UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Ferramenta visual para especificação de hiperdocumentos, segundo o método OOHDM

por

CARLOS EMILIO PADILLA SEVERO

Dissertação submetida à avaliação, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. José Valdeni de Lima
Orientador

Porto Alegre, dezembro de 2001.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Severo, Carlos Emilio Padilla

Ferramenta Visual para especificação de hiperdocumentos, segundo o método OOHDM / por Carlos Emilio Padilla Severo. – Porto Alegre: PPGC da UFRGS, 2001.

99p.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2001. Orientador: Lima, José Valdeni de.

1. Métodos de especificação de hiperdocumentos. 2. Ferramentas gráficas. 3. Modelos de apresentação. I. Lima, José Valdeni de. II. Título.

Agradecimentos

Gostaria de realizar alguns agradecimentos às pessoas, que de alguma forma, direta ou indiretamente, foram importantes durante o desenvolvimento de meu trabalho.

Ao prof. Dr. José Valdeni de Lima pela orientação, apoio e incentivo durante o desenvolvimento de minha dissertação.

Ao prof. Dr. Gustavo Hector Rossi, pela grande demonstração de amizade, apoio e profissionalismo durante a etapa de desenvolvimento do trabalho em La Plata, Argentina.

Aos colegas e amigos que conheci durante o período que passei na Argentina.

A CAPES pela bolsa concedida.

Aos professores Evaldo e Marcos da Universidade da Região da Campanha, pelo apoio para que a etapa de conclusão da dissertação pudesse ser realizada no exterior.

Ao meu pai, minha mãe e irmãos, que sempre me incentivaram e apoiaram durante minha vida estudantil, profissional e pessoal.

Um agradecimento especial a minha esposa Elenara Robaina dos Santos Severo, por toda a compreensão e carinho.

A Deus, por ter me oportunizada toda esta vivência que tem contribuído muito para minha realização e crescimento profissional e pessoal.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

O desenvolvimento de artefatos de software é um processo de engenharia, como todo processo de engenharia, envolve uma série de etapas que devem ser conduzidas através de uma metodologia apropriada. Para que um determinado software alcance seus objetivos, as características conceituais e arquiteturais devem ser bem definidas antes da implementação. Aplicações baseadas em hiperdocumentos possuem uma característica específica que é a definição de seus aspectos navegacionais. A navegação é uma etapa crítica no processo de definição de softwares baseados em hiperdocumentos, pois ela conduz o usuário durante uma sessão de visita ao conteúdo de um *site*. Uma falha no processo de especificação da navegação causa uma perda de contexto, desorientando o usuário no espaço da aplicação. Existem diversas metodologias para o tratamento das características de navegação de aplicações baseadas em hiperdocumentos. As principais metodologias encontradas na literatura foram estudadas e analisadas neste trabalho. Foi realizada uma análise comparativa entre as metodologias, traçando suas abordagens e etapas. O estudo das abordagens de especificação de hiperdocumentos foi uma etapa preliminar servindo como base de estudo para o objetivo deste trabalho. O foco é a construção de uma ferramenta gráfica de especificação conceitual de hiperdocumentos, segundo uma metodologia de modelagem de software baseado em hiperdocumentos. O método adotado foi o OOHDM (*Object-Oriented Hypermedia Design Model*), por cercar todas as etapas de um processo de desenvolvimento de aplicações, com uma atenção particular à navegação. A ferramenta implementa uma interface gráfica onde o usuário poderá modelar a aplicação através da criação de modelos. O processo de especificação compreende três modelos: modelagem conceitual, modelagem navegacional e de interface. As características da aplicação são definidas em um processo incremental, que começa na definição conceitual e finaliza nas características de interface. A ferramenta gera um protótipo da aplicação em XML. Para a apresentação das páginas em um navegador *Web*, utilizou-se XSLT para a conversão das informações no formato XML para HTML. Os modelos criados através das etapas de especificação abstrata da aplicação são exportados em OOHDM-ML. Um estudo de caso foi implementado para validação da ferramenta. Como principal contribuição deste trabalho, pode-se citar a construção de um ambiente gráfico de especificação abstrata de hiperdocumentos e um ambiente de implementação de protótipos e exportação de modelos. Com isso, pretende-se orientar, conduzir e disciplinar o trabalho do usuário durante o processo de especificação de aplicações.

Palavras-Chave: métodos de especificação de hiperdocumentos, ferramentas gráficas, modelos de apresentação, XML, XSLT, HTML e OOHDM-ML.

TITLE: “VISUAL TOOL FOR HYPERDOCUMENTS SPECIFICATION”

Abstract

The development of software artifacts is an engineering process, as any other engineering process, it involves a series of steps that needs to be conducted through an appropriate methodology. For a determined software to achieve its objectives, conceptual characteristics and architectures should be well defined before the implementation. Applications based on hyperdocuments have a specified characteristic that is the definition of its online aspects. The online process is a critical phase in the process of softwares specifications based on hyperdocuments, because it conducts the user during a visit session of a site. An error in the specification of the online process causes a loss of the context, disorientating the user in the application interface. There are diversified methodologies for the treatment of online characteristics of applications based on hyperdocuments. The most important methodologies found in literature were studied and analyzed in this research. It was made a comparative analyze between methodologies, outlining their approaches and steps. The study of the specification approaches of hyperdocuments was a preliminary phase, serving as a base for the study of the objectives of this research. The focus is the construction of a graphic tool of conceptual hyperdocuments, according to a methodology of software modeling based on hyperdocuments. The method adopted was the OOHDM (Object-Oriented Hypermedia Design Model), because it contains all the steps of an application development process, with special attention on the online process. The tool implements a graphic interface where the user will can mold the application through the developing of models. The tool generates a prototype of the application in XML format. For the presentation of the pages in a Web browser, was used XSLT to convert informations from the XML to HTML format. The models created through the steps of an abstract specification of applications are exported in OODHDM-ML format. A study of the case was implemented for the validation of the tool. As the most important contribution of this research, we can mention the construction of a graphic interface of an abstract specification of hyperdocuments and an interface of implementation of prototypes and exportation of models. With that, the intend is to orientate, conduct and regulate the work of the user during application specification process.

Keywords: hyperdocuments specification methods, graphic tools, presentation models, XML, XSLT, HTML e OOHDM-ML.

1 Introdução

O grande volume de informações publicadas na *Web* é notável. A grande variedade de propósitos e classes de usuários que as aplicações baseadas em hiperdocumentos atingem é algo surpreendente. A evolução das tecnologias e a disseminação rápida de informações apoiada pela *Web* permitem ao desenvolvedor a elaboração de aplicações de uma forma rápida, onde o mesmo pode expressar todo seu potencial criativo na elaboração das páginas da aplicação. Desta forma, as aplicações baseadas em hiperdocumentos se tornam uma estratégia poderosa para a interação com clientes, tornando-se assim, uma poderosa ferramenta de apoio ao comércio eletrônico.

Entretanto, o projeto de hiperdocumentos, assim como em qualquer processo de desenvolvimento de software, deve ser gerenciado para que não produza um produto distinto das características propostas durante a sua especificação conceitual. O desenvolvimento de grandes aplicações tende a ser complexo. Principalmente, quando utilizam tecnologias distintas, envolvendo uma grande equipe de desenvolvimento. O domínio da aplicação deve ser estruturado e organizado, para que as tarefas de desenvolvimento possam ser coordenadas, distribuídas e sincronizadas pela equipe de trabalho. A aplicação final deve se tornar clara e acessível aos diferentes perfis de usuários que irão utilizá-la.

Um projeto deve seguir um modelo que possibilite o desenvolvimento de uma aplicação clara e de fácil operação por parte do usuário [GAR 91][GAR 93][GAR 95]. Além disso, deve ser fácil de ser mantida do ponto de vista dos autores. Tal como em aplicações convencionais, torna-se clara a necessidade de uma abordagem sistemática e formal para a definição da estrutura organizacional de uma aplicação baseada em hiperdocumentos. O uso de um modelo formal disciplina o autor a projetar a aplicação antes de desenvolver o conteúdo das páginas [SCH 95].

Um bom método de projeto deve ser eficiente de se usar e deve produzir uma aplicação hipermídia coerente que ajude o usuário a obter a informação desejada [IZA 95]. Para isso, o método de projeto deve tratar o domínio da aplicação e as conexões entre elas, como representá-las, manipulá-las e armazená-las. Além disso, um ponto importante a ser considerado é a dinâmica de interação com o usuário. Desta forma, projetos de hiperdocumentos devem incorporar características específicas, diferentes dos métodos tradicionais da Engenharia de Software.

Há poucos anos atrás, algumas abordagens de projeto de aplicações hipermídia foram discutidas nos trabalhos de [GAR 93], [LAN 94] e [IZA 95], onde foram verificados vários problemas associados ao processo de projeto de hipermídia, tal como a combinação de navegação pela complexa estrutura da informação formada pelos dinâmicos dados multimídia. A solução constatada para esses problemas está na adoção de um processo sistemático e modular para o desenvolvimento de aplicações, como citado em [HAR 93]. Cada atividade de uma metodologia de projeto deve tratar os diferentes aspectos de projeto em estágios apropriados ao nível de abstração [NAN 95], onde as decisões de projeto precisam ser traçadas de forma incremental.

Dentro do contexto de especificação das características de aplicações baseadas em hiperdocumentos, surge o interesse pelo desenvolvimento de ferramentas de apoio. Essas ferramentas utilizam os conceitos encontrados em um determinado método de projeto, para a implementação das primitivas em um ambiente de especificação. Os

ambientes de especificação podem utilizar uma descrição textual onde, através de regras, determinam as características de domínio, navegação e interface da aplicação. Outra forma de implementação de um ambiente de especificação é através de ferramentas visuais em um ambiente gráfico. O desenvolvedor, através das ferramentas disponibilizadas de acordo com o método de especificação implementado, poderá elaborar o projeto de aplicação.

A elaboração de ferramentas de especificação visual tem como foco principal o controle, documentação e apoio ao projetista, durante o processo de especificação de aplicações. A implementação de primitivas de um método formal de especificação em um ambiente gráfico permite que um projeto seja prototipado rapidamente. Modelos de apresentação de hiperdocumentos, baseados em características conceituais, podem ser gerados automaticamente. Ainda, um ambiente de especificação gráfica permite que projetos sejam consistidos durante sua especificação abstrata. A passagem de uma etapa a outra pode ser controlada, permitindo que visões navegacionais possam ser fidedignas as suas delimitações conceituais. Assim como esquemas de contextos navegacionais possam delimitar o conjunto de objetos que serão instanciados em um determinado momento, de acordo com o modelo de navegação definido. Enfim, envolve uma série de características que levam a implementação final da aplicação de forma incremental, de acordo com o nível de abstração adequado.

A ferramenta apresentada neste trabalho implementa um ambiente gráfico para especificação de hiperdocumentos, baseado nas primitivas do método OOHDM [ROS 96]. Através do ambiente gráfico da ferramenta, o projetista poderá esboçar projetos, desde um nível conceitual até a implementação de modelos de apresentação.

Este trabalho tem como objetivo a implementação de uma ferramenta de apoio à especificação de hiperdocumentos em OOHDM, conforme relatado anteriormente. Para isso, o ambiente deverá dar suporte a especificação dos requisitos da aplicação baseada no modelo conceitual, modelo de navegação e modelo de interface abstrata descritos na especificação OOHDM [VIL 99]. Após a etapa de especificação abstrata do hiperdocumento, a ferramenta irá gerar automaticamente as bases de dados necessárias para a obtenção das informações que serão apresentadas na aplicação. Para a apresentação dos hiperdocumentos, são gerados arquivos no formato XML [W3C 98b], implementando o código da aplicação de acordo com os modelos produzidos nas etapas de projeto.

Inicialmente, faz-se necessário um relato sobre os principais métodos de especificação de aplicações baseadas em hiperdocumentos, encontrados atualmente na literatura. No capítulo 2, esses métodos serão apresentados, onde seus principais conceitos, etapas e primitivas são relatados.

Uma breve comparação entre os métodos de especificação, como resultado do estudo de suas características, é apresentada no capítulo seguinte. Comparando-se assim, os principais aspectos tratados entre os métodos.

No capítulo 4, a ferramenta visual de especificação de hiperdocumentos será apresentada. Sua arquitetura, características, módulos de interface e funcionalidades serão demonstradas.

No capítulo 5, um estudo de caso é desenvolvido como forma de validar e avaliar o processo de especificação de hiperdocumentos com o software. O trabalho é finalizado com uma breve conclusão, onde as principais contribuições do trabalho serão relatadas, assim como sugestões para trabalhos futuros serão apresentadas.

2 Abordagens de projeto de hiperdocumentos

Nesta seção serão apresentadas algumas abordagens para especificação e projeto de hiperdocumentos encontrados na literatura, onde serão estudadas suas características fundamentais, associadas ao tratamento dos problemas de especificação citados anteriormente.

Segundo [GAR 93] e [GAR 94] o grau de sucesso de uma aplicação hipermídia está diretamente relacionado com a habilidade do projetista em capturar e organizar a estrutura complexa de uma aplicação para que possa ser manipulada e entendida por um grande número de usuários. Uma determinada aplicação não precisa englobar toda a informação abrangente de um determinado assunto, mas sim as partes mais importantes da informação, de maneira natural e que tenham um forte significado.

O projetista, durante o projeto de uma aplicação hipermídia, enfrenta pelo menos duas categorias correlacionadas de problemas. A primeira são as tarefas globais de projeto, onde são identificadas todas as classes de informação, elementos e estruturas navegacionais da aplicação. Uma outra categoria são as tarefas locais, onde são colocados conteúdos nos nós de informação.

