An Extensible Architecture for Shareable Media
with a Sample Application Text2Movie

by

Cathy Ling Lin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science and Engineering
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001
© Cathy Ling Lin, MMI. All rights reserved.

The author hereby grants to MIT permission to reproduce and
distribute publicly paper and electronic copies of this thesis document
in whole or in part.
An Extensible Architecture for Shareable Media with a
Sample Application Text2Movie
by
Cathy Ling Lin

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2001, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

In this thesis, I designed and implemented an extensible web architecture for sharing multimedia contents among users as well as different types of online applications.

Shareable Media is a network-based system that explores how a community of users can share stories and express ideas through a shared database of digital multimedia objects. To adapt this to the rapidly evolving Internet, Shareable Media has the capability to deploy multiple applications on wired and wireless devices through connections of both broad and narrow bandwidth. The current architecture consists of five modules: Storage Manager, Shareable Media Framework, Access Control Unit, System Application and Application Manager.

Text2Movie is an application for amateurs to make a movie by writing a paragraph of textual description and selecting appropriate scenes suggested by the agent. This is shown as a sample application Shareable Media System hosts. Other applications, such as Individeo and PlusShorts, are hosted in the system too.

Thesis Supervisor: Glorianna Davenport
Title: Principal Research Scientist
MIT Media Laboratory, Interactive Cinema
Acknowledgments

First, I would like to thank my thesis advisor, Glorianna Davenport, for her insightful vision and trust. It is my great honor to take the responsibility for designing, developing and maintaining such an important system for Interactive Cinema Group in the past year. I would also like to thank all members of the Shareable Media Team: James Jung-Hoon Seo, Aisling Kelliher, Pengkai Pan, David Cheng, Emery Lin, Daniel Talavera, Jocelyn Lin and Stephan Kao. David Cheng, in particular, has been working intensively with me in detailed design and implementation issues. James Seo and Aisling Kelliher have integrated their visual interfaces onto the Shareable Media System to demonstrate the system works well. Pengkai Pen's application, M-Views, will be an application to demonstrate how the Shareable Media Architecture supports wireless and narrow bandwidth.

Text2Movie is a project for Context Aware course offered by Professor Henry Lieberman and Professor Ted Selker. They gave me enlightening ideas. Aisling and Glorianna helped me make detailed user scenarios how the Text2Movie is going to achieve.

This work is funded by the Digital Life Consortium, the News in the Future Consortium, IBM, Oracle, and Sun Microsystems. We also particularly thank Tom Gardos of Intel Corporation for his contributions to Shareable Media.
## Contents

1 Introduction 13
   1.1 Motivations for Shareable Media ............................................. 13
   1.2 System Objectives ........................................................................... 15
       1.2.1 Abstract Features ...................................................................... 16
   1.3 User Scenario .................................................................................. 18

2 Architectural Design Issues 23
   2.1 System Data Model ........................................................................... 23
       2.1.1 Media ......................................................................................... 23
       2.1.2 User ......................................................................................... 24
       2.1.3 Application ............................................................................... 25
       2.1.4 Attribute ................................................................................. 27
   2.2 Architectural Design and Analysis ..................................................... 27
   2.3 Related Work ................................................................................... 29

3 System Implementation 31
   3.1 Business Logic Modules .................................................................... 31
       3.1.1 Storage Manager ........................................................................ 31
       3.1.2 Shareable Media Framework ..................................................... 32
       3.1.3 Access Control Unit ................................................................... 33
   3.2 Application Manager ........................................................................ 34
       3.2.1 Compilation ............................................................................... 34
       3.2.2 Attributes ............................................................................... 34
## List of Figures

1-1 The Control Flow of an Online Database-backed Applications .......................... 16  
1-2 Shareable Media System’s Control Flow ....................................................... 17  
1-3 Confirmation Page for Submission ................................................................ 19  
1-4 Invoke Applications, Search, and View Search Results ............................... 19  
1-5 User Profile .................................................................................................. 20  
1-6 Membership in Communities ........................................................................ 21  
1-7 Developers’ Page to Add/Update Applications ............................................... 21  

2-1 The Media Object Hierarchy ......................................................................... 24  
2-2 The Shareable Media 3-Tier architecture ...................................................... 28  

3-1 The Dependencies of Business Logic Modules .............................................. 32  
3-2 Attribute Mapping Example .......................................................................... 35  
3-3 Generating Web Pages in System Application .............................................. 37  
3-4 The Shareable Media 2-Tier architecture (Prototype) .................................... 38  

4-1 Text2Movie’s User Interface .......................................................................... 44  
4-2 Attribute Mapping for Text2Movie ................................................................ 44  
4-3 Servlets Settings for Text2Movie .................................................................. 44  
4-4 A Sample Configuration page for Individeo ................................................. 46
List of Tables

2.1 Examples of Customized Attributes ........................................ 26
2.2 Currently Available Attributes’ Data Types .......................... 26
4.1 Servlets Registration Settings for Individeo .......................... 46
Chapter 1

Introduction

This thesis focuses on the design and implementation of an extensible web architecture, Shareable Media System, a multiple-application platform for online media sharing among its users. A sample application, Text2Movie, helps users make a short movie based on text description.

This chapter introduces the Shareable Media System, including its motivations, objectives and a user scenario.

Chapter Two describes the conceptual design criteria and preliminaries.

