[SHRS90] *Ralf Steinmetz, Reinhard Heite, Johannes Rückert, Bernd Schöner;* **Compound Multimedia Objects - Integration into Network and Operating Systems;** International Workshop on Network and Operating System Support for Digital Audio and Video, International Computer Science Institute, Berkeley, CA, USA, 8.-9. November 1990.

Compound Multimedia Objects - Integration into Network and Operating System

Ralf Steinmetz, Reinhard Heite, Johannes Rückert, Bernd Schöner

IBM European Networking Center,
Tiergartenstr. 8, 6900 Heidelberg, P.O.Box 10 30 68,
tel: +49 6221 404280, FAX: +49 6221 404450
e-mail: STEINMET at DHDIBM1 (BITNET)
Germany

Abstract

Developing a distributed multimedia application in an efficient and comfortable way requires the support of "compound multimedia objects" (CMO).

A CMO can be created by an application as a composition of multiple individual objects. Having its own set of methods (comprising operations and control structures) it also contains the definition of relations among its objects and operations, the so-called script. The CMO satisfies the application's needs to handle and manage various multimedia resources within an application specific context in an uniform way. The components of such a CMO may be distributed within the network and represent distinct media resources.

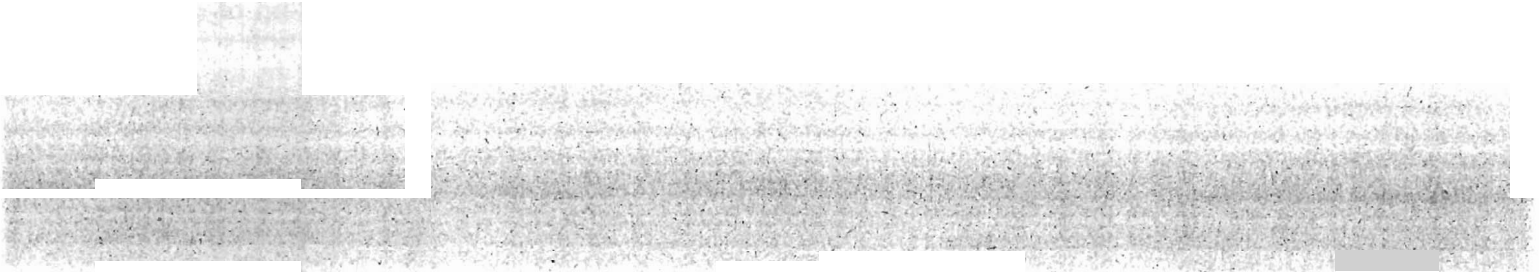
In the following we will give a short overview of the DiME (Distributed Multimedia Environment) prototype, motivate and explain the needs for CMOs. Subsequently, we will outline our object model and show briefly the status of the project.