A autoria de uma aplicação hipermídia pode ser classificada em duas dimensões que estão associadas às categorias mencionadas no parágrafo anterior, são elas: autoria em ponto grande e autoria em ponto pequeno. A autoria em ponto grande está relacionada ao projeto global da aplicação envolvendo aspectos estruturais, preocupa-se mais com a modelagem do domínio da aplicação. Já autoria em ponto pequeno trata do desenvolvimento do conteúdo dos nós de informação, mais dependente de ferramentas utilizadas para implementação. A autoria em ponto grande é a etapa onde se concentram os processos de especificação de requisitos e projeto do hiperdocumento. Esta é a principal preocupação dos métodos de projeto encontrados na literatura.

Existe muita pesquisa sendo realizada em torno de modelos e métodos para a especificação de hiperdocumentos. Estas pesquisas estão produzindo várias publicações e protótipos de ferramentas para avaliação dos métodos. Com isso, busca-se através deste trabalho apresentar as principais características de alguns métodos e modelos para especificação de hiperdocumentos. Existem vários modelos de especificação de hiperdocumentos, sendo inviável apresentar todos neste trabalho. Entretanto, o número de modelos escolhidos representa o que está sendo desenvolvido nesta linha de pesquisa, abrangendo distintas abordagens de projeto.

Os métodos escolhidos foram HDM, OOHDM, RMM, HMBS/M, que abrangem muitas abordagens atuais para a especificação dos requisitos e projeto de hiperdocumentos, tais como entidade-relacionamento, orientação a objetos, diagramas de estados e redes de petri.

2.1 HDM (*Hypermedia Design Model*)

HDM é um modelo para especificação de aplicações baseadas em hipertexto que mantém as peculiaridades globais de outros modelos de especificação de software, através de certas características de modelagem, mas ao mesmo tempo inclui

características próprias relacionadas ao projeto de hiperdocumentos, que as abordagens tradicionais não possuem. Tratando-se de um ponto de vista terminológico, HDM foi desenvolvido através de uma combinação de conceitos implementados em modelos pré-existentes, como na área de Banco de Dados e Hipermídia, com novos termos que são considerados relevantes na especificação de hiperdocumentos.

2.1.1 Modelagem de domínio

Segundo o método, uma aplicação é composta por partes de informação que são representadas por entidades. Cada entidade atua como um objeto conceitual que modela o domínio da aplicação. O conjunto de entidades é modelado em um diagrama, demonstrando os tipos de objetos que podem ser criados na aplicação. As entidades são porções de informações autônomas, ou seja, não são condicionadas pela existência de outros objetos de domínio [GAR 93].

O conceito de componente denota uma porção de informação que é parte de uma entidade. De acordo com o modelo, uma entidade pode ser subdividida em porções menores para o agrupamento de informações de contexto. Em suma, uma entidade é formada por uma hierarquia de componentes.

Uma unidade é uma porção menor de informação que combina informações de um componente. A unidade, no contexto de hipertexto, é considerada um nó de informação, que é criada sob uma determinada perspectiva.

As estruturas de informação são interconectadas por ligações (representações conceituais de um elo). As ligações são classificadas em estruturais, de perspectiva e de aplicação. As ligações estruturais interconectam somente componentes de uma mesma entidade, enquanto que as ligações de perspectiva conectam as unidades que possuem um mesmo componente em comum, sob uma determinada perspectiva. As ligações de aplicação especificam os relacionamentos de dependência entre componentes e entidades do domínio da aplicação de acordo com a visão do projetista.

As ligações de perspectivas e a maioria das aplicações estruturais não precisam ser modeladas, pois podem ser derivadas do domínio, desde que possam ser derivadas automaticamente da estrutura das entidades.

Em HDM existe uma distinção entre o modelo de aplicação e a navegação da aplicação. O modelo da aplicação é elaborado através de um esquema que define as entidades, componentes, unidades e ligações definindo o modelo conceitual. Já a navegação é uma instanciação do modelo que é realizada através da definição de estruturas de acesso que são pontos de entrada da aplicação, orientados ao leitor, que dão acesso as informações.

As semânticas de navegação definem como as estruturas de informação serão visualizadas e navegadas pelo leitor durante a execução da aplicação. HDM provê uma semântica de navegação padronizada e simplificada.

Uma entidade identifica um objeto do mundo real, que faz parte do domínio da aplicação, através da definição de características conceituais. As entidades são agrupadas em tipos de entidades, representando classes de objetos do mundo real. A noção de entidade é considerada satisfatória para a modelagem de informações, como

na área de banco de dados através do modelo entidade-relacionamento (E-R), por exemplo. Entretanto é importante salientar que o conceito de entidade em HDM é diferente do modelo E-R. Uma característica que diferencia o modelo de entidades HDM são as estruturas de ligação e semânticas de navegação associadas. Estas características formam padrões muito mais complexos que as relações permitidas pelo modelo E-R tradicional [GAR 93].

Já foi mencionado, anteriormente, que uma entidade é composta por um conjunto de componentes de forma hierárquica. Um componente é uma abstração para um conjunto de unidades que são os recipientes de informação. A hierarquia é organizada de forma linear por um componente ancestral que deriva subcomponentes, mantendo as mesmas características do ancestral e acrescentando suas próprias características. Os componentes não são autônomos, pois dependem um dos outros para existirem.

Muitos autores identificam que as hierarquias são muito úteis para auxiliar o usuário durante a navegação. As hierarquias podem ser introduzidas por muitos critérios semânticos como as relações de domínio, como, por exemplo, as relações “parte de”.

Em HDM uma hierarquia é um dispositivo sintático que forma a base de organização e edificação de blocos de informação dentro de uma aplicação.

A perspectiva é uma característica comum em hiperdocumentos. Ela tem como objetivo possibilitar a modelagem de diversos pontos de vista, que podem ser derivados de um mesmo domínio. Razões não faltam para que esta característica seja forte em aplicações baseadas em hiperdocumentos. Um exemplo disto é uma aplicação multinacional, onde um mesmo tópico pode ser observado em diferentes línguas, ou uma aplicação educacional onde o estilo de apresentação de um tópico pode seguir vários estilos retóricos, como: discursivo, sintético, esquemático, etc. Estes exemplos levam em consideração o perfil dos leitores.

Atualmente, uma mesma informação tem sido apresentada em mídias diferentes (texto, imagem, áudio) formando diferentes perspectivas. Em HDM a possibilidade de capturar diferentes formas de apresentar uma informação é denominada perspectiva. Segundo a organização hierárquica, se uma entidade tem duas perspectivas diferentes, seus componentes herdarão estas perspectivas. Assim como todas as instâncias herdarão as perspectivas definidas para suas entidades conceituais.

As perspectivas são uma forma de organizar a sintaticamente a informação, porém HDM não possui nenhuma semântica pré-definida para interpretar este conceito. Portanto a consistência só é realizada durante a implementação de técnicas distintas para a apresentação de informação sob perspectivas diferentes de uma entidade.

Uma unidade corresponde à associação de um componente a uma determinada perspectiva. Cada unidade é identificada por um nome e um corpo. O corpo é o lugar onde se encontra o conteúdo atual que será apresentado, podendo ser um texto, imagem, vídeo ou som. As unidades são equivalentes aos nós, sendo que somente podem ser criadas sob uma perspectiva do modelo conceitual. Não existe a possibilidade de criação de nós arbitrários em HDM.

Em HDM é permitido que unidades diferentes compartilhem de um mesmo corpo, ou seja, um mesmo conteúdo de informação pode ser compartilhado em diferentes contextos de forma não mutuamente exclusiva. Este é um problema verificado, pois possibilita a desorientação do leitor, mas por outro lado simplifica o projeto da aplicação [GAR 93].

Em hiperdocumentos as ligações exercem um papel conceitual, que são representados no modelo de domínio, e navegacional, que são as ligações que ditam os padrões navegacionais da aplicação. Às vezes estes dois aspectos podem ser consistentes entre si, ou então geram conflitos. Pode ocorrer que determinadas ligações de domínio não são satisfatórias para a derivação de navegação. Por exemplo, podem induzir padrões de navegação que não são relevantes à aplicação. Portanto, na definição das ligações deve ser consideradas as metas de representação e navegação da informação. As ligações são classificas em três grupos: de perspectiva, estruturais e de aplicação, conforme citado anteriormente.

As ligações de perspectiva conectam unidades que correspondem a um mesmo componente. Estas são ligações muito simples e claras para o leitor, pois a navegação entre as unidades não altera o contexto atual onde o leitor se encontra.

Ligações estruturais unem componentes de informações de uma mesma entidade. Existem tipos de ligações estruturais, onde cada tipo corresponde a uma relação especificada pela árvore ordenada de entidades. Exemplos de ligações estruturais são:

- *up*: é um tipo de ligação estrutural que une um componente com seu ancestral direto;
- *down*: outro tipo de ligação estrutural que permite a ligação de um componente com seu descendente direto;
- *down-1*: ligação estrutural que relaciona um determinado componente com o seu primeiro descendente indireto;
- *next-sibling*: relaciona um componente com o próximo componente derivado de seu mesmo ancestral direto;
- ligações do tipo *root*: que permite a ligação entre um determinado componente com o componente que é seu ancestral raiz (aquele que forma o topo da árvore).

As ligações estruturais são utilizadas para a navegação por porções de informação de uma mesma entidade. Em termos navegacionais, as ligações estruturais são um pouco mais complexas que as ligações de perspectiva. Todavia, são mais simples de entender por parte do leitor, pois mantém o leitor dentro de um mesmo contexto de informação, ou seja, a mesma entidade. Possibilitando mais segurança ao usuário durante a navegação evitando a desorientação.

As ligações de aplicação representam relações de dependência entre as entidades, ou os componentes delas, representados no domínio da aplicação. Elas são classificadas em tipos e são identificadas através de um nome, um conjunto de tipos de entidades fonte e destino e um atributo de simetria que pode assumir o valor simétrico ou assimétrico. Os tipos de entidade fonte e destino definem o que pode ser ligado a que. Por exemplo, desde os tipos de ligações que tenham sido definidas como tendo

como entidades fonte as entidades “A”, e como entidades destino as entidades “B”, serão permitidas instâncias de conexão somente para este tipo de ligação definida.

Um exemplo de ligação de aplicação é a ligação “ser-autor-de”, onde instâncias ou componentes do tipo “Compositor” são conectados as instâncias ou componentes do tipo “Ópera Musical”.

Em HDM, a escolha e implementação de tipos de ligação da aplicação é de responsabilidade do autor. Somente o autor irá definir as ligações necessárias e quais as entidades fonte e entidades destino envolvidas. Uma única regra que deve ser observada é a consistência entre os tipos de ligação.

2.1.2 Semântica de navegação

O uso atual de aplicações baseadas em hipermídia é amplamente definido através do conceito de navegação, a qual determina que objetos de informação o usuário irá manipular durante a execução da aplicação.

A semântica de navegação irá definir que objetos de informação serão visualizados durante a interação do usuário (entidades, ligações, componentes ou unidades). Quais as ligação serão percebidas e qual o comportamento necessário quando da ativação das ligações.

Dada a especificação de uma semântica de navegação compatível com um determinado sistema de hipermídia, torna-se possível a tradução da especificação da aplicação em HDM para uma aplicação através do mapeamento das primitivas HDM em estruturas de implementação de um determinado ambiente de hipertexto. Uma mesma especificação HDM pode derivar diferentes visões navegacionais, formando assim diferentes visões para um mesmo modelo. Cada modelo é caracterizado por características visuais e comportamentais próprias, ou sendo executados em diferentes sistemas de hipertexto.

De acordo com as semânticas de navegação padrão de HDM, somente as unidades podem ser visualizadas pelo leitor do hiperdocumento, na forma de nós. Um único nó de informação é ativado por vez. Os elos serão visualizados nas unidades que estabelecem as conexões entre as unidades.

Como em HDM somente os elos de perspectiva conectam unidades, estes precisam ser traduzidos em ligações concretas que são denominadas ligações *unit-to-unit*. Enquanto que elos estruturais e de aplicação unem componentes de entidades e são considerados elos abstratos.

As regras para tradução de elos abstratos em elos concretos são baseadas na idéia de padrão representativo para cada objeto abstrato como componentes ou entidades. A definição de padrões representativos para componentes e entidades é realizada em função das perspectivas padrão para cada tipo de entidade. Assume-se que o padrão representativo para um componente é a sua unidade, de acordo com sua perspectiva padrão. Já o padrão representativo para entidades é o padrão representativo de seu componente raiz. Isto quer dizer que o componente padrão de uma entidade é sua perspectiva *default*.

2.2 OOHDM (Object-Oriented Hypermedia Design Model)

O método OOHDM é uma abordagem de desenvolvimento de aplicações hipermídia baseada em modelos. O método compreende quatro etapas, onde o modelo da aplicação é construído e enriquecido a cada passo. Cada etapa é baseada na anterior, cujo estilo de desenvolvimento é interativo e incremental [SCH 94b].

A modelagem conceitual é a primeira etapa do processo. Nesta etapa, o projetista define as semânticas do modelo conceitual de domínio da aplicação através de mecanismos como classificação, composição, generalização e especialização. O produto final desta etapa é um modelo do domínio da aplicação envolvendo classes, sub-sistemas, relacionamentos e perspectivas de atributos.

O projeto navegacional é a próxima etapa. Nesta, o projetista estuda o perfil do usuário e as tarefas que o mesmo poderá executar na aplicação. A ênfase está nos aspectos cognitivos da aplicação hipermídia em relação as preferências do usuário. O mapeamento do modelo conceitual em objetos navegacionais gera o produto final desta etapa que é o conjunto de nós, elos, estruturas de acesso, contextos navegacionais e transformações navegacionais da aplicação.

