



DEPARTAMENTO DE
ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

**Concepção
de
Serviços Multimédia
em
Sistemas Distribuídos**

20
06
6

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Rua dos Bragas, 4099 Porto Codex – PORTUGAL

6. 8. 88

B. 1. 00

Faculdade de Engenharia da Universidade do Porto

Departamento de Engenharia Electrotécnica e de Computadores

Carlos Manuel Cardoso de Oliveira

Licenciado em Engenharia Electrotécnica e de Computadores
pela Faculdade de Engenharia da Universidade do Porto

004 (013)
0110 / con

Concepção de Serviços Multimédia em Sistemas Distribuídos

206 70-01
004 (043)
6 6 90

Dissertação submetida para
satisfação parcial dos requisitos do programa do
curso de Mestrado em Engenharia Electrotécnica e de Computadores
(Área de especialização de Telecomunicações e Computadores)

Porto
Fevereiro de 1995

043 M
046 c

Dissertação realizada sob a supervisão do

Doutor Eurico Manuel Elias de Moraes Carrapatoso

Professor Auxiliar do

Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Sumário

A evolução conjunta das Tecnologias da Informação e da Comunicação abre novas perspectivas para o desenvolvimento de aplicações telemáticas multimédia com conectividade global. Reconhecendo este facto, as Organizações Internacionais de Normalização ISO e ITU, reuniram os seus esforços na iniciativa *AudioVisual Interactive* (AVI), para definir o enquadramento dos futuros sistemas de informação multimédia distribuídos, e propor representações normalizadas e poderosas da informação multimédia e hipermédia.

Neste contexto, descrevem-se os modelos de representação da informação e as arquitecturas de comunicação, necessárias ao desenvolvimento de sistemas de informação multimédia distribuídos. As normas internacionais de Codificação da Informação Multimédia e Hipermédia (MHEG) e de Computação Distribuída Aberta (ODP), e outras normas relacionadas, são também descritas.

A funcionalidade de um engenho interpretador MHEG é analisada em pormenor e é proposta uma implementação, como componente de serviço de um sistema distribuído, para a plataforma PC/*Windows*. A dissertação conclui com a apresentação de duas arquitecturas de Sistemas Multimédia e Hipermédia Distribuídas, baseadas em MHEG e suportadas nas redes comerciais disponíveis actualmente.

Palavras Chave

Multimédia	MHEG	Sistemas Distribuídos	ODP
Hipermédia	AVI	Sistemas de Telecomunicações	CORBA

Abstract

The joint evolution of Information and Communication Technologies opens new perspectives for the development of multimedia telematic applications with global connectivity. Recognizing this fact, the International Standards Organizations ISO and ITU joined their efforts, in the AudioVisual Interactive (AVI) initiative, for the definition of a framework for the future distributed multimedia information systems, and to produce standards and powerful representations of multimedia and hypermedia information.

In this context, the models of information representation and the communication architectures; needed for the development of distributed multimedia information systems, are described. The International Standards for Multimedia and Hypermedia Information Coding (MHEG), and Open Distributed Processing (ODP), as other related standards, are also described.

The MHEG interpreter engine functionality is fully described, and an implementation is also proposed, as a service component of a distributed system, for the the PC/Windows platform. The thesis concludes with the presentation of two MHEG based system architectures, supported by commercial networks available today.

Keywords

Multimedia	MHEG	Distributed Systems	ODP
Hypermedia	AVI	Telecommunication Systems	CORBA

Resumé

L'évolution parallèle des Technologies de l'Information et de la Communication ouvre nouvelles perspectives pour le développement d'applications telematiques multimedia avec connectivité global. En reconnaissant ce fait, les Organisations Internationales de Normalisation ISO et ITU, ont réuni ses efforts dans l'initiative AudioVisual Interactive (AVI), pour définir l'enquadrément des futurs systèmes d'information multimedia distribués, et proposer representations normalisés et plus puissants de l'information multimedia et hipermedia.

Dans ce contexte, on décrit les modèles de representation de l'information et les architectures de communication, nécessaires au développement de systèmes d'information multimedia distribués. Les normes Internationales de Codification de l'Information Multimedia et Hipermedia (MHEG) et de Compuation Distributé Ouverte (ODP), et d'autres normes rapportés, sont aussi décrites.

La funcionalité d'un engin interpréteur MHEG est discuté en détail, et est proposé un développement, comme composant de service d'un système distributé, pour la plateforme PC/Windows. La dissertation termine avec l'apresentation de deux architectures de système, basés en MHEG et supportés sur les réseaux commerciaux disponibles actuellement.

Mots Clé

Multimedia	MHEG	Systèmes Distribués	ODP
Hypermedia	AVI	Systèmes de Télécommunications	CORBA

Agradecimentos

Quarta-feira de Cinzas do ano da graça de N.S.J.C. de 1995

Esta nau das descobertas chamada SARI, que tão bem serviu nestes tempos a esquadra do INESC, El'Rei e Portugal, chegou finalmente a bom Porto.

Os meus agradecimentos vão para o seu comandante aventureiro, o Professor Eurico Carrapatoso, e para a restante tripulação. Em particular, quero aqui exprimir a minha gratidão para com os meus companheiros de camarata nestes anos de aventura: o Mané (Francisco), o Jaime e o Mário. A amizade que nos une, e as 'grandes cousas' que fizemos juntos, são uma grata recordação que trago da viagem. A nau arribou ao Porto, mas cedo chegará a hora de partir na demanda de novas descobertas.

No âmbito mais pessoal quero recordar aqui a palavra amiga, e sempre incentivadora, do meu tio Carlos, que me levou a cursar engenharia. Agradeço também aos meus pais a confiança que sempre em mim depositaram.

Dedico o esforço destes anos ao membro mais novo da família, o meu filho Carlos. Nele deposito as minhas esperanças na eternidade, esse fio contínuo de vidas a que é preciso dar sentido.

Por último quero agradecer à minha esposa dilecta, a São, esse rio de amor persistente que sempre me esperou no cais. Agradeço o apoio incondicional que sempre deste ao meu desejo de navegar nas águas turvas da Investigação Portuguesa.

Ao leitor quero deixar umas palavras:

Uma dissertação numa área tecnológica é como um vinho leve e frutado, a beber jovem e sem demora, pois a garrafeira não lhe melhora o sabor.



(Carlos Manuel Cardoso de Oliveira)

Índice

Sumário	iii
Abstract	iv
Agradecimentos	v
Índice	vi
Lista de Figuras	ix
1. Introdução	1
1.1 Objectivos da Dissertação	2
1.2 Contexto do Trabalho Realizado	2
1.3 Estrutura da Dissertação	4
2. Modelos e Architecturas Multimédia	5
2.1 Modelos de Referência	5
2.1.1 Modelos de Composição e Sincronização	6
2.1.2 Modelo de Processamento da Informação	10
2.2 Architecturas de Serviços Multimédia	12
2.2.1 A Iniciativa AVI	12
2.2.2 Compatibilidade de Sistemas Multimédia	14
2.3 Síntese	17
3. Representação de Informação	18
3.1 Níveis de Informação	19
3.2 Codificação de Informação Multimédia e Hipermedia	20
3.2.1 Classes MHEG	20
3.2.2 Comportamentos	22
3.2.3 Intercâmbio da Informação	31

3.2.4 Apresentação e Interacção	33
3.2.5 Mecanismos de Identificação	36
3.3 Informação de <i>Script</i>	37
3.3.1 Funcionalidades de um <i>Script</i>	38
3.3.2 Níveis de Representação de um <i>Script</i>	39
3.3.3 Extensões para o suporte de <i>Scripts</i>	40
3.4 Autoria e Criação de Informação	42
3.4.1 HyperODA	42
3.4.2 HyTime	43
 4. Modelos de Computação Distribuída	 45
4.1. Evolução dos Sistemas Distribuídos	46
4.2. Modelos de Distribuição	48
4.3. Ambientes de Computação Cliente-Servidor	50
4.4. Modelos Orientados para Objectos	52
4.4.1 O Modelo de Objectos OMG	54
4.4.2 A Arquitectura CORBA	56
4.5. Modelo de Referência para Computação Distribuída Aberta	61
4.5.1 Mecanismos de Transparência	61
4.5.2 Modelos de Abstracção	62
 5. Componentes de Serviços Multimédia	 71
5.1 Processamento e Apresentação de Objectos Multimédia	71
5.1.1 Hierarquia de Componentes	73
5.1.2 Componente de Serviços Básicos	74
5.2 Modelo de Componentes do <i>Windows</i>	79
5.2.1 A tecnologia OLE2	80
5.2.2 O Modelo de Objectos COM	82
 6. Implementação de um Engenho MHEG	 86
6.1 Funcionalidades de um Engenho MHEG	87
6.2 Gestor de Entidades MHEG	89
6.3 Gestor de Sincronização	92

6.4 Gestor de Comunicação	96
6.5 Gestor de Apresentação	97
6.6 Gestor de <i>Scripts</i>	98
6.7 Interface do Engenho MHEG	99
7. Conclusões	101
7.1 Objectivo: Conectividade Global	104
7.1.1 <i>World Wide Web</i>	104
7.1.2 O Modelo <i>Theseus</i>	105
7.2 Perspectivas Futuras	108
Bibliografia	110

Lista de Figuras

1.1 Lista das normas referidas nesta dissertação	3
2.1 Objectos componentes multimédia	9
2.2 Composição de controlo de objectos multimédia	9
2.3 Níveis funcionais de processamento da informação	11
2.4 Modelo geral de um serviço audiovisual interactivo	13
2.5 Cenário de aplicação de consulta "inteligente" em tempo real	14
2.6 Arquitectura de referência dos sistemas multimédia IMA	15
3.1 Níveis lógicos de informação e representações de criação e edição da informação	19
3.2 Níveis de implementação dos objectos MHEG	21
3.3 Hierarquia das classes MHEG	21
3.4 Atributos de um Objecto <i>Action</i>	24
3.5 Atributos de uma Acção Elementar	24
3.6 Algoritmo de resolução de <i>Macro</i>	25
3.7 Estrutura de um objecto <i>Link</i> e exemplo de combinação lógica de condições genéricas	27
3.8 Técnicas de sincronização temporal	28
3.9 Evento primitivo	28
3.10 Disjunção de eventos	28
3.11 Conjunção de eventos	29
3.12 Sequência de eventos	29
3.13 Evento negativo	30
3.14 Objecto composto e apresentação correspondente	35
3.15 Exemplo de identificação externa utilizando o formato IETF RFC 1630 (URL).	36
3.16 Esquema de blocos de um sistema de apresentação multimédia	41

3.17 Domínios funcionais do MHEG e do HyperODA	42
3.18 Decomposição do HyTime em módulos	43
4.1 Partilha de recursos	46
4.2 Ficheiros distribuídos	46
4.3 Sistemas distribuídos	47
4.4 Aplicação distribuída por invocação de RPC	48
4.5 Estrutura dos interfaces do <i>Object Request Broker</i>	56
4.6 Mecanismos de definição de interfaces e de implementação de objectos	58
4.7 Problema da interoperabilidade entre ORBs	60
4.8 Representação de uma conexão explícita do tipo <i>stream</i>	67
4.9 Tradução do modelo computacional para o modelo de engenharia	68
4.10 Exemplo de um canal entre um cliente e um servidor	69
5.1 Modelo de eventos do PREMO	72
5.2 Hierarquia de componentes do PREMO	73
5.3 Tipos definidos no PREMO	75
5.4 Exemplo de sincronização controlada por um <i>Controller</i>	77
5.5 Tipos de conexões de entrada	78
5.6 Tipos de conexões de saída	78
5.7 Exemplo de funcionamento de uma porta de entrada assíncrona com reescrita	79
5.8 Tecnologias fornecidas pelos componentes principais do OLE2	82
5.9 Representação de execução de uma instância de um interface	83
5.10 Definição do Interface <i>IUnknown</i>	84
5.11 Funcionamento da operação <i>QueryInterface()</i>	84
5.12 Gestão de referências a um objecto	85
6.1 Exemplo de um sistema MHEG	87
6.2 Modelo proposto para o engenho MHEG	89
6.3 Gestor de entidades	90
6.4 Declaração do gestor de entidades	91
6.5 Gestor de sincronização	93
6.6 Processamento de um conjunto de acções elementares	94

6.7 Ciclos de processamento de um grupo de acções	95
6.8 Algoritmo de processamento de um objecto <i>Action</i>	96
6.9 Gestor de comunicação	97
6.10 Gestor de apresentação	98
6.11 Gestor de <i>scripts</i>	99
6.12 Exemplo de declaração da classe <i>CMHEGComposite</i>	100
7.1 Arquitectura do sistema <i>Integrated Multimedia Services</i> (IMS1).	102
7.2 Exemplo de percursos temáticos definidos sobre entidades	106
7.3 Arquitectura de um serviço de consulta em tempo real	108
7.4 Arquitectura de um serviço global de informação	109

Capítulo 1

Introdução

“Num futuro não muito distante, as redes de telecomunicações [...] constituirão o sistema nervoso da economia, e de modo geral da sociedade de amanhã.”

White Paper-“Growth, Competitiveness, Employment: The Challenges and Ways forward into the 21st Century”

Este fim de década que atravessamos é geralmente referido como a era da **Informação**. É verdade que muitos outros temas, preocupantes ou aliciantes, estão no centro da atenção mundial: a defesa do ambiente, a conquista do espaço e dos oceanos, as biotecnologias e os mundos virtuais, eis algumas das conquistas a efectuar no próximo milénio. No entanto, o tema da informação é dominante, porque tem implicações em qualquer dos outros.

São ainda muitos os desafios que enfrentamos para atingir a verdadeira sociedade da informação, e muitas questões surgem sobretudo na esfera económica ou social, centradas na importância relativa da informação ou em como assegurar o seu uso correcto. A nível tecnológico, as dificuldades residem no acesso à informação de qualidade, na quantidade e no instante desejados, numa perspectiva de **conectividade global**.

A resposta a este desafio já começou com as autoestradas da informação, cujos termos são já amplamente conhecidos: as redes de alto débito, a integração de serviços e o ciberespaço. Mas a concepção de soluções abertas para o problema da conectividade global dá ainda os primeiros passos, entre a indefinição existente e as dificuldades tecnológicas ou comerciais dos **sistemas distribuídos** propostos.

A **orientação para objectos** e a **multimédia** são os qualificativos que, neste contexto, mais usualmente são referidos para caracterizar os sistemas do futuro.

1.1 Objectivos da Dissertação

O objectivo principal desta dissertação é balizar o caminho que conduzirá, nos próximos anos, aos sistemas de informação multimédia distribuídos e abertos e a representações da informação mais poderosas e universalmente aceites.

É forte convicção do autor de que a peça chave desta engrenagem se situa na norma de representação da informação MHEG e que a aceitação progressiva, que o *'lobby'* MHEG tem conseguido obter junto dos fornecedores de sistemas e de serviços de telecomunicações, a posiciona competitivamente no contexto tecnológico mundial.

A iniciativa *AudioVisual Interactive* (AVI), que reúne os esforços das organizações internacionais de normalização ISO e ITU, é um compromisso claro e inovador para a definição de uma estratégia de integração das tecnologias de informação e das telecomunicações que vigorarão na próxima década.

Assim, os objectivos concretos fixados consistem em, atempadamente, perspectivar as tecnologias que farão parte integrante desse futuro risonho, e em especificar os serviços baseados nessas tecnologias que tenham viabilidade hoje e constituam uma ponte para o amanhã.

1.2 Contexto do Trabalho Realizado

O trabalho apresentado nesta dissertação foi desenvolvido no âmbito da actividade do grupo de Serviços Avançados sobre Redes Integradas, do Instituto de Engenharia de Sistemas e Computadores (INESC). Este grupo tem como linhas de interesse fundamentais as aplicações telemáticas multimédia, a engenharia de serviços e as redes de alto débito.

As primeiras aplicações telemáticas multimédia desenvolvidas pelo grupo, no início dos anos 90, tiveram como requisito essencial o suporte das normas de telecomunicações e de representação de informação existentes na altura [Carrapatoso+ 93ab]. Recentemente o grupo participou no projecto EURESCOM *Integrated Multimedia Services* (IMS1), cujo objectivo

consistiu em especificar e demonstrar um serviço de consulta multimédia, utilizando redes públicas com débito mínimo de 1.5 Mbit/s.

Este projecto foi desenvolvido em estreita colaboração com o grupo de trabalho da ISO responsável pelo desenvolvimento da norma MHEG (SC29 WG12), o que permitiu um conhecimento profundo da complexidade e das tecnologias necessárias à implementação, da representação da informação multimédia e hipermédia.

O trabalho apresentado nesta dissertação tem como base os resultados obtidos neste projecto, relacionando-os com os conhecimentos existentes na componente de engenharia de serviços do grupo, e está inserido numa acção estratégica de investigação aplicada nestes dois domínios [Henriques 95], [Leal 95]e [Silva 95].

O contexto desta apresentação está também relacionado com a relativa indefinição que ainda subsiste, no enquadramento definido pelas várias normas em desenvolvimento nos organismos creditados, *International Standards Organization* (ISO) e *International Telecommunication Union* (ITU), e com a consideração de outras forças em jogo.

O quadro da figura 1.1 apresenta alguns dos esforços que decorrem em áreas relacionadas com a multimédia, e que podem conduzir à adopção ‘*de facto*’ de normas definidas por organizações com grande representatividade como: o *Object Management Group* (OMG), a *Interactive Multimedia Association* (IMA) ou a *Internet Engineering Task Force* (IETF).

Organizações	Informação MM&HM	Informação Script	Sistemas & Serviços MM	Sistemas Distribuídos	Protocolos e Interfaces	Bases de Dados
ISO SC18 SC24 SC29 SC21	MHEG	HyTime,SMSL MHEG-S	MHMF PREMO	 ODP	 DSMCC,ILM API	 SQLMM
ITU-T SGVIII SGI	T.171	T.172,173	T.170 F.740	X900	T.175,176	
ETSI			M&HIRS		SBV	
IMA	MDE	MSL	MSS			
OMG				CORBA	ODL	OQL
IETF				WWW	HTTP, MIME	
Proprietários	OMFI	ScriptX		OLE		ODBC

Figura 1.1: Lista das normas referidas nesta dissertação.

A norma de codificação multimédia e hipermédia ISO/MHEG [MHEG 94], que constitui o núcleo central desta dissertação, encontra-se em fase de votação *Draft International Standard* (DIS). A descrição da especificação apresentada no capítulo 3 está de acordo com

esta versão e contempla já algumas correcções posteriores, permitindo afirmar que está aproximadamente de acordo com a versão definitiva.

1.2 Estrutura da Dissertação

Esta dissertação aborda a concepção de serviços multimédia na perspectiva da sua integração em sistemas distribuídos. A estruturação que foi delineada reflecte esse objectivo, apresentando primeiro os aspectos mais directamente relacionados com o desenvolvimento de aplicações multimédia para, seguidamente, aprofundar a vertente dos sistemas distribuídos.

A parte final da dissertação apresenta um modelo, para a implementação de um engenho interpretador de informação multimédia, que constitui a peça fundamental de duas arquitecturas para aplicações telemáticas que são propostas.

Este **primeiro** capítulo apresenta os objectivos que guiaram o trabalho desenvolvido e o contexto em que foi realizado.

O **segundo** capítulo discute aprofundadamente os modelos mais utilizados para a concepção de aplicações multimédia e as arquitecturas que servem de suporte a essas aplicações.

No **terceiro** capítulo abordam-se os níveis de representação da informação que são usualmente considerados para a criação de informação multimédia. A norma MHEG é descrita com algum pormenor.

Os sistemas de computação distribuída são objecto do **quarto** capítulo. A relação entre dois modelos concretos, o CORBA e o ODP, constitui a base de discussão.

O **quinto** capítulo dedica-se a metodologias de implementação de objectos componentes multimédia. À apresentação dos serviços fundamentais definidos no PREMO segue-se a descrição da tecnologia OLE2, que permite implementar componentes no ambiente *Windows*.

O capítulo **sexto** propõe um modelo de implementação de um engenho interpretador MHEG, suportado na plataforma *PC/Windows*.

Finalmente, o **sétimo** capítulo tece considerações sobre a abordagem que foi seguida e sugere duas arquitecturas viáveis para sistemas multimédia utilizando as tecnologias disponíveis actualmente.

Capítulo 2

Modelos e Architecturas Multimédia

“O objectivo é tornar fácil e intuitivo o acesso, a manipulação e a criação da informação, independentemente do seu tipo, da sua localização ou da fonte de criação.”

Em *Information At Your Fingertips*, Microsoft 1990

A concepção de sistemas e serviços com capacidades multimédia deve partir de duas perspectivas complementares, integrando os aspectos relacionados com a produção de aplicações multimédia e o desenvolvimento de serviços telemáticos que usem essas aplicações. Um passo essencial para esta integração consiste na definição dos requisitos das várias entidades envolvidas e na normalização decorrente dos processos de criação, gestão e apresentação da informação.

Este capítulo começa por descrever os modelos utilizados para a criação de aplicações multimédia e apresenta em seguida as architecturas dos sistemas e serviços adequadas para suportar essas aplicações.

2.1 Modelos de Referência

O desenvolvimento de aplicações multimédia pode ser analisado segundo um paradigma de visualização em que duas entidades, o produtor e o consumidor de informação, são relacionadas por uma composição de objectos multimédia, cujo objectivo é a apresentação e a percepção efectiva pelo utilizador.

2.1.1 Modelos de Composição e Sincronização

As funcionalidades de autoria que permitem o desenvolvimento de uma aplicação multimédia consistem normalmente numa abstracção de alto nível das relações de **composição** existentes entre os vários elementos atómicos, que constituem a aplicação, e dos mecanismos de **sincronização**, que coordenam a sequência de funcionamento pretendida.

As perspectivas associadas aos modelos de composição e sincronização, estão estreitamente relacionadas e são mesmo complementares para a descrição completa do comportamento de uma aplicação.

Modelos de Composição

Existem vários modelos de composição que permitem descrever, de modo estruturado, as ligações existentes entre os elementos de uma aplicação. Os modelos mais úteis fornecem uma composição visual descrita por **grafos**.

Grafos são estruturas constituídas por **nós** que representam os elementos da composição, e **ligações** entre estes elementos. Cada elemento suporta um número determinado de ligações através de **pontos de conexão** específicos.

Um modelo de composição define uma semântica própria e as regras que devem ser seguidas para a construção de grafos. Estão definidos vários modelos de composição adequados à autoria de aplicações multimédia [Mey+ 92].

O modelo de **composição de fluxos de informação** considera o encapsulamento, em objectos, dos comportamentos da aplicação e a circulação de fluxos de dados entre os objectos. Neste modelo, um nó corresponde a um objecto que disponibiliza portos de entrada e saída nos quais podem ser estabelecidas conexões com outros objectos.

O modelo de **composição de actividades** descreve o comportamento da aplicação em termos das actividades, que decorrem em cada um dos seus componentes, e dos eventos, que estas podem gerar ou consumir. A geração de eventos por uma actividade, presente num nó, despoleta o estabelecimento de ligações entre pontos de notificação e pontos de aceitação de eventos. Os **eventos**, definidos por este modelo, permitem descrever comportamentos não determinísticos dos componentes activos de uma aplicação, como a reacção a uma interacção com o utilizador.

O modelo de **composição temporal** descreve relações temporais entre as sequências que constituem uma apresentação definida para a aplicação. A técnica de composição visual associada a este modelo corresponde a uma abstracção do eixo temporal, sobre o qual são definidas as relações entre os objectos. Este modelo é um caso particular do modelo de composição de actividades, em que só são permitidas actividades com duração bem definida e eventos determinísticos.

Modelos de Sincronização

Os modelos de sincronização utilizados na autoria de aplicações multimédia são normalmente usados para definir relações temporais entre componentes, podendo considerar ligações directas a um eixo temporal ou posições temporais relativas entre elementos.

Vários modelos são propostos, com vantagens e desvantagens decorrentes do tipo de aplicação desenvolvida [Blakowski+ 92].

Um modelo de **sincronização hierárquica** organiza os componentes da aplicação numa árvore, cujos nós expressam uma apresentação **sequencial** ou **concorrente** dos comportamentos definidos nos ramos. As acções activadas em sequência ou em paralelo podem ser atómicas ou compostas de outras acções e operações de sincronização.

Acções atómicas correspondem a apresentações, interacções ou **atrasos**, que permitem modelar sincronizações mais complexas. A restrição principal deste modelo consiste em só ser possível definir sincronizações no início ou no fim das acções.

O modelo de **sincronização por pontos de referência** permite definir posições relativas entre sequências de elementos básicos de apresentação. Este modelo considera os objectos divididos em sub-unidades periódicas, definindo pontos de sincronização entre sub-unidades de diferentes objectos. Este modelo permite efectuar sincronizações ao longo de toda a duração do objecto, sendo indicado para a sincronização de *streams* contínuas de dados como audio ou vídeo.

É ainda possível definir um modelo de **sincronização num eixo temporal**, directamente relacionado com a composição temporal de objectos determinísticos anteriormente referida.

Outra abordagem possível para modelar aspectos de sincronização temporal entre os objectos recorre a técnicas de especificação formal baseadas em redes de Petri temporizadas

[Little+ 90a]. Este modelo, baseado em **pontos de sincronização** entre processos, permite definir relações entre intervalos temporais arbitrariamente complexas. Contudo, as redes de Petri têm grande dificuldade em exprimir eventos não determinísticos.

Os modelos de sincronização podem também ser classificados segundo os níveis de abstracção correspondentes aos serviços que pretendem descrever [Little+ 90b]. É genericamente aceite a existência de sincronização a dois níveis distintos.

Uma **sincronização entre os objectos multimédia** que constituem a aplicação, definindo a alto nível relações expressas em pontos de sincronização e controlo entre objectos.

Uma **sincronização dos fluxos de informação** que são mantidos entre os objectos, com preocupações de baixo nível, como a qualidade de serviço ou a sincronização de *frames* de áudio e vídeo.

Existe uma correspondência directa entre o modelo de sincronização a dois níveis apresentado e os modelos de composição de actividades e de fluxos de informação. Conceptualmente, existe uma distinção clara entre os dados de controlo e os fluxos de dados multimédia que constituem a informação gerada ou consumida pelos componentes definidos numa aplicação. Nos sistemas distribuídos, o modelo de computação distribuída aberta definido pela ISO [ODP 94b] distingue também interfaces operacionais e interfaces de *stream* entre objectos.

Uma possível implementação com base nestes modelos, assenta numa infraestrutura para o desenvolvimento de aplicações multimédia orientada para objectos [Gibbs 91, Nierstrasz+ 92]. Uma aplicação é construída não por programação de objectos mas por composição de componentes pré-existentes no sistema, que fornecem determinadas funcionalidades multimédia.

Um objecto componente é uma entidade **activa**, no sentido em que pode gerar espontaneamente um evento relacionado com uma alteração no seu estado ou invocar uma acção sobre outro objecto. Os objectos componentes classificam-se pelo tipo de portos que possuem em objectos geradores (*Sources*), consumidores (*Sinks*) ou simultaneamente geradores e consumidores de informação (*Filters*).

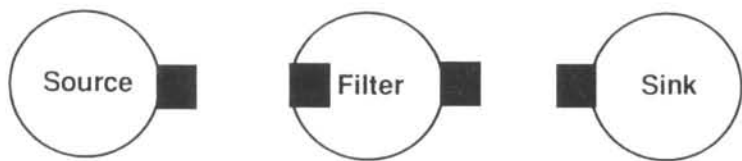


Figura 2.1: Objectos componentes multimédia.

A descrição da configuração topológica, necessária para a construção de um serviço específico, recorre a um **modelo de composição de objectos** que associa componentes e fluxos de informação num único objecto multimédia, que pode inclusivamente encapsular a distribuição dos objectos componentes no sistema. Ao nível de cada objecto multimédia pode existir sincronização entre os fluxos de informação que ligam os componentes.

O funcionamento da aplicação é então descrito por um **modelo de composição de controlo** que associa as actividades e os eventos gerados nos objectos multimédia [Pereira 94]. O suporte, em alguns objectos, de funcionalidades de geração de eventos em marcas temporais permite também obter algum controlo temporal da composição.

O resultado de uma composição de controlo é um processo **gestor de sincronização**, consistindo numa máquina de estados que invoca acções sobre os objectos, mediante alterações no seu estado provocadas pela recepção de eventos. Este conceito está também presente no modelo de processamento e apresentação da informação definido pela ISO [PREMO 94b].

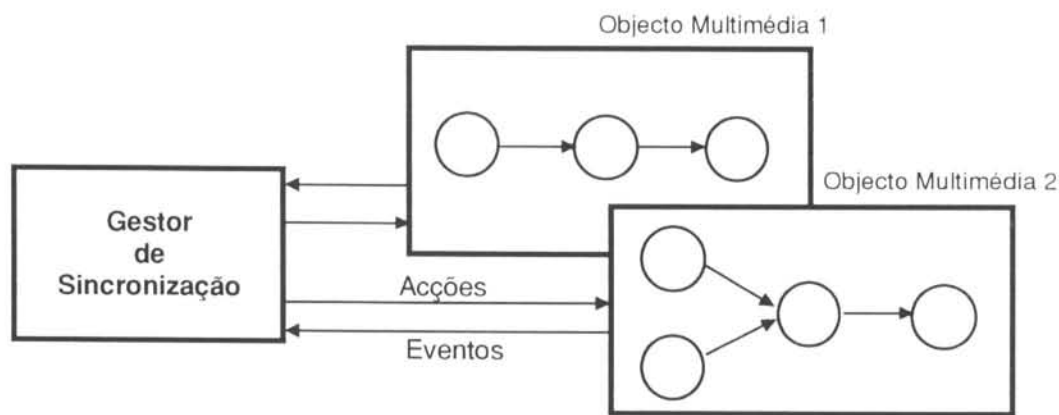


Figura 2.2: Composição de controlo de objectos multimédia.

Um gestor de sincronização pode ser encapsulado num objecto multimédia, possibilitando a criação de hierarquias mais complexas de sincronização. Outra alternativa de

sincronização consiste num controlo descentralizado encapsulado em cada objecto multimédia, o que possibilita descrever relações dinâmicas entre os objectos, introduzindo contudo maior complexidade no sistema [Bernardo 94].

2.1.2 Modelo de Processamento da Informação

A maior parte da informação produzida pela civilização, e de modo especial a informação multimédia, destina-se a ser perceptível por consumidores humanos. Por essa razão, um modelo de referência para o processamento de informação multimédia deve ter como base um paradigma de visualização da informação pelo utilizador final.

A aproximação actual mais geral e prática a este problema corresponde provavelmente à norma ISO para estruturação de documentos HyTime [HyTime 92]. Contudo, a produção de um documento HyTime considera sobretudo uma perspectiva de criação estruturada da informação, não especificando os processos que conduzem à apresentação da informação.

Uma outra aproximação, que pretende integrar os vários modelos especificados pela ISO no contexto alargado dos sistemas de informação, é apresentada no relatório técnico *Multimedia and Hypermedia Model and Framework* [MHMF 94]. Este documento considera os modelos de referência de computação gráfica (CGRM), distribuída (ODP) e de sistemas abertos (OSE e OSI) para identificar, nos sistemas multimédia, as relações existentes entre os aspectos de apresentação, armazenamento e representação da informação e as architecturas de comunicação. O modelo proposto baseia-se nos conceitos de aplicação e interface e em entidades exteriores que encapsulam a comunicação da informação e a interacção com o utilizador.

As duas perspectivas têm em comum a identificação de **níveis funcionais de processamento** da informação multimédia e a total independência do processamento em relação às infraestruturas de computação e ao ambiente de apresentação e interface com o utilizador usados. O fluxo da informação é descrito a três níveis funcionais, onde decorrem actividades em que essa informação é modelizada, formatada e finalmente apresentada ao utilizador.

O modelo de processamento e apresentação de objectos multimédia definido pela ISO (PREMO) considera as actividades de cada um dos níveis e o tipo de informação que lhes está associada [OMHEGA 94].

No nível conceptual de **modelização** da informação podemos então considerar a existência de aplicações, envolvidas no processo de criação de informação, manipulando a informação no seu estado bruto, estabelecendo relações lógicas entre os vários média.

A informação processada pelas aplicações é estruturada em documentos que descrevem, ao nível lógico, a apresentação desejada para a informação. O processo de **formatação** desta informação mapeia as relações lógicas definidas em representações físicas específicas, correspondentes à visualização da informação.

A informação obtida consiste numa representação codificada em formato final, sendo o processo de **apresentação** e interação com o utilizador controlado por um engenho interpretador que faz a gestão de instâncias concretas dessa informação.

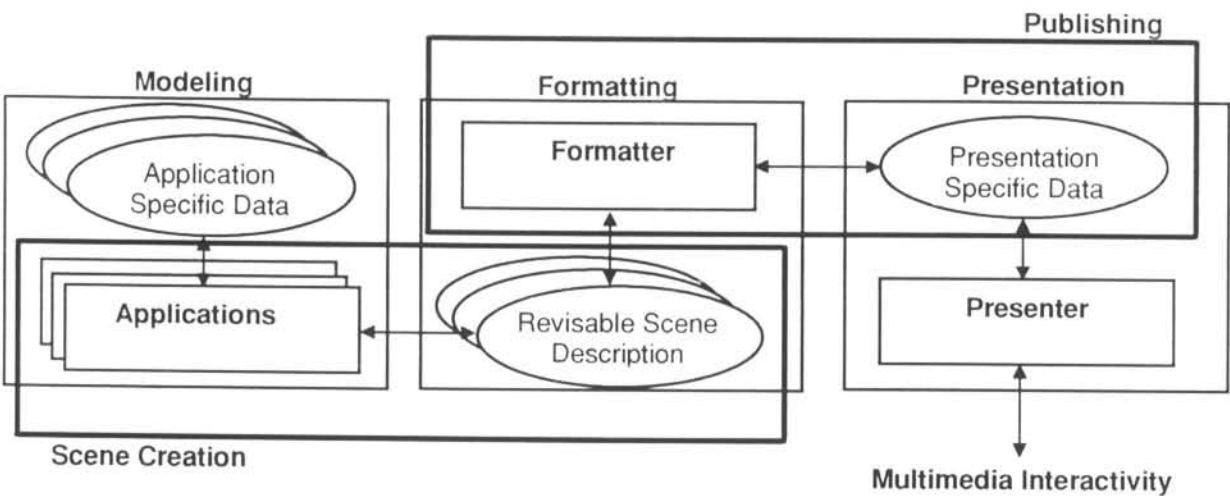


Figura 2.3: Níveis funcionais de processamento da informação.

O modelo considera ainda a manipulação da informação segundo as perspectivas, separadas no tempo ou no espaço, do produtor ou do consumidor e introduz dois processos correspondentes.

Um processo de **criação de informação** (*scene creation*) multimédia e hipermédia, que contempla a existência de formatos normalizados de representação (HyTime, HyperODA), destinados ao intercâmbio de informação estruturada entre aplicações de autoria.

