Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e Engenharia da Computação

Departamento de Engenharia de Computação e Automação Industrial

Gerenciador de acesso a dados multimídia

Autor: **Cláudia de Andrade Tambascia**

Orientador: **Prof. Dr. Ivan Luiz Marques Ricarte**

Dissertação submetida à Faculdade de Engenharia Elétrica e Engenharia da Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Dezembro 1997

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

T151g

Tambascia, Cláudia de Andrade
Gerenciador de acesso a dados multimídia. / Cláudia de Andrade Tambascia.--Campinas, SP: [s.n.], 1997.

Orientador: Ivan Luiz Marques Ricarte.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas multimídia. 2. Processamento eletrônico de dados - Processamento distribuído. 3. Banco de dados orientado a objetos. I. Ricarte, Ivan Luiz-Marques. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Uma das maiores dificuldades no processamento distribuído de informação envolvendo objetos multimídia é o tratamento de dados envolvendo dimensões temporais, tais como áudio e vídeo. Há uma diversidade de formatos de representação deste tipo de dados e esforços de padronização são fundamentais na integração destas aplicações.

A gerência de acesso a um repositório especializado para este tipo de informação multimídia apresenta diversas dificuldades relacionadas, principalmente, aos aspectos de qualidade de serviços (QoS) e à adequação de taxas de recuperação e exibição.

O principal objetivo deste trabalho foi o desenvolvimento de um gerenciador que possibilitasse o acesso e apresentação de dados multimídia e que permitisse a integração com banco de dados orientados a objetos e repositórios de dados. Esse sistema permite que o usuário possa escolher apresentações previamente definidas e armazenadas em um banco de dados e a partir desta escolha, efetuar a apresentação.

Os resultados obtidos incluem a implementação deste sistema em uma arquitetura local e distribuída, além de um mapeamento de suas principais funcionalidades utilizando a filosofia do padrão MHEG, o *kit* de desenvolvimento Java e a ferramenta *Java Media Player*.

Palavras-chave: MHEG, gerenciador de acesso, dados multimídia, objetos multimídia.

Abstract

One of the major difficulties in distributed information systems with multimedia objects is handling time-constrained data such as audio and video. There are many representation formats for this kind of data and creating a standard is fundamental for integration of these applications.

The access management of a specialized repository for multimedia information shows many difficulties related to aspects of quality of services (QoS) and recovery and exhibition at adequate rates.

The main goal of this work was the development of a manager that enables access and presentation of multimedia data and allows integration of object-oriented data base and data repository. This system allows users to choose presentations already defined and stored in a data base and exhibit it.

The results achieved are the implementation of this system in local and distributed architecture and the mapping of the main functionalities using the MHEG standard philosophy, Java development kit and Java media player tools.

Keywords: MHEG, access management, multimedia data, multimedia objects.

Agradecimentos

À Deus por iluminar meu caminho.

Ao meu orientador, pela confiança, dedicação e presença constante em todo o desenvolvimento deste trabalho. Obrigado por tudo.

Aos meus pais e irmãos, por todo amor, todo carinho e principalmente por acreditarem em mim e estarem sempre ao meu lado.

À Miriam, pela grande amizade, carinho e força durante toda esta jornada.

Aos meus pais de coração, Dino e Rosana, e aos meus irmãos de coração, Letícia e Fabiano, pelo apoio, amizade e amor.

Ao Ismael e à Lucimara, por todos os trabalhos juntos e principalmente pela amizade e companheirismo no início do mestrado.

À TUTA, em especial ao Paulinho, Morte e Flávia, por todos os momentos maravilhosos juntos, pela convivência diária e apoio constante.

Ao Naur, André, Raquel e todos do grupo multimídia, por todas as interações e trocas de experiência.

Ao André, Paulinho, Rodrigo, Adela, Silvia, Luciano, Plínio, Affonso e todos os meus amigos do LCA, por todos os momentos compartilhados, pelo companheirismo e pela forte amizade.

À Capes, pelo apoio financeiro.

Aos meus pais, Cláudio e Wanda,
à minha irmã Luciana,
ao meu irmão Rafael
e aos meus avós.

Capítulo 1

Introdução

"É melhor tentar e falhar, que preocupar-se e ver a vida passar. É melhor tentar, ainda que em vão, que sentar-se fazendo nada até o final. Eu prefiro na chuva caminhar, que em dias tristes em casa me esconder. Prefiro ser feliz, embora louco, que em conformidade viver."

(Martin Luther King)

Recentemente, vem acontecendo um crescimento muito grande de aplicações multimídia em vários domínios, tais como treinamento e educação (conhecido como tele-treinamento), simulação e jogos, publicações eletrônicas e livros eletrônicos, informações públicas, trabalho cooperativo de multimídia suportada por computadores em aplicações de televisão interativa. Nestes vários domínios houve uma necessidade de se aperfeiçoar as ferramentas para ser possível trabalhar com textos, imagens e sons. O desenvolvimento destes sistemas passou a ser baseado no suporte à informação multimídia nas áreas relacionadas a plataformas de armazenamento e comunicação.

Utilizando estas novas facilidades, as aplicações multimída passaram a ser desenvolvidas para executarem em plataformas heterogêneas, interconectadas para oferecerem serviços multimídia.

Atualmente o mercado de multimídia está se caracterizando pelo grande número de aplicações, cada qual utilizando sua própria tecnologia para áudio e vídeo digital, linguagem *script*, codificadores e decodificadores, protocolos de comunicação, etc.

Não existe ainda nenhuma forma padrão de compor uma apresentação multimídia interativa complexa, armazená-la em alguma estrutura, enviá-la através da rede e apresentá-la para um usuário remoto. Em vista disso, no mercado de redes, a definição de protocolos padrões criaram um grande potencial para troca de informações entre diversos computadores distribuídos pelo mundo. A idéia é que a criação de padrões para apresentações multimídia venha a ter o mesmo impacto.

O fato de existir usuários multimídia utilizando as aplicações em diversos sistemas aumentou a necessidade em se definir um padrão universal para manipulação de dados multimídia e implementá-los na maioria das plataformas. Um grande problema é com relação ao limitado uso de padrões internacionais para representação de conteúdos multimídia e relacionamentos temporais entre esses elementos.

Um outro problema é com relação à gerência do acesso a um repositório especializado para o tipo de informação multimídia, principalmente relacionado aos aspectos de qualidade de serviços (QoS) e à adequação de taxas de recuperação e exibição (sincronização). Por isso a necessidade em se desenvolver um gerenciador de acesso a dados multimídia que possa agilizar o processo de recuperação de forma a não prejudicar o desempenho da aplicação.

É com base nas tendências atuais de manipulação de dados multimídia que este projeto foi desenvolvido, tendo como principal objetivo a criação de um gerenciador de acesso a dados multimídia, independente de plataforma e com a capacidade de compartilhar dados em diversos formatos, sem a necessidade de se preocupar com a forma como isso acontece, além de possibilitar uma interação da aplicação com banco de dados e repositórios especializados.

1.1 Definição do problema

Com a tendência atual das aplicações distribuídas em rede e da comunicação através da Internet, os sistemas passaram a ter a necessidade de serem desenvolvidos utilizando novas tecnologias, para que pudessem ser executados em plataformas heterogêneas sem prejudicar o desempenho da aplicação final.

A idéia deste projeto partiu desta necessidade em fornecer uma ferramenta que

possibilitasse a gerência dos dados presentes em apresentações multimídia independentes de plataforma e compartilhadas através da rede. Estas apresentações são basicamente formadas por objetos com formato de difícil manipulação e armazenamento tais como áudio, vídeo, hipertextos e imagens. Esses dados são armazenados em repositórios especializados e suas características são disponibilizadas em um banco de dados orientado a objetos. Essas características são fornecidas ao usuário para que este possa decidir que tipo de apresentação deve ser executada.

Seguindo a filosofia tradicional da arquitetura cliente-servidor, as aplicações são requisitadas pelos clientes e são fornecidas por um respectivo servidor. Além disso, os objetos da aplicação podem estar distribuídos, sendo necessário também a criação de servidores distribuídos para atender este aspecto. Neste caso é importante salientar que no contexto do gerenciador de acesso a dados multimídia, as informações devem ser transmitidas para o cliente antes do início da apresentação, para evitar problemas com relação a possíveis atrasos na transferência dos dados.

A implementação utilizando Java (como linguagem de desenvolvimento e não de criação de *applets*) permitiu o desenvolvimento deste tipo de aplicação, além de permitir a execução em diferentes plataformas. O fato de *browsers* serem limitados com relação a restrições de acesso do cliente tornaria o desenvolvimento da aplicação baseado em *applets* difícil de ser executada, podendo comprometer em alguns casos o desempenho do sistema. Para o desenvolvimento do projeto usando a filosofia de uma arquitetura cliente-servidor, utiliza-se uma implementação da arquitetura CORBA (*OrbixWeb*).

1.2 Objetivo

O desenvolvimento, seguindo a filosofia do padrão MHEG, tem como objetivo principal receber e apresentar objetos multimídia obtidos de um repositório, permitindo assim que dados sejam compartilhados entre aplicações. Tais aplicações envolvem a apresentação de imagens estáticas, imagem em movimento, sons (de forma sincronizada ou não) e animações, através da apresentação de sequências de imagens estáticas. É com base neste conceito que trabalhamos no desenvolvimento de um gerenciador de acesso, responsável pela negociação destes tipos de objetos.

Esse gerenciador, por sua vez, está ligado a um banco de dados orientado a objetos, responsável pelo armazenamento das características dos componentes de cada aplicação. Para essa integração foi necessário a definição de uma interface que possibilitasse a comunicação entre os dois. Além do banco de dados, a aplicação esta fortemente ligada a um repositório ou ao sistema de arquivos para poder recuperar as informações a serem apresentadas da forma mais eficiente possível.

A interface principal do sistema permite que o usuário possa escolher as aplicações a serem apresentadas, além de ter acesso aos componentes de cada uma delas, com a possibilidade de fazer alterações de acordo com suas necessidades, antes ou durante a apresentação.

1.3 Descrição dos capítulos

Esta dissertação apresenta um estudo do padrão MHEG, utilizado na representação de dados multimídia, ao mesmo tempo que propõe a implementação de um sistema gerenciador de acesso a dados multimídia, seguindo a filosofia de um engenho MHEG.

O capítulo 1 apresenta o problema e define seus objetivos principais. O capítulo 2 define conceitos relacionados a sistemas multimídia distribuídos, situando o projeto no contexto relacionado a multimídia e hipermídia. O capítulo 3 apresenta o padrão MHEG, ressaltando conceitos importantes utilizados no decorrer do projeto, dando ênfase à definição de um engenho MHEG. O capítulo 4 especifica a estrutura do sistema, sua descrição funcional, apresentando a visão conceitual e o mapeamento das funcionalidades do sistema de acordo com as ferramentas a serem utilizadas. São levantados também aspectos de implementação e interface do sistema. O capítulo 5 apresenta uma descrição das funcionalidades, cenários de uso, relacionamentos da aplicação proposta com o padrão MHEG, detalhes de implementação e definição das interfaces, além de uma breve apresentação da ferramenta *Java Media Player*. São apresentadas também as funcionalidades do sistema e seu mapeamento utilizando o compilador Java, o padrão MHEG e o *Java Media Player*. O capítulo 6 apresenta conclusões sobre o trabalho, resultados obtidos e sugestões para trabalhos futuros. O apêndice A fornece os diagramas de fluxos de dados, especificados durante a fase de análise do projeto do gerenciador de acesso a dados multimídia. O apêndice B fornece as tabelas completas com o mapeamento das funcionalidades do sistema utilizando

a especificação MHEG, o compilador JDK 1.1 e o *Java Media Player*.

Capítulo 2

Sistemas Multimídia Distribuídos

"Não se preocupe em entender. Viver ultrapassa todo entendimento."
(Clarice Lispector)

Multimídia pode ser entendida como qualquer combinação de textos, sons e imagens organizados de forma a proporcionar ao usuário uma percepção mais completa de alguma informação que se deseja apresentar.

Entende-se por multimídia interativa a forma pela qual o usuário pode controlar o fluxo e até mesmo o tipo de informação que lhe é entregue, isto é, o usuário participa interativamente dos acontecimentos que lhe são percebidos através do terminal multimídia.

O termo hipermídia, por sua vez, é utilizado para descrever uma estrutura de multimídia interativa formando uma espécie de corpo de informação, possuindo ligações cruzadas entre os diversos elementos que o compõe, criando-se assim uma estrutura complexa, não linear, que não pode ser descrita como um fluxo sequencial de elementos de informação [8].

A respeito do histórico na utilização dos vários tipos de mídias de informação, pode-se observar que recentemente têm ocorrido um crescente e significativo aumento na diversidade de mídias utilizadas para a transmissão de informação, como apresentado na Figura 2.1.

Com relação à sua estrutura, os sistemas multimídia distribuídos são considerados complexos, devido ao grande número dos elementos que são utilizados, incluindo: facilidades

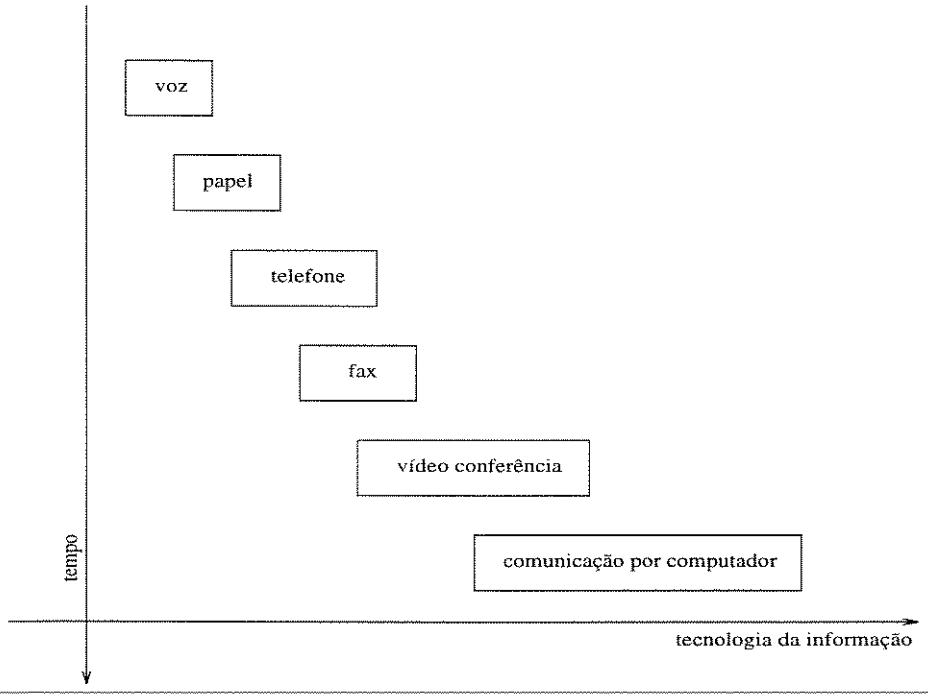


Figura 2.1: Tendências na utilização de mídias de informação.

para gerência de objetos distribuídos, arquitetura de documentos multimídia, formatos para intercâmbio de documentos e apresentação multimídia, linguagens de *script*, formatos de mídia, ferramentas de aplicação, serviços do sistema operacional, protocolos e arquiteturas de rede. De acordo com o número de consumidores potenciais da informação e de usuários conectados, tem-se um aumento no valor da informação e da comunicação multimídia, tendo-se ainda mais a necessidade de que os sistemas multimídia suportem funções distribuídas em larga escala.

Para isso, foi necessário o desenvolvimento de mecanismos para armazenamento e busca de arquivos multimídia, permitindo a execução de aplicações como teleconferências, distribuição de anúncios e notícias, entretenimentos, etc. Foi fundamental considerar as diferenças entre as mídias existentes, como no caso da relação de texto com mídias do tipo áudio e vídeo, no que diz respeito principalmente aos requisitos de armazenamento.

Uma das características que devem ser levadas em consideração é a existência de múltiplos fluxos de dados, onde um dado objeto pode ser composto de um ou mais fluxos de

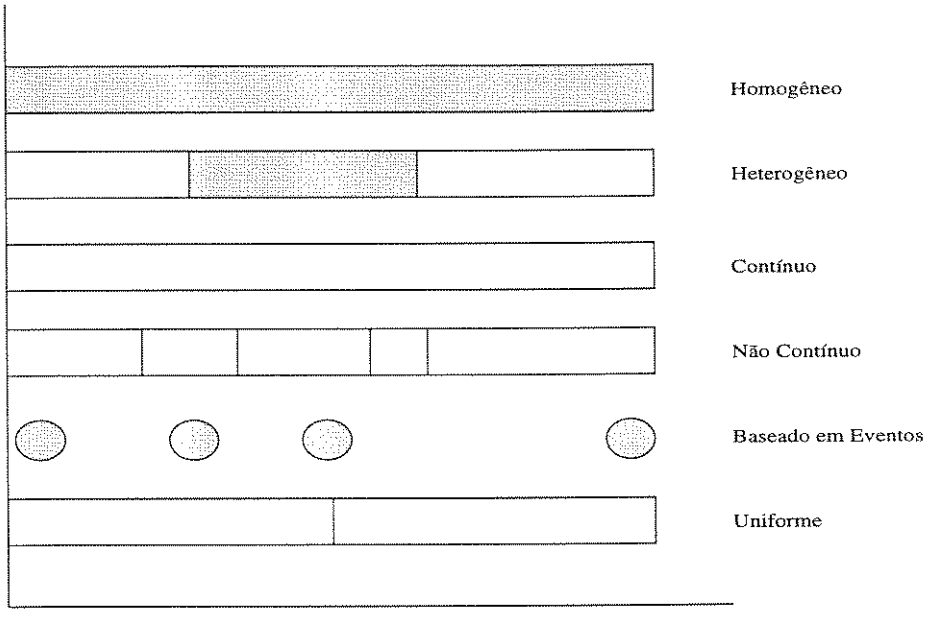


Figura 2.2: Tipos de fluxos de dados.

áudio, vídeo e texto que são encaminhados para os *buffers* dos seus respectivos dispositivos de saída. No caso do armazenamento deste objetos em um mesmo fluxo multiplexado, seria necessário o desenvolvimento de mecanismos que fossem capazes de fazer a combinação dos fluxos durante o processo de gravação e de separação dos mesmos durante a recuperação. Estes mecanismos são relativamente trabalhosos, devido à complexidade e às diferenças existentes entre as mídias. Além disso, é necessário que o sistema de arquivos considere os relacionamentos temporais, para garantir a sincronização durante a recuperação e reprodução.

De acordo com a forma em que os elementos são armazenados, podemos ter diferentes tipos de fluxos de dados multimídia (Figura 2.2):

- homogêneos: possuem somente um único elemento de mídia, com o mesmo formato.
- heterogêneos: possuem elementos e formatos variados.
- contínuos: apresentam um único elemento.
- não contínuos: apresentam espaços vazios entre os dados.
- baseados em eventos: um tipo especial de fluxos não contínuos, com a diferença que neste caso, a duração da apresentação do dado é muito pequena.

- uniformes: apresentam várias mídias com duração e tempo constantes.

O projeto de sistemas de arquivos feito para acomodar os requisitos dos dados multimídia passa a ser um problema relacionado aos sistemas distribuídos, devido ao fato dos objetos multimídia poderem estar presentes em diversas máquinas. Entretanto, estes sistemas passam a ter o requisito especial do processamento e transporte de mídias contínuas.

Durante o atendimento de um serviço solicitado pela aplicação cliente, o gerente do recurso responsável pela execução de determinada tarefa deve monitorar o uso pela aplicação de um modo que sobrecargas inesperadas não provoquem problemas nos serviços dos outros clientes, possibilitando assim que outros serviços possam ser requisitados sem problemas de desempenho. A natureza temporal do vídeo e do áudio digitais, denominados de dados isócronos, requer que o atraso (*delay*) e a variação no atraso (*jitter*) sejam rigidamente limitados entre o ponto de geração e/ou recuperação e o ponto de apresentação. Este requisito é conhecido como sincronização intramídia. Caso vários fluxos contínuos sejam apresentados em paralelo, potencialmente a partir de pontos distintos de geração e/ou recuperação, a relação de temporização relativa aos fluxos denomina-se sincronização intermídia. Ambos os tipos de sincronização requerem uma gerência coordenada dos recursos de modo a garantir que a sincronização fim-a-fim possa ser atendida.

Através dos tempos, é possível ainda observar uma tendência contínua no desenvolvimento de mídias cada vez mais rica e das mais variadas formas. Há alguns anos atrás as pessoas limitavam-se a formas de transmissão de informação simples, tais como a voz e o papel. Com o passar dos tempos, passamos a presenciar a introdução de uma grande variedade de tipos de mídia como o telefone e mídias de forma visual. Recentemente, essa tendência tem se acelerado, existindo hoje uma grande variedade de tipos de mídia disponíveis para transportar informações. Com todo esse desenvolvimento, criam-se novas oportunidades para outros avanços em áreas como a educação, o comércio e a saúde.

O usuário necessita de suporte para obter a informação certa destinada à pessoa certa no momento certo. Apesar da dificuldade, isto pode ser obtido através de sistemas manuais, motivando então a automação do processo de gerência da informação. De forma mais específica, a tecnologia de informação é vista como a combinação da tecnologia computacional com a infra-estrutura de comunicação. O objetivo último desta tecnologia é gerenciar a informação para o usuário final. A tendência mais notável na tecnologia de

informação, ainda do ponto de vista dos sistemas, é a integração de mídias, como áudio, vídeo e textos.