Após o modelo conceitual e navegacional da aplicação, o projetista deverá modelar a interface. Esta etapa visa a obtenção dos objetos de interface perceptíveis durante a navegação para a descrição da interface navegacional. Com os mecanismos de mapeamento entre o modelo de navegação e os objetos perceptíveis será elaborada a interface, procedimentos de resposta a eventos externos e as transformações de interface.

A implementação da aplicação é obtida pelo mapeamento dos modelos de navegação e interface abstrata em objetos concretos avaliados em ambientes de implementação. Exemplos de ferramentas para implementação de aplicações hipermídia são: *Hypercard*, *Toolbook*, *MacWeb*, KMS, *Guide*, *Microcosm*, entre outras.

2.2.1 Modelagem conceitual

Um modelo do domínio da aplicação é construído através do modelo orientado a objetos proposto por Rumbaugh [RUM 91], acrescentado de primitivas como perspectiva de atributos e sub-sistemas. A principal característica desta etapa é a captura das semânticas de domínio da aplicação, levando-se em consideração o perfil do usuário.

Classes conceituais podem ser construídas por hierarquias de agregação, generalização ou especialização. A abordagem de projeto orientado a objetos foi adotada em OOHDM porque as semânticas de agregação, generalização e especialização estão bem definidas e estabelecidas nos modelos de dados em [BAN 87] e, também, por ser uma poderosa abordagem para a descrição das visões de objetos que são utilizados na definição do modelo navegacional. E ainda, por se tratar de uma escolha padrão para a construção de aplicações interativas complexas [JAC 92][JAC 95].

Uma extensão do modelo clássico baseado em objetos é a perspectiva de atributos. Um determinado atributo de uma classe conceitual pode possuir um possível

tipo de dado, definido a partir de uma perspectiva pré-definida. A perspectiva de atributos é obtida através de uma lista de possíveis valores que um atributo pode possuir, como por exemplo: texto, imagem, animação ou som. Este aspecto é representado no modelo conceitual.

A Figura 3.1 apresenta parte do esquema conceitual do projeto de uma aplicação exemplo, o modelo foi simplificado para ilustrar melhor a explicação. Note no diagrama que as informações contidas podem ser utilizadas para diferentes perfis de usuários e para diferentes propósitos, pois representam informações de domínio que podem ser combinadas a fim de produzir uma informação mais concisa e contextualizada.

A classe “Evento” possui o atributo *Description*, que por sua vez possui uma perspectiva de dois tipos de dados possíveis: *Text e Picture*, representando uma informação textual ou gráfica. Esta é uma extensão do modelo OOHDM em relação ao OMT denominada perspectiva de atributos. A seta que parte da classe “Referência” em direção à caixa tracejada é uma relação com outras classes de outros sub-sistemas da aplicação. O mesmo é verificado com a classe “Comentários Históricos”.

As classes podem relacionar-se com subsistemas, que são abstrações do esquema conceitual de uma outra aplicação completa. Os detalhes de domínio do subsistema não interessam no momento, mas os relacionamentos do modelo conceitual da aplicação em questão com o subsistema é uma informação relevante e que precisa ser modelada. A Figura 3.2 demonstra um relacionamento em uma aplicação para turismo com um subsistema. Note que um museu pode representar um conjunto de classes relacionadas, formando um outro sistema, o qual está relacionado ao sistema de turismo.

Um subsistema pode ser auto-contido, possuindo um único ponto de entrada e saída. Desta forma, torna-se "servidor de informações". Esta é uma situação apropriada para lidar com aplicações hipermídia simples, mas com a desvantagem de ser muito restrita [ROS 96]. Os pontos de entrada de um sub-sistema são classes que podem ser acessadas por outras classes ou subsistemas. Na Figura 3.2, por exemplo, o museu pode ser composto por classes como “Salas”, “Pinturas”, “Exibições”, etc. Neste caso, “Exibições” poderá ser um ponto de entrada significativo, pois faz sentindo a exploração do museu por uma exposição [ROS 96].

A modelagem conceitual gera um produto semelhante as abordagens de projeto orientadas a objetos atuais, como o Modelo de Relacionamento de Objetos de Rumbaugh [RUM 91].

A expressão de relacionamentos pode ser de duas formas. Uma onde o relacionamento possui um atributo e, eventualmente um comportamento, sendo este relacionamento implementado no modelo. A Figura 3.3 demonstra como ocorre este tipo de relacionamento, onde o relacionamento “Possui” indica a data de compra da “Obra” por uma determinada “Pessoa”.

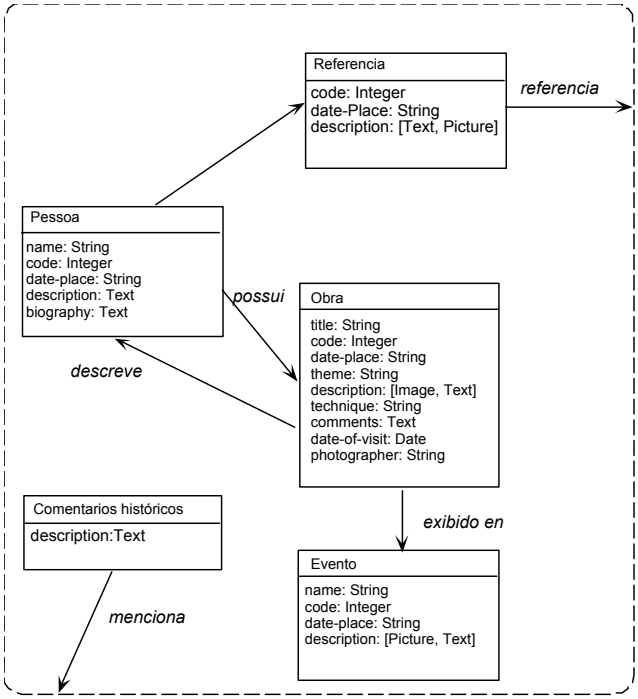


FIGURA 3.1- Parte do esquema conceitual do projeto Portinari [ROS 96]

Outra forma de relacionamento é o implícito, onde determinado relacionamento não é importante para o domínio da aplicação. Este exemplo é demonstrado através da Figura 3.4, onde não estamos interessados na pessoa e na obra de arte como conceitos separados. Neste caso o foco de interesse está na obra em si.

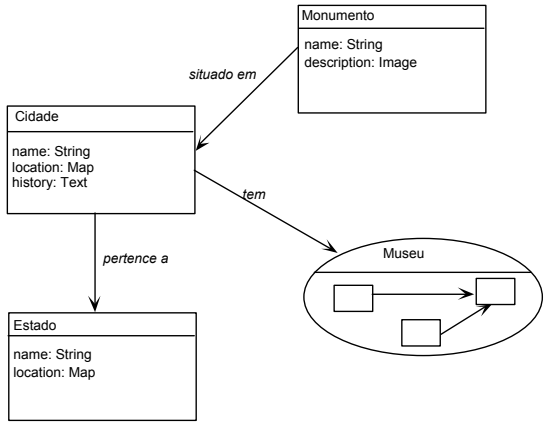


FIGURA 3.2- Esquema Conceitual da Aplicação para Turismo [ROS 96]

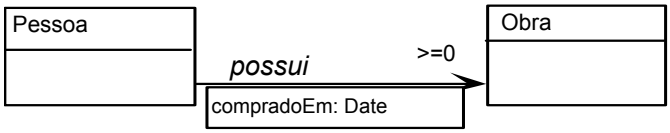


FIGURA 3.3- Relacionamento com um atributo [ROS 96]

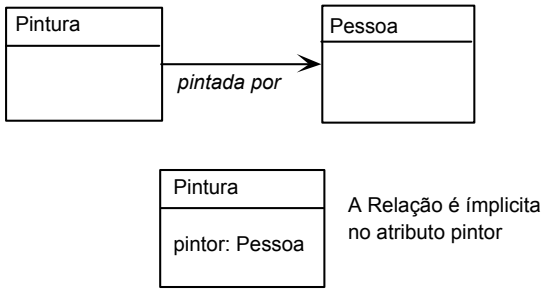


FIGURA 3.4- Relações x Atributos [ROS 96]

Quando um determinado atributo representa uma entidade conceitual complexa e que se faz importante explorar no modelo, deve-se modelar o relacionamento e não esconder o mesmo em uma entidade. Em OOHDM são utilizados cartões de classes e relacionamentos para a especificação e documentação completa. Nestes cartões são especificadas as cardinalidades dos relacionamentos.

As classes conceituais possuem atributos tipados que representam propriedades conceituais dos objetos. As aplicações são definidas como visões do esquema conceitual, de acordo com os perfis de diferentes usuários, algumas destas visões podem conter mais ou menos atributos de uma classe conceitual. A aparência visual de um atributo na aplicação final está relacionada ao seu tipo, que pode ser derivado de uma lista de possibilidades. Cada aparência visual possível de um atributo é denominada de perspectiva de atributo. As semânticas de perspectivas de atributos em OOHDM são as mesmas definidas em HDM [GAR 91].

As perspectivas de atributos de uma classe podem ser modeladas de uma forma mais genérica, através de uma hierarquia de classes dos tipos de um determinado atributo. Assim, pode-se modelar uma composição de tipos para um determinado atributo. Na Figura 3.5, pode-se verificar um exemplo deste conceito, onde o tipo de dado música digital pode ser decomposto em duas sub-classificações: "Música Computador" ou "Música CD". As sub-classes herdam atributos, estrutura de partes, comportamentos e relacionamentos.

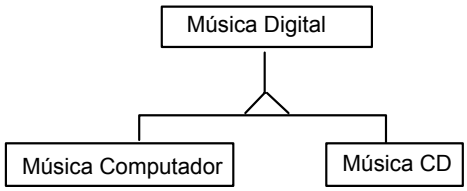


FIGURA 3.5- Uma hierarquia de classes para atributos [ROS 96]

Os relacionamentos que formam parte-de são descritos através de relacionamentos de agregação. Os agregados, em uma definição de classes, são semelhantes aos componentes em HDM [GAR 91]. A Figura 3.6 apresenta um modelo de agregação, onde uma cidade é composta por *n* ocorrências de imagens.

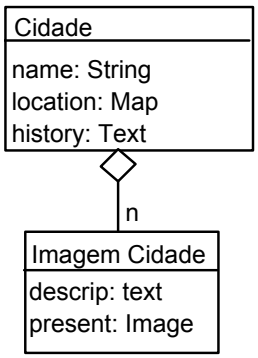
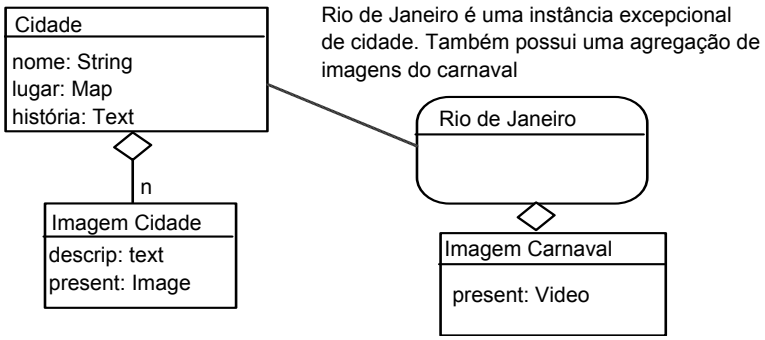


FIGURA 3.6- Estrutura de Agregação [ROS 96]

Em alguns modelos surge algumas instâncias excepcionais, que possuem algumas características que a diferenciam do modelo conceitual. Tais características podem ser a presença de alguns atributos específicos que não foram especificados no modelo conceitual, por exemplo. Isto torna a instância um objeto especializado de uma determinada classe. Em OOHDM foi implementado um esquema de instâncias (e Cartões de Objetos) para expressar esta situação. O esquema de instâncias mostra instâncias excepcionais, isto é, aquelas que não correspondem exatamente as definições do esquema de classes.

Neste esquema, pode-se acrescentar atributos a um objeto, mudar o tipo de algum dos atributos existentes, refinar relacionamentos, mostrar valores possíveis, etc. É importante observar que a construção de um esquema de instâncias é ainda uma preocupação de autoria em ponto grande.

A definição de um esquema de instâncias é diretamente relacionada com o processo de prototipagem, como foi descrito em [NAN 95]. A Figura 3.7 mostra o exemplo de instâncias excepcionais da classe “Cidade”, observe que “Rio de Janeiro” é uma instância excepcional de “Cidade”, pois apresenta uma série de imagens do carnaval, que em outras instâncias de “Cidade” não existem.



FFIGURA 3.7- Instancias excepcionais [ROS 96]

Ao final da etapa de modelagem conceitual da aplicação são gerados diagramas e documentos sobre o domínio da aplicação. Os diagramas gerados são modelos que especificam cada um dos subsistemas envolvidos no domínio da aplicação. Estes diagramas são complementados por cartões que especificam subsistemas, classes, objetos excepcionais e relacionamentos. Os cartões incluem informações sobre a constituição de cada elemento abstrato e informações sobre rastreamento.

Um rastreamento para trás representa uma informação sobre a origem de um determinado elemento abstrato, como por exemplo, classe, relacionamento ou instâncias. Já um rastreamento para frente, permite uma visão sobre as influências que determinado elemento abstrato reflete no modelo quando este sofrer alguma alteração. As informações de rastreamento indicam a origem e alvo, bem como o tipo de relacionamento com a origem e alvo. As Figuras 3.8, 3.9 e 3.10 apresentam modelos de cartões de subsistemas, classes e objetos (instâncias) excepcionais.

