohen. (1995, June) A Format for Bibliographic Records, RFC 1807. [Online]. Available: <http://www.ietf.org/rfc/rfc1807.txt?number=1807>
- [56] C. J. Lindblad, "The VuSystem: a programming system for visual processing for digital video," in *Proc. of ACM Multimedia'94*, 1994, pp. 307-314.
- [57] H. Liu and H. A. Jacobsen, "A-topss - a publish/subscribe system supporting approximate matching," in *Proc. of the Intl. Conference on Very Large Data Bases (VLDB)*, 2002, pp. 1107-1110.
- [58] Ling Liu, Calton Pu, and Wei Tang, "Continual queries for Internet scale event-driven information delivery", *Special issue on Web Technologies, IEEE Transactions on Knowledge and Data Engineering*, vol.11, no.4, pp 610-628, July 1999.
- [59] X. Liu, K. Maly, and M. Zubair. Arc. [Online]. Available: <http://sourceforge.net/projects/oaiarc/>.
- [60] E Loftus and W Marburger, "Since the eruption of Mount St Helens, has anyone beaten you up? Improving the accuracy of retrospective reports with landmark events", *Memory and Cognition*, vol. 11, pp. 114-120, 1983.
- [61] Lycos Music Downloads. [Online]. Available: <http://mp3.lycos.com>.
- [62] E. Lyon and J. Maslin, "Audio and video on-demand for the performing arts: Project PATRON", *International Journal of Electronic Library Research*, vol. 1 no 2, pp. 119-131, 1997.
- [63] C. Lynch, "Searching the Internet," *Scientific American*, no. 276, pp. 50-56. Mar. 1997.

- [64] G. R. Malan, F. Jahanian, and S. Subramanian, "Salamander: a push-based distribution substrate for Internet applications," in *Proc. USENIX Symposium on Internet Technologies and Systems*, Dec. 1997, pp. 171-181.
- [65] K. Maly, M. Nelson, and M. Zubair. (1999, March). Smart objects, dumb archives, a user-centric, layered digital library framework. *D-Lib Magazine*. [Online]. 5(3). Available: <http://www.dlib.org/dlib/march99/maly/03maly.html>.
- [66] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham, "Toward the digital music library: tune retrieval from acoustic input," in *Proc. of DL '96: the first ACM Conference on Digital Libraries*, 1996, pp. 11-18.
- [67] M. Melucci, and N. Orio, "SMILE: a system for content-based musical information retrieval environments," in *Proc. of RIAO 2000*, April 2000, vol. 2, pp. 1261 – 1279.
- [68] The MSDN Library: Event definition. [Online]. Available: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/off2000/html/defEvent.asp>.
- [69] MySQL reference manual. [Online]. Available: <http://mysql.dyntex.com/>
- [70] A. Nagasaka and T. Tanaka, "Automatic video indexing and full-video search for object appearances," in *IFIP Proc. of Visual Database Systems*, 1992, pp. 113-27.
- [71] The Napster Music Community. [Online]. Available: <http://www.napster.com>.
- [72] M. Nelson, K. Maly, and S. Shen, "Building a multi-discipline digital library through extending the Dienst protocol", in *Proc. of the Second International ACM Conference on Digital Libraries*, July 1997, pp. 262-263.
- [73] W. M. Newman, M. Eldridge, and M. G. Lamming, "PEPYS: generating autobiographies by automatic tracking," in *Proc. of the second European Conference on Computer-Supported Cooperative Work*. Kluwer, 1991.
- [74] W. M. Newman and M. G. Lamming, *Interactive System Design*. Reading, MA: Addison-Wesley, 1995.
- [75] Peter J. Nuernberg, Richard Furuta, John J. Leggett, Catherine C. Marshall, and Frank M. Shipman III, "Digital Libraries: Issues and Architectures," in *Proc. of Digital Libraries '95, The Second Annual Conference on the Theory and Practice of Digital Libraries*, June 1995.
- [76] Open Archives Initiative. [Online]. Available: <http://www.openarchives.org/>.