Chapter Three discusses the implementation of major modules and the running environment. A step-by-step tutorial describes how to add a new application or update an existing application onto the Shareable Media System.

Chapter Four introduces the design and implementation of Text2Movie, an interface to create a movie (i.e. a sequence of shared media clips) that matches images to a story text a user has typed into the system. It also discusses how sample applications, such as Individeo and PlusShorts, have been integrated to run on the system.

Chapter Five concludes the thesis by presenting future directions.

1.1 Motivations for Shareable Media

Reality TV and TV games are quite popular today. One of the reasons for this popularity is the audience participation. Ordinary people can become involved in an
increasing number of television programs and can become stars like the Survivors. We imagine people would like to be film writers, directors, editors, etc. In fact, everyone is a good writer and director of his own life. No one knows own person better than oneself. Although films of one's own life might not be as artistic as a film by a professional, one could easily find a 30-second clip from an amateur that would be interesting enough to be collected by a professional. Editing is different. Common perception is that professionals can do better than amateurs, although ironically the success is determined by mass consumers. Is there a way to draw everyone’s attention, to encourage consumer creativity and participation, and to make popular commercial products designed by ordinary consumers? The answer is yes.

Technology has changed our everyday life. Digital still and video cameras are available to millions of families. Internet has guided people from all over the world into a virtual village. Rich media will inevitably be created, used and shared by millions of amateurs in their daily life. Some people can contribute their media content, others may select and edit content from the large pool, and every participant can offer comments and evaluate the films that are created. We can imagine an MTV can be re-edited by music fans in thousands of variations. The mass participation itself has made the singer and song familiar to audience. Producers sample consumers’ taste and note what they like most. Further more, one or two amateur versions might become more popular than the original cut.

This kind of participation suggests a business trend. The Shareable Media project provides a platform by which we hope to enable interaction between the consumer and constructed films shared online.

Nowadays, the two basic models for sharing rich media content allow transfer of objects as complete entities from one place to another. The first model is a server-client broadcast or video-on-demand model, such as Real Server-Player and Microsoft’s Media Server-Player. The second one is peer-to-peer data sharing model, such as Napster and Gnutella. Both models fail to foster interaction and collaboration among users. Although people enjoy watching other multimedia works, they are not able to collaborate in the production of stories by reusing the shared content of the file. Instead,
they have to download all the pieces they need onto their local machines and edit them like the traditional way. These models consider new stories created from shared contents as whole new entities. It will be difficult to study their relationships to original contents, unless the content is explicitly specified or the system contains a complex image recognition feature.

Inspired by the “intercreativity” idea, a term coined by Tim Berners-Lee [3], we have addressed the following questions: What type of interchange might be developed to encourage people to create and share rich-media stories on the Internet? Can we build an easy-to-use model to facilitate sharing and reusing media content on the Internet?

The Shareable Media project provides a platform to explore these questions. Design principles focus on providing a coherent structure that will facilitate distributed collaboration and communication among film makers, storytellers, artists and the audience. The project invites communities of users to create web-based media stories through a shared database of digital video/audio clips. Each user can contribute the original material; this material becomes available to all other users. Prototype applications include interactive editing tools which allow users to compose new media objects based on original submitted small media components. The extensible web architecture resolves the following problems:

1. How can an architecture deploy multiple applications on wired and wireless devices through connections of both broad and narrow bandwidth?

2. How can new developers write novel applications for Shareable Media?

1.2 System Objectives

Most online systems are stand-alone applications. After the server receives a user request, it fetches related information directly from the database. As shown in Figure 1-1, the server program knows exactly where the data stored. When multiple programs (i.e. applications) are hosted on one server and share one database, each
of them can consider the database schema and modify this schema if needed. This requires application developers to learn database programming.

![Diagram](image)

Figure 1-1: The Control Flow of an Online Database-backed Applications

Unlike those stand-alone applications the Shareable Media Architecture (SMA) is a platform hosting multiple online applications (Figure 1-2). By the indirect database column mapping, developers are free from coding on the database level. A complete application programming interface (API) is offered, which provides direct database access, one of our major objectives.

The SMA supports multiple users, multiple communities and multiple applications. The access control is a crucial security and privacy issue. Using indirect mapping, the system can effectively prevent actions from being overruled.

1.2.1 Abstract Features

The Shareable Media System (SMS) stores multimedia objects uploaded by its members. The reuse of these elements can generate new artistic forms sculpted by original applications, which are uploaded on to the extensible platform. Here are four basic features:

1. **Membership**: An online user becomes a SMS member after he signs up. Members can create open or closed communities and become administrators in those communities. Any user can join or leave an open community; however,
only the administrator of a closed community can update the user list of that community.

2. **Raw Content**: Users submit multimedia files from their local machines to Shareable Media. The system accepts any media type that has been registered, by checking whether or not the uploaded binary object is valid. Upon the time of submission, the user has to decide whether the object is public in the SMS or only open to selected communities in which he is a member. This has some implications regarding longevity of the object on the system.

3. **Application Registry**: Any user (developer) may apply to create a new application to run on the SMS. The user (developer) begins by submitting the source code for both the client side and the server side. Once an application is online, any user can invoke it from the SMS. The primary goal and the major challenge is to design an architecture that can flexibly support many applications which are able to utilize raw media content in the database.

4. **Derivative Content**: Users can create derivative media contents using applications registered in the SMS. Derivatives are made of a selection of raw content.
The access rights of these derivatives are limited by the privacy of their raw contents.

