6

information system architecture

From a system development perspective, a multimedia information system may be considered as a multimedia database, providing storage and retrieval facilities for media objects. Yet, rather than a solution this presents us with a problem, since there are many options to provide such storage facilities and equally many to support retrieval. In this chapter, we will study the architectural issues involved in developing multimedia information systems, and we will introduce the notion of media abstraction to provide for a uniform approach to arbitrary media objects. Finally, we will discuss the additional problems that networked multimedia confront us with.

6.1 architectural issues

The notion of *multimedia information system* is sufficiently generic to allow for a variety of realizations. Let's have a look at the issues involved.

As concerns the database (that is the storage and rerieval facilities), we may have to deal with homegrown solution, commercial third party databases or (even) legacy sources. To make things worse, we will usually want to deploy a combination of these.

With respect to the information architecture, we may wish for a common format (which unifies the various media types), but in practice we will often have to work with the native formats or be satisfied with a hybrid information architecture that uses both media abstractions and native media types such as images and video.

The notion of media abstraction, introduced in [MMDBMS], allows for uniform indexes over the multimedia information stored, and (as we will discuss in the next section) for query relaxation by employing hierarchical and equivalence relations.

Summarizing, for content organisation (which basically is the information architecture) we have the following options:

content organisation

- *autonomy* – index per media type
- *uniformity* – unified index
- *hybrid* – media indexes + unified index

In [MMDBMS], a clear preference is stated for a uniform approach, as expressed in the *Principle of Uniformity*:

Principle of Uniformity

> *... from a semantical point of view the* content *of a multimedia source is independent of the source itself, so we may use statements as meta data to provide a description of media objects.*

Naturally, there are some tradeoffs. In summary, [MMDBMS] claims that: metadata can be stored using standard relational and OO structures, and that manipulating metadata is easy, and moreover that feature extraction is straightforward. ¡/lp¿ Now consider, is feature extraction really so straightforward as suggested here? I would believe not. Certainly, media types can be processed and analysis algorithms can be executed. But will this result in meaningful annotations? Given the current state of the art, hardly so!

research directions– *the information retrieval cycle*

When considering an information system, we may proceed from a simple generic software architecture, consisting of:

software architecture

- a database of media object, supporting
- operations on media objects, and offering
- logical views on media objects

However, such a database-centered notion of information system seems not to do justice to the actual support and information system must provide when considering the full information retrieval cycle:

information retrieval cycle

1. specification of the user's information need
2. translation into query operations
3. search and retrieval of media objects
4. ranking according to likelihood or relevance
5. presentation of results and user feedback
6. resulting in a possibly modified query

When we look at older day information retrieval applications in libraries, we see more or less the automation of card catalogs, with search functionality for keywords and headings. Modern day versions of these systems, however, offer graphical userinterfaces, electronic forms and hypertext features.

When we look at the web and how it may support digital libraries, we see some dramatic changes with respect to the card catalogue type of applications. We can now have access to a variety of sources of information, at low cost,

including geographically distributed resources, due to improved networking. And, everybody is free to make information available, and what is worse, everybody seems to be doing so. Hence, the web is a continuously growing repository of information of a (very) heterogeneous kind.

Considering the web as an information retrieval system we may observe, following [IR], that:

- despite high interactivity, access is difficult;
- quick response is and will remain important!

So, we need better (user-centered) retrieval strategies to support the full information retrieval cycle. Let me (again) mention someof the relevant (research) topics:*user interfaces, information visualisation, user-profiling and navigation.*

6.2 media abstractions

Let's have a closer look at media abstractions. How can we capture the characterization of a variety of media types in one common media abstraction. A definition of such a media abstraction is proposedin [MMDBMS]. Leaving the formal details aside, a media abstraction has the following components:

media abstraction

- *state* – smallest chunk of media data
- *feature* – any object in a state
- *attributes* – characteristics of objects
- *feature extraction map* – to identify content
- relations – to capture state-dependent information
- (inter)relations between 'states' or chunks

Now, that characterization is sufficiently abstract, and you may wonder how on earth to apply this to an actual media database.

However, before giving some examples, we must note that the *feature extraction map* does not need to provide information about the content of a chunk of media data automatically. It may well be a hand-coded annotation.

