# An Extensible Architecture for Multiple Applications with Shareable Media

Cathy Lin, David Cheng, Pengkai Pan,
James Jung-Hoon Seo, Aisling Kelliher, Glorianna Davenport
Room E15-435 Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{llin, drcheng, ppk, jseo, aisling, gid}@media.mit.edu

November 1, 2000

## Abstract

*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 video clips. To adapt this to the rapidly evolving Internet, we need to design and experiment with an extensible architecture for Shareable Media, which 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 three modules: Application Manager, Shareable Media Framework, and Storage Manager. Through the Application Manager, application designers register, test, release, and monitor their products on top of the architecture. The Shareable Media Framework provides application programming interfaces that allow user-defined applications to access data in the system. Both modules retrieve and store content through the Storage Manager.*

*Currently, three applications are under development for the extensible architecture: PlusShorts, Individeo, and M-Views.*

## 1 Introduction

Rich media, such as audio and video, will inevitably be used and shared by many people in their daily life through the rapidly evolving Internet. We could imagine that, in the very near future, millions of hours of rich media content will be created by $200 wireless digital video cameras, edited on 1 GHz handheld computers and distributed over gigabyte fiber Internet connections. Nowadays, there are two basic models for sharing rich media content. The first is a server-client broadcast or video-on-demand model, such as Real Server-Player and Microsoft's Media Server-Player. The second is peer-to-peer data sharing model, such as Napster and Gnutella. In both models, the media objects being shared are treated as complete entities and simply transfer from one place to another. The two models are not able to foster interaction and collaboration among users, while also being ill-suited to the requirements of either learning or entertaining. Inspired by the "intercreativity" idea, a term coined by Tim Berners-Lee [1], 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 create easy-to-use models that facilitate sharing and reusing available content which exists on the Internet?

The Shareable Media project is an experimental project for exploring and addressing these questions. It is an effort to provide a coherent structure that will facilitate distributed collaboration and communication among filmmakers, storytellers, artists and audiences. The project explores how a community of users can share web-based stories and express ideas through a shared database of digital video/audio clips. Each user can contribute original material, which then becomes available to all other users.

To adapt this idea to the Internet, we need to design and experiment with an extensible architecture for Shareable Media. In particular, we focus on the following questions in this paper.

1. How can an extensible architecture be designed to deploy multiple applications on wired and wireless devices through connections of both broad

and narrow bandwidth?

2. How can we provide an interface for any developer to write novel applications for Shareable Media?

To address these two questions, we take an approach consisting of three modules: Application Manager, Shareable Media Framework and Storage Manager. Through the application manager, developers are able to register, test and release their applications on top of the architecture. The shareable media framework provides an application programming interface (API) for application developers. Through the framework, developers do not need to know the database schema. The storage manager is used by all other modules as a black box to retrieve and store content. In this paper, we focus on the application manager.

In the past year, we have built three prototype applications for Shareable Media: PlusShorts, Individeo and M-Views. PlusShorts and Individeo use various interfaces to involve the user in telling stories through video while M-Views is an application for wireless devices which continuously provides usage information of Shareable Media for mobile users. We present these applications to demonstrate the extensible capabilities of our architecture.

The paper is organized as follows. We first define the data model and describe the architecture with its three modules (Section 2). In Section 3, we discuss the implementation of the system. Then we refer to related works (Section 4), and outline possible extensions (Section 5). Finally, draw conclusions about our contributions.

## 2 Design

To build a system for shareable media, we create a Shareable Media Architecture (SMA). At the heart of the SMA is multimedia content, drawn from an eclectic source of individuals. Additionally, applications run on the SMA and exclusively use the database of shared media. The goal of the SMA is to efficiently manage media and user information while providing application designers with a powerful API for media manipulation.

### 2.1 Data Model

We can set four types of data for the SMA to manage: Media, Applications, Profiles, and Feedback.