2.1 Mídia digital

Os sistemas computacionais lidavam exclusivamente com cálculos numéricos. O processamento de texto tornou-se um aspecto importante para os projetistas de computadores. Tecnologias de comunicação também foram desenvolvidas para suportar a transmissão de textos e de dados numéricos. Dentro deste contexto temos a mídia digital.

Os dados multimídia podem ser definidos como uma composição dos dados "tradicionais" e de novos tipos de dados, tais como, áudio, vídeo, gráficos, imagens (figuras estáticas), fala, música, animação e qualquer possível combinação, passando a incluir a dependência do tempo na sua manipulação e gerência. Isso pode ser facilmente explicado pelo fato destes tipos de mídia, para que possam transmitir de forma completa a informação que carregam, deverem ter seus quadros (vídeo) ou amostras (áudio) de informação reproduzidos a taxas constantes. Entre os vários tipos de mídia, podemos destacar:

- Textos: considerado um conjunto de palavras usados para exprimir idéias ou pensamentos. Como o computador só reconhece dois estados lógicos, a representação de caracteres exige o desenvolvimento de códigos sequenciais, normalmente como uma sequência fixa de bits para se definir estes códigos.
- Gráficos e imagens: os gráficos incluem todos os conceitos que permitem gerar desenhos e outras imagens baseadas em descrições formais, programas ou estruturas de dados, principalmente pelo fato da principal forma de interação entre o homem e a máquina ser a tela. Com relação ao estudo do uso de imagens em multimídia, dois tipos de apresentações são levadas em consideração, imagens estáticas e imagens animadas. Um sistema de informação multimídia tem que suportar a funcionalidade para importar e manipular dados digitais equivalentes a este tipo de mídia, ou seja, prover o suporte a operações de manipulação básicas como recorte de imagens para uma nova composição, escalonamento, correção cromática e composição de diversas fontes de imagens.
- Áudio e voz: uma das mídias mais complexas é o áudio, cujo sinal se apresenta como uma onda analógica que pode assumir um número infinito de valores dentro de uma

faixa contínua de variação. O tipo de mídia fala passou a ser reconhecido no contexto de sistemas multimídia, podendo ser armazenado como áudio, mas apresentando algumas características únicas associadas ao processamento de linguagem natural, que permite o reconhecimento de palavras e a identificação de pessoas.

- Mídia gerada: engloba principalmente animação e música gerada por computador, podendo também ser vistos como um tipo especial de mídia contínua.
- Vídeo: pode ser considerado como uma sequência de imagens em movimento, criadas a partir da justaposição de imagens separadas por um determinado intervalo de tempo, combinando as características das mídias imagem e áudio sendo, com isso, altamente temporal.

A utilidade da integração de várias mídias não possui contestação atualmente, pois seus benefícios são notórios. Assim, os recursos para interação do homem com a informação multimídia encontra-se em um processo acelerado, que pode ser evidenciado pelo crescimento no uso da tecnologia digital para processamento de áudio e os avanços na computação gráfica.

Dois aspectos principais destacam-se nesta discussão: variedade dos tipos de mídia e integração destas mídias, conduzindo a uma definição de trabalho para multimídia, sendo esta a soma de variedade e integração. É necessário para este tipo de sistema suportar uma variedade de tipos de mídia, podendo isto ser tão simples como suportar texto e gráficos, ou bastante complexo, como suportar animação, áudio e vídeo.

Entretanto, isto ainda não é suficiente para um ambiente multimídia, sistema no qual as várias fontes de tipos de mídia são integradas em um único contexto de sistema. Assim, uma apresentação multimídia pode ser definida como aquela que permite aos usuários finais compartilhar, comunicar e processar uma variedade de formas de informação de uma maneira integrada. A multimídia é composta de vários componentes e várias mídias integradas, existentes nos sistemas computacionais e transmitidas através da infra-estrutura existente de comunicação, mas que expressam objetos do mundo real.

Avanços na tecnologia das estações de trabalho e o desenvolvimento de redes de computadores de alta velocidade e de alto desempenho têm viabilizado as recentes pesquisas na área de computação multimídia. Os resultados dessas pesquisas levaram às mais diversas aplicações, explorando a habilidade de integrar, armazenar e transmitir informação, tais

como voz, vídeo, áudio de alta qualidade e gráficos. Aplicações bem sucedidas incluem:

- vídeo-conferência em tempo real, onde vários usuários podem interagir, mesmo estando em pontos distantes do planeta;
- trabalho cooperativo auxiliado por computador, pelo qual um projeto pode ser realizado por uma equipe que esteja ou não no mesmo ambiente físico de trabalho;
- correio eletrônico multimídia, no qual a correspondência eletrônica entre duas ou mais pessoas não será somente feita através de texto, mas também através de sons e imagens;
- ensino mediado por computador, que possibilita um mesmo instrutor ensinar várias pessoas em diferentes localidades;
- vídeo interativo, como o sistema por demanda *pay-per-view*, onde o assinante de uma estação multimídia poderá selecionar o filme que deseja assistir e locá-lo de casa, sendo o mesmo transmitido pela rede de comunicação;
- entretenimento, incluíndo jogos interativos, tridimensionais, diversões com realidade virtual, holografias, etc;
- sistemas médicos, onde especialistas em determinadas áreas médicas podem consultar ou operar um determinado paciente em uma localidade afastada, com o auxílio de médicos não especialistas e de um sistema de transmissão de informações multimídia.

Apesar do sucesso dessas aplicações, existem ainda muitas limitações tecnológicas que devem ser vencidas antes que seus benefícios possam ser plenamente realizados. Uma dessas limitações reside no suporte ao armazenamento, gerência e integração da mídia contínua junto com os outros tipos de mídia, o que fornece uma motivação para os trabalhos a serem realizados na área de tratamento da informação multimídia. Além disso, o crescente surgimento de diversas aplicações multimídia, que passam a trabalhar interativamente entre si, exige que dados multimídia sejam tratados e gerenciados de forma integrada, como é possível fazer com dados convencionais.

2.2 Transmissão da mídia digital

Computação multimídia requer integração de textos, imagens, áudio e vídeo em uma variedade de ambientes. Genericamente, cada integração é relacionada a restrições

temporais sobre estes dados, que por sua vez podem ser fortemente dependentes, como áudio e vídeo em um filme, e podem com isso requerer uma apresentação ordenada no tempo durante sua utilização.

A sincronização pode ser aplicada à apresentação de fluxos de dados sequenciais ou concorrentes, e também a eventos externos gerados pelos usuários. O problema da sincronização multimídia para mídias dependentes de tempo pode ser descrita em três níveis: nível físico, nível de servidor e nível de interface com o usuário [4].

A sincronização é a coordenação ordenada de eventos no tempo. Vários mecanismos e formalismos de sincronização têm sido desenvolvidos, variando desde técnicas de baixo nível, do ponto de vista de *hardware*, até abstrações do ponto de vista de linguagens de programação concorrentes. Um requisito fundamental para os sistemas multimídia é suportar sincronização intermídia e intramídia, ou seja, a sincronização entre mídias independentes exibidas em paralelo e a sincronização entre fluxos relacionados em mídias compostas, respectivamente [7]. A coordenação dos vários gerenciadores dos recursos para fim de obtenção de sincronização fim-a-fim denomina-se orquestração, onde parâmetros de qualidade de serviços são considerados peças básicas [8]. Tanto a sincronização quanto a transferência em tempo real dependem, basicamente de fatores como:

- faixa de compressão dos vários tipos de dados;
- políticas otimizadas de alocação de *buffers* e distribuição dos dados através da rede, de uma forma que não prejudiquem o desempenho do sistema;
- algoritmos de escalonamento de páginas, para servir uma nova requisição;
- dispositivos de armazenamento secundários, para otimizar o armazenamento dos dados, garantindo uma recuperação mais rápida;
- alocação de banda passante no canal de comunicação, suficiente para suportar o volume de dados a ser transmitido [1].

Sistemas que utilizam dados do tipo mídia contínua não requerem novas primitivas de sincronização, mas devem levar em conta dois aspectos importantes das aplicações multimídia: prazos de tempo real dos eventos de sincronização e falhas na sincronização, que podem ser tratadas através de técnicas, como repetição ou salto de quadros, de modo que a aplicação possa continuar sua execução sem atrapalhar os requisitos de tempo. A

organização em camadas do projeto de sistemas multimídia faz com que a granularidade dos eventos de sincronização seja, em geral, maior no nível de aplicação, tornando-se mais detalhada nos níveis inferiores do sistema, por exemplo, uma aplicação pode definir pontos de sincronização no nível do quadro, enquanto que a rede na qual o vídeo será transmitido terá a responsabilidade da sincronização no nível de pacotes ou células. Consequentemente, a representação dos requisitos de sincronização pode variar de camada para camada, assim como entre diferentes aplicações.

De modo a cumprir os requisitos de escalonamento no caso dos fluxos de mídia contínua, cada subsistema deve fornecer um atraso máximo com alguma probabilidade. Além disso, visando também restringir os requisitos de alocação de *buffers*, esta variação deve ser limitada. Elementos de mídia podem necessitar de sincronização, como por exemplo, a apresentação de um vídeo e um áudio associado. Neste caso os elementos de mídia que necessitam ser sincronizados devem estar presentes nos respectivos *buffers* dos dispositivos de saída, de modo a cumprir o prazo da apresentação.

Em geral, cada mídia possui requisitos próprios de escalonamento devido ao atraso específico daquela mídia, das características do atraso em função das taxas de amostragem e da resolução da mídia. A variação no atraso entre elementos correspondentes de uma ou mais mídias sincronizadas é conhecido como *skew*, que acontece quando erros ou atrasos em um fluxo impedem que o sistema atenda o escalonamento de apresentação para aquele fluxo. Quando o *skew* excede o nível tolerável para a aplicação, o sistema deve ser re-sincronizado, sendo então necessário o uso de técnicas (por exemplo, salto de amostras de áudio) que permitem ao fluxo em atraso alcançar o outro mais rapidamente, ou mesmo, atrasar este último, para que os requisitos da aplicação sejam atendidos.

A orquestração é tida como um elemento importante na arquitetura dos sistemas multimídia distribuídos, podendo ser vista como uma camada presente entre as aplicações e as funções tradicionais dos sistemas e da rede, como ilustrado na figura 2.3. Neste caso, cada gerente de recurso inclui uma função de escalonamento que ordena as requisições de serviço com o objetivo de cumprir os limites de desempenho requeridos, por exemplo, um sistema de arquivo de mídia contínua escalona as operações de busca ao sistema de armazenamento, enquanto que a camada de rede escalona o tráfego para a camada de transporte.

Uma aplicação necessita da operação coordenada destas funções de escalonamento

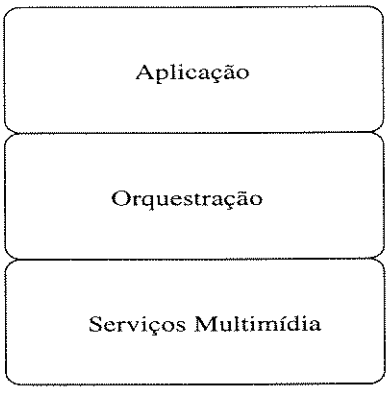


Figura 2.3: Camada de orquestração.

no caso da necessidade de que limites de desempenho fim-a-fim sejam cumpridos. Uma abordagem para a coordenação do escalonamento dos recursos no caso de sistemas multimídia é adicionar uma camada entre a aplicação e os gerentes de recursos. Esta camada terá a finalidade de implementar uma ação coordenada e integrada denominada orquestração ou meta-escalonamento. Além disso, irá fornecer funções de escalonamento e sincronização globais com a finalidade de atender os requisitos da aplicação.

2.3 Aspectos de armazenamento e de acesso da mídia digital

Usualmente um sistema de informação possue dados que podem ser manipulados de várias formas: sequencial ou aleatóriamente, em grupo ou isoladamente, podendo ser filtrados através de, por exemplo, operações de manipulação da teoria de conjuntos, muitas vezes organizados em um banco de dados que provê um sistema de consulta amigável com o usuário. O que se espera de um sistema de informação multimídia, que desempenha papel similar ao banco de dados convencional, é que a consulta a um objeto multimídia seja transparente ao usuário e independente de sua estrutura de armazenagem. É exatamente essa necessidade de manipulação de mídias de forma intuitiva pelo usuário que motiva estudos e pesquisas na área de banco de dados orientados a multimídia.

Mídias contínuas têm como característica principal o fato de possuirem uma dinâmica natural expressada por uma frequência de captura e apresentação, que se não for obedecida, reduz a qualidade da informação associada. A integração destes tipos de mídia em

ambiente computacional requer que a plataforma de suporte forneça meios que permitam, por exemplo, a reserva antecipada de recursos, implicando novas abstrações relacionadas a comunicação e armazenamento das informações.

Tipos de mídias contínuas tais como áudio e vídeo podem ser representados tanto na forma analógica quanto na forma digital, sendo que tradicionalmente tem-se usado representação na forma analógica através de um conjunto contínuo de valores, ao invés de utilizar valores discretos, como no caso da representação digital. Os primeiros sistemas multimídia manipulavam representações analógicas de dados de mídia contínua, onde em alguns casos, os projetistas desenvolviam recursos de armazenamento de vídeo e de voz baseados em dispositivos próprios de armazenamento controlado por computador. A saída destes dispositivos de armazenamento era transmitida na forma analógica e apresentados através dos dispositivos. O uso desta tecnologia permitiu o desenvolvimento de vários trabalhos pioneiros com a finalidade de evidenciar os benefícios potenciais dos sistemas multimídia.

Recentemente, entretanto, tem havido um movimento em direção à representação digital de todas as formas de mídia, movimento particularmente forte nas indústrias da música e do cinema. Este movimento, do ponto de vista da computação multimídia, tinha uma razão muito forte em existir, pois iria permitir a integração total dos tipos de mídias existentes, além de facilitar uma integração de novos tipos de mídias que possam surgir.

Um gerenciador de armazenamento multimídia deve oferecer capacidade de expansão e transparência de armazenamento, operar em ambientes de discos magnéticos e de discos ópticos regraváveis, discos ópticos de escrita única, discos ópticos somente de leitura, CD-WOs (*write-once*), e discos CD-reescrevíveis (definidos pelo novo padrão internacional da Philips e Sony), e ser capaz de facilmente acomodar dispositivos de armazenamento com novas características.

Além disso, deve ser capaz de gerenciar (isto é, inserir, apagar, atualizar, etc.) informações nestes dispositivos, garantir que estes obedeçam aos padrões internacionais de interconexão, tão bem quanto os padrões que descrevem a alocação de informação nestes dispositivos. As operações básicas do gerenciador de armazenamento devem ser transladadas transparentemente em operações apropriadas para cada diferente tipo de dispositivo. Neste caso é requerido um modelo generalizado formal de armazenamento, que seja capaz de capturar as características de tipos diferentes de dispositivos e que mapeie um conjunto

generalizado de operações em um conjunto correspondente para um tipo de dispositivo particular.

Outro ponto é que os dados multimídia requerem uma capacidade muito grande de espaço para armazenamento (volume), o que aumenta a dificuldade do projeto de um sistema de armazenamento de dados multimídia. Em alguns ambientes e para algumas aplicações, faz-se necessário o uso de armazenamento terciário e de hierarquias de armazenamento multiníveis. Para tanto, a integração uniforme e efetiva dos vários níveis de armazenamento para que se atenda aos requisitos das aplicações multimídia é extremamente importante. Para se otimizar o desempenho na recuperação, o sistema deve fazer um esforço para armazenar as partes de um objeto grande (*Binary Large Object — BLOB*) multimídia (tais como imagem, vídeo, etc) de forma contígua, seguindo os requisitos destes tipos de objetos. O BLOB funciona como uma região de memória reservada para objetos que requerem grande espaço para armazenamento.

No entanto, pode ser difícil e lento para o sistema encontrar espaço contíguo no dispositivo secundário para armazenar esses objetos no tempo de inserção. Então, para ter um bom tempo de resposta para escritas, o sistema deve tentar escrever as partes do objeto longo da forma mais eficiente possível. Deve-se, então, tentar otimizar as leituras em *background* agrupando as partes do objeto grande e dos outros objetos relacionados que estiverem próximos. Tal projeto, que envolve tanto otimização de leitura quanto de escrita, tem muitas variações dependendo da importância de cada um dos dois tipos de otimizações.

O sistema deve suportar recuperação eficiente de partes de objetos grandes, o que implica agrupamento apropriado de dados e indexação dentro de um objeto multimídia grande. Em termos de compressão, a recuperação de imagens comprimidas é mais difícil, bem como a reprodução de um ponto arbitrário para vídeos e áudios de longa duração, principalmente quando métodos de compressão interquadros foram utilizados.

Neste caso, um suporte do sistema é necessário para garantir desempenho. Em alguns ambientes, BLOB's podem ter múltiplas representações para satisfazer as várias necessidades dos usuários ou as características de cada dispositivo de apresentação. Por exemplo, a existência de telas de alta-resolução e de baixa resolução nas estações de trabalho cliente podem implicar uma qualidade diferente de imagem a ser enviada a diferentes clientes. Quando e como diferentes representações devem ser armazenadas será uma das

características de otimização do sistema. É óbvio que a recuperação de objetos grandes pode resultar em atrasos devido a tempos de transferência envolvidos. O sistema deve tentar reduzir os atrasos explorando paralelismo dos dispositivos secundários. A natureza diversa dos diferentes tipos de dados multimídia implica diferentes algoritmos de colocação de dados serem mais lucrativos para diferentes tipos de mídia.

Portanto, pude observar o quanto é importante projetar e implementar ferramentas que gerenciam os dados em estruturas hierárquicas que garantam tempos de respostas rápidos. Isso implica movimento dos dados entre os níveis de hierarquia de acordo com a frequência de acesso.

Deve ser considerado ainda o fato de que algumas das características de desempenho na recuperação dos dados em tempo real dos objetos multimídia requerem um suporte a operações de tempo real bastante substancial. Escalonamento apropriado e algoritmos de alocação de recursos devem ser desenvolvidos para permitir alto desempenho em ambientes que misturam dados sensitivos a atrasos e dados não sensitivos a atrasos. Esse fato implica a necessidade de que os dados sempre alcançam o receptor a tempo de que não ocorram interrupções na apresentação, por exemplo, se a voz ou o vídeo é recuperado de um dispositivo secundário, o próximo pacote a ser apresentado deve alcançar o receptor antes que o pacote atual seja totalmente consumido; caso contrário, interrupções na apresentação serão ouvidas ou vistas. No caso onde dois ou mais diferentes tipos de dados multimídia são recuperados para apresentação, existe ainda o problema da sincronização a ser provida no receptor. No caso do vídeo associado com voz e texto, os três dados devem ser sincronizados. O pacote de vídeo deve alcançar o receptor junto com o pacote de áudio e o pacote de texto.

Os dados multimídia, com grandes volumes e recuperação em tempo real, impõem requisições de demanda muito altos ao sistema e, portanto, o uso efetivo de paralelismo no processamento e no armazenamento da informação é essencial para um bom desempenho desses sistemas. Além disso, a maioria das aplicações de sistemas de informação multimídia tem um caráter de distribuição por natureza, pois os dados encontram-se espalhados em várias mídias e em vários pontos diferentes do sistema. Portanto, suporte a sistemas distribuídos se faz necessário para garantir bom desempenho na recuperação dos dados.

Os sistemas de banco de dados são considerados como o componente principal de sistemas de informação multimídia, que têm por objetivo suportar o armazenamento e

recuperação eficiente de dados para garantir sua persistência, isto é, sua existência além do período de execução da aplicação. Este requisito é desnecessário em algumas aplicações, como no caso de um sistema elementar de teleconferência, e imprescindível em outras, como o caso de manipulação de documentos multimídia.

Devido ao fato das características dos dados multimídia serem muito complexas, os sistemas convencionais não são adequados para sua manipulação. A principal razão é o fato de sistemas relacionais de banco de dados representarem todos os dados em forma de tabelas, não sendo um mecanismo adequado para representação de dados contínuos, sendo entretanto adequados à maior parte das aplicações nas áreas comerciais e administrativas.

Levando-se em consideração as tecnologias de banco de dados existentes, o mecanismo mais adequado para a representação de dados multimídia é o fornecido por banco de dados orientado a objetos, onde a abstração suportada por este modelo é adequada para a representação da informação e de sistemas associados a estes documentos, permitindo o armazenamento e o acesso aos dados representados diretamente em termos de primitivas do padrão. Outro aspecto a ser considerado é com relação às características de desempenho necessárias. Sob este aspecto, pesquisas vem sendo desenvolvidas no sentido de propor arquiteturas de banco de dados específicas para aplicações multimídia, de uma forma que através de sua utilização, o desempenho da aplicação não seja comprometido.

2.3.1 Sistemas de banco de dados orientados a objetos

Aplicações recentes de computadores exigem o suporte efetivo à gerência de objetos complexos, onde sistemas de informação multimídia são apenas um dos exemplos desta classe de aplicações. Além das características associadas à estrutura de tais objetos, outros requisitos estão associados à sua evolução, tais como controle de múltiplas versões e suporte a transações longas. Os sistemas de banco de dados orientados a objetos têm como objetivo principal atender a estes e outros requisitos das aplicações. A representação das estruturas dos dados é feita utilizando-se o paradigma de orientação a objetos, onde os dados em um sistema de banco de dados são representados através de primitivas definidas em um modelo de dados.

Um modelo de dados orientado a objetos permite ao usuário modelar cada entidade

conceitual utilizando o mesmo conceito simples de modelagem, ou seja, o objeto, além da existência de mecanismos adicionais que permitem que o usuário represente relacionamentos entre objetos e coleções de objetos. Dentro dos paradigmas, cada entidade do mundo real é modelada por um objeto, associado a um identificador único.

