Incorporating Multimedia in Distributed Object-Oriented Systems: the Importance of Flexible Management

Gordon S. Blair and Nigel Davies

Distributed Multimedia Research Group,
Lancaster University, U.K.

Abstract
The integration of new media types into open distributed systems presents a number of problems. In this paper we discuss these problems within the context of an experimental distributed multimedia system being designed at Lancaster. Object management is highlighted as a vital component of our system, and we outline an approach to management termed management by exception. This approach is illustrated by considering the provision of persistence and location transparent invocation in our system.

1. Introduction
Multimedia computing has emerged in the last few years as a major area of research. This work is motivated by the wide range of potential applications made feasible by combining information sources such as voice, graphics, hi-fi quality audio and video. Most of the work to date has centred on the standalone multimedia workstation, and a variety of software packages are now available in areas such as music composition, computer-aided learning and interactive video.

The combination of multimedia computing with distributed systems support offers even greater potential. A wide variety of application areas for distributed multimedia systems have been identified such as office information systems, scientific collaboration, conferencing systems and distance learning. However, distributed multimedia systems are still at an early stage of development and many of the proposed applications have yet to be realised.

Due to the nature of these application areas, it is likely that future distributed multimedia environments will be heterogeneous; consisting of a number of different workstations interconnected by one or more types of network. With this inherent heterogeneity, it is of particular importance that distributed multimedia systems are open, and that their design takes full account of standardisation work such as the ODP initiative of the ISO [8].

Research at Lancaster is focusing on the design of open distributed systems to support multimedia. The research is based on the architecture developed by the ANSA/ISA project [2] which is a major contributor to the ODP standardisation process. This paper discusses the problems of introducing multimedia in a distributed object-oriented architecture. In particular, object management is highlighted as a critical area in the design of a distributed multimedia system.

Section 2 presents a brief introduction to the ANSA architecture. Section 3 describes our extensions to this architecture to cater for multimedia, highlighting the importance of maintaining a flexible approach to object management. Future work in the area is outlined in section 4, and section 5 contains our conclusions.

2. The ANSA/ISA project
The ANSA/ISA project aims to define a complete framework for the design and construction of distributed systems. The scope of the architecture is wide and takes on board the full range of issues from overall business objectives to detailed implementation choices. The complexity inherent in this broad view is managed by partitioning the architecture into five viewpoints:-

• Enterprise. This is concerned with distributed systems in the wide context of the organisations which exploit such computing systems. Its particular concerns include security, tariffs and domain management.

• Information. This is an abstract view on the detailed information processing requirements of an organisation or system.

• Computational. This is a programming language model of potentially distributed objects and their modes of interaction.

• Engineering. This sets out specifications, guidelines and concepts by which an abstract computational model may be realised at a systems level.
2.1. ANSA computational model

In the ANSA computational model, all interacting entities are treated uniformly as objects, i.e. encapsulations of state together with operations defined on that state. Objects are accessed through interfaces which define named operations together with constraints on their invocation. Interfaces are first class entities in their own right, and references to them may be freely passed around the system. Interface references are also the sole entities which may be passed to and from operations as arguments and results. Operations may return different combinations of types of interface reference in different circumstances: these are known as alternative terminations.

Services are made available for access by exporting an interface to a trader. The trader therefore acts as a database of services available in the system. Each entry in this 'database' describes an interface in terms of an abstract data type signature for the object and a set of attributes associated with the object. A client wishing to interact with a service interface must import the interface by specifying a set of requirements in terms of operations and attribute values. This will be matched against the available services in the trader and a suitable candidate selected. Note that an exact match is not required: ANSA supports a subtyping policy whereby an interface providing at least the required behaviour can be substituted. Finally, once an interface has been selected, the system can arrange a binding to the appropriate implementation of that object and thus allow operations to be invoked.

![Figure 1: Trading and Binding](image)

2.2. ANSA engineering model

ANSA has released a software system, known as ANSAware, which is a partial implementation of the computational model. In particular it does not enforce the computational model requirement that all operation arguments and results are interface references: most arguments and results are passed by value. In engineering terms, the ANSAware package is a fairly complete implementation of the ANSA engineering viewpoint as described in the ANSA Reference Manual (2).

To provide a platform conformant with the computational model, the ANSAware suite augments a general purpose programming language (usually C) with two additional languages. The first of these is IDL (interface definition language), which allows interfaces to be precisely defined in terms of operations as required by the computational model. The second language, prepc (preprocessed language) is embedded in a host language, such as C, and allows interactions to be specified between programs which implement the behaviour defined by these interfaces. Specifically, prepc statements allow the programmer to import and export interfaces, and to invoke operations in those interfaces.

In the engineering infrastructure, the binding necessary for such invocations is provided by a remote procedure call protocol known as REX (remote execution protocol). This protocol is layered on top of a generic transport layer interface known as a message passing service (MPS). A number of additional protocols may be included at both the MPS and the execution protocol levels and these may be combined in a number of different configurations. The infrastructure also supports lightweight threads within objects so that multiple concurrent invocations can be dealt with. ANSAware currently runs on a variety of operating system platforms including UNIX, VMS and MS-DOS.

