Uma Arquitetura Distribuída para o Modelo de Contextos Aninhados com Intercâmbio de Objetos MHEG

Luiz Fernando G. Soares

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

Marco Antonio Casanova

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

Resumo

Os sistemas hipermídia usuais foram desenvolvidos como uma aplicação única e auto-contida, fazendo com que o sistema seja especializado e difícil de ser reutilizado em outras aplicações. Este artigo apresenta um modelo conceitual que provê não apenas um modelo de dados hipermídia em camadas, como uma interface orientada a objetos, permitindo uma independência da plataforma final de exibição. Estes dois componentes combinados vão fornecer uma ferramenta capaz não só de possibilitar a construção de sistemas hipermídia monolíticos, como também introduzir características hipermídia a uma aplicação qualquer. Como uma ferramenta estruturada em camadas dará também a flexibilidade adicional de oferecer diferentes níveis de interface, e ainda permitir que os mecanismos de armazenamento possam ser adaptados aos requisitos de uma aplicação particular. A obediência à proposta de padrão MHEG faz com que todos os objetos gerados possam ser intercambiados pelas mais diversas aplicações que obedeçam ao padrão e, mais ainda, permite a utilização de objetos gerados por estas aplicações.

A implementação de sistemas hipermídia multi-usuários baseados em uma única estação de trabalho dificilmente vai permitir a apresentação das várias mídias em tempo real. Isto vai exigir que o modelo hipermídia possa ser construído em uma arquitetura distribuída. A exigência de uma arquitetura distribuída vem também do fato que muitas aplicações serão desenvolvidas para estações hetereogêneas que serão interconectadas de forma a oferecerem serviços multimídia. Uma arquitetura distribuída em rede para a estrutura hipermídia em camadas é também apresentada no artigo.

1 - Introdução

Nos anos recentes, serviços multimídia têm sido incorporados em várias áreas, como sistemas de educação e treinamento, sistemas para automação de escritório, sistemas de informação e pontos de vendas, etc... . Esta tendência pode ser explicada pela introdução de recursos multimídia em muitos computadores. Padrões como JPEG e MPEG têm permitido a construção de pastilhas que no futuro próximo serão incluídas na placa mãe de um computador, ou mesmo serem absorvidos na pastilha da UCP de um computador, como qualquer outro recurso básico. Por outro lado, a crescente disponibilidade de discos óticos e de meios de transmissão digital, como as B-ISDNs e redes de difusão vêm permitir o intercâmbio de uma grande quantidade de dados.

Neste contexto, muitas aplicações multimídia serão construídas para plataformas heterogêneas, ou para serem interconectadas de forma a oferecer serviços multimídia . Estes serviços usarão uma grande quantidade de objetos estruturados residentes em estações, armazenados em meios digitais, ou recuperados de fontes remotas através de redes. Estes dados representam um investimento significativo, tornando-se imperativo que esta informação permaneça disponível e não seja perdida pela incompatibilidade de estruturas de dados usadas nas aplicações.

Os sistemas hipermídia usuais foram desenvolvidos como uma aplicação única e auto-contida, fazendo com que o sistema seja especializado e difícil de ser reutilizado em outras aplicações. Muitos dos esforços empregados na construção de sistemas hipermídia seguiram esta direção, com exceções dignas de menção, como o NEPTUNE [DeSc86] e o Hypertext Abstract Machine (HAM) [CaGo88]. O enfoque monolítico vai exigir que cada novo sistema que incorpore características hipermídia redesenvolva toda a capacidade de armazenamento e exibição hipermídia. Se cada projetista de sistema tiver que desenvolver todos os mecanismos hipermídia para suas aplicações, certamente ficará inviável estender estes conceitos para diversos novos domínios de aplicação. Como exemplos de aplicações em que hipermídia pode ter um papel importante pode-se citar a manipulação (exibição e autoria) de documentos multimídia não lineares, ferramentas para apoio à engenharia de software, sistemas de teleconferência, etc.

O Modelo de Contextos Aninhados - MCA provê não apenas um modelo de dados hipermídia, como uma interface orientada a objeto. O modelo de dados abrange a estrutura hipermídia e os dados da aplicação. O modelo de interface separa os componentes de dados e de exibição dos objetos, o que permite a construção de interfaces independentes da plataforma final de exibição. Estes dois componentes combinados vão fornecer uma ferramenta capaz não só de possibilitar a construção de sistemas hipermídia monolíticos, como também introduzir características hipermídia a uma aplicação qualquer. Como uma ferramenta estruturada em camadas dará também a flexibilidade adicional de oferecer diferentes níveis de interface, utilizadas pelas aplicações conforme suas necessidades.

A definição de uma estrutura em camadas vai ainda permitir que os mecanismos de armazenamento possam ser adaptados aos requisitos de eficiência e tamanho impostos por uma aplicação particular. A obediência à proposta de padrão MHEG faz com que todos os objetos gerados possam ser intercambiados pelas mais diversas aplicações multimídia e hipermídia que

obedecerão ao padrão e, mais ainda, permite a utilização de objetos gerados por estas aplicações.