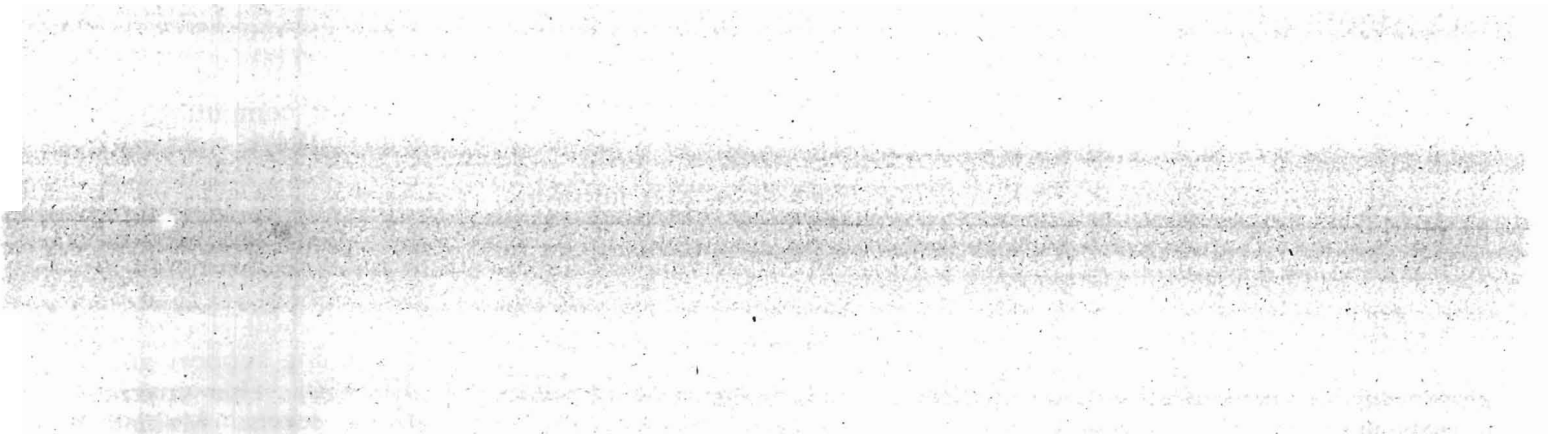
Introduction

With the trend towards powerful workstations integrating multimedia resources into the computing environment [Pres90], new types of applications like Computer-Supported Cooperative Work combined with multimedia arise. In a network environment it requires control, communication and manipulation of multimedia data and resources of different types at various locations. The well known aspects of heterogeneity and distribution gain relevance anew, since the required bandwidth for communication of multimedia data (e.g. digital video and audio) and their control pose new requirements to the involved communication networks and operating systems.

One of the primary objectives of the DiME project is to provide a suitable programming support and communication services for distributed multimedia applications [SSSW90; RSSS90]. This covers heterogeneity, presentation and distribution transparency aspects at various levels. Making heterogeneity transparent concerns hardware architectures, operating systems, communication facilities, devices and data encoding.

Digital video and audio as continuous data streams with real-time requirements need isochronous transmission via the network and continuous delivery of these data to the sink. On the opposite, ordinary data can be transmitted and delivered as a whole using asynchronous data transmission. The composition of these information types requires asynchronous, synchronous and isochronous data communication, remote control of devices and data flow. Key issues are synchronization between data streams and processes [Stei90], delay and bandwidth management [Ferr90]. The use of CMOs relieves the applications from these tasks. Some issues like simultaneous presentation of text, audio and video derived from different distributed sources involve data delay management and the control of the appropriate sources. Such a task is heavilydependent on the specificly used devices and networks. But, as it is a common concern to be solved for many applications, it should be a component of a distributed system





service and is part of DiME. Another motivation for the use of CMOs relates to the existence of objects which always comprise various media as for example some clips are usually played as video and audio together or a video conferencing device is made up of many individual devices.

Compound Multimedia Objects

A multimedia application comprises the interaction between information coded in various continuous and time independent media. Within an object oriented view we assume the application being a multimedia object consisting of many objects representing different media and devices. Such a **compound multimedia object (CMO)** is composed of other CMOs and **basic multimedia objects (BMO)**. The **basic multimedia class (BMC)** typically represents a single medium, of input (e.g. camera, stored audio sequence) or output type (e.g. video window, loudspeaker) dealing with either transient or persistent data. In the case of e.g. an image retrieved from the fixed disk, the properties "persistence", "input type" and the "image" medium are combined. So far we distinguish between the media text, image, audio and video. But, there are certainly further media like spreadsheet data or drawing which can be added within the scope of BMCs. The use of BMOs as sources and sinks of media streams is another of their essential properties. Concerning the class hierarchy of BMC we may choose any of the properties as the top level classification. In Figure 1 the sink/source/sink_and_source property is the primary characteristic, whereas the media is defined at the next level. Note, these sink/source and media related classes are abstract classes in the object oriented sense. Many BMOs derived from the same BMC may exist at the same time.