1.3 User Scenario

The first time a user visits http://shareable.media.mit.edu (the Shareable Media web site), he needs to sign up. A unique user name and a valid email address is required. He can choose to be a regular user or a developer. A developer will be able to download the SMS Java API and create new applications to run on the Shareable Media System.

After logging on, a normal user can submit a new media file from his local machine. The system will validate the file format. Currently a QuickTime movie validating tool is available; we expect to support more media formats, such as MPEG and RealMedia, and we will make validating utilities in the near future. There are some media constraints on the submission. At present, the Shareable Media accepts a QuickTime movie that is shorter than 2 minutes, its resolution is at most 240 x 180, and its file size is less than 5 Megabytes. All constraints are stored in a database table. An application can make its own media constraints if it allows users to submit from the application interface.

If a movie file is valid, the system will randomly choose six frames and ask the user to choose one as a representing image for this short clip. The system can generate more frames if the user does not like any of the first six.

The user then types the title and 5 keywords for the clip. By default, the clip is public, which means any SMS users can watch and reuse it. The user can set this clip as visible to multiple communities where he is currently a member. In this case, only the users in the selected communities will be able to find it.

The last step in the submission is to confirm all the settings, shown in Figure 1-3.

The user goes to the View page, as shown in Figure 1-4 to search for clips and watch them. The system will match the keyword he types in with the title, keywords and author's user name of all clips in the communities he has selected. The user clicks
please confirm what you have submitted

title: purple flash
keywords: zoom flower purple flashing
filename: fw-zoom-4.mov
privacy: open to public

Figure 1-3: Confirmation Page for Submission

Figure 1-4: Invoke Applications, Search, and View Search Results
on a clip’s representative image from the search result to watch it and get its detailed
information. All applications available on the system are shown on the View page. By clicking the application logo, the user is directed to the application’s Invoke page; he can then use that application to watch existing derivative content and create more derivative content.

<table>
<thead>
<tr>
<th>title</th>
<th>filename</th>
<th>uploaded on</th>
</tr>
</thead>
<tbody>
<tr>
<td>ben reads postcard</td>
<td>bengpostcard01</td>
<td>2001-05-16</td>
</tr>
<tr>
<td>ben writes a postcard</td>
<td>benwriting01</td>
<td>2001-05-15</td>
</tr>
<tr>
<td>ben writes a postcard</td>
<td>benwriting0201</td>
<td>2001-05-15</td>
</tr>
<tr>
<td>ben tears the postcard</td>
<td>benwritingtear01</td>
<td>2001-05-15</td>
</tr>
<tr>
<td>meg chases david and ben</td>
<td>boys_megchase01</td>
<td>2001-05-15</td>
</tr>
<tr>
<td>meg david ben in the beach</td>
<td>boys_megchase0201</td>
<td>2001-05-15</td>
</tr>
<tr>
<td>bracelet on the table</td>
<td>bracelet01</td>
<td>2001-05-15</td>
</tr>
<tr>
<td>chelsea is waiting</td>
<td>chelswait01</td>
<td>2001-05-15</td>
</tr>
<tr>
<td>Meg Leaves the Game</td>
<td>cardsMegleave01</td>
<td>2001-05-15</td>
</tr>
</tbody>
</table>

Figure 1-5: User Profile

The Home page shows and updates the user profile (Figure 1-5). A user can join or leave an open community by indicating his desire on the add/remove page (Figure 1-6). He can create a new open or closed community by typing the name and description of the featured community. If the user is an administrator of a particular community, he is able to modify the user list of this community, assign more administrators and change the community’s description.

A Develop page (Figure 1-7) allows a developer to test a new application or update an existing application. The details will be discussed in Section 3.2.
Account

Membership in communities

Add yourself to the following open communities:

- [ ] honeymoon
  - reset add

Remove yourself from the following open communities:

- [ ] asdf
- [ ] test
  - reset remove

To create a new community

Community Name:

Community Type: Open

Description:

- reset update

Figure 1-6: Membership in Communities

You have administrative privileges on the following applications. You can update the status of an application by clicking on its thumbnail:

Online:
- individeo
- pluss shorts

Figure 1-7: Developers' Page to Add/Update Applications

21
Chapter 2

Architectural Design Issues

According to the primary objectives, we designed a 3-tier architecture. In this chapter, we first introduce the system data model. We will then give an architectural overview and design analysis, followed by a discussion of the related works.

2.1 System Data Model

The SMS data are categorized into four types: Media, User, Attribute and Applications.

2.1.1 Media

There are primary media objects, called MediaElements, and secondary media objects, called MediaCollections in the system. All MediaElements are submitted as solitary binary multimedia objects (audio/video clips, images and text). They are considered atomic in the SMS. Applications let users construct MediaCollections from a set of MediaElements. The role of each MediaElement in a MediaCollection is stored in MediaProperties. Possible MediaProperties of a MediaElement for a story-making application are PositionInSequence or Duration. Figure 2-1 shows the relationship between these objects.