Um processo de **publicação da informação** (*publishing*), que consiste na representação da informação multimédia e hipermédia num formato de codificação normalizado (MHEG), adequado para distribuição e apresentação ao consumidor.

2.2 Arquitecturas de Serviços Multimédia

A evolução conjunta das tecnologias da informação e das telecomunicações, que nos últimos anos se tem vindo a concretizar, abre um leque de novas oportunidades no domínio dos serviços de informação multimédia. A conjugação de técnicas de compressão de informação multimédia, suportes de armazenamento de grande capacidade e redes de comunicação integradas e de alto débito possibilita o desenvolvimento de sistemas multimédia interactivos.

Os esforços actuais dos vários organismos internacionais e dos agentes envolvidos na área multimédia concernem a problemática da compatibilidade a vários níveis entre sistemas multimédia e a definição de serviços independentes das arquitecturas utilizadas.

2.2.1 A Iniciativa AVI

Num esforço conjunto de normalização, os organismos internacionais ISO e ITU-T deram origem a uma iniciativa de grande escala no domínio dos serviços audiovisuais interactivos [AVI 93c]. O principal objectivo consiste em definir um modelo funcional que permita descrever serviços de consulta de informação (*Multimedia and Hypermedia Information Retrieval Services*), de forma genérica, e identificar os requisitos de normalização necessários para os implementar.

Os requisitos funcionais são definidos com base num modelo genérico que considera os principais papéis envolvidos na criação, comunicação, modificação, gestão e execução de aplicações [AVI 93a]. Nessa perspectiva, uma **aplicação** é considerada como um uso particular do serviço, com funcionalidades específicas dependentes dos requisitos envolvidos.

O modelo define o conceito genérico de **repositório de informação** associado ao conceito de **serviço**, responsável pelo acesso ao repositório para consulta ou actualização da informação. Os utilizadores do serviço são identificados como pertencentes a várias categorias associadas a requisitos funcionais específicos.

O **autor** de uma aplicação é responsável pela produção de informação multimédia e hipermédia, destinada ao consumo por vários utilizadores. Os requisitos funcionais do autor são principalmente facilidades de edição das aplicações implementadas.

O **gestor da informação** é responsável pela administração do repositório de informação, recorrendo sobretudo a funções de gestão para manutenção das aplicações existentes, controlo de acesso e tarifação.

O **consumidor** de informação utiliza a aplicação para fins pessoais, podendo ainda ser considerada a figura intermediária do **preparador** da informação. As funcionalidades requeridas são basicamente de execução.



Figura 2.4: Modelo geral de um serviço audiovisual interativo.

A implementação de serviços audiovisuais interactivos está sujeita a **requisitos** em variados domínios que requerem a normalização de procedimentos e metodologias.

A heterogeneidade existente entre **terminais** de acesso aos serviços recomenda a normalização dos processos que constituem a aplicação. Uma aplicação combina *scripts*, estruturas multimédia e hipermédia e informação monomédia, numa unidade consistente de produção ou exploração da informação (*scriptware*). Esta normalização consiste na especificação de semânticas adequadas ao contexto dos serviços AVI para a utilização de objectos MHEG e será desenvolvida na recomendação T.171 (*Coded representation of M&HI objects*).

Os aspectos de **segurança** devem ser considerados na implementação do serviço. Devem ser suportadas funcionalidades para controlo de acesso, confidencialidade e integridade da informação. Outros requisitos que devem também ser considerados são os aspectos de **tarifação** da informação, a normalização dos **procedimentos** de acesso a um serviço e o suporte de **intercomunicação** entre serviços heterogéneos.

Por último, o serviço deve ser capaz de oferecer uma **qualidade de serviço** aceitável aos utilizadores, independentemente das características do equipamento terminal, das redes de comunicação e do tipo de informação. As condições necessárias ao intercâmbio da informação são consideradas na especificação de diferentes **protocolos** de comunicação para situações de consulta e de transferência em tempo real, descritos nas recomendações T.175 e T.176 (*Protocols for non and real-time interchange of scriptware*). A definição de **perfis** funcionais, ao nível do terminal (*F profiles*) e da comunicação e armazenamento da informação (*A e T profiles*), permite identificar várias categorias de aplicações.

Uma metodologia para a classificação das diversas arquitecturas existentes consiste na definição de cenários possíveis de serviço [M&HIRS 92]. Um cenário considera o problema do acesso à informação com os requisitos de transferência em tempo real para processamento imediato ou para armazenamento. A taxonomia utilizada considera ainda os vários níveis de informação e a sua distribuição pelo terminal e pelo servidor.

Uma análise de todas as arquitecturas possíveis levou à selecção de seis perfis preferenciais: videotex tradicional, aplicação local com actualização de dados voláteis, aplicação carregada em diferido a partir da rede e aplicações de consulta em tempo real com diversos níveis de inteligência no terminal.

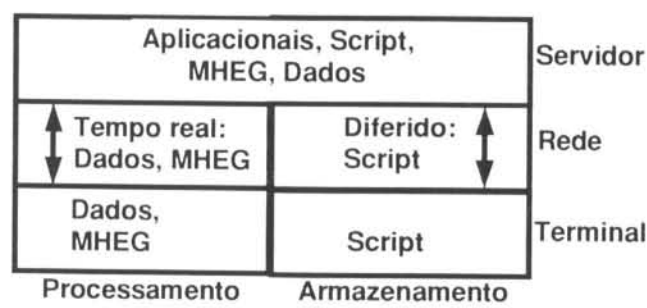


Figura 2.5: Cenário de aplicação de consulta “inteligente” em tempo real.

2.2.2 Compatibilidade de Sistemas Multimédia

A compatibilidade entre sistemas multimédia assenta na existência de compatibilidade em três níveis complementares: a informação, os serviços de suporte e as aplicações. Com base neste pressuposto, a organização internacional de produtores e utilizadores de sistemas de informação multimédia, *Interactive Multimedia Association* (IMA), definiu uma arquitectura de referência para a selecção das tecnologias adequadas (*Request for Technology*), que

epmitem assegurar a compatibilidade a vários níveis: transferência de dados, linguagem de *script* e sistemas multimédia.

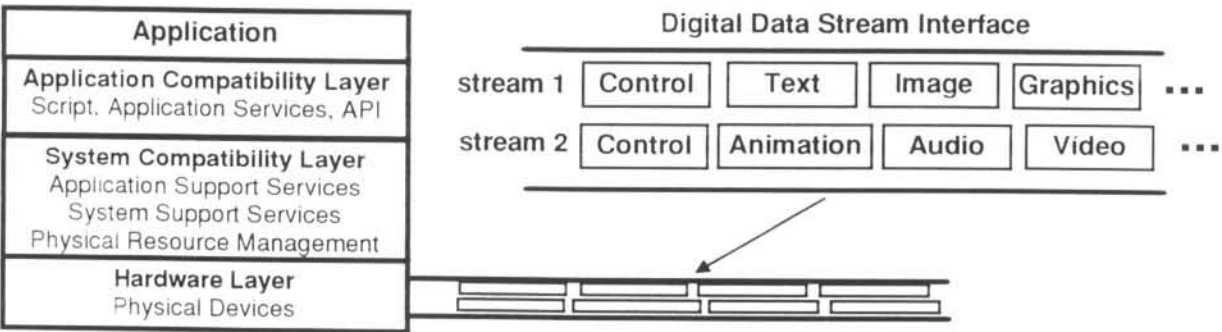


Figura 2.6: Arquitectura de referência dos sistemas multimédia IMA.

Multimedia Data Exchange

Na arquitectura de referência, a transferência de informação entre sistemas é efectuada sobre um interface de *stream* de dados. A informação é estruturada num *contentor* (*Portable Data Container Format*), que pode incluir componentes monomédia ou informação de controlo, por exemplo para sincronização de *streams*. É também fornecido um descritor (*Exchange Set Descriptor*), descrevendo os requisitos mínimos do terminal, *copyrights*, qualidade de serviço, procedimentos de inicialização, etc. Os formatos de dados considerados contemplam a transmissão, o armazenamento e a portabilidade da codificação da informação multimédia (*Portable Data Type*).

A proposta actualmente em discussão para a compatibilidade a nível da informação multimédia, denominada *Open Media Framework Interchange* (OMFI), é liderada pela Avid Technology, uma empresa da área do vídeo digital.

Este formato tem como objectivo fundamental a representação das dependências temporais e utiliza um modelo de composição de fluxos de dados e várias funcionalidades de sincronização temporal adequadas para a produção de vídeo. A interacção com o utilizador, as ligações dinâmicas entre objectos ou a utilização de *scripts*, funcionalidades suportadas pelo MHEG, não são contempladas neste formato [Buford+ 94].

Multimedia Scripting Language

A nível de compatibilidade entre aplicações, esta arquitectura de referência considera a existência de interfaces de programação (API) multimédia para a apresentação, como os

comandos MCI ou a tecnologia Quicktime, e interfaces de mais alto nível (*Application Services*), que oferecem serviços sobre a rede, como a vídeoconferência ou o trabalho cooperativo. Estes interfaces são comandados por aplicações que geram ou interpretam *scripts*. A linguagem de *script* é definida como uma notação de transferência entre aplicações de autoria e engenhos de apresentação.

A principal proposta em discussão para normalização de uma linguagem de *script* é denominada ScriptX e foi apresentada pela associação Kaleida Labs, liderada pela Apple Computer. O ScriptX consiste num ambiente de programação orientado para objectos que permite gerar *scripts* num formato codificado adequado para transmissão e interpretação por engenhos de apresentação dedicados [ScriptX 93]. O ambiente de programação oferece facilidades complexas de composição e sincronização de objectos construídos a partir de uma biblioteca de classes com várias funcionalidades, como a manipulação de monomédia, gestão de interações e do tempo ou engenhos de procura e de simulação.

O *script* especificado nesta arquitectura confunde num único conceito a codificação das relações multimédia e hipermédia entre os objectos e as funcionalidades de controlo de fluxo, processamento e acesso a processos externos, tradicionalmente separadas pelas organizações internacionais de normalização ISO e ITU-T em codificação MHEG e linguagens de *script*.

Multimedia System Services

O terceiro nível de compatibilidade tem como objectivo a interoperabilidade entre serviços. A arquitectura de referência considera a existência de serviços de suporte às aplicações que permitem o controlo virtual de periféricos, comunicações e objectos, serviços infraestruturais do sistema com funcionalidades de gestão de recursos, do tempo e da sincronização e por último, dos serviços de gestão de recursos físicos.

Os requisitos de compatibilidade entre serviços colocam-se na criação, transmissão e execução de aplicações multimédia sobre plataformas de computação distribuída heterogéneas. Embora existam propostas de normalização submetidas, como a iniciativa *Common Open Software Environment* (COSE) da área dos sistemas operativos, a arquitectura que maior consenso mundial reúne resulta dos esforços de normalização do **Object Management Group** (OMG).

A proposta da ISO para o processamento e a apresentação de objectos multimédia (PREMO) pretende conjugar os esforços destas organizações [Herman+ 94] num modelo geral

e extensível, que englobe também as normas internacionais de codificação da informação multimédia e hipermédia (MHEG) e de computação distribuída aberta (ODP).

2.3 Síntese

As relações de concorrência ou complementaridade entre os vários esforços de normalização dos sistemas e serviços multimédia são extraordinariamente complexas e muitas vezes movidas por interesses comerciais e não por vantagens tecnológicas. A integração de todas estas funcionalidades em aplicações heterogéneas, em que esteja contudo assegurada a compatibilidade e conectividade globais, é um desafio que nem sempre parece transponível.

Claramente, as linhas traçadas pelas organizações de normalização internacionais ISO e ITU-T são as mais abrangentes e apresentam finalmente uma visão estruturada consistente. As normas chave deste esforço consideram a distribuição dos serviços (ODP), o processamento e a apresentação (PREMO) e a codificação da informação (MHEG). Contudo, alguma inércia, inevitável face a interesses comerciais já estabelecidos, pode obrigar a soluções de compromisso e é um facto inegável que a heterogeneidade se manterá a vários níveis.

Nos próximos três capítulos discutem-se aprofundadamente os esforços de normalização em curso que devem ser tomados em consideração na concepção dos sistemas e serviços de informação multimédia actuais.

Capítulo 3

Representação de Informação

“A ideia básica do sistema *Xanadu* é a de um repositório para tudo o que já alguém escreveu e, portanto, de um hipertexto verdadeiramente universal.”

Ted Nelson, *Xanadu System*, 1965

A informação multimédia consiste normalmente numa colecção, ricamente estruturada e de grande complexidade, de componentes multimédia básicos. Esta estrutura pode ser organizada de várias maneiras: os documentos multimédia, por exemplo, apresentam uma estrutura hierárquica de componentes, enquanto que o conceito de hipermédia envolve uma estruturação em rede, seguramente mais complexa.

Embora estas duas formas de organização da informação sejam certamente úteis em vários contextos, tendem a negligenciar as possíveis relações temporais existentes entre os vários objectos.

Uma metodologia de estruturação da informação multimédia, verdadeiramente abrangente e geral, necessita de tomar em consideração todo o tipo de relações possíveis. A representação da informação deve pois ser estruturada segundo vários níveis de complexidade e ser capaz de descrever composições espaciais e temporais, ligações condicionais e mesmo interactividade.

Outra condicionante da representação da informação reside no fim a que se destina: informação trocada entre entidades computacionais tem requisitos diferentes dos da informação destinada à percepção por um ser humano. Numa perspectiva clássica, que separa o produtor e o consumidor de informação, também existem diferenças significativas na representação mais adequada.

3.1 Níveis de Informação

Na secção 2.1.2 foi apresentado um modelo para o processamento da informação segundo uma lógica de ambientes de criação e edição de informação multimédia. O maior mérito da perspectiva apresentada reside em apresentar uma distinção clara entre informação estruturada e informação formatada para apresentação.

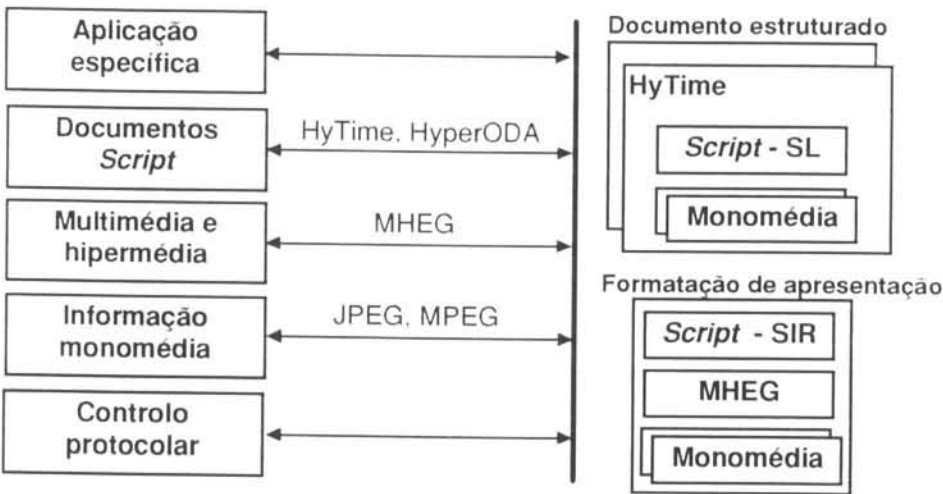


Figura 3.1: Níveis lógicos de informação e representações de criação e edição da informação.

Uma outra perspectiva possível constrói-se por enumeração dos níveis lógicos existentes no intercâmbio de informação entre duas aplicações. Neste contexto, e de acordo com a fig. 3.1, podemos enumerar os seguintes níveis de representação da informação:

- nível **aplicacional**: em que informação específica do contexto de cada aplicação é veiculada, podendo utilizar funcionalidades dos níveis inferiores;
- nível de **script**: informação estruturada em documentos, com descrição de elementos de sincronização e ligações entre objectos arbitrariamente complexas; as relações descritas podem fazer uso das facilidades disponíveis no nível inferior;
- nível **multimédia e hipermédia**: representação de sincronizações espaço-temporais ou condicionais simples e formatação da apresentação da informação monomédia;
- nível **monomédia**: utiliza as técnicas de codificação mais adequadas para cada tipo de informação (texto, imagem, audio, vídeo,...);
- nível **protocolar**: mensagens de controlo do fluxo da informação entre aplicações.

3.2 Codificação de Informação Multimédia e Hipermedia

A norma da ISO, conhecida pelo nome do grupo de trabalho que lhe deu origem - *Multimedia and Hypermedia Experts Group MHEG* [MHEG 94a], define uma representação estruturada das funcionalidades necessárias no nível da informação multimédia e hipermedia.

O formato de codificação dessa representação utiliza a sintaxe **ASN.1** [ASN.1 87] e é adequado para o intercâmbio de unidades de informação entre aplicações heterogêneas ou componentes de um serviço, utilizando suportes de armazenamento ou de transmissão.

Os objectos assim estruturados cumprem requisitos muito importantes no âmbito dos serviços multimédia:

- representação em **formato final**,
- suporte de sistemas com **recursos mínimos**,
- descrição de aspectos de **sincronização e interactividade**,
- facilidades para **apresentação** em tempo real,
- facilidades para **comunicação** em tempo real.

3.2.1 Classes MHEG

A norma MHEG usa o termo **classe** somente para descrever estruturas de objectos multimédia e hipermedia que podem ser transacionadas. O uso de metodologias orientadas a objectos facilita a especificação da estrutura dos objectos MHEG, mas a norma não define métodos correspondentes à implementação de interfaces com os objectos. Nesse sentido, os objectos MHEG não constituem objectos vivos mas sim repositórios estruturados de informação multimédia e hipermedia.

A norma distingue também três variantes diferentes de representação dos objectos MHEG instanciados a partir das classes, relacionadas com os vários fluxos de informação existentes numa lógica de apresentação:

- uma implementação dos objectos, contendo informação multimédia e hipermedia, destinada a **repositório** permanente de uma dada aplicação e codificada num formato adequado para transmissão;

- uma representação interna ao engenho interpretador de objectos MHEG, obtida por conversão dos objectos do repositório, que define os **modelos** e os comportamentos que condicionam uma determinada apresentação;
- instâncias dos objectos modelo destinadas à apresentação e interacção com o utilizador, denominadas **objectos de execução** (*run-time objects*).

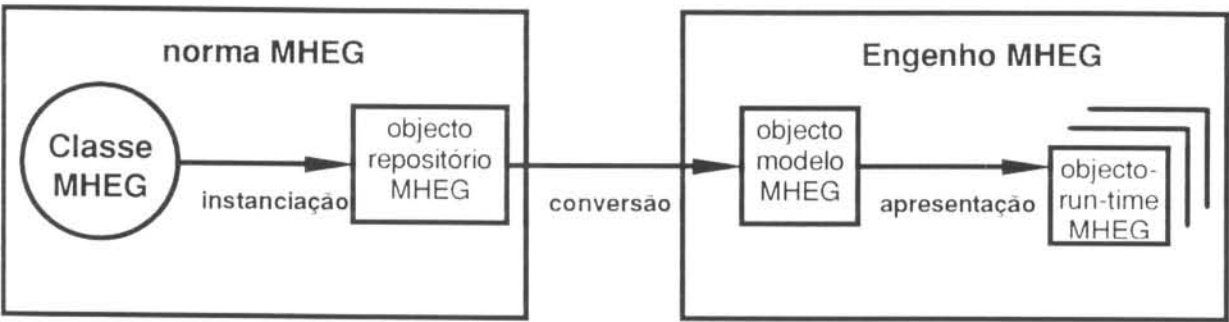


Figura 3.2: Níveis de implementação dos objectos MHEG.

As várias classes estão hierarquizadas da forma apresentada na figura 3.3. Classes abstractas foram adicionadas para descrever atributos comuns ou agrupar classes em tópicos.

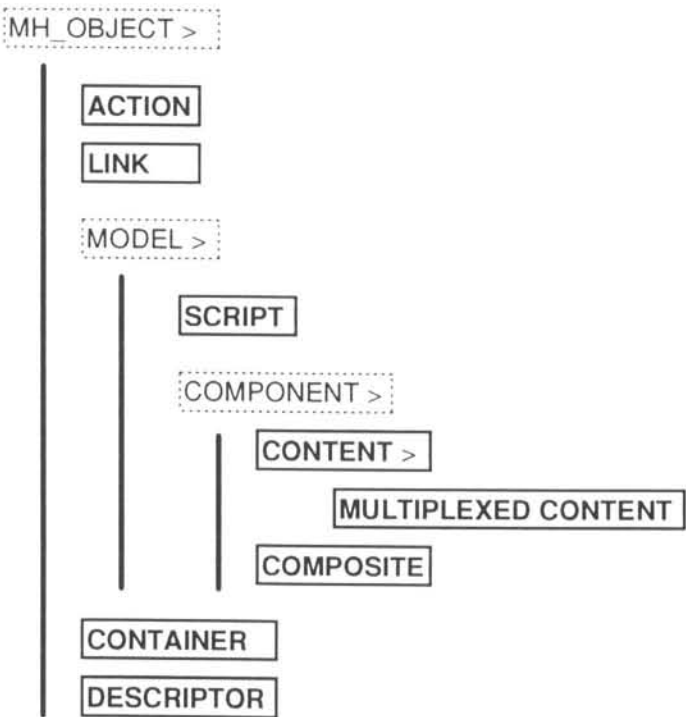


Figura 3.3: Hierarquia das classes MHEG.

3.2.2 Comportamentos

A norma MHEG define também os comportamentos (*behaviour*) possíveis para cada objecto MHEG, para as suas instâncias de execução e para os canais de apresentação, através de atributos e estados de cada entidade, permitindo em cada instante invocar novas acções ou consultar os comportamentos das entidades.

O processamento das acções sobre os objectos pode ter efeitos perceptíveis ao utilizador ou ter somente reflexos no comportamento do engenho MHEG. No âmbito da especificação definida pela norma apenas é considerado o segundo aspecto, deixando ao implementador do engenho MHEG a tradução da apresentação num interface efectivo com o utilizador.

O comportamento inicial de qualquer entidade pode ser alterado pela definição de instâncias de objectos MHEG das seguintes classes:

- da classe **Action**, que é utilizada para descrever conjuntos de acções a efectuar sobre entidades referenciadas,
- da classe **Link**, *que* é utilizada para especificar em que condições devem ser despoletadas determinadas acções,
- da classe **Script**, que é utilizada para veicular *scripts* necessários para descrever comportamentos mais complexos.

A utilização conjunta destas classes permite obter vários **níveis de sincronização** dos objectos multimédia.

As sincronizações **complexas**, por exemplo tomando em conta respostas do utilizador, recursos do sistema ou computação de variáveis, podem ser definidas por um *script*, que é encapsulado numa classe *Script* e entregue a um engenho interpretador.

Mecanismos de sincronização **relacional** e **ligações** (*hiperlink*) entre objectos são traduzidos em condições expressas em *Links* cujos efeitos são descritos por *Actions* sobre esses objectos.

As operações de sincronização podem ser de natureza **condicional**, no sentido em que o estado corrente do sistema pode condicionar o despoletar de acções destinadas a alterar o

comportamento futuro do sistema, ou consistir em sincronizações **espaciais** e **temporais**, descrevendo o posicionamento relativo no tempo ou no espaço de dois objectos de execução.

A sincronização **inter-média**, que diz respeito por exemplo a relações estreitas entre instantes bem definidos na duração de dois objectos, é normalmente considerada pelas próprias normas monomédia (o MPEG fornece mecanismos de sincronização estreita entre as *streams* de áudio e vídeo). Contudo, é também suportado um mecanismo de **marcas temporais** (*timestones*) que permite obter sincronizações deste nível através da definição de eventos temporais susceptíveis de serem processados por um *link*.

Os objectos *Action* são utilizados para descrever as acções a aplicar sobre uma dada entidade quando verificadas as condições definidas nos objectos *Link* que as referenciam. As acções definidas consistem num conjunto de **acções elementares** que podem invocar diversos comportamentos dos objectos MHEG, objectos de execução ou mesmo canais de apresentação:

Preparação: acções para controlar a disponibilidade de um objecto MHEG perante o engenho (*prepare, destroy, copy*),

Criação: acções de criação de objectos de execução a partir de classes *Model* (*new* e *delete*),

Apresentação: acções para controlar a progressão temporal dos objectos de execução (*run, stop, plug*),

Projecção: acções para controlar a projecção dos objectos de execução e a percepção pelo utilizador (*set speed, set size, set channel*),

Interacção: acções para controlar os resultados da interacção do utilizador com um objecto de execução (*set selectable, set modifiable*),

Activação: acções para o controlo por um engenho interpretador do processamento de cópias de execução dos *scripts* (*run, stop*),

Avaliação: acções para obter os valores dos estados ou atributos de um objecto, um objecto de execução ou canais de apresentação (*get*),

Retorno: acção de notificação de informação, cujo alvo é sempre implicitamente a aplicação que controla o engenho MHEG (*return*).

A classe *Action* consiste numa estrutura que especifica conjuntos sincronizados de acções elementares executadas sobre entidades referenciadas como alvos (*targets*). É constituída pelos seguintes atributos:

- um **indicador de sincronismo**, que especifica o tipo de processamento das acções sincronizadas (paralelo ou série),
- uma lista opcional de referências a entidades **alvo** do grupo de acções definidas,
- o número de **ciclos de processamento** das acções (poderá ser um ciclo infinito que só é terminado por remoção dos alvos dessas acções),
- um **conjunto de acções** sincronizadas.

Objecto Action

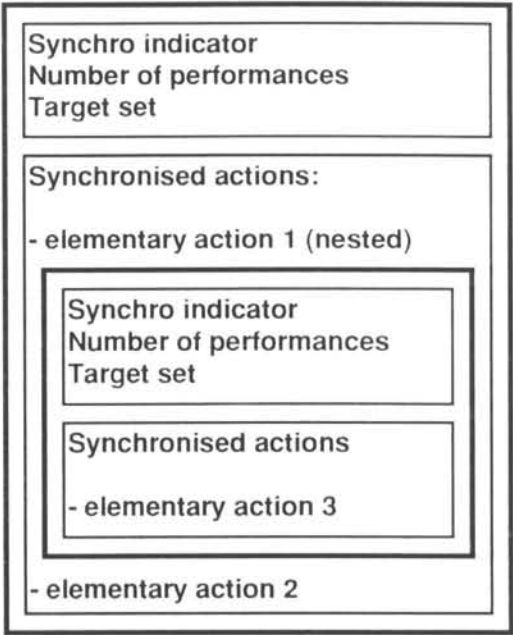


Figura 3.4: Atributos de um Objecto Action.

As acções definidas podem ser outros objectos *Action*, incluídos de forma **recursiva** (*nested*), ou acções elementares. A estrutura de cada **acção elementar** é constituída pelos seguintes atributos:

- uma lista opcional de referências a entidades **alvo** do grupo de acções definidas, que se sobrepõe à lista incluída no início do objecto,
- uma **duração da transição** também opcional,
- os **parâmetros** específicos de cada acção.

Acção Elementar



Figura 3.5: Atributos de uma Acção Elementar.

Objectos *Action* que descrevam acções frequentemente realizadas, em que apenas alguns parâmetros mudam, podem ser eficientemente codificados recorrendo a representações do tipo *macro*, constituídas por um **identificador de macro** e um **valor por defeito** opcional.

A execução de uma acção *macro*, por efeito do disparo de um *link*, desencadeia a resolução dos identificadores para os valores dos parâmetros correspondentes nesse contexto.

```
BEGIN PROCESSING of macro resolution attribute
  FOR I=1 TO number_of_macro_resolution_parameter
  DO
    PERFORM
      Macro_Def_ID=usage_value

    FOR I=1 TO number_of_macro_parameter_not_assigned
    DO
      PERFORM
        Macro_Def_ID=default_value
    END OF PROCESSING of macro resolution
```

Figura 3.6: Algoritmo de resolução de *Macro*.

Os *links*, definidos pela classe *Link*, descrevem um conjunto de condições determinantes para o disparo das acções correspondentes, definidas em objectos *Action*. As instâncias da classe *Link* servem portanto para especificar relações temporais, espaciais ou de interacção lógica e os efeitos dessas relações sobre as entidades referenciadas.

Os *links* definidos pelo MHEG são completamente dinâmicos, no sentido em que podem ser introduzidos ou removidos do engenho em qualquer instante e são independentes das entidades cujo comportamento é alterado pelo seu efeito. O mecanismo de disparo utilizado consiste na avaliação de **eventos** (*event driven*), gerados por modificações no comportamento das várias entidades sob controlo do engenho MHEG.

Relações complexas, que requerem processamento adicional para determinar os efeitos, não são suportadas com base unicamente nos mecanismos simples de sincronização disponíveis, razão por que estes *links* são geralmente referidos como *goto links*.

Uma instância de um objecto *Link* é constituída por uma combinação lógica de **condições** a serem verificadas e um conjunto de **efeitos** que ocorrem no caso de disparo do *link*. As condições descritas num *link* consistem em **condições de disparo** (*triggers*) e **restrições** (*constraints*), denominadas condições genéricas, que são articuladas numa combinação lógica, que perante um dado evento, poderá despoletar os efeitos do *link*. Uma **condição genérica** é estruturada segundo a seguinte operação:

$$\text{Result} = [\text{CondiçãoPrévia}(\text{SVP oper PV}) \text{ AND } \text{CondiçãoCorrente}(\text{SVC oper CV})] \tag{3.1}$$

Uma condição de disparo é verdadeira se ambas as condições, a condição prévia e a condição corrente, são simultaneamente verdadeiras. Uma condição de disparo exprime a ocorrência de um evento particular, relacionado com interações do utilizador, marcos temporais ou consequências de uma acção.

Uma restrição é verdadeira se a avaliação da sua condição corrente o é. Uma restrição exprime um estado contextual no momento em que uma condição de disparo é satisfeita e, como tal, não avalia condições anteriores.

As variáveis SVP e SVC exprimem os valores anterior e actual do atributo de uma entidade (*SourceValue*) que se deseja avaliar, sendo o valor actual obtido por uma *get action* no instante de avaliação do *link*.

Com base nos valores do atributo e nas variáveis de teste correspondentes (PV *Previous Value* ou CV *Current Value*), são efectuadas comparações. O valor de teste pode ser constante ou resultado também de uma *get action* sobre outro atributo, permitindo efectuar comparações contextuais no instante em que o evento ocorrer.

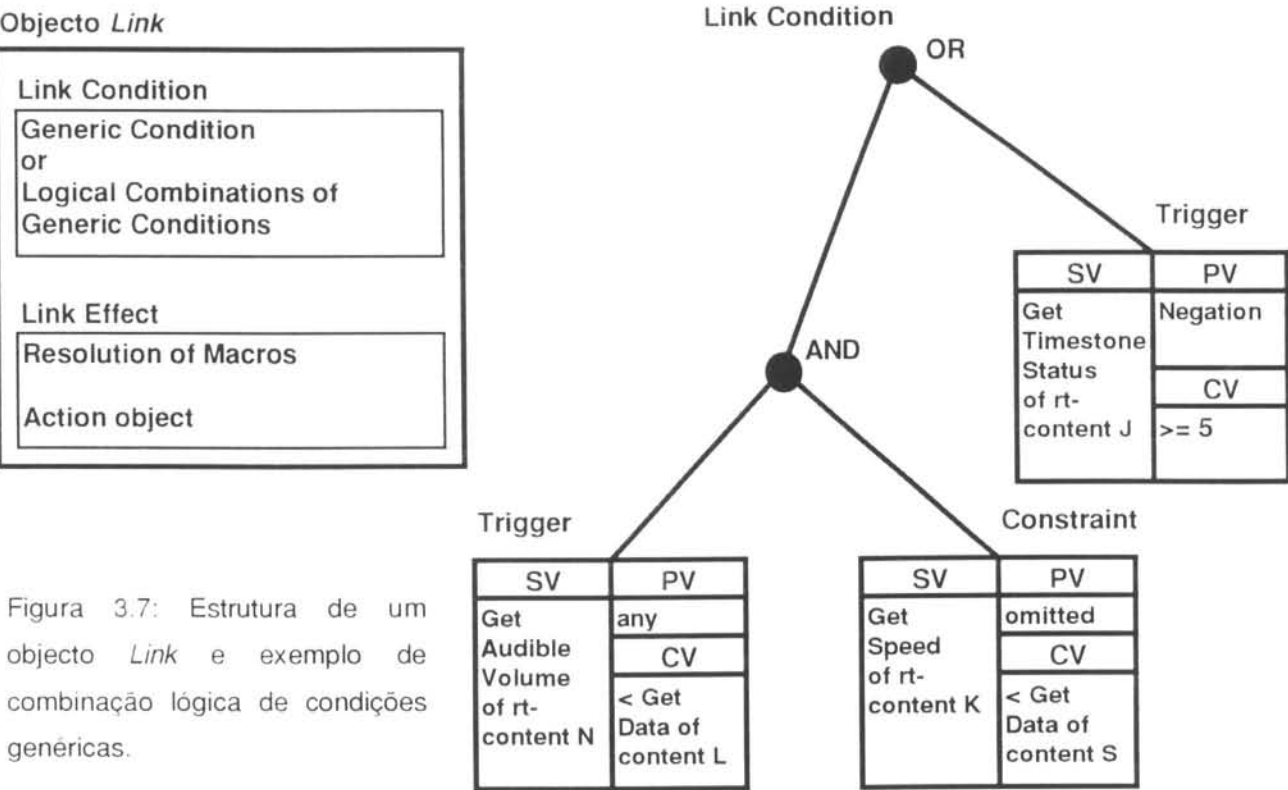
A operação de comparação utilizada em qualquer das condições poderá ser de igualdade, desigualdade, inferioridade ou superioridade (estrita ou não), negação da outra condição ou não representar qualquer comparação em particular (*any*). O resultado de uma operação de comparação depende do tipo de valores que estão a ser comparados.

Uma **combinação lógica** de condições genéricas é definida por um **operador lógico** aplicado a duas **listas de condições** genéricas. Por sua vez, cada lista pode ser também uma combinação lógica de condições, constituindo uma estrutura recursiva em forma de árvore.

Os operadores lógicos possíveis são dos tipos AND, OR, XOR ou as suas negações. As operações são efectuadas dos ramos da árvore para a raiz, sendo o valor booleano obtido na raiz aquele que define a condição do *link*.