Our first example is an image database.

example – image database

states: { pic1.gif,...,picn.gif }
features: names of people
extraction: find people in pictures
relations: left-of, ...

In an image database it does not make much sense to speak about relations between 'states' or chunks of media data, that is the images.

For our next example though, video databases, it does make sense to speak about such relations, since it allows us to talk about scenes as sequences of frames.

example – video database

states: set of frames
features: persons and objects
extraction: gives features per frame
relations: frame-dependent and frame-independent information
inter-state relation: specifies sequences of frames

Now, with this definition of media abstractions, we can define a simple multimedia database, simply as

simple multimedia database

- a finite set M of media abstractions

But, following [MMDBMS], we can do better than that. In order to deal with the problems of *synonymy* and *inheritance*, we can define a structured multimedia database that supports:

structured multimedia database

- *equivalence relations* –to deal with synonymy
- *partial ordering* – to deal with inheritance
- *query relaxation* – to please the user

Recall that we have discussed the relation between a 'house of prayer' and 'church' as an example of synonymy in section 4.3. As an example of inheritance we may think of the relation between 'church' and 'cathedral'. Naturally, every cathedral is a church. But the reverse does not necessarily hold. Having this information about possible equivalence and inheritance relationships, we can relax queries in order to obtain better results. For example, when a user asks for cathedral in a particular region, we could even notify the user of the fact that although there are no cathedrals there, there are a number of churches that may be of interest. (For a mathemathical characterization of structured multimedia databases, study [MMDBMS].)

query languages Having media abstractions, what would a query language for such a database look like? Again, following [MMDBMS], wemay extend SQL with special functions as indicated below:

SMDS – functions

Type: object $\mapsto$ type
ObjectWithFeatures: $f \mapsto \{o \mid \text{object } o \text{ contains } f\}$
ObjectWithFeaturesAndAttributes: $(f, a, v) \mapsto \{o \mid o \text{ contains } f \text{ with } a = v\}$
FeaturesInObject: $o \mapsto \{f \mid o \text{ contains } f\}$
FeaturesAndAttributesInObject: $o \mapsto \{(f, a, v) \mid o \text{ contains } f \text{ with } a = v\}$

Having such functionswe can characterize an extension of SQL, which has been dubbed SMDS-SQL in [MMDBMS], as follows.

SMDS-SQL

SELECT – media entities

- m – if m is not a continuous media object
- $m : [i, j]$ – m is continuous, i, j integers (segments)
- $m.a$ – m is media entity, a is attribute

FROM

- <media><source><M>

WHERE

- term IN funcall

As an example, look at the following SMDS-SQL snippet.

example

```
SELECT M
FROM   smds source1 M
WHERE  Type(M) = Image AND
       M IN ObjectWithFeature("Dennis") AND
       M IN ObjectWithFeature("Jane") AND
       left("Jane","Dennis",M)
```

Note that M is a relation in the image database media abstraction, which contains one or more images that depict Jane to the left of Dennis. Now, did they exchange the briefcase, or did they not?

When we do not have a uniform representation, but a hybrid representation for our multimedia data instead, we need to be able to: express queries in specialized language, and to perform operations (joins) between SMDS and non-SMDS data.

Our variant of SQL, dubbed HM-SQL, differs from SMDS-SQL in two respects: function calls are annotated with media source, and queries to non-SMDS data may be embedded.

As a final example, look at the following snippet:

example HM-SQL

```
SELECT M
FROM smds video1, videodb video2
WHERE M IN smds:ObjectWithFeature("Dennis") AND
    M IN videodb:VideoWithObject("Dennis")
```

In this example, we are collecting all video fragments with Dennis in it, irrespective of where that fragment comes from, an (smds) database or another (video) database.

research directions– *digital libraries*

Where *media abstractions*, as discussed above, aremeant to be technical abstractions needed for uniform access to media items, we need quite a different set of abstraction to cope with one of the major applications of multimedia information storage and retrieval: digital libraries.

