An Architecture for Hypermedia Systems Using MHEG Standard Objects Interchange

Luiz Fernando G. Soares

Departamento de Informática, PUC-Rio
R. Marquês de São Vicente 225
22453 - Rio de Janeiro, RJ - Brasil
E-mail: lfgs@inf.puc-rio.br

Marco Antonio Casanova

Centro Científico Rio, IBM Brasil
Caixa Postal 4624
20001 - Rio de Janeiro, RJ - Brasil
E-mail: casanova@vnet.ibm.com

Sérgio Colcher

Departamento de Informática, PUC-Rio
R. Marquês de São Vicente 225
22453 - Rio de Janeiro, RJ - Brasil
E-mail: colcher@inf.puc-rio.br

Abstract

Most hypermedia systems have been developed as a unique and self-contained application preventing information interchange and code reusability between applications. To address this problem, a generic layered architecture for hypermedia systems with four major interfaces is first introduced. The architecture adequately uses the MHEG proposal to provide the desired independence between hypermedia applications' conceptual models and storage strategies for hypermedia objects. Then, some design problems of the topmost layer, using as example the nested context hypermedia model, are discussed. Finally, motivated by the fact that a distributed environment may be necessary to meet the real time requirements of hypermedia applications, some issues concerning the design of a distributed system based on the proposed architecture are briefly addressed.

Keywords:

1. Introduction

In recent years, many application domains such as education, training systems, office and business systems, information and point of sales systems, etc. saw an explosion of multimedia services. This trend can be explained by the increasing availability of multimedia resources on many modern computers, sometimes called *Multimedia Platforms*. Also, compression standards, like JPEG or MPEG, allowed the development of dedicated integrated circuits which can be installed in mother boards, like any other basic system resource. Additionally, the increasing availability of optical disks and digital transmission media, such as broadband telecommunications networks (like B-ISDN) permit the exchange of very large amounts of data.

In this context, many multimedia applications will be designed to run on heterogeneous platforms, or to be interconnected to offer a more sophisticated multimedia service. These services will use large quantities of structured multimedia objects, which can be either locally stored on a workstation, or retrieved from remote sources through a communication network. This multimedia data may represent a significant investment, so it becomes vital to ensure that this information is not lost due the incompatibilities in data structures supported by the different applications.

Most hypermedia systems have been developed as a unique and self-contained application preventing the interoperability, information interchange, and code reusability between applications. Some exceptions must be mentioned in this context, such as the NEPTUNE system [DeSc86]and the Hypertext Abstract Machine (HAM) [CaGo88].

From the previous scenario, a number of underlying requirements for a hypermedia architecture can be identified:

1. the architecture must be *open* with respect to two distinct aspects:
   a) open with respect to interconnection, which means that interchangeable objects must follow an international standard, allowing the interoperability and data availability among systems;
   b) open with respect to its structure, which means that it must present well defined interfaces, corresponding to different levels of services, that enable:
      - properly engineered applications software to be ported with minimal changes across systems,
      - maximal code reusability among applications;

2. applications must handle multimedia information in such a way that real-time interactivity (including acquisition of multimedia data), as well as real-time interchange, can be ensured;

3. the architecture must be possible to be used in a distributed environment, which is necessary to provide both:

- real-time functionality, in spite of the high storage and bandwidth requirements of multimedia information,
- value-added communication services between applications or users, like multimedia messaging services, videophony, teleconferencing systems, etc.

In this paper, we introduce a generic layered architecture for hypermedia systems with four major interfaces. The architecture adequately uses the MHEG proposal to provide the desired independence between hypermedia applications' conceptual models and storage strategies for hypermedia objects. When describing the interfaces and layers, we will focus mostly on the aspects related to object interchange.

We are also working on other aspects, such as system and data management, protocols, storage and retrieval of multimedia objects, spatio-temporal composition of multimedia objects, etc., although we do not address these issues in this paper. They will be treated in more detail in future works.

2 - Overview of the MHEG Standard Proposal

This section provides an overview and summarizes the major objectives of the ISO/IEC JTC1/SC29/WG12 group called the *Multimedia and Hypermedia Information coding Expert Group* (MHEG) [MHEG92].