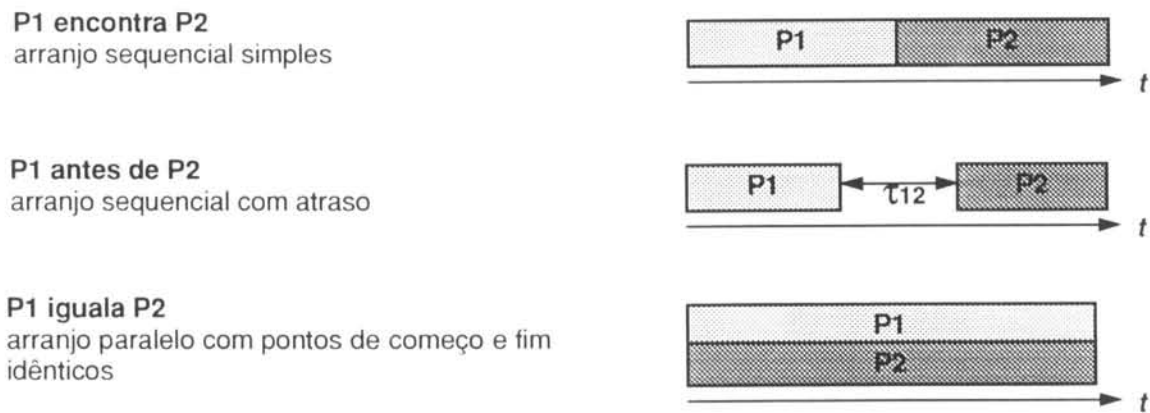
Na ocorrência de um evento, todos os *links* são avaliados em paralelo por forma a executar concorrentemente as acções correspondentes aos *links* disparados, tentando evitar assim alterações contextuais que possam falsear os resultados das operações.

O disparo de um *link* provoca a execução do **efeito** correspondente, que consiste no processamento das acções referenciadas num objecto *Action*. Se esse objecto incluir acções *macro*, existirá uma lista de parâmetros *macro* e valores correspondentes para resolução dessas acções no contexto particular existente.

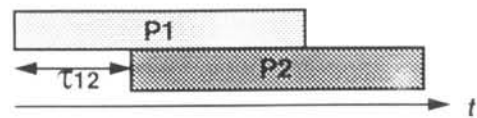


Utilizando as funcionalidades existentes nas classes *Action* e *Link*, é possível exprimir comportamentos complexos das entidades recorrendo a técnicas de sincronização espacial, temporal e condicional.

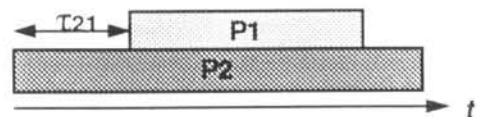
As estruturas básicas de **sincronização temporal** permitem descrever no tempo relações de sequência ou paralelismo entre as apresentações de dois objectos. A figura seguinte apresenta os arranjos mais representativos, a partir dos quais é possível construir relações temporais mais complexas entre os objectos.



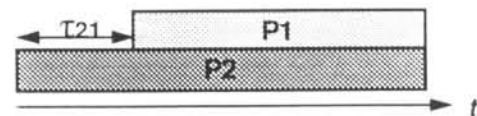
P1 intersecta P2
arranjo paralelo com atraso



P1 durante P2
arranjo paralelo com atraso e ponto de fim de P1 antes do ponto de fim de P2



P1 acaba P2
arranjo paralelo com pontos de fim idênticos



P1 começa P2
arranjo paralelo com pontos de começo idênticos

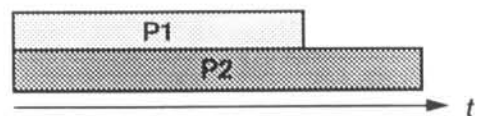


Figura 3.8: Técnicas de sincronização temporal.

As técnicas utilizadas para descrever **sincronizações condicionais** permitem atribuir efeitos a eventos não determinísticos, como as interações com o utilizador. Descrevem-se de seguida as técnicas básicas a partir das quais é possível construir sincronizações bastante complexas.

Eventos Primitivos

Um evento primitivo é definido num determinado instante do eixo temporal e pode ser gerado por transposição de um marco temporal, como resultado do processamento de uma acção ou por uma interacção com o utilizador.

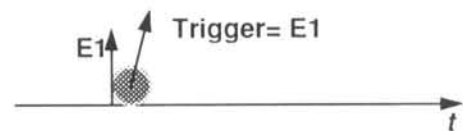


Figura 3.9: Evento primitivo.

Disjunção de eventos

A disjunção de eventos é disparada quando qualquer um dos eventos da disjunção acontece. Esta condição pode ser obtida por uma combinação lógica do tipo OR das condições de disparo associadas a cada evento.

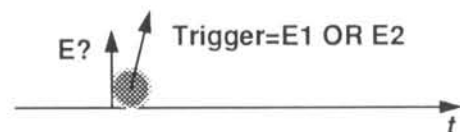


Figura 3.10: Disjunção de eventos.

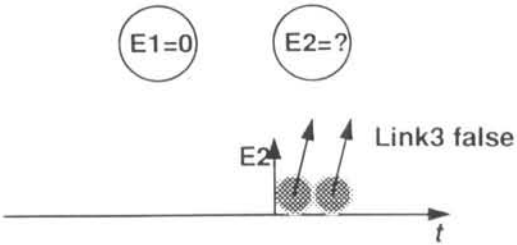
Conjunção de Eventos

A conjunção de eventos é verdadeira quando todos os eventos da conjunção aconteceram, seja por que ordem for. Uma conjunção não pode ser directamente traduzida num *link* porque requeriria que o engenho MHEG memorizasse eventos.

Existe no entanto a possibilidade de utilizar objectos de dados para armazenar variáveis, permitindo obter a mesma operação sobre os eventos por transformação da conjunção numa disjunção:

- a geração de cada um dos eventos dispara um *link* cuja acção é armazenar num objecto de dados uma variável booleana;
- é ainda definido um último *link*, cuja condição de disparo é uma disjunção de todos os eventos, sujeito à restrição de todas as variáveis serem verdadeiras, excepto uma. Esta restrição é necessária porque não se pode garantir que a última variável seja actualizada antes do disparo deste *link*.

Caso 1:



Caso 2:

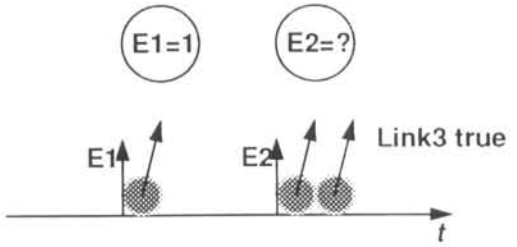


Figura 3.11: Conjunção de eventos.

Sequência de Eventos

Uma sequência de eventos é descrita por uma cadeia de *links* que vão sendo preparados à medida que vão acontecendo os eventos e cuja condição de disparo é o evento seguinte.

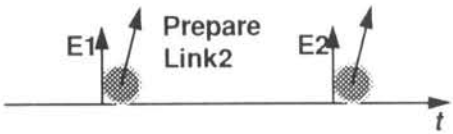


Figura 3.12: Sequência de eventos.

Eventos Negativos

Um evento negativo é aquele que não ocorreu num intervalo de tempo específico.

Este evento é descrito por um *link* com uma duração pré-determinada (por outro *link*), cujas acções associadas quando o evento é gerado são:

- modificar uma variável booleana armazenada num objecto *Content* e invocar sobre o próprio *link* uma acção *destroy* que o desactiva.

Um segundo link tem como condição de disparo o primeiro ficar desactivado e como restrição a variável ser falsa.

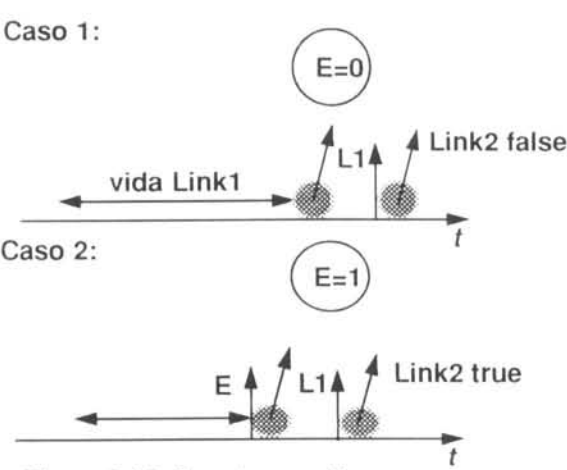


Figura 3.13: Evento negativo.

A inclusão de funcionalidades mais complexas de especificação de comportamentos ou de manipulação de outros atributos pode ser conseguida de duas formas:

- efectuando modificações proprietárias no engenho interpretador que permitam a inclusão de novos atributos dos objectos ou mesmo de novas acções elementares, como a manipulação de atributos de cor ou acções de pagamento; neste caso um engenho MHEG básico simplesmente ignorará essas novas acções;
- uma solução mais correcta será recorrer a um *script* que interacciona com o engenho invocando acções no seu interface, conforme definido nas extensões à norma **MHEG-S** em desenvolvimento [MHEG 94b]. O intercâmbio da informação de *script* ou da sua referência, juntamente com informação de descrição e classificação, é efectuado numa estrutura definida pela classe *Script*. A classe *Script* é um modelo a partir do qual podem ser instanciados e activados vários *scripts* de execução (*rt-scripts*) independentes.

Um exemplo típico é a simulação de um jogo de bilhar. O comportamento de uma bola é definido pela força e direcção do impacto inicial, colisão com obstáculos e, possivelmente, atrito. Este comportamento é por certo complexo e só possível de modelar por um *script*. Uma vez definido o algoritmo, este é aplicável a qualquer bola desde que introduzidos os parâmetros de entrada correspondentes e processados os eventos de colisão gerados.

A passagem de **parâmetros de entrada** para um *script* de execução é possível pela invocação da acção *set parameters*. É possível passar parâmetros por **valor** ou por **referência** a um objecto de dados. Neste último caso, os resultados do processamento do *script* podem

ser obtidos pelo engenho MHEG por invocação de uma acção *get data* sobre o objecto referenciado.

O interface do engenho MHEG fornece uma acção *return* que permite **notificar** a aplicação ou o *script* que controla o engenho MHEG. A acção *return* tem como parâmetro obrigatório um **indicador** numérico que identifica, perante a aplicação, o tipo de notificação. Os parâmetros opcionais são uma lista de **valores** genéricos e uma lista de **objectos** que devem ser passados para a aplicação. É possível também retornar objectos por **referência**, utilizando como valor uma referência genérica.

As particularidades dos *scripts* e a relação existente com o engenho MHEG são discutidas mais adiante, no ponto 3.3 deste capítulo.

3.2.3 Intercâmbio da Informação

O conceito de contentor, definido pela classe **Container**, fornece uma ferramenta de empacotamento da informação destinada a armazenar ou comunicar em bloco um grupo de objectos associados a uma dada aplicação.

Os elementos armazenados ou referenciados pelo contentor são identificados por um **índice** (*index*), que possibilita um rápido acesso a cada objecto quando a preparação deste é invocada.

A estrutura dum objecto *Container* consiste basicamente em dois subconjuntos distintos: um comportamento pré-definido e uma lista de elementos incluídos ou referenciados pelo contentor.

Comportamento pré-definido

Este comportamento é opcional e composto por dois objectos *Link* denominados *Availability start-up* e *Availability close-down*.

As condições de disparo destes *links* estão relacionadas com o estado de preparação do contentor. Os efeitos relacionados com o disparo dos links são acções de preparação ou destruição, invocadas sobre os elementos do contentor, ou outras acções especificadas pelo utilizador.

Lista de elementos do contentor

Esta é uma lista de elementos descritos por um **índice** único, dentro do contentor, e pela inclusão ou referência de uma instância de qualquer **objecto MHEG**, incluindo outro contentor de forma recursiva.

Tipicamente o contentor é utilizado como elemento agregador para a codificação estruturada de objectos do tipo **modelo** (subtipos da classe *Model*) e **comportamentos** (classes *Link* e *Action*) relacionados e pertencentes a uma dada aplicação.

Normalmente é também incluído um objecto de **descrição** (classe *Descriptor*), que fornece informação em vários níveis, relacionada com os objectos, os canais de apresentação, os estilos de interacção requeridos e informação genérica. A descrição assim obtida pode ser útil na negociação, instalação, operação e gestão da aplicação.

O suporte de funcionalidades, como a comunicação da informação e apresentação em tempo real, ou a execução em sistemas com recursos mínimos, é obtido a partir de informação constante da descrição. Por exemplo, a descrição da qualidade de serviço desejada, dos recursos necessários e também da dimensão da informação pode implicar a comunicação parcial ou progressiva de informação monomédia de forma a cumprir os requisitos de tempo real definidos.

O suporte de tempo real condicionou a especificação de objectos de **dados** com informação adicional para descrição de **dados multiplexados** (classe *Multiplexed Content*). Essa informação consiste numa lista de parâmetros descrevendo e identificando as várias *streams* constantes de uma multiplexagem. As *streams* podem estar recursivamente incluídas dentro de outras *streams*, sendo por isso cada identificador expresso de forma indexada relativamente ao objecto multiplexado.

A funcionalidade introduzida por esta classe destina-se a assegurar a interoperabilidade entre a norma MHEG e as várias versões da norma ISO de codificação audio e vídeo **Moving Pictures Expert Group** [MPEG 94]. Dentro de uma *stream* MPEG pode ser veiculada informação monomédia (ex. animações ou legendagem) e objectos MHEG, que permitem controlar a apresentação, e também as *streams* audio e vídeo existentes.

O controlo das *streams* MPEG é obtido pela invocação de acções sobre objectos da apresentação, que são mapeadas em comandos definidos na norma ISO **Digital Storage Media**

Control Commands [DSMCC 94]. Esta norma descreve funcionalidades de controlo para selecção de *streams*, comando de uma *stream* (*play*, *pause*, *jump*, ...) e armazenamento de *streams*. Estão também previstas extensões para edição de *streams* e estabelecimento de serviço, incluindo negociação, autenticação, etc.

3.2.4 Apresentação e Interacção

A classe **Composite** permite agrupar a informação e os comportamentos que lhe estão associados, em unidades lógicas de apresentação encapsuladas num objecto composto. Os objectos compostos podem também ser estruturados de forma recursiva, definindo relações de propagação entre as unidades lógicas de apresentação que constituem uma dada aplicação.

A descrição da **interacção** com o utilizador é considerada um comportamento associado às acções de **selecção** ou **modificação** de objectos da apresentação.

Um objecto composto é estruturado basicamente em dois subconjuntos que descrevem os objectos do tipo **modelo**, que serão instanciados numa composição, e os comportamentos, que os relacionam de forma consistente.

Comportamento da composição

O comportamento da composição é constituído por um comportamento opcional pré-definido e um comportamento específico.

O **comportamento pré-definido** é composto por quatro objectos *Link* denominados: *Availability start-up*, *Availability close-down*, *Rt-Availability start-up* e *Rt-Availability close-down*. As condições de disparo estão relacionadas com o estado de preparação do objecto composto ou o estado de activação de uma instância de execução criada a partir deste.

Os efeitos relacionados com o disparo dos links são acções de preparação ou activação sobre os elementos descritos no objecto composto. A destruição de um objecto de execução não tem nenhum efeito definido por defeito. O efeito de qualquer destes *links* pode ser personalizado, sendo possível, por exemplo, criar objectos autoexecutáveis.

O **comportamento específico** é definido por uma lista de objectos *Link* e *Action*, possivelmente transferidos para o engenho num contentor.

Percepção original da composição

A percepção original é uma funcionalidade opcional e é constituída por valores lógicos de duração temporal e dimensão espacial da apresentação, que são utilizados para inicializar o canal de apresentação.

Um **canal de apresentação** é um espaço lógico no qual os componentes de execução são posicionados e apresentados. Cada componente só pode ser apresentado num canal num determinado instante e o canal atribuído a um componente é definido por uma acção elementar *set channel assignment*, ou é herdado do composto de execução de que deriva o componente.

Existe um canal por defeito no qual são projectados todos os componentes que não definiram um canal. Um canal pode ser criado, destruído ou inibido em qualquer instante.

Um canal define um **espaço tridimensional**, um **eixo temporal** e uma gama de **volume audio**. A atribuição de valores físicos aos valores lógicos definidos para cada uma destas entidades resulta numa dada percepção pelo utilizador.

Não é requerido que um dado monomédia se projecte no espaço lógico mais usual. Por exemplo, um texto pode ser projectado sobre um canal de áudio, utilizando técnicas de síntese vocal da informação escrita.

Lista de elementos da composição

Cada elemento da composição é constituído por um **índice**, que o identifica de forma unívoca dentro do objecto composto, e um **modelo** associado. O modelo que é associado a cada elemento da composição condiciona o tipo de apresentação que é originada por activação de uma instância de execução.

Um elemento de um composto de execução pode ter conectados um objecto de dados (**socket de apresentação**) ou outro composto de execução (**socket estrutural**). As acções de conexão e desconexão (*plug actions*) permitem mudar dinamicamente a apresentação, por substituição dos objectos de dados, ou a própria estrutura e as relações de propagação entre objectos.

O **estilo de interacção** com o utilizador é definido por comportamentos (*set style actions*) aplicados a um *socket*. Os estilos **menu** e **scrolling list** são aplicados a *sockets* estruturais, os estilos **botão**, **slider** e **campo de entrada** a *sockets* de apresentação.

Nos três primeiros casos, e se o item de selecção for descrito por uma etiqueta (*label*), é possível definir como modelo associado essa mesma etiqueta. No decorrer da execução é dinamicamente criado um conteúdo temporário contendo essa informação.

A possibilidade de **selecção** ou de **modificação** de um objecto é definida por um determinado **estilo de interacção** e é controlada por acções específicas (*set selectability*, *set modifiability*).

Uma selecção gera um evento de selecção correspondente. Uma modificação, além de gerar um evento, altera os dados existentes no objecto de dados original. Caso não seja conveniente a alteração do conteúdo original, é possível efectuar primeiro uma cópia desse conteúdo para outro objecto, usando a acção *copy*.

A apresentação de informação monomédia perceptível pelo utilizador é obtida por conexão de objectos de dados ou de dados multiplexados a *sockets* de apresentação. Relações de propagação entre objectos são descritas pela conexão de outros compostos de execução, a *sockets* estruturais e de forma recursiva, permitindo a criação de novos cenários de apresentação perceptíveis ao utilizador.

É possível também deixar uma conexão em aberto, permitindo efectuar dinamicamente uma alteração estrutural ou da apresentação.

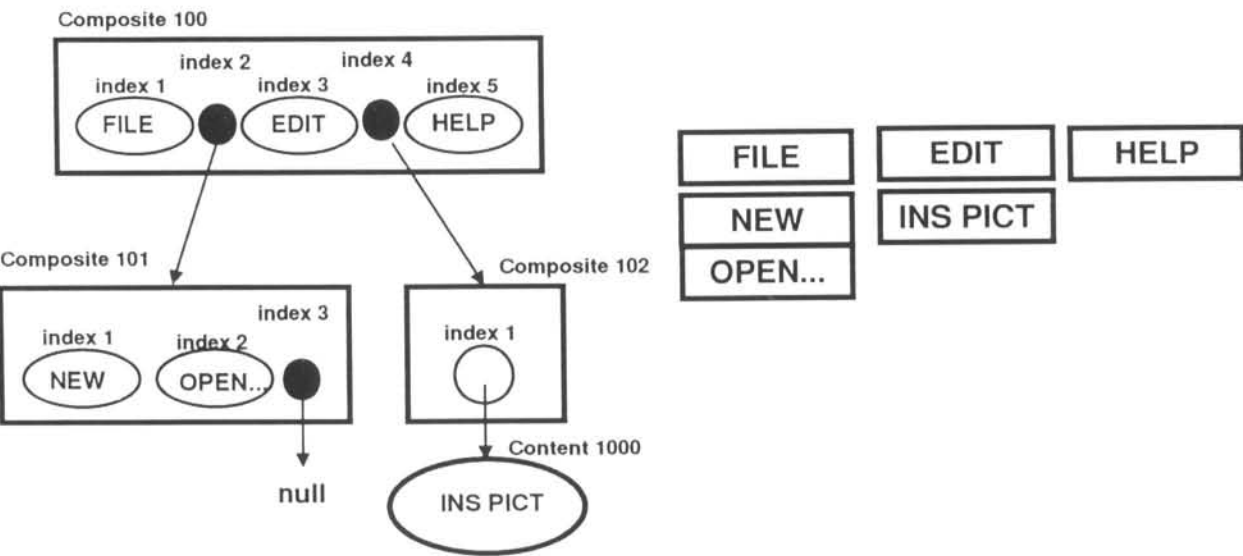


Figura 3.14: Objecto composto e apresentação correspondente.

Na figura 3.13 apresenta-se um exemplo de um objecto composto com algumas das características referidas. A composição consiste num *menu* com vários níveis de elementos seleccionáveis, representados por objectos compostos integrados recursivamente no objecto principal. Os vários itens de selecção são representados por etiquetas, à excepção do elemento indexado pelo composto 102, que é representado por um objecto de dados que pode ser alterado dinamicamente. O composto 101 possui uma conexão em aberto, o que também permite alterar dinamicamente o comportamento do *menu* de selecção.

3.2.5 Mecanismos de Identificação

O MHEG fornece mecanismos genéricos que permitem dentro de uma aplicação identificar univocamente qualquer entidade. As entidades podem estar incluídas ou ser referenciadas por outras entidades, existindo facilidades que cobrem essas situações. Existem três tipos de mecanismos de identificação e uma entidade pode ser identificada de várias formas simultaneamente.

Identificação externa

É possível identificar qualquer objecto MHEG utilizando uma das seguintes identificações externas:

- identificador **público**, registado segundo a norma ISO 9070, ou noutro formato reconhecido por outras organizações ou privado,
- identificador de **sistema** que permite explicitar um acesso a uma base de dados, a uma rede ou a um disco local de um sistema em particular.

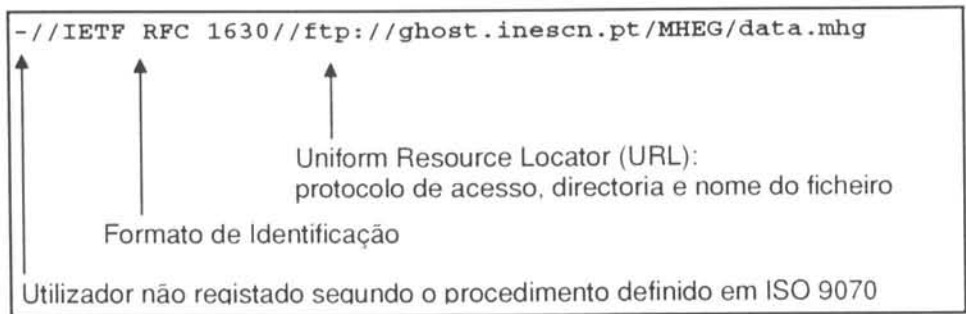


Figura 3.15: Exemplo de identificação externa utilizando o formato IETF RFC 1630 (URL).

Identificação interna

Uma identificação interna pode ser atribuída a qualquer entidade sob controlo do engenheiro MHEG (objectos, objecto de execução, e canais de apresentação) e consiste em:

- um **identificador**, que deverá ser único numa aplicação; o identificador de um objecto MHEG é constituído por um identificador opcional da aplicação e um número do objecto; um objecto de execução é identificado pelo identificador do objecto modelo que lhe deu origem e por um número; um canal de apresentação é identificado somente por um número;
- uma **indexação**, caso o objecto pertença a uma entidade estruturada: elementos de um objecto composto, contentor ou *socket* são identificados por uma lista constituída por uma identificação do objecto exterior e por uma sequência de índices que descrevem a posição do elemento nos vários níveis de recursividade da estrutura.

As referências a entidades indexadas utilizam o identificador de indexação, seguido de uma indicação (*Tail*), que permite referenciar uma só entidade (*single*), as entidades de um nó (*child*) ou todas as entidades que descendem de um nó (*descendant*).

Os objectos de execução podem ser referenciados **dinamicamente** recorrendo a caracteres especiais (*wildcards*). É possível referenciar dinamicamente todos os objectos criados a partir de um objecto modelo, ou só um em particular, utilizando como índice os caracteres * ou ?. Um *socket* pode ser identificado dinamicamente numa estrutura utilizando os caracteres ?, *?child* ou *?descendant*.

Identificação simbólica

É ainda possível associar uma identificação simbólica a qualquer dos tipos de identificação anteriormente referidos. Esta identificação complementar é normalmente conhecida por *alias*.

3.3 Informação de *Script*

Em muitos domínios de aplicação, a representação da estrutura multimédia e hipermédia da informação definida pelo MHEG não é suficiente para descrever todas as funcionalidades desejadas. Relações arbitrariamente complexas entre os objectos e facilidades não directamente relacionadas com a apresentação (decisão, processamento, controlo) devem também ser

passíveis de representação normalizada. Para conseguir estes objectivos surgiu a informação de *script*.

3.3.1 Funcionalidades de um *Script*

O conceito de *script* é bastante geral e aplica-se em muitos campos não relacionados com o domínio da representação multimédia. Funcionalidades como procedimentos de instalação ou inicialização, controlo de acesso e segurança, acesso a periféricos ou bases de dados, fazem parte do espectro genérico de utilização de linguagens de *script*.

Na área dos serviços multimédia pode identificar-se um conjunto mínimo de funcionalidades que devem ser suportadas pela linguagem de representação de *script*. Essas funcionalidades inserem-se numa de duas categorias: a exploração de resultados ou o controlo de processos externos.

A **exploração dos resultados** de interações com o utilizador refere-se a processamentos de decisão não triviais, impossíveis de representar com os mecanismos simples fornecidos pelo MHEG (*Actions* e *Links*).

O **controlo de processos externos** consiste em funcionalidades de acesso a periféricos variados (comunicação, leitor de cartões), ou a serviços não controlados por objectos MHEG, como interrogações a bases de dados ou transações bancárias.

Um *script* deve fornecer algumas características básicas de representação que permitam traduzir em funcionalidades operações de controlo, sequência e sincronização entre elementos multimédia. Nomeadamente são identificadas as seguintes características de um *script*:

- expressão de **estruturas de controlo**: operações de controlo de fluxo traduzidas por iterações (*loops*) e condições (*if*) tradicionais,
- processamento e manipulação de **variáveis**: memorização de contextos, operações de processamento e mecanismos de decisão,
- cooperação com o **engenho MHEG**,
- acesso a **processos externos**.

3.3.2 Níveis de Representação de um *Script*

O intercâmbio de informação de *script* apresenta requisitos distintos consoante o receptor dessa informação é um produtor ou um consumidor. Esta distinção está associada a dois níveis de descrição de linguagens de *script*: uma linguagem, para estruturação e edição da informação de *script*, e uma representação em formato final, com objectivos de execução.

Uma entidade produtora de informação necessita de uma **linguagem de *script*** que permita descrever as funcionalidades desejadas de forma legível. Uma linguagem de *script* permite a edição e alteração destas funcionalidades, constituindo um formato de representação não definitivo. As linguagens de *script* fornecem também, usualmente, facilidades de codificação para representações não editáveis e executáveis. Existem inúmeros exemplos de linguagens genéricas com estas características como Basic, C, Hypertalk ou OpenScript.

A especificação de uma linguagem com essas características, e suportada numa descrição sintáctica normalizada e amplamente divulgada como é o SGML [SGML 86], é o objectivo dos esforços conjuntos do ITU-T e da ISO conhecido como ***Standard Method for Scripting Language (SMSL)***

O intercâmbio de informação de *script*, com total portabilidade e independentemente da linguagem original, deve ser baseado numa representação que possibilite expressar genericamente a informação procedimental de forma codificada. Esse tipo de metodologia, que consiste na compilação para uma linguagem intermédia independente da plataforma, é comum nos vários ambientes de *script* existentes, como o *ScriptX*.

Nos organismos de normalização está a ser considerada a utilização de um formato intermédio, com características poderosas de compactação e reversibilidade, baseado em extensões à linguagem *Architecture Neutral Distribution Format (ANDF)*.

A necessidade de um formato adequado para transmissão em tempo real e para execução no terminal consumidor, tomando em conta aspectos de compactação, eficiência e protecção do código, levou à definição de uma representação codificada denominada ***Script Interchange Representation (SIR)***.

Esta representação consiste num formato intermédio, independente da plataforma utilizada, mas também final, no sentido em que não necessita de informação adicional para ser executado, nem pode ser alterado.

O estudo das extensões do MHEG necessárias para a comunicação entre o engenho MHEG e um processo de *script*, bem como a codificação adequada para o intercâmbio e execução dos *scripts*, é objecto da terceira parte da norma MHEG, conhecida como **MHEG/S** [MHEG 94b], que se encontra ainda em fase de desenvolvimento.

3.3.3 Extensões para o suporte de *Scripts*

Em algumas aplicações complexas, como jogos, simulações ou treino assistido por computador, a coordenação dos fluxos de informação, da sincronização e de sequência entre os objectos é controlada por processos monitores.

Por outro lado, muitas aplicações podem ser descritas como sequências simples de objectos relacionados, com alguns requisitos complementares de funcionalidades, como a gestão de um periférico ou a gravação de um ficheiro, despoletadas pela interacção com os objectos da apresentação.

Esses dois tipos de situações levam a considerar que o processamento de um *script* é gerido ao mesmo nível que a apresentação dos objectos pelo engenho MHEG. Esse facto implica que a comunicação entre os dois processos deverá ser **bidireccional**, ou seja, que é tão importante o *script* poder controlar o engenho MHEG como o contrário.

Estas considerações, associadas à necessidade de controlo de processo externos, implicam que o formato de representação SIR, interpretado pelo processador de *scripts*, deverá constar de três blocos complementares: um núcleo, uma linguagem de interface com o engenho MHEG e um mecanismo para acesso a ambientes externos.

O **núcleo** (*Kernel*) deverá conter um conjunto mínimo de instruções que possibilitem a criação de estruturas de controlo e a manipulação de variáveis (IF, JUMP, GOTO, LABEL). O acesso a funcionalidades externas, como bibliotecas de funções, e o interface com o engenho MHEG e outros processos serão suportados por uma instrução CALL.

Este núcleo é descrito numa notação do tipo pseudo-código (*p-code*), cuja principal característica é usar uma pilha (*stack*) em vez de registos. Esta descrição obedece aos requisitos anteriormente apontados, para uma linguagem intermédia em formato final, de portabilidade, compactação, eficiência e irreversibilidade do código.

A representação intermédia do *script* assim descrita é codificada na sintaxe ASN.1 [ASN.1 87] ou noutras notações isomórficas desta.

A linguagem de mediação, *Intermediate Language for Manipulation of MHEG objects* (ILM), deverá servir para a manipulação dos objectos MHEG, e consiste numa representação abstracta das operações que um *script* pode invocar num engenho MHEG. Com base na especificação ILM, uma dada implementação converte este interface numa linguagem de programação específica.

A definição de um interface normalizado com o engenho MHEG é objecto do esforço mais recente das organizações de normalização nas áreas das telecomunicações ITU-T e ETSI [AVI 93b], no contexto dos serviços áudio-visuais interactivos, denotando influências dos trabalhos da ISO em interfaces de programação [API 93 e PREMO 94a].

O mecanismo para acesso a processos exteriores, será usado para representar possíveis mapeamentos de ambientes de execução *Run-time Environment Mappings* (REM); específicos para acesso ao sistema operativo, a bases de dados e a outros serviços e periféricos. A definição de técnicas de mapeamento, unicamente, permitirá incorporar virtualmente qualquer processo externo e ultrapassa as dificuldades inerentes à inexistência de normalização em várias áreas.

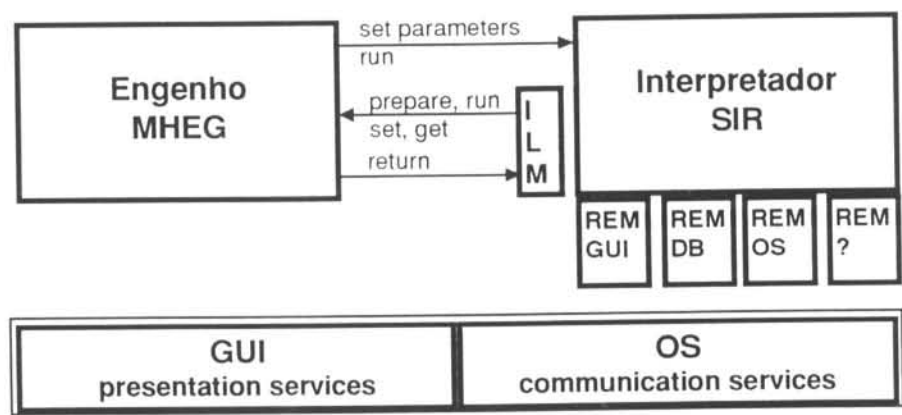


Figura 3.16: Esquema de blocos de um sistema de apresentação multimédia.

Dos pontos anteriores resulta o esquema de blocos da figura 3.15, que representa o conjunto de processos envolvidos numa apresentação complexa, envolvendo interações entre o engenho MHEG e um interpretador de *scripts*.

3.4 Autoria e Criação de Informação

A criação de informação multimédia estruturada é um conceito recente e, por vezes, uma apresentação multimédia chega a ser considerada um produto de arte [Harnden+ 93]. Contudo, o processo de normalização internacional na área multimédia prosseguido, pela ISO desde o início da década, parte da analogia entre os documentos convencionais e os documentos multimédia. O conceito de documento está tão entranhado na vivência humana que, mesmo nos anos mais recentes, ouvimos apregoar soluções baseadas em documentos multimédia, como a tecnologia *Object Linking and Embedding* da Microsoft [OLE 93].

3.4.1 HyperODA

A norma ISO/ITU-TS *Open Document Architecture* (ODA) baseia-se na visão clássica de um documento, em que a estrutura lógica está separada da estrutura de apresentação. Actualmente estão em discussão extensões a este modelo que permitam definir estruturas não lineares, como por exemplo referências a outros documentos e a informação multimédia, ou relações temporais entre elementos de apresentação [HyperODA 93].

A distinção funcional entre os domínios do HyperODA e do MHEG, apresentada na figura 3.17, considera que o primeiro consiste numa ferramenta de estruturação lógica da informação, permitindo um conjunto limitado de funcionalidades de apresentação de que está ausente, por exemplo, a interactividade. Enquanto que o MHEG consiste num formato de representação final que descreve completamente as relações espaciais, temporais e lógicas entre elementos da apresentação.

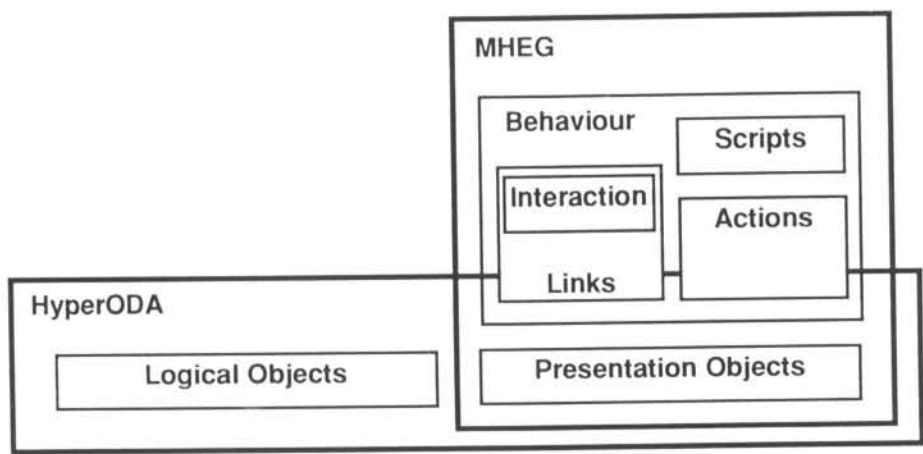


Figura 3.17: Domínios funcionais do MHEG e do HyperODA.