Diversos requisitos nortearam o desenvolvimento do modelo e da estrutura hipermídia em camadas:

- O sistema deve operar independente da plataforma: tipo de máquina, sistema operacional, dispositivos de exibição, ferramentas e redes.
- O sistema deve ser independente das aplicações e poder ser estendido para utilização por diversas aplicações.
- O sistema deve prover uma arquitetura aberta, permitindo às aplicações diversos níveis de interface.
- O número típico de usuários é muito grande.
- Os dados são objetos longos espalhados em várias máquinas.
- O sistema não tem conhecimento do conteúdo dos dados armazenados, exceto seu tipo.
- Os objetos intercambiáveis devem seguir um padrão internacional, possibilitando não só a utilização de seus objetos em outros sistemas, mas a utilização de objetos gerados por outros sistemas.
- O sistema deve suportar interatividade em tempo real, incluindo a aquisição de dados multimídia.

A implementação de sistemas hipermídia multi-usuários baseados em uma única estação de trabalho dificilmente vai permitir a apresentação de áudio e vídeo em tempo real, devido à pequena faixa passante de entrada e saída dos dispositivos de armazenamento e ao tamanho destes objetos, que os obrigam a ficar residentes nestes dispositivos. Isto vai exigir que o modelo hipermídia possa ser construído em uma arquitetura distribuída.

A exigência de uma arquitetura distribuída vem não apenas do fato mencionado acima, como também do fato que muitas aplicações serão desenvolvidas para estações hetereogêneas que serão interconectadas para oferecerem serviços multimídia., como por exemplo, trabalho cooperativo, sistemas de mensagem multimídia, videofonia, publicações e livros eletrônicos, sistemas telemáticos audiovisuais para treinamento e educação, jogos e simulação, pontos de vendas, além de novas classes de aplicação multimídia.

Uma arquitetura distribuída para a estrutura hipermídia em camadas é apresentada ao final do artigo, embora sem entrar em detalhes sobre a implementação do sistema de armazenamento, principalmente no que tange às suas características de tempo real, que é assunto de outro trabalho.

2 - Estrutura Hipermídia em Camadas

A arquitetura proposta define quatro interfaces, como pode ser visto pela figura 1.

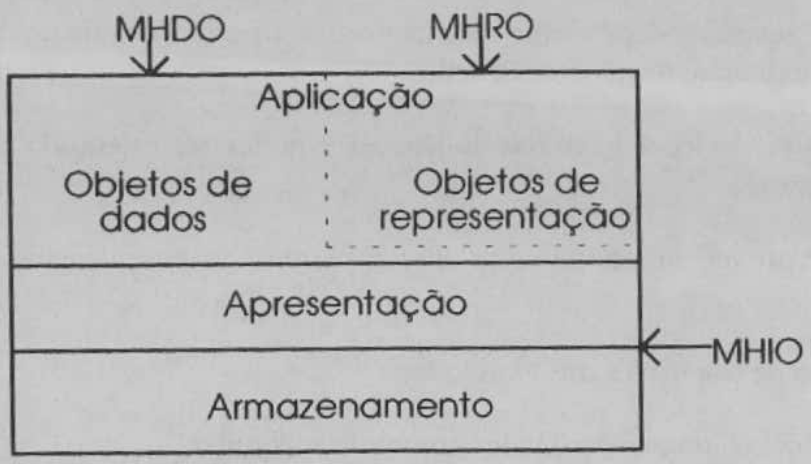

Figura 1 - Estrutura Hipermídia em Camadas

A Camada de Armazenamento oferece o armazenamento persistente para objetos multimídia. Objetos persistentes são as entidades mantidas pelo sistema de armazenamento; eles têm um identificador único, um tipo específico e podem ser compartilhados por várias aplicações. Um objeto persistente contém apenas atributos (identificador, nome do objeto, proprietário, número de versão, palavras chave, direitos de acesso, tamanho, duração, conteúdo de dados, etc). A camada de armazenamento pode ser parcialmente implementada usando qualquer sistema de banco de dados comercialmente disponível, ou um servidor de banco de dados em uma implementação distribuída, como a por nós proposta.

A Interface MHIO (Multimedia Hypermedia Interchangeable Objects) define a representação codificada dos objetos multimídia a serem intercambiados com o sistema de armazenamento, se constituindo na interface entre a camada de armazenamento e a camada de apresentação. A MHIO pode ser usada diretamente por uma aplicação, o que é bastante apropriado quando usada em ferramentas com uma interface com o usuário bem definida.

Cabe à Camada de Apresentação a conversão dos objetos a ela entregues pela camada de aplicação para o formato padrão definido pela MHIO. Qualquer semelhança com as funções do nível de apresentação definido pelo modelo OSI não é mera coincidência.

A Interface de Apresentação define a codificação dos objetos trocados pelas camadas de apresentação e de aplicação. A princípio, os objetos desta interface não são intercambiáveis entre aplicações diferentes e, embora não haja uma restrição explícita, não é esperado que uma aplicação faça uso dos objetos definidos por esta interface.