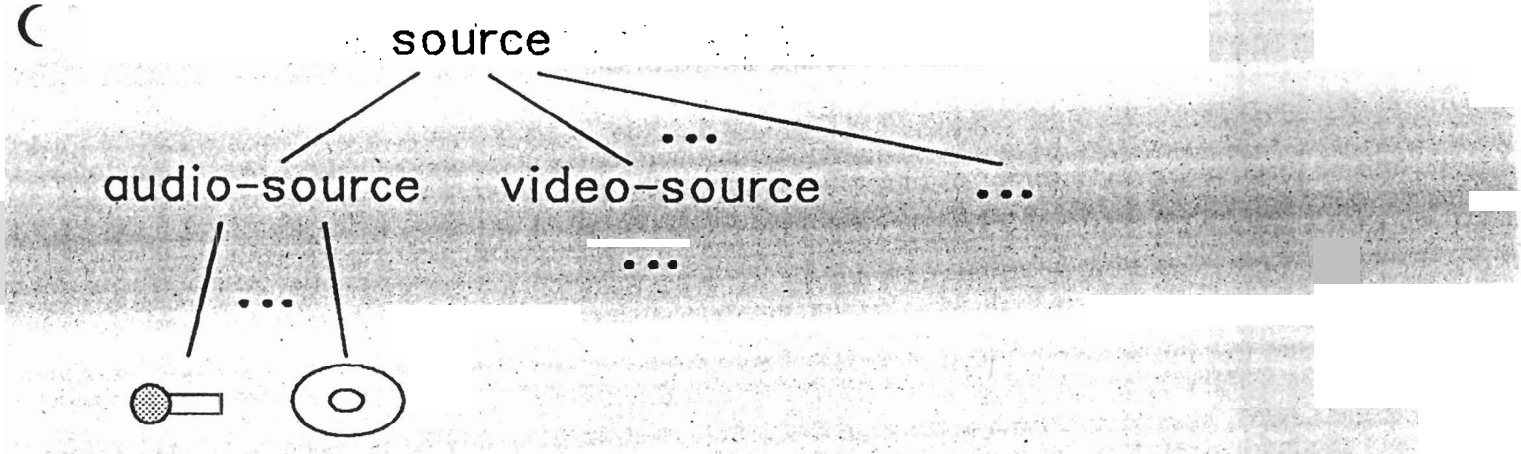


Figure 1. basic multimedia class hierarchy

For a BMO, the **data object** part specifies the binding, i.e. the file or device and its location. Therefore BMOs are location bound. It makes no sense to transfer these objects to remote locations, but objects referring to these BMOs at any other locations may exist. As shown in the example the binding is specified in the DATA section (see Appendix).

Each BMO has a set of object **methods** available, depending on it's class. Some frequently used methods within the scope of continuous media are "play" and "stop". The methods supported for such an BMO are provided as a system service and are based on the system calls. The state of certain multimedia objects like those identifying audio and video is varying continuously, because the actual data like the images are only valid for a specific period. Therefore some of the methods perform time depending actions like "slow motion", "hold scene for 2 seconds" or "increase volume".

BMOs contain some **event** processing, in order to inform another object of a certain condition like it's start or end. They are used to signal situations which can occur asynchronously generated within the object. Events are handled by special methods. A BMO bound to a continuous data stream has it's own well defined time scale and time events namely its start, end and user defined events in between. For each BMO the combination between events and their effects is specified in the EVENTS section.

The task of data delivery and its transmission via the network is done by a multimedia specific extension of the operating system. This requires interpretation of the objects control information, initiating the routing of different data parts from the specified distributed locations via appropriate network services to the location, where the object's data is to be delivered. Taking into account the time constraints specified in the respective multimedia class, message delay and bandwidth management of the underlying network(s) must be handled. Data itself is delivered by **access_points objects** from where applications can route the multimedia data to local resources for storage, presentation or processing. Access points for BMOs are objects comprising not only location addresses but also protocols of how data is communicated. They contain all source and sink specific information necessary for connecting them. For basic objects access points like VIDEO_SOURCE, VIDEO_SINK may exist.