According to [IR], digital libraries will need a long time to evolve, not only because there are many technical hurdles to be overcome, but also because effective digital libraries are dependent on an active community of users:

digital libraries

> *Digital libraries are constructed – collected and organized – by a community of users. Their functional capabilities support the information needs and users of this community. Digital libraries are an extension, enhancement and integration of a variety of information institutions as physicalplaces where resources are selected, collected, organized, preserved and accessed in support of a user community.*

The occurrence of digital libraries on the web is partly a response to advances in technology, and partly due to an increased appreciation of the facilities the internet can provide. From a development perspective, digital libraries may be regarded as:

> *... federated structures that provide humans both intellectual and physical access to the huge and growing worldwide networks of information encoded in multimedia digital formats.*

Early research in digital libraries has focussed on the digitization of existing material, for the preservation of our cultural heritage, as well as on architectural issues for the 'electronic preservation', so to speak, of digital libraries themselves, to make them "immune to degradation and technological obsolescence", [IR].

To bring order in the variety of research issues related to digital libraries, [IR] introduces a set of abstractions that is know as the 5S model:

digital libraries (5S)

- *streams*: (content) – from text to multimedia content
- *structures*: (data) – from database to hypertext networks
- *spaces*: (information) – from vector space to virtual reality
- *scenarios*: (procedures) – from service to stories
- *societies*: (stakeholders) – from authors to libraries

These abstractions act as "a framework for providing theoretical and practical unification of digital libraries". More concretely, observe that the framework encompasses three technical notions (streams, structures and spaces; which correspond more or less with data, content and information) and two notions related to the social context of digital libraries (scenarios and societies; which range over possible uses and users, respectively).

For further research you may look at the following resources:

D-Lib Forum – http://*www.dlib.org*
Informedia – http://*www.informedia.cs.cmu.edu*

The D-Lib Forum site gives access to a variety of resources, including a magazine with background articles as well as a test-suite that may help you in developing digital library technology. The Informedia site provides an example of a digital library project, with research on, among others, video content analysis, summarization and in-context result presentation.

6.3 networked multimedia

For the end user there should not be much difference between a stand-alone media presentation and a networked media presentation. But what goes on *behind the scenes* will be totally different. In this section, we will study, or rather have a glance at, the issues that play a role in realizing effective multimedia presentations. These issues concern the management of resources by the underlying network infrastructure, but may also concern authoring to the extent that the choice of which media objects to present may affect the demands on resources.

To begin, let's try to establish, following [Networked], in what sense networked multimedia applications might differ from other network applications:

networked multimedia

- real-time transmission of continuous media information (audio, video)
- substantial volumes of data (despite compression)
- distribution-oriented – e.g. audio/video broadcast

Naturally, the extent to which network resource demands are made depends heavily on the application at hand. But as an example, you might think of the retransmission of television news items on demand, as nowadays provided via both cable and DSL.

For any network to satisfy such demands, a number of criteria must be met, that may be summarized as: throughput, in terms of bitrates and burstiness; transmission delay, including signal propagation time; delay variation, also known as jitter; and error rate, that is data alteration and loss.

For a detailed discussion of criteria, consult [Networked], or any other book on networks and distributed systems. With respect to distribution-oriented multimedia, that is audio and video broadcasts, two additional criteria play a role, in particular: multicasting and broadcasting capabilities and document caching. Especially caching strategies are of utmost importance if large volumes of data need to be (re)transmitted.

Now, how do we guarantee that our (networked) multimedia presentations will come across with the right quality, that is free of annoying jitter, without loss or distortion, without long periods of waiting. For this, the somewhat magical notion of *Quality of Service* has been invented. Quoting [Networked]:

Quality of Service

> Quality of Service *is a concept based on the statement that not all applications need the same performance from the network over which they run. Thus, applications may indicate their specific requirements to the network, before they actually start transmitting information data.*

Quality of Service (*QoS*) is one of these notions that gets delegated to the other parties, all the time. For example, in the MPEG-4 standard proposal interfaces are provided to determine *QoS* parameters, but the actual realization of it is left to the network providers. According to [Networked] it is not entirely clear how *QoS* requirements should be interpreted. We have the following options: we might consider them as hard requirements, or alternatively as guidance for optimizing internal resources, or even more simply as criteria for the acceptance of a request.

At present, one thing is certain. The current web does not offer *Quality of Service*. And what is worse, presentation formats (such as for example *flash*) do not cope well with the variability of resources. More specifically, you may get quite different results when you switch to another display platform

virtual objects

Ideally, it should not make any difference to the author at what display platform a presentation is viewed, nor should the author have to worry about low-quality or ill-functioning networks. In practice, however, it seems not to be realistic to hide all this variability from the author and delegate it entirely to the 'lower layers' as in the MPEG-4 proposal.