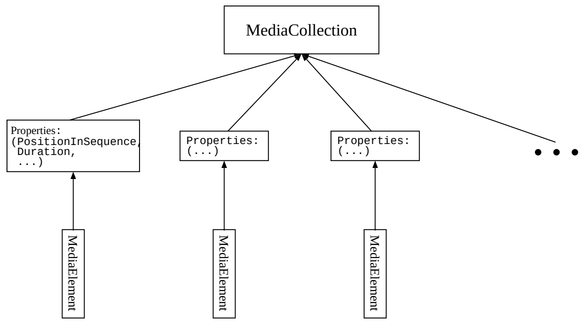

Figure 1: The Media Object Hierarchy

Media consists of *MediaElements*, *MediaCollections*, and *Properties*. These three interrelated Media objects are the atoms on which we base the SMA. Applications let users construct MediaCollections from MediaElements, which are any solitary multimedia objects (audio/video clips, images, and text). In doing this, it is often necessary for applications to define Properties that indicate the role of each MediaElement with respect to the entire MediaCollection. Possible Properties of MediaElements for a storymaking application are `PositionInSequence` or `Duration`. The object hierarchy of Media is shown in Figure 1.

Applications are either system or user-defined. Because system applications handle low-level and basic tasks such as user registration, user login, and media submission, we focus on the user-defined applications. These interact with an API that provides access to all the Media objects. They construct MediaCollections internally that the SMA can interpret and store.

Profiles are all user-related information. We can specify two types of Profiles: *User* and *Developer*. Users are ordinary consumers who wish to access Shareable Media for fun or work. They can search through existing Media objects and add their own MediaElements or create unique MediaCollections. Users wishing for additional features beyond the scope of existing applications can take the initiative and write their own programs. These Developers may work independently on the SMA and eventually deploy their work as a new application.

Feedback is any action performed on a Media object by a User. There are three types of Feedback: *UserAction*, *Rating*, and *Comments*. The UserAction feedback works without explicit control by the user. A MediaElement receives UserAction when a user views it or uses it in a MediaCollection. Users may wish to

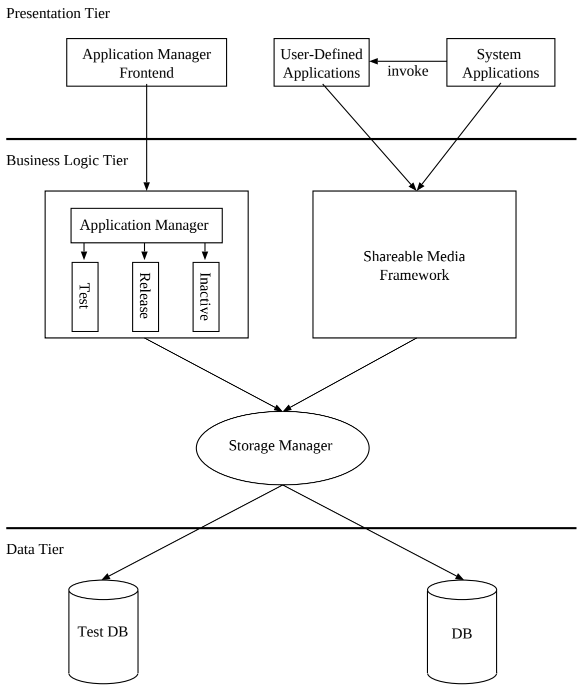

Figure 2: The Shareable Media 3-Tier architecture

provide a little more direct feedback by rating a Media object. Typically represented numerically, Ratings give the SMA a concrete measure of the Media quality. Finally, users can attach text in the form of Comments. These complex objects attached to Media objects can support threads to extend user collaboration.

## 2.2 Architecture

To handle the objects in the data model, we develop a 3-tier architecture, shown in Figure 2.

This design has two distinct advantages for application development:

1. Modularity. Applications remain unaware of the low-level database schema through the Shareable Media Framework.

2. Extensibility. A dedicated module helps developers bring applications online and debug their programs with a test database.

3. Scalability. With the Storage Manager as a black box, we have the ability to expand the system to a distributed environment.

This roughly follows the standard protocol. The first tier is the Data Tier, containing the database and the file system. The second tier contains the Business Logic, and comprises the three components detailed below. Finally, the third tier is the Presentation Tier, with applications that directly access only the Business Logic.