Estes objetos possuem um conjunto de atributos de instâncias e de métodos, características que permitem definir objetos arbitrariamente complexos como uma agregação de outros objetos. O comportamento dos objetos são definidos por operações e o estado dos objetos, pelos valores dos atributos. Os objetos que compartilham a mesma estrutura e o mesmo comportamento são agrupados em classes, podendo ser definidas como uma especialização de uma ou mais classes, permitindo, com isso, o uso dos conceitos de herança simples e herança múltipla.

Um sistema gerenciador de banco de dados orientado a objetos (OODBMS) é definido como um sistema que suporta diretamente um modelo de representação baseado no paradigma de orientação a objetos. Como qualquer gerenciador, um OODBMS deve suportar o armazenamento persistente para objetos e seus descritores (esquema), assim como linguagens para definição e manipulação de objetos e esquemas. Opcionalmente, um OODBMS pode suportar linguagens de consulta (*query*), mecanismos de otimização de acesso, controle de concorrência e de acesso para sistemas multi-usuários e mecanismos de recuperação. Nestes sistemas, uma alternativa de armazenamento dos dados complexos seria a utilização de BLOB's, que poderiam garantir, parcialmente, o armazenamento e recuperação dos dados de uma forma eficiente.

2.3.2 Sistemas convencionais de banco de dados

Atualmente, os sistemas convencionais de arquivos não são suficientes para permitir que clientes façam acesso a estes tipos de dados através de um sistema de informação multimídia, de modo a atingir as taxas necessárias de recuperação, armazenamento ou processamento de dados. Desta forma, pesquisas em curso tentam suportar mecanismos adequados para a construção de subsistemas de armazenamento e manipulação de dados para sistemas de informação multimídia.

Estes sistemas representam as estruturas dos dados em forma de tabelas, sendo

necessário uma conversão da estrutura complexa dos dados em entidades e relacionamentos, durante a fase de armazenamento, e uma nova conversão durante a recuperação destes dados. Nestes sistemas, a alternativa de armazenamento dos dados em BLOB's também poderia ser utilizada.

Tanto os sistemas de banco de dados orientado a objetos como os sistemas de banco de dados convencionais não são adequados para a manipulação e gerência de dados multimídia, pois não garantem a recuperação dos dados de forma eficiente, nem garantem o cumprimento dos requisitos de tempo requeridos pela aplicação, além de não permitirem uma consulta eficiente aos dados armazenados. Devido aos problemas de interpretação destes dados, ocasionam uma perda de desempenho, pois não conseguem cumprir os requisitos de tempo das aplicações, não garantem que a demanda e qualidade de serviços requeridas sejam atendidas e, no caso dos banco de dados orientados a objetos, tem-se ainda o problema da não existência de tipos e classes específicos para multimídia.

Os sistemas de banco de dados, convencionais ou não, devem permitir a execução de funções básicas como: armazenamento e cadastramento de dados, consulta e recuperação, de uma forma eficiente e modificação e remoção de dados desatualizados.

2.3.3 Sistemas de banco de dados orientados a multimídia

Os sistemas de banco de dados disponíveis comercialmente, orientados a objetos ou não, são inadequados para aplicações multimídia distribuída, não existindo ainda nenhum OODBMS que consiga atender de forma satisfatória a todos os requisitos destes tipos de aplicações. Com base neste problema, existe a grande necessidade em se criar um sistema gerenciador de banco de dados orientado a multimídia, que tenha como objetivo principal armazenar e processar informações e dados contínuos, de forma a garantir que os requisitos de tempo destas aplicações sejam cumpridos. Para tanto, é necessário o desenvolvimento de um suporte adequado a estes dados, via sistemas de arquivos (repositórios especializados que suportem a estrutura complexa dos dados).

Os requisitos básicos para o desenvolvimento destas aplicações estão relacionados a heterogeneidade, suporte a mídias contínuas e com grande volume de informação, representação de estruturas complexas e integração dos serviços do gerenciador de banco de

dados aos serviços oferecidos por uma plataforma distribuída para atender aos requisitos das aplicações multimídia distribuída.

Esta integração de serviços pode ser resolvida através da adoção de sistemas abertos distribuídos como base para a arquitetura de sistemas multimídia. Com relação à melhor maneira de se atender as distintas necessidades de armazenagem e processamento de cada tipo de mídia, poderiam ser utilizados servidores dedicados, que atenderiam os requisitos dos tipos de mídias simples [9].

Para resolver estes problemas no desenvolvimento de um sistema gerenciador de banco de dados orientado a multimídia, estão em curso pesquisas com relação a um sistema de arquivos adequado, ao suporte a facilidade de acesso e consulta aos dados, a disponibilidade de metadados sobre as informações multimídia e métodos de busca baseado em conteúdo. O objetivo com isso é facilitar o processo de consulta aos dados e também com relação à qualidade de serviços e desempenho, levando em conta principalmente a recuperação dos dados contínuos, com os respectivos requisitos de sincronização, atendendo de forma satisfatória os requisitos de tempo da aplicação [16].

A respeito de todos os problemas levantados, tem-se ainda que considerar que muitos trabalhos e pesquisas ainda terão que ser feitos para que se chegue a níveis de desempenho satisfatórios. Se for considerado ainda a natureza multisensorial do ser humano, a inclusão dos outros três sentidos do homem (a multimídia hoje trabalha só com visão e audição) possibilitaria realizarmos avanços na área da realidade virtual, holografia, e outros tantos, o que aumentará ainda mais os níveis de exigência dos sistemas multimídia.

Assim, pode-se concluir que esta é uma área fascinante e que ainda tem muito a ser feito. Além disso, muito trabalho está sendo feito dentro de modelos de como armazenar e, especialmente, como recuperar informações multimídia. Alguns aspectos principais relacionados com sistemas de gerência de banco de dados para multimídia distribuída também foram apresentados, além de questões como armazenamento de endereços, busca, recuperação, comunicação e exibição de objetos multimídia complexos em um ambiente distribuído aberto. Tanto ambientes distribuídos heterogêneos como abertos requerem um mecanismo de interoperabilidade para suportar comunicações e serviços entre aplicacões desenvolvidas independentemente, especialmente entre sistemas gerenciadores de banco de dados e seus clientes.

Questionamentos ainda existem com relação ao que deve ser considerado serviços básicos e facilidades requeridas por atividades de manipulação de dados multimídia e também com relação à alocação dos banco de dados. Algumas destas facilidades poderiam ser suportadas tanto por componentes especializados de uma plataforma distribuída como por sistemas gerenciadores de banco de dados de objeto. As características de documentos multimídia computacionais foram exploradas e devem ser consideradas por todos os serviços e facilidades envolvidas com sua manipulação, em especial pelas facilidades de armazenamento e recuperação.

Apesar do volume de pesquisas na área de sistemas gerenciadores de banco de dados orientados a objetos, os resultados ainda não foram bem aplicados. No entanto, pode-se ter a certeza de que as pesquisas em sistemas gerenciadores de banco de dados ativos, sistemas gerenciadores de banco de dados em tempo real e sistemas gerenciadores de banco de dados distribuídos e heterogêneos irão contribuir de forma significativa para a resolução de problemas dos sistemas gerenciadores de banco de dados para multimídia. A expectativa é que os desenvolvimentos nas áreas de sistemas gerenciadores de banco de dados multimídia, sistemas operacionais multimídia (incluindo sistemas de arquivos) e protocolos de rede para multimídia estejam fortemente ligados, devido ao domínio do tratamento de informação multimídia, para que com isso possam trazer benefícios mútuos.

2.4 Trabalhos relacionados

No contexto de sistemas para multimídia, outros projetos estão sendo desenvolvidos. Por exemplo, o projeto GLASS (*Globally Accessible Services*) [10] [17], que apresenta como cenário principal um *kioski* turístico, possibilitando ao usuário consultar informações (multimídia e hipermídia) relacionadas ao seu interesse. O objetivo principal deste projeto é definir um ambiente de autoria empregando a técnica WYSIWYG (*What You See Is What You Get*), definindo uma arquitetura e realizando a implementação de um pré-produto de um sistema aberto para serviços multimídia de usuários profissionais e consumidores.

Outro projeto importante é com relação ao estudo das plataformas distribuídas no suporte a sistemas multimídia [2], onde é feito um levantamento das plataformas mais influentes ao suporte a sistemas de computação distribuída, mostrando as tendências de cada

uma delas, os requisitos dos sistemas distribuídos e o processamento distribuído aberto, tendo como objetivo principal garantir a interoperabilidade entre sistemas distribuídos heterogêneos.

Com relação ao armazenamento e recuperação de objetos multimídia, um trabalho está sendo realizado visando enfocar estas questões, discutindo principalmente aspectos referentes ao suporte pelos sistemas gerenciadores de banco de dados para esta finalidade [23].

Um sistema também relacionado a este projeto é o de um correio eletrônico multimídia/hipermídia [3], principalmente por levantar questões relacionadas aos problemas com o tráfego na rede e espaço de armazenamento, tentando apresentar soluções para cada caso.

Com o objetivo de complementar o desenvolvimento deste projeto, dois outros trabalhos se tornam necessários, para um desempenho mais satisfatório da aplicação. O primeiro deles é o projeto de um banco de dados para multimídia, onde os dados e meta-dados dos componentes da apresentação estarão armazenados. O outro é o desenvolvimento de um repositório para os dados multimídia, que tenha como objetivo principal tentar garantir a recuperação do dado a ser apresentado da forma mais otimizada possível, evitando com isso a degradação do desempenho da aplicação.

Capítulo 3

Integração de Subsistemas Multimídia

"A possibilidade de arriscar é que nos faz homens. Vôo perfeito. No espaço que criamos ninguém decide sobre os passos que evitamos. Certeza de que não somos pássaros e que podemos voar; tristeza de que não vamos por medo dos caminhos."

(Fernando Pessoa)

Alguns padrões relacionados à multimídia, como *Joint Photographic Expert Group* (JPEG) e *Moving Picture Experts Group* (MPEG) descrevem somente o conteúdo dos objetos de informação, não descrevendo os inter-relacionamentos entre as diversas partes de uma apresentação multimídia. Outros, como o caso da *Hypermedia Time-Based Structuring Language* (Hytime) [20], uma proposta de padrão internacional definida para incorporar e integrar notações distintas e representações relacionadas com objetos de dados que podem estar inter-relacionados no tempo e no espaço, trabalha somente com o problema de resolução de *hiperlinks* como um problema de endereçamento, ou seja, localização de objetos no tempo e no espaço, sem uma preocupação com o tipo de objetos manipulados. Além disso, reflete a visão de que todas as tecnologias multimídia, tecnologias gráficas, tecnologias de áudio, etc, competem de tal forma pelo mercado que deve ser tecnicamente possível acomodar qualquer combinação de tais formatos e tecnologias em qualquer produto de informação.

A *Standard Generalized Markup Language* (SGML) [20], por sua vez, foi desenvolvida para atender às necessidades associadas à descrição de documentos em termos de sua estrutura, estando principalmente voltada para a definição de famílias de documentos. Ba-

seado nesta linguagem, surgiu a *Hypertext Markup Language* (HTML) [20], uma coleção de estilos usados para definir os vários componentes de um documento, permitindo integrar diversos tipos de mídias, embora em sua representação apenas textos sejam usados.

A representação de documentos com integração de outras mídias pode ser feita usando uma combinação de Java e HTML. Java é baseado em C++ e, no caso da utilização de *browsers*, os programas são feitos na forma de *applets* e são incluídos dentro de um documento HTML. A meta principal para o Java é a criação de uma aplicação que é completamente portátil, executando corretamente em computadores de características distintas sem necessidade de adaptação [19].

Um outro conceito que também deve ser levado em consideração é o *Multimedia and Hypermedia Expert Group* (MHEG) [13] [5] [12], outra proposta de padrão internacional, para uniformizar os chamados objetos de informação multimídia, que constituem de unidades de informação e tem como objetivo principal atender aos requisitos das aplicações multimídia. O desenvolvimento do padrão MHEG tornou-se uma grande necessidade devido ao fato de padrões para formatos monomídia (como JPEG e MPEG) não atenderem aos requisitos exigidos pelas aplicações com dados multimídia.

As características principais do padrão MHEG relevantes para este projeto foram retiradas da primeira versão do documento [13] e da sexta versão do documento, ainda em versão *draft* [12], destacando-se o formato de compartilhamento de apresentações multimídia interativas (que permite a troca de informações multimídia entre duas entidades pertencentes a sistemas heterogêneos, com a possibilidade de reuso da informação de conteúdo, além de permitir a troca de dados dentro da aplicação), a entrega e apresentação de objetos multimídia em tempo real, a necessidade de recursos mínimos para a apresentação das informações, a interação com o usuário (fornecendo um conjunto de objetos de interação independentes de plataformas, que possuem as mesmas características do ambiente de execução) e o uso de modelos de orientação a objetos (com propriedades de herança de classes), onde as instâncias de uma classe são conhecidas como objetos MHEG [6].

Podemos considerar o padrão MHEG como uma das chaves para o desenvolvimento de aplicações e serviços multimídia distribuídos, sendo especialmente designado para aplicações que necessitam de:

- sincronização multimídia entre os componentes de um objeto composto único e facilmente endereçado;
- interatividade, com estruturas específicas para suportar esta interatividade com o usuário final;
- troca em tempo real, através de mecanismos simples e eficientes, para otimizar a troca de objetos compostos na sequência correta;
- representações da informação na forma final, sem processamento adicional requerido para reestruturar a informação a ser apresentada.

Este padrão fornece uma codificação de informações multimídia para serem utilizadas e trocadas por aplicações em um amplo domínio e seu projeto tem sido dirigido pelas análises dos requisitos das aplicações, para com isso identificar seus aspectos e incorporá-los em uma codificação simples.

Existem muitas razões para a definição de estruturas de informação multimídia. A padronização somente no nível da informação monomídia não é suficiente para garantir a portabilidade das aplicações, que apesar de não utilizarem dados monomídia separados, necessitam definir um conjunto associado de parâmetros, necessários para apresentação. Além disso, esta padronização não é suficiente para o projeto e troca de informações multimídia.

As aplicações multimídia contam com abstrações ou estruturas de dados que fornecem características como ligações de sincronização e ligações lógicas entre dados monomídia. O projeto e a gerência de aplicações multimídia e hipermídia em ambientes distribuídos serão facilitados se os detalhes internos da apresentação da informação forem mapeados pelo uso de abstrações apropriadas. A aplicação somente deve estar de acordo com funções como: gerência de distribuição da informação e da interação com o usuário; escalonamento de apresentação e outras atividades de alto nível.

Neste contexto, o MHEG tem o objetivo de fornecer estruturas de informação multimídia genéricas que satisfaçam os requerimentos e estejam de acordo com: apresentações em tempo real utilizando apresentações multimídia e facilidades de sincronização disponíveis na plataforma da aplicação; troca em tempo real utilizando facilidades de comunicações disponíveis nas plataformas das aplicações; representação da forma final, onde a informação é representada e codificada para representação direta, sem o requerimento de procedimentos adicionais destas estruturas.

3.1 Objetivos do padrão MHEG

Este padrão tem como objetivo fornecer facilidades de troca de vários tipos de mídias. Os dados de mídia devem ser codificados de acordo com outros padrões internacionais, como JPEG para imagens estáticas, MPEG para vídeo (imagem em movimento), ou podem também ser encapsulados utilizando-se técnicas de codificação privadas e proprietárias. Com o objetivo de suportar troca multimídia, MHEG fornece estruturas para a composição de diferentes tipos de mídias dentro de uma única unidade de troca.

Além disso, MHEG deve suportar apresentação na forma final de vários tipos de mídia, fornecendo facilidades para a identificação da técnica de codificação para permitir ao usuário o uso de recursos de apresentação apropriados em uma plataforma específica e estruturas para a composição de diferentes tipos de mídia em uma apresentação. Suporta também interação e modificação de dados de mídia, juntamente com seus atributos de apresentação, como tamanho, volume, etc.

Para que o dado codificado possa ser apresentado é necessário que suporte à especificação de recursos mínimos requeridos, obtendo, com isso, facilidades para a troca de informações relacionadas a estes recursos, fornecidos pela informação fonte.

3.2 Descrição de objetos MHEG

O objeto MHEG é considerado um componente básico nos diversos padrões existentes e, com sua utilização, diversas aplicações são capazes de trocar recursos de informação básicos. Na descrição destes objetos é importante salientar a existência de atributos comuns, dados de conteúdo, comportamento, interação com o usuário e composição. Pelo fato do MHEG estar ainda em um estágio não definitivo, este padrão está sujeito a mudanças. Os atributos comuns fornecem um modelo chamado "definições úteis" para garantir consistência de tipos das estruturas de dados, e uma classe de objetos MHEG, considerada raiz da hierarquia de classe.

Esta classe de objetos tem como objetivo principal definir estruturas de dados comuns para todas as outras classes e herdadas por todas as outras de mais baixo nível, como a estrutura de dados "identificador de classe" que identifica o tipo de cada classe codificada

por um valor inteiro pré-definido. É possível também caracterizar cada objeto MHEG por um número de atributos, ou seja, uma pequena descrição de suas características, que vão servir como chaves de pesquisa e são opcionais, sendo sua definição de responsabilidade da aplicação.

Outro aspecto importante a ser destacado é que MHEG não previne a troca de objetos separadamente, sendo necessário uma estrutura de dados para compor a apresentação multimídia.

As classes compostas fornecem o esqueleto da apresentação. Em um primeiro passo, objetos pertencentes a uma certa parte da apresentação têm que estar agrupados. Esse recurso é conseguido através da definição de recursividade de classes compostas, que permitem que objetos compostos façam parte de outros objetos compostos.

Além disso, as classes compostas fornecem suporte para associação de objetos multimídia. Este mecanismo apresenta uma abordagem consistente para sincronização em tempo, espaço e ligação de um conjunto de objetos. Fornece também a estrutura lógica para descrever a lista de interações possíveis oferecidas para o usuário, mas não define as facilidades das interações fornecidas pela interface do usuário.

De um ponto de vista lógico, um objeto composto pode ser entendido como um tipo de repositório. Comparando com classes de índice, nas classes compostas MHEG há uma distinção entre objetos referenciados ou inclusos. As referências externas ajudam a particionar uma apresentação complexa em pedaços pequenos, com o objetivo de reduzir a quantidade de memória necessária no sistema final. Suportam também requisitos em tempo real, permitindo a recuperação de objetos de forma parcial.

O padrão MHEG define mais três outras classes: a classe repositório, a classe descritor e a classe *script*. A classe repositório fornece um empacotamento de vários objetos com o objetivo de trocá-los como um conjunto total, mas sem a informação de como reconstruir a apresentação.

A classe descritor, por sua vez, codifica informações relacionadas à apresentação de objetos, como o tipo de representações de mídia que são usadas para um objeto conteúdo. Além disso, deve conter informações sobre qualidade de serviços, para suportar troca em tempo real de objetos MHEG. Um engenho pode utilizar esta informação para avaliar os

requisitos da apresentação, respeitando os recursos disponíveis da plataforma de execução, ou seja, define uma estrutura para troca de recursos de informação sobre um único ou um conjunto de outros objetos trocados.

A informação pode ser usada para facilitar a correspondência entre os recursos requeridos para apresentar os objetos e os recursos disponíveis para o sistema, ou para fornecer uma negociação entre o "fonte" do objeto MHEG e o "destino" da apresentação.

A classe *script* fornece uma interface para funções ou programas externos e também um repositório para relacionamentos complexos entre objetos MHEG e objetos executáveis, definido por uma linguagem não MHEG.

Uma aplicação típica são as consultas em banco de dados, onde uma entrada é codificada por objetos MHEG, a não ser que objetos *scripts* solicitem uma consulta executada em um ambiente *script* e transmita os dados entre o engenho MHEG (ver Seção 3.3) e o ambiente *script*.

Dentro deste contexto são definidas as informações que devem ser fornecidas para possibilitar a recuperação dos dados multimídia a serem apresentados. As características mais importantes envolvem os meta-dados de cada objetos, ou seja, informações como nome da mídia, tipo de compressão, tamanho e taxa de apresentação, estimada em quadro (vídeo) ou amostra (áudio) por segundo.

Estas informações são mantidas no banco de dados, caracterizando cada informação possível de ser apresentada e ainda permitindo a composição de uma apresentação, baseada nas características de cada objeto em particular.

Com relação à integração com o repositório, somente informações a respeito do nome e tipo de compressão da mídia são relevantes, para que esta possa ser recuperada de forma transparente para a aplicação.

3.3 Engenho MHEG

Um engenho MHEG [5] é um processo ou um conjunto de processos que interpretam objetos MHEG codificados de acordo com as especificações deste padrão. A estrutura básica de um engenho é apresentada na Figura 3.1.

Todos os processos e módulos são de responsabilidade do engenho e estão fora do escopo do padrão MHEG. Para os serviços de gerência de sistemas, o engenho utiliza as facilidades oferecidas pelo sistema operacional utilizado.

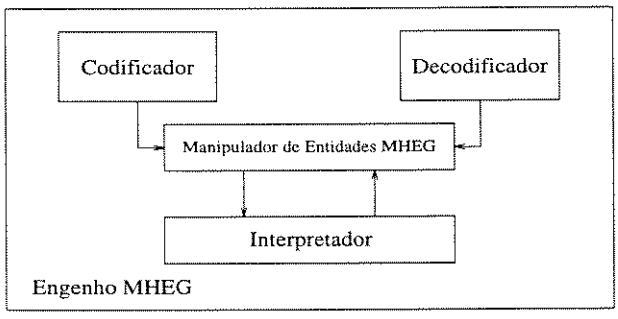


Figura 3.1: Módulos de um engenho MHEG.