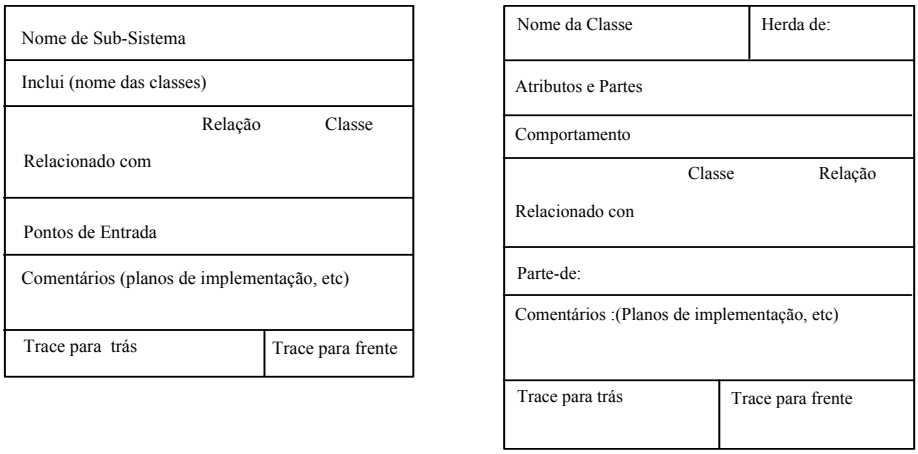


FIGURA 3.8- e 3.9- Cartões de Subsistema e de classe [ROS 96]

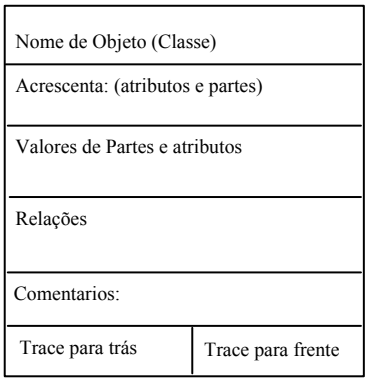


FIGURA 3.10- Cartão de Objeto [ROS 96]

2.2.2 Projeto navegacional

O projeto navegacional visa a elaboração do modelo de navegação de uma aplicação hipermídia, que é uma visão do modelo conceitual da aplicação. Nesta etapa entra a noção de navegação é uma característica particular de aplicações hipermídia. A navegação é a forma como o usuário irá interagir com a aplicação. O projeto navegacional leva em consideração o perfil do usuário que irá manipular a aplicação e, ainda, as tarefas que o usuário irá realizar durante uma sessão de navegação. Diferentes modelos navegacionais poderão ser derivados de um mesmo modelo conceitual, de acordo com as características e tarefas que o usuário irá realizar durante a manipulação da aplicação final.

O projeto navegacional é expresso em dois esquemas, o esquema de classe navegacional e o esquema de contexto navegacional. Os objetos navegacionais de um hiperdocumento são definidos em um esquema de classe navegacional. Estas classes refletem uma visão do esquema conceitual de domínio da aplicação. Em OOHDM existe um conjunto de tipos de classes navegacionais, como: nós, elos e estruturas de acesso, semelhante a HDM [GAR 93] e RMD [IZA 95].

Os nós são visões orientadas a objetos de classes conceituais. Com a utilização de uma linguagem de consulta definida em [KIM 91] e [KIM 94], pode-se especificar um nó pela combinação de atributos de diferentes classes do esquema conceitual. As classes conceituais contém atributos tipados e elos de ligação que podem ser simples ou compostos. A descrição dos nós define um grupo de atributos e um conjunto de métodos que implementam comportamentos e são organizados em hierarquias de *parte-de* ou *é-um*. Os atributos podem ser de diferentes tipos, de acordo com suas perspectivas no modelo conceitual, mas somente um tipo deverá ser representado em uma classe navegacional.

Os comportamentos pré-definidos da classe Nó implementam o modo como os nós reagem quando ativados durante a navegação, como reagem quando uma âncora é selecionada e a maneira como interagem com os elos e com suas interfaces. Além disto, comportamentos adicionais, específicos da aplicação, são desejáveis no caso de lidarmos com aplicações híbridas, isto é, aquelas que combinam navegação com computações mais "convencionais", ou aquelas em que certos tipos de computações são úteis.

Um nó tem a aparência de um caixa contendo o nome do nó e seus atributos, semelhante ao das classes conceituais, ver Figura 3.11. Uma especificação completa do nó é documentada em Cartões de Nós definidos conforme a Figura 3.12.

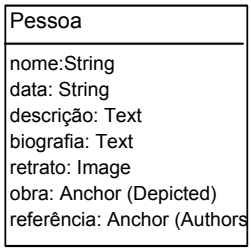


FIGURA 3.11- Um nó [ROS 96]

Os nós podem ser acessados em diferentes contextos, podendo os seus atributos e âncoras sofrerem alterações de um contexto a outro. As primitivas de hierarquias de generalização/especialização organizam as estruturas de nós implementando herança, do mesmo modo como as classes conceituais herdam de suas superclasses e podem ser atômicos ou compostos.

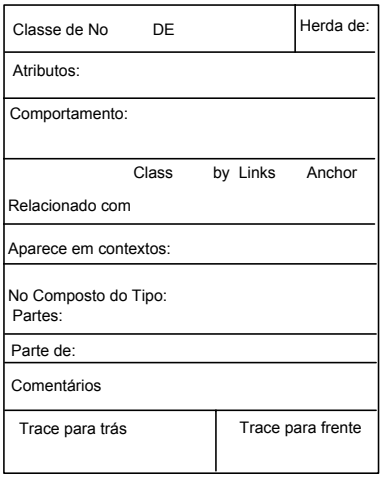


FIGURA 3.12- Cartão de um nó [ROS 96]

Existem características especiais que um nó pode adotar. Isto é observado devido ao fato de que um nó pode ser uma instância particular com algumas diferenças em relação a sua classe, ou por que são visões de objetos excepcionais no esquema conceitual, conforme apresentado na seção anterior. Cartões de Instância de Nós são utilizados para a definição de nós excepcionais. Na Figura 3.13, pode-se observar um cartão instância de nó.

Os elos refletem relacionamentos encontrados no modelo conceitual e são representados por classes de elos. Estas são definidas por atributos, comportamento, origem, destino e cardinalidade. Os atributos de elos expressam propriedades do próprio elo e podem ser demonstrados durante a navegação, desde que sejam definidos objetos de interface apropriados para eles. Em tais casos o elo pode agir como um objeto intermediário durante a navegação. Uma seta é a aparência visual do elo no modelo navegacional. Veja exemplo ilustrado pela Figura 3.14.

O Comportamento do elo influencia diretamente o comportamento da aplicação. A ativação de um elo causa um efeito sobre o espaço navegacional da aplicação, que é uma alteração sobre os objetos navegacionais. Novos nós aparecem e outros podem ou não desaparecer, como o nó de origem, por exemplo. Um outro tipo de comportamento é o necessário para que o nó alvo seja alcançado. Um nó alvo pode ser fixo ou computado através de um algoritmo, uma consulta em um banco de dados, etc. Além disto, o tipo de algoritmo pode ser diferente para diferentes instâncias da mesma classe de elos. Para o tratamento destes problemas, em OOHDM as características do próprio elo e do seu destino são abstraídas em classes distintas, podem ser subclassificadas. Isto permite ao projetista mais liberdade durante as decisões sobre implementação de estratégias para encontrar um nó destino. Com a construção de hierarquias separadas: elos e destinos, pode ocorrer uma evolução independente entre estas duas classes, permitindo a combinação de diferentes objetos de elo com objetos de estratégia.

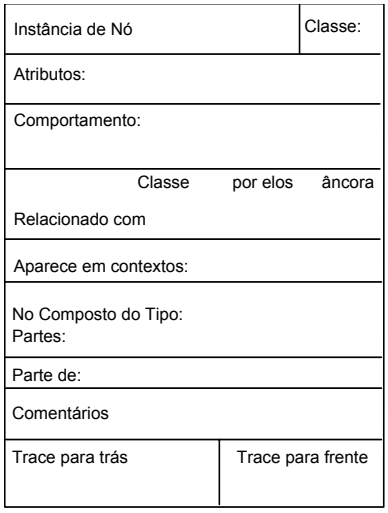


FIGURA 3.13- Cartão de Instância de Nó [ROS 96]

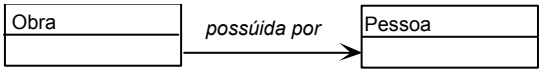


FIGURA 3.14- Um exemplo de elo [ROS 96]

Quanto as transformações navegacionais que serão executadas quando o elo for atravessado, segue a mesma definição. Uma hierarquia de classes de transformações é elaborada, fornecendo os algoritmos para diferentes tipos de transformações navegacionais. Desta forma são formadas três hierarquias durante a definição de elos, são elas:

- hierarquia de elos: que é organizada pelos tipos de origens e destinos;
- hierarquia de destino de elo: que fornece os algoritmos necessários para encontrar os nós de destino;
- hierarquia de transformações de elo: onde os algoritmos para diferentes tipos de transformações navegacionais serão implementados. Esta técnica permite que o mesmo tipo de algoritmo possa ser utilizado para diferentes tipos de elos e em diferentes contextos navegacionais. A Figura 3.15 apresenta o esquema hierárquico de classes relacionadas aos elos.

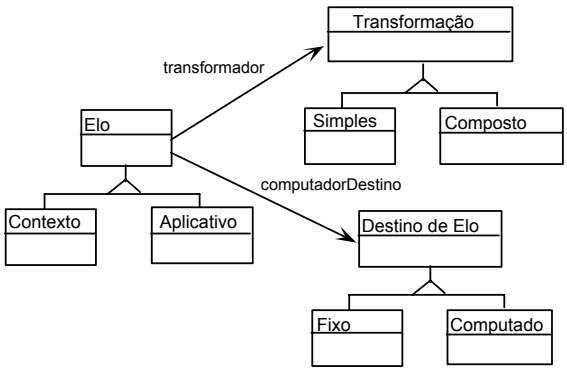


FIGURA 3.15- Elo, destino de elo e transformação de elo [ROS 96]

Âncoras são gatilhos navegacionais de uma aplicação. Quando uma âncora é ativada ocorre uma navegação. As âncoras são objetos de interface física, formando a origem de elos e estruturas de acesso de uma aplicação. A definição de uma âncora é

realizada com base no elo ou estrutura de acesso originada quando da ativação da âncora. Uma âncora não precisa ser definida como um objeto visual na interface da aplicação, como por exemplo um botão, ela pode ser implementada por tempo de sincronização entre objetos, definindo-se uma navegação por tempo.

Em OOHDM a principal estrutura de um esquema navegacional é o contexto navegacional [SCH 95]. O contexto navegacional é composto por um conjunto de nós, elos, classes de contexto e outros contextos navegacionais aninhados, definidos a partir de classes navegacionais.

Contextos navegacionais complementam o conceito de classe navegacional, como elos por exemplo, indicando que informações serão apresentadas e que âncoras serão disponibilizadas quando o usuário estiver acessando um determinado contexto. Os contextos navegacionais são semelhantes as coleções [GAR 94] e foram inspirados no conceito de contextos aninhados [CAS 93]. Os contextos navegacionais podem ser de vários tipos, classificados de acordo com suas origens, são eles: arbitrários, derivados de classe, derivados de um elo e derivados de um composto.

Contexto navegacional arbitrário: seus itens são definidos arbitrariamente como em um roteiro guiado [TRI 88]. Os elementos do roteiro guiado podem ser originados de instâncias de diferentes classes navegacionais;

- contexto navegacional derivado de uma classe: é formado por todas as instâncias de uma determinada classe. Este é um tipo de contexto que pode ser especializado por um conjunto de instâncias de uma classe que satisfaçam um determinado predicados de atributos;
- contexto navegacional derivado de um elo: determina uma navegação originada no conjunto de nós que serão acessados a partir de um determinado elo. Isto ocorre quando ativamos um elo do tipo *um-para-muitos*;
- contexto navegacional derivado de um composto: forma um conjunto de partes de um nó.

As classes navegacionais e o esquema de navegação são boas fontes para se encontrar contextos navegacionais. A grande maioria dos contextos navegacionais são originados de classes de contexto, sendo que alguns objetos navegacionais podem ser criados a partir de contextos navegacionais, tais como: estruturas de acesso, elos de contexto e outros. A documentação de um contexto navegacional é realizada pela utilização de um cartão de contexto, conforme ilustrado na Figura 3.16.

Os esquemas de navegação e de contextos navegacionais especificam a estrutura navegacional estática de uma aplicação hipermídia, entretanto aplicações apresentam padrões dinâmicos complexos quando a navegação é colocada em funcionamento. A dinâmica da estrutura navegacional é complementada pela especificação das transformações que ocorrem quando da transição entre os nós. Durante o projeto navegacional é necessário especificar de modo preciso o que acontece a cada vez que uma conexão é completada, em termos do conjunto de objetos navegacionais acessíveis no espaço navegacional da aplicação. Em OOHDM é utilizado o modelo de transição de estados orientado a objetos derivado de *Statecharts* [HAR 87]. Este modelo é denominado de diagrama navegacional, suportando estruturas de comportamento aninhado para expressar o comportamento navegacional dinâmico da aplicação.

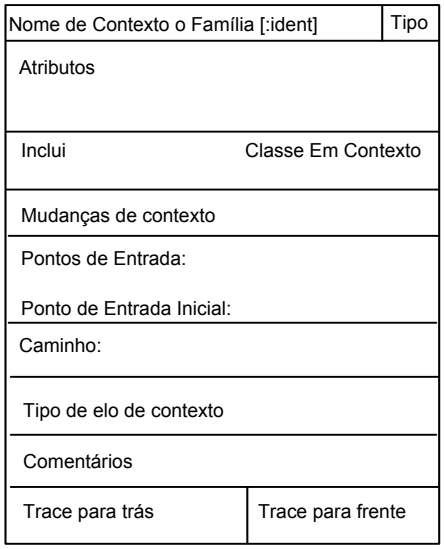


FIGURA 3.16- Cartão de contexto [ROS 96]

Em uma definição de comportamento navegacional em OOHDM, pode-se especificar que quando o usuário abandonar um nó, este será retirado do espaço navegacional até que seja ativado novamente, ver Figura 3.17.