The connection between sources and sinks is realized as follows:
A **channel object** is created, which has methods to connect sources and sinks to it. After creation sources and sinks may be connected to the channel by invoking connect of the appropriate channel, thus realizing a n_to_m connection. Access rights are granted by getting the right to invoke the connect of the channel. Using this mechanism sinks and sources can be connected to the channel independently. Depending on the channel realization single or multiple sources may be allowed. With a single source the usage is like a TV broadcast channel: A channel is created, the data of a source object switched onto it, and every authorized user may connect to the channel and "listen" to the program controlled by the user of the source object. In the more general case of multiple sources, the channel has to act as a combiner of their data and make it available to the sinks in an appropriate way, e.g. add voice samples or assemble video data as a collage. With the usage of channel objects we are free to change connections between multimedia objects dynamically. Therefore channel objects have "connect" and "disconnect" methods. To fully integrate BMOs, this multimedia connection management is to be realized as an extension of the communication system. Especially isochronous data transfer for time dependent object data is to be addressed in addition to asynchronous data transfer for other data.

So far BMOs represent only one device or a single medium. If we investigate existing technology like DVI (digital video interactive) data, we find out, that it supports not only individual media in an independent fashion, but also combines different media like audio and video (AVSS data format). The respective driver within the operating system also provides commands which affect both media. This brings us to the necessity to incorporate some device dependent media combinations in order to make use of such hardware. Also an application may want to define such additional classes referring to objects of the BMC and objects of already existing **compound multimedia classes (CMC)**.

An application can define CMCs using BMCs or other CMCs. It names the component types forming the object, specifying DATA and ACCESS_POINTS and defining its own METHODS and EVENTS. Using the CMCs, an application can create a CMO and afterwards specify the binding of the DATA parts of the involved BMOs. Associated with the object might be information about lifetime, time scale synchronization, access rights, distribution aspects and others.

The methods of the CMOs are defined by the applications using methods of all component objects and some additional private methods. For example, a private method for timed coordination of methods of some components may be provided, e.g. execute in parallel or serial.

Similar to the method specification, events of CMOs are defined. Typically methods of component objects like "play(object)", "stop(object)" are invoked, or an event is send to other objects like "inform_object(event)". Events are a call-back like mechanism. Using the methods and events an application can build up a script specifying the relationship between objects and use this script to present multimedia information.

A very important aspect concerns distribution: **Compound multimedia objects (CMO)** may consist of other CMOs or BMOs located at different computing nodes. The "connect" method of the channel object provides the means for data communication between these nodes. For method invokation a transparent mechansim for calling local as well as remote objects is to be provided. Thus, CMOs may be distributed in the network and can be accessed transparently. Since the owner and user of an object may differ, security issues are important: Access to remote object parts will only be granted by the system if appropriate access rights have been received from the object owner.