O decodificador é a entidade responsável por converter objetos de dados MHEG enviados seguindo uma sintaxe abstrata para um formato interno, permitindo assim que o engenho manipule objetos MHEG contidos nestes dados. Este módulo é utilizado quando um engenho MHEG envia objetos de uma aplicação para outra. O codificador é a entidade responsável por formatar objetos internos MHEG na sintaxe abstrata para que possam ser trocados com outros sistemas. É utilizado quando um engenho MHEG recebe objetos de outra aplicação.

O manipulador de entidades é responsável por manipular objetos MHEG, objetos executáveis e canais em um formato interno, podendo alocar entidades e controles da gerência de memória enquanto o interpretador é responsável por processar os objetos. Alguns dos principais processos existentes em um interpretador são: de preparação, de criação de canais e objetos executáveis, de ativação, de *script*, e processos de apresentação e interação. Dentre estes processos, os relevantes à aplicação são:

- processo de preparação: permite a preparação de quaisquer objetos MHEG antes que estes sejam processados pelo engenho, como no caso de se fazer a recuperação de uma sequência de áudio. Este processo pode ser muito demorado, sendo muitas vezes mais eficiente começar o processo de recuperação antes do momento da apresentação.
- processo de criação de canais: permite a criação de qualquer canal, possibilitando assim o compartilhamento dos objetos a serem apresentados.

- processo de *script*: interpreta os dados de *script*, como descrito no objeto *script* modelo. Os parâmetros podem ser trocados bidirecionalmente entre o objeto *script* e o *script* executável.
- processo de apresentação: é responsável por manipular os componentes executáveis para o usuário. Relaciona-se com a interface do usuário e serviços de apresentação existentes no sistema. Este processo interpreta as ações elementares de apresentação que podem afetar dinamicamente os componentes executáveis com relação à aceitação de canais, nível de percepção, componentes temporais, espaciais, auditíveis, e escolha de fluxo em um objeto multiplexado. Além disso, este processo interpreta a apresentação já sugerida dentro do estilo da interação.
- processo de interação: permite a interação com o usuário. O padrão MHEG não se preocupa com a definição da forma da apresentação multimídia, trabalhando sim com ferramentas de interface gráfica existentes no sistema. É composto de dois sub-processos existentes independentemente: processos de seleção e de modificação, e ainda deve requisitar serviços do processo de apresentação, quando os estilos das interações forem especificados.

Com relação às questões de implementação, descreve-se uma estrutura e transferência de objetos MHEG codificada em detalhes, mas trata muito pouco da questão da implementação. Algumas experiências anteriores em ambientes de implementação MHEG vieram através de protótipos de sistemas. É necessário primeiro uma visão da arquitetura do ambiente de execução para depois detalhar a reprodução em um terminal de apresentação [17].

O padrão MHEG não define o modo pelo qual o engenho manipula e troca objetos MHEG ou objetos executáveis, mas assume que um objeto, uma vez trocado, será usado por uma aplicação e manipulado pelo engenho, não considerando sua estrutura interna ou organização. As únicas considerações feitas são em relação à capacidade do engenho em avaliar *status* e valores de atributos dos objetos sob seu controle, as mudanças que ocorrem e que podem ser usadas para expressar condições em ligações. Além disso, o engenho trabalha com um sistema de suporte à interface do usuário, onde os resultados destas interações são formalizados pelo engenho, com o objetivo de modificar o *status* das interações, seleções e modificações correspondentes.

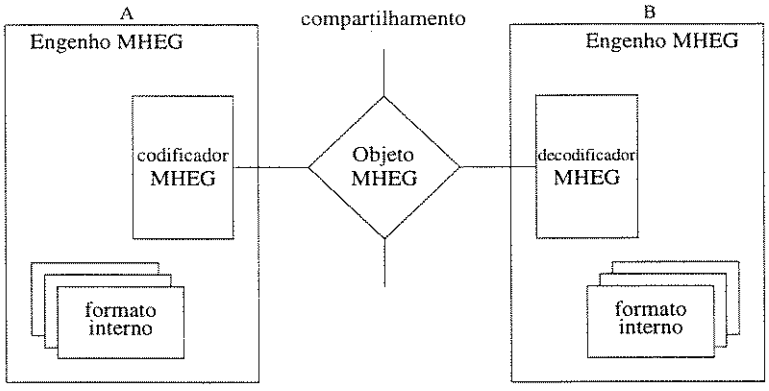


Figura 3.2: Troca de objetos MHEG.

A Figura 3.2 ilustra a localização de um objeto MHEG, sendo trocado entre duas aplicações. O objeto é definido somente no ponto de troca entre as aplicações A e B. Quando um sistema A deseja enviar um objeto para B, chama um codificador MHEG que irá converter o formato interno usado por A para o formato definido pelo padrão. Quando B recebe o objeto, este passa por um decodificador MHEG que irá convertê-lo para o formato interno usado por B. É importante ressaltar que o formato usado em A e B pode ser o mesmo que o formato MHEG, mas não obrigatoriamente. Se a troca entre A e B puder ocorrer nas duas direções, é necessário a existência de um decodificador e um codificador nos dois engenhos.

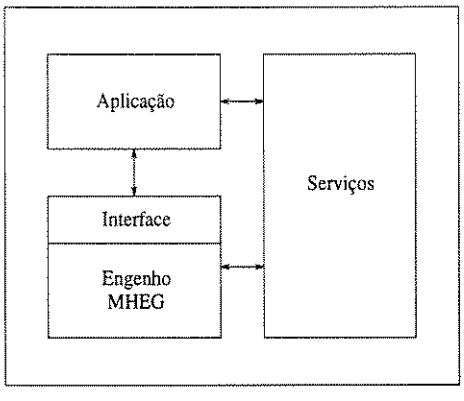


Figura 3.3: Módulos responsáveis por serviços extras.

O engenho MHEG fornece uma interface para a aplicação, que oferece facilidades de controle, sendo o único ponto de acesso das aplicações para objetos MHEG dentro dos

engenhos. É importante salientar que este modelo assume que o engenho é responsável apenas pelo processamento de objetos MHEG e interações com a aplicação. Serviços extras como acesso a banco de dados, comunicação, apresentação gráfica, são fornecidos pela aplicação ou por outros módulos, conforme ilustrado na Figura 3.3. Neste caso, existe a necessidade de definir interfaces de comunicação entre cada módulo, para que uma fase de negociação (onde as requisições da aplicação são reconhecidas e a possibilidade de sua execução é avaliada) possa ocorrer entre eles.

Este projeto segue como base esta arquitetura, no que diz respeito a definição de um cliente, responsável pela gerência de acesso aos dados multimídia, bem como a interação deste gerenciador com o repositório e com o Banco de Dados *Multiware*, conforme ilustrado na Figura 3.4.

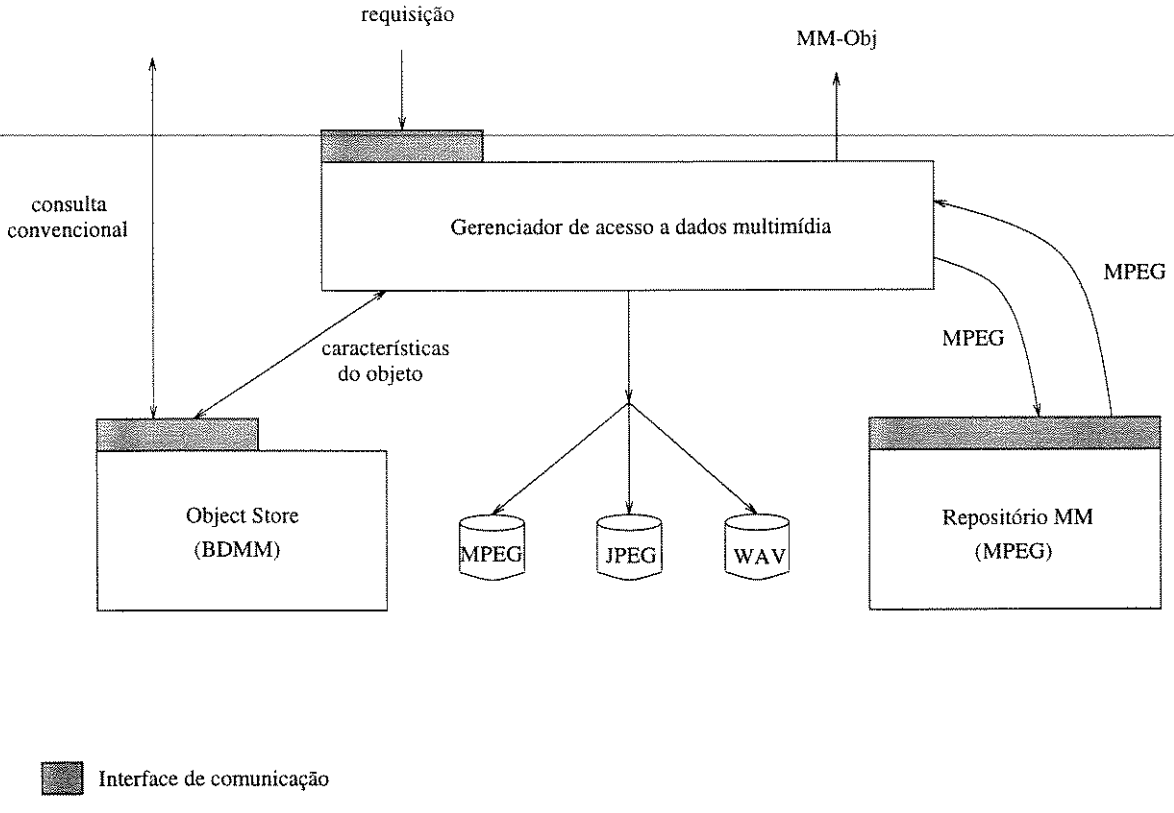


Figura 3.4: Arquitetura do sistema.

A interação do gerenciador de acesso com o repositório e com o banco de dados são

feitos através de interfaces pré-definidas possibilitando essa comunicação. A consulta ao banco de dados pode ser efetuada pela aplicação de forma direta, visando a recuperação das caracterísiticas do objeto, ou também através de uma consulta convencional, com o objetivo de fazer um levantamento dos objetos existentes.

Após a seleção dos objetos a serem apresentados, estes podem ser recuperados por um repositório especializado, através de uma interface pré-definida, ou ainda de dentro do sistema de arquivos. Uma vez recuperados, os dados são enviados para a aplicação cliente, para então serem apresentados.

Capítulo 4

Gerenciador de Acesso

"Se a nossa opção é progressista, se estamos a favor da vida e não da morte, da equidade e não da injustiça, do direito e não do arbítrio, da convivência com o diferente e não de sua negação, não temos outro caminho senão viver plenamente a nossa opção. Encarná-la, diminuindo assim a distância entre o que dizemos e o que fazemos."
(Paulo Freire)

4.1 Estrutura do sistema

A estrutura do sistema gerenciador de acesso foi inspirada na estrutura de um engenho MHEG, sendo apresentada na Figura 4.1.

A estrutura de um engenho MHEG, da forma como foi apresentada, não pode ser considerada suficiente para cumprir todos os requisitos exigidos por um sistema multimídia, principalmente por não especificar as possíveis entidades e relacionamentos entre elas e o sistema. Para isso, dentro desta estrutura, foi necessário a inclusão de algumas funcionalidades, bem como a definição das entidades que interagem com o sistema e das interfaces para que esta interação seja efetuada:

1. Cliente: entidade que representa a aplicação que está fazendo a requisição de um serviço específico. A comunicação do cliente com a aplicação é feita através de um canal de comunicação específico e a interface é feita usando-se um *script* Java.

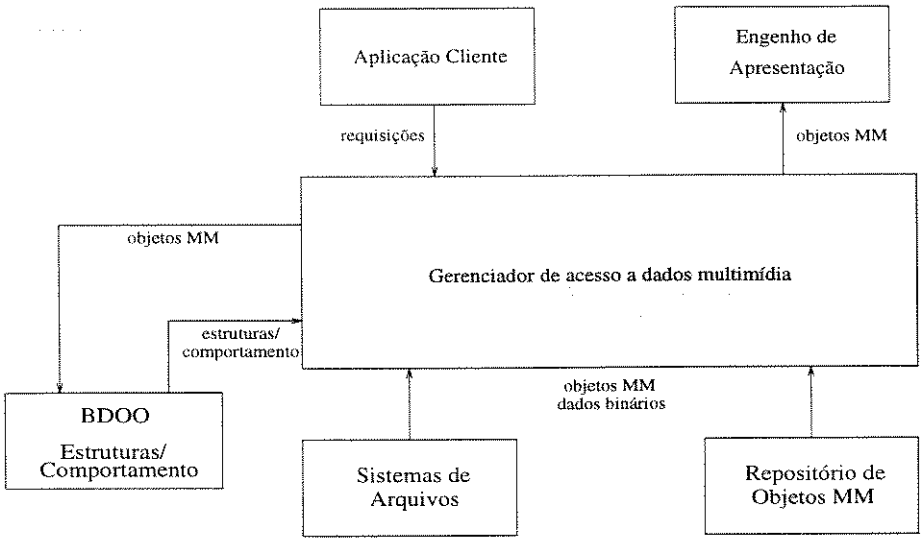


Figura 4.1: Módulos externos ao gerenciador de acesso.

2. Banco de dados orientado a objetos: entidade que contém informações com relação às características dos objetos a serem utilizados na montagem da aplicação solicitada, seguindo a filosofia da orientação a objetos. Sua comunicação com o sistema é feita através de uma interface padrão de comunicação.
3. Engenho de apresentação: entidade que representa a aplicação do cliente que está recebendo o serviço solicitado, para ser apresentado de acordo com os padrões exigidos anteriormente. A negociação com este engenho pode ser feita através do mesmo protocolo utilizado pelo cliente, sendo necessário também um protocolo "dedicado", por onde possam ser transmitidos os dados existentes na aplicação, tentando com isso garantir qualidade de serviço.
4. Repositório de objetos multimídia: entidade que representa um repositório específico onde são armazenados os objetos a serem transmitidos. A comunicação desta entidade com o sistema também é feita através de uma interface. Um dos objetivos deste repositório é a otimização no acesso a dados multimídia.
5. Sistemas de arquivos: entidade que representa o local de armazenamento dos dados que não são mantidos em um repositório específico.

O processo de negociação do cliente com a aplicação é feito de forma direta, pois o cliente conhece a localização do servidor, não sendo necessário nenhuma plataforma especial

para permitir esta comunicação. Em uma implementação distribuída deste sistema, este processo pode ser feito através da solicitação de um serviço na rede e sua execução por um servidor especializado.

4.2 Descrição funcional

O sistema desenvolvido é composto por cinco processos básicos: negociação, recuperação das características do objeto, recuperação dos objetos a serem apresentados, montagem do objeto multimídia e escalonamento de envio do objeto para apresentação, esquematizados na Figura 4.2.

O processo inicial de **negociação** recebe uma mensagem de negociação, decodifica-a de acordo com a estrutura de um objeto multimídia (objeto MM) e verifica se o objeto solicitado existe no banco de dados. Após esta verificação, se o objeto existir, passa-se para o processo de recuperação das características do objeto.

A **recuperação** das características do objeto é o processo que recebe o objeto solicitado pela aplicação, verifica se as características do objeto existem no banco de dados e após esta verificação, recupera estas características, armazenando-as em um depósito para consultas futuras. Estas características são repassadas para o processo seguinte de preparação da montagem do objeto multimídia.

O processo de **montagem** do objeto multimídia verifica, a partir da identificação do objetos e de suas características recuperadas anteriormente, a consistência destas características com as requisitadas pela aplicação. A partir daí, depois de confirmada sua existência, o objeto a ser transmitido é recuperado do repositório, através do processo de **recuperação do objeto a ser apresentado.**

O processo de **escalonamento** recebe as características finais do objeto a ser transmitido e através da "interação com a rede", o objeto é recuperado do repositório de acordo com a necessidade e vai sendo transmitido para o engenho de apresentação correspondente, de forma sincronizada. Esta interação é necessária para garantir a QoS.

Entre as funcionalidades relevantes ao projeto do desenvolvimento de um gerenciador de acesso a dados multimídia, é importante ressaltar funcionalidades relacionadas

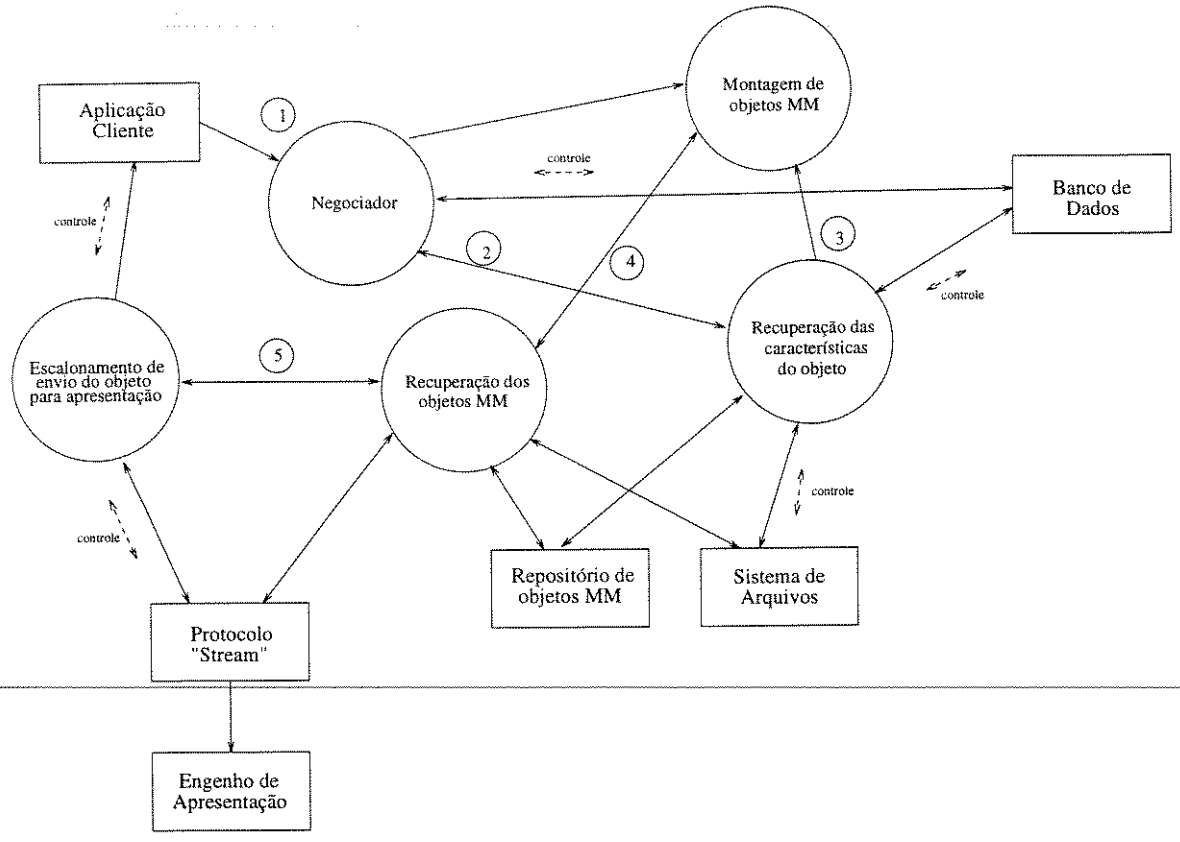


Figura 4.2: Estrutura de comunicação do sistema.

com manipulação, o armazenamento e a recuperação de mídias contínuas. Para tanto, é necessário a criação de processos que iniciem e disparem sua execução, e possam também parar temporária ou definitivamente, e ainda reiniciar a mídia quando sua apresentação chega ao fim. Para a execução destes objetos é necessário a alocação de recursos para a apresentação da mídia, utilizando componentes multimídia.

Processos relacionados à sincronização e gerência de tempo podem ser definidos através da utilização de *threads*, para eventualmente poder iniciar a apresentação de diferentes tipos de mídia ao mesmo tempo. Deve ser possível também permitir que o processo de *download* de uma mídia possa ser verificado, para com isso poder se garantir que não haja nenhum problema em sua apresentação.

Para uma integração eficiente com o banco de dados, é necessário definir os meta-

dados relevantes para a aplicação, para facilitar a manipulação e recuperação da mídia. Já com o repositório, esta integração é feita de forma mais direta, sem a necessidade de muitos detalhes relacionados aos dados a serem apresentados.

4.3 Interfaces do sistema

As interfaces do sistema foram definidas de forma a facilitar a interação do usuário com a aplicação, permitindo que este pudesse escolher uma apresentação previamente montada ou selecionar objetos, a partir de consulta ao banco de dados, de forma separada. Na tela principal do sistema, apresentada nas Figuras 4.3 e 4.4, as apresentações disponíveis são mostradas e a partir deste *menu*, é feita a seleção de qual será executada, ou ainda, quais objetos pertencem a cada uma delas. As apresentações selecionados são colocadas em um *frame* de objetos selecionados para apresentação. Os dados colocados nestes dois *frames* principais são obtidos do banco de dados. Para esta aplicação, foram definidos objetos persistentes utilizando o *ObjectStore Persistent Storage Engine* (PSE).

O *ObjectStore Persistent Storage Engine* é um banco de dados mono-usuário que fornece um armazenamento de transações orientadas a objetos, permitindo a escrita de aplicações Java que criam e consultam dados persistentes. Permite que um programa execute as funções de criação, abertura, encerramento e destruição de banco de dados, funções de início, conclusão e aborto de transações, leitura e escrita de um *root database*, que fornecem pontos de navegação iniciais sobre objetos persistentes. Além disso, permite o armazenamento de objetos em banco de dados e recuperação e alteração destes objetos [18].