After users upload MediaElements onto the SMS, the system needs to validate the
binaries. All the recognizable multimedia types are called \textit{MediaType}. For example, the SMS knows how to stream \textit{QuickTime} movie files, but does not yet know \textit{Real} format files. Applications may put constraints on the MediaTypes. For example, MediaElements cannot be too large in size or resolution. Those constraints are called \textit{MediaConstraints}.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{media-object-hierarchy.png}
\caption{The Media Object Hierarchy}
\end{figure}

\subsection{User}

Users can not only submit video clips and watch shared clips online, but also can develop their own applications and upload them onto the system. Therefore we can specify two general user roles: \textit{regular user} and \textit{developer}. Regular users are ordinary consumers who wish to access Shareable Media for fun or work. They can search through existing Media objects, add their own MediaElements and create MediaCollections. Users wishing for additional features beyond the scope of existing applications can take the initiative and write their own program interfaces. These developers may work independently on the SMS and eventually deploy their work as a new application.

Above the user level, \textit{Communities} are formed. Communities represent users of similar interest who require a little privacy in order to work closely together. Each
community can define itself as either opened or closed. It can be administrated by one or more users. Any user can add himself to an open community, while only community administrator can update the membership of a closed community. One user can join any number of communities.

Media which is uploaded to the system can be similarly designated. A user can set his clip public, so that anyone can watch and reuse it, or limit his clip to use by a subset of communities in which this user is currently a member. If public, the user grants the copyright of this clip to the SMS. If one clip is limited to a set of communities upon the time of submission, the copyright is held by those communities jointly. In other words, if the author of the clip leaves those communities, he will no longer be able to access that clip even though he is the author.

UserActions provide user activities for the SMS captures. For example, users can rate or write reviews of online clips. These actions are defined by applications that utilize the statistical record kept by the database.

\section*{2.1.3 Application}

An interface that receives user requests and displays related information is called an Application. Applications may be They are either system built in or user defined. System applications handle low-level and basic tasks such as user registration, user login, and MediaElement submission; user-defined applications make use of the low level API and provide higher level uses for the system. These applications are dynamically linked onto the SMS. The first time, a developer creates a new application name on the SMS, he fills out a form detailing conditions and requirements, and specifying a few attributes; he then submits the client-side, server-side and two-side-shared programs onto the system. Developers can intentionally take these off-line or update them after that.
<table>
<thead>
<tr>
<th>MediaType</th>
<th>Available Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attribute ID</td>
</tr>
<tr>
<td>MediaConstraints</td>
<td>INT_0</td>
</tr>
<tr>
<td></td>
<td>INT_1</td>
</tr>
<tr>
<td></td>
<td>INT_2</td>
</tr>
<tr>
<td></td>
<td>INT_3</td>
</tr>
<tr>
<td>MediaElement</td>
<td>STR50_0</td>
</tr>
<tr>
<td></td>
<td>STR250_0</td>
</tr>
<tr>
<td>MediaProperties</td>
<td>INT_0</td>
</tr>
<tr>
<td>MediaCollection</td>
<td>STR50_0</td>
</tr>
<tr>
<td></td>
<td>STR250_0</td>
</tr>
<tr>
<td></td>
<td>STR250_1</td>
</tr>
<tr>
<td></td>
<td>STR2000_0</td>
</tr>
</tbody>
</table>

Table 2.1: Examples of Customized Attributes

<table>
<thead>
<tr>
<th>AttributeType</th>
<th>DataType</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>Integer</td>
<td>0</td>
</tr>
<tr>
<td>LONG</td>
<td>Long</td>
<td>0</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>Double</td>
<td>0</td>
</tr>
<tr>
<td>DATE</td>
<td>Date</td>
<td>0</td>
</tr>
<tr>
<td>STR50</td>
<td>String</td>
<td>50</td>
</tr>
<tr>
<td>STR250</td>
<td>String</td>
<td>250</td>
</tr>
<tr>
<td>STR2000</td>
<td>String</td>
<td>2000</td>
</tr>
</tbody>
</table>

Table 2.2: Currently Available Attributes' Data Types
2.1.4 Attribute

Since the SMS is an open system, new applications may require new types of information to be added into the database. Attributes represent customized columns in the common data model (Table 2.1). Similar to MediaTypes for MediaElements, AttributeTypes define all possible data types we deal with in the customized columns (Table 2.2).

Applications can share information stored in some customized columns but give them different meanings ("name"). As a result we introduce AttributeMapping for applications to identify customized columns.

2.2 Architectural Design and Analysis

To handle the objects in the data model, we develop a 3-tier architecture, shown in Figure 2-2. This roughly follows the standard protocol. The first tier is the Data Tier, containing the database and the file system. The second tier is the Business Logic Tier, and comprises the three components: Access Control Unit, Shareable Media Framework, and Storage Manager. Finally, the third tier is the Presentation Tier. The main web site is our System application. Application Manager and all user-defined applications are linked to the System. User-defined applications can only directly access the Control Unit.

This design has three distinct advantages for application development:

1. Modularity: There are three modules in the business logic tier. Applications remain unaware of the low-level database schema through the Access Control Unit.

2. Extensibility: Application Manager helps developers bring applications online and debug their programs with a test database.

3. Scalability: Storage Manager is the only component to access database. We don’t need to modify any other modules to expand the current centralized database system to a distributed server system or even a peer-to-peer system.
Figure 2-2: The Shareable Media 3-Tier architecture
2.3 Related Work

A significant amount of research exists on the design of scalable systems that can serve multiple distributed multimedia applications or devices.