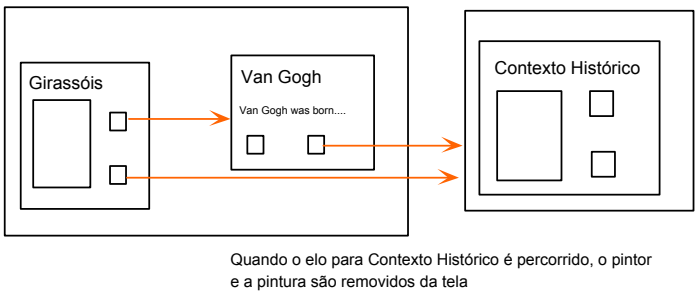


FIGURA 3.17- Transformação navegacional simples [ROS 96]

Entretanto, pode-se especificar um comportamento um pouco diferente, onde nós distintos poderão ser mantidos ativos no mesmo espaço navegacional. Isto pode ser implementado em um ambiente de múltiplas janelas, de modo a permitir a abertura de duas janelas diferentes, uma para cada nó. A Figura 3.18 ilustra esta especificação.

Em OOHDM é utilizado um formalismo diagramático chamado Diagramas de Navegação para a especificação de transformações navegacionais. A escolha de Diagramas de Navegação é devido a este ser um formalismo análogo a outros discutidos anteriormente como o esquema navegacional da aplicação. Além disso, os Diagramas de Navegação são facilmente mapeados para Statecharts**,** e pode-se usar ferramentas de análise já existentes para estes Diagramas como Statemate [ZHE 92].

Diagramas de Navegação tem por objetivo definir a dinâmica da aplicação em termos de transformações ocorridas no espaço navegacional. Para isso, oferecem um formalismo visual que permite a especificação do modo como os nós serão abertos, se simultaneamente ou mutuamente exclusivos, assim como condições de abertura dos nós. Isto reflete mudanças de estado no contexto navegacional, portanto OOHDM utiliza um modelo de transição de estados baseados em Statecharts, com uma notação adaptada. Um Diagrama de Navegação é uma adaptação de um Statechart para o campo da

hipermídia, garantindo o mesmo poder de computação do formalismo original de [HAR 87].

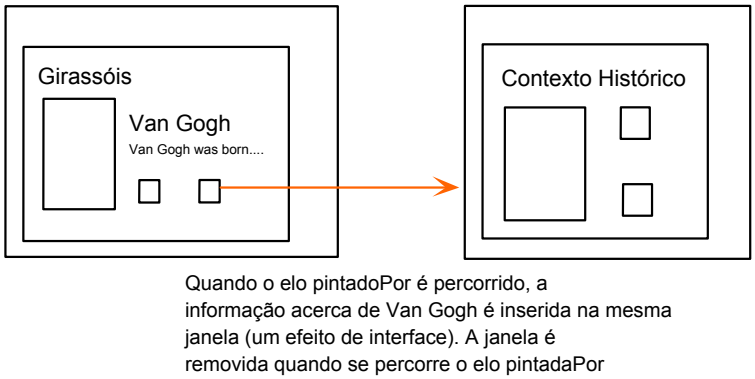


FIGURA 3.18- Outra interface para a transformação navegacional [ROS 96]

Um Diagrama de Navegação é composto por objetos navegacionais, como nós e estruturas de acesso e, ainda, estados e transições. Um objeto navegacional é chamado de *Nob* e é representado com pequenas caixas de cantos arredondados com um nome de identificação na parte superior. Todo *Nob* contém um estado no mínimo. Caso contenha somente um estado, este será seu estado ativo.

Os *nobs* e os Estados podem ser aninhados como nos statecharts em tipos de aninhagem *AND* ou *XOR*. A aninhagem *AND* é usada em statecharts para expressar simultaneidade. A aninhagem *XOR* representa a decomposição de um estado em outros, mutuamente exclusivos. Um mesmo *nob* pode estar em dois estados aninhados distintos.

As transições podem ser realizadas de duas formas, entre estados ou entre *nobs*. Sendo que a transição entre *nobs* representam tipos de elos. Estes dois tipos de transição são representados por setas que ligam dois estados, um de origem e outro destino. Uma tripla composta por evento, pré-condição e pós-condição, pode ser utilizada para anotar a transição.

Uma transição entre dois *nobs* gera uma mudança no espaço navegacional. A origem é fechada e o alvo aberto. Quando o alvo é um *nob* composto, todos os *nobs* internos são fechados. Caso a transição for assinalada com um marcador, os *nobs* de origem são mantidos abertos. Uma transição de saída de um estado exerce o mesmo efeito nos *nobs* internos, se existir um marcador na origem, todos que já estavam abertos permanecem abertos, caso contrário são fechados. Para a implementação de comportamentos mais complexos, deve-se utilizar a tripla evento, pré-condição e pós-condição. Quando as transformações navegacionais forem simples, os diagramas de navegação podem ser construídos pela modificação do esquema navegacional e permanecem facilmente compreensíveis.

Os cartões de estado e de transição completam a documentação do modelo. Estes modelam as diferentes condições de ativação e desativação de *nobs*. A Figura 3.19 ilustra um diagrama de navegação simples e na Figura 3.20 o seus cartões de estado e transição.

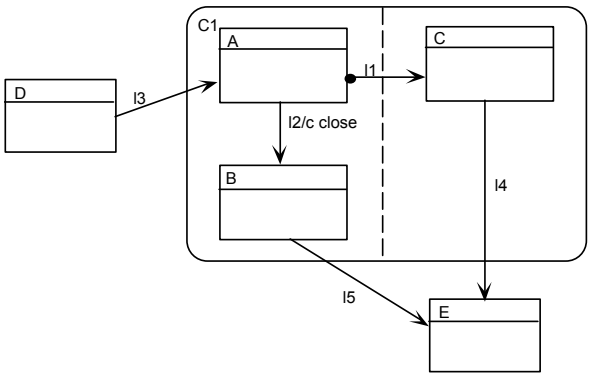


FIGURA 3.19- Diagrama de Navegação Simples [ROS 96]

Estado: C1
Aninhado em: Main Sub-estados:
Inclui: A,B,C (Nos)
Transições que entram: l3 (de Main)
Transições que saem: l5 (a Main)
Transições internas: l1, l2
Comentários: Usar janelas diferentes para A e C

Transições (C1)
Nome: l1 Pós- Cond: open C Trace info: ADV A-Button l1
Nome: l2 Pós-Cond: close A, close C Trace info: ADV A-Button l2

FIGURA 3.20- Cartões de Estado e Transições [ROS 96]

2.2.3 Projeto da interface abstrata

Uma vez que a estrutura navegacional foi definida a interface da aplicação deve ser elaborada, levando-se em consideração uma perspectiva do usuário através da aplicação. Já foi dito que a construção de uma interface Hipermídia é um aspecto crítico da criação de uma grande aplicação hipermídia [SCH 94a]. Sendo assim, a questão de interfaces homem-máquina deve ser levada em consideração, pois é tão importante quanto o projeto conceitual e navegacional da aplicação, possuindo vários problemas de projeto associados. No projeto da interface abstrata devem ser considerados os seguintes aspectos:

- a aparência de cada objeto navegacional que será visualizado pelo usuário durante a navegação. Um mesmo objeto navegacional poderá possuir diferentes representações, de acordo com a situação navegacional. Um nó, por exemplo, pode ser visto como uma figura, caso represente uma informação; ou como um ícone em um mapa de imagens, representando um índice;
- os relacionamentos entre os objetos de interface e os objetos navegacionais, que são gerados através dos eventos ocasionados durante a navegação pelo usuário;
- as transformações de interface que ocorrem devido aos eventos externos gerados por interações do usuário, influenciando no comportamento dos objetos de interface;
- e a sincronização de alguns objetos de interface, principalmente as mídias dinâmicas como áudio e vídeo.

Uma separação entre o projeto navegacional e a interface abstrata é evidenciada, para que diferentes interfaces possam ser produzidas para um mesmo modelo navegacional. Desta forma, atendendo a diferentes tecnologias de interface do usuário, bem como a variedade de preferências do usuário. Outro aspecto importante é a desvinculação do projeto de interface com o ambiente de implementação, separando o processo de projeto de ferramentas de implementação.

Existem inúmeros recursos para criação de interfaces, através de bibliotecas para definição de interfaces, onde os objetos de mídia e programação das transformações de interface podem ser implementadas sem muito esforço. Entretanto, um modelo formal de projetos deve ser usado anteriormente à implementação, de modo a maximizar a independência de diálogo e o reuso em grande escala dos componentes de interface. Além disto, um bom modelo de interface pode aperfeiçoar a comunicação entre projetistas e implementadores durante as decisões de projeto de interface. A documentação de projeto pode ser usada tanto como um campo de testes para validar a implementação, como uma referência durante a manutenção [ROS 96].

OOHDM utiliza o conceito de ADV (*Abstract Data View*) para o projeto de interfaces [CAR 94a][CAR 94b][COW 93][COW 95]. ADV é um modelo formal orientado a objetos que utiliza agregação e generalização/especialização como mecanismos de abstração. ADVs expressam a estrutura de *layout* estático e dinâmicos através das metáforas de interface de [SCH 94b], assim como seus relacionamentos com o modelo navegacional.

Um ADV permite a definição da aparência de uma interface de objetos navegacionais de forma independente de implementação, onde as propriedades de percepção são especificadas por atributos. A Figura 3-21 apresenta o modelo de um ADV. Neste exemplo o ADVPintura é composto por três ADVs, o ADVImagem, o ADVTexto e o ADVBotão.

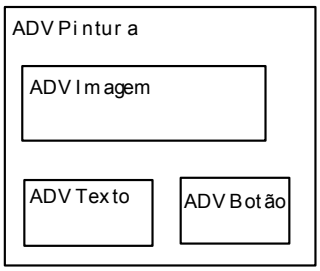


FIGURA 3.21- Composição de um ADV [ROS 96]

Os ADVs são objetos, pois possuem um estado e uma interface. As interfaces são ativadas através mensagens de outros objetos, ou de eventos de entrada e saída. Os ADVs são abstratos porque representam apenas a interface e o estado, e não a implementação.

Em uma aplicação comum que utilize ADVs, há um conjunto de ADOs gerenciando estruturas de dados e controle dentro da aplicação, assim como um conjunto de objetos de interface (instâncias de ADVs) gerenciando aspectos de interface da aplicação, tais como entrada de usuário e saída do sistema ao usuário.

Um ADO é constituído por um ou mais ADVs que representam algum aspecto de seu estado. Cada ADV deve ser consistente com seu ADO proprietário e todos os ADVs possuídos por um ADO devem ser consistentes entre si. Estes dois tipos de relacionamentos de consistência são chamados, respectivamente, de consistência vertical e horizontal e são ilustrados na Figura 3.22.

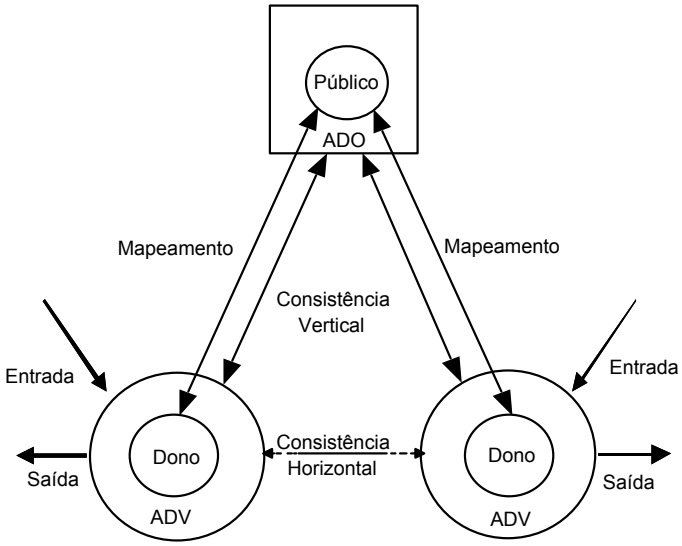


FIGURA 3.22- ADVs e ADOs [ROS 96]

Diagramas de Configuração foram definidos no contexto do modelo *ObjectCharts* [COL 92] e são utilizados para expressão de padrões de comunicação entre objetos, em termos de serviços oferecidos e requeridos. No projeto de interface, os ADVs são usados na representação de eventos gerados pelo usuário e que são gerenciados pelo próprio ADV. Exemplos são a exibição de conteúdo e comunicação entre ADVs e ADOs. Um diagrama de configuração também pode mostrar a estrutura de aninhamento de ADVs compostos.

Em OOHDM o maior interesse está no modo como o usuário irá interagir com o aplicação hipermídia, principalmente nos objetos de interface que causarão a navegação. Os ADVs representam uma interface pública com os serviços oferecidos por seus ADOs e, como os ADOs representam os objetos navegacionais, temos um modelo de interação entre os objetos navegacionais e os objetos de interface. A Figura 3-23 apresenta um diagrama de configuração para o *ADVPintura*. Veja que o *ADVPintura* reage a eventos externos gerados pelo usuário como: botão do mouse pressionado e botão do mouse pressionado duas vezes, comunicando-se com o seu dono (o ADO *NóPintura*) através das mensagens *pegarImagem, pegarComentários* e *âncoraSelec.* Os diagramas de configuração permitem a especificação da interface abstrata de uma aplicação hipermídia, ou seja, seus objetos de interface, seus relacionamentos estruturais e os relacionamentos entre objetos de interface (ADVs) e objetos navegacionais (ADOs).