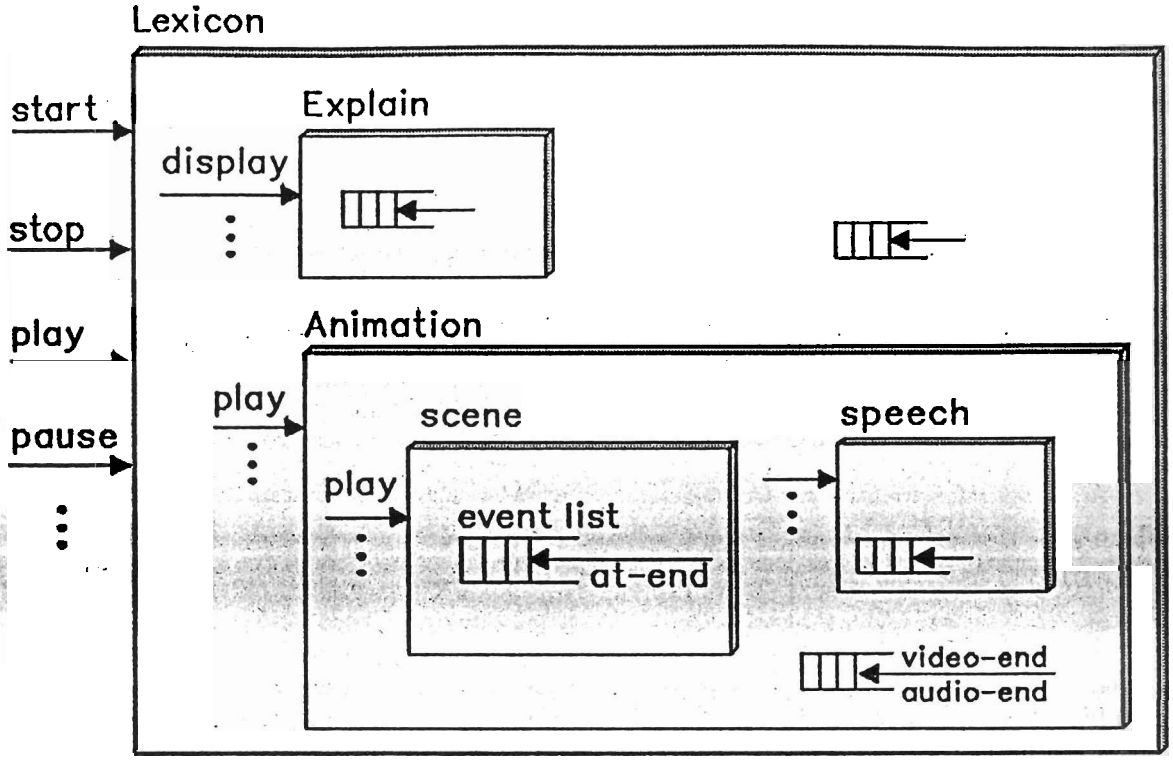


Figure 2. example of a CMO

As a simple example consider the class of a CMO named Lexicon as shown in Figure 2 and in the appendix. Note, the syntax used in this example is just for explanation purposes. A CMO consists of four sections named DATA, ACCESS_POINTS, METHODS and EVENTS. The example shows the used objects. Data of BMO are abbreviated as _filename. If "started", Lexicon "displays" some text from the text object Explain. Then the user may call the operation "play" within an application, which invokes the according method of a CMO Lexicon. "Play" starts an audio/video sequence of the object Animation. An application may allow the user to start the operations "pause", "stop" or "play". Then the appropriate methods of Lexicon are invoked. Animation is a composition of audio and video objects. Let us assume the video and audio parts are not of the same length. Therefore some actions have to be performed, if one of them finishes before the other. The method "play" of Animation starts the video Scene and audio Speech in parallel. If audio finishes, it signals "audio_end" to Animation. If video finishes, it "displays" the last picture of the video sequence and signal video_end to Animation. All events within the Animation object cause the actual medium to stop, perform some medium dependent operation and wait for the other involved object. Subsequently the Scene is "stopped". This type of audio/video synchronization is the restricted blocking as described in [Stei90].

In the same manner an appropriate CMO as sink can be specified. It may consist of data output devices like windows, monitors or loudspeakers for presentation purposes, as well as files or external devices for storage or processing purposes.

The DiME Prototype

To gain experience with distributed multimedia systems and to proof our proposed concepts we started to build a prototype [RSSS90]. The first step was to assemble a multimedia laboratory which consists of multimedia equipment controlled by conventional workstations interconnected via a LAN. For our lab we currently use PS/2 systems as workstations and have additional off-the-shelf remote controllable audio/video equipment like audio/video switches, VCR, CD-player, optical memory, cameras, monitors, microphones and loudspeakers. Special purpose hardware for full-motion video display, frame grabbing, audio storage, processing and presentation is integral part of the workstation as plug-in boards. The network services will make use of a transport system build on top of a high-speed LAN. For a first experiment, as there was no high-speed network with transport service available for multimedia data

streams available, a configuration based on interconnected audio and video switches in parallel to a token ring was used.