Os componentes do modelo de dados da Camada de Aplicação são os objetos usuais de um sistema hipermídia, isto é, nós e elos. Esta camada introduz duas novas idéias na arquitetura de aplicações interativas, parecidas com as encontradas em [PuGu91].

A primeira idéia é a definição de **objeto de dados** como uma instância local de objetos persistentes, uma versão destes objetos. Os objetos de dados introduzem um nível extra de abstração, interagindo diretamente com o código específico da aplicação e com sua representação visual ou audível. Ao contrário dos objetos persistentes, que contêm apenas atributos, os objetos de dados contêm, além dos atributos agregados dos objetos persistentes, métodos para manipulação destes atributos, exceto do atributo conteúdo de dados (ainda não interpretável neste nível de abstração), e atributos definidos pelo usuário, que não fazem parte da definição dos atributos padrões da camada de apresentação. Este conjunto de objetos forma a partição denominada conjunto de objetos de dados da camada de aplicação. Esta é a única partição da camada de aplicação que faz interface com a camada de apresentação.

A segunda idéia diz respeito à representação dos objetos de dados. Ela consta de uma abstração de nível ainda mais alto do objeto multimídia, criando um **objeto de representação** (uma nova versão do objeto persistente derivada do objeto de dados) que incorpora ao objeto de dados métodos para manipulação dos atributos definidos pelo usuário, que não fazem parte dos atributos definidos pela proposta de padrão do nível de apresentação, métodos para manipulação do atributo conteúdo de dados (texto, gráfico, áudio, vídeo, etc.), além de converter o atributo conteúdo de dados para um formato mais apropriado, conforme definido pelos métodos que o manipulam. O conjunto de objetos de representação forma a partição da camada de aplicação denominada conjunto de objetos de representação. Esta partição é derivada dos objetos de dados, não derivando seus objetos da camada de apresentação. Mesmo quando a partição conjunto de objetos de dados não é instanciada, como será o caso de várias aplicações, vai existir uma classe abstrata (sem objetos instanciados), a partir da qual os objetos de representação serão derivados, por herança.

Cabe aqui também mencionar, que qualquer semelhança da camada de aplicação com as funções do nível de aplicação definido pelo modelo OSI não é mera coincidência. As aplicações que desejam ter incorporadas características hipermídia, ou visam a construção de sistemas hipermídia monolíticos, podem fazer uso das interfaces MHRO (Multimedia Hypermedia Representation Object) ou MHDO (Multimedia Hypermedia Data Object), além da já mencionada MHIO.

A próxima seção tece maiores considerações sobre o modelo conceitual da implementação em curso da estrutura hipermídia em camadas. A seção 4 é dedicada a uma breve descrição da proposta de padrão MHEG, que vem servindo de base para definição da interface MHIO e da implementação em curso da camada de apresentação. A implementação da arquitetura definida nesta seção 2 em uma arquitetura distribuída é apresentada na seção 5. A seção 6 é reservada às conclusões.

3 - O Modelo de Contextos Aninhados - MCA

O modelo de contextos aninhados fornece o modelo conceitual para a implementação de toda a estrutura descrita no item anterior.

A definição de documentos hipermídia no MCA é baseado nos conceitos usuais de nós e elos. *Nós* são fragmentos de informação e *elos* são usados para a interconexão de nós que mantêm alguma relação.

O modelo distingue ainda duas classes básicas de nós, chamados de nós terminais e nós de contexto, sendo este último o ponto central do modelo. Intuitivamente, um *nó terminal* contém dados cuja estrutura interna é dependende da aplicação e não é parte do modelo. A classe de nós terminais pode ser especializada em outras classes (texto, gráfico, áudio, vídeo, etc.) conforme requerido pelas aplicações.

Um *nó de contexto* é usado para agrupar conjunto de nós terminais ou de contextos, recursivamente. Nós de contexto podem ser aninhados em qualquer profundidade e assim permitem organizar, hierarquicamente ou não, conjunto de nós. eles vão permitir, ainda, um mecanismo para a definição de diferentes visões de um mesmo documento, sintonizadas em diferentes aplicações ou classes de usuários.

Por exemplo, para organizar hierarquicamente um livro texto, pode-se primeiro definir um nó de contexto B, que contém um conjunto de nós representando capítulos do livro e um conjunto de elos indicando a organização dos capítulos, que não é necessariamente em sequência. De forma análoga, para o i-ésimo capítulo, pode-se definir o nó de contexto C_i que contém um conjunto de nós representando as seções do capítulo e um conjunto de elos indicando a organização das seções, e assim sucessivamente. Como ocorre frequentemente, o autor pode definir organizações diferentes para os capítulos, para classes distintas de leitores, definindo outros nós de contexto contendo os mesmos nós de B (os mesmos capítulos), mas com um conjunto diferente de elos indicando as novas organizações. Cada um deste novos contextos podem ser encarados como uma visão diferente do mesmo livro.