The first steps towards enabling the interchange of multimedia features were the standardization of still image pictures, audio and video data. However, due to the interactive nature of the applications that manipulate objects of these basic media, it is also necessary to provide a higher level interface between applications and such objects. Within this requirement we include:

- the association between the content and the presentation attributes;
- spatio-temporal synchronization of component objects that will be presented in conjunction;

The MHEG standard defines the coded representation of finite information entities, which are generic final form units of multimedia/hypermedia information that can be used and interchanged in real-time by applications. The multimedia objects defined by the MHEG standard are intended for use in a wide range of CCITT Recommendations, ISO as well as other standards, and user defined architectures and applications.

The specification of the MHEG standard follows the object-oriented approach. Objects classes are defined that correspond to units of multimedia/hypermedia information for interchange purposes. The design of the object classes relies on the analysis of their common behavior, but the final definition is based on the attribute structure of the objects, since this is what allows the interchange of information between systems. It must be clear that the adoption of an object-oriented description does not imply that the development of an MHEG conforming process also has to be object-oriented.

The standard provides a description of each class using an *abstract syntax notation*, whose goal is to describe data types in a machine and application independent way. At present, the description of MHEG objects is presented in only one abstract syntax, called *Abstract Syntax Notation One* -- ASN.1, but, in the future, the MHEG committee also plans to provide a description in other notations, such as SGML.

There are many possible concrete representations that can be used to code an object described by an abstract syntax. A data structure coded in a programming language could be used, for example. In this sense, the data objects defined within the application layer can be seen as an internal representation for this description that allows higher layers to use interchanged objects without any knowledge of the *transfer syntax*.

The transfer syntax is a concrete representation for the abstract syntax. Its structure is defined by an international standard, called the *Basic Encoding Rules*, that specify how an object description in ASN.1 is to be coded. Since this syntax is internationally accepted, it can be used to code objects prior to its exchange, allowing the decoding by other end systems and the mapping of object structures in internal representations (such as the data objects in our model - see the next section) after their reception.

3 - The Hypermedia Layered Architecture

The hypermedia architecture we describe comprises three layers and four interfaces, as shown in figure 1.

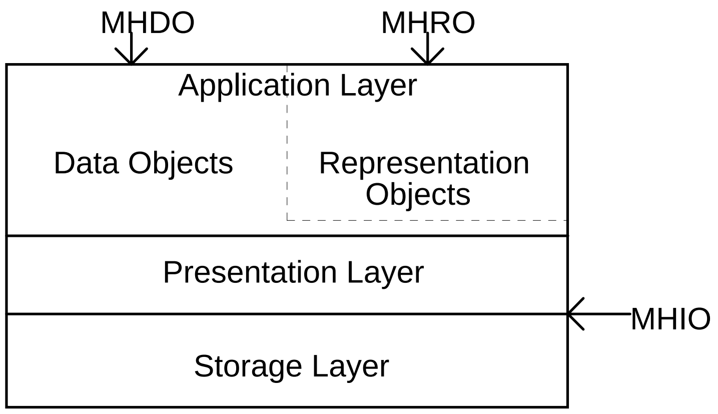

Figura 1 - Layered Hypermedia Architecture

The explanation of the layers and interfaces will be followed by a discussion about object organization.

The *application layer* implements the hypermedia conceptual model. It introduces the *data objects* and *representation objects* and offers two interfaces for hypermedia data manipulation, called the *multimedia hypermedia data objects* interface (MHDO Interface), and the *multimedia hypermedia representation objects* interface (MHRO Interface), which contain the methods associated with the data and representation objects, respectively, among others. Typical applications will directly use just the MHRO interface, while special applications may use both interfaces.

The *storage layer* implements persistent *storage objects* and offers an interface for hypermedia data interchange, called the *multimedia hypermedia interchangeable objects* interface (MHIO Interface). Special applications and other hypermedia systems may directly use this interface.

The main purpose of the *presentation layer* is to convert to and from the storage format of the data objects used by particular applications and platforms and the coded representation for the multimedia objects defined by the MHIO interface. We note that the presentation layer does not implement any of the methods associated with data objects.

The MHIO interface is the key to providing compatibility among applications and equipments, since it establishes two points at which the storage and the presentation layers must agree: (1) the coded representation for the multimedia objects to be interchanged, which corresponds to the ISO MHEG standard (see section 2); and (2) the messages, requests, confirmations etc., used by these layers to ask for the required object, content or action. These two points together are the subject of the CCITT T.170 series of recommendations (not yet provided) that will include the ISO MHEG standard as its T.171 recommendation.