So far, local control applications running under the Presentation Manager in OS/2 for all external devices were implemented and tested. One demo application was ported to AIX. The first version of DiME uses the DACNOS IPC mechanism [GeHo90] for both local and remote communication. It is planned to move to the RPC of OSF DCE [OSFDCE90]. We are currently implementing audio and video servers which will be available as BMOs within DiME. An object manager dealing with the various aspects described in this paper is currently being designed. It is a repository for all types of objects at the application programming interface. The individual and combined data stream connections will be handled by cooperating audio/video managers at each workstation. At the time being first distributed demonstration are also running. Our next steps comprise the provision of adequate isochronous communication facilities on top of high-speed networks.

References

[Ferr90] *Domenico Ferrari:* **Client Requirements for Real-Time Communication Services;** ICSI, Berkley CA, Mar 1990, pp. 1-15.

[GeHo90] *Kurt Geihs, Ulf Hollberg:* **Retrospective on DACNOS;** Communications of the acm, vol. 33, no. 4, pp. 439-448, 1990.

[OSFDCE90] *The Open Systems Newsletter,* **OSF chooses the technologies for its distributed computing environment,** Vol.4, No.6, June 1990.

[Pres90] *Larry Press:* **Compute or Teleputer?;** Comm. of the ACM, Vol. 33, No. 9, September 1990, pp.29-36.

[RSSS90] *Johannes Rückert, Hermann Schmutz, Bernd Schöner, Ralf Steinmetz:* **a Distributed Multimedia Environment for Advanced CSCW Applications;** IEEE Multimedia '90, Bordeaux, 15-17 November 1990.

[SSSW90] *Ralf Steinmetz, Hermann Schmutz, Bernd Schoener, Michael Wasmund:* **Generic Support for Distributed Multimedia Applications;** IEEE ICC 90, Atlanta, April 1990.

[Stei90] *Ralf Steinmetz:* **Synchronization Properties in Multimedia Systems;** IEEE Journal on Selected Areas in Communication, vol.8, no.4, April 1990, pp.401-412.

Appendix: Example

```
Lexicon:  compound_object;
 DATA:    Explain external;
          Animation external;
 ACCESS_POINTS: VIDEO_SOURCE Animation.VIDEO_SOURCE;
                AUDIO_SOURCE Animation.AUDIO_SOURCE;
                TEXT_SOURCE  Explain.TEXT_SOURCE;
 METHODS:
  start: display(Explain);
  play:  play(Animation);
  pause: pause(Animation);
  stop:  stop(Animation);
  ...
 EVENTS:
  ...
```

```
Animation:  compound_object;
  DATA:     Speech  external;
            Scene external;
  ACCESS_POINTS: VIDEO_SOURCE Scene.VIDEO_SOURCE;
                 AUDIO_SOURCE Speech.AUDIO_SOURCE;
  METHODS:
   play:  play(Speech),  play(Scene)  in_parallel;
   pause: pause(Speech), pause(Scene) in_parallel;
   stop:  stop(Speech),  stop(Scene)  in_parallel;
   ...    ...
  EVENTS:
   audio_end: wait(video_end); stop(Scene);
   video_end: wait(audio_end); stop(Scene);
   ...

Scene:    basic_object;
  DATA:     VIDEO_filename at node_1;
  ACCESS_POINTS: VIDEO_SOURCE;
  METHODS:
   play;
   pause;
   stop;
   ...
  EVENTS:
   at_end:  display(LAST_PICTURE), inform_PARENT (video_end);
   ...      ...

Speech:     basic_object;
  DATA:     AUDIO_filename at node_2;
  ACCESS_POINTS: AUDIO_SOURCE;
  METHODS:
   play;
   pause;
   stop;
   ...
  EVENTS:
   at_end:  inform_PARENT (audio_end);
   ...      ...

Explain:    basic_object;
  DATA:     TEXT_filename at node_1;
  ACCESS_POINTS: TEXT_SOURCE;
  METHODS:
   display;
   ...
  EVENTS:
   ...
```