A good system for application development is the DAVE model, developed by Mines et al in [12]. DAVE implements an application programming interface, a connection manager, and an object manager with device objects to demonstrate a plug and play design for distributed multimedia applications. Using this model, application developers can easily develop distributed multimedia applications and create reusable multimedia toolkits. Analogous to DAVE, our architecture aims to provide "plug and play" system so that application developers can put their tools online. However, DAVE lacks a defined server structure; applications use the powerful DAVE API only to handle data transfers between connected devices.

The work of Blum and Molva [4] is similar to the DAVE model but also introduces the concept of an 'application pool.' As the center of control and coordination, the application pool resembles the framework of the SMS and invokes applications. However, it fails to address the issue of adding, modifying, or removing applications.

During 1998 and 1999, our group focused on designing a system for asynchronous sharing of streaming video on Internet and built a prototype, I-Views [13]. It allows widely distributed groups of people to view, edit, compare, evaluate and discuss video material over the Internet. In I-Views, the design and implementation of server side modules are primarily based on the client side needs. The architecture of I-Views is not able to easily deploy new applications.
Chapter 3

System Implementation

The SMS was implemented using Java 2 Enterprise Edition (J2EE) [16] v1.3 API. The business logic modules are Enterprise JavaBeans (EJBs) [15]. The system application and the application manager was programmed using Extensible Markup Language (XML) [18] and Java Servlets [17].

3.1 Business Logic Modules

Among the three modules in the business logic tier, Access Control Unit interacts with user-defined applications; Storage Manager queries or executes SQL statements; Shareable Media Framework contains all entities in the data model.

3.1.1 Storage Manager

While the Storage Manager is blind to the database schema, it is the only gateway to the data tier. In other words, no other components know where data is stored and how many mirrors the server can access.

In the current version, Storage Manager is a stateless session bean, simply passing SQL statements down to a centralized database. Nevertheless, it can flexibly accommodate the addition of functionality to the Storage Manager without changing code in any other components. This provides a rich research area for future system
improvement. For example, one storage manager can access distributed databases and optimize data transfer to the user.

The storage manager is not the focus of our current research. Firstly, it has little impact on the extensibility of the SMS. Secondly, much research has been done in this area and we can use an existing model. Finally, increasing Internet speed reduces the severity of this problem.

3.1.2 Shareable Media Framework

The SMS provides an API for application developers. It contains four main packages: Attribute, User, Application and Media. These are the four types of data models we discussed in Chapter 2. Figure 3-1 shows the dependencies between these packages. All these objects were implemented as entity beans.

![Figure 3-1: The Dependencies of Business Logic Modules](image)

One Attribute instance represents a customized column in a database table. For example, the column known to the System application as clip.title is a customized column. Its real name is MEDIAELEMENT.STR50.0. Its AttribType is String of size 50. Figure 3-2 shows how the attributes are used in the Application Manager.
The User and Community are customizable. Certain applications may store some user-specific or community-specific information. Therefore, they are dependent on the Attribute.

An Application does not have any customized information and there is no interaction between two online applications. The developer can create an attribute-mapping for his application (Figure 3-2).

The Media package wraps up all attributes attached to a media object and hides internal representations and mappings.

3.1.3 Access Control Unit

UserSession and Search are two stateful session beans that Servlets from the presentation tier can create themselves. They are gateways to ensure no security holes. For instance, a user should not be able to find a media clip whose access is limit to the “EastCampus” community if he is not a member of that community. To be certain, he does not have right to access the MediaElementHome interface directly. A Search or UserSession object can return a MediaElement to him if he is indeed allowed to watch the clip.

In the implementation of Search and UserSession we check the user’s access right before a Shareable Media Framework instance is returned to the user.

The Access Control Unit validates the community (privacy) restrictions, the user roles for developing (developer vs. regular user) and user roles for community administration (administrator vs. ordinary member).

Be noted that the community restriction for any derived media content (MediaCollection instances are inherited from the limitation on the MediaElement level). For example, a MediaCollection contains two MediaElements, one is open to “EastCampus” and “RandomHouse”; the other is open to “LCS” and “MediaLab.” Accordingly, a user must be a member of either “EastCampus” or “RandomHouse,” as well as a member of either “MediaLab” or “LCS,” if he wants to read this MediaCollection object. This check is non-trivial. And it has implications for system users who are members of more than one community [7].
3.2 Application Manager

The Application Manager, a crucial component of the SMS, monitors the application pool. Developers put applications online through a process of registering, testing, and releasing. To handle this, the Application Manager responds to several issues such as compiling, attribute handling, application states, and access control.

3.2.1 Compilation

Prior to submitting the program code, a developer compresses all server-side source code into a package. He does the same to all client-side code, code shared in to server side and client side, and binary code (i.e. images used by this application’s web pages). He then logs onto the Shareable Media, goes to the Develop page, and clicks to create an application with a unique name. He selects an appropriate compiler (default set to Java JDK v1.3), an image file for the logo, and all compressed files. The Application Manager will first store these to the database, and then attempt to compile the application. A list of compiling error messages will be reported on the web if the compiler fails.

3.2.2 Attributes

New applications can define their attributes for customizable beans, such as MediaElement, MediaCollection, and so on. Attributes are informative tags attached to an object that the application finds significant, like time, location, or characters. The Application Manager updates an internal attribute map after developers register application attributes. Thus, when receiving a call from applications later on, the framework can modify and retrieve data by examining the attribute map and making queries.