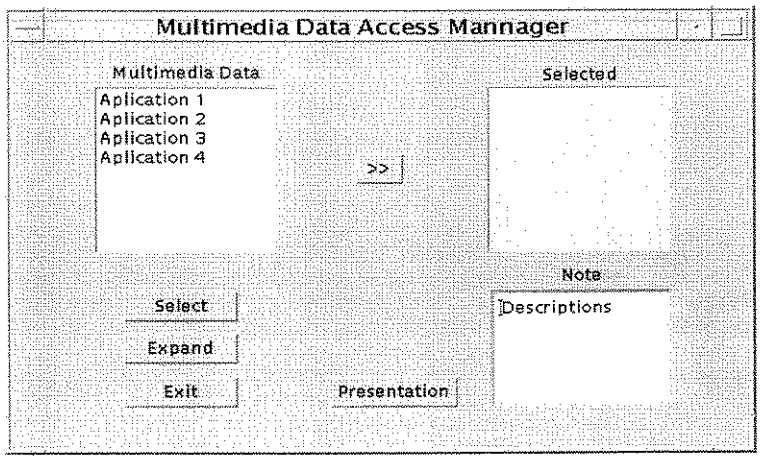


Figura 4.3: Interface principal do sistema.

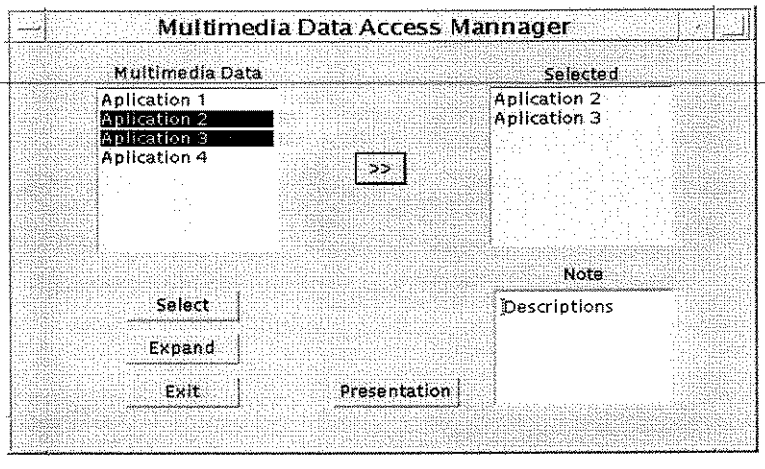


Figura 4.4: Interface principal do sistema com a seleção de apresentações.

Uma das funcionalidades implementadas nesse sistema é a apresentação de uma imagem estática, conforme Figura 4.5, no padrão de compressão JPEG ou GIF, com a possibilidade da apresentação de um áudio associado, iniciada pelo cliente ou pela própria aplicação. Desta forma, se o fluxo de áudio é disparado no momento em que a imagem é apresentada, pode-se manter a imagem até o final da apresentação do áudio, para depois dar continuidade à aplicação.

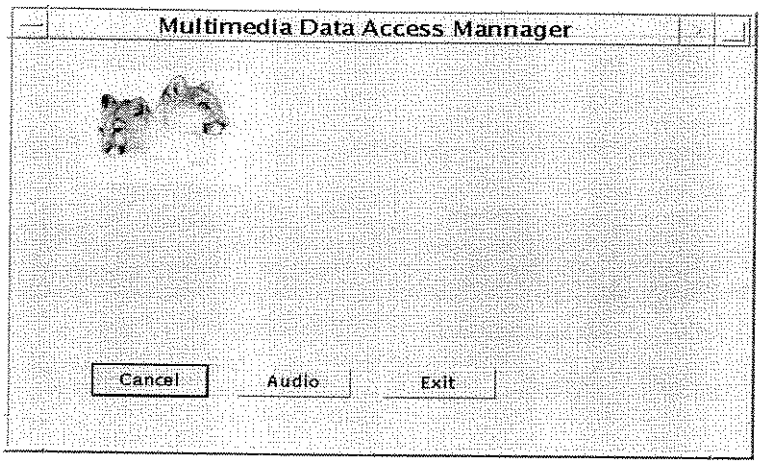


Figura 4.5: Interface de apresentação de uma imagem estática.

Uma outra possibilidade é a apresentação de uma sequência de imagens JPEG em um mesmo *frame* também com a possibilidade de iniciar de forma sincronizada ou não um arquivo de áudio associado. Além disso, a aplicação permite que a apresentação dos quadros sejam controladas, podendo parar a animação e reiniciá-la em um outro instante. Esta possibilidade é mostrada nas Figuras 4.6 e 4.7

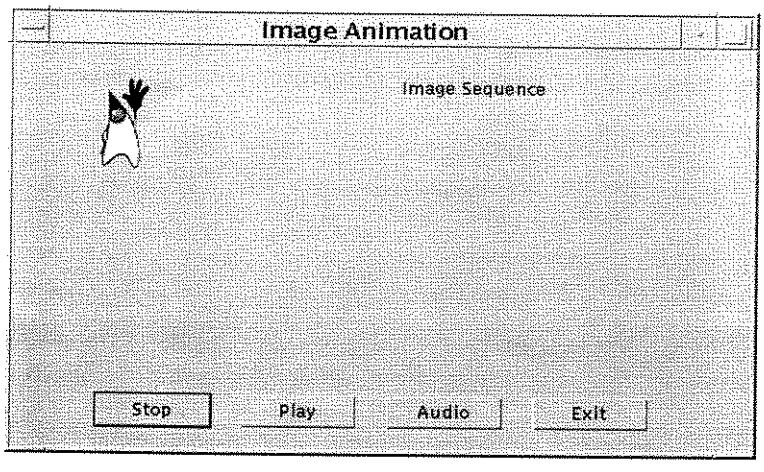


Figura 4.6: Interface de apresentação de um quadro de uma animação.

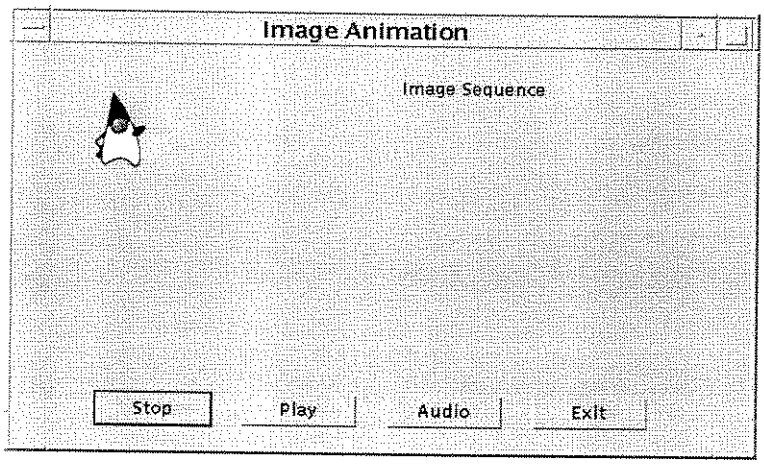


Figura 4.7: Interface de apresentação de outro quadro de uma animação.

Uma última possibilidade é a apresentação de um vídeo em movimento, com ou sem áudio, sendo neste caso utilizado um recurso já disponível para apresentação de arquivos MPEG, o *mpegplay*, como ilustrado na Figura 4.8. Os comandos de controle desta apresentação são fornecidos pela própria ferramenta. Neste caso também arquivos de áudio podem ser disparados simultaneamente à apresentação do vídeo.

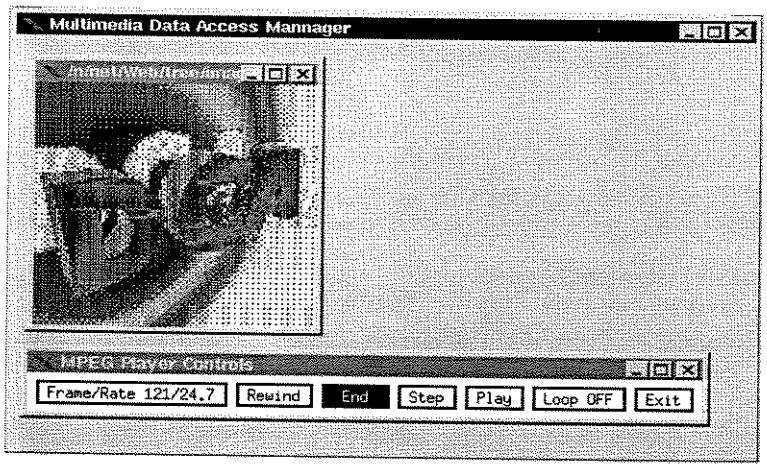


Figura 4.8: Interface de apresentação de um vídeo usando o recurso do *mpegplay*.

Desta forma, é apresentado através das interfaces do sistema, uma aplicação onde o gerenciador de acesso aos dados multimídia se faz necessário. A funcionalidade suportada

pelo gerenciador constitui um conjunto de métodos que pode ser utilizado para a montagem de apresentações multimídia usando objetos registrados em um banco de dados.

4.4 Cenário de uso

De acordo com as interfaces definidas para o sistema gerenciador de acesso a dados multimídia, a interação do usuário com o sistema é feita através de uma tela inicial, onde são mostradas as apresentações já disponíveis no banco de dados, conforme a Figura 4.3. A partir destas apresentações já definidas, o usuário tem a opção de verificar quais os objetos pertencentes a cada uma e, com isso, alterá-las de acordo com suas necessidades. A partir da seleção de uma apresentação, conforme mostrado na Figura 4.4, a aplicação já pode ser iniciada.

A aplicação desenvolvida envolve a apresentação de arquivos de imagem estática, imagem em movimento e som. Em uma primeira instância foi desenvolvida uma interface para a apresentação de uma imagem estática, com a possibilidade de iniciar a apresentação de um arquivo de áudio. Essa interface é apresentada na Figura 4.5.

Posteriormente, foi desenvolvida uma interface que permite a apresentação de uma sequência de imagens, com definição de comportamentos possíveis através de componentes gráficos. Finalmente, foi desenvolvida uma interface para a apresentação de um vídeo, com a utilização da ferramenta *mpegplay*.

Todas as interfaces definidas suportam a inclusão de arquivos de áudio durante a apresentação. Entretanto, mecanismos de sincronização entre mídias distintas ainda não foram implementados.

Capítulo 5

Resultados Obtidos

"É preciso ter confiança na capacidade que cada pessoa tem de ensinar a si mesma."
(Paulo Coelho)

Através deste trabalho pode-se verificar que quando é necessário uma decisão entre MHEG e sistemas baseados em HTML, as respectivas aplicações e seus contextos devem ser cuidadosamente levados em consideração. Sistemas baseados na linguagem HTML, trabalhando em uma arquitetura cliente-servidor, com o protocolo http *(hypertext transfer protocol)*, não conseguem atender os requisitos de aplicações multimídia e hipermídia devido a limitações do protocolo, que não garante sincronização entre dados ou mesmo a continuidade de uma apresentação já iniciada.

Para contornar este problema, o projeto foi desenvolvido usando-se Java [22]. Nesta aplicação os dados necessários são transmitidos antes do início da apresentação, de forma a minimizar os problemas relacionados ao seu desempenho. Em contra-partida, pode-se ter problemas com relação à capacidade de armazenamento no cliente sendo, em alguns casos, mais apropriado que os dados necessários para uma apresentação sejam enviados de forma gradativa, de acordo com o desempenho da aplicação.

Outro aspecto importante a ser ressaltado é que o uso da linguagem Java facilitou a adição de algumas funcionalidades ao engenho MHEG, impossíveis de serem conseguidas nas implementações tradicionais [12]. A partir do momento em que um engenho com funcionalidades MHEG pode ser construído usando-se a tecnologia Java em geral e classes Java puderam ser definidas para suportar MHEG, tornou-se viável o uso de Java como

base tecnológica para o desenvolvimento, troca e apresentação de documentos multimídia, através da Internet.

5.1 Aspectos de implementação

A idéia principal do projeto de um gerenciador de acesso a dados multimídia partiu da necessidade em se criar um sistema que tivesse algumas facilidades apresentadas pelo padrão MHEG, onde os dados pudessem ser integrados naturalmente às aplicações atuais como a *Word Wide Web* (WWW), além de poderem ser executados em diferentes plataformas.

O desenvolvimento deste projeto foi feito utilizando-se a linguagem Java, principalmente pelo fato desta linguagem ter trazido benefícios que acabaram por facilitar a implementação do sistema. Através dela, é possível transmitir *scripts* escritos em HTML com códigos Java incluídos, sendo necessário apenas a existência de um interpretador Java no engenho de apresentação para que a aplicação seja executada.

A linguagem Java pode funcionar como uma linguagem de programação ou como uma linguagem *script*. Como linguagem de programação, é usada para desenvolver uma aplicação completa, passando pelas etapas de compilação, depuração e testes. Como linguagem *script*, permite que códigos Java sejam incluídos em, por exemplo, uma página HTML através de *applets* [15].

Um ambiente de programação Java é composto normalmente de um *kit* de desenvolvimento de aplicações Java e um *browser* compatível com esta linguagem (recomendável). Atualmente, estes *browsers* tem se tornado complexos devido à quantidade de padrões existentes de dados multimídia. A linguagem Java, por sua vez, possue algumas características principais que precisam ser destacadas.

Java possue uma proximidade com outras linguagens de programação, tendo a aparência de C ou C++, embora a filosofia da linguagem seja diferente, sendo importante mesmo assim uma constante comparação com estas outras linguagens. Possui também características herdadas de muitas outras linguagens de programação como objective-C, Smalltalk, Eiffel e Modula-3. Muitas das características desta linguagem não são total-

mente novas, mas a união de todas elas certamente é.

De acordo com as características desta linguagem, um programa Java é compilado e desta compilação é gerado um *bytecode*, que pode ser comparado a instruções de máquina, mas não de uma máquina real, principalmente pelo fato da utilização de Java visar a independência da máquina.

Além disso Java foi desenvolvida para criar aplicações portáteis. O *bytecode* gerado pelo compilador para uma aplicação específica pode ser transportado entre plataformas distintas que suportem Java (Solaris, Windows-NT, Windows-95, entre outras). Não é necessário recompilar um programa para que ele seja executado em uma máquina em um sistema diferente, ao contrário do que acontece, por exemplo, com programas escritos em C e outras linguagens. Esta portabilidade é importante para a criação de aplicações na heterogênea Internet. Um exemplo bem clássico é o fato do próprio compilador Java ter sido escrito em Java, de modo que com isso possa ser portado para qualquer sistema que possua o interpretador de *bytecodes*.

Java é uma linguagem orientada a objetos, onde a portabilidade é sua característica principal, obtida de maneira inovadora com relação ao grupo atual de linguagens orientadas a objeto. Com relação à implementação, suporta herança simples e permite que herança múltipla seja implementada através do uso da definição de interfaces onde, dessa forma, uma classe pode herdar o comportamento de sua superclasse além de oferecer uma outra implementação para uma interface definida.

Java é uma linguagem considerada segura e além disso possui uma funcionalidade muito importante de coleta automática de lixo, evitando com isso que erros comuns cometidos pelos programadores, quando são obrigados a gerenciar memória, aconteçam. A eliminação do uso de ponteiros, em favor do uso de vetores, objetos e outras estruturas substitutivas traz benefícios em termos de segurança. O programador não tem permissão de acesso à memória que não pertence ao seu programa, além de não ter chances de cometer erros comuns com ponteiros e uso indevido de aritmética de ponteiros. Estas medidas são particularmente úteis quando se pensa em aplicações comerciais desenvolvidas para a Internet.

A presença de mecanismos de tratamento de exceção torna as aplicações mais robustas, facilitando o processo de controle da execução, tentando com isso evitar que programas

travem ou "abortem" sem uma possível justificativa. O uso de tratamento de exceção é útil, principalmente em situações de falhas de transmissão e formatos incompatíveis de arquivos.

Java suporta concorrência, permitindo a criação de vários *threads* de execução, característica particularmente útil nos ambientes distribuídos em que as aplicações Java são usualmente suportadas.

Por fim, Java é uma linguagem considerada eficiente, pois como foi criada para ser usada em computadores pequenos, exige pouco espaço e pouca memória. Java é muita mais eficiente que a maioria das linguagens *script*, embora seja mais lenta que C ou C++. Com a evolução da linguagem serão criados geradores de *bytecodes* cada vez mais otimizados que trarão as marcas de desempenho da linguagem mais próximas das de C++ e C. Já existem ferramentas que permitem traduzir rapidamente *bytecodes* para códigos nativos, como meio para acelerar o desempenho. São os chamados compiladores *Just in Time* (JIT).

Apesar das vantagens apresentadas, Java ainda apresenta alguns problemas, principalmente com relação à sincronização. Neste caso este problema pode ser resolvido através da utilização dos recursos de *multithreads* inerentes à linguagem.

Devido a estes problemas é necessário uma análise bem detalhada dos benefícios conseguidos com a linguagem Java, para poder saber até que ponto estas vantagens suprimem as perdas existentes com esta utilização.

5.2 Relacionamento com MHEG

A partir da definição das funcionalidades pertencentes ao sistema, foi feito um mapeamento com relação ao engenho MHEG, para mostrar a equivalência entre os dois. O resultado mostra como as funcionalidades definidas para este projeto suportam equivalência com o engenho MHEG, podendo ser utilizada como uma proposta de mapeamento para a construção de um engenho MHEG a partir do gerenciador de acesso a dados multimídia.

Para a aplicação é necessário a criação de um objeto apresentador para a mídia, que pode ser definido utilizando o *Media Type*, tipo existente na especificação MHEG. A partir da definição deste objeto, é necessário a criação de um objeto responsável por mostrar onde a mídia será apresentada. Neste caso são utilizados os métodos *new()*, para a criação do

objeto e o método *delete()* para a destruição do objeto após a apresentação. Os aspectos relacionados a objetos de controle da mídia não são relevantes neste caso, uma vez que é necessário a criação de um engenho de apresentação para a execução da aplicação e este engenho, por sua vez, será responsável pela criação de uma instância de um apresentador para a mídia solocitada.

Para a identificação do nome da mídia que será apresentada é necessário a definição de uma variável através do método *setStreamChoice(media name, attributes)*. Após a definição das mídias a serem apresentadas, pode-se obter informações com relação ao nome da mídia e desta forma recuperar suas características através de métodos como *originalsize()*, *duration()* e *volume()*.

O processo de inicialização do aquivo de mídia a ser executado é feito por dois métodos: *preparationStatus()*, que pode estar nos estados de *ready()* ou *not ready()*, e a função *prepare()*, que torna o objeto disponível para o engenho. A execução de qualquer arquivo pode ser suspensa, mediante a utilização do método *destroy()*, e um novo apresentador pode ser criado, mediante a chamada de um novo engenho de apresentação. A apresentação das mídias solicitadas é feita em uma área apropriada para a execução, criada a partir de componentes visuais, pela definição de um *Generic Space (GS)*.

O padrão MHEG permite a utilização de funções que podem ser gerenciadas pelo usuário como, no caso de apresentação de vídeo, realizar o controle de brilho e contraste. Para isso possui métodos como *setSpeed()*, *setAudibleVolume()*, que fornece uma interface para ajuste do áudio obtido, *AudioVolumeRange()* e outros.

Através da chamada de um engenho de apresentação é iniciada a execução de um apresentador. O início desta apresentação também pode ser feito em um ponto específico da mídia, sendo necessário neste caso a utilização do método *set current temporal position()*. Antes do início da apresentação, a mídia é preparada através do método *prepare()* e a qualquer momento a execução pode ser interrompida através do método *destroy()*.

A gerência de tempo e sincronização também é possível de ser controlada através da utilização do indicador *Synchro*, que pode ser definido como serial ou paralelo. Além disso, pode-se definir um ponto no fluxo onde um apresentador pode reiniciar, parar ou iniciar uma apresentação. Para isso são utilizados métodos como *get inicial temporal position()* e *get terminal temporal position()*. Através do método *set current temporal position()* uma

posição de leitura dentro de um fluxo de mídia pode ser ajustado, além de existir a possibilidade de sincronizar dois apresentadores, estabelecendo prioridades para os apresentadores através do método *set presentation priority()*. O indicador *Synchro*, neste caso, é usado para iniciar dois apresentadores de forma sincronizada.

As funcionalidades principais equivalentes à utilização da proposta do engenho MHEG incluem a definição de objetos apresentadores para a mídia, objetos para mostrar onde a mídia será apresentada, obter informações com relação ao nome da mídia e definição de controles a serem utilizados durante a reprodução. O início da apresentação de uma mídia também é suportado, bem como funções de parada e destruição da apresentação.

A especificação MHEG também permite a montagem de uma área para a execução da mídia, bem como a criação de um novo objeto apresentador e a utilização de funções que podem ser gerenciadas pelo usuário como, no caso da apresentação de um vídeo, poderem ser controladas características como brilho e contraste e, no caso de áudio, volume.

Com relação à sincronização, a especificação MHEG utiliza o mecanismo de prioridades para estabelecer a ordem de execução dos processos.

Algumas funcionalidades do gerenciador, que não são suportadas pelo padrão MHEG, incluem a definição de variáveis para receber a URL onde se localiza a mídia ou mesmo a aplicação, a definição de objetos para mostrar e controlar o processo de *download* que está sendo efetuado. O gerenciador também não possue funções para liberar os recursos que estão sendo utilizados pela aplicação, tendo que estes serem liberados pelo próprio programador, pois o MHEG não despende nenhuma atenção para este fato.

5.3 Implementação usando Java

Na implementação do gerenciador de acesso foi necessário, inicialmente, a implementação de uma classe *dataBase*, que fornece métodos para criação, instanciação e consulta a um banco de dados. Esses métodos são implementados com a utilização da biblioteca de classes do PSE. A partir desta implementação, objetos persistentes são criados, como possíveis aplicações prontas para serem visualizadas. Cada uma destas aplicações é formada por vários objetos multimídia, que podem ser combinados entre si, gerando novas

apresentações.