O conceito de nós contexto generaliza o conceito homônimo introduzido no sistema Neptune [DeSc86 e DeSc87], que por sua vez se baseou em algumas idéias do PIE [GoBo87]. Ele também generaliza as Webs do sistema Intermedia [Meyr86] e as fileboxes do Notecard [Hala88].

Um elo conecta dois nós. Como o conteúdo de um nó possui uma estrutura interna, que pode ser bastante complexa, os elos também indicam, para cada extremidade, a região onde o elo está ancorado. Por exemplo, para nós texto, a região pode ser simplesmente uma cadeia de caracteres dentro do texto e, para nós de imagem bidimensional, pode ser um retângulo determinado por um par de coordenadas.

Formalmente, uma classe nó terminal possui dois atributos especiais, CONTENTS e MASK. Dado um nó N, o valor de CONTENTS é chamado o conteúdo de N e depende da classe de N (se texto, áudio, vídeo, etc.). O valor de MASK é uma lista de posições ou regiões dentro do conteúdo de N, também dependente da classe de N.

A lista de posições ou regiões de um nó terminal funciona como uma interface externa do nó, no sentido que outros nós não se referirão diretamente a pontos dentro do conteúdo do nó, mas apenas indiretamente através de um índice da lista. Assim, qualquer mudança no conteúdo de um nó não se refletirá nos elos e, portanto, nos nós de contexto que contêm estes elos, se a lista de um nó for sempre atualizada adequadamente.

Uma âncora é um par (N_0,s_0), onde N_0 é um nó terminal ou de contexto, e s é um deslocamento definido como:

- o valor nulo λ; ou
- um inteiro menor ou igual ao comprimento da lista MASK, se N_0 é um nó terminal; ou
- um par (N_1, s_1), onde N_0 é um nó de contexto que contém N_1 e s_1 é um deslocamento.

N_0 é chamada de base da âncora e s_0 seu deslocamento.

Um elo é um par (s,d) de âncoras, onde s é a âncora de origem e d a âncora de destino. As extremidades do elo são as bases de s e d.

A cada elo são associadas condições e ações, conforme descrito na próxima seção.

O modelo introduz uma classe especial de nó, o nó de contexto. O conteúdo de um nó de contexto N é um par da forma (S,L), onde S é um conjunto de nós e L é um conjunto de elos, tais que suas extremidades estão contidas em S. Diz-se que N contém cada nó em S e cada elo em L.

Por exemplo, seja A um nó de contexto que contém um outro nó de contexto B que por sua vez contém dois nós, C e D. Como B contém C e D, um elo conectando estes nós pode, em princípio, ser definido em B como $((C,i),(D,j))$, onde i e j são deslocamentos válidos para C e D, respectivamente. Se se quer criar um elo em A conectando C e D, ele deve ser definido como $((B,m),(B,n))$ e especificado os deslocamentos m e n como (C,i) e (D,j), respectivamente.

Uma vez que o modelo de contextos aninhados engloba todos os objetos definidos pela interface MHIO (content nodes, composite nodes, elos, etc., em conformidade com a proposta de padrão MHEG), ele também vai englobar todas as características e definições destes objetos, como por exemplo os mecanismos de sincronização definidos nos elos, conforme resumido na próxima seção. Para não ser redundante, o artigo deixa para aquela seção a apresentação da proposta MHEG. Nesta seção apenas as definições de atributos e métodos acrescentados pelo MCA à proposta de padrão são apresentados, em conjunto com as definições necessárias ao seu entendimento.

Continuando as definições, uma hiperbase é qualquer conjunto H de nós, tais que, para qualquer nó $N \in H$, se N é um nó de contexto, então todos os nós contidos em N também pertencem à hiperbase H.

O modelo de contextos aninhados permite que diferentes nós de contexto contenham o mesmo nó, e que nós de contexto possam ser aninhados em qualquer profundidade. Assim, é necessário

uma forma de identificar sobre que sequência de contextos aninhados um dado nó está sendo pesquisado e que elos de fato tocam no nó a partir do alinhamento. Este fato é capturado pela noção de perspectiva de um nó e a noção de elos visíveis por um nó a partir de uma perspectiva.

A perspectiva de um nó N é uma sequência $P = (N_1,....,N_m)$, $m \geq 1$, tal que $N_1 = N$ e N_i está contido em N_{i+1}, para $i \in [1,m]$. Como N é implicitamente dado por P, P é referido simplesmente como perspectiva. Um nó M está presente em uma perspectiva $P = (N_1,....,N_m)$ se e somente se $M = N_i$ para algum $i \in [1,m]$.

Seja N_1 um nó e P sua perspectiva. Diz-se que um elo l é visível a partir de P por N_1 com deslocamento i_1, se e somente se:

- l está contido em N_2 e (N_1,i_1) é uma âncora de l; ou
- l é visível a partir de P por N_2 com deslocamento i_2 e o deslocamento i_2 é (N_1,i_1)

Um elo é visível a partir de P por N, se e somente se ele é visível a partir de P por N para algum deslocamento.