3.4.2 HyTime

A norma *Hypermedia/Time-based Document Structuring Language* [HyTime 92] é uma infraestrutura normalizada para a representação de uma forma abstracta, de relações espaciais e temporais ou de ligações (*links*) entre objectos multimédia. Um documento HyTime é uma colecção de construções semânticas representadas por convenções sintácticas definidas segundo a norma *Standard Generalized Markup Language* [SGML 86].

Esta estrutura permite que engenhos HyTime efectuem operações de procura, formatação, interrogação, ou mesmo de apresentação do documento, ainda que não suportem alguns objectos multimédia nele definidos. É portanto possível garantir um máximo de interoperabilidade entre hiperdocumentos, sem requerer a normalização dos objectos multimédia e da semântica definida para as ligações entre objectos.

Um documento pode conter *scripts* em SMSL, para exprimir comportamentos específicos mais complexos, e a estrutura abstracta e reversível definida pelo HyTime pode ser codificada num formato final, como o MHEG, para permitir a apresentação de modo eficiente.

A arquitectura concebida para um dado documento em SGML, *Document Type Definition*, é construída a partir de classes (*architectural forms*), que encapsulam as várias funcionalidades definidas pelo HyTime. Estas classes estão agrupadas em diversos módulos que suportam facilidades básicas, funções de localização de objectos e a especificação de relações hipermédia e espaço-temporais.

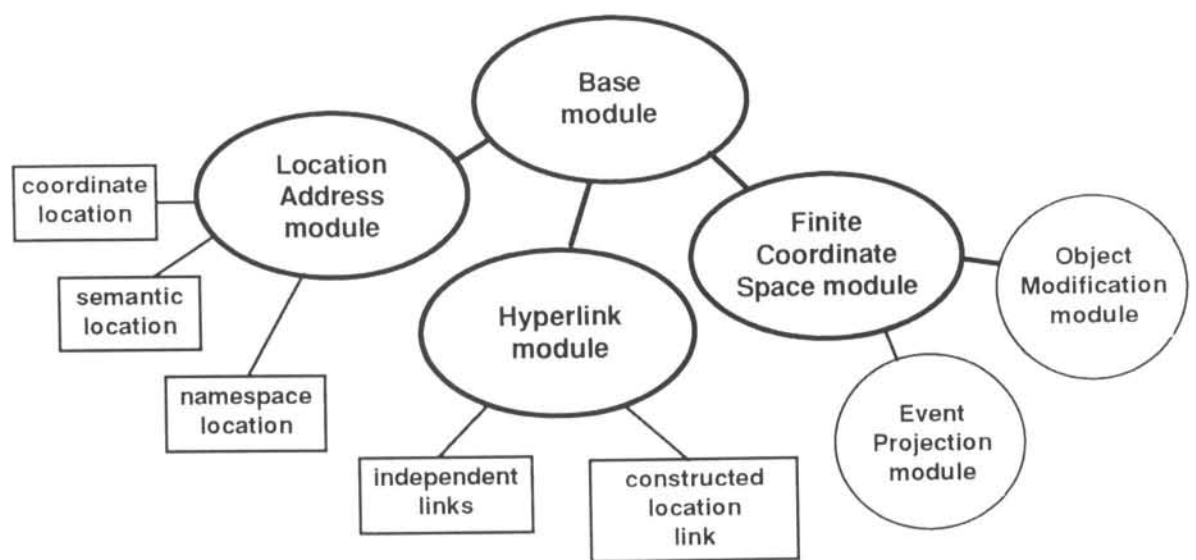


Figura 3.18: Decomposição do HyTime em módulos.

Base module

Este módulo inclui um conjunto de facilidades básicas de gestão e identificação de hiperdocumentos, e permite a geração de expressões não normalizadas (*xenofoms*). São ainda fornecidas, entre outras, facilidades para endereçamento por coordenadas e detecção de alterações (*activity-tracking*) num elemento, por meio de notificações.

Location Address module

O módulo de localização fornece identificadores locais únicos para cada elemento de um documento. Os elementos externos podem ser endereçados por nome se pertencerem a outros documentos SGML. Se estiverem localizados num determinado espaço n-dimensional podem ser referenciados por vários métodos de endereçamento (*string*, *token*, *tree*, *path* e *FCS location addresses*). Existe ainda a possibilidade de efectuar endereçamentos semânticos, relacionados com o conteúdo específico dos elementos.

Hyperlink module

Este módulo define cinco classes (*independent*, *property*, *contextual*, *aggregate* e *span links*), que podem ser combinadas de modo a cobrir todo o tipo de necessidades de *links*, desde os mais clássicos às variantes múltiplas, bidireccionais ou agregadas.

Finite Coordinate Space module

A ligação de objectos multimédia (*hyperlinks*) e a sincronização temporal entre estes são aplicações distintas dum conceito de **endereçamento homogéneo** num espaço e num tempo virtuais, ou mesmo num espaço n-dimensional, se considerados outros domínios de medida. Um objecto num espaço destes ocorre como conteúdo de um **evento** de coordenadas bem definidas. Os eventos são organizados por gestores denominados *event schedulers*.

Podem ser definidos dois módulos opcionais sobre um espaço, que controlam a projecção dos eventos e as modificações efectuadas sobre os objectos, quando são mapeados noutro espaço. O módulo de **projecção** considera projecções (*projectors*) de objectos em posições do espaço n-dimensional (*proscopes*) e gestores denominados *batons*. A **modificação** de objectos não tem impacto sobre a posição ou duração dos eventos sobre os quais actua e a sua semântica só é minimamente especificada no HyTime. Os modificadores (*modscopes*) de um objecto são organizados por gestores denominados *wands*.

Capítulo 4

Modelos de Computação Distribuída

“Um sistema distribuído é aquele que nos impede de trabalhar devido à falha de uma máquina de que nem sequer ouvimos falar.”

L. Lamport

Nos últimos anos, o declínio dos custos de hardware e o rápido aumento do desempenho dos computadores pessoais bem como o aparecimento de infraestruturas de comunicação mais acessíveis e com maior capacidade provocaram uma revolução na utilização dos computadores em rede. Contudo o modelo de desenvolvimento de aplicações mudou surpreendentemente pouco, sendo ainda hoje predominante a arquitectura cliente-servidor.

Embora as poderosas redes actuais e o relativo baixo custo das estações de trabalho tornem a descentralização viável, apenas em áreas em que havia já necessidades concretas surgiram aplicações distribuídas. Não é por acidente que as aplicações em rede mais comuns são o correio electrónico, os serviços de impressão e de partilha de ficheiros.

Este capítulo pretende introduzir a problemática dos sistemas distribuídos actuais, e questionar da sua adequação à implementação de sistemas e serviços multimédia.

4.1. Evolução dos Sistemas Distribuídos

Um sistema distribuído apresenta diferenças significativas em relação a um sistema centralizado no que respeita à heterogeneidade e ao âmbito geográfico. Se estes são os atributos que poem problemas à gestão dos sistemas distribuídos, em termos de segurança e comunicação, constituem também os seus trunfos principais: um sistema distribuído possui uma modularidade inerente que, aliada às características mencionadas, permite uma escalabilidade e uma tolerância a falhas que não está ao alcance dos sistemas centralizados.

As tecnologias dos sistemas distribuídos têm evoluído no sentido de uma cada vez mais completa integração dos vários elementos envolvidos [Veríssimo 94].

A **partilha de recursos** foi o primeiro benefício introduzido, materializado em dois mecanismos: a sessão remota (*rlogin*) e a transferência de ficheiros (*ftp*).

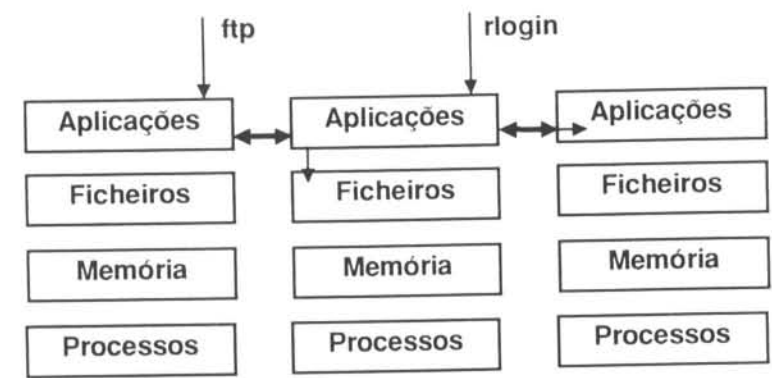


Figura 4.1: Partilha de recursos.

Segundo a óptica dos sistemas operativos baseados em disco, a partilha eficaz de informação passava por evitar tanto quanto possível a transferência explícita de ficheiros, nascendo assim os sistemas de **ficheiros distribuídos**.

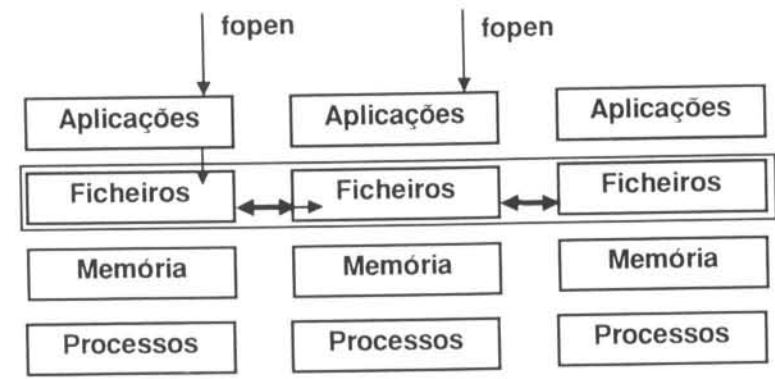


Figura 4.2: Ficheiros distribuídos.

Inevitavelmente, da partilha de aplicações e de dispositivos de hardware chegou-se à partilha de poder de cálculo e memória, sobretudo a partir da instalação de infraestruturas de redes locais com débito elevado (>10Mbit/s).

No sentido de poder suportar programação concorrente em diferentes máquinas, e assim abrir caminho ao processamento distribuído, surgiram sistemas com **memória partilhada distribuída** (DSM). Neste modelo é possível suportar actividades concorrentes, sincronizadas sobre uma ilusoriamente partilhada memória distribuída, beneficiando dos recursos de uma rede de máquinas de forma transparente.

Outro modelo adoptado consiste em ter actividades cujos processos se encontram em várias máquinas, normalmente associado a um estilo de programação com **invocação a procedimentos remotos**.

As aplicações distribuídas podem assim obter maior desempenho por distribuição de tarefas muito intensivas do ponto de vista computacional, como computação gráfica, a máquinas de maior velocidade de processamento e capacidade.

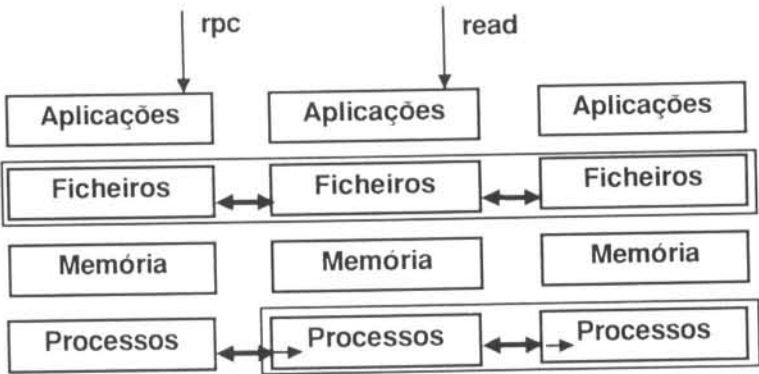


Figura 4.3: Sistemas distribuídos.

Os factores chave da evolução dos sistemas distribuídos residem no estudo dos problemas resultantes da heterogeneidade das plataformas e da funcionalidade das redes actuais. Aspectos de extrema importância são a segurança, a fiabilidade e a congruência do funcionamento das aplicações.

Outros desafios colocam-se no suporte a novos tipos de aplicações distribuídas que usam o conceito de grupo e no estudo dos processos de concorrência, cooperação e de partilha de funcionalidades e de carga.

4.2. Modelos de Distribuição

Os vários modelos de programação propostos nos sistemas distribuídos actuais consistem:

- na invocação de **operações remotas**, utilizada normalmente no modelo de distribuição cliente-servidor;
- em operações fiáveis denominadas **transações**,
- na utilização de **memória partilhada distribuída**, que é uma forma de distribuição indirecta transparente à distribuição dos processos por várias máquinas,
- num paradigma de **difusão** multiponto em circuito aberto, baseado na passagem de mensagens com semânticas de entrega variadas, que é particularmente adequado para a concepção de aplicações para grupos de trabalho.

O modelo de comunicação mais comum associado ao modelo de distribuição do tipo cliente-servidor é conhecido por **Remote Procedure Call (RPC)**. A invocação de operações remotas fornece aos programadores uma interface familiar, de alto nível, com os vários componentes distribuídos da aplicação. Questões, como a localização do serviço requerido, a segurança e o encapsulamento de dados, são tratadas transparentemente.

Externamente, uma RPC é igual a qualquer chamada procedimental, no que respeita ao seu interface, ao valor de retorno e aos parâmetros suportados. Contudo internamente uma RPC difere substancialmente de um procedimento vulgar, pois exige um código específico (*stub*) para se conectar ao componente distribuído do qual requer uma acção específica.

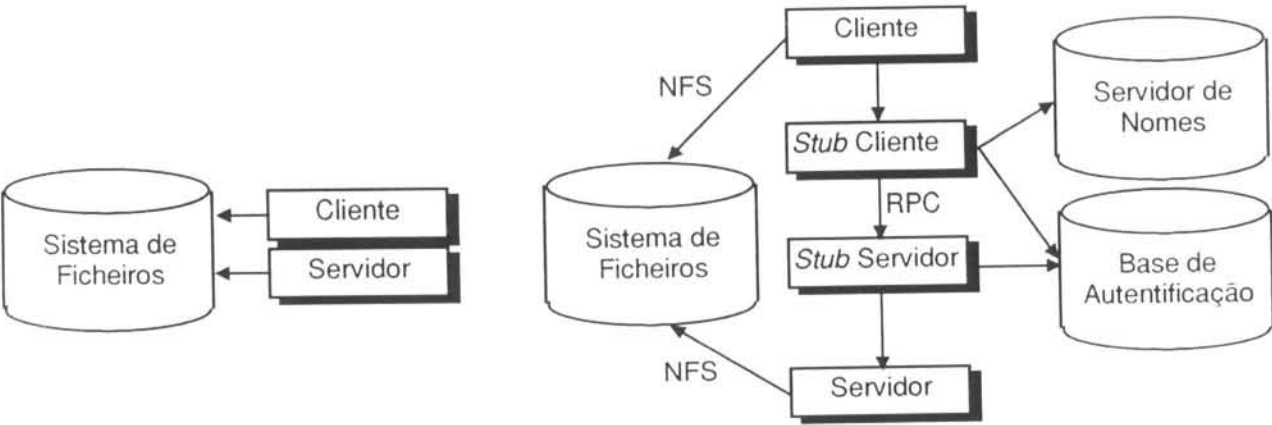


Figura 4.4: Aplicação distribuída por invocação de RPC.

O modelo de distribuição baseado em chamadas a procedimentos remotos é facilmente suportado por qualquer tecnologia de rede utilizada. O suporte de linguagens de programação procedimentais clássicas permite também uma evolução sem dor e garante a compatibilidade com grande parte das infraestruturas de sistemas e serviços instaladas.

Estas características explicam a evolução lenta que tem existido na exploração da flexibilidade e configurabilidade das novas redes para o desenvolvimento de novos tipos de aplicações distribuídas.

Contudo, o modelo de cliente-servidor tradicional apresenta problemas tanto do ponto de vista da funcionalidade como da confiabilidade.

No que respeita à **funcionalidade**, a invocação de procedimentos remotos é por definição bloqueante, unilateral e assimétrica. Os sistemas mais recentes baseados em RPCs tentam obviar a essas limitações criando automaticamente actividades (*threads*) para cada pedido efectuado, num processo conhecido por *fork-and-call*.

As aplicações distribuídas emergentes, do tipo multiparticipante, exigem um modelo de distribuição baseadas em interacções não bloqueantes (assíncronas), multilaterais e simétricas, ou *peer-to-peer*, que são geralmente baseadas em mecanismos de passagem de mensagens.

Relativamente à **confiabilidade**, a semântica básica dos RPCs é do tipo “pelo-menos-uma-vez”, dada a dificuldade em destringir entre uma falha de comunicação e uma falha no servidor, e pode, com ligeiras alterações, ser convertida numa semântica “no-máximo-uma-vez”.

O modelo de **transacções** vem possibilitar a execução de operações remotas com semântica “exactamente-uma-vez”, o que é extremamente útil para suportar interacções fiáveis com repositórios de dados persistentes.

A programação orientada para grupos, campo inovador e de interesse crescente, baseia-se no modelo de **difusão**. Este consiste num conjunto simples de premissas que permitem definir semânticas de comunicação fiável entre os elementos do grupo, tipos de actividades dos elementos e protocolos de gestão do grupo.

As actividades básicas consideradas são a **replicação** dos serviços, a **cooperação** entre os elementos do grupo e a **partilha** dos recursos do sistema. Os grupos são um modelo bastante estruturante do ponto de vista da comunicação distribuída, suportando todos os estilos de programação anteriormente referidos.

4.3. Ambientes de Computação Cliente-Servidor

Os ambientes de computação cliente-servidor consistem numa colecção integrada de serviços de rede que suportam o desenvolvimento, o uso e a manutenção de aplicações distribuídas.

Os componentes nucleares de um ambiente de computação distribuída como o *Distributed Computing Environment* (DCE), desenvolvido pela *Open Software Foundation* (OSF), permitem suportar vários tipos de funcionalidades [Shirley 92].

Modelo de comunicação

Os ambientes existentes actualmente são baseados em comunicação por RPCs e incluem ferramentas de geração automática dos componentes funcionais (*stubs*) que encapsulam os detalhes de comunicação e permitem recuperar de erros de comunicação. Os ambientes libertam assim os programadores dos problemas relativos à comunicação entre componentes, permitindo que se concentrem no desenvolvimento das funcionalidades da aplicação.

Outra funcionalidade de comunicação importante consiste no empacotamento e desempacotamento de estruturas de dados complexas para transmissão entre sistemas heterogéneos, num processo conhecido por *marshalling*. Antes de serem empacotadas, as estruturas de dados são traduzidas para uma representação intermédia normalizada e independente da representação existente no hardware específico de cada utilizador.

Serviço de Directoria

Num sistema distribuído, um factor crucial para o bom funcionamento reside na colocação correcta dos componentes do serviço, com impacto directo no desempenho, na disponibilidade, na confiabilidade e na segurança de uma aplicação distribuída.

Intimamente ligados ao problema da distribuição dos componentes do serviço está o serviço de directoria, denominado *Cell Directory Service* (CDS) no ambiente OSF/DCE, que é responsável pelo mapeamento dos nomes dos componentes da aplicação nos respectivos endereços físicos.

Este serviço básico é escalonado em células, que são as unidades fundamentais de administração. Dentro de cada célula as aplicações partilham os mesmos serviços de directoria,

administração e segurança. Os serviços registam perante o serviço de directório o seu nome e localização. O acesso a serviços de outras células é efectuado por conexão a um serviço global de directório; *Global Directory Service* (GDS).

No ambiente OSF/DCE o serviço DCS está integrado com o serviço de RPCs, permitindo gerar *stubs* que localizam o serviço e estabelecem a conexão.

Uma funcionalidade particular do CDS é que permite que os serviços incluam informação dos seus atributos no seu registo perante o directório. A procura de um serviço com recurso ao CDS poderá assim conduzir a decisões inteligentes, por parte do cliente, na escolha entre duplicações de um dado serviço (por exemplo: a escolha de impressoras com *postscript* ou laser, ou simplesmente a mais próxima do utilizador).

Infelizmente, esta funcionalidade é algo limitada naquilo em que poderia ser mais poderosa. Se o serviço CDS efectuasse ele próprio a filtragem dos atributos dos vários serviços e se a informação descrita nos atributos fosse dinâmica, seria possível introduzir mecanismos de gestão inteligente na rede, como a gestão de filas de espera de clientes de serviços.

Segurança

As questões de segurança são muito importantes em sistemas distribuídos. Numa aplicação distribuída por dois ou mais componentes é difícil garantir que os componentes são aquilo que reclamam ser. Mesmo que o sejam, a comunicação entre ambos é insegura e, portanto, vulnerável a um ataque. As questões de segurança podem subdividir-se em três categorias: autenticação, autorização e confidencialidade.

As questões de **autenticação** e de **confidencialidade** são resolvidas por um conjunto de facilidades interligadas, nomeadamente por recurso a encriptação por chave pública baseada em funções matemáticas não reversíveis, que permitem também a autenticação por geração de assinaturas digitais. Todas estas facilidades estão também integradas no modelo de comunicação DCE/RPC.

A funcionalidade de **autorização** controla o acesso e os privilégios de serviços, ficheiros e outros recursos do ambiente distribuído. No ambiente DCE, a autorização é controlada por meio de listas de acesso, *Access Control Lists* (ACL), que são administradas de forma centralizada e permitem enorme flexibilidade e granularidade na atribuição de permissões.

Esta facilidade não está integrada no modelo de comunicação, sendo os programadores responsáveis por incluir nos seus serviços o código necessário a decisões de autorização, recorrendo a uma API fornecida com o ambiente.

Ficheiros Distribuídos

Os sistemas de ficheiros dos sistemas operativos tradicionais não foram pensados para suportar operações distribuídas. Contudo, hoje encontramos várias extensões aos sistemas mais populares que permitem um acesso transparente, por rede, a discos remotos. É o caso do *Network File System* (NFS) da *Sun Microsystems*.

No entanto, estas extensões não apresentam uma visão lógica única da rede e dos vários sistemas de ficheiros existentes, que permita obter um espaço de ficheiros uniforme que atravesse as fronteiras das várias máquinas.

O sistema de ficheiros distribuídos do ambiente OSF/DCE, denominado *Distributed File System* (DFS), fornece uma visão consistente e global do sistema: qualquer ficheiro ou disco pode ser referenciado por meio de um nome único no sistema, que especifica a máquina em que reside, a célula administrativa, etc.

4.4. Modelos Orientados para Objectos

Actualmente, é consensual a importância do uso de tecnologias orientadas para objectos no desenvolvimento de novos sistemas operativos, serviços distribuídos e bases de dados.

Lenta mas seguramente, as linguagens orientadas para objectos, como o C++ e o *SmallTalk*, tem vindo a ser escolhidas pelos programadores em detrimento das linguagens procedimentais convencionais.

As vantagens usualmente apontadas são várias, destacando-se a possibilidade de criação de módulos de software reutilizáveis em contextos diferentes dos da sua concepção, recorrendo aos mecanismos usuais nas linguagens orientadas por objectos [Nierstrasz 89].

Contudo, a falta de arquitecturas normalizadas impede que objectos criados em linguagens ou em sistemas diferentes comuniquem entre si, conduzindo a um mar de objectos coexistentes mas não dialogantes.

Por outro lado, a linguagem C++, a mais utilizada para programação orientada por objectos, nunca foi pensada numa lógica de reutilização de objectos em grande escala, mas sim no âmbito de um projecto ou a nível departamental.

A utilização de facilidades poderosas já conhecidas das linguagens procedimentais, como as bibliotecas dinâmicas de software, *Dynamic Linking Libraries* (DLL), torna-se complexa em linguagens orientadas para objectos por falta de compatibilidade binária entre compiladores, na representação dos objectos.

Os sistemas distribuídos orientados para objectos dão um novo fôlego a esta questão, introduzindo o conceito de componentes de software, que promete ser o motor de mudanças dramáticas em vários sectores de negócio e na sociedade em geral. Os componentes de software são uma nova e mais produtiva forma de concepção, programação e venda de software, com impacto nos vendedores de software, nos utilizadores e nas empresas.

Os modelos orientados para objectos são extremamente adequados para arquitecturas de processamento distribuído uma vez que os conceitos de encapsulamento e componente são coerentes com as propriedades básicas da distribuição: a separação, o isolamento e a autonomia.

Além disso, os conceitos de modularidade e abstracção de dados facilitam a estruturação e especificação do serviço, e as interações entre os vários objectos são representadas pelo comportamento destes, de forma independente da sua estrutura interna.

A utilização de componentes de software em sistemas abertos exige a definição de um modelo geral de objectos e de um conjunto de serviços básicos, associados à procura e à conexão (*linking*) dinâmica de componentes. A primeira tentativa de normalização efectuada a este nível partiu do **Object Management Group (OMG)**, uma organização internacional fundada em 1989 para promover o uso de tecnologia orientada por objectos.

Trabalhando em estreita relação com os organismos internacionais de normalização e com os maiores grupos de fabricantes interessados, o OMG publicou o documento *Object Management Architecture* [OMG 90], que especifica um modelo de objectos e um conjunto básico de serviços necessários à utilização destes em sistemas distribuídos.

4.4.1 O Modelo de Objects OMG

O modelo de objects OMG define uma semântica comum para especificar as características dos objects visíveis do exterior, de forma normalizada e independente da implementação escolhida por cada programador. O modelo descreve ainda conceitos relacionados com a implementação dos objects.

Um objecto é uma entidade identificável e encapsulada, que fornece um ou mais serviços que podem ser requeridos pelos clientes. A identificação unívoca do objecto é efectuada por meio de uma ou mais **referências** distintas a ele (*object reference*).

As características visíveis de um objecto são descritas por meio de um *interface*, que descreve um conjunto de **operações** possíveis (*operation signatures*) suportadas pelo objecto e que um cliente pode invocar. As operações modelizam as características visíveis do objecto no que respeita ao seu **comportamento** (*behaviour*) e ao seu **estado** (*state*).

Os interfaces são descritos utilizando a linguagem *Interface Definition Language* (**IDL**), que permite especificar as operações e os atributos por que são constituídos.

A declaração de um **atributo** (*attribute*) é logicamente equivalente a declarar um par de operações para obter (*get*) ou atribuir (*set*) um valor ao atributo. Opcionalmente o atributo poderá suportar unicamente a operação de leitura (*readonly*).

(readonly) attribute <simple_type_spec><base_defs> (4.1)

O formato geral de uma **operação** é o seguinte:

(oneway) <op_type_spec><identifier>(param1,...,paramL) (4.2)
(raises(except1,...,exceptN))(context(name1,...,nameM))

A palavra-chave opcional **oneway** indica que foi seleccionada a semântica de execução do tipo “melhor-esforço”. Uma operação deste tipo não retorna ao cliente o resultado nem este se sincroniza com a conclusão do pedido (operação assíncrona). A semântica usual é do tipo “no-máximo-uma-vez”, correspondendo a “exactamente-uma-vez” se a operação retorna com

sucesso (operação síncrona). É possível a um cliente efectuar uma ou várias operações com sincronismo adiado (*deferred synchronous*), utilizando um interface de invocação dinâmica.

O tipo do resultado que a operação retorna é definido por **op_type_spec** e a operação do interface invocada é especificada no campo **identifier**. Os parâmetros necessários à operação são indicados com os modificadores **in**, **out** ou **inout**, indicando a direcção de fluxo da informação relativamente ao cliente, bem como o tipo de valor que é passado.

As excepções definidas no objecto são indicadas pela expressão opcional **raises**, permitindo sinalizar na variável de retorno um comportamento anormal da implementação do objecto. Todas as operações incluem implicitamente um conjunto de excepções comum.

A expressão opcional **context** define informação contextual que é possível passar à implementação do objecto.

Os tipos são utilizados nas operações para restringir os valores possíveis para cada parâmetro. Para além dos tipos básicos (formatos de inteiros, *floats*, *strings*, ...) e dos tipos estruturados usuais (*records*, *arrays*,...), destacam-se alguns tipos especiais: um tipo **opaco** (*octet*), que não é convertido (*marshalled*) na fronteira entre sistemas, e um tipo (*any*) que pode representar qualquer outro. Tipos para **interfaces** (*interface type*) e **objectos** (*object type*) também são definidos.

Podem estabelecer-se relações hierárquicas entre interfaces utilizando as regras de **herança** (*inheritance*), **sub-tipos** (*subtypes*) e **super-tipos** (*supertypes*). Um objecto é super-tipo de outro se a cada operação definida no seu interface corresponde uma operação no interface do sub-tipo com o mesmo nome e idêntico número e tipo de parâmetros e de resultados. O mecanismo de herança permite definir um tipo a partir de um ou mais super-tipos (herança múltipla), o que significa que um objecto pode suportar múltiplos interfaces.

Uma implementação de um dado tipo de objecto é denominada **classe** (*class*) e um objecto individual, acessível num dado instante, é uma **instância** (*instance*) dessa classe. Uma operação requerida sobre uma instância de um objecto é implementada por execução do **método** (*method*) correspondente, que opera sobre os dados que representam o estado do objecto. Ao requer uma operação sobre um objecto poderá ser necessário criar uma nova instância desse objecto, sendo este processo denominado **activação** (*activation*).

4.4.2 A Arquitectura CORBA

Uma proposta de arquitectura para sistemas distribuídos, orientada para objectos, baseada no modelo do OMG e que reúne amplo consenso, é designada CORBA. A *Common Object Request Broker Architecture* [CORBA 91], consiste numa especificação dos serviços de suporte de uma arquitectura distribuída, que permite a utilização de componentes de serviço de forma transparente. O objectivo é permitir a interoperabilidade entre aplicações em máquinas diferentes e em ambientes distribuídos heterógeneos, e a interconexão de múltiplos sistemas orientados para objectos que cumpram esta especificação.

O *Object Request Broker (ORB)* é responsável por todos os mecanismos necessários para procurar a implementação de um objecto, eventualmente preparar nova instância do objecto para receber a chamada, e comunicar os dados necessários à invocação do método correspondente.

O interface que o cliente vê é completamente independente da localização do objecto, da linguagem de programação em que está implementado ou de qualquer outro aspecto que não esteja traduzido no interface do objecto.

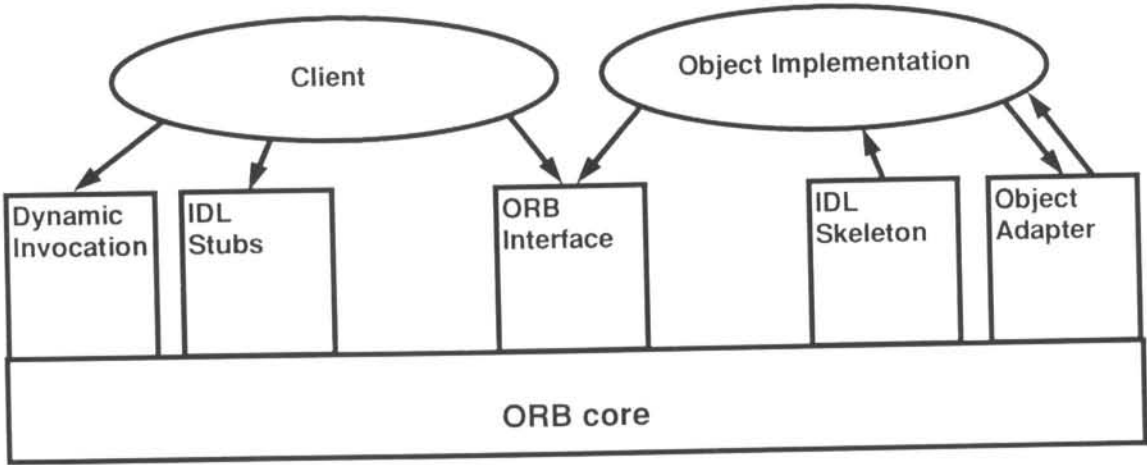


Figura 4.5: Estrutura dos interfaces do *Object Request Broker*.

A arquitectura apresentada na figura anterior define o conjunto dos interfaces do ORB. Esta arquitectura é independente da implementação e não implica que todos os interfaces do ORB sejam implementados num único componente.

Para invocar uma operação, um cliente necessita de ter acesso a uma referência ao objecto que deseja utilizar e de conhecer o interface desse objecto. O cliente inicia a chamada ou por intermédio de rotinas específicas do objecto em causa (**IDL stubs**), ou recorrendo a um interface normalizado que permite a construção de um pedido de forma dinâmica (**Dynamic Invocation**).

Tanto o interface de invocação dinâmica como as rotinas de *stub* satisfazem a mesma semântica, de modo que o objecto invocado não saberá distinguir entre as duas formas de invocação.

O ORB localiza o código de implementação do objecto, transmite os parâmetros necessários e transfere o controlo para a instância do objecto por intermédio de rotinas específicas de *stub* (**IDL skeleton**).

Um *Object Adapter* disponibiliza serviços do ORB de forma adaptada a necessidades de grupos específicos de objectos. A arquitectura que temos vindo a referir define um **Basic Object Adapter (BOA)**, que pode ser usado pela maior parte das implementações convencionais de objectos.

Um exemplo de um *Object Adapter* específico é o de uma base de dados orientada a objectos, OODB. Uma vez que a OODB fornece os métodos correspondentes às operações bem como os estados dos objectos de forma persistente, os objectos podem ser registados de forma implícita, não sendo necessário guardar qualquer estado no Object Adapter.

O BOA é responsável por um conjunto vasto de serviços que devem incluir:

- a geração e interpretação de referências a objectos,
- a invocação de métodos (podendo ser através do *skeleton* correspondente ou não),
- a activação e desactivação das implementações dos objectos,
- mecanismos de autenticação e controlo de acesso,
- o mapeamento de referências nas implementações dos objectos correspondentes,
- o registo das implementações.

Algumas outras operações, comuns a todos os objectos sejam eles clientes ou componentes de serviços, são oferecidas pelo **ORB interface**, que é comum a todas as implementações de ORBs.