O método principal é responsável por criar e iniciar o banco de dados. Durante o processo de criação são definidas instâncias dos objetos definidos, montando cada aplicação a ser apresentada. Após este processo, a consulta ao banco de dados já pode ser efetuada.

A implementação da interface principal do sistema é feita através de uma classe *mainMenu*, onde são definidos todos os componentes gráficos presentes na interface. Entre estes componentes, destacam-se uma área de texto, onde as informações existentes no banco de dados são apresentadas e os botões responsáveis por eventos específicos. De acordo com o processo de manipulação de eventos em Java, para cada componente é criado uma classe com a definição do evento relacionado.

De acordo com os objetos presentes em cada aplicação, foram definidas classes específicas responsáveis pela apresentação de cada um deles. Para a apresentação de áudio, foi definida uma classe *playAudio*, responsável por direcionar o arquivo selecionado para seu respectivo dispositivo de saída.

Na representação de imagem, foi considerada a possibilidade de estar apresentando imagens estáticas e animadas. As imagens animadas são construídas através da apresentação de uma sequência de imagens estáticas.

Para a representação de uma imagem estática, foi definida uma classe *imageShow*, responsável por estabelecer uma região gráfica e apresentar a imagem. Além disso, é criado um componente que permite que um arquivo de áudio seja apresentado, de acordo com a necessidade do usuário. Na representação de uma animação, foi definido uma classe *animation*, responsável por realizar a apresentação de um conjunto de imagens estáticas. Essa apresentação é definida em uma classe *imageLoop* que recupera todas as imagens em um vetor antes de iniciar a animação. Para esta apresentação também são definidos componentes que permitam a inclusão de arquivos de áudio, além de componentes que manipulem a animação, desempenhando funções de início e pausa da animação.

A apresentação de um vídeo foi feita com a definição de uma classe *showVideo*, que recupera o arquivo a ser apresentado e utiliza o *mpegplay* para fazer a apresentação. Os controles da apresentação do vídeo, como pausa, volta ao início, parada da apresentação, entre outros, são fornecidos pelo *mpegplay* e podem ser manipulados pelo usuário.

As bibliotecas Java utilizadas para a implementação do gerenciador de acesso a dados multimídia foram java.awt, java.lang, java.awt.image, java.awt.event, java.net. A biblioteca do PSE utilizada foi a COM.odi. A Figura 5.1 apresenta o diagrama da hierarquia de classes definidas para a implementação do projeto, usando a notação do modelo de objetos da *Object Modeling Technique* (OMT) [21].

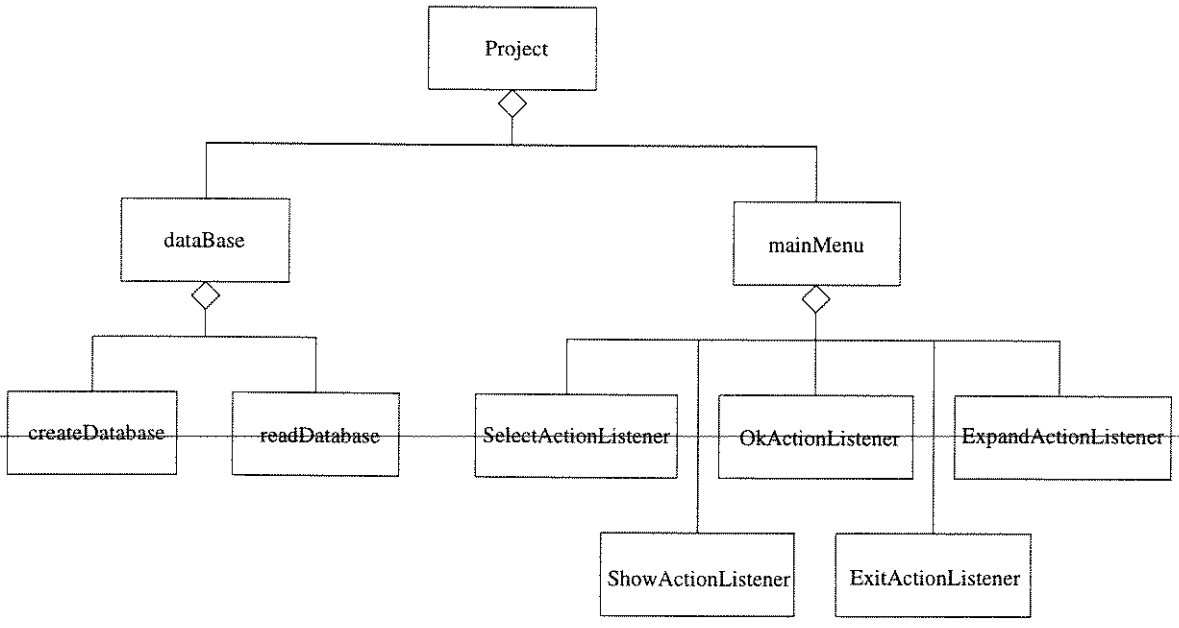


Figura 5.1: Hierarquia das classes utilizadas no desenvolvimento do sistema.

A Figura 5.2 apresenta o diagrama da classe *showActionListener*. Da mesma forma diagramas das classes *showVideo*, *imageShow* e *animation* são mostrados nas Figuras 5.3, 5.5 e 5.4, respectivamente.

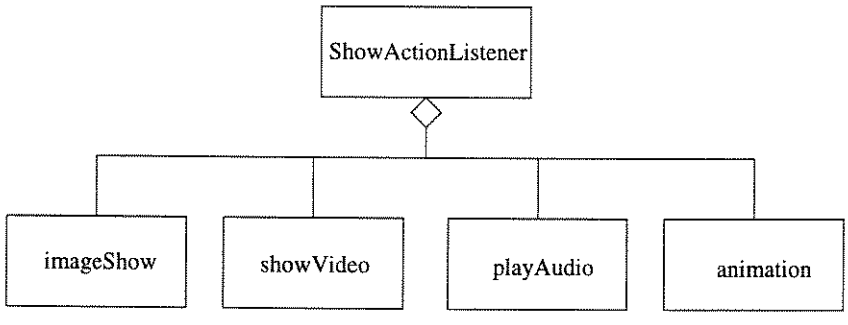


Figura 5.2: Hierarquia da classe *showActionListener*.

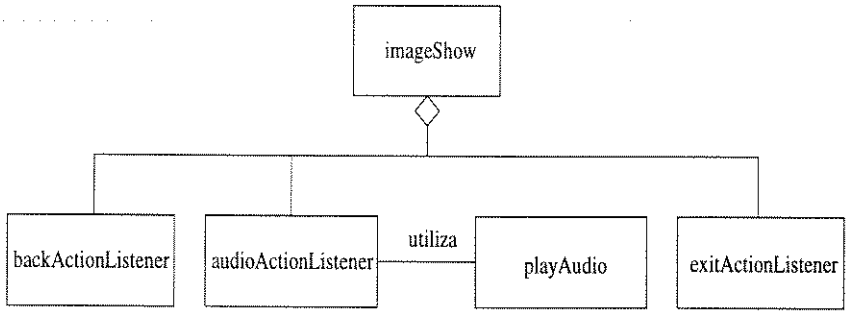


Figura 5.3: Hierarquia da classe *imageShow*.

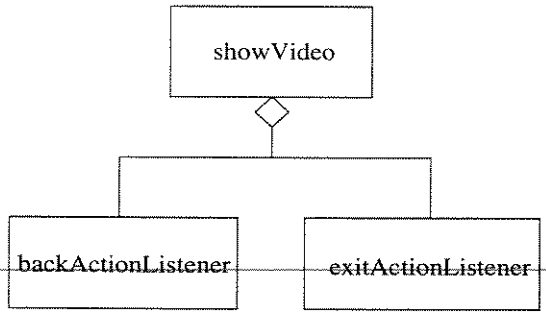


Figura 5.4: Hierarquia da classe *showVideo*.

A Tabela B.1 mostra, de forma detalhada, como cada funcionalidade pertencente ao sistema pode ser implementada através da utilização do *Java Development Kit* (JDK1.1), junto com suas bibliotecas criadas com o objetivo de facilitar o desenvolvimento de aplicações.

5.4 Implementação distribuída

A programação distribuída envolve a criação de um conjunto de componentes de *software* sendo executados em um número de computadores conectados à rede. A tecnologia distribuída é desenvolvida para fornecer uma estrutura de comunicação que abstrai níveis mais baixo de comunicação. A tecnologia distribuída orientada a objetos adiciona um *framework* para encapsulamento e reuso, acelerando-se com isso o desenvolvimento e integração entre as diversas aplicações.

A *Object Management Group's Common Object Request Broker Architecture* (OMG

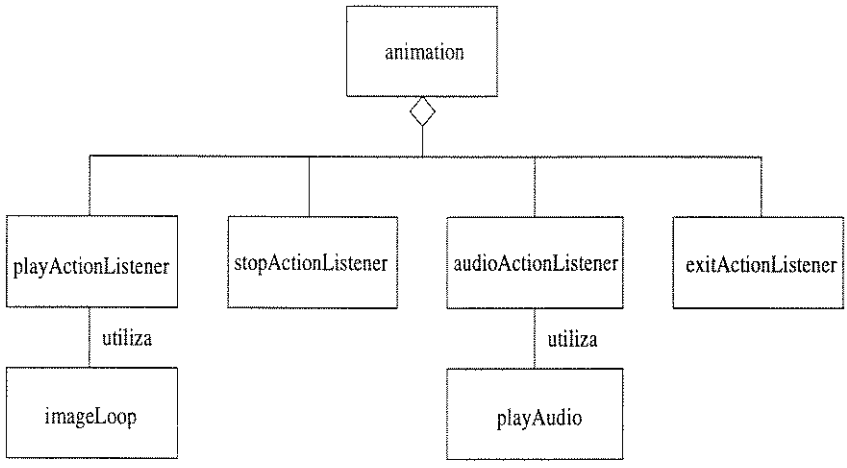


Figura 5.5: Hierarquia da classe *animation*.

CORBA) é uma especificação padronizada para a criação de sistemas orientados a objetos distribuídos. Esta especificação define a funcionalidade de um *Object Request Broker* (ORB), que permite que programadores definam objetos que possam ser acessados através da rede, através de uma linguagem própria para definição de interfaces [14].

Uma das diversas implementações do CORBA existentes é o OrbixWeb, que permite que programadores Java construam aplicações usando o padrão CORBA para comunicação inter-objetos. Por essa razão, o sistema foi desenvolvido usando OrbixWeb.

Em uma programação distribuída, é possível implementar diferentes objetos usando diversas linguagens de programação. Para permitir que a interação entre objetos ocorra, é comum abstrair a funcionalidade fornecida por cada objeto através da definição de interfaces, utilizando para isso a *Interface Definition Language* (IDL), linguagem padrão definida pela OMG para especificação de interfaces.

Após ter definidos as interfaces dos objetos, o programador pode passar para a etapa de implementação do objeto usando a linguagem de programação mais apropriada.

No desenvolvimento do projeto do gerenciador de acesso a dados multimídia, foram definidas interfaces para a comunicação da aplicação com o banco de dados, com o sistema de arquivos e interfaces relacionadas à execução de métodos específicos de apresentação de áudio, de vídeo e de sequência de imagens.

Depois de definidas as interfaces IDL, os arquivos são compilados usando-se o compilador IDL OrbixWeb. Essa compilação resulta na geração de vários tipos de dados Java que correspondem à definição IDL, de acordo com o mapeamento OrbixWeb de IDL para Java. O compilador IDL produz um conjunto de tipos de dados Java que permite um cliente fazer referência a um objeto através da interface definida, e ainda um outro conjunto de tipos de dados que permite um objeto desta interface ser implementado em um servidor.

Esta compilação produz sete arquivos Java (cinco classes e duas interfaces), de acordo com a Figura 5.6, a partir da definição da IDL. De acordo com os requisitos de Java, cada classe pública ou interface é colocada em um arquivo fonte único com extensão java. Cada um destes arquivos são localizados em um diretório que segue o mapeamento em Java de nomes de pacotes para estruturas de diretório.

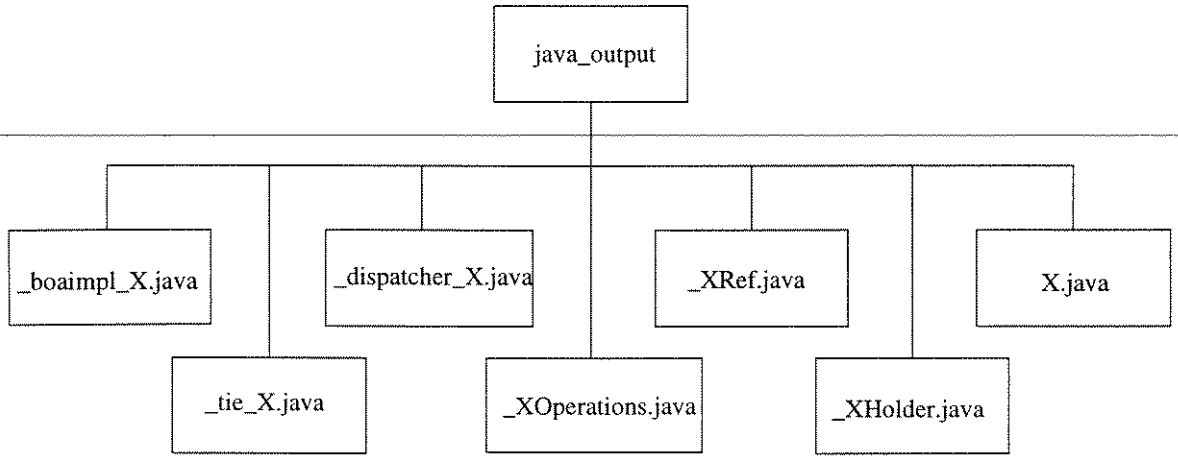


Figura 5.6: Estrutura dos arquivos gerados pelo compilador IDL.

Do lado do cliente, o arquivo *XRef.java* possui a definição de uma interface que descreve a visão do cliente. O arquivo *X.java* é uma classe Java que implementa os métodos definidos na interface *XRef.java* e fornece funcionalidades que permitem aos clientes invocar métodos a serem passados para o servidor.

Do lado do servidor tem-se o arquivo *XHolder.java*, classe que define um tipo *Holder* para a classe X. Não é necessário fazer nenhuma alteração. O arquivo *XOPerations.java*

Depois de definidas as interfaces IDL, os arquivos são compilados usando-se o compilador IDL OrbixWeb. Essa compilação resulta na geração de vários tipos de dados Java que correspondem à definição IDL, de acordo com o mapeamento OrbixWeb de IDL para Java. O compilador IDL produz um conjunto de tipos de dados Java que permite um cliente fazer referência a um objeto através da interface definida, e ainda um outro conjunto de tipos de dados que permite um objeto desta interface ser implementado em um servidor.

Esta compilação produz sete arquivos Java (cinco classes e duas interfaces), de acordo com a Figura 5.6, a partir da definição da IDL. De acordo com os requisitos de Java, cada classe pública ou interface é colocada em um arquivo fonte único com extensão java. Cada um destes arquivos são localizados em um diretório que segue o mapeamento em Java de nomes de pacotes para estruturas de diretório.

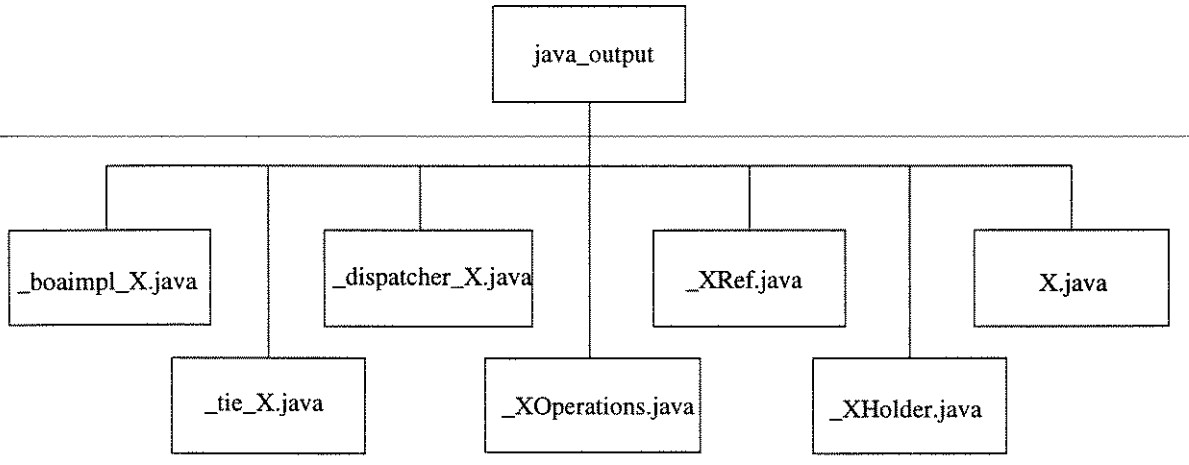


Figura 5.6: Estrutura dos arquivos gerados pelo compilador IDL.

Do lado do cliente, o arquivo *XRef.java* possui a definição de uma interface que descreve a visão do cliente. O arquivo *X.java* é uma classe Java que implementa os métodos definidos na interface *XRef.java* e fornece funcionalidades que permitem aos clientes invocar métodos a serem passados para o servidor.

Do lado do servidor tem-se o arquivo *XHolder.java*, classe que define um tipo *Holder* para a classe X. Não é necessário fazer nenhuma alteração. O arquivo *XOPerations.java*

é a definição de uma interface Java que faz o mapeamento dos atributos e operações da definição IDL para métodos Java. Estes métodos devem ser implementados por uma classe no servidor.

Os arquivos _boaimpl_X.java e _tie_X.java são classes Java abstratas que permitem o desenvolvimento do lado do servidor para implementar a interface usando uma das duas abordagens do OrbixWeb (*tie* ou *boa*). O arquivo _dispatcher_X.java, por sua vez, é uma classe Java usada internamente pelo OrbixWeb para enviar requisições que chegam ao servidor, para implementar objetos. Não é necessário fazer qualquer alteração ou até mesmo entender sua sintaxe.

A partir da geração destes arquivos, as implementações do lado do cliente e do servidor são efetuadas. Os servidores tem o objetivo principal de fornecer objetos para o uso do cliente e de outros servidores. Quando um objeto chama uma operação em outro objeto, este passa a ser considerado o cliente e o responsável por atender o chamado, o servidor. Os clientes por sua vez podem implementar um subconjunto de funcionalidades normalmente associadas com os servidores. Isso é uma característica que dá uma certa flexibilidade às aplicações.

Desta forma, foi implementada uma aplicação cliente, como por exemplo *animationClient.java*, responsável pelos métodos relacionados à interação do usuário com a aplicação, e um servidor, *animationServer.java*, responsável por fornecer os métodos possíveis de serem utilizados pelas aplicações clientes.

Para a interação com o banco de dados e o repositório, foi necessário definir os dados e meta-dados relevantes para a aplicação, como o nome da mídia a ser apresentada, seus atributos e características relacionadas a sua apresentação, como tamanho, número de quadros por segundo (no caso de vídeo) e amostras por segundo (no caso de áudio).

5.5 Java Media Player

Java Media Player é uma ferramenta que possibilita o trabalho com apresentações de mídias, encapsulando a origem das mídias identificadas por uma URL ou ainda trabalhando com arquivos locais, além de permitir que o programador sincronize mídias baseadas em

tempo. Esta ferramenta inclue o *Java Media Framework*, interface de programação que permite incorporar tipos de dados multimídia em aplicações ou *applets* Java.

Java Media Player engloba três diferentes domínios de aplicação, apresentação, captura de mídias e conferências, e ainda três categorias de desenvolvedores, projetistas de aplicações, aperfeiçoadores de características e projetistas de tecnologias. O *Java Media Framework* fornece uma área de trabalho independente de plataforma para construção de apresentações de mídias, sendo projetado para suportar diferentes tipos de mídias como MPEG, QuickTime, AVI, WAV, AU e MIDI [11].

O mapeamento apresentado no Apêndice B mostra as funcionalidades do sistema gerenciador de acesso a dados multimídia com relação ao *Java Media Player* (JMP). É de se esperar que quando esta ferramenta estiver disponível, o processo de desenvolvimento destes tipos de sistemas seja muito mais rápido e fácil, uma vez que as principais funcionalidades são completamente suportadas pela ferramenta.

Capítulo 6

Conclusão

"Algo só é impossível até que alguém duvide e acabe provando o contrário".
(Albert Einstein)

Até hoje, pesquisas e desenvolvimentos na área de multimídia e sistemas distribuídos têm se concentrado em esquemas de compressão e codificação, para tipos de mídias individuais e, em linguagens de alto nível, para estruturas de documentos.

A troca e reprodução de documentos multimídia passaram a requerer formatos padronizados não somente para o conteúdo das mídias, mas também para a estrutura de uma apresentação multimídia interativa e complexa. O padrão MHEG suporta tais formatos, permitindo distribuição de informações multimídia complexas em sua forma final e reprodução de sistemas heterogêneos. Com esta nova tecnologia, novas aplicações multimídia poderão ser desenvolvidas.

O ambiente de trabalho para disseminação e apresentação multimídia mais utilizado atualmente é, com certeza, a *World Wide Web*. O sucesso deste ambiente deve-se a várias razões, como a infra-estrutura disponível na Internet via protocolo TCP/IP, sobre os quais o protocolo de hipertexto *http* é baseado. A disponibilidade livre de clientes (*browsers*) e servidores (*httpd's*) acabou impulsionando a difusão da *Web*, sendo que seu principal sucesso foi em função da facilidade em se criar hiper-documentos utilizando a linguagem HTML, para apresentações não interativas de hipertextos primários com figuras anexadas, como diagramas e gráficos, para explicações de conteúdo técnico. Entretando, a *Web* não é o meio mais adequado à apresentação multimídia devido à limitação dos protocolos utilizados.