Both in the SMIL and RM3D standards, provisions are made for the author to provide a range of options from which one will be chosen, dependent on for example availability, platform characteristics, and network capabilities.

A formal characterization of such an approach is given in [MMDBMS], by defining *virtual objects*.

virtual objects

- $VO = \{(O_i, Q_i, C_i) \mid 1 \leqslant i \leqslant k\}$

where

- $C_1, \ldots, C_k$ – mutually exclusive conditions
- $Q_1, \ldots, Q_k$ – queries
- $O_1, \ldots, O_k$ – objects

In general, a virtual object is a media object that consists of multiple objects, that may be obtained by executing a query, having mutually exclusive conditions to determine which object will be selected. Actually, the requirement that the conditions are mutually exclusive is overly strict. A more pragmatic approach would be to regard the objects as an ordered sequence, from which the first eligible one will be chosen, that is provided that its associated conditions are satisfied.

As an example, you may look at the Universal Media proposal from the Web3D Consortium, that allows for providing multiple URNs or URLs, of which the first one that is available is chosen. In this way, for instance, a texture may be loaded from the local hard disk, or if it is not available there from some site that replicates the Universal Media textures.

networked virtual environments

It does seem to be an exageration to declare *networked virtual environments* to be the ultimate challenge for networked multimedia, considering that such environments may contain all types of (streaming) media, including video and 3D graphics, in addition to rich interaction facilities. (if you have no idea what I am talking about, just think of, for example, Quake or DOOM, and read on.) To be somewhat more precise, we may list a number of essential characteristics of networked virtual environments, taken from [VE]:

networked virtual environments

- *shared sense of space* – room, building, terrain
- *shared sense of presence* – avatar (body and motion)
- *shared sense of time* – real-time interaction and behavior

In addition, networked virtual environments offer

- *a way to communicate* – by gesture, voice or text
- *a way to share ...* – interaction through objects

Dependent on the visual realism, resolution and interaction modes such an environment may be more or less 'immersive'. In a truly immersive environment, for example one with a haptic interface and force feedback, interaction through objects may become even threathening. In desktop VEs, sharing may be limited to the shoot-em-up type of interaction, that is in effect the exchange of bullets.

Networked virtual environments have a relatively long history. An early example is SIMNET (dating from 1984), a distributed command and control simulation developed for the US Department of Defense, [VE]. Although commercial multi-user virtual communities, such as the *blaxxun* Community server, may also be ranked under networked virtual environments, the volume of data exchange needed for maintaining an up-to-date state is far less for those environments than for game-like simulation environments from the military tradition. Consider, as an example, a command and control strategy game which contains a variety of vehicles, each of which must send out a so-called *Protocol Data Unit* (PDU), to update the other participants as to their actual location and speed. When the delivery of PDUs is delayed (due to for example geographic dispersion, the number of participants, and the size of the PDU), other strategies, such as *dead reckoning*, must be used to perform collision detection and determine possible hits.

To conclude, let's establish what challenges networked virtual environments offers with respect to software design and network performance.

challenges

- *network bandwidth* – limited resource
- *heterogeneity* – multiple platforms
- *distributed interaction* – network delays
- *resource management* – real-time interaction and shared objects
- *failure management* – stop, ..., degradation
- *scalability* – wrt. number of participants

Now it would be too easy to delegate this all back to the network provider. Simply requiring more bandwidth would not solve the scalability problem and even though adding bandwidth might allow for adding another hundred of entities, smart updates and caching is probably needed to cope with large numbers of participants.