From the point of view of object organization, the layered architecture dictates the following. A storage object has a unique identifier and a specific type, as well as other attributes. As the name implies, its existence is independent of the execution of a specific application. Storage objects are internal to the storage layer, but their coded representation is available through the MHIO interface.

A data object is created either as a totally new object or as a local copy of a storage object, adorned with new (non-persistent) attributes that are application-dependent. It contains methods to manipulate the new attributes, as well as methods to manipulate information originally pertaining to the storage object, if it is the case. The storage format of a data object corresponds to an internal concrete representation of an MHEG object.

A representation object class is a specialization of a data object class with new methods to exhibit the multimedia data contents in the format most appropriate to that particular use of the data. Representation objects are also directly accessible to the applications and offer, in a sense, different views of data objects.

Therefore, to access multimedia data, an application proceeds roughly as follows. Using query of navigational facilities of the hypermedia system, it first indirectly identifies a persistent object containing the desired data and requests the creation of a data or representation object corresponding to it.

The semantics of a particular hypermedia model and the aspects of multimedia data presentation are therefore entirely hidden within the classes of data and representation objects a specific hypermedia system defines. The appendix describes the Nested Context Model [Casa91], which we adopted for the design of data and representation objects in the implementation of our hypermedia system, in conformance with the defined architecture.

4 - Comments on Client-Server Implementations

The implementation of a hypermedia system based on the layered architecture outlined in section 3 and adopting a client-server architecture may follow several different alternatives. Figures 2 and 3 show a generic model for an implementation that leaves the application layer and the presentation layer on the client side and the storage layer and again the presentation layer on the server side.

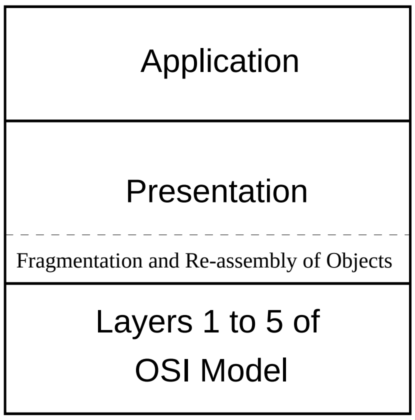
Application
Presentation
Fragmentation and Re-assembly of Objects
Layers 1 to 5 of OSI Model

Figura 2 - Client Archicteture

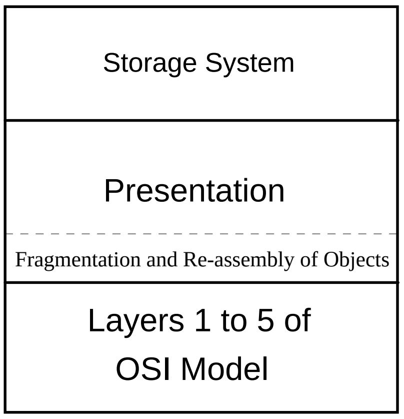
Storage System
Presentation
Fragmentation and Re-assembly of Objects
Layers 1 to 5 of OSI Model

Figura 3 - Server Archicteture

At a client, the application and presentation layers have exactly the functionality described in section 3, except that the presentation layer has a sublayer whose function is to fragment and re-assemble the multimedia objects specified by the MHIO interface. A similar observation applies to the server with respect to the storage and presentation layers. In figures 2 and 3, these sublayers are shown in dashed lines inside the presentation layer. The fragmentation and re-assembly of a multimedia object may in fact be quite complex, depending on the implementation of the server and the real time characteristics of its media, regarding data transmission delays, bandwidth and data continuity.

The implementation of the server may vary from purely centralized to completely distributed. A distributed implementation of the server is justified by the bandwidth limitations of storage devices. As a consequence, the storage of multimedia objects may have to be distributed across several storage systems so that, by using parallel retrieval, the real time requirements are met [GRAQ91].

In the distributed case,the fragmentation/re-assembly sublayer is essential, both at the clients and at the server, and is managed by the latter. The server has a component, called *hypermedia manager*, that is responsible for the storage of all attributes of the persistent objects, except their data contents, and for the management of the fragmentation/re-assembly process so as to meet the applications' real time requirements. The hypermedia manager may run on a single machine or be distributed over several machines. The other components of the server are responsible for storing the data contents of the multimedia objects, and they may be file servers specialized in storing specific types of media.