Os ADV-*Charts* são uma generalização dos *statecharts* [HAR 87] e *ObjectCharts* [COL 92], sendo utilizados para a definição do comportamento de reação aos eventos externos, tais como o fluxo de navegação e as transformações de interface que ocorrem quando o usuário interage com a aplicação. ADV-*Charts* adotam estruturação e comportamento aninhado e uma notação semelhante as redes de Petri para expressar sincronização, que é uma característica típica dos dados multimídia.

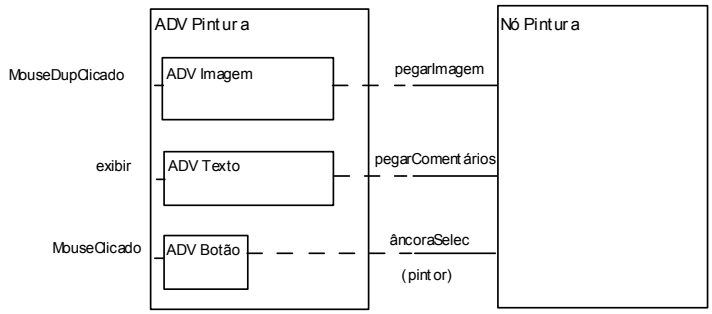
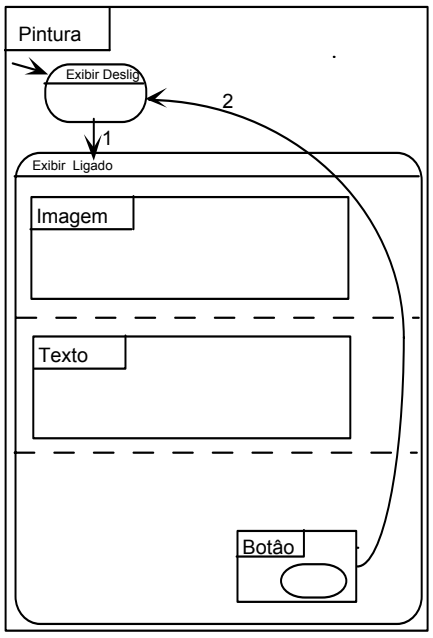


FIGURA 3.23- Diagrama de Configuração [ROS 96]

Os construtores para a modelagem são muito similares aos do projeto navegacional e de interface abstrata. De fato, são utilizadas classes e objetos em um modelo formal de conexão, permitindo a transição entre os dois modelos de forma incremental. A Figura 3-24 apresenta um modelo de um ADV*Chart*.

O ADV*Chart* apresentado na Figura 3-24 possui dois estados mutuamente exclusivos (**OR**): "Exibir Desligado" ou "Exibir Ligado". A transição 1 leva ao estado "Exibir Ligado", que é ocasionada pelo evento externo "Exibir" (gerado por uma interação do usuário), tornando as informações apresentadas pelo estado "Exibir Ligado" ativas, formando assim um novo contexto perceptível (contextoPerceptível = contextoPerceptível + Pintura). O estado "Exibir Ligado" é composto por três estados simultâneos (**AND**), correspondentes aos ADVs Imagem, Texto e Botão. A Transição 2 que leva ao estado "Exibir Desligado" é disparada quando o evento "Mouse Clicado" é acionado por interação do usuário. Esta transição é especificada através da função "Foco" que indica que a interação ocorreu sobre o botão, gerando outro contexto perceptível (contextoPerceptível = contextoPerceptível – Pintura), aplicando a operação "âncora selecionada" no dono do ADVPintura.



Transições

1:
Evento: Exibir
Pré-Cond:
Pos-Cond: contextoPerceptivo = contextoPerceptivo + Pintura

2:
Evento: MouseClicado
Pré-Cond: foco (Botão)
Pos-Cond: contextoPerceptivo = contextoPerceptivo -Pintura; dono.âncoraSelec (pintor)

FIGURA 3.24- Um ADVChart [ROS 96]

ADV*Charts* são semelhantes aos *statecharts* pois possibilitam a especificação do comportamento de objetos compostos através da definição de estruturas e comportamentos aninhados.

A partir de um esquema navegacional da aplicação, deve-se realizar os seguintes passos:

- definição ADVs para cada objeto navegacional encontrado no esquema navegacional;
- especificação do diagrama de configuração, demonstrando os relacionamentos estáticos entre ADVs e ADOs e entre ADOs e seus objetos navegacionais;
- especificação de ADV*Charts* para cada ADV apresentando os comportamentos dinâmicos da aplicação.

2.2.4 Implementação

A implementação de uma aplicação hipermídia leva em consideração questões técnicas e não-técnicas que devem ser resolvidas. Após a definição do ambiente de implementação, o projeto deve ser mapeado para objetos de implementação e todos os componentes hipermídia têm de ser instanciados.

O ponto chave é a definição dos objetos de interface de acordo com a especificação da interface abstrata, implementação das transformações na forma como foram definidas nos ADVcharts e fornecimento de suporte para a navegação através do hiperdocumento [ROS 96].

Na definição dos objetos de interface a informação fornecida pela especificação baseada em ADVs e, em especial, a estrutura aninhada dos ADVs, oferece uma indicação sobre quais os objetos de interface que precisamos definir.

Durante a programação da interface, deve-se observar os aspectos dinâmicos da interface, tanto nas transformações de interface dentro de um ADV como nas transformações de interface envolvendo navegação. Em ambos os casos, a natureza orientada a eventos dos ADVcharts e o poder expressivo de seu modelo de transição de estados permite a implementação destas transformações em um estilo *bottom-up* [ROS 96].

Uma navegação ocorre, geralmente, quando o usuário seleciona uma âncora em um nó. Entretanto, torna-se possível implementar um comportamento navegacional mais rebuscado, tal como navegação baseada em tempo. Este tipo de comportamento navegacional é especificado através de ADVcharts, onde transições de estado e o evento gerador de cada mudança estão documentados na especificação da transição.

2.3 RMM (*Relationship Management Design Methodology*)

O RMM é uma metodologia para projeto e desenvolvimento de aplicações hipermídia baseadas no gerenciamento de relacionamentos entre objetos de informação [IZA 95]. RMM é adequada para a especificação de aplicações que possuem um domínio regular e estruturado, através de classes, objetos e relacionamentos que podem ser estabelecidos entre estas classes. Existindo múltiplas instâncias de objetos em cada

classe. Estas aplicações devem requerer freqüentes atualizações das informações. Exemplos de tais aplicações são catálogos de produtos e *front-ends* baseados em hipermídia para banco de dados tradicionais. Como estes dados precisam de freqüentes atualizações, faz-se necessário adotar uma abordagem que automatize os processos de projeto e atualização destas aplicações. Entretanto, em aplicações que não requerem constantes atualizações ou aquelas que possuem estruturas dinâmicas, RMM não é uma metodologia adequada para a especificação. A Tabela 3.1 ilustra a utilização da metodologia RMM em função da estruturação e atualização da informação empregada em uma determinada aplicação.

RMM utiliza como modelo de dados o RMDM (*Relationship Management Data Model*). O modelo de dados é um conjunto de objetos lógicos que fornecem uma abstração de uma porção do mundo real. O RMDM fornece uma linguagem para descrição dos objetos de informação e dos mecanismos de navegação em aplicações hipermídia.

TABELA 3.1- Utilização da Abordagem RMM

		Volatility of Information	
		Low	High
Structure	Low	Medium Usefulness [e.g., Kiosk Application]	High Usefulness [e.g., Product Catalog, DBMS Interface]
	High	Not Useful [e.g., Literacy Work]	Low Userfulness [e.g., Multimídia news service]

Fonte: "RMM: A methodology for structured hypermedia design" [IZA 95].

O RMDM possui uma linguagem que descreve os objetos de informação e os mecanismos navegacionais em uma aplicação hipermídia. Esta linguagem define as primitivas de domínio e primitivas de acesso.

O método RMM, como todo método de projeto de software, possui um ciclo completo de desenvolvimento. Este ciclo envolve as etapas de estudo de viabilidade, análise de requisitos, projeto E-R, projeto de fatias, projeto navegacional, projeto da interface com o usuário, construção, teste e avaliação [IZA 95]. Em cada fase do método são gerados alguns produtos como resultado da especificação. Observe, através da FIGURA 3-25, que durante as fases iniciais são gerados os documentos de possibilidades, requisitos e definição de hardware. Através do documento de possibilidades será avaliada a necessidade, objetivos e análise de mercado para a futura aplicação, bem como um levantamento sobre mídias que serão utilizadas, canais de distribuição e custo/benefício.

Como o desenvolvimento de hiperdocumentos é diferenciado através da descrição das etapas envolvidas no projeto dos mecanismos de acesso à informação, Izakowitz [IZA 95] enfoca, principalmente, estes aspectos. Sendo assim, serão enfatizados os seguintes passos: projeto E-R, projeto de fatias e projeto navegacional, identificados no interior da caixa cinza representada na Figura 3.25. Cada uma destas fases, também gera um produto. A fase de projeto E-R gera um diagrama entidade relacionamento que modela o domínio da aplicação, através das entidades e seus respectivos relacionamentos.

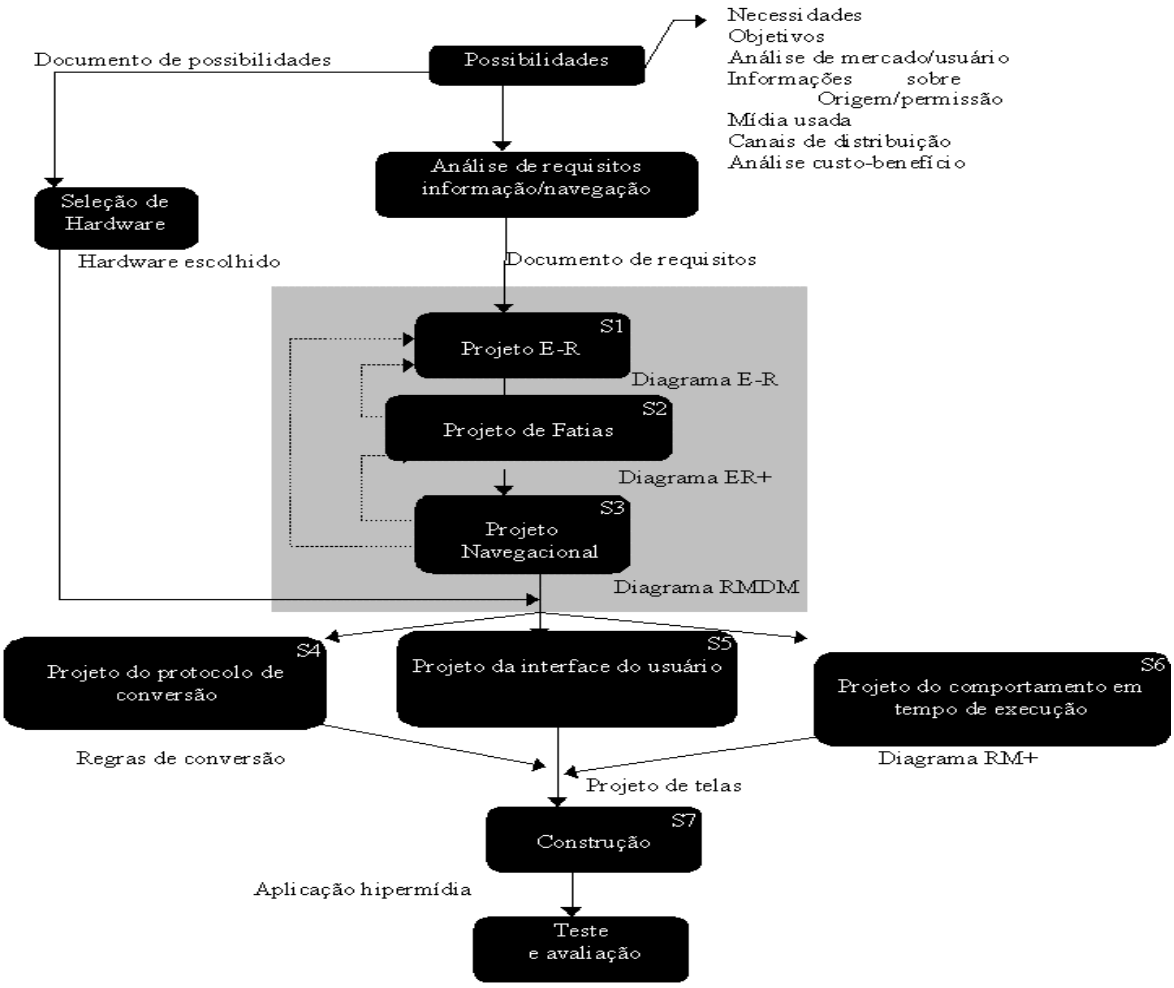


FIGURA 3.25- Fases do Método RMM [IZA 95]

Na fase de projeto de fatias o diagrama E-R é acrescentado por um modelo que especifica as partes de uma informação – denominado diagrama E-R+. Já a fase de projeto navegacional especifica o modelo de navegação da aplicação, definindo as estruturas de acesso às informações, ao final desta fase será gerado o diagrama RMDM. Note que o processo de projeto não é linear, existe um *feedback* durante as fases de projeto que são demonstrados por linhas tracejadas.

2.3.1 Projeto entidade relacionamento (E-R)

No projeto E-R será modelado o domínio da aplicação na forma de um diagrama entidade relacionamento (E-R). Nesta etapa as entidades e relacionamentos que fazem parte do domínio da aplicação são estudados, pois formam a base da aplicação. Muitas destas entidades e relacionamentos aparecerão na aplicação final na forma de nós e ligações. O diagrama E-R serve de base para a identificação dos principais relacionamentos que formarão as estruturas de navegação da aplicação.