O modelo de contextos aninhados define ainda extensões para acomodar a noção de objetos virtuais (nós e elos), isto é, objetos resultantes da avaliação de alguma expressão. A referência [CCLS92] aborda o tópico.

A noção de versão é capturada no MCA introduzindo uma nova classe de contextos chamados contextos de versões.

Os nós contidos em um contexto de versões não precisam ser da mesma classe e são chamados versões correlatas (assim, neste nível de abstração todos os nós passam a ser chamados de versão). Por exemplo, um usuário pode criar em um mesmo contexto de versões dois nós, T e S, que descrevem o mesmo pedaço de texto, exceto que T é uma versão escrita e S uma versão falada.

Um elo $((v_1,i_1),(v_2,i_2))$ no contexto de versões V indica que o conteúdo da versão v_2 foi criado a partir do conteúdo da versão v_1. Diz-se neste caso que v_1 é o ascendente de v_2 e que v_2 é descendente de v_1. Os deslocamentos, neste caso, simplesmente tornam mais precisa a informação sobre qual parte do conteúdo de v_1 gerou que parte de v_2.

O conteúdo de uma versão pode ser construído a partir do conteúdo de várias outra versões, mas não é permitida a derivação cíclica.

Usando a noção de contextos de versões, pode-se de fato capturar duas relações. Primeiro, ao se adicionar dois nós em um mesmo contexto de versões está-se indicando que eles representam versões de um mesmo objeto, sem implicar que um é diretamente ou indiretamente derivado do outro. Segundo, ao se ligar os dois nós, indica-se adicionalmente que o conteúdo de um nó foi derivado do conteúdo do outro, pelo menos em parte.

Para dar suporte a trabalho cooperativo, entendido aqui como o processo de criação ou modificação da hiperbase, ou um subconjunto da hiperbase, por um grupo de usuários, define-se a noção de estado de uma versão.

Uma versão pode estar em um dos seguintes estados: temporário, permanente e obsoleto. Uma versão ao ser criada assume o estado temporário e permanece neste estado enquanto sofre modificações. Quando se torna estável, uma versão pode ser promovida ao estado permanente, quer por um pedido explícito do usuário, quer implicitamente por certas operações oferecidas pelo modelo [CCLS92]. Uma versão no estado permanente não pode ser destruída diretamente, mas o usuário pode torná-la obsoleta, dando a oportunidade para que outras versões que a referenciam ou dela derivadas possam ser notificadas.

Uma versão no estado permanente, chamada versão permanente, tem as seguintes características:

- não pode ser modificada;
- se for um nó contexto, só pode ter versões permanentes ou obsoletas;
- pode ser usada na derivação de novas versões;
- não pode ser destruída diretamente;
- pode ser tornada obsoleta, mas não temporária.

Uma versão no estado temporário, chamada de versão temporária, tem as seguintes características:

- pode ser modificada;
- pode conter versões em qualquer estado, se for um nó de contexto;
- não pode ser usada para derivar novas versões;
- pode ser deletada diretamente;
- pode se tornar permanente, mas não obsoleta.

Uma versão no estado obsoleto, chamada versão obsoleta, tem as seguintes características:

- não pode ser modificada;
- se for um nó contexto, só pode ter versões permanentes ou obsoletas;
- não pode ser usada na derivação de novas versões;
- é automaticamente destruída pelo sistema, quando todas as versões que a referenciam ou que são derivadas dela são obsoletas;
- não pode mudar de estado.

Como consequência das definições, os ascendentes de uma versão no grafo de versões são versões permanentes ou obsoletas. É também digno de menção que estas restrições garantem que um nó de contexto permanente ou obsoleto contém apenas nós permanentes ou obsoletos, o que por sua vez implica que todos os seus elos têm como base apenas nós permanentes ou obsoletos. Porém, estas propriedades não se aplicam a nós de contextos temporários.

O modelo cria também a noção de uma base pública, definida como uma partição da hiperbase, de tal forma que todas as versões da base pública devem ser permanentes ou obsoletas.

Um nó na base pública não pode migrar desta base para uma outra partição da hiperbase, embora possam ser criadas versões destes nós nas outras partições. A recíproca no entanto é verdadeira, nós podem mover de uma partição qualquer da hiperbase para a base pública, desde que seja tornado antes permanente.

Conforme salientado em vários pontos do artigo, o modelo de contextos aninhados é o modelo conceitual para os objetos das camadas de apresentação, aplicação e objetos trocados pela interface MHIO. Todos estes objetos (nós), são nada mais nada menos que uma versão, dentro dos conceitos do modelo. Os objetos trocados com o sistema de armazenamento obedecem ao padrão MHEG, como descrito na próxima seção. São exatamente estes objetos que constituirão a base pública.

À camada de apresentação cabe a criação de uma versão (objeto de dados) do nó MHEG entendido pelo sistema de armazenamento, quando o nó trafega no sentido da camada de armazenamento para a camada de aplicação. Cabe a ela também realizar as operações de migração dos nós de uma partição da hiperbase para a base pública, realizando as conversões necessárias, quando o nó trafega no sentido da camada de aplicação para a camada de armazenamento.