5 - Conclusions

We introduced in this paper a generic layered architecture for hypermedia systems that adequately uses the MHEG proposal to provide independence between hypermedia applications' conceptual models and storage strategies for hypermedia objects. The notions of data and representation objects played an important role in this respect. Finally, we briefly explored client-server implementations that follow the layered architecture. The layered architecture and the nested context model are being developed by the Department of Computer Science of the Pontifical Catholic University of Rio de Janeiro and the Rio Scientific Center of IBM Brasil. A single-user prototype system, that follows the nested context model, has been concluded, and a second prototype, that in addition conforms with the MHEG proposal and includes versioning, is nearly completed.

References

[CaGo88] Campbell, B.; Goodman, J.M. "HAM: A General Purpose Hypertext Abstract Machine". *Communications of the ACM,* Vol. 31, No. 7. July 1988.

[Casa91] Casanova, M.A.; Tucherman, L.; Lima, M.J.; Rangel Netto, J.L. Rodriguez, N.R.; Soares, L.F.G. "The Nested Context Model for Hyperdocuments". *Proceedings of Hypertext ' 91*. Texas. December 1991.

[CCLS92] Casanova, M.A.; Cavalcanti, M.; Lima, M.J.D.; Soares, L.F.G. "Versions in the Nested Context Hypermedia Model". *Technical Report, Depto. de Informática, PUC-Rio*. August de 1992.

[Cola92] Colaïtis, F. "MHEG, The Future International Standard for Multimedia and Hypermedia Objects". *ISO/IEC JTC1/SC29/WG12 N054*. October 1992.

[DeSc86] Delisle, N.; Schwartz, M. "Neptune: A Hypertext System for CAD Applications". *Proceedings of ACM SIGMOD '86*. Washington, D.C. May 1986.

[DeSc87] Delisle, N.; Schwartz, M. "Context - A Partitioning Concept for Hypertext". *Proceedings of Computer Supporteed Cooperative Work*. December 1986

[GoBo87] Goldstein, I.; Bobrow, D. "A Layered Approach to Software Design". *Interactive Programming Environments*. McGraw Hill, pag. 387-413. New York. 1987.

[GRAQ91] Ghandeharizadeh, S.; Ramos, L.; Asad, Z.; Qureshi, W. "Object Placement in Parallel Hypermedia Systems". *Proceedings of Hypertext '91*. Texas. 1991.

[Hala88] Halasz, F.G. "Reflexions on Notecards: Seven Issues for the Next Generation of Hypermedia Systems". *Communications of ACM,* Vol.31, No. 7. July 1988.

[ISO 86] ISO 8879. "Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML). 1986.

[ISO87a] ISO 8824. "Information Processing - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1). 1987.

[ISO87b] ISO 8825. "Information Processing - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1987.

[ISO 88] ISO 9069. "Information Processing - SGML Support Facilities - SGML Document Interchange Format (SDIF). 1988.

[KrCo92] Kretz, F.; Colaïtis, F. "Standardizing Hypermedia Information Objects". *IEEE Communications Magazine*. May 1992.

[Mark91] Markey, B.D. "Hypermedia Marketplace Prepares for HyTime and MHEG". *ISO/IEC JTC1/SC18/WG8 N1312*. June 1991.

[Meyr86] Meyrowitz, N. "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework". *Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications*. Portland, Oregon. September 1986.

[MHEG92] MHEG. " Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects - Part1: Base Notation. *Working Document S.7. ISO/IEC JTC1/SC29/WG12*. November 1992.

[PuGu91] Puttress, J.J.; Guimarães, N.M. "The Toolkit Approach to Hypermedia". 1991.

[Soar93] Soares, L.F.G.; Casanova, M.A.; Rodriguez, N.R. "Tratamento de Versões em um Modelo Conceitual Hipermídia com Nós de Composição". *Technical Report PUC-Rio - Departamento de Informática*. Rio de Janeiro. May 1993.

Appendix -- The Nested Context Model

The definition of hypermedia documents in the nested context model is based on two familiar concepts, node and links. *Nodes* are fragments of information and *links* interconnect nodes into networks of related nodes.

The model goes further and distinguishes two basic classes of nodes, called terminal and context nodes, the latter being the central concept of the model.

Intuitively, a *terminal node* contains data whose internal structure, if any, is application dependent and will not be part of the model. The class of terminal nodes may be specialized into other classes (*text, voice, image,* etc.) as required by the applications.