O padrão MHEG é projetado para atender os requisitos de aplicações e serviços multimídia, rodando em estações de trabalho heterogêneas trocando informações em tempo real. Características do MHEG incluem estruturas de interação e especificação para troca em tempo real de dados multimídia, composição e sincronização de dados multimídia no tempo e espaço, ligação entre elementos de objetos multimídia compostos e reuso de dados multimídia em diferentes contextos.

Utilizando essa filosofia foi desenvolvido um sistema que permite a integração de vários tipos de mídias, podendo ser executado em ambientes heterogêneos sem a necessidade de alterações no projeto.

Os mapeamentos feitos com relação às três abordagens, utilizados durante o projeto, apresentam soluções viáveis e práticas para alguns casos levantados durante a implementação. Esses resultados podem ser avaliados nas tabelas existentes no Apêndice B.

6.1 Trabalhos futuros

Como sugestão para trabalho futuro, pode-se citar a adequação desta aplicação para um sistema aberto distribuído, onde o processo de negociação do cliente com a aplicação será feito através de uma plataforma CORBA acrescida de características de processamento em tempo real e com interfaces integradas à linguagem Java. Um outro protocolo de transporte (do tipo *stream*) é necessário para a transmissão efetiva dos dados multimídia, para garantir os parâmetros de qualidade de serviço. A requisição do serviço e todas as outras funções de gerência continuam sendo realizadas através da plataforma CORBA, facilitando com isso a comunicação entre as entidades de forma transparente, como mostrado na Figura 6.1.

Nesta arquitetura, inicialmente é feita uma requisição de um canal pelo cliente ao servidor. Através deste canal, será estabelecida a ligação. O servidor irá instanciar o objeto controlador de canal (CC) e os objetos *stub* (S), *binder* (B) e *protocol* (P) locais ao nó do produtor e do consumidor. As operações de controle serão requeridas ao controlador de canal que interage com os objetos do canal através do ORB. O fluxo dos objetos é transmitido pelo *stub* através de um protocolo específico.

A comunicação entre o cliente e a aplicação (o processo de negociação) é feito através

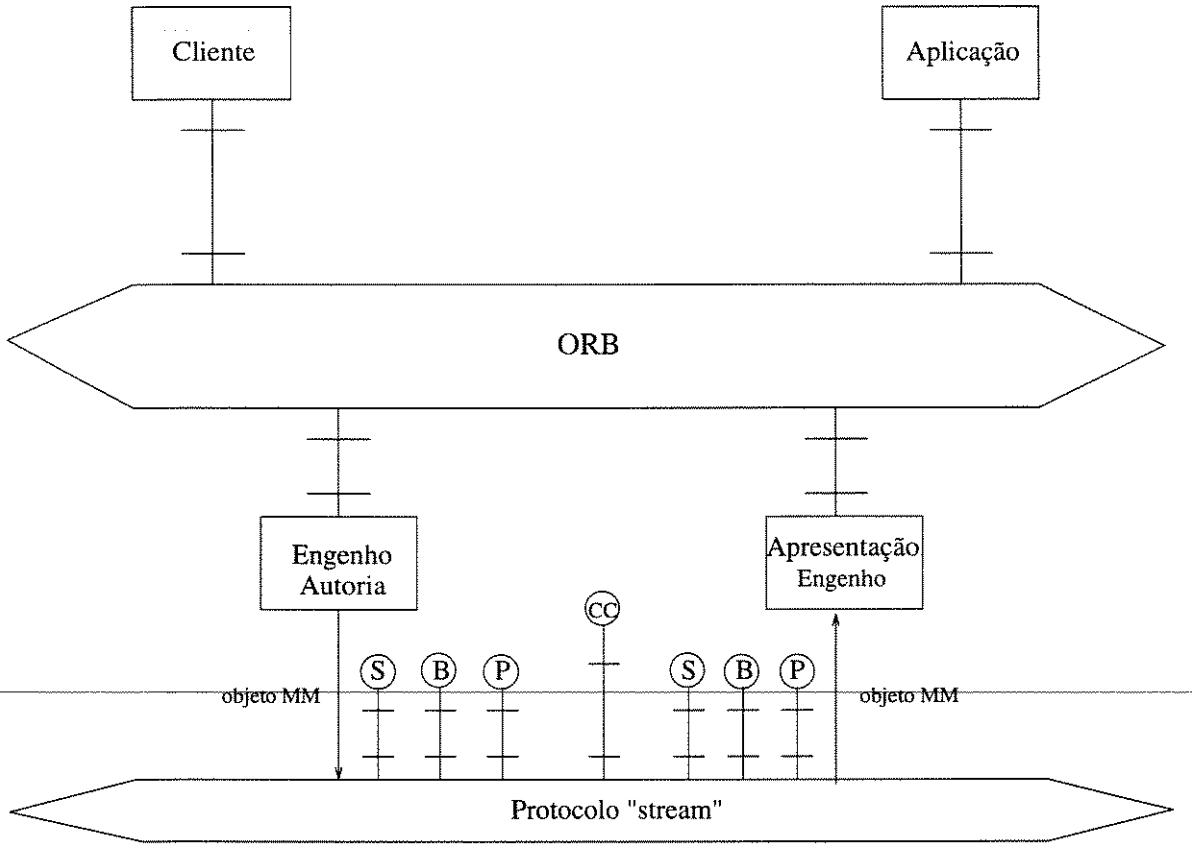


Figura 6.1: Estrutura para comunicação do cliente com a aplicação.

do ORB, enquanto que a transmissão efetiva dos dados é feita usando-se outro protocolo, que garante uma reserva de banda mais adequada à transmissão de dados multimídia.

Neste esquema está representada apenas uma ligação entre dois objetos. Entretanto, podem ser definidos canais ligando vários produtores a vários consumidores. Neste caso seria necessário instanciar um conjunto S/B/P para cada ligação ponto-a-ponto. Tanto o engenho de autoria como o de apresentação também se comunicam através deste canal, utilizando um protocolo dedicado para transmissão dos dados multimídia.

Outra sugestão a ser feita é com relação a utilização da ferramenta *Java Media Player* para auxiliar no desenvolvimento deste tipo de projeto. A substituição do PSE pela nova versão do banco de dados orientados a objetos *ObjectStore*, que apresenta interface para a linguagem Java, permitirá suportar o acesso eficiente e concorrente ao banco de

dados.

Apêndice A

Modelo Funcional

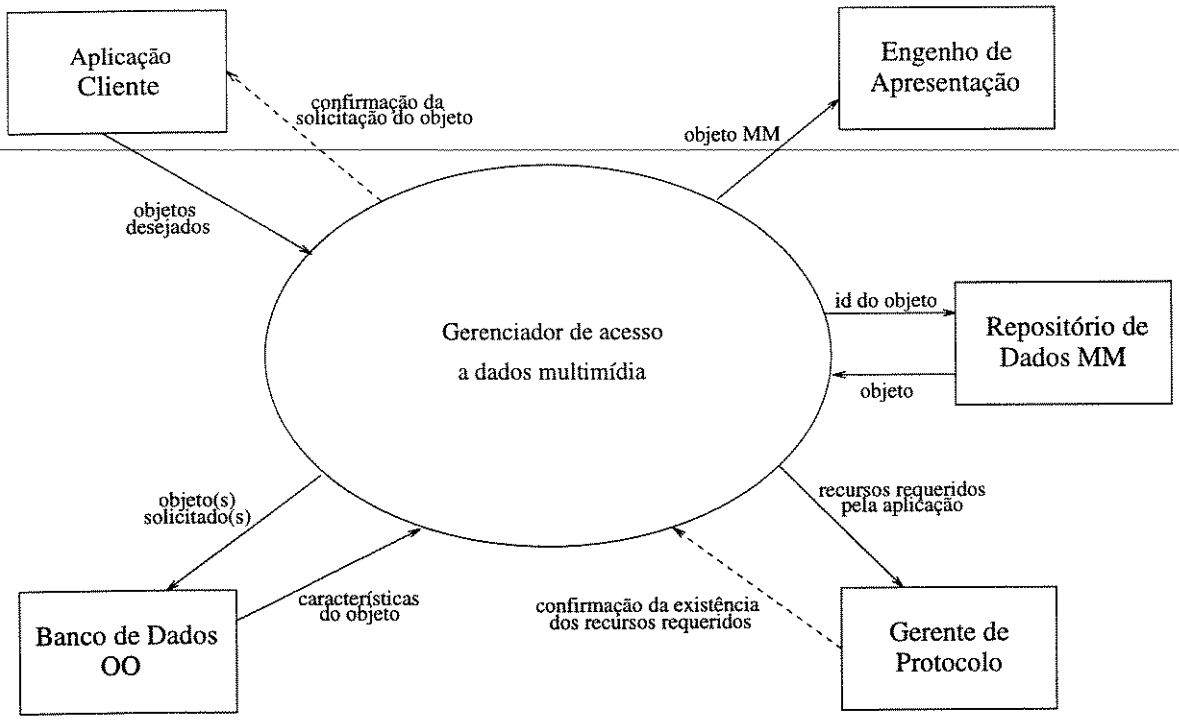


Figura A.1: M.F. nível 0 - Especificação das entidades externas que interagem com o gerenciador.

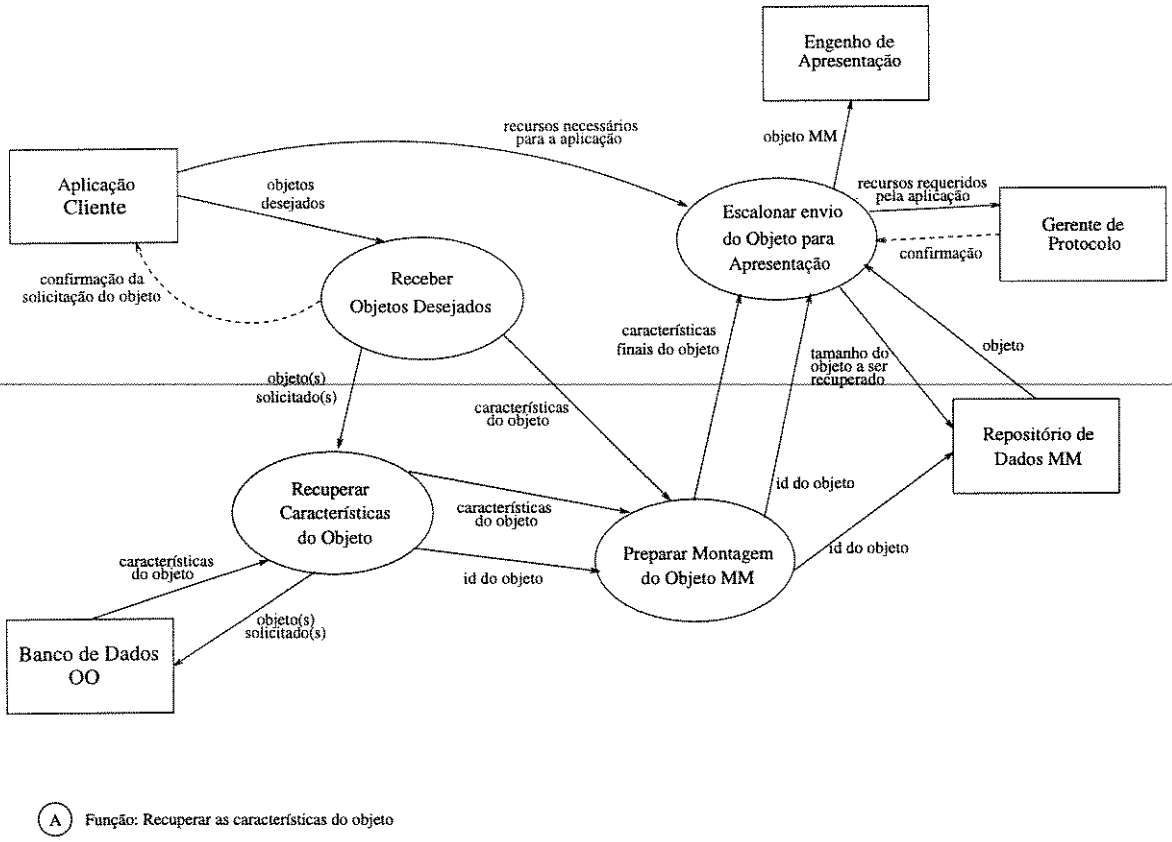


Figura A.2: M.F. nível 1 - Definição das funcionalidades principais do gerenciador.

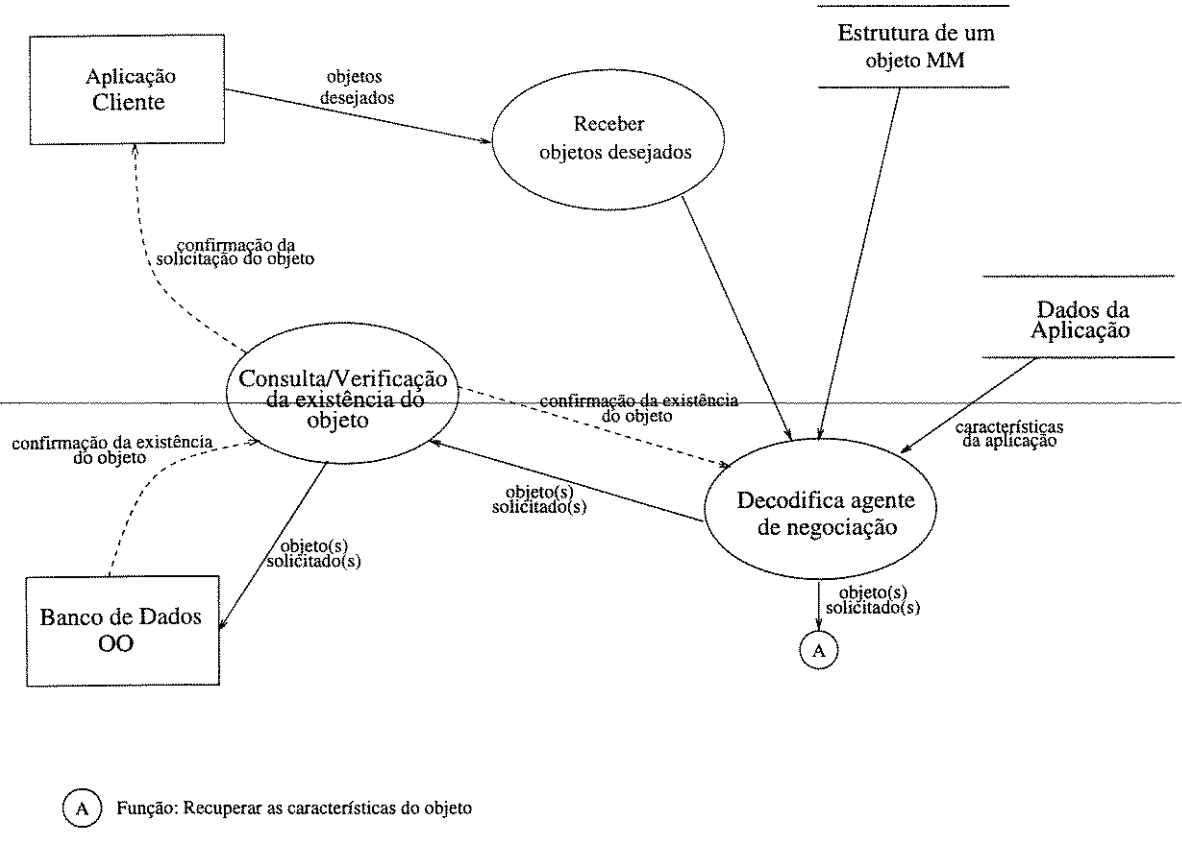
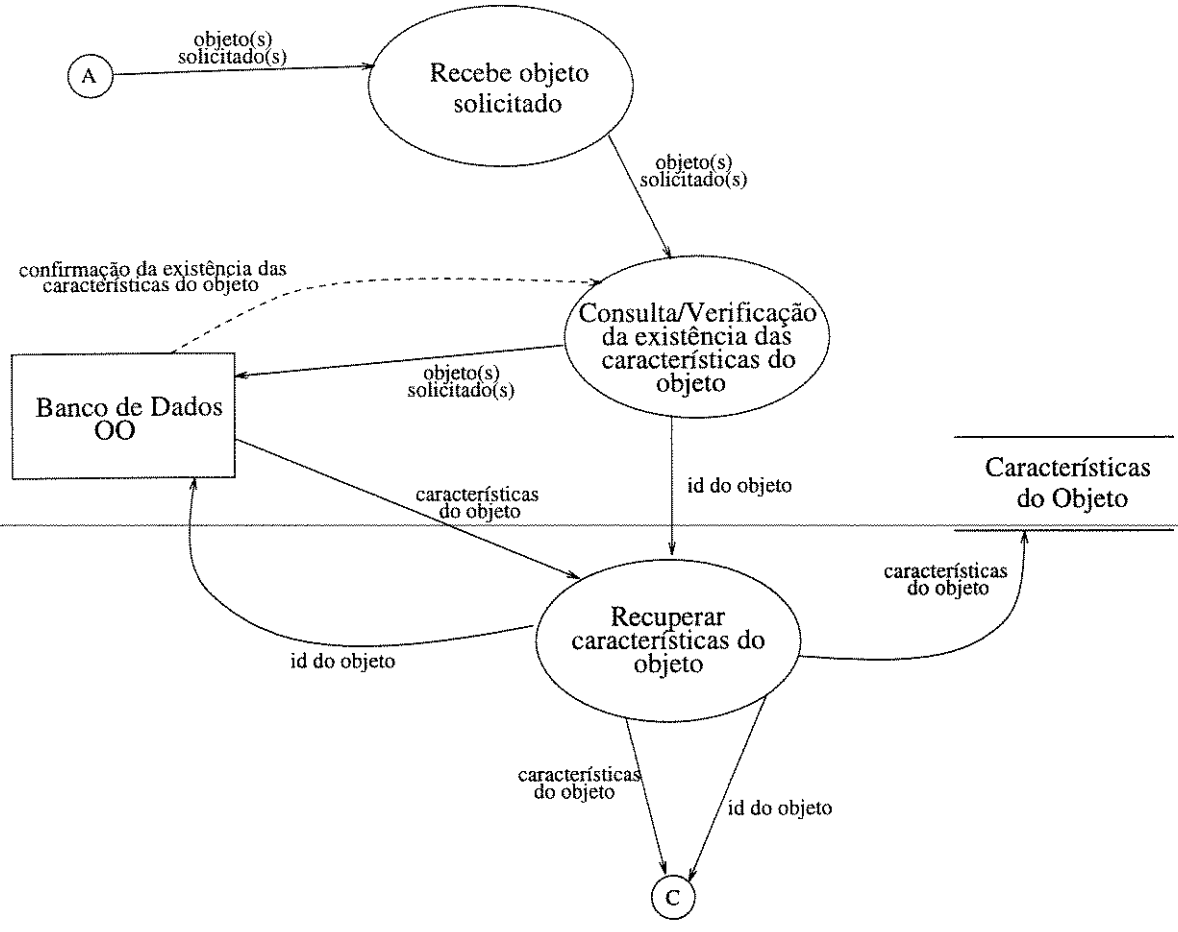


Figura A.3: M.F. nível 1.1 - Definição das funcionalidades do processo de receber objetos desejados.



A Função: Recuperar características do objeto

C Função: Preparar montagem do objeto MM

Figura A.4: M.F. nível 1.2

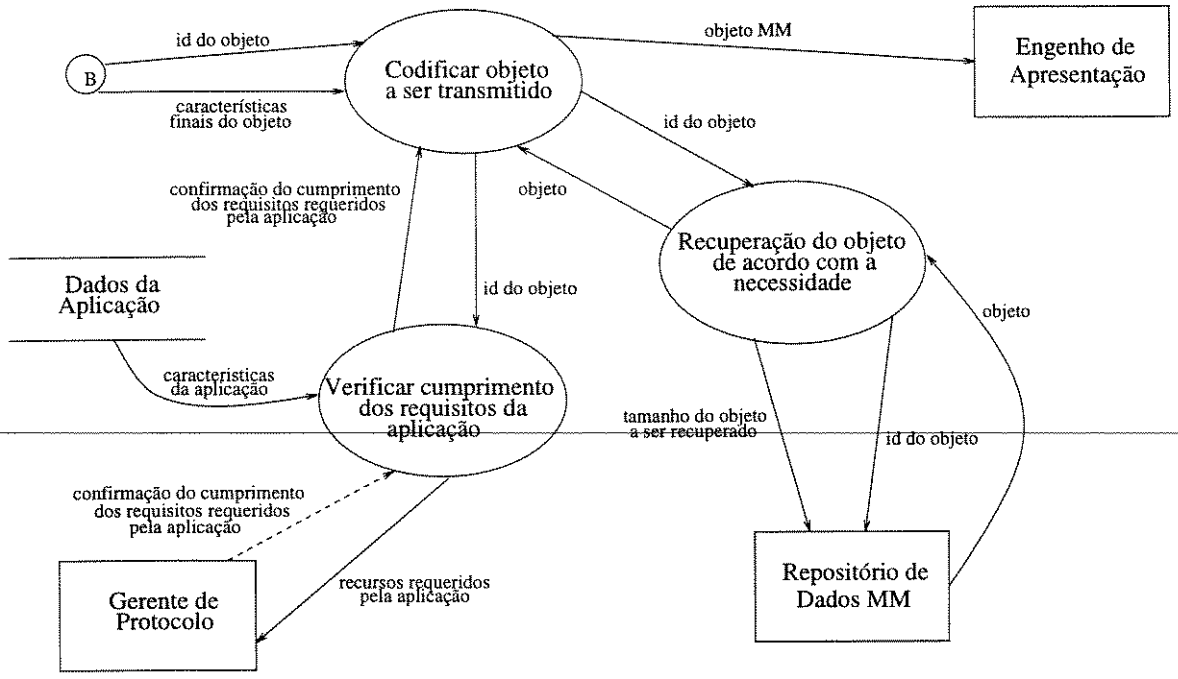


Figura A.5: M.F. nível 1.3 - Definição das funcionalidades do processo de preparar montagem do objeto MM.