No exemplo, demonstrado através da Figura 3-26, pode-se verificar uma parte do diagrama E-R de uma aplicação acadêmica. Note que os relacionamentos expressam como a informação será apresentada. Posteriormente, estes relacionamentos serão transformados em estruturas de acesso.

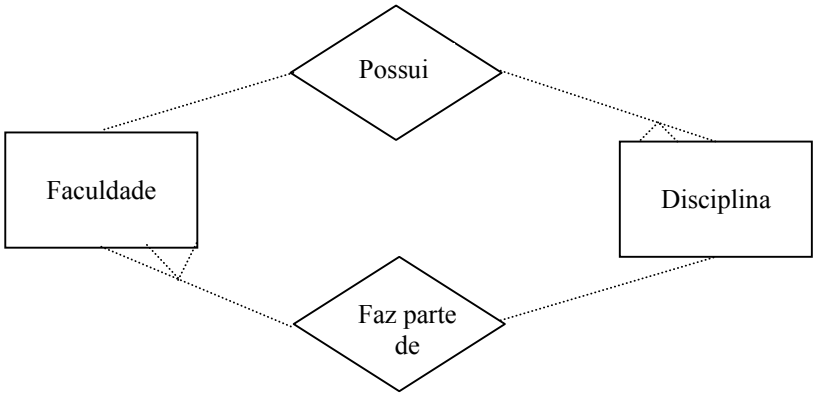


FIGURA 3.26- Diagrama ER, modelo RMDM.

As primitivas de domínio modelam a informação relevante ao domínio da aplicação. Os conceitos de entidade a atributos representam classes de objetos físicos ou abstratos do mundo real, como por exemplo: pessoa ou conta bancária. Os relacionamentos descrevem as associações entre os diferentes tipos de entidades.

A Figura 3-27 ilustra as primitivas de domínio do modelo E-R do método RMDM. Observe que as entidades são representadas por retângulos de cantos arredondados. Os atributos das entidades são representados por elipses. Os relacionamentos são representados por linhas tracejadas, onde os relacionamentos um-para-muitos são diferenciados dos relacionamentos um-para-um por uma seta na extremidade.

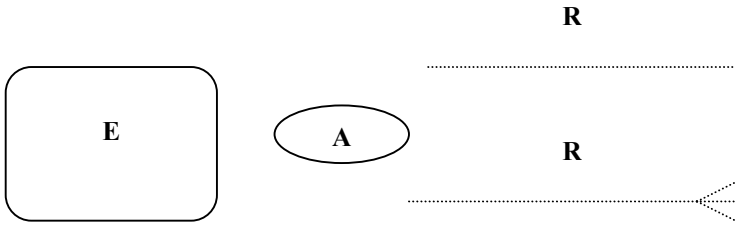


FIGURA 3.27- Primitivas de domínio.

2.3.2 Projeto de fatias

Uma característica própria de aplicações hipermídia é o conceito de porções de informações que poderão ser apresentadas ao mesmo tempo. Como as entidades possuem um grande número de atributos de diferentes naturezas, podendo ser desnecessário ou impraticável apresentá-los todos juntos durante uma navegação. Foi acrescentado o conceito de *slice* (fatia). Um *slice* permite o agrupamento de um conjunto de atributos de uma mesma entidade. Assim, uma mesma entidade pode ser subdivida em fatias para melhor representar a informação.

As fatias determinam como uma determinada parte da informação será apresentada e acessada pelo usuário. Uma determinada entidade pode ser subdividida em fatias que agrupam atributos relacionados e que são apresentadas de forma separada. Esta é uma técnica para a divisão da carga de informação ao usuário. Durante o projeto de fatias a divisão de entidades em porções menores deverá ser realizada de forma que a informação seja melhor estruturada. Uma das fatias deverá ser a principal e servirá como ponto de referência para a entidade.

A Figura 3-28 apresenta duas fatias da entidade professor. Esta entidade pode ser subdividida em duas porções menores para melhor representar informações. Uma porção (*slice*) pode ser definida como geral e outra como formação. A fatia geral pode ser composta por informações gerais a respeito do docente, tais como: nome, endereço e telefone. Enquanto que a fatia formação apresenta informações curriculares, como: nome, formação acadêmica e atividades.

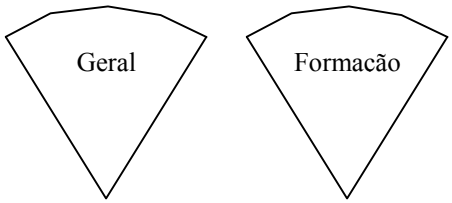


FIGURA 3.28- Parte das Fatias da entidade Professor

As fatias de informação de uma aplicação são modeladas através de um diagrama de fatias. A Figura 3-29 apresenta um diagrama de fatias da entidade Professor para uma aplicação acadêmica. As ligações unidirecionais e bidirecionais modelam a navegação entre as fatias. Estas ligações são denominadas ligações estruturais, diferenciando-se das ligações associativas do diagrama E-R. Uma ligação estrutural modela a navegação entre partes de uma mesma entidade. Enquanto que uma ligação associativa modela as relações entre entidades. Estes dois tipos de conexão, estrutural e associativo, são diferenciados para uma melhor especificação do contexto navegacional. Isto é evidenciado, pois quando o usuário navega por uma ligação estrutural o contexto é mantido, mas se o usuário navega por uma ligação associativa estará ultrapassando os limites de uma entidade, mudando de contexto. Para diferenciar os tipos de conexão, relacionamentos associativos são indicados graficamente por linhas pontilhadas e relacionamentos estruturais por linhas cheias.

2.3.3 Projeto navegacional

O projeto navegacional permite a elaboração dos caminhos de navegação em hiperdocumentos. Estes caminhos são definidos pelos relacionamentos associativos constantes no diagrama E-R. O projetista define se um determinado relacionamento associativo será um caminho de navegação, através da transformação do relacionamento em uma estrutura de acesso. O processo inicia pelo projeto de navegação entre as entidades estruturais. Uma estrutura de acesso entra em uma determinada entidade pela sua fatia de principal, sendo que o projetista pode identificar um outro ponto de entrada como *default.* O produto final do projeto navegacional é o diagrama RMDM.

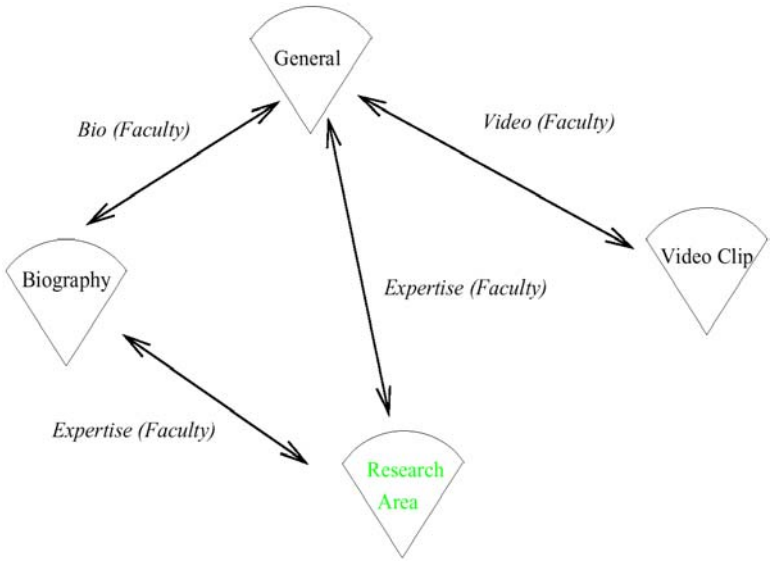


FIGURA 3.29- Fatias da entidade Professor [IZA 95]

As primitivas de acesso possibilitam a descrição dos mecanismos de navegação do modelo RMDM. Estes mecanismos traduzem o modelo conceitual em estruturas de navegação.

- Ligações unidirecionais e ligações bidirecionais: permitem a especificação de acesso entre fatias de uma mesma entidade. A navegação é realizada somente em porções de informação de uma mesma entidade, nunca ultrapassam os limites de uma entidade.
- Índices de acesso: um índice representa uma lista de instâncias de entidades permitindo acesso direto a cada elemento da lista.
- Roteiro guiado: permitem a criação de um caminho linear por um conjunto de elementos, permitindo ao usuário percorrer o caminho para frente e para trás. Existem variações de um roteiro guiado como roteiro guiado circular, onde o último elemento aponta para o primeiro; roteiro guiado que retorna para o principal, quando existe um nó que contém informações sobre o próprio roteiro agindo como o início e o fim da navegação; e um roteiro guiado de entrada e saída, onde existe um nó de entrada diferente do nó de saída.
- Agrupamentos: permitem o acesso aos diferentes componentes do hiperdocumento. Um exemplo é um menu de opções. O índice de acesso é um caso particular de um agrupamento.

Tanto nos índices de acesso como nos roteiros guiados podem ser implementadas condições lógicas para restringir o conjunto de instâncias que podem ser acessadas. Observe, através da Figura 3-30, um diagrama RMDM completo de um sistema de informação acadêmico. Note que o diagrama RMDM, complementa o modelo entidade relacionamento através da especificação de caminhos de navegação que o usuário poderá percorrer. Para simplificar a especificação e tornar o diagrama mais claro, *slices* não estão inclusos no modelo. Ao topo do diagrama está um menu principal da aplicação, onde o usuário poderá optar por começar navegação entre *Faculty, Courses* ou *Programs.* Caso o usuário optar por *Faculty* ou *Courses* serão

desencadeados roteiros guiados para cada caso. Se o usuário optar por *Programs*, o acesso será por meio de um índice. Entre *Faculty* e *Courses* existe um índice condicional determinado pelo predicado *Teaches(F, C)* que permite a navegação entre *Faculty* e *Courses*. E o índice condicional recíproco determinado pelo predicado *taught_by(C, F)* que permite o acesso entre *Courses* e *Faculty*. Esta especificação ilustra um relacionamento muitos-para-muitos entre as entidades *Faculty* e *Courses*.

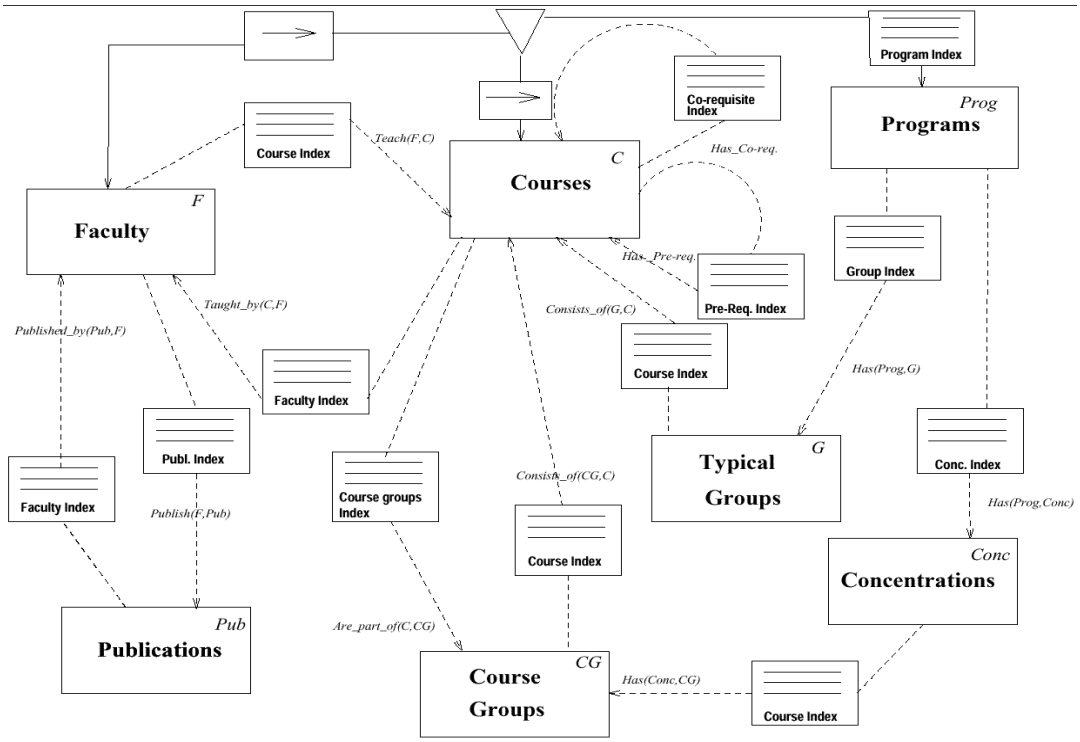


FIGURA 3-30. Diagrama RMDM para um sistema de informação [IZA 95]

2.4 HMT (Hypermedia Modeling Technique)

HMT implementa uma técnica para a especificação e projeto de hiperdocumentos com a finalidade de auxiliar o desenvolvedor durante as tarefas de desenvolvimento de aplicações. O método leva em consideração questões como:

- divisão do domínio da aplicação em nós de informação;
- tratamento das conexões entre os nós de informação;
- e a interação do usuário com a aplicação.

Para tratar estes aspectos, HMT é subdividido em quatro etapas que utilizam modelos apropriados. O modelo de objetos, primeira etapa do processo, permite a descrição dos objetos de domínio, bem como seus relacionamentos. O modelo de hyperobject estende a etapa anterior, acrescentando mais semânticas para a descrição dos relacionamentos. O modelo de navegação descreve os elos e as estruturas de acesso da aplicação. A última etapa é o modelo de interface, onde é realizada a descrição de como o usuário perceberá os objetos de interface durante a navegação [NEM 98].