### 2.2.1 Application Manager

A crucial component of the SMA is the application manager. Developers use this to put applications online through a process of registering, testing, and releasing. To handle this, the application manager responds to several issues concerning compiling, attribute handling, application states, and access control.

*Compilation*

All source code and compiled application programs are stored in a centralized web server. After developers submit their source code, they select an appropriate compiler. The application manager will then attempt to compile the application and automatically email the developer with a status report: either a listing of error messages or a web address for further input.

*Attributes*

New applications can define their attributes for MediaElements, MediaCollections, and Properties. Attributes are informative tags attached to an object that the application finds significant, such as 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 application may consider "city" 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. 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.

*Application States*

There are three states for an application: testing, released and inactive. In the testing state, applications are only open for developers and invited users. Data access is limited to a separated testing database. In the released state, applications are open to all users and have access to the main database. Finally, in the inactive state, applications are invisible to users but retain their MediaCollections in the database. Developers can change application states if the application manager verifies that the new mode is applicable.

The testing mode is the default mode. When in a testing state, an application is only accessible through a testing URL. The system will keep the testing application for some time and then recover the testing system unless its developer asks for an extension.

After the system internally verifies that the testing application's MediaCollections can be stored and retrieved without errors, it allows the developer to release the application to the public. After a release, applications can still be tested but additionally have the ability to be upgraded or deactivated.

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 test logs and statistical information for developers.

*Access Control*

Developers of one application can grant users the right to read data, write data, or open application interface packages. The predefined system applications handle administrative tasks such as user registration, login, and content submission. All built-in applications have full access to user-defined applications. This is necessary for them to launch applications and specify parameters. However, user-defined applications have limited access to areas such as user information or application management.

### 2.2.2 Shareable Media Framework

The SMA provides an API for application developers. It contains four packages: Media package, Utility package, Profile package, and Feedback package.

*Media Package*

The media package is designed to wrap up all attributes attached to a Media object and hide internal representations and mappings. For example, if an application receives a MediaElement as a search result, it can access all its attributes, such as content, thumbnail, keywords, or authors. *MediaElement*, *MediaCollection*, *Property* are all included in the media package.

*Utility Package*

Applications call the framework to save and retrieve information. The utility package contains beans to carry out these operations. For sample, applications can call *SearchBean* to search through all MediaElements or *SaveCollectionBean* to save its representation of MediaCollection objects.

*Profile Package*

The SMA makes profiles for users and developers. The profile package contains interfaces to query and update data on those profiles. Interfaces are designed not only for user-defined applications but also for system applications. However, the user-defined applications will have limited access to some profile methods.

*Feedback Package*

The feedback package easily reflects the data model for the Feedback data type. The package provides logging functionality to construct the UserAction feedback. Additionally, methods can render user input of Rating systems into meaningful numerical values. Finally, Comments are grouped in different threads.

### 2.2.3 Storage Manager

The application manager and the shareable media framework access and update content through the storage manager while it knows nothing about the database schema.

We design the storage manager as the only gateway to the data layer. It allows us to add more functionality to the storage manager without requiring code changes in any other components. As a result, it leaves 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 SMA. Secondly, much research has been done in this area and we can use an existing model. Finally, increasing Internet speed is reducing the severity of this problem.

## 3 Implementation

We implemented the system based on the design architecture discussed in the previous sections. The prototype was developed on Microsoft Windows NT Server 4.0. We used Oracle 8*i* 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. The web server supports common server-side technology: CORBA, JSP, Servlets, CGI (Common Gateway Interface) and EJB. Therefore the system accommodates major web applications.

For a system to be stable, backing up data periodically is necessary. The commercial product with online backup support is Oracle 8*i* 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 about the relative slowness of Java executables, the large size of media files make network connections the more significant bottleneck of the SMA.

### 3.1 Application Development

The three 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.

#### 3.1.1 PlusShorts

PlusShorts is a web-based application that allows a distributed group of users to contribute to, and collaborate upon, 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 level of interactivity that is both playful and meaningful.