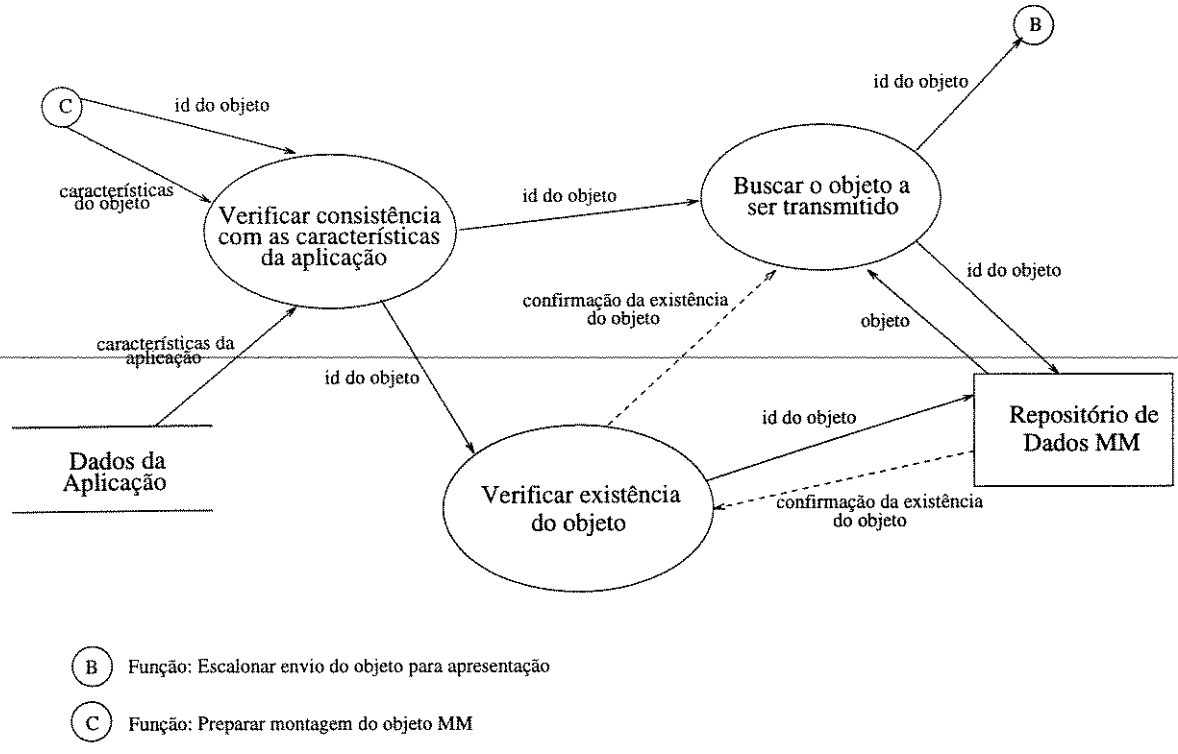


Figura A.6: M.F. nível 1.4 - Definição das funcionalidades do processo de escalonar envio do objeto para apresentação.

Apêndice B

Mapeamento das Funcionalidades

Neste apêndice, apresenta-se o mapeamento de conjuntos de funcionalidades suportadas pelo gerenciador de acesso para as funcionalidades do *Java Development Kit* (B.1) e do *Java Media Player* (B.2).

B.1 Mapeamento das funcionalidades para JDK

Tabela B.1: Mapeamento das funcionalidades usando o JDK 1.1

FUNCIONALIDADE	JDK 1.1
Definição de um objeto apresentador para a mídia	*Media Tracker object.*
Definição de um objeto para mostrar os componentes onde a mídia será apresentada	biblioteca java.awt.
Definição de um objeto para definir os controles a serem utilizados durante a reprodução da mídia	através da definição de botões com aventos associados a cada um deles, definidos através da biblioteca *java.awt.event.*

Tabela B.1: (continua)

Funcionalidade	**JDK 1.1**
Definição de uma variável para receber o nome da mídia a ser apresentada como parâmetro	*String image* = diretório onde se encontra a mídia solicitada.
Definição de uma variável para receber a URL relativa a mídia a ser apresentada	*String source location* = "http://.....".
Definição de uma variável para receber a localização da mídia a ser apresentada	*Image im = getImage(url);*.
Obtêm informações com relação ao nome da mídia a ser apresentada	*getParameter()*; *getName()*; *getSource()*; *getProperties()*;.
Cria uma URL a partir do nome do arquivo e da URL da aplicação	URL url = *new* URL("http://.....");.
Cria uma instância de um apresentador para a mídia solicitada	feito através da definição de um construtor para o método que apresenta uma determinada mídia.
Responde aos eventos de outros apresentadores através da implementação do método *ControllerUpdate()*, chamado automaticamente quando o apresentador dispara um evento	feito através da definição de classes para cada evento disparado. Estes eventos são tratados individualmente.
Inicia a execução do arquivo de mídia	*player.start()*; ou *Thread.start()*;.
Para a execução do arquivo de mídia que está em execução	*player.stop()*; ou *Thread = null*;.
Montagem de uma área para a execução da apresentação da mídia, utilizando componenentes visuais	utilizando a biblioteca *java.awt*.
Montagem dos componentes de controle para a execução da mídia	definição de botões com eventos associados.

Tabela B.1: (continua)

Funcionalidade	**JDK 1.1**
Checar se o processo de *download* foi completado	feito através da comparação do tamanho do *buffer (buffer.lenght)* com o tamanho do arquivo (*getLength()*).
Reinicializa a mídia quando sua apresentação chega ao fim e começa a apresentar de novo	processo feito manualmente utilizando um botão para parar a apresentação e outro botão, que dispara novamente a execução da apresentação.
Criação de um novo apresentador	nova chamada do construtor da classe de apresentação.
Mostrando os componentes visuais de um apresentador	método da biblioteca awt, *show();*.
Mostrando os componentes de controle de um apresentador	método da biblioteca awt, *show();*.
Utilização de funções que podem ser gerenciadas pelo usuário, como, no caso de apresentação de vídeo, controle de brilho e contraste	através da biblioteca *awt*, utilizando os métodos *brighter()* e *darker();*.
Iniciar um apresentador	*Thread.start();*.
Parar um apresentador	*Thread.stop();*.
Preparação de uma mídia para ser apresentada	*getImage();* método executado antes do início da apresentação.
Gerência de tempo e sincronização	não tem. Os processos são disparados simultaneamente.
Sincronização de apresentadores	através da diretiva *synchronized.*
Iniciar dois apresentadores sincronizados	feito através da chamada dos dois processos separadamente, utilizando a diretiva *synchronized.*

B.2 Mapeamento das funcionalidades para JMP

Tabela B.2: Mapeamento das funcionalidades usando *Java Media Player*

FUNCIONALIDADE	*Java Media Player*
Definição de um objeto apresentador para a mídia	*Player player = null;*. O tipo *Player* é definido no pacote java.media do *Java Media Framework*
Definição de um objeto para mostrar os componentes onde a mídia será apresentada	*Component visualComponent = null;*.
Definição de um objeto para definir os controles a serem utilizados durante a reprodução da mídia	*Component controlComponent = null;*.
Definição de um objeto para mostrar como está o processo de *download* da mídia solicitada	*Component progressBar = null;*.
Definição de uma variável para receber o nome da mídia a ser apresentada como parâmetro	*String mediaFile = null;*.
Definição de uma variável para receber a URL relativa a mídia a ser apresentada	URL url = null;. Este tipo está definido no pacote *java.net.*
Definição de uma variável para receber a localização da mídia a ser apresentada	*URL codeBase = getDocumentBase();*.
Obtêm informações com relação ao nome da mídia a ser apresentada	*mediaFile = getParameter("FILE").*
Cria uma URL a partir do nome do arquivo e da URL da aplicação	url = *new URL(codeBase, mediaFile).*
Cria uma instância de um apresentador para a mídia solicitada	*player = Manager. createPlayer(url).*

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Adiciona o apresentador como escutador de eventos de outros apresentadores	*player.addControllerListener(this).*
Responde aos eventos de outros apresentadores através da implementação do método *ControllerUpdate*, chamado automaticamente quando o apresentador dispara um evento	*public synchronized void controllerUpdate (ControllerEvent event).* Responde a um tipo de evento através do método *RealizeCompleteEvent*, que quando é disparado, faz com que sejam mostrados os componentes do apresentador. A aplicação espera que um evento seja disparado para mostrar os componetes visuais e o painel de controle *default* para ser colocado junto com a apresentação da aplicação.
Inicia a execução do arquivo de mídia	*player.start();.*
Para a execução do arquivo de mídia que está em execução	*player.stop();.*
Desaloca os recursos que estão sendo utilizados pela aplicação antes de sair totalmente	*player.deallocate();.* Este método libera os recursos que foram utilizados por um apresentador, de forma que isso não prejudique a execução de um novo apresentador.
Montagem de uma área para a execução da apresentação da mídia, utilizando componenentes visuais	*player.getVisualComponent().*
Montagem dos componentes de controle para a execução da mídia	*player.getControlPanelComponent().*

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Apresentação de uma barra de progressão para verificar como está sendo o processo de *downloading* da mídia a ser apresentada	*CachingControlEvent e = (CachingControlEvent) event.*
Controle do processo de *download* de uma mídia, para que se possa ter uma idéia de quanto tempo o processo vai demorar	feito através da interface *CachingControl.*
Utilização da barra de controle do processo de *download*	é necessário implementar a interface *ControllerListener* e executar os processos de *CachingControlEvents* no *ControllerUpdate.*
Checar se o processo de *download* foi completado	*getContentProgress = getContentLength.*
Reinicializa a mídia quando sua apresentação chega ao fim e começa a apresentar de novo	*player.setMediaTime(0).* *player.start();.*
Criação de um novo apresentador	É feito através da interface *Manager*, através da utilização do método *createPlayer.* É usado a própria URL da mídia especificada para criar um apresentador apropriado. Para o cliente, a criação deste apresentador é feito de forma totalmente transparente.

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Mostrando os componentes visuais de um apresentador	Estes componentes são responsáveis por mostrar os dados da mídia a ser apresentada. Tanto um apresentador de áudio como um ícone *speaker* ou caráter animado devem ser associados com um componente visual. Para apresentar um componente visual é necessário: pegar o componente usando a função *getVisualComponent*; adicionar este componente no espaço de apresentação do *applet* ou janela da aplicação.
Utilização do painel de controle *default* do *Java Media Player*	pode ser chamado pela função *getControlPanelComponent.*
Utilização de funções que podem ser gerenciadas pelo usuário, como, no caso de apresentação de vídeo, controle de brilho e contraste	uso da interface *Control*, através dos métodos *getControls* instanciados no apresentador. *Control[]controls = player.getControls();.*
Utilização de uma API para ajustar áudio obtido	feito através dos métodos da classe *GainControl* que herda as características da interface *Control.* Para se ter este controle, usa-se a interface *getPlayerGainControl()* que fornece métodos para ajuste de volume, como *setLevel()* e *setMute().*

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Iniciar um apresentador	as interfaces *Player()* e *Clock()* definem dois métodos diferentes para iniciar e parar um apresentador. Na interface *Player()*, o método *start()* diz ao apresentador para iniciar a apresentação o mais cedo possível. Se necessário este método prepara o apresentador para iniciar pelo desempenho das operações de *prefetch()* e *realize()*. Na interface *Clock()* o método *syncStart()* diz a um apresentador para sincronizar a mídia de acordo com um tipo particular. É usado para sincronizar a apresentação de vários apresentadores em um mesmo instante de tempo.
Iniciar a apresentação em um ponto específico da mídia	primeiro é necessário especificar o ponto do fluxo onde se deseja iniciar a apresentação através do método *setMediaTime()*, depois o método *Start()* pode ser chamado.

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Parar um apresentador	existem três situações onde um apresentador pode parar, que são: quando o método *stop()* é chamado, quando for especificado o tempo de parada e quando o apresentador estiver executando sem dados. Para que uma parada seja feita em um tempo específico, pode-se usar a função *setStopTime()* para indicar quando um apresentador deve parar. Isso acontece quando o tempo da mídia passa sobre o tempo especificado de parada.
Gerência dos estados de um apresentador	as transações são controladas através dos métodos de *realize()*, *prefetch()*, *start()*, *deallocated()* e *stop()*.

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Preparação de uma mídia para ser apresentada	A maioria das mídias não podem iniciar instantaneamente, pois algumas condições de hardware e de software devem ser estabelecidas, como por exemplo, alocação de *buffers* para armazenamento dos dados, estabelecimento de conexões de rede para um *download* dos dados e posicionamento do fluxo dos dados de mídia onde a apresentação deverá ser iniciada. Este processo é feito através da interface *ControllerListener()*, onde o método *realize()* é chamado para mover o apresentador para o estado de *Reallizing* e com isso iniciar o processo de realização. Depois, é chamado o método de *prefetch()* para mover o apresentador para o estado de *Prefetching* e iniciar com isso o processo.
Implementação da interface *ControllerListener()*	É uma interface assíncrona para manipulação de eventos gerados por objetos de controle. Para a implementação é necessário: registrar a classe como *listener* através do método *ControllerListener()*. A partir daí deve ser implementado o método *ControllerUpdate()*, normalmente definido como uma série de *if-then-else:*. *if(instance of EventType)* *else if(instance of OtherEventType)...*

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Gerência de tempo e sincronização	em muitos casos, pode-se não querer apresentar somente uma mídia do começo ao fim, pode-se desejar apresentar vários fluxos ou ainda fazer a sincronização de um único fluxo. As interfaces *Clock()* e *TimeBase()* definem o mecanismo para gerência de tempo e sincronização.
Gerência de tempo	é feito através da interface *TimeBase()* e não pode ser transformado ou reinicializado. O *systemTimeBase* é dirigido pelo *Clock* do sistema e é acessado pelo método *Manager.get.SystemTimeBase()*.
Definição de um ponto no fluxo onde o apresentador pode reiniciar, parar ou iniciar uma apresentação	feito através do *player's media time*
Setar a posição de leitura dentro de um fluxo de mídia	é feito através da setagem do *player's media time*, usando-se o método *setMediaTime()*, especificando o tempo em nanosegundos.
Retorna o tempo de mídia corrente do apresentador	através do método *getMediaTime()*. Este tempo é retornado em nanosegundos.

Tabela B.2: (continua....)

Funcionalidade	***Java Media Player***
Sincronização de apresentadores	para sincronizar a apresentação de vários fluxos de mídia, os apresentadores podem ser sincronizados através da associação dos mesmos com o mesmo *timeBase*, usando os métodos *getTimeBase()* e *setTimeBase()* definidos pela interface *Clock()*. Por exemplo, *player1* e *player2*: *player1.setTimeBase()* *(player2.getTimeBase())*.
Iniciar dois apresentadores sincronizados	deve ser chamado o método *syncStart()* em cada um deles, especificando o *TimeBase* em que cada um deve iniciar

Apêndice C

IDL's definidas para o projeto

Neste apêndice são apresentadas as IDL's definidas para o projeto. A IDL abaixo foi definida para chamar o método responsável por recuperar as imagens e fazer a animação dos quadros.

```
-------------------------------------------------------------------------
//  IDL
//  In file animation.idl

module Project{
interface animation{
        //  operations with parameters passed by in, because it goes from
            client to server.
        //  method that makes the image loop.
string imageLoop(in long nimgs);
        //  method that gets all images that will be animated
string getImage(in String imageName);
        // method that starts the animation of an image
        string startAnimation(in string image);
// method that stops the animation of an image
string stopAnimation(in String image);
};
};
```

Mapeamento da IDL animation.idl para Java:

```
// In file _animationOperations.java

Package Project;

public interface _animationOperations {
    public String imageLoopImplementation(int nimgs)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String getImageImplementation(String imageName)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String startAnimationImplementation(String image)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String stopAnimationImplementation(String image)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
}
```

IDL definida para permitir uma interação da aplicação com o banco de dados.

```
//  IDL
//  In file interfacedb.idl

module Project{
interface interfacedb{
        //  operations with parameters passed by in, because it goes from
            client to server.
        //  method that creates database.
string createDatabase(in string dbName);
        //  method that opens database
```

```
string open(in string dbName, in string dbType);
        // method that reads the elements fo a database
        string readDatabase(in string dbName);
// method that gets an element of the database
string getRoot(in string element);
// method that gets the objects of an element
string getChildren(in string element);
};
};
```

Mapeamento da IDL interfacedb.idl para Java:

```
// In file _interfacedbOperations.java

package Project;

public interface _interfacedbOperations {
    public String createDatabaseImplementation(String dbName)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String openImplementation(String dbName, dbType)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String readDatabaseImplementation(String dbName)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String getRootImplementation(String element)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String getChildrenImplementation(String element)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
}
```

IDL definida para executar os métodos principais da aplicação, que interagem diretamente com o bando de dados e com os repositórios.

```
//  IDL
//  In file main.idl

module Project{
interface main{
        //  operations with parameters passed by in, because it goes from
            client to server.
        //  method that starts the presentation of the selected objects
string presentationObject(in string object);
        //  method that selects the presentation needed
string selectPresentation(in string presentationName);
        // method that shows the components of one selected presentation
        string expandPresentation(in string presentationName);
};
};
```

Mapeamento da IDL main.idl para Java:

```
//  In file _mainOperations.java

package Project;

public interface _mainOperations {
    public String presentationObjectImplementation(String object)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String selectPresentationImplementation(String presentationName)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    public String expandPresentationImplementation(String presentationName)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
```

```
}
```

Referências Bibliográficas

[1] D. P. Anderson and G. Homsy. A continuous media I/O server and its synchronization mechanism. *IEEE Multimedia*, 1991.

[2] R. B. Araujo. Plataformas distribuídas no suporte a sistemas multimídia. In *I Workshop em Sistemas Hipermídia Distribuídos*, São Carlos, SP, July 1995.

[3] T. V. Batista, N. L. Rodriguez, R. A. M. Soares, and L. F. G. Soares. Correios eletrônicos multimídia. In *I Workshop em Sistemas Hipermídia Distribuídos*, São Carlos, SP, July 1995.

[4] P. B. Berra, C-Y. R. Chen, A. Ghafoor, and T. D. C. Little. Issues in networking and data management of distribuited multimedia systems. In *Symposium on High-Performance Distributed Computing*, 1992.

[5] T. M. Boudnik. MHEG explained. *IEEE Multimedia*, pages 26–38, Spring 1995.

[6] T. M. Boudnik and Wolfgang Effelsberg. MHEG: An interchange format for interactive multimedia presentations. *IEEE Multimedia*, 1995.

[7] John F. Koegel Buford. Architectures and issues for distributed multimedia systems. In John F. Koegel Buford, editor, *Multimedia Systems*, pages 45–64. ACM Press, 1994.

[8] E. Cardozo, M. F. Magalhães, L. Mendes, and I. L. M. Ricarte. Sistemas multimídia. Apostila de Curso, FEEC/UNICAMP.

[9] André Luiz Vasconscelos Coelho. Caracterização de um serviço de gerência de persistência de objetos multimídia sob um ambiente distribuído. Master's thesis, Universidade Estadual de Campinas — UNICAMP, em andamento.

[10] K. Hofrichter. GLASS — globally accessible services. Technical report, Center of Information Technology, German National Research, GMD FOKUS, March 1995.

[11] Sun Microsystems Inc.and Silicon Graphics Inc. and Intel Corporation. *Java Media*

Players. http://www.dstc.uts.edu.au/java/javamedia/player/index.html, version .95 edition, January 1997.

[12] International Organization for Standardization — International Electrotechnical Comission. *MHEG-6 Working Draft*, March 1996.

[13] International Organization for Standardization (ISO). *Information Technology — Coding of Multimedia and Hypermedia Information*, September 1995. Part 1, ISO/IEC DIS 13522-1.

[14] IONA Technologies. *OrbixWeb Programming Guide*, release 2.0 edition.

[15] The source for java, 1995. http://www.javasoft.com.

[16] Naur José Janzantti Junior. Gerência de dados multimídia em banco de dados orientados a objetos (título provisório). Master's thesis, Universidade Estadual de Campinas — UNICAMP, em fase de conclusão.

[17] T. Leidig. Authoring MHEG presentations with GLASS-Studio. In *International Workshop on Multimedia Software Development*, pages 150–158, Berlin, Germany, March 1996.

[18] P. O'Brien. Java data management using ObjectStore and PSE. Technical report, Object Design White Paper, 1996.

[19] H. Rocha. Java: A linguagem da internet. *LAN Times Brasil*, 1996.

[20] A. A. Rodriguez, M. Fisher, and B. Markey. Scripting languages emerge in standards bodies. *IEEE Multimedia*, 1995.

[21] James Rumbaugh and et al. *Object-Oriented Modeling and Design.* Prentice Hall, 1991.

[22] Sun microsystems inc, bussiness, 1995. http://www.sun.com.

[23] M. T. P. Vieira and M. T. P. Santos. Armazenamento e recuperação de objetos multimídia. In *I Workshop em Sistemas Hipermídia Distribuídos*, São Carlos, SP, July 1995.