Conforme mencionado anteriormente, os objetos do nível de aplicação possuem métodos associados aos mecanismos de versão, sincronização, notificação, navegação, etc. Como o objetivo do artigo é abordar uma estrutura de referência aberta para sistemas hipermídia, não é relevante a apresentação destes conceitos, que são tratados com detalhes nas referências [Casa91 e CCLS92].

4 - A Proposta de Padrão MHEG

Os primeiros passos em direção a interoperabilidade e intercâmbio de facilidades multimídia foi dado com a padronização de imagens estáticas, informações de áudio e audiovisuais. Devido à natureza interativa com que as aplicações manipulam estes elementos básicos, surgiu a necessidade de uma interface de mais alto nível entre as aplicações e estes elementos. Entre as necessidades pode-se citar:

- A associação entre o conteúdo de dados destes elementos e seus atributos de apresentação.
- A sincronização no espaço (por exemplo, a superfície da tela) e no tempo entre os componentes que vão ser apresentados em conjunto.
- A definição de relações entre componentes.

Esta seção trata dos trabalhos desenvolvidos dentro do ISO/IEC JTC1/SC29/WG12, chamado *Multimedia and Hypermedia information coding Expert Group*, abreviadamente, MHEG; tendo em vista as necessidades mencionadas acima.

O futuro padrão internacional, denominado *Coded Representation of Multimedia and Hypermedia Information Objects*, mais conhecido como padrão MHEG, define a representação e codificação de objetos multimídia e hipermídia na sua forma final, de forma a poderem ser intercambiados dentro de uma aplicação ou entre aplicações ou serviços, por qualquer meio de intercâmbio, incluindo dispositivos de armazenamento, redes de telecomunicações e redes de difusão. Os objetos multimídia são definidos pelo padrão MHEG para o uso pelas Recomendações da CCITT, Padrões ISO e outros padrões, além é claro das arquiteturas e aplicações definidas pelo usuário. Uma vez que o MHEG é um grupo conjunto CCITT (SG VIII/Q9) e ISO JTC1, seu escopo reflete o modo de pensar orientado a comunicação, tendo em mente comunicação em tempo real.

O padrão segue a metodologia orientada a objeto (embora uma implementação do padrão não precise seguir necessariamente esta metodologia), definindo classes de objetos, cujo projeto se apoia na análise de similaridades entre as propriedades das várias categorias de objetos. O padrão provê a descrição de cada classe, uma definição precisa da representação de cada objeto multimídia hipermídia, que são as instâncias das classes, e uma codificação base para estes objetos. A codificação base da representação utiliza o padrão ISO 8824 *Specification of Abstract Syntax Notation One* (ASN.1) [ISO87a] e ISO 8825 *Specification of Basic Encoding Rules for ASN.1* [ISO87b]. Notações e regras de codificação alternativas são oferecidas: definição da estrutura de um objeto na forma de uma DTD (*Document Type Definition*) de acordo com o padrão ISO 8879 *Standard Generalized Markup Language* (SGML) [ISO 86]. As representações alternativas devem ser isomorfas e, portanto, equivalentes.

As principais classes de objetos MHEG são:

- *Content class*: são as *Media Content classes* que contém a informação nas diversas mídias (uma classe para cada tipo de dados: *text content, graphic content, still content, audio content, audiovisual content*), e a classe *NumericContent*. O conteúdo de dados desta classe pode ser codificado usando diferentes algoritmos; por exemplo, áudio tem vários padrões de codificação. Nestes casos, o atributo *hook* do objeto indica o algoritmo que deve ser usado.

- *Projector class*: estas classes definem todos os parâmetros de exibição que são relevantes para cada tipo de dados enumerados acima. Por exemplo, *audio projector class* contém atributos tais como referência de volume, estéreo/mono, balanço, ... As *projector classes* contêm também a indicação de como os atributos de apresentação mudam com o tempo, conceito chamado de *life progression*.

- *Projectable class*: constituídas pela agregação de um objeto *content* mais um objeto *projector*.

- *Link class*: definem o objeto *Link* que conecta um objeto fonte a um ou mais objetos destino. Os *links* fornecem a posição dos objetos *projectables* de destino nas coordenadas X, Y, Z e T (tempo) do espaço genérico MHEG, indicando a posição espacial e temporal relativa do objeto (sincronização espacial e temporal). O *link* MHEG é condicional: as ações definidas pelo *link* são processadas somente se condições de disparo forem satisfeitas.

- *Composite class*: define um objeto cujo conteúdo é um conjunto de objetos componentes com *links* entre eles. Os *links* descrevem a relação temporal e espacial entre os objetos componentes. Os objetos componentes podem ser objetos *projectable* ou *composite*, recursivamente.