A *context node* groups together sets of terminal or context nodes, recursively. The concept of context node therefore permits organizing, hierarchically or not, sets of nodes and offers a mechanism to define different views of the same document, tuned to different applications or classes of users.

For example, to hierarchically organize a textbook one may first define a context node *B* that contains a set of nodes standing for the chapters of the book and a set of links indicating the chapter organization, which is not necessarily a linear sequence. Similarly, for the i-th chapter, one may define a context node C_i that contains a set of nodes standing for the sections of the chapter and a set of links indicating the section organization, and so on. Now, as frequently occurs, the authors may define different chapter organizations for distinct classes of users by defining other context nodes containing the same nodes as *B* (the same chapters), but a different set of links indicating the new chapter organization. All such context nodes can be seen as different views of the same book.

The concept of context node generalizes the homonym concept introduced in the Neptune system [DeSc86], which was in turn based on some ideas from PIE [GoBo87]. It also generalizes Intermedia's webs [Meyr86] and Notecard's fileboxes [Hala87].

A *link* basically connects two nodes. Since the content of a node may have an internal structure, that can be arbitrarily complex, links either anchor on whole nodes or indirectly indicate *regions* where they touch the nodes. For example, for text nodes, a region may correspond to a character string within the text, for 2D images, a region may be determined by a pair of coordinates that define a rectangle and, for context nodes, a region may be a displacement, as defined in the next paragraphs..

Each node has a *mask* that acts as the external interface of the node. That is, links will actually refer to regions inside the content of the node indirectly by indexing entries in the mask. Thus, changes to the content of a node do not necessarily propagate to the links that touch the node, if the entries in the mask can be updated accordingly.

To summarize, a node is an object that has a content and a mask, which is a list of regions. The exact definitions of content and regions depend on the class of the node. In what follows, we give more precise definitions for the key concepts of anchor, link and context node.

An *anchor* is a pair of the form (*M,i*), where *M* is a node and *i* is either the null value λ, or an integer less than or equal to the length of the mask of *M*, or an anchor (*N,j*) such that *N* is contained in *M*, in which case *M* must be a context node. We say that *M* is the *base* and *i* is the *displacement* of the anchor.

A *link* contains a pair (*s,d*) of anchors, where *s* is the *source* anchor and *d* is the *destination* anchor of the link. The *end nodes* of a link are the bases of *s* and *d*. A link may have conditions and actions associated with it, as described in the MHEG objects.

A *context node class N* is any class of objects that have one special attribute, CONTENTS, whose values are pairs of the form (*S,L*), where *S* is a set of terminal or context nodes and *L* is a set of links such that their end nodes are contained in *S*. We say that *N* contains each node in *S* and each link in *L*.

For example, let *A* be a context node that contains another context node *B* that in turn contains two nodes, *C* and *D*. Since *B* contains *C* and *D*, a link connecting these nodes may, in principle, be defined in *B* as ((*C,i*),(*D,j*)), where *i* and *j* are valid displacements for *C* and *D*. If one wants to create a link in *A* connecting *C* and *D*, he may define the link as ((*B,m*),(*B,n*)) and specify the displacements m and n to be entries in the mask of B that contain the anchors (C,i) and (D,j), respectively. Alternatively, he may define the link as $((B,(C,i)), (B,(D,j)))$.

A *hyperbase* is any set of nodes H such that, for any node $N \varepsilon$ H, if N is a context node, then all nodes contained in N also pertain to H.

The nested context model allows different context nodes to contain the same node and context nodes to be nested to any depth. Thus, we need a way of identifying through which sequence of nested context nodes a given node is being observed and which links actually touch the node from that nesting. This is captured by the notion of *perspective* of a node.

A *perspective* for a node N is a sequence P = $(N_1,....,N_m)$, with $m \geq 1$, such that $N_1 = N$, N_{i+1} is a context node and N_i is contained in N_{i+1}, for $i \varepsilon [1,m)$. Since N is implicitly given by P, we will refer to P simply as a perspective. Note that there can be several different perspectives for the same node N, if this node is contained in more than one context node. The *current perspective* of a node is that traversed by the last navigation to that node.

The nested context model also accommodates other features, such as *virtual structures*, that is, objects that result from the evaluation of expressions, synchronization mechanisms defined for links, as defined in the MHEG proposal, and versioning. A complete coverage of these topics is, however, outside the scope of this paper.