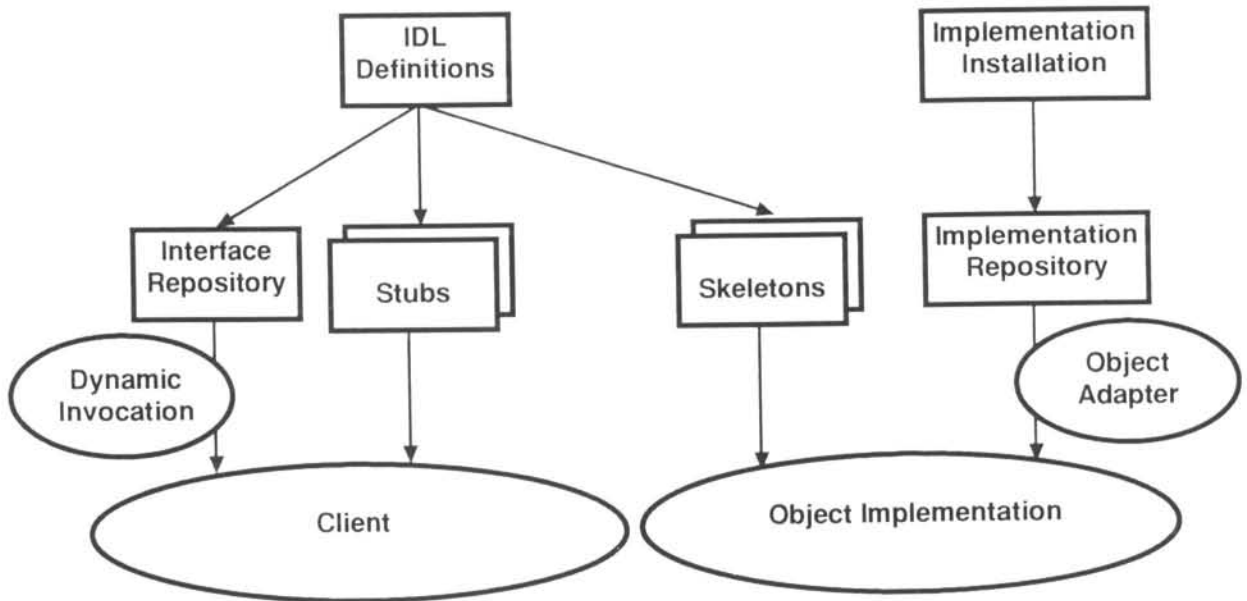


Figura 4.6: Mecanismos de definição de interfaces e de implementação de objectos.

A informação contida numa definição IDL é utilizada para gerar *stubs*, para uso dos clientes, e *skeletons*, para invocação da implementação do objecto pelo ORB. Esta informação também pode ser disponibilizada no **Interface Repository**, num formato adequado para a construção dinâmica de uma operação, utilizando o interface *Dynamic Invocation* do ORB.

Do mesmo modo, a informação de implementação de um determinado objecto pode ser armazenada num **Implementation Repository**, a partir do qual é possível ao ORB não só controlar a instalação, activação e execução de objectos, mas também fornecer uma série de serviços adicionais de grande importância, como funções de controlo administrativo, alocação de recursos ou procedimentos de segurança.

Um servidor pode definir, em cooperação com o BOA, várias políticas possíveis de implementação dos métodos definidos no interface de um objecto. A activação das implementações dos objectos é efectuada de acordo com a política definida para esse objecto:

- *shared server*: quando várias instâncias de um objecto existem num mesmo servidor,
- *unshared server*: quando existe um servidor por instância do objecto,

- *server-per-method*: quando cada invocação de um método é implementada por um servidor,
- *persistent server*: quando o servidor é activado exteriormente ao BOA, funcionando depois numa política de *shared server*.

Um ORB pode estar residente nas implementações do cliente e do objecto, caso esteja disponível um mecanismo de comunicações que actue de forma transparente na localização das implementações no sistema distribuído. Pode também ser baseado em bibliotecas para objectos, cujas implementações possam ser partilhadas, assumindo que o cliente é bem comportado no que respeita à inviolabilidade dos dados do objecto.

O ORB pode ser constituído por programas convencionais do ponto de vista do sistema operativo, que operam como um conjunto de servidores centralizadores da gestão do sistema. Outra opção, mais integrada, considera o ORB como um serviço básico do sistema operativo, permitindo melhorar a segurança e a confiabilidade do sistema e otimizar os processos de autenticação e *marshalling*.

Uma questão importante na arquitectura de um ORB é a **interoperabilidade** entre os vários sistemas que constituem um ambiente distribuído. A utilização de uma linguagem de alto nível, como o IDL, permite a especificação do interface dos objectos de modo independente de qualquer ORB.

Contudo, a especificação independente do interface não resolve completamente o problema da interoperabilidade entre diferentes sistemas: a semântica de transferência de dados utilizada está intimamente ligada a cada ORB, podendo introduzir incompatibilidade binária entre as implementações.

No caso mais simples, o mesmo ORB está implementado em dois sistemas diferentes, estando neste caso a interoperabilidade assegurada, uma vez que as referências a objectos e os mecanismos de comunicação utilizados são os mesmos em ambos os sistemas.

Cada ORB deve também ser capaz de distinguir as referências a objectos sob o seu controlo das referências da responsabilidade de outro ORB, passando estas transparentemente como parâmetros. Esse facto implica que, numa mesma plataforma, é possível coexistirem duas arquitecturas ORB diferentes, podendo um cliente requerer serviços a qualquer uma das implementações.

Se em dois ambientes não há um ORB comum, é necessário introduzir um mecanismo de *gateway* que traduza as referências a objectos e as chamadas de um ORB para o outro.

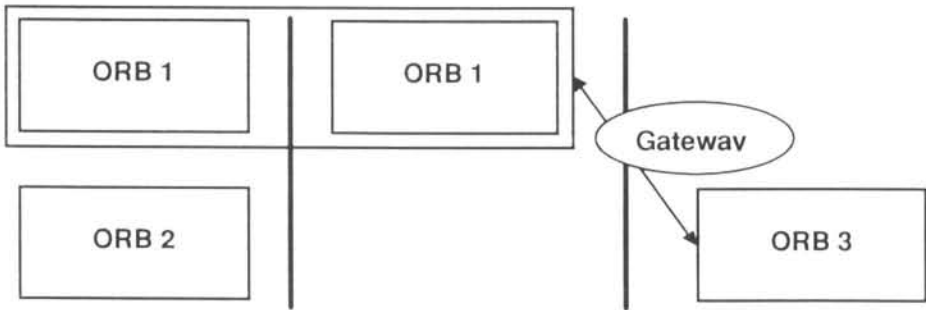


Figura 4.7: Problema da interoperabilidade entre ORBs.

O consenso reunido à volta da especificação do CORBA resultou em grande parte da flexibilidade introduzida na primeira versão, na qual o problema da interoperabilidade entre sistemas ficou em aberto. Entretanto foi já aprovada uma segunda versão, baseada na proposta UNO, que irá assegurar essa interoperabilidade para os produtos da próxima geração.

Um subgrupo do OMG, o *Object Database Management Group* (ODMG), lançou em 1993 uma especificação destinada à portabilidade e interoperabilidade de bases de dados orientadas para objectos (OODB).

Basicamente, esta especificação consiste numa extensão à linguagem de definição de interfaces - *Object Definition Language* (ODL), que fornece um mecanismo para exprimir modelos de objectos, e também numa extensão à linguagem SQL - *Object Query Language* (OQL), que fornece um interface de acesso para interrogações a bases de dados.

Esta especificação inclui ainda a transcrição em linguagem C++ e *SmallTalk*, bem como considerações sobre a relação com as outras recomendações do modelo OMG. À semelhança do processo evolutivo do CORBA, também esta especificação não cobre ainda a interoperabilidade entre bases de dados distribuídas e outras funcionalidades.

4.5. Modelo de Referência para Computação Distribuída Aberta

Os modelos de sistemas distribuídos anteriormente referidos (OSF/DCE e OMG/CORBA) não fornecem uma visão abrangente e um modelo de referência que estabeleçam uma estrutura comum de utilização das tecnologias da informação em sistemas distribuídos.

A norma ISO que define o modelo de computação distribuída tem o nome de *Open Distributed Processing (ODP)*. O modelo de referência (*ODP Reference Model*) [ODP 94a] apresenta uma visão consistente dos objectivos de um sistema aberto, introduzindo a definição dos conceitos relevantes e a identificação das funcionalidades necessárias.

Outros modelos de referência, específicos para estruturas de serviços com necessidades particulares, podem ser obtidos a partir do ODP-RM.

O objectivo principal do modelo de referência ODP é permitir que componentes de sistemas distribuídos cooperem, apesar das heterogeneidades reconhecidas no equipamento, nos sistemas operativos, nas redes e na gestão das mesmas, nas linguagens de programação ou nos modelos de bases de dados.

Pretende-se igualmente definir as condições para a existência de interoperabilidade entre aplicações distintas.

4.5.1 Mecanismos de Transparência

Um sistema ODP tem que fornecer os mecanismos que escondam a heterogeneidade dos utilizadores e das aplicações. Estes mecanismos consideram um conjunto de transparências fundamentais, características de um sistema distribuído.

Transparência de acesso

É necessário ocultar as diferenças de representação de dados e dos mecanismos de invocação entre arquitecturas de computação, modelos de programação e protocolos de rede, ou mesmo domínios administrativos ou tecnológicos heterogéneos.

Transparência de localização

Devem ser transparentes as alterações na configuração topológica, no número de instâncias e na localização dos componentes.

Transparência de utilização

Os recursos devem estar sempre disponíveis para o cliente. Devem ser fornecidos mecanismos que ocultem a activação de componentes, as falhas do sistema, a utilização concorrente de serviços e a possibilidade de os objectos estarem agrupados.

Para atingir estes objectivos, o Modelo de Referência ODP propõe a especificação de um conjunto integrado de serviços que possibilite a realização dos componentes infraestruturais básicos (*basic functions*), necessários à implementação de um sistema distribuído com estas características de transparência.

4.5.2 Modelos de Abstracção

O modelo de referência considera também a existência de cinco **perspectivas** (*viewpoints*) consistentes e satisfazendo cada uma um conjunto particular de preocupações, correspondente a um dado nível de abstracção. Cada perspectiva está associada a uma linguagem, que expressa as regras e os conceitos dessa perspectiva.

A especificação do sistema, resultante do conjunto das perspectivas, fornece uma visão estruturante e completa. As perspectivas foram definidas de acordo com os requisitos associados aos serviços e ao ambiente computacional distribuído em que estes são implementados.

A nível mais conceptual, são propostas duas perspectivas relacionadas com os serviços disponibilizados, com preocupações no âmbito do contexto, dos objectivos e da política **empresarial** do serviço fornecido (*enterprise viewpoint*), ou da semântica e do processamento da **informação** no sistema (*information viewpoint*).

É também considerada uma perspectiva puramente **computacional** (*computational viewpoint*), associada à descrição do sistema como um conjunto de componentes de serviço e definindo um modelo de programação abstracto para sistemas distribuídos.

E, finalmente, são definidas duas perspectivas relacionadas com um modelo de **engenharia** das funcionalidades necessárias ao fornecimento do serviço (*engineering viewpoint*) e com a sua implementação **tecnológica** concreta (*technology viewpoint*).

Perspectiva empresarial

A perspectiva empresarial introduz os conceitos de agente (*agent*) e artefacto (*artefact*). Um agente é um objecto cujo comportamento pode alterar relações entre papéis e actividades do sistema. O agente tem iniciativa de execução no sistema e usa os artefactos como recursos para actuar.

Outra noção importante é a de **contrato** (*contract*), que corresponde a um conjunto de regras que constroem dinamicamente a cooperação entre objectos. O **contexto** (*context*) de uma relação entre objectos envolve, em cada momento, as obrigações e garantias associadas ao contrato corrente e o comportamento actual dos objectos envolvidos.

A especificação fornecida pela perspectiva empresarial define os objectivos, o contexto e as condições contratuais de um dado sistema e do ambiente em que se integra (*community*). O comportamento e as acções empreendidas pelos objectos presentes em cada momento no sistema estão sujeitos a essas condições.

Concretamente, uma perspectiva empresarial poderá definir regras de utilização de recursos, de resolução de conflitos, de segurança, de qualidade de serviço, organizativas e de muitos outros domínios. O cumprimento das regras definidas pela perspectiva empresarial confere a um serviço garantias de portabilidade para outros sistemas e de interacção com serviços complementares, caso cumpra as regras definidas nas outras perspectivas.

O seguinte exemplo é paradigmático:

Será correcto supor que dois sistemas de reserva aérea, desenvolvidos na mesma plataforma, com os mesmos protocolos e na mesma linguagem de programação podem interagir ?

Consideremos ainda que as estruturas de informação para reserva e cancelamento de lugares são as mesmas. Contudo, basta que as políticas comerciais relacionadas com o cancelamento sejam diferentes. Uma companhia poderá ter uma política de troca, permitindo ao cliente reservar noutra companhia; a outra companhia poderá assegurar uma política de reembolso.

A tentativa de conexão dos sistemas não exhibe uma política de cancelamento consistente, correspondente a uma única especificação contratual do serviço prestado.

Perspectiva de informação

Do ponto de vista da informação, um dado sistema é representado por uma especificação em termos de estado e estrutura (*schema*) dos objectos de informação, e de regras e relações (*information flow*) entre eles, oferecendo um comportamento previsível. As acções que podem ser invocadas para manipulação de objectos são de três tipos: criação ou destruição de objectos e mudança de estado de um objecto, podendo ser ainda incluídas relações de ordem ou temporais entre acções.

Perspectiva computacional

O modelo computacional descreve o sistema como um conjunto de objectos interoperantes e uma infraestrutura de suporte. A infraestrutura é constituída por um núcleo de funcionalidades, que fornecem uma distribuição transparente, e um conjunto de funções de interesse geral.

A linguagem computacional é definida com base num protocolo de chamadas remotas (RPCs) e num estilo de programação concorrente (*lightweight threads*). Esta escolha é fundamentada pela extensão natural que fornece ao estilo procedimental da maioria das linguagens de programação existentes e porque permite a automatização de optimizações variadas e de teste estático dos programas.

A escolha deste modelo de comunicação permite também abranger mais facilmente os modelos de sistemas distribuídos adoptados nas especificações OSF/DCE e OMG/CORBA.

O conceito mais importante definido pelo modelo computacional é o de **interface comportamental** (*behavioural interface*), que descreve completamente um serviço no contexto do ambiente em que está inserido.

Este interface comportamental é constituído não só pelos conjuntos de **assinaturas** (*signatures*), que descrevem grupos de operações relacionadas e associadas em interfaces, mas

também por conjuntos de sequências de acções, nas quais um objecto pode tomar parte em cada momento e que descrevem as regras de **comportamento** (*behaviour*) do objecto.

O comportamento de um objecto pode ser muito complexo e incluir actividades concorrentes com necessidade de introdução de mecanismos de sincronização.

São ainda atributos do interface comportamental os níveis de **qualidade de serviço** e **transparência** requeridos pelo objecto e uma indicação do **papel** de cliente ou servidor que é atribuído a cada interface.

Desta definição resulta que um objecto do modelo ODP pode ter vários interfaces, que constituem conjuntos de operações relacionadas. Um cliente de um dado objecto utilizará, provavelmente, apenas um conjunto de operações de cada vez, num dado contexto. Isso significa que é conveniente passar identificadores de interface entre objectos, contribuindo para aumentar a coerência e a segurança do modelo de referência.

Outra conclusão que se retira é que interfaces compatíveis devem partilhar um conjunto das mesmas assinaturas. Isto significa que é possível, com base na notação normalizada para definição de interfaces **Interface Definition Notation (IDN)**, gerar automaticamente as rotinas de *stub* necessárias à tradução (*marshalling*) de parâmetros e à comunicação com um objecto que implemente um dado interface.

Contudo, a compatibilidade de assinaturas não assegura, por si só, a compatibilidade do interface a nível comportamental. A compatibilidade de assinaturas pode ser verificada na altura da execução (*run-time*), mas a compatibilidade do interface comportamental deve ser testada durante a fase de especificação.

Dois comportamentos são compatíveis se podem substituir-se um ao outro no ambiente sem que possa ser detectada qualquer diferença.

O comportamento de um objecto é definido no seu **template**. Objectos que forneçam o mesmo serviço podem ser descritos pelo mesmo **template**. O sistema distribuído oferece funcionalidades (*factories*) que permitem a criação de um novo objecto, a partir de um **template**, numa operação denominada **instanciação** (*instantiation*).

Um sistema pode ser construído por composição hierárquica de vários objectos, num processo conhecido por **agregação** (*aggregation*). O **template** de um objecto composto é uma especificação da sua configuração em termos dos objectos constituintes e das suas interconexões.

No modelo computacional são definidos dois tipos de **conexões** (*binding*) possíveis entre objectos computacionais: invocação de operações sobre objectos remotos e fluxos contínuos de informação (*streams*) entre objectos.

A invocação de operações sobre objectos resulta normalmente de uma conexão implícita com um interface, em que o identificador que referencia o interface foi obtido como resultado de uma interacção anterior, ou mediante a utilização das funcionalidades de informação sobre serviços disponíveis no sistema (*trader*).

O **interface operacional** assim obtido consiste em interacções do tipo **anúncio** (*announcement*), ou invocação sem resposta, e **interrogações** (*interrogation*), constituídas por uma invocação seguida de um conjunto finito de respostas (*termination*). Uma interacção com um objecto de conexão é denominada sinalização (*signalling*).

A obtenção de fluxos contínuos de informação entre objectos computacionais é resultado de uma acção de conexão explícita entre interfaces. Na sequência dessa acção, é criado um objecto de controlo da conexão e é devolvido, ao objecto requisitante, o identificador do interface correspondente.

O **interface de controlo** de conexão pode fornecer, além das funcionalidades básicas relacionadas com a manutenção da conexão, facilidades de controlo dinâmico de conexões múltiplas, controlo da qualidade de serviço, notificação de erros e de eventos de interesse para os objectos.

Os **interfaces de stream** conectados descrevem fluxos de informação contínuos unidireccionais entre objectos, como por exemplo sinais de audio ou vídeo de uma aplicação multimédia. O encapsulamento dos vários fluxos de informação existentes, bem como as regras de configuração (*half-duplex*, *full-duplex*, *multicast*, ...) e o controlo desses fluxos são assegurados pelo objecto de controlo correspondente.

As regras de identificação de subtipos, associadas aos interfaces de *stream*, diferem das correspondentes aos interfaces operacionais, no sentido em que permitem comunicação entre objectos cujos interfaces oferecem diferentes níveis de funcionalidade.

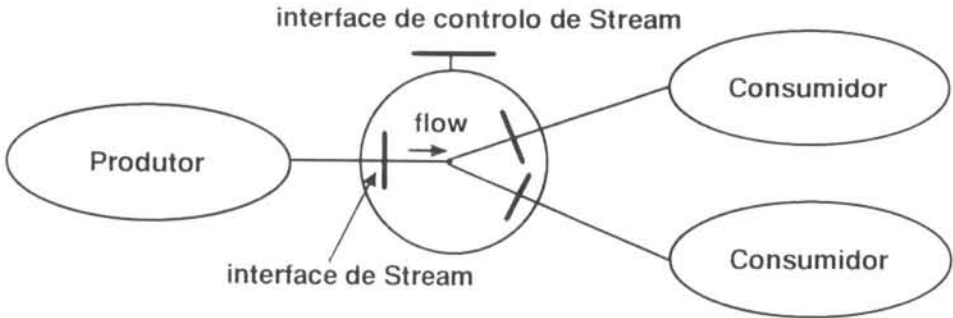


Figura 4.8: Representação de uma conexão explícita do tipo *stream*.

Em geral, as regras podem ser descritas em dois passos consecutivos. Primeiro é necessário efectuar a identificação das correspondências entre os vários fluxos definidos nos dois tipos, para detectar se é de algum modo possível a interconexão entre interfaces. Seguidamente, procede-se à comparação a nível dos aspectos de qualidade de serviço para determinar se uma relação de subtipos existe.

Um interface de audio poderá ser considerado como um subtipo de um interface de audio e vídeo, permitindo conectar um serviço telefónico com um videotelefone.

Perspectiva de engenharia

Um objecto definido no modelo computacional é mapeado num conjunto de objectos funcionais básicos, *Basic Engineering Objects (BEO)*.

A tradução de *templates* computacionais para funcionalidades definidas neste modelo corresponde à noção de compilação de um programa.

Os objectos básicos que constituem a implementação de um interface definem uma unidade de **agrupamento** (*cluster*), gerida por um objecto específico (*cluster manager*). Este conceito corresponde, nas linguagens de programação, à noção de *linking* de módulos de código para gerar um programa executável.

Um objecto computacional corresponde a uma ou mais implementações de interfaces agrupados numa **cápsula** (*capsule*), que pode ser mapeada num processo ou no espaço de endereçamento do sistema operativo, gerida por um objecto (*capsule manger*).

Uma máquina pertencente ao domínio do sistema distribuído corresponde a um **nó** (*node*), no qual um objecto, que constitui o **núcleo** (*nucleus*), coordena o processamento, o armazenamento ou a comunicação de uma ou mais cápsulas residentes nesse nó

Os *clusters* dialogam entre si utilizando as funcionalidades de objectos que fornecem a transparência necessária à comunicação, constituindo um **canal** (*channel*) de comunicação que conecta os interfaces dos objectos entre os quais a interacção ocorre.

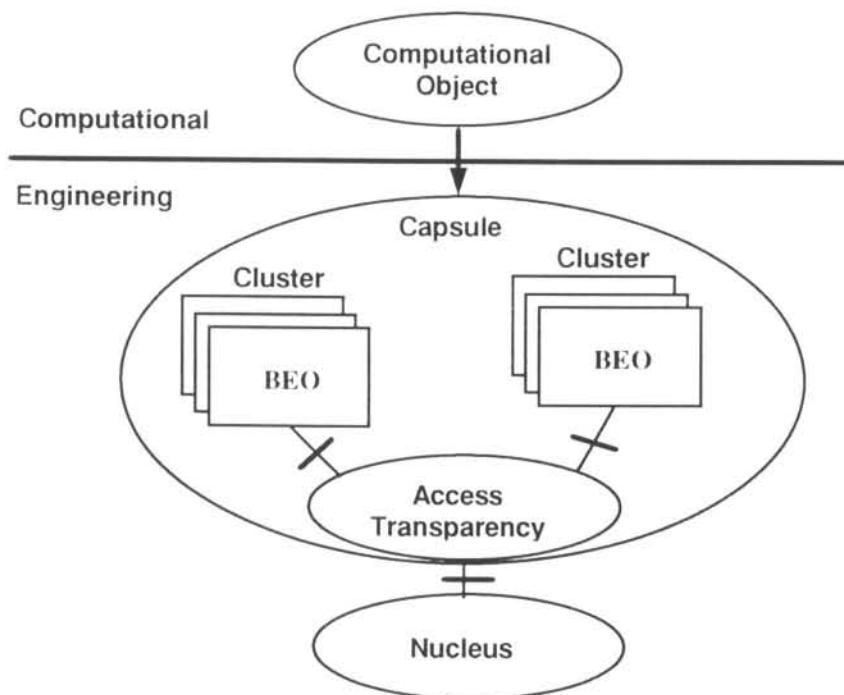


Figura 4.9: Tradução do modelo computacional para o modelo de engenharia.

Um canal é uma conexão entre objectos do nível de engenharia, assegurada por uma configuração de vários objectos funcionais: *stubs*, *binders*, *interceptors* e *protocols*.

Os **Stubs** asseguram a conversão dos parâmetros dos interfaces para formatos de comunicação de dados e recorrem aos objectos **Binders** para estabelecer e manter a conexão ponto-a-ponto. **Interceptors** são objectos que podem ser necessários se forem ultrapassadas fronteiras entre domínios administrativos ou tecnológicos e daí decorrer a necessidade de operações de tradução e verificação sobre os dados. Os **Protocols** são objectos que fornecem funcionalidades de comunicação com outros objectos de protocolo, noutros clusters, cápsulas ou mesmo nós.

Os objectos *stub* e *binder* possuem um interface de controlo que fornece funcionalidades de gestão da qualidade de serviço e de alterações na configuração (*stub*) e no funcionamento do canal (*binder*).

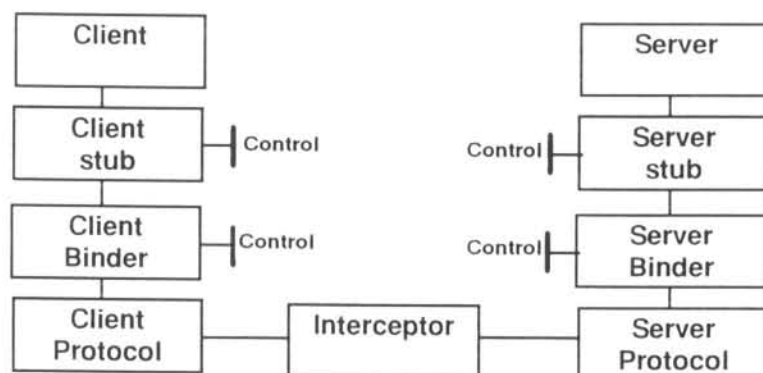


Figura 4.10: Exemplo de um canal entre um cliente e um servidor.

Perspectiva tecnológica

A perspectiva tecnológica oferece ao cliente do serviço dois níveis conceptuais diferentes, sendo um relacionado com o interface de programação que fornece e o outro relativo às funcionalidades suportadas pelo sistema distribuído e destinadas a oferecer vários tipos de transparência.

O **interface de programação**, directamente acessível aos clientes de um serviço, oferecido por implementações do modelo ODP considera de modo integrado as perspectivas empresarial, informacional e computacional [API 93].

Quanto às funções básicas requeridas para a criação de um sistema distribuído **transparente** são baseadas sobretudo nas descrições computacional e de engenharia. No ODP são definidos vários grupos de **funções básicas**, existindo também relações de dependência entre algumas funções:

- a **gestão** dos objectos definidos no modelo de engenharia constitui o grupo fundamental que assegura o funcionamento transparente do sistema distribuído,
- as funções de **coordenação** disponibilizam funcionalidades de transparência relacionadas com os problemas da distribuição dos serviços em rede,

- as funções de **armazenamento** consideram a problemática do armazenamento e da procura da informação,
- o grupo de funções de **segurança** fornece mecanismos de protecção no acesso e da integridade da informação.

Existem algumas implementações utilizando a tecnologia definida no modelo aberto de computação distribuída (ODP). A arquitectura para processamento distribuído ANSAware [ANSA 93], por exemplo, fornece um ambiente de programação com uma linguagem de especificação de interfaces (IDL), e serviços que suportam mecanismos de transparência de localização (*Trader*), de acesso (PREPC) e de utilização. Este ambiente é implementado com base nas normas ANSI-C e POSIX, possibilitando a portabilidade para a maior parte das plataformas.

Capítulo 5

Componentes de Serviços Multimédia

“Os objectos reutilizáveis são como os poemas - é fácil falar acerca deles, mas é difícil escrever um realmente bom.”

O.Nierstras et al. Em *Component-Oriented Software Development*, 1992

O desenvolvimento de aplicações multimédia em ambientes distribuídos assenta na normalização dos serviços de suporte, que asseguram a transparência da comunicação entre os vários componentes de uma aplicação, e na especificação dos procedimentos e métodos de representação necessários para o processamento e a apresentação da informação multimédia.

Neste capítulo pretende-se integrar as ideias desenvolvidas anteriormente, relativas à representação de informação multimédia e aos sistemas de computação distribuída, no modelo de processamento e apresentação de objectos multimédia, proposto pela ISO. Seguidamente, são também analisadas as diferenças conceptuais e as funcionalidades existentes, numa plataforma concreta de grande divulgação (o ambiente *Windows*), para o desenvolvimento de componentes de serviços multimédia distribuídos.

5.1. Processamento e Apresentação de Objectos Multimédia

No enquadramento geral da área multimédia, definido pelos esforços de normalização da ISO, só muito recentemente foi considerada a necessidade da especificação de um conjunto de metodologias, envolvendo aspectos relacionados com a construção, apresentação e interacção com objectos multimédia em sistemas distribuídos, dando origem ao ambiente integrado denominado *Presentation Environment for Multimedia Objects* [PREMO 94a].

Esta norma pretende definir um modelo estruturante das várias tecnologias e normas de representação e apresentação da informação actualmente existentes, à semelhança do modelo geral definido pelo ODP para os sistemas distribuídos. O objectivo principal consiste na especificação de um ambiente de programação flexível e modular, que ofereça facilidades de configuração, extensibilidade e portabilidade entre plataformas.

Além da definição de técnicas de apresentação e interacção, a norma exhibe também preocupações da área da computação gráfica, da qual é originária. Uma ideia chave do modelo reside na integração entre os vários média, por oposição à simples coexistência.

O ambiente definido pelo PREMO deve permitir mecanismos avançados de sincronização, necessários em áreas de aplicação diversas como: a realidade virtual, a visualização científica ou os sistemas de informação geográfica. Um exemplo típico, de técnicas avançadas de processamento e sincronização, consiste na possibilidade de integrar objectos multimédia com aspectos geométricos numa apresentação, na qual as propriedades acústicas dependem do ponto de vista do utilizador sobre a cena que está a visualizar.

A construção de serviços multimédia recorre ao **modelo de objectos** definido pelo OMG [OMG 90], considerando que as tecnologias orientadas para objectos fornecem um encapsulamento consistente para a distribuição dos serviços. Outra característica importante consiste num **modelo de eventos**, que têm um papel importante nos mecanismos de composição e sincronização de objectos multimédia. Os eventos são gerados espontaneamente pelos objectos e consumidos por outros objectos que manifestem interesse em os receber.

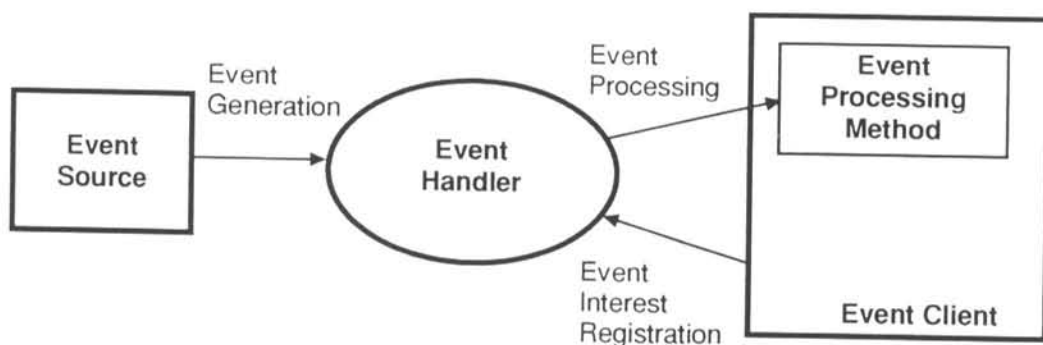


Figura 5.1: Modelo de eventos do PREMO.

Os objectos PREMO são portanto **ativos**, o que permite considerar os aspectos de sincronização segundo as metodologias bem conhecidas da concorrência entre processos. Outra particularidade consiste nas operações invocadas sobre os objectos, que podem

obedecer aos modelos síncrono ou assíncrono convencionais, ou ser **amostras** (*samples*) de notificações geradas em grande número pelo cliente, e que se vão sobrepondo no interface do servidor. Este conceito permite manipular facilmente a informação gerada por um periférico de entrada de interacção contínua, como por exemplo um “rato”.

5.1.1 Hierarquia de Componentes

O PREMO define um componente de **Serviços Básicos** (*Foundation Component*) que suporta todas as funcionalidades do modelo já referidas. Outros componentes de serviços multimédia mais específicos para diferentes famílias de aplicações, podem ser derivados hierarquicamente das classes descritas no componente básico, possibilitando a extensão e evolução da norma. Várias normas internacionais e “de-facto” existentes actualmente podem assim ser englobadas numa infraestrutura genérica. Duas áreas importantes foram já consideradas para ser especificadas por componentes abstractos, estando previstos outros componentes, por exemplo na área do processamento de imagem.

Um componente de **Modelização, Apresentação e Interacção** constitui a base de componentes de serviços da área da computação gráfica. Um engenho de apresentação MHEG pode também ser implementado com base nos serviços definidos neste componente.

Um componente de **Sistemas e Serviços Multimédia** fornece uma infraestrutura para o desenvolvimento de aplicações multimédia interactivas em sistemas distribuídos, e pretende encapsular os resultados obtidos pela Interactive Multimedia Association [IMA 92].

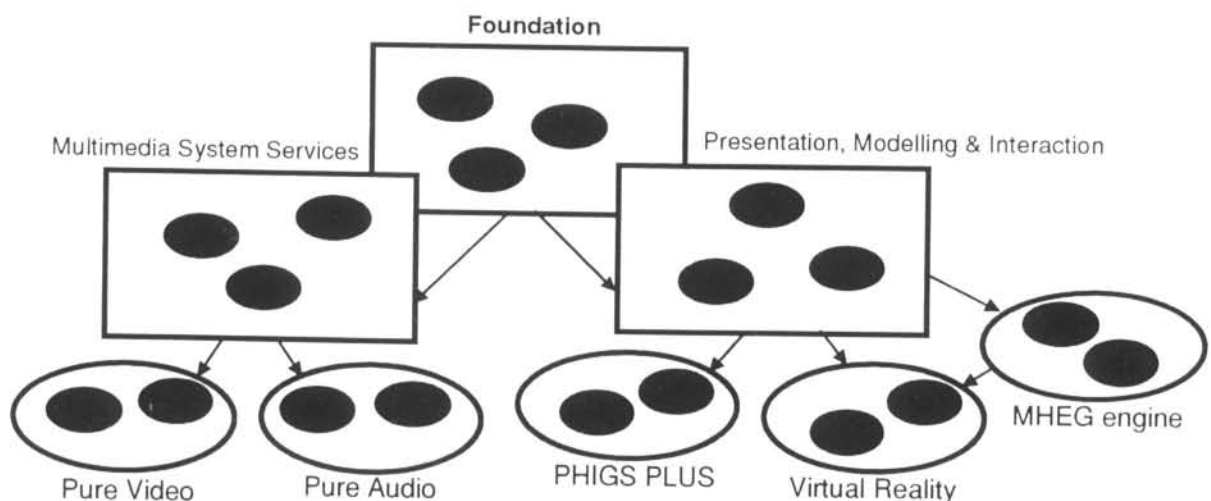


Figura 5.2: Hierarquia de componentes do PREMO.

A norma PREMIO pretende definir a especificação de modelos para diferentes famílias de aplicações multimédia, que constituam uma infraestrutura adequada para a implementação das tecnologias de processamento e apresentação necessárias. Estes objectivos são substancialmente diferentes dos de outros esforços de normalização da ISO nesta área, mais direccionados para a estruturação de documentos e a representação da informação multimédia.

A norma **MHEG** especifica uma representação da informação multimédia e hipermédia num formato final, não reversível e adequado para a apresentação. O MHEG pode portanto ser considerado o formato ideal de codificação da informação gerada num ambiente PREMIO e destinada à apresentação e interacção com o utilizador.

Tanto o **HyTime** como o **HyperODA** representam normas para a estruturação de documentos multimédia. O HyTime apresenta um modelo temporal muito completo mas que é possível mapear no mecanismo de sincronização com base em eventos definido pelo PREMIO. É possível criar aplicações utilizando as funcionalidades do PREMIO que permitam produzir ou interpretar documentos HyTime e HyperODA.