- *Interaction class*: descreve as interações dos usuários de uma forma virtual, independente da interface homem-máquina usada pela aplicação, uma vez que o objetivo é a generalidade e independência de plataformas, implementações e configurações.

MHEG não faz nenhuma tentativa para definir a natureza dos métodos associados aos objetos, deixando esta função para os níveis superiores.

Ao se considerar o problema da apresentação de objetos com relações e requisitos espaciais e temporais, pode-se identificar vários níveis de sincronização.

O primeiro é quando a relação temporal e espacial é definida por um *script* fornecido pela aplicação do usuário, acima do modelo de contextos aninhados. Este *script* deve ser processado por processos da aplicação, que controlam a apresentação dos objetos. Neste caso o padrão MHEG não provê qualquer facilidade, uma vez que o intuito é definir um nível genérico de sincronização.

O segundo é quando a própria informação contida nos objetos é agregada de uma forma não mutável. É o que se chama sincronização a nível de sistema. Um exemplo deste caso é a sincronização da informação de áudio e vídeo em uma sequência audiovisual MPEG.

O terceiro é a sincronização fornecida pela agregação de objetos, que se quer encarar como uma unidade. Sincronização espacial e temporal é definida então entre os objetos agregados baseado no conceito de um espaço genérico. A representação deste caso é responsabilidade do padrão MHEG.

Finalmente, o quarto nível é a sincronização condicional fornecida pelo gerenciamento de eventos gerados por outros objetos, interação com o usuário ou com a aplicação do usuário. Esta sincronização é necessária para fornecer arranjos mais complexos entre objetos. A descrição de tal sincronização condicionada é baseada no conceito de ações condicionadas, que define que ações devem ser realizadas quando um determinado conjunto de condições forem satisfeitos. Os mecanismos gerais de sincronização condicionada podem ser usados no caso particular da descrição de *hyperlinks* (elos convencionais hipermídia) entre objetos. A representação deste caso também é responsabilidade do padrão MHEG.

Os mecanismos gerais de sincronização do padrão MHEG são definidos pelas classes *link* e *composite*, para a sincronização de objetos *projectable*, sendo que, nas condições associadas aos *links*, a interação com o usuário pode ser testada através dos objetos da classe *interaction*. Os objetos *projectable* são identificados pelos *links* pelo seu identificador e pelo caminho até este objeto através de nós *composite*.

Maiores detalhes sobre a proposta de padrão podem ser conseguidos nas referências [KrCo92, Mark91, MHEG92 e Cola92].

5 - Arquiteturas Distribuídas para a Estrutura Hipermídia

A implementação do MCA em uma arquitetura cliente servidor pode ter várias opções, dependendo principalmente se a própria implementação do servidor será distribuída ou não. As figuras 2 e 3 apresentam de uma forma genérica o modelo cliente e servidor MCA.

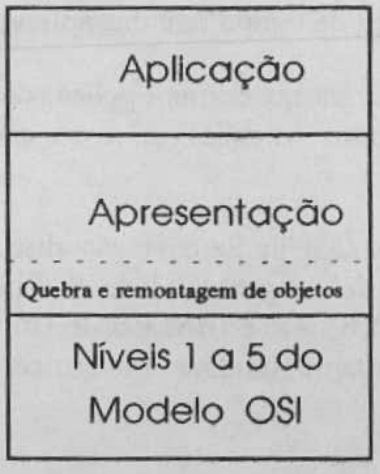

Figura 2 - Arquitetura do cliente

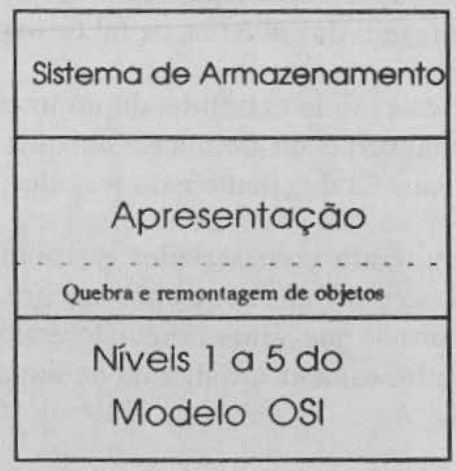

Figura 3 - Arquitetura do servidor

Nas estações clientes o nível de aplicação e apresentação é exatamente o do modelo MCA, com uma pequena diferença: o nível de apresentação deverá também ser capaz de desmontar e remontar os objetos multimídia especificados pela interface MHIO, da estrutura hipermídia apresentada na seção 2. Na arquitetura do cliente e servidor esta subcamada é apresentada pontilhada dentro do nível de apresentação.

A quebra e remontagem de objetos multimídia pode ser bastante complexa, dependendo da implementação do servidor e das características de tempo real exigidas pelas mídias, em termos de retardo, banda passante e continuidade de dados.