A problem arises when developers use the same word to describe different attributes in different contexts. For example, the “location” of a video clip in different applications may have different meanings in the database. Some applications care about nations and can more accurately use the “country” attribute. Another appli-
cation may consider GPS coordinates to be a location. This problem is compounded when development of a new application occurs prior to registration, so the developers lack a set naming scheme for their attributes.

To solve this problem, we introduce an attribute map that creates a binding between the name developers use in an application and the associated data in the database, shown in Figure 3-2. To minimize the possibility of duplicate attributes, the application manager displays the existing attributes with detailed descriptions before developers can register new attributes. Attribute creation requires the developer to provide the name used by the application, a detailed description, and data type (AttributeType).

```
Attribute A:  table.column = MEDIAELEMENT.STR250_0
Application appl:
    appl.createAttributeMapping (attr, "keywords");
Application appl2:
    appl2.createAttributeMapping (attr, "clip_keywords");

To get the data in MEDIAELEMENT.STR250_0
for MediaElement mel:
In appl: mel.getAttribute (appl, "keywords");
In appl2: mel.getAttribute (appl, "clip_keywords");
```

Figure 3-2: Attribute Mapping Example

### 3.2.3 Application States

There are three states for an application: testing, online and offline. In the testing state, applications are registered in a testing server. Data access is limited to a separated testing database. In the online state, applications are open to all users and have access to the main database. Finally, in the off-line state, applications are invisible to users but retain their derived information in the database. Developers can change application states if the Application Manager verifies that the new mode is applicable.
At the first time, the developer is directed to submit his code in the testing mode on a testing machine. After a few days, if the application is running well and has passed required system tests, it is designated as ready to put online. This happens in the next interval during which there is no active user on the system.

To upgrade applications, developers are required to update source code and add, remove, or modify application attributes. The system maintains two versions of one application – one in the testing state and the other in the released state. If the old version is inactive, the new one will replace it in the testing state. Re-releasing an existing application will replace the old active version.

For all applications in the system, the application manager provides the test log and statistical information for developers.

3.2.4 Access Control

Access Control is an issue for applications as well. How do we prevent one application from getting and modifying data from another application? A temporary solution is to pass the application name to many functions in the API. However, it is not secure.

Another solution is to force application developers to set the application name as a constant in a specific file. When all source code is submitted, the SMS scans the source and reinforces the correct application name in the parameter list. Another solution is to give the developer a private key. Applications have to pass the private key in order to have access to the requested data.

All the server side programs are by default accessible to any request. We need to create a unique role for each application or pass private keys as part of any request in order to make sure all server side programs only handle appropriate requests.

3.3 Shareable Media System Main Website

The System application runs on http://shareable.media.mit.edu, the Shareable Media’s main website. It handles clip and user management, invokes the application
manager as well as all user-defined online applications.

```
userSession = SMSession.(sessionId, "userSession");
xml = userSession.getPage(cargo); // gets XML from UserSession
SMSession.put(sessionId, page, xml); // puts the xml into Session
```

Redirect to

```
WebDriverId = sessionId & page=
```

StaticPage

```
funcName = env.get(page);
xml = SMSession.(sessionId, page).
```

Non-Static

```
xm1 = invokeMethod(funcName);
```

XSL.parse(xml)

```
final html page
```

Figure 3-3: Generating Web Pages in System Application

Initial implementations of Shareable Media used a simple two tier architecture. All the web pages are generated from XSL (Extensible Style Language) template and XML documents. Figure 3-3 shows how the SMS automatically generate all web pages.

We made a persistent SMSession to simulate HttpSession between different servlets. There is a garbage collector to clean up SMSession.

### 3.4 Project Development History

The SMS implementation has been through three major stages.

#### 3.4.1 Early Prototype

Initial implementations of Shareable Media used a simple two tier architecture (Figure 3-4). This design allowed for quick server development. With each component
of the SMS modularized into a program that produced raw data, developers easily worked in parallel with each other. Although the model served well as a basic system, the support for multiple applications was minimal. Application developers had to work closely with the server administrators to deploy applications. The lack of a defined standard API resulted in duplicated code. Finally, schema changes became very costly where application procedures operated directly with the database. The current implementation of the SMA addresses each of the drawbacks in the initial system while retaining the modularity of architecture components.

This early prototype was developed on Microsoft Windows NT Server 4.0. We used Oracle 8i for database management and Oracle Application Server 4.0 as a web and application server. All software components were implemented using Sun’s Java technologies, mainly Enterprise JavaBeans (EJB). System applications such as user registration and media submission were developed using Java Servlets, Java Server Pages (JSP) and Java Applets.

3.4.2 FreeWare Environment

We reimplemented the Shareable Media Framework in J2EE from summer 2000. The 3-tier architecture gave us flexibility to accommodate different kinds of applications but the indirection and Storage Manager’s database block became a bottleneck.
The software used was Sun J2EE Server, W3C Jigsaw Servlet Engine, and Apache Web Server. The system was not sufficiently stable to keep running for an extended time. The speed performance was not satisfactory, especially when users browsed many movie clips.

3.4.3 Current Environment

The SMS currently runs on the Borland AppServer v4.5 [5], Tomcat3 [2] for Servlet and Apache [1] as the web server. Since the application server pools data objects and database connections, the running performance has been greatly speeded up.