#### 3.1.2 Individeo

Individeo is a web-based tool that features interfaces 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 storytelling and messaging using video as a primary media. The system may also serve as a basis for collaborative cinematic productions.

#### 3.1.3 M-Views

M-Views enables users to construct and share video communication using the shareable database over a wireless Internet connection. The project emphasizes mobile and place sensitive applications.

## 4 Related Works

A significant amount of research has already been conducted to build scalable systems that can serve multiple distributed multimedia applications or devices.

A better system for application development is the DAVE model, developed by Mines *et al* in [6]. It 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 at providing a "plug and play" interface for application developers to 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 [2] 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 SMA and invokes applications. However, it fails to address the issue of adding, modifying, or removing applications.

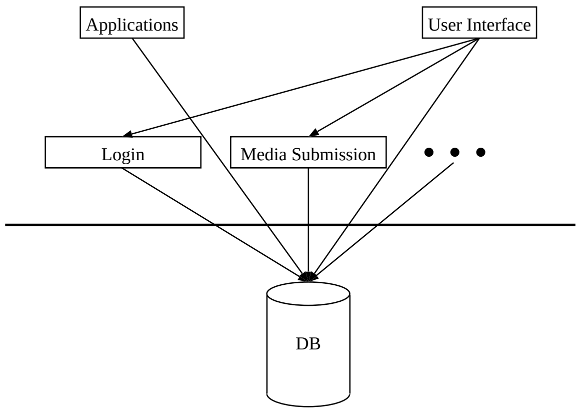

Figure 3: The Shareable Media 2-Tier architecture (Prototype)

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 [7]. 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 deploy multiple applications in an easy way.

### 4.1 Early Prototype

Initial implementations of Shareable Media used a simple two tier architecture (Figure 3). This design allowed for quick server development. With each component of the SMA modularized into a program that produced raw data, developers easily worked in parallel with each other. Although the model served well for 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 of the initial system while retaining the modularity of architecture components.

## 5 Future Work

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

### 5.1 Scalability

Although the current model for the SMA is significantly more efficient than the prototype of Section 4.1, it lacks scalability as implemented. 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 filesystem before streaming it to the user. Keeping often-used media readily available saves server-side bandwidth from repetitive data transfer [3]. It is also important to explore the possibilities of caching media on client machines [4]. 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 [5]. 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

We will 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 [8]. 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.

## 6 Conclusions

The architecture we designed has many possibilities for future growth. Nevertheless, it is currently 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 M-Views. Shareable Media is a viable project for the future Internet. Shared applications built on top of the shared media make this open system unique.

## Acknowledgments

We would like to thank all members of the Shareable Media Team. Emery Lin, Jocelyn Lin, and Daniel Talavera have been working intensively to develop the Shareable Media project. 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.

## References

[1] Tim Berners-Lee. Keynote address. In *the Fifth International World Wide Web Conference*, May 1996.

[2] C. Blum and R. Molva. A CORBA-based platform for distributed multimedia appliations. In *Proceedings of Multimedia Computing and Networking*, February 1997.

[3] A. Dan and D. Sitaram. Multimedia caching strategies for heterpgeneous application and server environments. *Multimedia Tools and Applications*, 4(3), May 1997.

[4] S. G. Dykes, C. L. Jeffery, and S. Das. Taxonomy and design analysis for distributed web caching. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, Jan 1999.

[5] E. Frecon and M. Stenius. DIVE: A scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal*, 1998.

[6] R. F. Mines, J. A. Friesen, and C. L. Yang. DAVE: A plug and play model for distributed multimedia application development. In *Proceedings of the Second ACM International Conference on Multimedia*, 1994.

[7] P. Pan and G. Davenport. A community-oriented system for sharing streaming video on the internet. In *Proceedings of the Ninth World Wide Web Conference*, 2000.

[8] J.R. Smith, R. Mohan, and C. Li. Scalable multimedia delivery for pervasive computing. In *Proceedings of the Seventh ACM International Conference on Multimedia*, 1999.