5.1.2 Componente de Serviços Básicos

O componente de serviços básicos [PREMIO 94b] descreve os interfaces e os atributos de um conjunto mínimo de objectos, que fornecem as funcionalidades necessárias para suportar o processamento e a apresentação da informação em plataformas distribuídas heterogéneas.

Estes objectos constituem os tipos fundamentais nos quais podem ser decompostas funcionalidades complexas, descritas por componentes derivados do componente de serviços básicos. A descrição dos interfaces dos objectos definidos neste componente é completamente especificada utilizando a *Interface Definiton Language* (IDL) definida na arquitectura distribuída adoptada pelo OMG [CORBA 91].

Os tipos de dados que podem ser trocados entre os objectos são os mesmos que são utilizados na arquitectura CORBA, com algumas excepções: foram criados tipos específicos para os portos de entrada e saída (`InPort` e `OutPort`), e para suportar o modelo de eventos (`Event`); o atributo opcional **oneway** existente numa operação foi substituído pelos atributos **async** e **sampld** e as expressões contextuais não são permitidas.

A norma define ainda um conjunto de excepções de sistema. Estas excepções retornam também o estado em que se encontrava a operação quando foram disparadas

(`CompletionStatus`), indicando que o processamento foi concluído, nunca foi invocado ou o resultado é indefinido.

Os objectos que fornecem as funcionalidades do componente de serviços básicos formam uma hierarquia de classes, que se descreve a seguir.

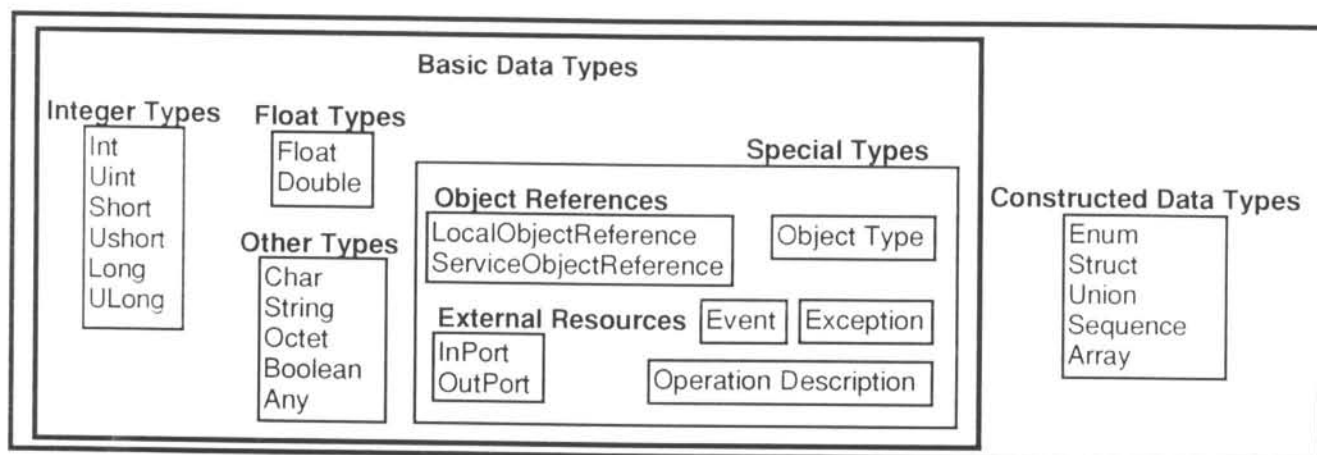


Figura 5.3: Tipos definidos no PREMO.

Gestão de Objectos

Os serviços de **gestão de objectos** (*life cycle manager objects*) permitem criar e destruir instâncias de objectos do tipo `PREMOObject` ou fazer a gestão das referências a esses objectos. São fornecidos dois serviços diferenciados para gestão de instâncias dos objectos destinadas a ser acedidas por vários componentes (`ServiceLifeCycleManager`) e para objectos de uso exclusivo da componente em que foram instanciados (`LocalLifeCycleManager`).

Comportamento Fundamental de um Objecto

`PREMOObject` é um tipo abstracto que define os comportamentos fundamentais (*fundamental object behaviour*) de qualquer objecto do modelo. Permite especificar comportamentos de inicialização e destruição do objecto e mesmo dos tipos de que deriva, e também obter informações sobre a hierarquia de tipos de qualquer objecto.

Objectos de Dados

`EnhancedData` é um tipo de dados abstracto do qual podem ser derivados objectos de dados com uma semântica básica que consiste na inicialização e em operações de consulta e modificação do valor definido para o objecto.

Agregação de Objectos

Os objectos podem ser agregados em colecções utilizando objectos derivados do tipo `Aggregate`. As especializações deste tipo fornecem operações para adição, remoção, procura e invocação em modo *broadcast* de elementos que consistem em referências para outros objectos.

Os subtipos `List`, `Table`, `IndexedArray` e `AssociativeArray`, estão definidos, e suportam agregações comuns, as quais podem ainda ser estruturadas hierarquicamente para construir árvores, grafos, etc.

Gestor de Eventos

O gestor de eventos `EventHandler` permite que objectos registem o seu interesse num determinado evento e que geradores de eventos os notifiquem da sua ocorrência. O modelo de eventos é baseado num tipo de dados específico `Event`. Os eventos são descritos por uma assinatura constituída pelo nome do evento e dados associados na forma de pares do tipo (chave, valor).

Um objecto regista o seu interesse num evento fornecendo ao gestor de eventos uma referência para uma operação do seu interface destinada a receber a notificação. O objecto pode ainda fornecer uma lista de **restrições** que têm que ser cumpridas para o evento ser comunicado. Cada restrição consiste numa operação de comparação e num par (chave, valor).

Quando um evento é gerado as restrições são comparadas com os dados associados com a chave correspondente. A condição de disparo da notificação é resultado da conjunção ou disjunção das várias comparações, conforme indicado pelo objecto no registo.

Invocação Dinâmica

O objecto `DynamicCaller` permite invocar uma operação num dado objecto a partir de uma referência a esse objecto, da descrição (`OperationDescription`) e dos

argumentos da operação. Esta funcionalidade é baseada directamente no interface `DynamicInvocation` descrito na arquitectura CORBA. Um exemplo típico de utilização consiste no registo de interesse, perante o gestor de eventos, na notificação de um determinado evento.

Controladores

Os objectos do tipo `Controller` encapsulam mecanismos de coordenação e sincronização complexos numa máquina de estados finitos (*Finite State Machine*). Um objecto controlador regista os eventos de interesse junto do gestor de eventos. A notificação de um evento invoca a função assíncrona `HandleEvent` e pode causar uma alteração no estado do controlador. O estado do controlador pode ser consultado (`getState`) em qualquer momento, ou levado ao valor inicial (`reset`).

Um controlador pode servir para definir mecanismos de sincronização condicional ou temporal, ou, por exemplo, para coordenar interacções complexas como operações *de drag-and-drop*. A figura seguinte apresenta um exemplo de sincronização temporal e condicional, em que *frames* de imagem e legendas de texto são apresentadas simultaneamente e a um débito pré-definido específico.

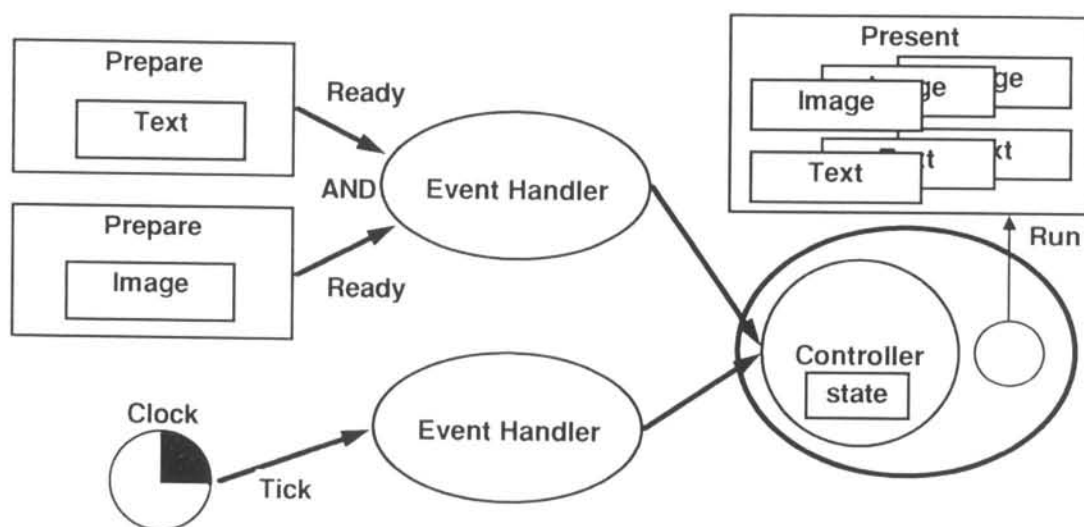


Figura 5.4: Exemplo de sincronização controlada por um `Controller`.

Conexões

As conexões são tipos abstractos que permitem derivar objectos com as funcionalidades básicas necessárias para suportar fluxos de dados de entrada (`InConnection`) ou saída (`OutConnection`), em modo síncrono (*Control Driven*) e assíncrono (*Data Driven*).

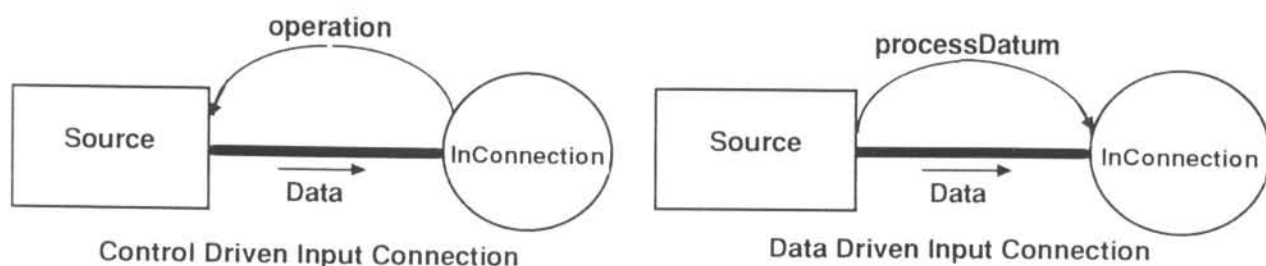


Figura 5.5: Tipos de conexões de entrada.

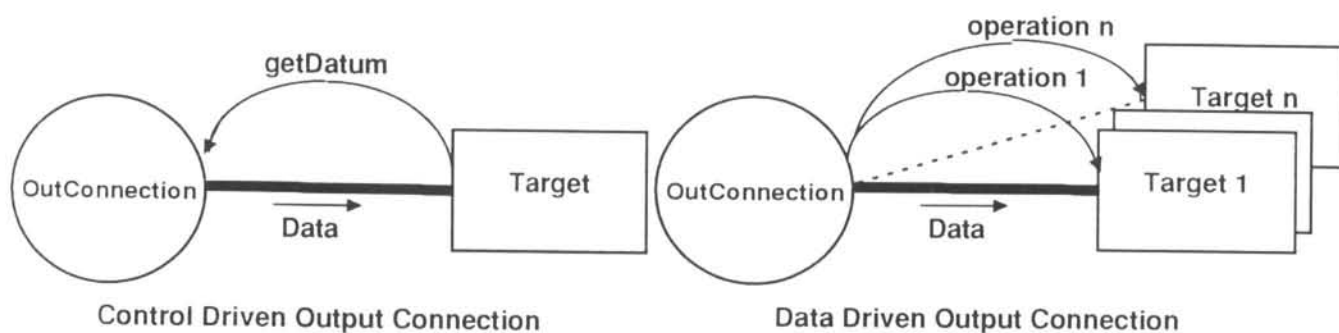


Figura 5.6: Tipos de conexões de saída.

Produtor

O tipo `Producer` processa informação obtida numa conexão do tipo `InConnection` e publica o resultado através de uma conexão `OutConnection`. O funcionamento em modo assíncrono ou síncrono, de cada uma das conexões, determina o comportamento do objecto e em particular a necessidade de armazenamento local. O processamento é controlado por acções de início e fim de processamento definidas no tipo abstracto `Process`.

Portas

As portas permitem definir conexões a sistemas exteriores ao ambiente PREMO, mapeando o comportamento desses sistemas em dispositivos virtuais (*virtual devices*). São baseadas nos tipos abstractos `InputPort` e `OutputPort`, que permitem associar e remover portos de ligação com o exterior e também registar um gestor de eventos para aceitar eventos do exterior.

Os objectos portas `InputPorter` e `OutputPorter` são derivadas de tipos abstractos de porto, conexão e processo. O modo de funcionamento da conexão determina o comportamento da porta definido na acção de início de processamento. A composição de

portas e produtores permite definir as relações entre fluxos de dados de uma aplicação, à semelhança do modelo de *Sources*, *Filters* e *Sinks* apresentado no capítulo 2 [Gibbs 91].

A figura seguinte mostra um exemplo de uma porta de entrada assíncrona. O registo de armazenamento é reescrito cada vez que chegam dados e o evento correspondente é enviado ao gestor. O objecto que controla a porta recebe notificações de eventos por amostras (*sample*), sendo deste modo informado da existência de dados para processar.

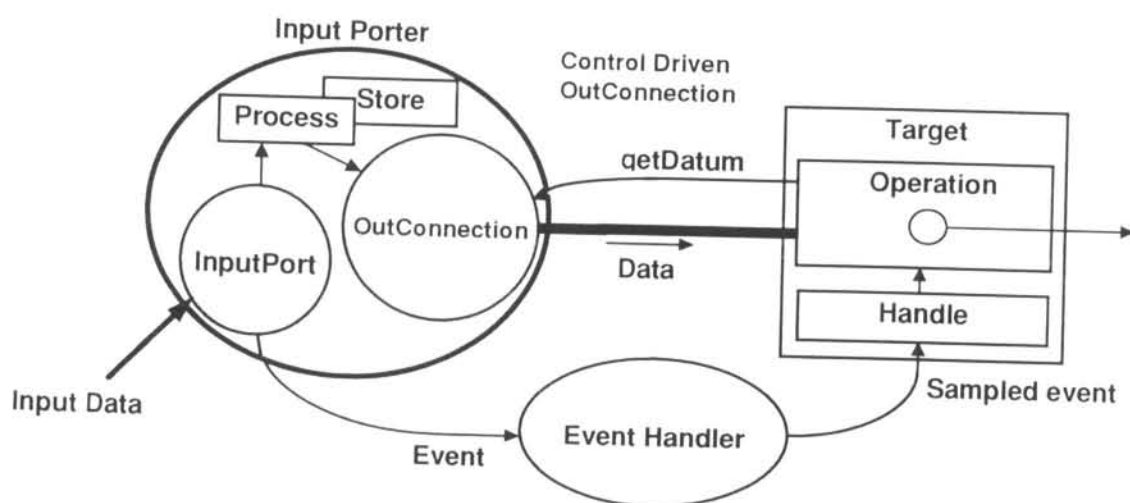


Figura 5.7: Exemplo de funcionamento de uma porta de entrada assíncrona com reescrita.

5.2 Modelo de Componentes do *Windows*

A computação distribuída no ambiente *Windows* é actualmente apenas *vapourware*, porque, em termos práticos, não ultrapassa as fronteiras de uma única máquina. Contudo, este é o sistema operativo mais divulgado em todo o mundo e para o qual maior número de aplicações, nas mais variadas áreas, são produzidas todos os anos. A explosão do mercado multimédia fez aumentar o número de equipamentos instalados com estas capacidades, enquanto que as redes locais e o número de acessos à *Internet* se multiplicam. Se acrescentarmos a estes dados alguma garantia de continuidade na evolução futura do sistema, chegamos à conclusão da inevitabilidade da sua continuação como líder do mercado nos tempos mais próximos.

É quase consensual uma visão crítica do ambiente *Windows*, baseada em facilidades de programação confusas, oferecidas por um crescendo de APIs, e em serviços de rede básicos e completamente anárquicos. Mas, lentamente, e um pouco à margem das arquitecturas

propostas pelas organizações de normalização, o *Windows* tem evoluído para um sistema operativo orientado para objectos, com facilidades de processamento distribuído. As versões NT3.5 e *Windows95* suportam mesmo uma versão “local” de RPCs, presumivelmente compatível com o OSF/DCE. Serviços desenvolvidos nestas plataformas poderão funcionar “sem alterações”, e de modo completamente distribuído, no futuro sistema operativo orientado para objectos da Microsoft (Cairo), que irá oferecer serviços de ficheiros distribuídos, segurança e outras facilidades necessárias à criação de componentes de software.

A tecnologia *Object Linking & Embedding* [OLE 93], que está na base do ambiente de programação por componentes do *Windows*, apresenta semelhanças com a arquitectura definida no CORBA e a interoperabilidade dos dois sistemas foi mesmo já demonstrada, pela implementação de uma *gateway* entre o OLE2 e o ObjectBroker, a versão do CORBA da Digital Equipment Corporation.

É a partir destas considerações, e porque o modelo de sistemas distribuídos proposto apresenta algumas diferenças, que se descreve o conjunto de tecnologias que compõem o OLE2 e se analisa com algum pormenor, o modelo de objectos em que é baseado.

5.2.1 A Tecnologia OLE2

A tecnologia OLE2 fornece um conjunto básico de interfaces normalizados, com os quais é possível obter a integração de um grande número de aplicações. Estes interfaces são suportados por uma infraestrutura orientada para objectos, que oferece serviços de conexão e comunicação entre os vários componentes.

Grande parte dos serviços oferecidos nestes interfaces devem ser entendidos numa perspectiva de funcionamento das aplicações centrada no documento (*Document-Centric*). Segundo este conceito, as aplicações são cooperantes na criação e apresentação de **documentos compostos** (*Compound Document*), que contêm informação heterogénea e manipulada em exclusivo pela aplicação correspondente.

Os documentos compostos podem englobar (*Embedding*) efectivamente a informação, ou definir ligações (*Linking*), que apontam a localização de cada elemento no sistema de ficheiros, ou dentro de um documento composto.

A criação e gestão de documentos compostos, envolvendo múltiplas aplicações, coloca problemas de eficiência no acesso e na transferência de informação entre aplicações. A

tecnologia OLE2 introduz um sistema de **armazenamento estruturado** (*Structured Storage*), que suporta algumas funcionalidades importantes, como a actualização parcial de informação (*Incremental Save*), as operações de transacção (*Transaction*) e o controlo dos direitos de acesso à informação.

Os documentos compostos são armazenados em ficheiros estruturados (*Compound Files*), oferecendo um acesso normalizado à informação que contêm. A definição de blocos de informação comuns, contendo a descrição do conteúdo do documento, permite que qualquer aplicação, incluindo o sistema operativo, efectue operações de consulta e interrogação sobre o documento.

As ideias básicas expressas nesta perspectiva apresentam grandes semelhanças com a descrição de documentos em SGML, proposta por normas destinadas à automatização de ambientes de escritório [ODA 89]. As aplicações que melhor se enquadram neste modelo são, fundamentalmente, destinadas à edição e revisão de documentos compostos por vários monomédia. A tecnologia de ficheiros estruturados não oferece representações normalizadas de informação multimédia e hipermédia, que permitam, por exemplo, definir relações temporais ou interactividade com o utilizador.

Com base nas tecnologias para a manipulação de documentos estruturados, o OLE2 suporta a integração de aplicações e de serviços segundo duas vertentes diferentes: a transferência de dados entre aplicações e a edição local de informação utilizando serviços de uma aplicação remota.

A **transferência de dados** entre aplicações é convenientemente encapsulada num objecto, que oferece um interface único (*Uniform Data Transfer*), independentemente do método de transferência utilizado. Esta tecnologia é suportada por ficheiros estruturados, que fornecem informação de descrição dos dados que contêm, permitindo seleccionar, entre a transferência por memória partilhada ou simplesmente por apontadores para ficheiros, a operação mais adequada a cada tipo de dados. Além disso, o OLE2 define um conjunto de funções que permitem implementar facilmente os mecanismos usuais de *drag-drop* e *clipboard* para transferência de objectos de dados.

A **edição local de informação** consiste em permitir que várias aplicações cooperem no acesso ao interface de interacção com o utilizador. Deste modo é possível, numa janela sob controlo de uma aplicação, aceder a operações disponibilizadas por outra aplicação (*In-Place Activation*). Esta funcionalidade é importante para aplicações de autoria de documentos

compostos, porque integra operações de edição, dependentes em exclusivo de cada tipo de informação, num interface de utilizador único.

Utilizando as tecnologias de transferência de dados e de edição local de informação, é possível implementar aplicações que consistem basicamente num contentor (*Container*) de informação, sendo toda a manipulação da informação efectuada a partir de servidores remotos.

O OLE2 fornece ainda mecanismos para que linguagens de programação de *script*, como o *VisualBasic*, possam ser utilizadas para desenvolver hiper-aplicações, que **automatizam** (*Automation*) as tarefas executadas por outras aplicações, e permite assim maior flexibilidade na implementação de soluções específicas com recurso a componentes.

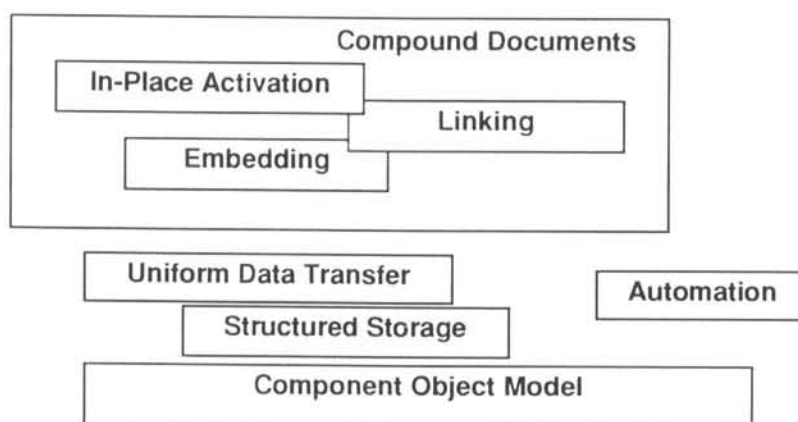


Figura 5.8: Tecnologias fornecidas pelos componentes principais do OLE2.

Um objecto *Automation* descreve funcionalidades de alto nível, com sentido para o utilizador e que constituem geralmente a estrutura de mensagens principal de uma aplicação. Esta tecnologia disponibiliza mecanismos que permitem consultar os interfaces de programação das aplicações que estão disponíveis (*Type Library*) e invocar dinamicamente as funcionalidades requeridas [Webb 93].

5.2.2 O Modelo de Objectos COM

O modelo de objectos *Component Object Model* (COM) fornece a infraestrutura necessária para a conexão e comunicação entre componentes em sistemas distribuídos. Esta tecnologia apresenta grandes semelhanças com os modelos de computação distribuída referidos no capítulo 4. Os serviços de suporte definidos na infraestrutura oferecem os

mecanismos de transparência necessários para uma utilização distribuída e facilidades para a invocação dinâmica e a activação de componentes.

Os interfaces são definidos utilizando uma linguagem semelhante ao IDL, com a particularidade de apontadores para interfaces também serem transferidos, com base num processo de *marshalling* proprietário. O mecanismo de chamadas a procedimentos remotos é compatível com a norma OSF/DCE e o modelo suporta a implementação de interfaces operacionais e interfaces de *stream*.

O modelo COM apresenta no conceito de objecto a sua principal originalidade. Um objecto *Windows* pode implementar um ou mais interfaces independentes, sendo um **interface** definido como um conjunto de funções relacionadas semanticamente. Se um objecto implementa uma das funções de um interface, então tem necessariamente de as implementar todas.

Em termos de implementação, um interface é uma norma binária de interacção entre componentes, no mesmo sentido em que existem várias convenções de declaração (*cdecl*, *pascal*,...) para a invocação de procedimentos. A representação de execução de um interface consiste numa estrutura de dados, contendo o estado correspondente do objecto e um apontador para uma tabela de apontadores de funções. As funções apontadas nessa tabela são os métodos que implementam as funcionalidades definidas no interface. Como convenção, o primeiro parâmetro passado a cada função é a referência ao objecto detida pelo cliente, o que possibilita ao objecto aceder ao seu próprio estado.

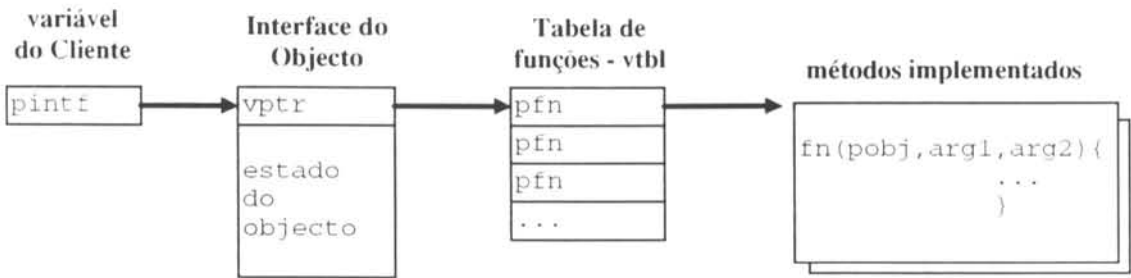


Figura 5.9: Representação de execução de uma instância de um interface.

O que caracteriza um objecto *Windows* é a herança de um interface comum **IUnknown**, que fornece um conjunto de funcionalidades mínimas, essenciais para o comportamento de qualquer objecto deste modelo. As características oferecidas pelo interface comum são: um mecanismo que permite a um objecto suportar **múltiplos interfaces** e um mecanismo de **contagem de referências** que possibilita fazer a gestão em memória de uma

instância do objecto. As funções definidas no interface IUnknown que suportam estas funcionalidades estão descritas na figura 5.10.

```
Interface IUnknown {
    virtual HRESULT QueryInterface(IID& iid, void** ppvObj)
    virtual ULONG AddRef()
    virtual ULONG Release()
};
```

Figura 5.10: Definição do Interface IUnknown.

Quando um cliente obtém um apontador para um objecto, esse apontador corresponde a um interface do objecto. Recorrendo ao mecanismo definido pela função **QueryInterface** é possível utilizar o identificador de um interface para obter o apontador para outro interface do mesmo objecto. O acesso aos interfaces, a partir de um mecanismo comum, permite conhecer todos os interfaces que um objecto suporta dinamicamente, ainda que estes sejam alterados, sem necessidade de recorrer a funções exteriores ao próprio objecto.

A agregação de vários objectos num único componente consiste basicamente na implementação de um interface IUnknown com funcionalidades de negociação e gestão de todos os interfaces expostos pelo serviço.

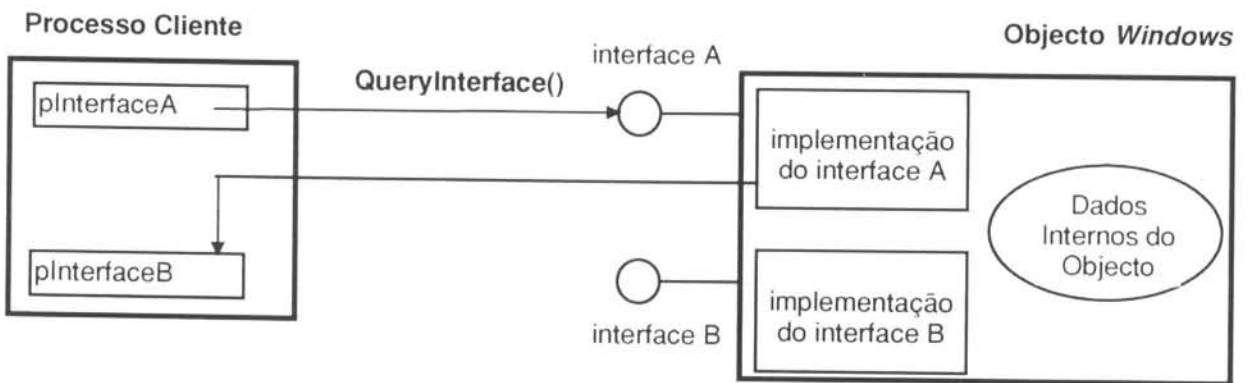


Figura 5.11: Funcionamento da operação `QueryInterface()`.

Os objectos acedidos pelos interfaces utilizam um mecanismo de contagem de referências, de modo a assegurar que a duração do objecto está de acordo com as referências que lhe são feitas. Este mecanismo foi adoptado para permitir que componentes independentes

obtenham e libertem o acesso a um objecto, sem ter que se coordenar para a gestão da sua existência.

O funcionamento correcto da gestão das referências a um objecto é um esforço cooperativo, que depende do bom comportamento de todos os clientes do objecto. Por vezes são possíveis simplificações na invocação das funções de contagem `Add()` e `Release()`, quando o mesmo código manipula várias referências, técnicas que estão exemplificadas na especificação do OLE2 [OLE 93].

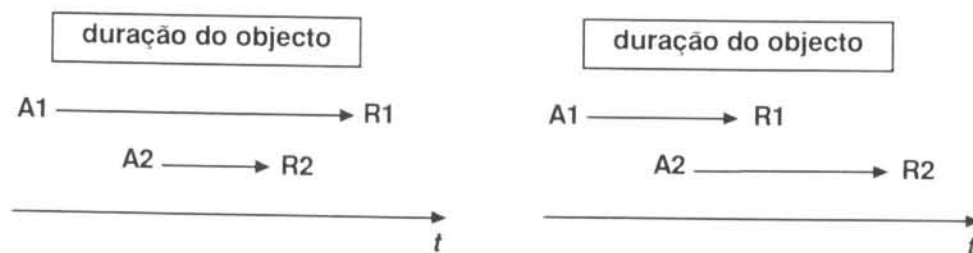


Figura 5.12: Gestão de referências a um objecto.

As funcionalidades definidas pelo interface `IUnknown` devem estar disponíveis em qualquer interface, razão pela qual todos os interfaces de um objecto *Windows* são dele derivados. Exceptuando este caso particular de herança, a extensão das funcionalidades de um objecto é geralmente implementada recorrendo à definição de novos interfaces, completamente independentes. Esta filosofia permite criar interfaces mais pequenos e em menor número para suportar as mesmas funcionalidades.

Em sistemas distribuídos, isto significa que a disponibilidade de milhares de componentes servidores não corresponde necessariamente à implementação de milhares de *stubs* (*proxies*) no lado do cliente. Como cada interface engloba a semântica de interacção necessária para efectuar um novo contrato com um serviço, a escala obtida na criação de conexões não é linear.

A implementação de componentes baseados no modelo COM é facilitada na linguagem de programação orientada para objectos *Visual C++*. Esta linguagem suporta uma biblioteca de classes, **Microsoft Foundation Classes** [MFC 93a], que permite a implementação de objectos *Automation* [MFC 93c] e de interfaces proprietárias de objectos *Windows* [MFC 93b], derivando a classe respectiva da classe MFC/OLE `CCmdTarget`.

Capítulo 6

Implementação de um Engenho MHEG

“Embora se diga frequentemente que um camelo é a ideia que os comités de normalização fazem de um cavalo, há pessoas que precisam de camelos em situações em que os cavalos não funcionam.”

S.Newcomb et.al. “The HyTime”, 1991

A norma MHEG não define a estrutura a que um engenho de interpretação deve obedecer, nem como este manipula os objectos que estão sob o seu controlo. Contudo, para uma correcta interpretação e apresentação dos objectos, um engenho MHEG tem que obedecer a alguns **requisitos**:

- o engenho deve ser capaz de avaliar os **estados** e o valor dos **atributos** dos objectos que estão sob o seu controlo. As alterações que ocorrem num estado, ou no valor de um atributo, podem ser usadas para exprimir relações de sincronização entre objectos, utilizando *links*.
- o engenho utiliza os serviços de um interface de **apresentação** e de **interacção** com o utilizador. As interacções com o utilizador são formalizadas no engenho em alterações correspondentes de estados de selecção ou modificação de objectos de execução. A progressão temporal de uma apresentação pode também gerar alterações de estado num objecto de execução, se neste tiverem sido definidas marcas temporais (*timestones*).
- o engenho utiliza os serviços de **comunicação** para aceder a objectos referenciados externamente. Quando um objecto está localizado numa referência externa ao engenho é preparado, pode ser conveniente iniciar a transferência deste para memória antes de ser necessário. Objectos com requisitos de transferência e apresentação em tempo real, como *streams* de audio e vídeo, são normalmente descritos em objectos de dados multiplexados, cuja

preparação poderá consistir no estabelecimento de uma ligação comutada entre a fonte de informação e o interface de apresentação.

- um controlo mais complexo do comportamento de uma apresentação pode ser fornecido por uma aplicação ou por *scripts* exteriores, que comunicam com o engenho através de um interface. O interface com o engenho MHEG deve fornecer um conjunto de funcionalidades mínimas que permitam à aplicação efectuar todas as acções elementares e aceitar os resultados de uma acção *return*. Poderão também ser implementadas outras facilidades, como por exemplo a criação de objectos por aplicações de autoria e a gestão de erros de funcionamento do engenho.

Este modelo assume que o engenho MHEG é responsável apenas pelo processamento de objectos MHEG e pela interacção com a aplicação. Outros serviços eventualmente necessários, como o acesso a bases de dados, comunicações ou a apresentação gráfica, são fornecidos pela aplicação e por outros módulos. A figura 6.1 ilustra esta arquitectura de um sistema MHEG.

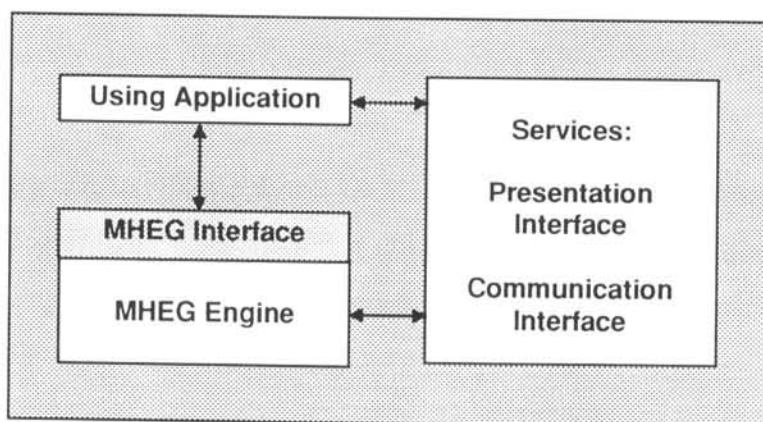


Figura 6.1:Exemplo de um sistema MHEG.

6.1 Funcionalidades de um Engenho MHEG

Um engenho MHEG pode ser constituído por várias entidades, em que algumas são dependentes da complexidade das funcionalidades que é necessário suportar. O módulo de codificação ASN.1, utilizado para o intercâmbio entre sistemas MHEG ou para a criação de objectos por um sistema de autoria, e o módulo de gestão de *scripts* são exemplos de facilidades opcionalmente suportadas por um engenho MHEG.