For the system to be stable, data must be periodically backed up. The commercial product with online backup support, Oracle 8i combined with a third-party software running on the Windows NT Server. Due to the objected-oriented nature of the language and the well-developed enterprise API of J2EE, this was the natural choice. Although issues arise concerning the relative slowness of Java executables, the large size of media files make network connections the most significant bottleneck of the SMA.
Chapter 4

Text2Movie and Other Applications

Text2Movie is an application uses video clips available to the system to assist users in telling a visual story from a paragraph of text. Two other applications available on the Shareable Media system will also be described.

4.1 Text2Movie

4.1.1 User Scenario

Scenario Setup

Elaine has recently shot some video footage of her relatives at a family reunion. In particular, she captured lots of fun moments of her adorable 3-year-old nephew Billy. Elaine by now has stock-piled a considerable amount of footage of her family, which she would love to put together and share with them. Elaine knows a little bit about capturing and editing so she picks and chooses some appropriate clips and converts them into QuickTime movies. She wonders what her whiz-kid cousin David would do with these clips, given his video-making experience from school and she would love to hear her grandmother’s comments about some of the clips featuring her many grandchildren. How could she do this?
Sending the QuickTime movies attached to emails isn’t really an option, as many of the files are quite large (> 2MB) and most of her family have mail accounts with strict attachment size limits. Posting them on the web could be another option, but Elaine doesn’t know too much about HTML or how she would go about setting up a host server. She also doesn’t really like the idea of putting movies of her family up on the web for all to see. These movies capture precious family moments that she would just like to share with her immediate relatives. Ideally Elaine would like it if her family could share their own footage, comments and stories with her in a some sort of secure environment that was readily accessible and fun to use.

4.1.2 Text2Movie’s Solution

The Shareable Media System allows users to create private communities to facilitate the controlled access of shared video files, which are hosted on the Shareable Media Server. Users of the system can upload short video clips to the server, specify which communities they want their clips to be accessible to (including starting a new community where they can then monitor membership of that community). Once the videos have been uploaded to the server (the user suggests a title for the clip, some keywords/background info etc.) - then the fun really begins!!

Text2Movie allows users to put together movie sequences to accompany their text messages, stories and comments. This novel approach to online movie-making utilizes the user-contributed text to drive a clip selection engine, based on keyword matching. Contextual information such as date, user name, time and location information is used to further refine the selection process. Using our example above, Elaine begins to type in her story in the text box about the recent family reunion as follows:

Hi there,

Well, I for one could not believe how much Billy had grown when I saw him at the reunion. Compared to last Christmas, he’s a real big boy now, or "man" as he corrected me there.

As Elaine inputs her story, the Text2Movie begins selection mechanism to assemble a collection of clips in the movie construction area, laying the clips out in a linear
fashion one after another as they are called by the keyword matching system. Elaine is first of all presented with some general footage of Billy carrying-on at the reunion, followed by a short clip from last Christmas that Uncle Simon had uploaded. This clip is followed by a clip that Elaine has linked directly to the text (<here>) of Billy proudly declaring that "I am not a boy - I'm a little man!!" Elaine continues to compose her story and the system tries to suggest meaningful clips to accompany her text. After she has finished typing Elaine can then preview her composition, play back the suggested movie sequence and then add to, delete or reorder the different movie clips. When she is happy with her composition, Elaine saves her work and logs out.

On Friday evening, all members in the family community receive an email about the recently created stories, Elaine's Little Man Billy includes. David reads the story and clicks the hyperlink to open a browser where he watches the movie playback. He likes Billy's innocent tone which reminds him of his own childhood. He starts to write his comments:

*could love to revisit my life twenty years or ten years age.*

Again Text2Movie suggests two groups of clips according to the time stamps and the active user David. Then David selects the funniest clips shot when he was a baby and a teenager.

When Granny watches these stories and comments, she is very excited and recalls the memorable moments with her grandchildren. Her comments are also interpreted as short movies.

### 4.1.3 Interface Design and Configuration

The main focus of the interface design is how to help the user find the right clip and navigate related clips. The user interface is shown in Figure 4-1. Text2Movie rates the popularity of clips and movies users have watched. It helps us know more about users' personalities. Figure 4-2 and Figure 4-3 show the the Attribute and Servlet Settings Text2Movie is configurated.

The Text2Movie is still in development to try out more heuristic functions in order
Media Attributes:

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Attribute</th>
<th>Description of Current Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIA_COLLECTION</td>
<td>SEQ_TOTALSCORE</td>
<td>the overall scores of the collection</td>
</tr>
<tr>
<td></td>
<td>SEQ_RATINGCOUNTER</td>
<td>the total number of ratings for a collection</td>
</tr>
<tr>
<td></td>
<td>SEQ_TEXT</td>
<td>story text to describe the movie, used by text2movie</td>
</tr>
<tr>
<td></td>
<td>SEQ_TITLE</td>
<td>the title for a movie</td>
</tr>
<tr>
<td>PROPERTIES</td>
<td>ORDER</td>
<td>the film's order in the movie</td>
</tr>
<tr>
<td>MEDIAELEMENT</td>
<td>CLIP_TOTALSCORE</td>
<td>the overall scores of the rating of 0 to 6</td>
</tr>
<tr>
<td></td>
<td>CLIP_TITLE</td>
<td>the title for a media element</td>
</tr>
<tr>
<td></td>
<td>CLIP_KEYWORDS</td>
<td>The keywords for a media element, delimited by \01</td>
</tr>
<tr>
<td></td>
<td>CLIP_RATINGCOUNTEF</td>
<td>the total number of ratings</td>
</tr>
</tbody>
</table>

Figure 4-1: Text2Movie's User Interface

Figure 4-2: Attribute Mapping for Text2Movie

Figure 4-3: Servlets Settings for Text2Movie
to correctly predict users’ preferences. It will work as a search interface for users to find their interested content.