3. An experiment in distributed multimedia computing

An experimental distributed multimedia system is currently under development at Lancaster based on the ANSAware platform. An object-oriented approach is adopted for this system. All services offered by the system are represented by encapsulated objects. Furthermore, this approach spans all levels of the system from basic multimedia devices such as cameras through to rich multimedia applications. The overall architecture of the system is illustrated in figure 2.

![Figure 2: Structure of the Application Platform](image)

This paper concentrates on the lower levels of this architecture, i.e. on the base objects and associated
management activities. Higher layers have however been instrumental in providing real application and end user requirements for distributed multimedia infrastructures. Results from this research can be found in [11] and [13].

3.1. Base services

The base services can be subdivided into three general categories as follows: storage services, multimedia devices and network services. Each of these areas is described in more detail below:

- **Network Services**

  The main task of the network services layer is to provide a range of protocols (of varying qualities of service) to handle multimedia communications. This required the provision of *stream* services through the ANSA architecture (this is in addition to the standard RPC protocols already supported by ANSA). The network services layer also provides a number of compression services which can be used to reduce the bandwidth requirements of multimedia data.

- **Multimedia Devices**

  At present, a small number of devices have been incorporated into the system including a camera service and an interactive videodisk player (as a source of still and moving image). A windowing system supporting video window capabilities is also provided at this level. To achieve this, the X-server component of X-Windows was ported on to a transputer environment driving the video subsystem and then modified to support video windows.

- **Storage Services**

  The approach taken at Lancaster is to provide a number of specialised storage services for each media type. This approach allows each storage service to be tailored to the characteristics of a particular media type. All storage devices present a common abstraction, referred to as chains. Chains are a generalisation of the voice ropes adopted in the Etherphone project for storing audio information [12]. A chain consists of a number of individual links, e.g. single picture frames, connected together into a sequence.

Further details of the base services can be found in [5] and [6].

3.2. Distributed object management

3.2.1 Impact of multimedia on management

Much of the distributed systems work of the 1980s tended to ignore the role of object management within their system. It was often claimed that the time taken to make a decision regarding migration for example, was substantially more than the time saved by migrating an object to the correct node. Hence, systems tended to develop simpler and simpler decision making algorithms until many stopped making decisions at all, and simply took the same course of action whatever the situation (e.g. always use a remote procedure call (RPC) package to communicate with an object [10], [4]). These decisions however were based upon the assumption that most objects had similar characteristics, an assumption which is inherently incorrect in multimedia systems. Indeed, one of the distinguishing features of multimedia systems is that they support objects with a wide range of characteristics (e.g. from video objects which occupy several hundreds of megabytes of storage, to text objects which only occupy a few bytes).

![Figure 3: "The Window of Scarcity"](image-url)
The question of performance is more fundamental when applied to multimedia systems. Rather than reducing management overheads in an attempt to improve performance, it may well be necessary in multimedia systems to pay considerable managerial overheads in order to ensure a minimum level of service. This notion of a system having sufficient resources only if they are managed correctly was identified by the developers of the Dash system [1]. They categorised such systems by saying that they lay within the "window of scarcity" (see figure 3).

Management is therefore of critical importance in distributed multimedia systems. Furthermore it is important that management is both flexible and responsive: flexible in that it must deal with a variety of object characteristics and responsive in that it must react to changing circumstances. However, it is important in introducing management to avoid an excessive overhead.

3.2.2 Management approach

Research at Lancaster has experimented with management strategies for multimedia objects [7]. There are two important aspects to this work. Firstly, the cost of management is proportional to the required level of service. Secondly, management is designed to be flexible in that the level and style of management can be tailored by user policies and required quality of service.

As an illustration, consider our approach to persistence and migration. This work extends the ANSA invocation mechanism making these features transparent to the user.

In the implementation, objects can be in one of two states, i.e. active or passive. In our system, all objects appear active, i.e. ready to receive invocations. However, objects in this state consume system resources (e.g. memory). Hence, it is useful to be able to make objects passive by temporarily surrendering these resources. To control activation and passivation, all objects provide the following management interface:

- **Passivate** - surrender system resources and save current state
- **Activate** - restore a passivated object
- **Checkpoint** - save a significant milestone in an object’s history as a checkpoint object
- **AssumeCheckpointState** - return an object to a previously created checkpoint
- **Move** - migrate an object to another site

The approach to management relies on an ordered list of address hints held in the unique identifier for an object. This normally has the actual address of the object as the first entry of the list, followed by the addresses of two managers, a resource manager and a location manager. In searching for an object, it is assumed that the object is active and has not moved. If this is the case, the invocation will pick the first address from the list and proceed accordingly. This will be exactly equivalent to a normal invocation, with no management overhead. However, consider the case if the object has gone passive. Passive objects register their state with the resource manager on the local site and then surrender their resources. On invoking a passive object, an exception will be raised because the first address is invalid. On this exception, the resource manager will be contacted. This resource manager will then be responsible for activating the object and returning it to the state before passivation. At this point, the address hints are updated by the resource manager. This updated address hint table is then used in future invocations.