- [77] J. Papadakis and C. Douligeris, "Design and architecture of a digital music library on the Web," *New Review on Hypermedia and Multimedia, NRHM*, vol. 7, pp. 115-126, 2001.
- [78] R. Pentland, W. Picard, and S. Sclaroff, "Photobook: Content-Based Manipulation of Image Databases", *International Journal of Computer Vision*, vol. 18, no. 3, pp. 233-254, 1996.
- [79] J. Pereira, F. Fabret, H. Jacobsen, F. Llirbat, R. Preotiuc-Prieto, K. Ross, and D. Shasha, "LeSubscribe: publish and subscribe on the web at extreme speed," in *Proc. of the ACM SIGMOD Conf. on Management of Data*, 2001.
- [80] M. Petkovic and W. Jonker. "Content-based video retrieval by integrating spatio-temporal and stochastic recognition of events," in *Proc. of the IEEE International Workshop on Detection and Recognition of Events in Video*, July 2001.
- [81] M. Petkovic and W. Jonker. "Content-based retrieval of spatio-temporal video events," in *Proc. of the IRMA International Conference*, May 2001.
- [82] R. W. Picard and F. Liu, "A new Wold ordering for image similarity," in *Proc. ICASSP*, 1994.
- [83] H. S. Sawhney and S. Ayer, "Compact representations of videos through dominant and multiple motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 814-830, 1996.
- [84] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps, "Content based routing with Elvin4," in *Proc. of the AUUG2K Conference*, 2000.
- [85] B. Shahraray and D. Gibbon, "Pictorial transcripts: multimedia processing applied to digital library creation," in *Proc. IEEE First Workshop on Multimedia Signal Processing*, June 1997, pp. 581-586.
- [86] R. L. Schwartz and T. Christiansen, *Learning Perl*. O'Reilly & Associates, July 1997.
- [87] S. Smoliar and H. Zhang, "Content-based video indexing and retrieval," *IEEE Multimedia Magazine*, vol. 1, no. 2, pp. 62-72, 1994.
- [88] Spire project. [Online]. Available: http://www.research.ibm.com/networked_data_systems/spire/.
- [89] R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward, "Gryphon: an information flow based approach to message brokering," in *Proc. of the International Symposium on Software Reliability Engineering*, 1998.

- [90] M. Swain and D. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11-32, 1991.
- [91] L. Teodosio and W. Bender, "Salient video stills: content and context preserved," in *Proc. ACM Multimedia'93*, 1993, pp. 39-46.
- [92] E Tulving, *Elements of Episodic Memory*. Oxford, UK: Clarendon Press, 1983.
- [93] H. Wactlar, S. Stevens, M. Smith, and T. Kanade, "Intelligent access to digital video: the Informedia project," *IEEE Computer, Digital Library Initiative Special Issue*, vol. 29, no. 5, May 1996.
- [94] W. A. Wagenaar, "My memory: a study of autobiographical memory over six years," *Cognitive Psychology*, vol. 18, pp. 225-252, 1986.
- [95] W. Li, S. Gauch, J. Gauch, and K. M. Pua, "VISION: a digital video library," in *Proc. ACM Digital Libraries '96*, March 1996, pp. 19-27.
- [96] R. Weiss, A. Duda, D. K. Gifford, "Composition and search with a video algebra," *IEEE Multimedia*, vol. 2, no. 1, pp. 12-25, 1995.
- [97] S. White, M. Fisher, R. Cattell, G. Hamilton, and M. Hapner, *JDBC(TM) API Tutorial and Reference: Universal Data Access for the Java(TM) 2 Platform*. Reading, MA: Addison-Wesley, June 1999.
- [98] R. Willett, "Recent trends in hierarchic document clustering: a critical review," *Information Processing and Management*, vol. 25, no. 5, pp. 577--597, 1988.
- [99] W. Wolf, "Key frame selection by motion analysis," in *Proc. ICASSP*, 1996, vol. 2, pp. 1228-1231.
- [100] W. Wolf and B. Liu, "A digital video library for classroom use," in *Proc. of the International Symposium on Digital Libraries*, 1995, pp. 250-255.
- [101] T. W. Yan and H. Garcia-Molina, "The SIFT information dissemination system," *ACM Transactions on Database Systems*, vol. 24, no. 4, pp. 529-565, 1999.
- [102] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. Archibald, and X. Liu, "Learning approaches for detecting and tracking news events," *IEEE Intelligent Systems: Special Issue on Applications of Intelligent Information Retrieval*, vol. 14, no. 4, pp. 32-43, July 1999.
- [103] H. Yu and W. Wolf, "Scenic classification methods for image and video databases, digital image storage and archiving systems," *SPIE*, vol. 26, no. 6, pp. 363-371, 1995.

- [104] H. Zhang, "Automatic partitioning of video," *Multimedia Systems*, vol.1, pp. 10-28, 1993.
- [105] H. Zhang, "Automatic parsing and indexing of news video," *Multimedia Systems*, vol. 2, no. 6, pp. 256-266, 1995.

VITA

Mohamed Hamed Kholief

Permanent address:

19 El-Moshier Ahmed Ismail St., Apt. #9-6

Sidi Gaber, Alexandria

Egypt

Department of study address:

Department of Computer Science, Old Dominion University

Norfolk, VA 23529-0162

USA

Education:

- Ph.D. Old Dominion University (2003)
- M.Sc. Alexandria University, Egypt (1998)
- B.Sc. Alexandria University, Egypt (1991)