4.2 Application Development

The two applications currently deployed on the SMA illustrate the flexibility of the architecture. MediaCollections used range from story-centered sequences of video clips to content-based retrieval of information. With each application, users have a level of control over the shared media that current commercial products do not support.

4.2.1 PlusShorts

PlusShorts[11, 10], a prototype application conceived by Ashling Kelliher, is a web-based application that allows a distributed group of users to contribute to, and collaborate in, the creation of shared movie sequences. PlusShorts uses punctuation as an iconic system for describing and augmenting edited video. The punctuation symbols are used to detail the structure of a video sequence and inspire dialogue about the essence of that structure. The intention of this application is to provide users with a method of sharing conceptual structures in an activity that is both playful and meaningful.

4.2.2 Individeo

Individeo[10], an application conceived by James Hung-Soon Seo, is a web-based tool that features an interface for browsing and editing sequences of video clips. The browser allows users to understand how the video is being reused by multiple storytellers. The editor lets users create expressive sequences with video and text. The goal of this application is to foster casual creative story-telling and messaging using video as a primary media. The system may also serve as a basis for collaborative cinematic productions.
4.2.3 Integration

Table 4.1: Servlets Registration Settings for Individeo

<table>
<thead>
<tr>
<th>Servlet Name</th>
<th>Servlet Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoke</td>
<td>edu.mit.media.ic.sm.individeo.servlet.InvokeServlet</td>
</tr>
<tr>
<td>SeqSave</td>
<td>edu.mit.media.ic.sm.individeo.servlet.SeqSaveServlet</td>
</tr>
<tr>
<td>AppSearch</td>
<td>edu.mit.media.ic.sm.individeo.servlet.AppSearchServlet</td>
</tr>
<tr>
<td>SMIL</td>
<td>edu.mit.media.ic.sm.individeo.servlet.SMILServlet</td>
</tr>
</tbody>
</table>

Both PlusShorts and Individeo have been integrated into the Shareable Media System. Table 4.1 shows how IndiVideo registers its Servlets on the server. A sample configuration page for Individeo is like what Figure 4-4 shows.
Chapter 5

Future Work

The SMA as outlined leaves room for improvement. Several sections of the SMA could be revised to improve service speed, application development, user interactions, and media handling.

5.1 Scalability

The scalability of the Shareable Media System can be significantly improved by making Storage Manager more powerful. Because of the flexibility in the 3-tier design, we can easily expand the functionality of the barely-implemented storage manager. Specifically, we can target two issues that will dramatically speed up content delivery to users: cache and distributed computing.

Our current implementation requires that we unload the media from the database into the file system before streaming it to the user. Keeping often-used media readily available saves server-side bandwidth from repetitive data transfer [6]. It is also important to explore the possibilities of caching media on client machines [8]. Further research is necessary to determine the effectiveness of using clients with fast Internet connections as small/partial data mirrors. This is not an unreasonable request for the user and can potentially decrease the load on a central server significantly.

There has been much research conducted into distributed multimedia applications [9]. Since the current data delivery is not fast enough for rich media content, it
brings the issue of possible server-side distributed computing. A network of both web
servers and database servers can optimize content delivery. One possible usage of
multiple web servers is to have each server ping a user on login and assign the fastest
machine to that user for the duration of the session. Later research can determine a
synchronization algorithm for the storage manager to use to ensure that all content
available is as current as possible regardless of the database server being used.

5.2 Application Development

Open source is an existing model to develop potentially high quality applications
by allowing a group of developers to collaborate on their projects. Additionally, if
not enough data exists or data is not correct in the test database, the application
manager will migrate data from the main database to the test one. It will also allow
developers to submit data directly to the test database.

5.3 Data Model Update

Our data model can be updated to support communities, or interest groups. One
possible community implementation is a content filter for members to focus their
work. Communities of developers can share source code and make group decision
about changing the state of the project.

5.4 Media Support

In the future, we can support more non-traditional, multimedia data types such
as Flash. We can also develop an intelligent agent to detect duplicated media content
to increase system integrity while minimizing wasted disk space.

It is possible to detect the device and connection speed rate between clients and
servers. A smart system will filter the media content to adapt media to the transfer
rate and satisfy users using a wide range of Internet connections [14]. For example,
if a user is watching a clip from her Palm, we want Shareable Media to only transfer low quality video or still thumbnails to represent the video.

5.5 Conclusions

The architecture we designed has many possibilities for future growth. In its current implementation it is flexible enough to accommodate the creativity of users and developers while keeping the internal database schema hidden. This is most effectively demonstrated by PlusShorts, Individeo, and Text2Movie. Shareable Media suggests a future in which creative shared applications can take advantage of shared media. The open nature of the system will allow us to test our hypothesis that developers of interface can interact with media makers to provide new potential media forms. Issues of community and permissions will no doubt become focal centers of the continuing research. is a viable project for the future Internet. Shared applications built on top of the shared media make this open system unique.
Bibliography


http://www.w3.org/Style/XSL/.