A similar situation exists if an object is migrated. Migration is implemented by an object de-registering itself with its resource manager, registering itself with the resource manager on the destination site and finally going passive. This change of location is then notified to the location manager. This time, the first address will fail and the invocation mechanism will contact the resource manager. However, the resource manager will not have the necessary information on that object. Hence, it is necessary in this case to contact the location manager to find the new location of the object. Again, the address hint list must be updated at this stage. Once this is established, the resource manager on that site can be contacted to activate the object.

With this approach, management overhead is incurred only when necessary. We refer to this feature as management by exception. More subtly, this overhead is proportional to the amount of passivation and migration activity. This can be established on a per object basis by policy guidelines from application programs.

The persistence and migration mechanisms described above have now been implemented within our experimental multimedia ANSA system. Ongoing research is now addressing the question of flexible management, i.e. the potential to tailor and control the style of management. Important features of this work are the ability to specify the quality of service of a particular object and the ability to specify policies for object management. With these two features, it is hoped to provide a platform suitable for a wide range of application requirements and media types.

4. Future work

4.1. Object Configuration and Re-Location

A considerable amount of work has been carried out in the field of process migration and configuration. Considerably less work has been done in the field of
object migration and configuration, despite the fact that the encapsulated nature of objects makes them an ideal unit of migration. The work that has been carried out in this field (e.g. [9]) has tended to concentrate on the problem of object granularity in distributed systems. That is, making a uniform model for objects ranging from simple integers to complex data structures which works with reasonable efficiency.

The introduction of multimedia and the increasing requirement to provide support for collaborative applications places new emphasis on this field. Many of the traditional algorithms for determining object migration may not be suitable for shared objects. For example, a policy such as 'move on call' would be prohibitively expensive if two or more people were jointly editing a single document. Instead, object configuration managers will need to be aware of the broad pattern of object access and move objects accordingly. However, such managers may also need to be aware of the object in question's characteristics. As discussed in section 3, multimedia systems are characterised by the wide variety of objects they have to manipulate, and policies suitable for one object may not be so appropriate for another. Specialist requirements in terms of hardware may also constrain potential object movement.

In addition, new techniques may be needed for multimedia objects. Many operations on such objects are concerned with presenting the data contained within the object on a remote site (e.g. editing a multimedia document). Such presentation often involves (through necessity) sending a significant portion of the object to a remote site. Object configuration policies which are aware of this could lead to significant improvements in the performance of distributed multimedia systems, with continuous multimedia objects being subjected to a 'move-on-display' type of policy.

4.2. Continuous and Event Based Synchronisation

Research at Lancaster is currently investigating the possibility of developing a manager responsible for maintaining continuous synchronisation between streams of information, based on the philosophy of management by exception. Synchronisation has always been an important issue on distributed systems. However, the introduction of new media types such as video and audio such systems has created renewed interest in the area. Synchronisation can be divided into two distinct types: event based synchronisation (e.g. providing subtitles synchronised with a sequence of video), and continuous synchronisation (e.g. maintaining lip-synch between a sequence of video and its associated sound track). Whilst the problems of event-based synchronisation have been examined in the field of object-oriented programming (using techniques such as events, triggers or active relationships), those associated with establishing and maintaining continuous synchronisation are still very much a research issue. The approach we are taking is to develop a synchronisation manager which will be responsible for synchronising two or more streams on behalf of a client. The client will be able to express certain synchronisation policies (e.g. dispose of surplus packets etc.) and be informed by the manager of exceptional circumstances which it is unable to deal with.

4.3. Groups as a unit of management

Whilst the variety of characteristics associated with distributed multimedia systems demands that management on a per-object basis is possible, the more general case is that of management of groups of objects. This has been recognised in projects such as Domino which uses the notion of management domains for this purpose.

However, the concept of groups is important in distributed systems irrespective of it's role in management. Systems such as ISIS [3] have demonstrated the effectiveness of process groups as a programming abstraction, and object groups are a key part of our system. Hence, rather than adopt a specialised means of grouping for management purposes we intend to utilise the power of object groups. As a precursor to this work we have extended ANSAware to include support for the invocation of object groups. These groups are created using a group factory. Currently this factory may be parameterised to control certain aspects of the created groups behaviour. For example a group which may have at most ten members may be created. In the future we hope to extend this notion of parameterised group creation to include the concept of creating managed groups. Objects joining such groups (objects may be members of multiple groups) would become subject to the policies in force for that group, and policy changes would affect all members of the group.

5. Concluding remarks

This paper has discussed the problem of extending distributed object-oriented architectures to support multimedia. It has been argued that object management is an important element in such an environment and experiments with multimedia object management have been described. The key features of this work are a flexible approach to management through application level tailoring and control over the cost of management by incurring management overhead only where necessary.
References