Várias são as arquiteturas distribuídas possíveis, das quais salientam-se duas. A primeira, arquitetura com um único servidor, é o caso mais simples. Nela, a troca de dados entre cliente e servidor pode ser o objeto multimídia inteiro, praticamente eliminando a necessidade da subcamada para quebra e remontagem de objetos. Neste caso simples, o nível de apresentação do servidor deve ser capaz de receber objetos MHEG e traduzí-los para o sistema de armazenamento, que pode ser um sistema de banco de dados qualquer, que atenda aos requisitos de tempo real e dimensão dos dados multimídia das aplicações.

Devida a pequena banda passante oferecida pelos dispositivos de armazenamento, pode ser que os dados de um objeto multimídia tenham de ser distribuídos por vários sistemas de armazenamento, de forma que a recuperação em paralelo possa contornar as dificuldades da recuperação em tempo real de tais objetos [GRAQ91]. Uma segunda arquitetura possível, para um sistema com tais características, é a arquitetura com servidor distribuído.

Neste caso específico, a subcamada de quebra e remontagem de objetos é essencial, tanto no cliente quanto no servidor, e tem suas funções comandadas pelo último. O servidor é dividido em um componente (executando em uma máquina, ou distribuído em várias máquinas), chamado SH (serviço hipermídia), responsável pelo armazenamento de todos os atributos dos objetos, exceto o atributo conteúdo de dados. O conteúdo de dados é espalhado pelos outros componentes do servidor, que podem ser servidores de arquivos especializados a cada uma das mídias. Ao componente SH do servidor distribuído cabe todo o gerenciamento de quebra e remontagem dos objetos, de tal forma a atender os requisitos de tempo real das aplicações.

Entre estes dois extremos de arquitetura, pode-se imaginar várias outras opções, dependendo principalmente da definição de qual é a estrutura de dados trocadas entre as camadas de apresentação do cliente e do servidor.

A arquitetura com servidor distribuído foi a escolhida para a implementação distribuída do MCA, devido aos requisitos de grande número de usuários e generalidade de aplicações e plataformas que vem orientando este projeto conjunto da PUC-Rio e IBM Brasil. Uma primeira versão baseada na arquitetura de um único servidor é, no entanto, uma meta intermediária.

6 - Conclusão

O desenvolvimento do modelo de contextos aninhados é baseado na idéia que as aplicações hipermídia são uma especialização de um conjunto maior de aplicações.

A estrutura em camadas apresentada possui os seguintes aspectos positivos:

- A MHIO provê uma interface flexível e padrão para suportar a construção de camadas superiores. A interface esconde os mecanismos de armazenamento, quer sejam realizados por servidores em rede ou não, que podem ser adaptados aos requisitos de eficiência e tamanho impostos por uma aplicação particular.

- A obediência ao padrão MHEG faz com que todos os objetos gerados possam ser intercambiados pelas mais diversas aplicações multimídia e hipermídia que obedecerão ao padrão, e mais ainda, permite a utilização de objetos gerados por estas aplicações.

- O MCA fornece um modelo de dados bastante flexível e uma plataforma aberta para a construção de aplicações extremamente genéricas, apresentando o controle de várias características desejáveis em um sistema hipermídia, tais como visão, versão, notificação, direito de acesso, navegação, sincronização, entre outras, facilitando a implementação de sistemas cooperativos e interativos.

- O encapsulamento fornecido pela camada de aplicação permite a construção de aplicações genéricas, independentes dos mecanismos de entrada e saída usados na interação do usuário e dos dados persistentes.

A estrutura e modelo proposto vem sendo desenvolvida pelo Departamento de Informática da PUC-Rio em conjunto com o Centro Científico da IBM Brasil, como um conjunto de classes C++ associada com os objetos mencionados. Tendo já sido implementado e testado um modelo

monousuário para o MCA em uma única estação de trabalho, encontra-se agora em desenvolvimento sua conformidade com a proposta de padrão MHEG, e a introdução dos mecanismos de controle de versão. Cabe mencionar que a implementação do primeiro protótipo monousuário já nos possibilitou, como representantes brasileiros no MHEG, a alteração na especificação dos objetos MHEG *link* e *composite nodes*, e não é por acaso sua adequação ao modelo de contextos aninhados, que é anterior à proposta de padrão.

Uma arquitetura distribuída para a estrutura hipermídia em camadas foi apresentada ao final do artigo, embora sem entrar em detalhes sobre a implementação do sistema de armazenamento, principalmente no que tange às suas características de tempo real, que é assunto de outro trabalho em desenvolvimento conjunto pelos mesmos grupos citados anteriormente.

Agradecimentos: Os autores gostariam de agradecer a Sérgio Colcher e Noemi Rodriguez, que muito contribuíram na discussão das idéias aqui apresentadas. O trabalho árduo de implementação realizado por estas pessoas e por outros membros do grupo, aos quais também gostaríamos de agradecer, tem tornado possível o refinamento constante do modelo apresentado.

Referências Bibliográficas

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

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

[CCLS92] Casanova, M.A.; Cavalcanti, M.R.; Lima, M.J.D.; Soares, L.F.G. "Versions in the Nested Context Hypermedia Model. *Relatório Técnico Departamento de Informática, PUC-Rio*. Agosto de 1992.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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