The distinguishing feature of networked virtual environments, in this respect, is the need to

manage dynamic shared state

to allow for real-time interaction between the participants. Failing to do so would result in poor performance which would cause immersion, if present at all, to be lost immediately.

research directions– *architectural patterns*

Facing the task of developing a multimedia information system, there are many options. Currently, the web seems to be the dominant infrastructure upon which to build a multimedia system. Now, assuming that we chose the web as our vehicle, how should we approach building such a system or, in other words, what architectural patterns can we deploy to build an actual multimedia information system? As you undoubtly know, the web is a document system that makes a clear distinction between *servers* that deliver documents and *clients* that display documents. See [OO], section 12.1. At the server-side you are free to do almost anything, as long as the document is delivered in the proper format. At the client-side, we have a generic document viewer that is suitable for HTML with images and sound. Dependent on the actual browser, a number of other formats may be allowed. However, in general, extensions with additional formats are realized by so-called *plugins* that are loaded by the browser to enable a particular format, such as *shockwave*, *flash* or *VRML*. Nowadays, there is an overwhelming number of formats including, apart from the formats mentioned, audio and video formats as well as a number of XML-based formats as for example SMIL and SVG. For each of these formats the user (client) has to download a plugin. An alternative to plugins (at the client-side) is provided by Java *applets*. For Java applets the user does not need to download any code, since the Java platform takes care of downloading the necessary classes. However, since applets may be of arbitrary complexity, downloading the classes needed by an application may take prohibitively long.

The actual situation at the client-side may be even more complex. In many cases a media format does not only require a plugin, but also an applet. The plugin and applet can communicate with eachother through a mechanism (introduced by Netscape under the name LiveConnect) which allows for exchanging messages using the built-in DOM (Document Object Model) of the browser. In addition, the plugin and applet may be controlled through Javascript (or VBscript). A little dazzling at first perhaps, but usually not to difficult to deal with in practice.

Despite the fact that the web provides a general infrasructure for both (multimedia) servers and clients, it might be worthwhile to explore other options, at the client-side as well as the server-side. In the following, we will look briefly at:

- the Java Media Framework, and
- the DLP+X3D platform

as examples of, respectively, a framework for creating dedicated multimedia applications at the client-side and a framework for developing intelligent multimedia systems, with client-side (rich media 3D) components as well as additional server-side (agent) components.

Java Media Framework The Java platform offers rich means to create (distributed) systems. Also included are powerful GUI libraries (in particular, Swing), 3D libraries (Java3D) and libraries that allow the use and manipulation of images, audio and video (the Java Media Framework). Or, in the words of the SUN web site:

http://java.sun.com/products/java-media

> *The JavaTM Media APIs meet the increasing demand for multimedia in the enterprise by providing a unified, non-proprietary, platform-neutral solution. This set of APIs supports the integration of audio and video clips, animated presentations, 2D fonts, graphics, and images, as well as speech input/output and 3D models. By providing standard players and integrating these supporting technologies, the Java Media APIs enable developers to produce and distribute compelling, media-rich content.*

However, although Java was once introduced as the *dial tone of the Internet* (see [OO], section 6.3), due to security restrictions on applets it is not always possible to deploy media-rich applets, without taking recourse to the Java plugin to circumvent these restrictions.

DLP+X3D In our DLP+X3D platform, that is introduced in section **??** and described in more detail in appendix **??**, we adopted a different approach by assuming the availability of a generic X3D/VRML plugin with a Java-based External Authoring Interface (EAI). In addition, we deploy a high-level ditributed logic programming language (DLP) to control the content and behavior of the plugin. Moreover, DLP may also be used for creating dedicated (intelligent) servers to allow for multi-user applications.

The DLP language is Java-based and is loaded using an applet. (The DLP jar file is of medium size, about 800 K, and does not require the download of any additional code.) Dua, again, to the security restrictions on applets, additional DLP servers must reside on the site from where the applet was downloaded.

Our plugin, which is currently the *blaxxun* VRML plugin, allows for incorporating a fairly large number of rich media formats, including (real) audio and (real) video., thus allowing for an integrated presentation environment where rich media can be displayed in 3D space in a unified manner. A disadvantage of

such a unified presentation format, however, is that additional authoring effort is required to realize the integration of the various formats.

questions

information system architecture

1. (*) What are the issues in designing a *(multimedia) information system architecture*. Discuss the tradeoffs involved.

concepts

2. What considerations would you have when designing an architecture for a multimedia information system.
3. Characterize the notion of *media abstraction*.
4. What are the issues in *networked multimedia*.

technology

5. Describe (the structure of) a video database, using *media abstractions*.
6. Give a definition of the notion of a *structured multimedia database*.
7. Give an example (with explanation) of querying a *hybrid multimedia database*.
8. Define (and explain) the notion of *virtual objects* in *networked multimedia*.