Como vimos no capítulo anterior, a tecnologia OLE2, disponível no ambiente Windows, não fornece actualmente todas as facilidades que é usual encontrar num sistema distribuído, razão pela qual foi necessário fazer algumas restrições no modelo proposto.

De tudo o que foi referido sobre as funcionalidades necessárias para a interpretação e apresentação de aplicações multimédia, resulta quase necessariamente um modelo com pouca distribuição. Partindo do exemplo referido na figura 6.1, podemos definir três componentes autónomos, que podem efectivamente ser distribuídos por várias máquinas: o engenho MHEG, a aplicação e os serviços de apresentação e interacção.

Outra restrição consiste em considerar que, em utilizações razoavelmente complexas do engenho, não é necessária uma aplicação de controlo, possivelmente remota, mas apenas um conjunto de *scripts*, que são transferidos para o terminal, juntamente com a restante informação, e interpretados localmente.

A utilização de *scripts*, mesmo em apresentações multimédia simples, é extremamente conveniente por permitir a manipulação de variáveis e de estruturas de controlo de fluxo. Outra situação que requer o uso de *scripts* consiste no desenvolvimento de aplicações telemáticas, em que o volume da informação multimédia sugere a estruturação em bases de dados.

Como a representação de intercâmbio de *scripts* está ainda em processo de normalização [MHEG 94b], foi considerada neste modelo a utilização do *VisualBasic*, que cumpre todos os requisitos de uma linguagem de *script* e possui funcionalidades de comunicação com serviços desenvolvidos com base na tecnologia OLE2.

Uma arquitectura bastante comum, e com interesse para aplicações multimédia, baseia-se na implementação de vários terminais de apresentação, controlados por um engenho MHEG central. Os serviços de apresentação e interacção são, nesta arquitectura, fornecidos por cada equipamento terminal. Para obedecer a este requisito, o modelo proposto considera a utilização da tecnologia *Network Dynamic Data Exchange* (NDDE) do ambiente Windows, de modo a permitir a comunicação entre o engenho MHEG e vários interfaces de apresentação de terminais ligados por uma rede *Windows for Workgroups* [Leal 95].

O interface de comunicação do ambiente Windows é suficientemente genérico para permitir implementar os métodos de referência externa mais usuais actualmente, como o *Universal Resource Locator*. O estabelecimento de ligações comutadas para transmissão em

tempo real de *streams* de dados multiplexados pode ser considerado, utilizando a tecnologia de Redes Digitais com Integração de Serviços (RDIS) ou redes de distribuição por cabo.

Tomadas em conta estas restrições resulta um modelo possível para um engenho MHEG, descrito na figura 6.2 e constituído por cinco módulos com responsabilidade de gestão das várias funcionalidades do engenho e um interface de programação acessível aos processos de *script*.

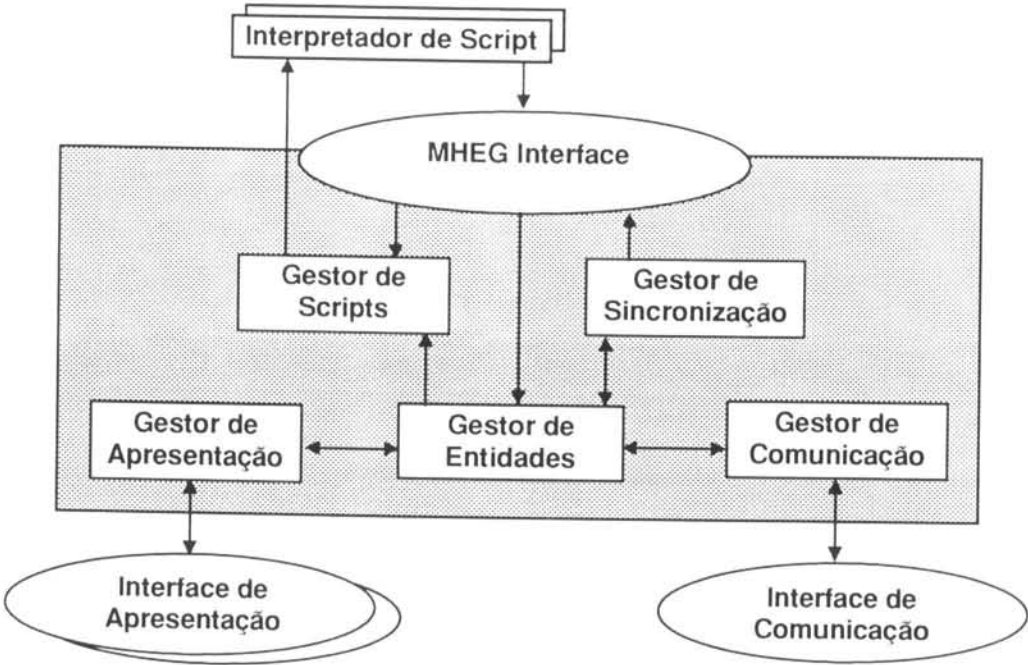


Figura 6.2: Modelo proposto para o engenho MHEG.

6.2 Gestor de Entidades MHEG

O gestor de entidades MHEG é um processo que permite manipular, num formato interno, os objectos MHEG, objectos de execução e canais de apresentação sob controlo do engenho.

Este processo oferece as funcionalidades necessárias para alocar as várias entidades e aceder aos estados e valores correntes dos atributos. Internamente, o gestor de entidades coordena a preparação dos objectos, incluindo operações de consulta perante o gestor de comunicação para resolução de referências externas, e a instanciação de objectos de execução e de canais de apresentação.

A criação de objectos de execução e os seus atributos são comunicados ao gestor de apresentação do canal a que foram associados. Os *scripts* de execução são inicializados com os parâmetros de entrada correspondentes, por invocação do gestor de *scripts*.

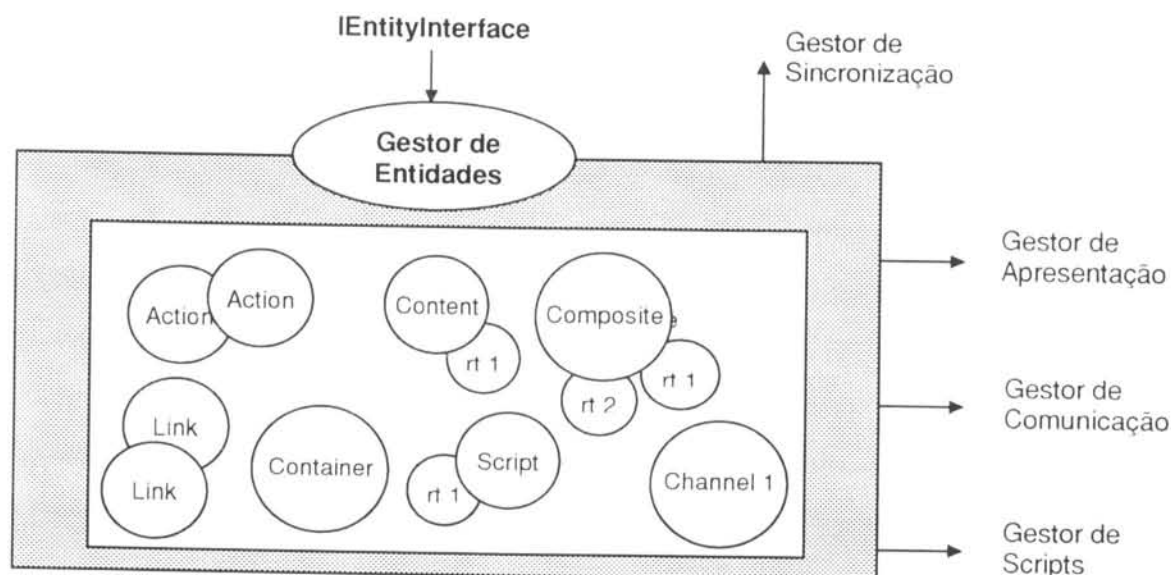


Figura 6.3: Gestor de entidades.

A figura 6.3 representa a estrutura interna do gestor de entidades e os vários interfaces de comunicação com os outros módulos. O gestor de entidades oferece um único interface, **IEntityInterface**, utilizado pela generalidade dos módulos do engenho para obter e modificar os estados e valores dos atributos dos objectos sob o seu controlo. Este interface fornece também funções para a criação e destruição de entidades, que são controladas a partir do interface do engenho MHEG.

Tipicamente, os objectos que são referenciados num repositório remoto são contentores ou objectos de dados. Uma acção de preparação de uma referência externa de um contentor, invocada sobre o **IEntityInterface**, desencadeia um pedido ao gestor de comunicação, para a transferência e decodificação do objecto. O módulo de decodificação ASN.1 transforma o objecto codificado numa representação interna das entidades por que é constituído, invocando acções de criação de objectos MHEG sobre o **IEntityInterface**.

A preparação de um objecto de dados com referência exterior consiste na transferência e no carregamento deste em memória ou, caso seja um objecto de dados multiplexados com

requisitos de tempo real, no estabelecimento de uma ligação comutada com o servidor onde este se encontra.

Por último, o gestor de entidades desempenha outro papel de extrema importância para o engenho, que consiste na geração de eventos. Os eventos são gerados pela alteração de um estado ou do valor de um atributo e comunicados ao gestor de sincronização, que desencadeia um ciclo de invocações do `IEntityInterface`, para obter valores correntes de atributos e estados de objectos, necessários para a avaliação das condições dos *links* disparáveis.

A declaração da classe correspondente ao gestor de entidades, utilizando as funcionalidades existentes nas *Microsoft Foundation Classes* (MFC) para implementação de interfaces, é exemplificada na figura 6.4. A classe `CCmdTarget`, da qual os vários módulos são derivados, permite gerir transparentemente o mecanismo de contagem de referências e a função `QueryInterface`, porque fornece uma implementação do interface básico dos objectos *Windows* `IUnknown`.

```
class CEntityHandler : public CCmdTarget
{
public:
    // atributos e funções específicos da classe

// Interface Maps
protected:
    DECLARE_INTERFACE_MAP()
    BEGIN_INTERFACE_PART(EntityObj, IEntityInterface)
        STDMETHOD_(void, create) (OBJTYPE, OBJID);
        STDMETHOD_(void, prepare) (OBJID);
        STDMETHOD_(void, destroy) (OBJID);
        STDMETHOD_(void, new) (OBJID, NUMBER);
        STDMETHOD_(void, delete) (OBJID, NUMBER);
        STDMETHOD_(void, run) (OBJID, NUMBER);
        STDMETHOD_(void, stop) (OBJID, NUMBER);
        STDMETHOD_(void, set) (OBJID, NUMBER, PARAM);
        STDMETHOD_(VALUE, get) (OBJID, NUMBER, PARAM);
    END_INTERFACE_PART(EntityObj)
};
```

Figura 6.4: Declaração do gestor de entidades.

O interface `IEntityInterface` é constituído por funções cujo comportamento corresponde, no geral, às acções elementares com o mesmo nome definidas no engenho MHEG.

Como foi referido, uma acção `prepare`, com uma referência `OBJID` externa para um objecto contentor, pode originar indirectamente a criação de vários objectos MHEG. Um objecto MHEG é gerado pelo decodificador ASN.1 utilizando para o efeito a função `create` do interface, na qual o parâmetro `OBJTYPE` identifica a classe do objecto MHEG. Um objecto de execução é instanciado como elemento de uma lista ligada, encabeçada pelo objecto modelo que lhe deu origem, utilizando a função `new` com o parâmetro `NUMBER` definido pelo autor do objecto.

Na inicialização do gestor de entidades é definido um canal de apresentação por defeito cujo `OBJID` é igual a zero, sendo uma acção elementar *new channel* traduzida por uma invocação da função `new` sobre esta entidade.

A função `run` é aplicada a objectos de execução associados a um canal de apresentação. Posteriores modificações de atributos relacionados com o objecto são imediatamente traduzidas em notificações perante o gestor de apresentação.

As funções de consulta e modificação de atributos e estados (`get` e `set`) têm a particularidade de permitir também o acesso aos parâmetros dos objectos MHEG.

6.3 Gestor de Sincronização

O gestor de sincronização é o interpretador dos comportamentos definidos para os objectos MHEG e é constituído por dois submódulos: o processador de *links* e o processador de acções. Este gestor oferece um interface **`ISchedulerInterface`** que está associado ao processador de *links*.

O interface `ISchedulerInterface` fornece unicamente uma função, **`sendEvent`**, que é utilizada pelo gestor de entidades para notificar alterações em estados e atributos dos objectos. Os parâmetros da função são a identificação da entidade modificada, o atributo ou estado e o seu valor actual.

Processador de *Links*

Quando uma acção prepara um objecto *link*, é gerado um evento e o processador de *links* é notificado. Mas antes de iniciar a avaliação das condições de disparo associadas ao evento, o processador consulta o gestor de entidades e obtém a condição do *link* correspondente ao novo objecto. Uma condição de *link* consiste numa condição de disparo, numa restrição, ou numa combinação lógica de várias condições. O processador mantém tabelas actualizadas com estes três tipos de informação.

Quando um evento é gerado, o processador de *links* avalia todas as condições de disparo constantes da tabela correspondente, seguidamente verifica as restrições associadas aos *links* disparados, e por fim determina as condições de *link* que se tornaram verdadeiras. Esta informação é utilizada para obter a lista de acções que devem ser despoletadas paralelamente.

Para cada acção a executar é criado o processador de acções correspondente e, se existirem parâmetros *macro*, estes são também passados na inicialização.

A figura 6.5 pretende mostrar a arquitectura interna de um gestor de sincronização.

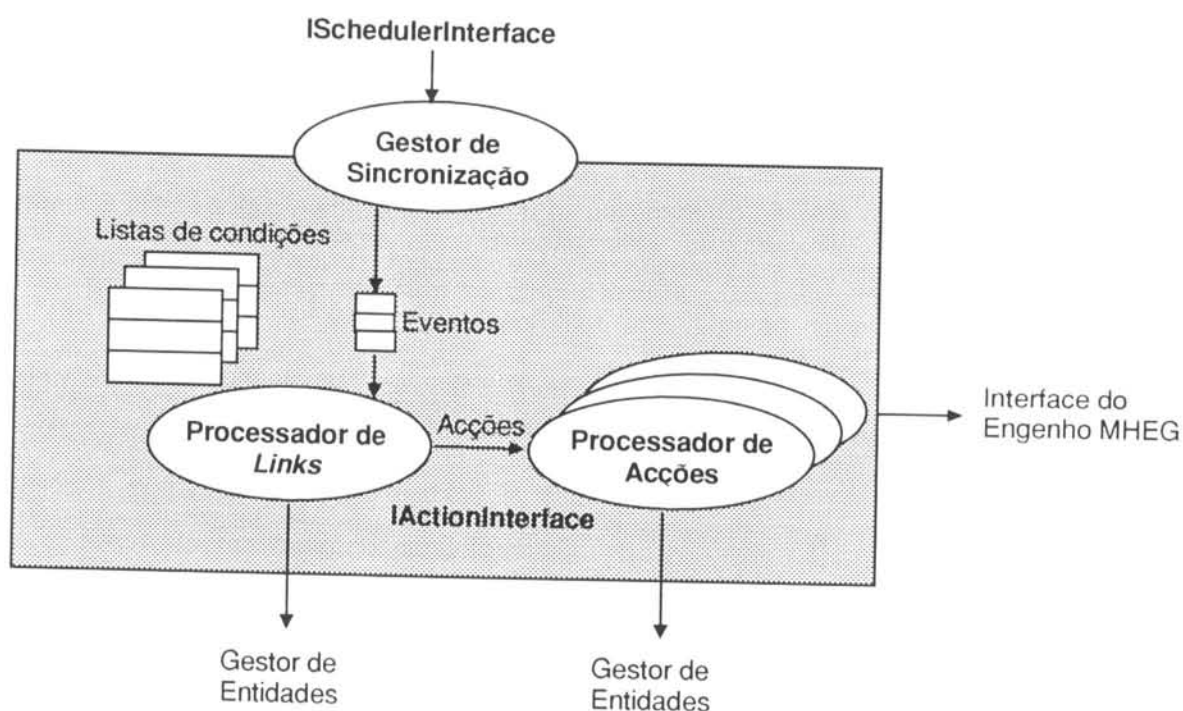


Figura 6.5: Gestor de sincronização.

Processador de acções

Um processador de acções oferece um interface **IActionInterface** com uma única função **sendAction**, cujos parâmetros são a identificação da acção a executar e uma lista de parâmetros *macro* opcional.

Quando um processador de acções é inicializado, consulta o gestor de entidades para obter os parâmetros do objecto *Action*. O processamento de uma acção não é trivial, uma vez que é possível ter várias acções estruturadas recursivamente e com sincronização em série ou em paralelo. Uma acção pode também ter definidos ciclos sucessivos de processamento.

Para compreender o comportamento do processador de acções é necessário introduzir os conceitos de efeito de uma acção e efeito perceptível de uma acção.

O **efeito de uma acção** elementar ocorre desde o instante em que a acção é invocada sobre os alvos até que os efeitos produzidos no engenho MHEG, como uma alteração num atributo ou num estado terminam. Excepto no caso de acções de preparação de um objecto referenciado externamente, não é previsível que este efeito demore muito tempo.

O efeito de uma acção *run* resulta na adição do efeito sobre o engenho MHEG e do efeito perceptível pelo utilizador. O **efeito perceptível** pelo utilizador ocorre enquanto o objecto não é retirado da apresentação por efeito de uma acção *stop*.

Durante o efeito sobre o engenho MHEG, os atributos do objecto estão indefinidos e o estado de execução (*running status*) indica que este se encontra em processamento. Se o alvo de uma acção *set* está em execução e se é definida uma duração da transição, o valor do atributo varia gradualmente até ao valor especificado.

A execução de uma acção de um conjunto definido num objecto *Action* só pode ser iniciada após a conclusão do efeito sobre o engenho MHEG da acção, ou das acções anteriores, se forem sincronizadas em paralelo.

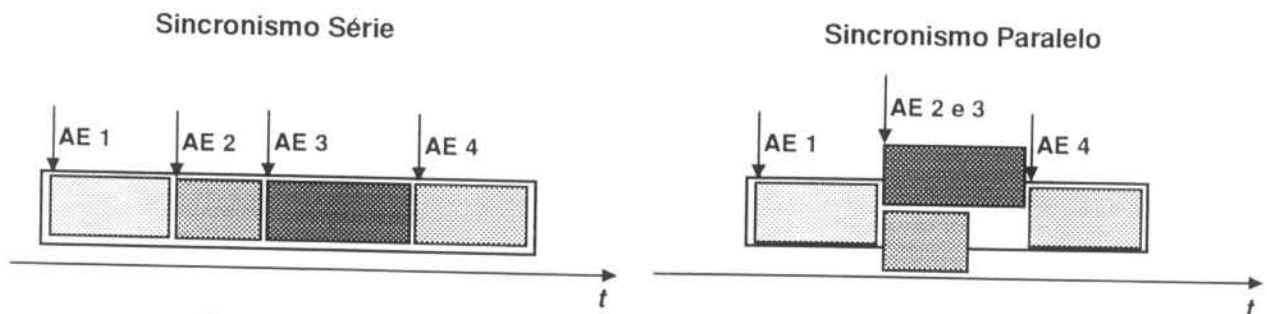


Figura 6.6: Processamento de um conjunto de acções elementares.

Se um grupo de acções sincronizadas está definido com um número de ciclos de processamento, a entrada num novo ciclo depende da conclusão dos efeitos perceptíveis pelo utilizador de todas as acções *run* existentes no grupo.

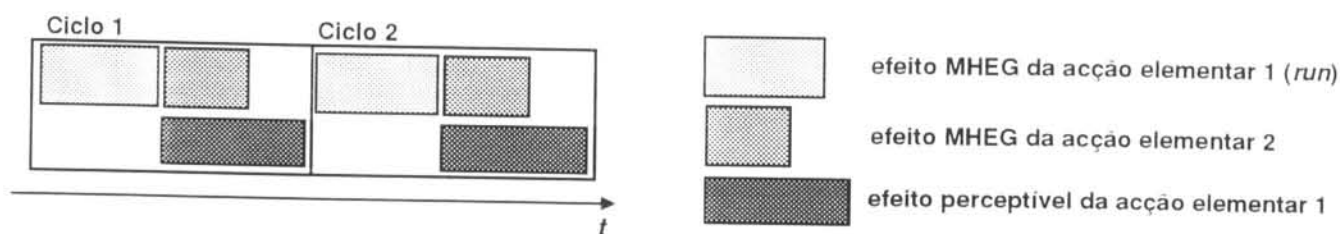


Figura 6.7: Ciclos de processamento de um grupo de acções.

A figura 6.8 descreve o algoritmo utilizado pelo processador de acções para atingir o comportamento descrito. No fim de um ciclo de processamento são avaliados os alvos que estão em execução e, se já concluíram o comportamento especificado, são retirados da apresentação, terminando o efeito perceptível ao utilizador.

Se uma acção for executada sobre vários alvos não é exigido ao engenho MHEG o processamento paralelo da acção em cada alvo.

As funcionalidades que o gestor de sincronização suporta, pela utilização coordenada de um gestor de *links* e vários gestores de acções, é consideravelmente mais complexa do que o esquema de sincronização apresentado no PREMO [PREMO 94b], baseado na combinação de objectos *Controller* e *EventHandler*.

A avaliação de condições para a geração do evento, suportada pelo objecto *EventHandler*, só permite considerar restrições. Além disso, uma máquina de estados finitos, como a definida no objecto *Controller*, é pouco flexível e muito menos poderosa do que a combinação dinâmica de *links* e acções, que é possível definir utilizando a representação MHEG.

```

BEGIN ACTION PROCESS of action describing the link effect
  FOR I = 1 TO number_of_performances
    DO
      IF synchro_indicator==serial
        FOR J= 1 TO number_of_synchronized_actions
          DO
            IF synchronised_action J==elementary_Action
              PERFORM Target Resolution of elementary_action J
              PERFORM the action effect of elementary_action J on target_set

            ELSE IF synchronised_action J==action_object
              PERFORM RECURSIVE ACTION PROCESS of action_object J
            ENDIF
          WAIT FOR END OF MHEG effect of synchronised_action J
        ENDIF
      ELSE IF synchro_indicator==parallel
        FOR J= 1 TO number_of_synchronised_actions
          DO SIMULTANEOUSLY
            IF synchronised_action J==elementary_action
              PERFORM Target Resolution of elementary_action J
              PERFORM the action effect of elementary_action J on target_set

            ELSE IF synchronised_action J==action_object
              PERFORM RECURSIVE ACTION PROCESS of action_object J
            ENDIF
          WAIT FOR END OF MHEG effect of each actions in each 'synchronised actions'
        ENDIF
      WAIT FOR END OF user effect of each action 'run' in each 'synchronised actions'
      DURING that time IF a target is 'RUNNING' AND
        IF 'its current temporal position==its terminal position'
          PERFORM a 'stop' action on this target
        ENDIF
    ENDIF
  END OF ACTION PROCESS of action object describing the link effect

```

Figura 6.8: Algoritmo de processamento de um objecto *Action*.

6.4 Gestor de Comunicação

O gestor de comunicação é responsável pela resolução de referências externas descritas em objectos MHEG. Este módulo oferece um interface **IPrepareInterface**, com funcionalidades para a transferência de ficheiros remotos e para o estabelecimento de ligações comutadas com servidores de informação.

A preparação de um objecto MHEG, identificado por uma referência externa, é iniciada pela transferência da informação para a memória local. Seguidamente, o preparador controla o decodificador ASN.1 [Silva 95] e converte o objecto para o formato interno de representação do gestor de entidades, por invocação do interface **IEntityInterface**.

A preparação de um objecto de dados poderá consistir na sua transferência para memória ou para um ficheiro armazenado localmente, dependendo da capacidade de memória do sistema e da dimensão da informação referenciada. Um objecto de dados multiplexados faz referência tipicamente a uma combinação de *streams* de áudio e vídeo existentes num servidor remoto que são acedidas por uma ligação comutada, cuja identificação é comunicada ao gestor de entidades, para posterior utilização pelo servidor de apresentação.

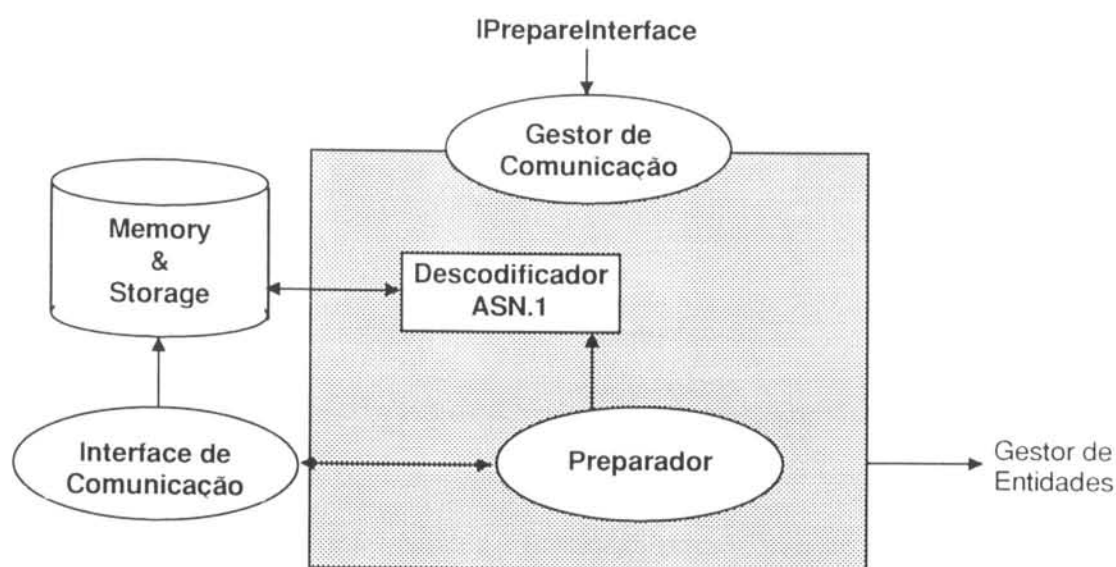


Figura 6.9: Gestor de comunicação.

6.5 Gestor de Apresentação

O gestor de apresentação tem sob o seu controlo a criação e gestão dos vários canais de apresentação requeridos pelo engenho MHEG. O gestor de entidades invoca o interface **ICannelInterface** para estabelecer ligações com engenhos de apresentação. Para permitir a utilização dos serviços de apresentação de vários terminais multimédia, a gestão dos canais é suportada por um interface de comunicação NDDE [Leal 95].

A interacção com o utilizador, nos modos de selecção e modificação de objectos, e a geração de eventos no decorrer da apresentação com base em marcas temporais são notificadas pelo gestor do canal respectivo, alterando o estado correspondente do objecto de execução sob controlo do gestor de entidades.

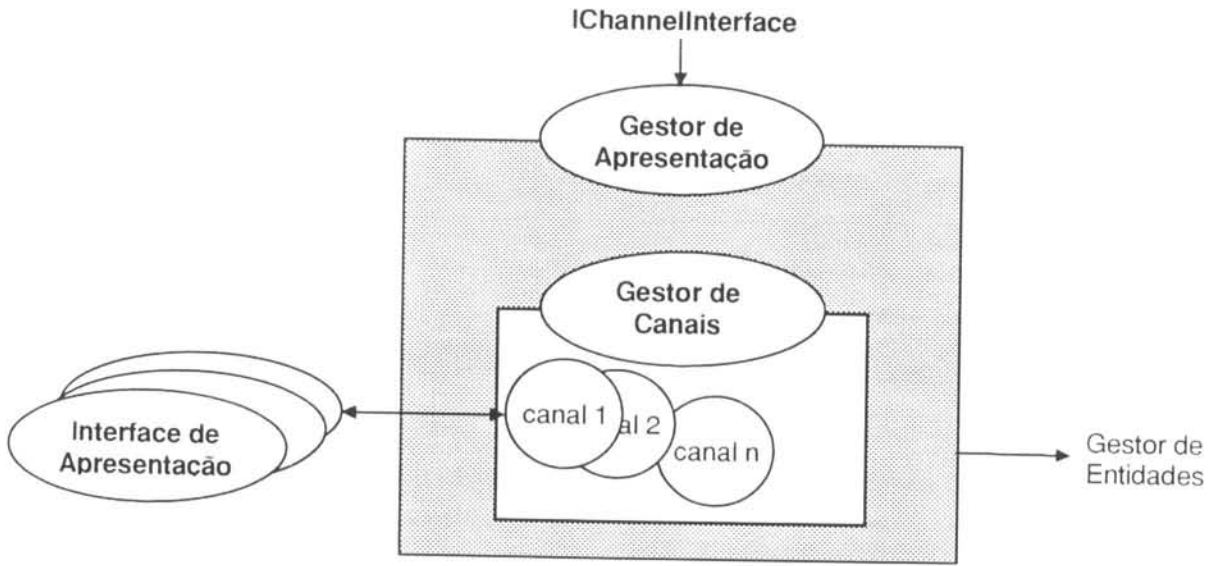


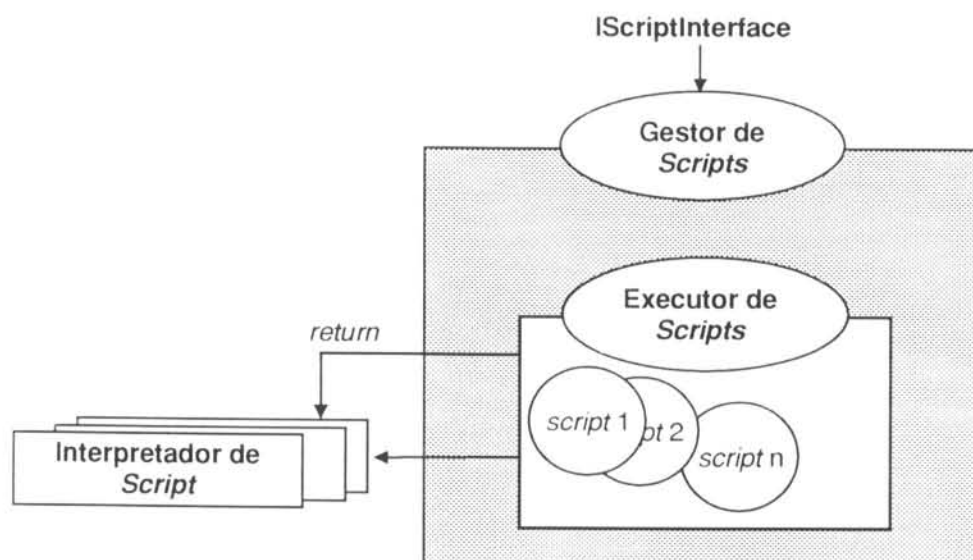
Figura 6.10: Gestor de apresentação.

6.6 Gestor de *Scripts*

O gestor de scripts oferece funcionalidades diversas através do interface **IScriptInterface**. Os *scripts* são instanciados pelo gestor de entidades e ficam sob o controlo do executor de *scripts*, podendo ser executados repetidamente.

Como foi referido no capítulo 3, um *script* pode ser inicializado antes de ser executado por efeito de uma acção *set parameters* sobre o objecto de execução correspondente. A passagem de parâmetros por referência a objectos de dados permite ao engenho MHEG consultar os resultados da execução de um *script*.

Um script pode também bloquear no decorrer da sua execução, à espera de uma acção de retorno de parâmetros pelo engenho MHEG. O *script* a que deve ser comunicada uma acção *return* é identificado pelo valor do indicador de retorno. Como é proposta a utilização da linguagem *VisualBasic* para programação do *script*, é possível utilizar o interface de comunicação DDE para retornar esta notificação.

Figura 6.11: Gestor de *scripts*.

6.7 Interface do Engenho MHEG

O interface do engenho MHEG é invocado pelo processador de acções do gestor de sincronização e pelos *scripts* que estiverem em execução num dado momento. As funcionalidades implementadas correspondem às acções elementares definidas na norma MHEG, acções que permitem criar novas entidades, ou modificar e consultar os comportamentos de todas as entidades sob controlo do engenho MHEG.

Um dos requisitos deste módulo consiste em ser possível a sua manipulação por linguagens de *script* de alto nível. Por esta razão, o módulo é implementado utilizando o interface de invocação dinâmica **IDispatch**, sobre o qual é suportada a tecnologia *Automation* para interfaces de programação do Windows, já referida no capítulo anterior.

Este interface permite definir nomes externos para métodos e variáveis pertencentes a uma classe, que são depois facilmente invocados no próprio código da linguagem de *script*. O módulo foi definido utilizando as *Microsoft Foundation Classes* (MFC), o que permite encapsular os detalhes de implementação da tecnologia *Automation*.

As acções elementares correspondentes a operações de consulta ou modificação de atributos e estados dos objectos foram implementadas por métodos correspondentes, que actuam sobre variáveis definidas em classes derivadas da classe **CMHEGEntity**. Acções de criação de entidades são mapeadas em métodos definidos pela classe correspondente a cada entidade.

A figura 6.12 exemplifica a declaração da classe `CMHEGComposite`, que permite invocar acções sobre um objecto composto, ou sobre um composto de execução.

```
class CMHEGComposite : public CMHEGEntity
{
public:
    string EntId;
    long TempTerm;
    ...
    LPDISPATCH GetTempTerm(void);
    void SetTempTerm(LPDISPATCH newValue);
    void Prepare(void);
    ...
    DECLARE_INTERFACE_MAP()
};

// Dispatch Maps

BEGIN_DISPATCH_MAP(CMHEGComposite, CMHEGEntity)
    DISP_PROPERTY(CMHEGComposite, "EntityID", EntId, ...)
    DISP_PROPERTY_EX(CMHEGComposite, "TemporalTermination",
                     GetTempTerm, GetTempTerm, ...)
    DISP_FUNCTION(CMHEGComposite, "Prepare", Prepare, ...)
END_DISPATCH_MAP()
```

Figura 6.12: Exemplo de declaração da classe `CMHEGComposite`.

Capítulo 7

Conclusões

“Num meio electrónico, o leitor é capaz, literalmente, de “tocar o texto”, num processo de leitura criativa muito diferente daquele a que a escrita tradicional nos habituou.”

D. Bolter, Em *Writing Space: the Computer, Hypertext and the History of Writing*, 1991

O advento das tecnologias multimédia é já hoje uma realidade, anunciada por experiências prometedoras que decorrem por todo o mundo e que deixam vislumbrar aplicações inovadoras e de grande utilidade, trazendo novo fôlego às indústrias de computadores, da electrónica de consumo e das telecomunicações.

É verdade que a oferta tecnológica está preparada para a produção em larga escala de aplicações multimédia interactivas mas, infelizmente, os produtos disponíveis são em grande parte dependentes da plataforma utilizada, o que inibe o potencial desenvolvimento desta nova indústria. Tanto os fabricantes de equipamento terminal, como os futuros editores de publicações electrónicas, necessitam de uma norma de aceitação generalizada, que lhes assegure a perenidade do investimento no mercado multimédia.

Com base nestes pressupostos, as principais operadoras de telecomunicações europeias, reunidas no projecto EURESCOM IMS1, avaliaram e comentaram a tecnologia definida pela norma MHEG, para a criação de serviços de consulta multimédia em redes públicas de telecomunicações.

EURESCOM IMSI

O principal objectivo do projecto *Integrated Multimedia Services* (EURESCOM IMS1) consiste na definição e experimentação de um serviço de consulta de informação multimédia, assente sobre redes públicas de telecomunicações, com débito mínimo de 1.5 Mbit/s.

O modelo baseia-se em bases de dados centralizadas contendo informação monomédia e sequências interactivas codificadas em MHEG. As aplicações disponíveis, residentes no servidor, definem a apresentação das sequências MHEG no terminal. A interactividade com o utilizador é utilizada pela aplicação para gerar interrogações à base de dados, ou para controlar uma *stream* MPEG1, transmitida num canal em tempo real [DSMCC 94].

O projecto utilizou um conjunto de ferramentas MHEG desenvolvidas pelo CCETT, as primeiras existentes no mercado [Picquet+ 93]. O núcleo destas ferramentas reside numa biblioteca de classes C++ [MCL 94], que define um formato de representação interna dos objectos MHEG, e em *codecs* ASN.1. O pacote para PC/Windows consiste ainda n um engenho interpretador da versão CD MHEG [Normand 94], num servidor DDE de apresentação e num editor visual de objectos MHEG [ANIMA 94], que permite fazer a composição espacial e temporal e definir o comportamento dos objectos em sequências MHEG.

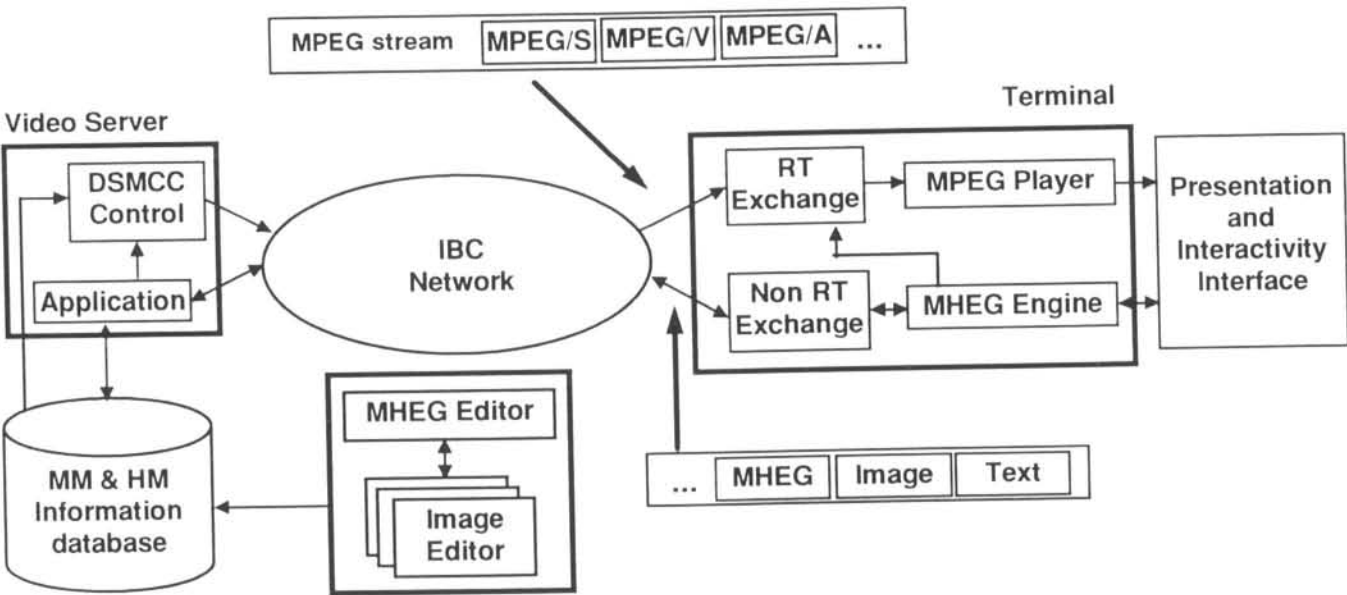


Figura 7.1: Arquitectura do sistema *Integrated Multimedia Services* (IMS1).

O projecto IMS1 permitiu, ao grupo de investigação em que o autor desta dissertação está inserido, testar as funcionalidades definidas na norma MHEG e experimentar as

dificuldades de criação de uma aplicação específica de *Video on-demand* [Henriques 95]. A utilização das ferramentas de geração de objectos MHEG fornecidas no projecto, pôs em evidência, fundamentalmente, a importância do nível de informação de *script* na implementação de uma aplicação com requisitos básicos de funcionalidade.

Outros projectos europeus têm também como base comum o MHEG, mas concentraram a sua investigação em áreas distintas, como os sistemas distribuídos (GLASS), os serviços de consulta em redes de banda larga (MARS), as redes de distribuição por cabo (AMMIS) e a produção de aplicações multimédia (OMHEGA). A base consensual de operadoras de telecomunicações, fabricantes de sistemas e editores, assim reunida, permite pensar que o MHEG reúne as condições necessárias para ser o motor de uma explosão na área da informação multimédia.

Em termos tecnológicos, o MHEG parece ser a norma mais adequada, porque define uma representação codificada em formato final, ideal para o armazenamento, a transmissão e a apresentação de informação intrinsecamente relacionada, e também porque oferece extensões para o suporte de representações mais complexas.

Na elaboração desta dissertação pesaram todas estas considerações e ficou bem patente a importância estratégica de dominar as tecnologias de representação de informação e de desenvolvimento de componentes de serviço, para conquistar um lugar no mercado das aplicações telemáticas multimédia, que irá dominar o futuro próximo.

As ferramentas de criação de informação multimédia existentes permitem, de um modo geral, todas as funcionalidades descritas no MHEG, pelo que o mais natural será adoptarem este como formato final da produção, em detrimento dos seus formatos proprietários.

O desenvolvimento de terminais multimédia, por outro lado, estará sempre relacionado com as características da plataforma, em termos de recursos de rede, capacidades gráficas ou desempenho, pela adopção do MHEG oferece maiores oportunidades na criação de produtos inovadores e mais eficientes.

A implementação de um engenho interpretador MHEG, proposta para a plataforma PC/*Windows*, enquadra-se nesta perspectiva e considera os graus de distribuição mais úteis e viáveis do módulo de interpretação e apresentação. Da análise efectuada concluiu-se que, por razões de eficiência, o interpretador MHEG deve consistir num único componente de serviço e também que pode fazer sentido a distribuição dos componentes de apresentação e de interpretação de *scripts*.

Em termos futuros, as funcionalidades de um interpretador MHEG poderão vir a ser incorporadas nos serviços de suporte multimédia do sistema operativo. A extensibilidade das funcionalidades básicas, definidas pelo MHEG para a interpretação de informação multimédia e hipermédia, é assegurada pelo módulo de *script*. A utilização de sistemas distribuídos permitirá conceber *scripts* que utilizem os serviços disponíveis na rede de modo inteligente e transparente.

7.1 Objectivo: Conectividade Global

A evolução tecnológica das telecomunicações aponta claramente no sentido da conectividade global das redes de informação. Com esta dificuldade infraestrutural em vias de resolução, as atenções voltam-se para os problemas de intercâmbio de informação, entre aplicações telemáticas heterogéneas, e para a definição de mecanismos de navegação, que possibilitem um acesso rápido e eficaz a qualquer informação existente.

Adoptando uma arquitectura com vários níveis de complexidade independentes, é possível utilizar os serviços de informação global actualmente disponíveis, para criar aplicações multimédia para o grande público, e oferecer aos autores garantias mínimas de compatibilidade da informação gerada com os serviços e sistemas distribuídos da próxima geração.

7.1.1 World Wide Web

O serviço *World Wide Web* (WWW) é o mecanismo de consulta remota de informação multimédia, baseado em hipertexto, mais utilizado na rede *Internet*. A criação de documentos com facilidades hipermédia é descrita numa variante do SGML, normalizada pelo IETF, a linguagem *HyperText Markup Language* [HTML]. Esta especificação permite descrever a estrutura de um documento, indicando a localização de âncoras e *links* no texto e determinando o modo de resolução de cada *link*.

Uma interacção entre um cliente e um servidor WWW consiste na transferência de um documento HTML ou de um ficheiro contendo informação multimédia, por interacção do utilizador com um *link*. A informação multimédia é interpretada e apresentada utilizando aplicações exteriores (*media viewers*).

Como a informação pode estar em qualquer parte da rede *Internet* o protocolo de transferência utilizado deve ser rápido e não possuir estados, suportando um sistema distribuído de informação hipermédia eficiente. O *HyperText Transport Protocol* [HTTP]

consiste num protocolo orientado para objectos, que fornece um conjunto simples de comandos e permite negociar a representação da informação.

O protocolo de comunicação HTTP é baseado em ligações TCP/IP, em que cada transacção de informação exige o estabelecimento de uma conexão, um pedido e a transferência da respectiva resposta e a desconexão. A informação multimédia é encapsulada num pacote do formato *Multipurpose Internet Mail Extensions* [MIME].

O terceiro componente do WWW é o conceito de *Universal Resource Locator* [URL], que fornece um esquema global de referência a objectos ou recursos da rede, permitindo a transferência de documentos, operações de procura textual ou o acesso a outros serviços disponibilizados pela rede.

A filosofia do WWW reside na simplicidade de abordagem da representação de um documento hipermédia e no encapsulamento de grande parte dos serviços usuais na *Internet*. A representação de documentos em HTML é baseada num modelo de hipertexto bastante convencional, cuja diferença consiste no grau de distribuição da informação e não na qualidade das suas relações intrínsecas.

Exceptuando algumas facilidades de interrogação, o modelo WWW não define mecanismos com expressividade suficiente para representar comportamentos complexos, como a sincronização temporal dos vários média ou a gestão dinâmica dos *links* e dos seus efeitos.

7.1.2 O Modelo *Theseus*

É usualmente reconhecido que os meios tecnológicos actuais permitem disponibilizar enormes volumes de informação multimédia, oferecendo assim oportunidades únicas no campo da educação e do desenvolvimento pessoal. Contudo, esse facto não garante *'per se'* o fornecimento de informação utilizável. Descobrir facilmente no ciberespaço a informação relevante para as nossas necessidades ou interesses e, seguidamente, voltar com rapidez ao ponto inicial, coloca enormes problemas.

Existe uma necessidade urgente de definir um método viável para formatar e organizar a informação multimédia de modo controlado e coerente. Como Teseu no labirinto, precisamos de encontrar o Minotauro, mas é fundamental que consigamos assinalar o caminho de volta. Com base nesta ideia foi proposto, no início da década, um modelo para o desenvolvimento de bases de dados distribuídas, em que a informação multimédia é cruzada com percursos temáticos, permitindo múltiplas associações de ideias [Harnden+ 93].

A eficiência e atractividade deste modelo é função da riqueza e diversidade da informação constante da base de dados. Por esta razão, os recursos disponibilizados na rede devem ser objectos activos, que reagem interactivamente a um utilizador e suportam referências a múltiplos percursos alternativos.

Ao nível operacional, o modelo é constituído por dois níveis de informação estruturada: o *subject path* e a *mediabase*. A combinação destes dois níveis constitui um meio dinâmico e evolutivo, o *hypermedium*.

Mediabase

A *mediabase* consiste numa base de dados convencional, contendo entidades multimédia que podem ser interactivas. A interactividade de uma entidade é determinada unicamente pela informação que contém e um requisito fundamental de uma entidade consiste na sua objectividade. Em particular, devem limitar-se ao essencial e ao universalmente aceite as descrições textuais ou orais encapsuladas no objecto. Em termos de significado, uma entidade é um objecto auto-suficiente, que pode ser referido e interactuado nos seus próprios termos.

Subject Path

O significado de uma entidade adquire relevância numa argumentação, definida num determinado contexto, que constitui um percurso temático. Um leitor deve poder interrogar uma entidade, acerca dos percursos que a referenciam, e mover-se **ortogonalmente** para qualquer um desses percursos temáticos alternativos.

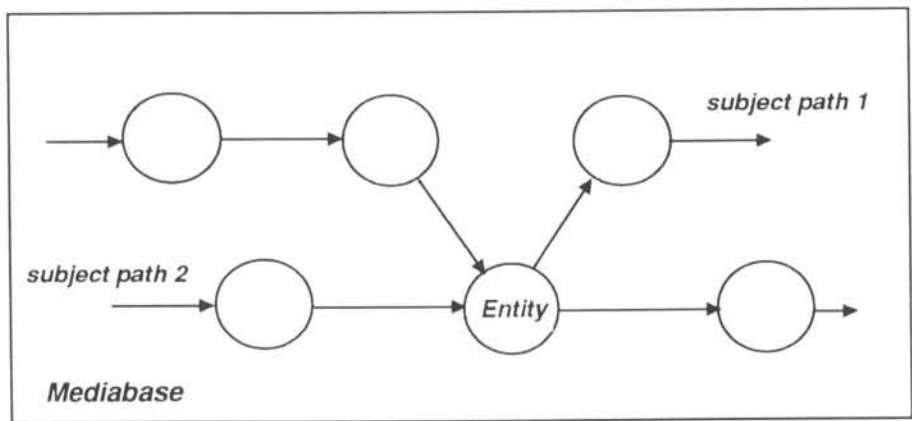


Figura 7.2: Exemplo de percursos temáticos definidos sobre entidades.

Hypermedium

O *Hypermedium* consiste no conjunto de todas as *Mediabases* acessíveis e de todos os percursos temáticos sobre elas definidos num dado momento. Apesar da sua simplicidade, este modelo fornece uma infraestrutura conceptual, que permite manipular enormes volumes de informação, e cuja estrutura global não só é indeterminada como se altera constantemente. Topologicamente, o *Hypermedium* tem tantas dimensões em cada entidade quantos percursos temáticos sobre ela se cruzam.

Este modelo permite distinguir entre os aspectos consensuais da expressão cultural, representados na *mediabase*, e a singularidade da expressão e experiência humana individuais, presente no percurso temático definido por um autor. Os conceitos propostos apresentam também uma correspondência directa com algumas investigações contemporâneas, em áreas tão distintas como a linguagem, o conhecimento e o funcionamento do sistema nervoso.

O desenvolvimento de entidades deve ter como base a definição de domínios de actividade virtuais, associados aos conceitos de entretenimento e de jogo. Na envolvente de um jogo executam-se por vezes tarefas complexas, cuja dificuldade lógica ou analítica pode ser considerada para além da perícia e das motivações diárias do utilizador. Nestes domínios a atenção e a motivação são conquistadas e mantidas utilizando técnicas do âmbito do drama e da estética [Harnden 94].

O modelo proposto não é muito exigente, em termos do desempenho do equipamento terminal e das redes de comunicação, visto consistir basicamente num processo de consulta remota (o percurso temático) e na transferência de entidades auto-suficientes em termos de comportamento.

O sucesso a longo prazo do modelo *Theseus* depende fundamentalmente do envolvimento massivo de utilizadores activos no sistema, que multipliquem as entidades disponíveis e, sobretudo, os percursos definidos sobre essas entidades. Quanto ao problema dos direitos de autor, é em larga medida ultrapassado pela própria natureza do *Hypermedium*, no sentido em que qualquer informação copiada do sistema perderá a sua mais importante qualidade: as composições de informação definidas pelos percursos [Stringer 92].

7.2 Perspectivas Futuras

Esta dissertação teve como objectivo a identificação das tecnologias chave, que permitam uma transição suave para os futuros sistemas distribuídos multimédia. No estado actual de normalização da área multimédia, a única aposta garantida parece ser a representação de informação MHEG. Na área dos sistemas distribuídos, não restam dúvidas relativamente à progressiva “componentização” do *software* e a interoperabilidade entre os vários sistemas existentes avança a bom ritmo.

Assim, os esforços futuros devem iniciar-se pela massificação da tecnologia MHEG, utilizando as infraestruturas mais adequadas à comunicação de informação multimédia actualmente existentes. Em termos tecnológicos e comerciais, as redes de comunicação que podem, com sucesso, servir de base à criação de serviços telemáticos multimédia são: as redes de difusão e a *Internet*.

Existem já vários projectos de televisão interactiva que utilizam as redes de difusão por cabo e as redes telefónicas convencionais para o retorno da informação de controlo. A integração do MHEG em *streams* do MPEG2, referida no capítulo 3, oferece uma das possibilidades mais interessantes para a criação de serviços de consulta multimédia com requisitos de tempo real e de alto débito.

A arquitectura deste sistema é apresentada na figura 7.3.

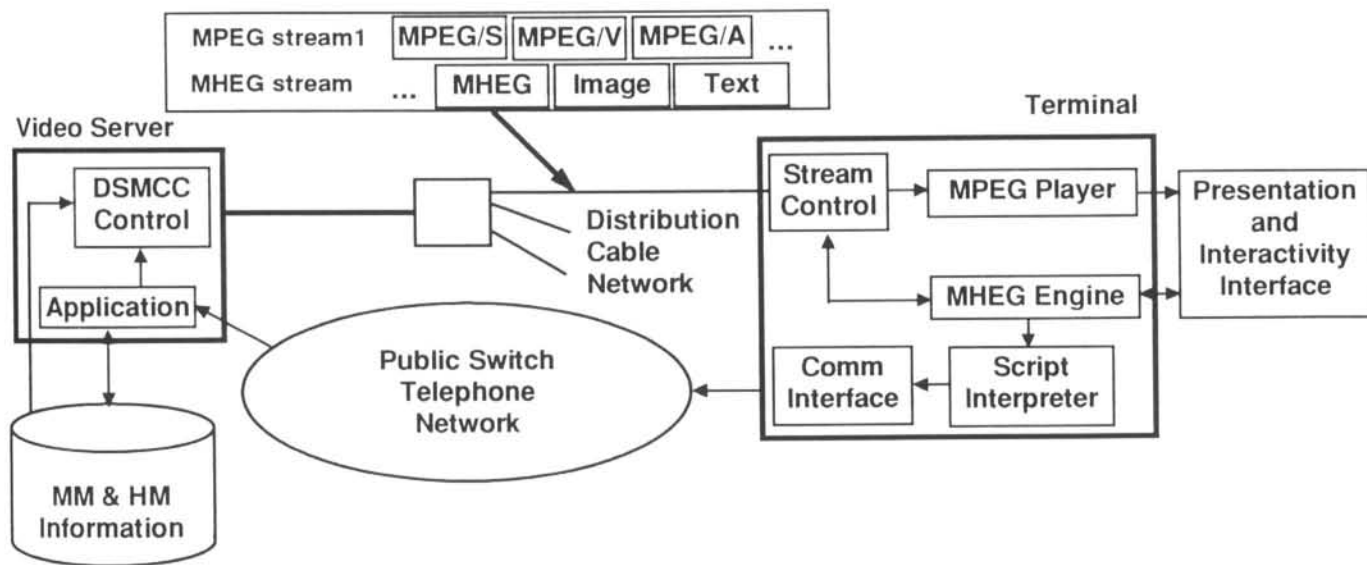


Figura 7.3: Arquitectura de um serviço de consulta em tempo real.

Os serviços multimédia com facilidades de conectividade global recorrem a redes de banda larga e de maior complexidade. Contudo, aplicações telemáticas de consulta multimédia, pouco exigentes em termos de volume de informação transferida, podem ser implementadas sobre o serviço WWW da *Internet* e utilizar eficientemente metodologias de representação da informação como o modelo *Theseus*, com base em objectos MHEG.

Esta arquitectura é apoiada na infraestrutura de conectividade global disponibilizada pelo serviço WWW, e utiliza o modelo convencional de hipertexto HTML, para desenvolver a argumentação de um percurso temático. A inovação da arquitectura proposta reside na codificação em MHEG, das funcionalidades interactivas e de interrogação das entidades da *mediabase*, requeridas pelo modelo *Theseus*.

Com base em clientes WWW convencionais, é possível transferir objectos interactivos MHEG. O engenho MHEG utiliza o protocolo HTTP para resolução de referências externas existentes nos objectos, e controla, por interacção do utilizador, a comutação entre percursos. Os percursos temáticos são definidos por *links* para objectos MHEG em páginas convencionais HTML, e por uma tabela que associa cada objecto com as páginas que o referenciam. Deste modo, o autor da argumentação, tem ao seu dispor as funcionalidades usuais do serviço WWW e mecanismos de interactividade e sincronização complexos, encapsulados em objectos MHEG.

Esta arquitectura permite desenvolver soluções viáveis em vários domínios, e está a ser considerada para o desenvolvimento de uma aplicação de ensino à distância, utilizando a Rede Digital com Integração de Serviços (RDIS) para obter um débito mínimo de acesso à *Internet* de 64 Kbit/s, a partir de qualquer instituição de formação, ou mesmo de casa do formando.

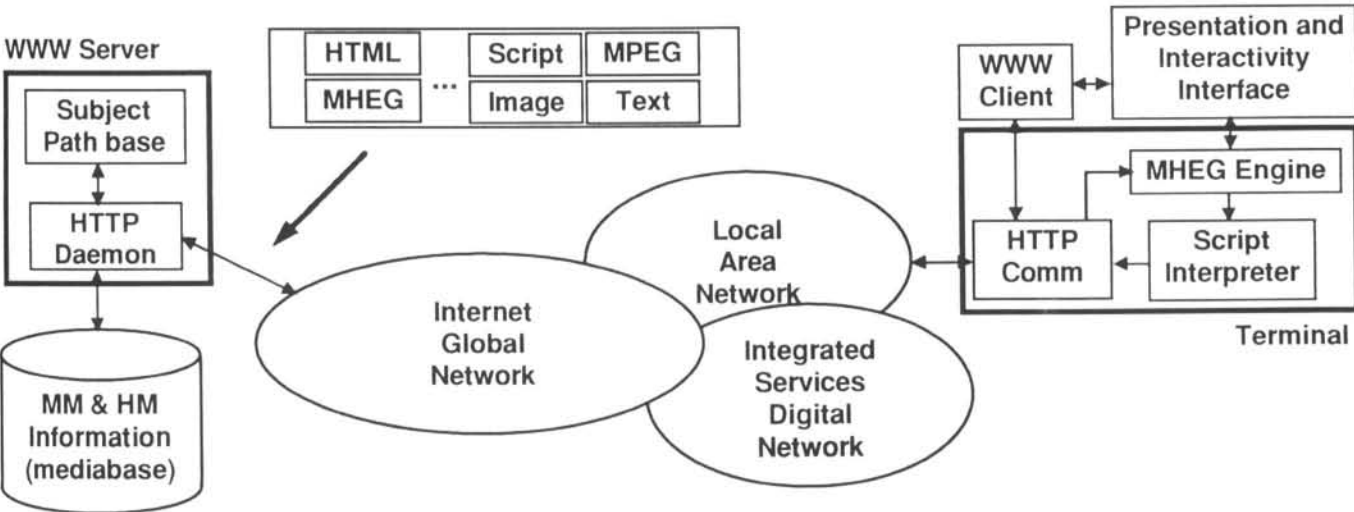


Figura 7.4: Arquitectura de um serviço global de informação.

Bibliografia

- [ANIMA 94] ADV Technologies
MHEG Editor - ANIMA User's Manual
vers. 1.0, Junho 1994
- [ANSA 93] Architecture Projects Management Ltd (ANSA Workprogramme)
System Programming in ANSAware 4.1
ANSAware 4.1, doc. RM.101.02, Fevereiro 1993.
- [API 93] ISO/IEC JTC1/SC21 N8045 rev.2
Report of the SC21 Study Group on APIs
Junho 1993.
- [ASN.1 87] ISO IS 8824/1987
Information Processing - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).
- [AVI 93a] ITU-TS Rec. F.740/1993
AudioVisual Interactive (AVI) Services: general description.
- [AVI 93b] ITU-TS Rec. T.170/1993
Audiovisual Interactive (AVI) Systems:General introduction, principles, concepts and models.
- [AVI 93c] ISO/IEC JTC1/SC29 N468/Junho 1993
The AVI Initiative: Functional Requirements Analysis and Proposed Framework for Standardization
- [Bernardo 94] Luis Bernardo.
Especificação e Sincronização de Aplicações Multimédia com Controlo Distribuído
Dissertação de Mestrado, Instituto Superior Técnico
Lisboa, Junho de 1994.
- [Blakowski+ 92] G. Blakowski, J.Hübel, U. Langrehr e M. Mühlhäuser
Tool Support for the Synchroization and Presentation of Distributed Multimedia
Computer Communications,vol 15 nº10, Dezembro 1992.

- [Brockschmidt 93] Kraig Brockschmidt
Inside OLE2
Microsoft Press, 1993.
- [Buford+ 94] John Buford and Chetan Gopal
Standardizing a Multimedia Interchange Format: A Comparison of OMFI and MHEG
Int. Con. on Multimedia Computing and Systems, 1994.
- [Carrapatoso+ 93a] E.Carrapatoso, F.Leal, C.Oliveira, J.Silva e P.Viana
PC based Architecture to Access Remote Image Databases
Int. Symp. on Fiber Optic Networks and Video Communications, Berlim, Abril 1993.
- [Carrapatoso+ 93b] E.Carrapatoso, F.Leal, C.Oliveira, J.Silva e P.Viana
PC based Multimedia Videotex
Int. Conf. on Advances in Communication & Control, Grécia, Junho 1993.
- [Casey+ 94] Thomas Casey e Joseph Fromont
MHEG, Scriptware, and Standardisation Issues
First Eurographics Conference on Multimedia Systems
Áustria, Junho 1994
- [Casey 95] Thomas Casey
Using MHEG in a Multimedia Hypermedia Training Application
Int. Conf. on Multimedia Computing and Systems
Washington, Maio 1995.
- [Colaïtis 94] Françoise Colaitis
Opening Up Multimedia Object Exchange with MHEG
IEEE Multimedia, Verão de 1994.
- [CORBA 91] Object Management Group
The Common Object Request Broker: Architecture and Specification
rev. 1.1, Dezembro de 1991.
- [Correia 94] Nuno Correia
Programação do Tempo em Sistemas Multimédia
Dissertação de Mestrado, Instituto Superior Técnico
Lisboa, Junho de 1993.
- [DSMCC 94] ISO/IEC CD 13818/1994
Digital Storage Media Control Command Extensions to MPEG-2 standard
- [Gibbs 91] Simon Gibbs
Composite Multimedia and Active Objects.
OOPSLA'91 Proceedings, Novembro de 1991.

- [Guimarães 92] Nuno Guimarães
A Concepção e Construção de Aplicações e Sistemas Interactivos
Tese de Doutoramento, Instituto Superior Técnico
Lisboa, Fevereiro de 1992.
- [Harnden+ 93] Roger Harnden and Roy Stringer
THESEUS - A Model for Global Connectivity
Systems Science: Addressing Global Issues. Proceedings, UK Systems Society Third Int. Con. , Paisley, Julho 1993
- [Harnden 94] Roger Harnden
THESEUS - A Way of Doing
Kybernetes, vol 23 (3), 1994
- [Henriques 95] Mário Henriques
Sistema de Telecomunicações de Acesso a Informação Multimédia e Hipermedia
Dissertação de Mestrado, Faculdade de Engenharia UP
Porto, Fevereiro de 1995.
- [Herman+ 94] I.Herman, G.Carson, J.Davy, D.Duce, P.Hagen, W.Hewitt, K.Kansy, B.Lurvey, R.Puk, G.Reynolds and H.Stenzel
PREMO: An ISO Standard for a Presentation Environment for Multimedia Objects.
ACM Multimedia 94 Proceedings, São Francisco, Outubro 1994.
- [HTML] IETF Draft RFC
HyperText Markup Language
- [HTTP] IETF Draft RFC
HyperText Transport Protocol
- [HyperODA 93] ISO/IEC DIS 8613-14/1993
Information Technology - Open Document Architecture (ODA) and Interchange Format - Temporal relationships and non-linear structures (HyperODA).
- [HyTime 92] ISO IS 10744/1992
Information Processing-Hypermedia/Time-based Structuring Language (HyTime).
- [IMA 92] Interactive Multimedia Association
Architecture Reference Model
Rev. 3.1, Novembro 1992
- [Kretz+ 92] Francis Kretz and Françoise Colaïtis
Standardizing Hypermedia Information Objects.
IEEE Communications Magazine, Maio 1992.

- [Leal 95] Francisco Leal
Apresentação Dinâmica de Objectos Multimédia e Hipermedia
Dissertação de Mestrado, Faculdade de Engenharia UP
Porto, Fevereiro de 1995.
- [Little+ 90a] T. Little e A. Ghafoor
Synchronization and Storage Models for Multimedia Objects
IEEE Journal on Selected Areas in Communications,
vol 8 nº 3, Abril de 1990.
- [Little+ 90b] T. Little e A. Ghafoor
*Network Considerations for Distributed Multimedia Object
Composition and Communication*
IEEE Network Magazine, Novembro de 1990.
- [Mason 92] Dean Mason
Distributed Programming.
UNIX Review, vol.11 n.1, 1992.
- [MCL 94] Think Systems
MCL: MHEG/C++ Class Library
MHEG Workshop, Rennes, Fevereiro 1994
- [MFC 93a] Visual C++ 1.5 MFC 2.0 Help
Microsoft Foundation Classes Library Reference
Microsoft Corporation, 1993.
- [MFC 93b] MFC Technical Notes
MFC/OLE IUnknown Implementation
Microsoft Corporation, 1993.
- [MFC 93c] MFC Technical Notes
MFC/OLE Automation Implementation
Microsoft Corporation, 1993.
- [Mey+ 92] V. Mey, C.Breiteneder, L.Dami, S.Gibbs e D.Tsichritzis
Visual Composition and Multimedia.
Eurographics'92 Proceedings, 1992.
- [MHEG 94a] ISO/IEC DIS 13522-1/1994
*Information Technology - Coding of Multimedia and Hypermedia
information - Part 1: MHEG object representation - Base notation
(ASN.1).*
- [MHEG 94b] ISO/IEC WD 13522-3/1994
*Information Technology - Coding of Multimedia and Hypermedia
information - Part 3: MHEG Extensions for Scripting Language
Support*

- [MHMF 94] ISO/IEC JTC1/SC18 N4526/Fevereiro 1994
Technical Report on Multimedia and Hypermedia: Model and Framework
- [MIME] IETF RFC1341
Multipurpose Internet Mail Extensions
- [MPEG 92] ISO IS 11172/1992
Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s
Moving Pictures Expert Group (MPEG-1)
- [MPEG 94] ISO/IEC DIS 13818/1994
Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media and Digital Video Transmission up to about 15 Mbit/s
Moving Pictures Expert Group (MPEG-2)
- [M&HIRS 92] ETSI ETR DTR/TE-01017/1992
Terminal Equipment (TE) - Multimedia & Hypermedia Information Retrieval Services (M&HIRS) - Investigation of candidate architectures for M&HIRS.
- [Newcomb+ 91] Steven Newcomb, Neill Kipp and Victoria Newcomb
"The HyTime" Hypermedia/Time-based Structuring Language
Communications of the ACM, Novembro 1991
- [Nierstrasz 89] Oscar Nierstrasz
A Survey of Object-Oriented Concepts
ACM Press, 1989.
- [Nierstrasz+ 92] O. Nierstrasz, S. Gibbs e D. Tsichritzis
Component-Oriented Software Development
Communications of the ACM vol.35, nº9, Setembro 1992.
- [NMF 93] Joint NM Forum/OMG Taskforce on Object Modelling
Comparaison of the OMG and ISO/CCITT Object Models
Abril 1993.
- [Normand 94] Dominique Normand
Alliance Qualité Logiciel
The MHEG Interpreter
MHEG Workshop, Rennes, Fevereiro 1994.
- [ODA 89] ISO IS 8613-1/1989
Information Processing - Text and office systems - Office Document Architecture (ODA) and interchange format.

- [ODP 94a] ISO/IEC DIS 10746-1/1994
Information Technology - Open Systems Interconnection, Data Management and Open Distributed Processing - Basic Reference Model of Open Distributed Processing - Overview and Guide to use.
- [ODP 94b] ISO/IEC DIS 10746-3/1994
Information Technology - Open Systems Interconnection, Data Management and Open Distributed Processing - Basic Reference Model of Open Distributed Processing - Prescriptive Model.
- [OLE 93] Microsoft OLE Design Team
OLE 2.01 Design Specification
Microsoft Corporation, 1993.
- [OMG 90] Object Management Group
Object Management Architecture Guide
versão 1.0, Novembro 1990.
- [OMHEGA 94] OMHEGA - Open MHEG Architecture ESPRIT project
MHEG Related Activities
draft paper, 1994.
- [Pereira 94] Paulo Pereira
Um Modelo para o Desenvolvimento de Aplicações Multimédia Interactivas Distribuídas
Dissertação de Mestrado, Instituto Superior Técnico
Lisboa, Junho de 1994.
- [Picquet+ 93] Philippe Picquet and Florence Bertrand
MHEG Toolkit, CCETT Experiments.
MHEG 93/627, Novembro de 1993.
- [PREMO 94a] ISO/IEC CD 14478-1/1994
Information Processing Systems - Computer Graphics and Image Processing - Presentation Environments for Multimedia Objects (PREMO). Part 1: Fundamentals of PREMO.
- [PREMO 94b] ISO/IEC CD 14478-2/1994
Information Processing Systems - Computer Graphics and Image Processing - Presentation Environments for Multimedia Objects (PREMO). Part 2: Foundation Component.
- [ScriptX 93] Kaleida Labs, Inc.
Kaleida, ScriptX, and the Multimedia Market
1993
- [SGML 86] ISO IS 8879/1986
Information Processing - Text and Office Systems Standard Generalized Markup Language (SGML).

- [Shirley 92] John Shirley
Guide to Writing DCE Applications
O'Reilly and Associates, 1992.
- [Silva 95] Jaime Silva
Interpretador Multimédia com Capacidade Fotográfica para Sistemas de Telecomunicações
Dissertação de Mestrado, Faculdade de Engenharia UP
Porto, Fevereiro de 1995.
- [Stringer 92] Roy Stringer
Theseus: A project at Liverpool Polytechnic to develop a Hypermedia Library for Open and Flexible Learning.
Int.Fed.of Lib.Ass. vol 18 (3), 1992
- [URL] IETF RFC 1630
Uniform Resource Locators
- [Veríssimo 94] Paulo Veríssimo
Sumário da Lição de Síntese sobre Comunicação e Computação em Sistemas Distribuídos
1994.
- [Webb 93] Jeff Webb
OLE 2.0 Automation
Abril 1993.



FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BIBLIOTECA



000006442