2.4.1 Modelo de objetos

O modelo de objetos descrito em HMT adota os conceitos e notações do modelo de objetos OMT (*Object Modeling Technique*), onde as estruturas dos objetos de domínio da aplicação são descritas, tais como: entidades, relacionamentos e outros objetos, atributos e operações do modelo. Segundo o modelo, uma classe de objetos descreve as propriedades e comportamentos comuns a um determinado grupo de objetos, além de seus relacionamentos com outros objetos e semânticas particulares. A classificação dos objetos leva em consideração o propósito de cada aplicação.

O objetivo do modelo de objetos HMT é descrever os objetos de domínio através de diagramas de objetos (ferramenta de descrição dos modelos), de acordo com as definições do OMT. Os conceitos de classes, atributos, operações, associações, cardinalidade, generalização e agregação são utilizados. A utilização do modelo de objetos na especificação de hiperdocumentos tem como principal vantagem o tratamento explícito de relacionamentos. Os relacionamentos formam construções e não são implícitos em atributos de classes como em outros modelos.

O modelo de objetos captura as semânticas de domínio da aplicação, onde cada classe do modelo de objetos é candidata a ser mapeada para um ou mais tipos de nós de informação. Entretanto, algumas classes do modelo poderão originar somente um tipo de nó. As associações do modelo irão gerar os elos de ligação ou estruturas de acesso da aplicação, dependendo da cardinalidade e do tipo de acesso desejado. Um aspecto importante a ser observado é que este tipo de decisão de projeto não é expressa no modelo de objetos. A Figura 3-31 apresenta um modelo de objetos de uma aplicação literária.

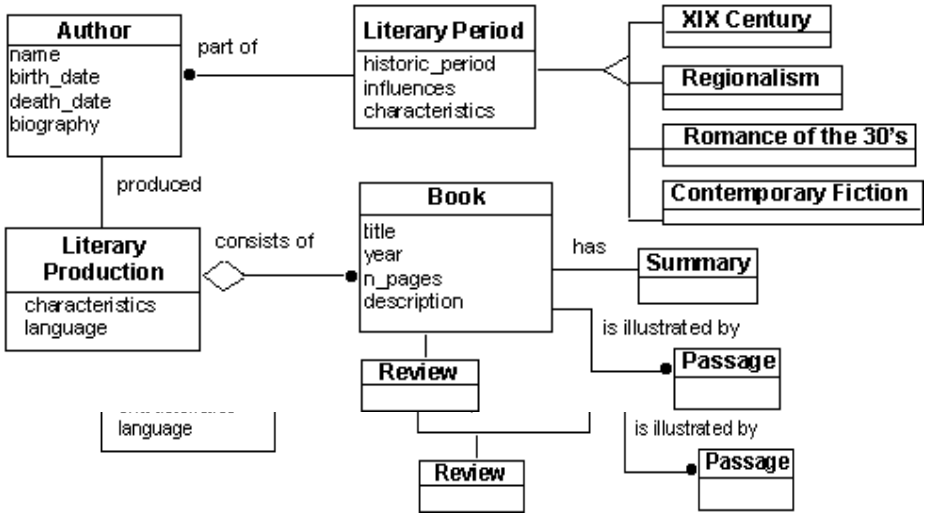


FIGURA 3.31- Modelo de Objetos de uma aplicação literária [NEM 98]

As entidades que compõem o domínio da informação são representadas por retângulos, que são divididos em duas partes. Na parte superior do retângulo é colocada a identificação da entidade e na parte inferior são colocados os atributos que formam as características dos objetos instanciados. Os relacionamentos entre as entidades de domínio são representados por linhas que unem as entidades. Estas linhas determinam o tipo de relacionamento podendo ser agregações ou generalizações/especializações, sendo que a cardinalidade é expressa na forma de círculos vazados ou sólidos na extremidade das linhas.

2.4.2 Modelo de *HyperObject*

Esta é a segunda etapa da modelagem de hiperdocumentos HMT. Trata-se de um refinamento do modelo de objetos. Este modelo inclui características que não são modeladas no modelo de objetos, como:

- definição de novas classes e associações que definem caminhos de navegação desejados;
- identificação de mídias que serão utilizadas na aplicação;
- identificação de classes abstratas.

O modelo de *HyperObject* pode ser utilizado como uma ferramenta para complementar decisões de projeto em aplicações existentes que possuem um modelo de domínio já definido.

2.4.3 Modelo de navegação

As associações determinam os caminhos de acesso entre os objetos do modelo e são definidas através de entidades conceituais utilizadas na modelagem conceitual. Na fase de modelagem navegacional algumas estratégias para guiar a implementação destas associações podem ser criadas. No modelo de objetos uma associação é uma abstração que define um relacionamento entre classes, mas em uma perspectiva de projeto ocorre uma preocupação de como estas associações serão representadas. Podendo ser representadas como um elo ou um conjunto de elos, ou ainda, como uma estrutura de acesso. A análise de como cada associação será representada não deve ser levada em consideração durante a fase de modelagem de navegação e sim como elas serão utilizadas na aplicação.

Apesar disto, contextos de navegação foram incluídos no modelo HMT. Esta característica permite não somente um melhoramento na especificação de semânticas dos elos de ligação, mas também serve como base para a especificação do modelo de interface. [NEM 98] cita que os contextos de navegação foram inspirados nas idéias propostas por [SCH 95], onde todos os objetos fazem parte de um contexto padrão e podem ser associados a outros contextos. A navegação pode ser sensitiva ao contexto, neste caso respeitando os limites do contexto padrão; ou independente de contexto, onde a busca por um determinado objeto navegacional ultrapassa os limites do contexto padrão. Após a definição de contextos navegacionais, deverá ser procedida a definição de pontos de entrada da aplicação. Cada contexto define um ponto de entrada de acordo com os requisitos da aplicação. Estes pontos de entrada indicam por onde é possível iniciar uma seção de navegação.

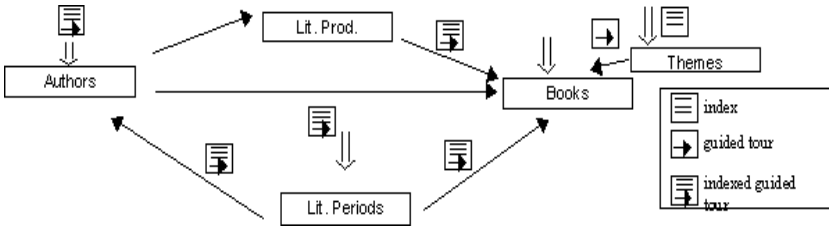


FIGURA 3.32- Modelo de navegação de uma aplicação literária [NEM 98].

2.4.4 Modelo de interface

A maneira como será dividido o material da apresentação, onde um determinado pedaço de informação poderá aparecer em diferentes contextos ou levar a diferentes rotas, pode ocasionar problemas ao usuário final da aplicação. O desenvolvedor deverá se preocupar não somente com a estrutura da informação que precisa ser apresentada ao usuário, mas também com a forma como a informação será apresentada, levando-se em consideração o contexto na qual encontra-se o usuário em determinado momento durante a navegação pela base de hiperdocumentos. Portanto, alguns mecanismos são necessários para manter a consistência e absoluta orientação durante a navegação pelo labirinto de informação [NEM 98].

A utilização de um modelo de interface permite ao desenvolvedor descrever como a informação será apresentada ao usuário. O projeto de interface inclui a definição do *layout* das telas, a aparência dos objetos e a identidade visual da aplicação. Esta definição é baseada no modelo de *hyperObject* e no modelo navegacional.

2.5 HMBS/M – *An Object Oriented Method for Hypermedia Design*

Assim como em outros métodos de projeto de aplicações hipermídia, HMBS/M possui uma seqüência de estágios de projeto. Começando pela modelagem conceitual, onde é elaborado um modelo do domínio da aplicação. Seguida por uma modelagem navegacional, que define os possíveis caminhos de navegação da aplicação baseados no perfil e tarefas que o usuário irá realizar durante a utilização do hiperdocumento. Em seguida, vem a fase de projeto da interface com o usuário com a criação de interfaces baseadas no perfil do usuário. As fases de implementação e teste concluem o método.

A estrutura organizacional e a dinâmica do comportamento navegacional do hiperdocumento são especificados através do modelo HMBS (Hyperdocument Model Based on Statecharts.

As fases do método HMBS/M são ilustradas através da Figura 3-33 Observe que o método é composto por um processo incremental, onde existe um *feedback* entre os modelos produzidos em cada fase.

Cada etapa do processo de projeto é baseada na anterior. A fase de especificação de requisitos, semelhante a outras abordagens, é a base do projeto da aplicação.

A partir desta primeira etapa, que é realizada em conjunto pelo projetista e o usuário, serão definidos os objetivos da aplicação e o produto desta etapa é o documento de especificação de requisitos.

A partir deste documento, as semânticas de especificação do método HMBS/M serão aplicadas para o projeto da aplicação hipermídia. Cada uma das etapas do método HMBS/M será apresentada a seguir.

2.5.1 Modelagem conceitual

Em posse do documento de especificação de requisitos será realizada a análise do domínio da aplicação, onde será criado o documento de representação da aplicação hipermídia. O modelo conceitual de HMBS/M é derivado do método *Fusion*, que utiliza

o paradigma de projeto de software orientado a objetos. Este método descreve as classes pela utilização de mecanismos como agregação e especialização/generalização e, ainda, os relacionamentos entre as classes do domínio.

Nesta etapa são criados dois modelos para a especificação do domínio da aplicação. O modelo de objetos e o modelo de fatias.

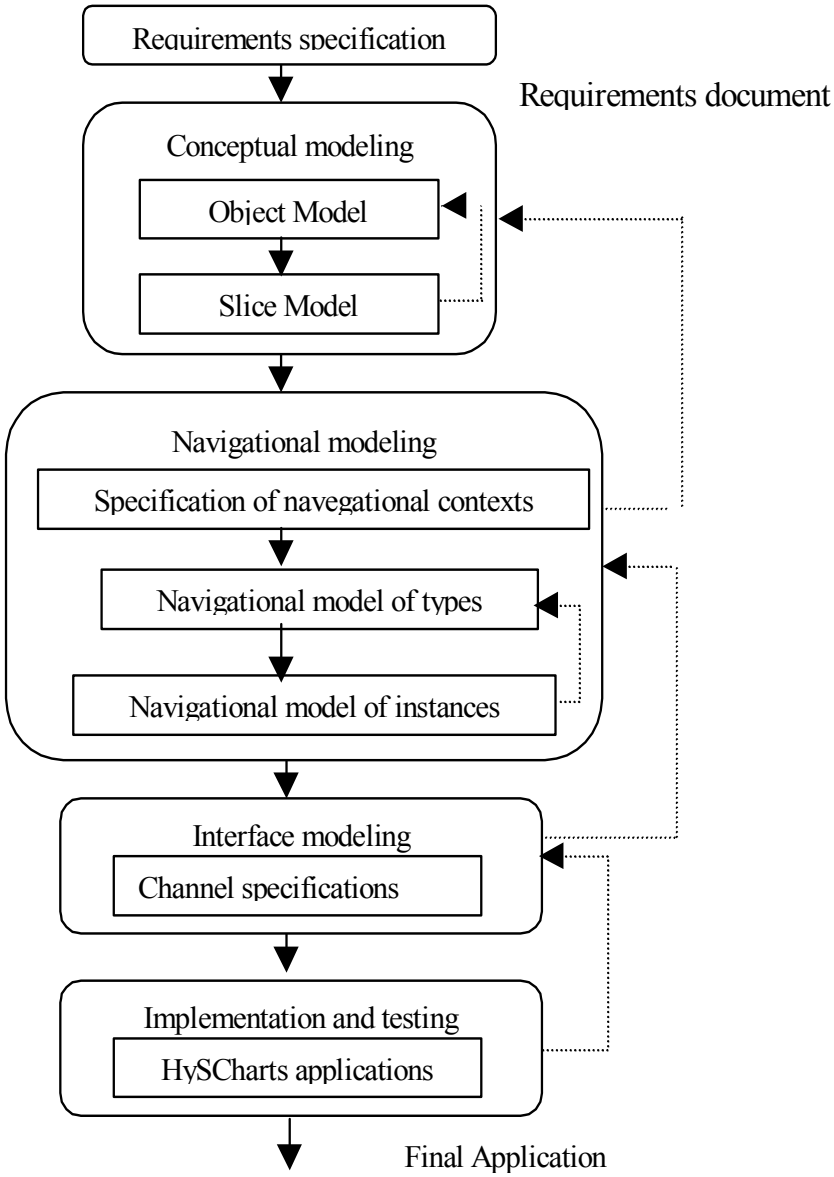


FIGURA 3.33- Fases do método HMBS/M [CAR 98]

O modelo de objetos define as informações de domínio mais relevantes (classes), através da identificação de seus atributos, bem como seus relacionamentos. Os atributos possuem diferentes perspectivas, assim como em HDM [GAR 93] e OOHDM [SCH 96]. Estas perspectivas são representadas no modelo por atributos que têm mais de um tipo e estes são modelados por colchetes. Classes complexas podem ser representadas por agregações ou especializações/generalizações. A Figura 3-34 apresenta a parte do modelo de objetos de uma aplicação acadêmica. Observe a representação das entidades com seus respectivos atributos e seus relacionamentos. Veja que o modelo apresenta atributos com diferentes perspectivas, como o atributo background da entidade Faculty. Outra característica do modelo de relacionamentos de

objetos é a subclassificação, demonstrada através da decomposição da entidade Faculty em duas subclasses: Graduate Course e Under-Graduate Course.

No modelo de fatias (*slices*) são representadas partes de informação e seus relacionamentos. Este modelo é semelhante ao adotado em RMM [IZA 95]. Existem dois tipos de fatias, a fatia principal e as fatias de ligação. Cada classe do modelo deve possuir uma fatia principal, pois por ela é realizado o acesso principal. A fatia principal é representada no modelo por um retângulo cuja extremidade superior direita é destacada.

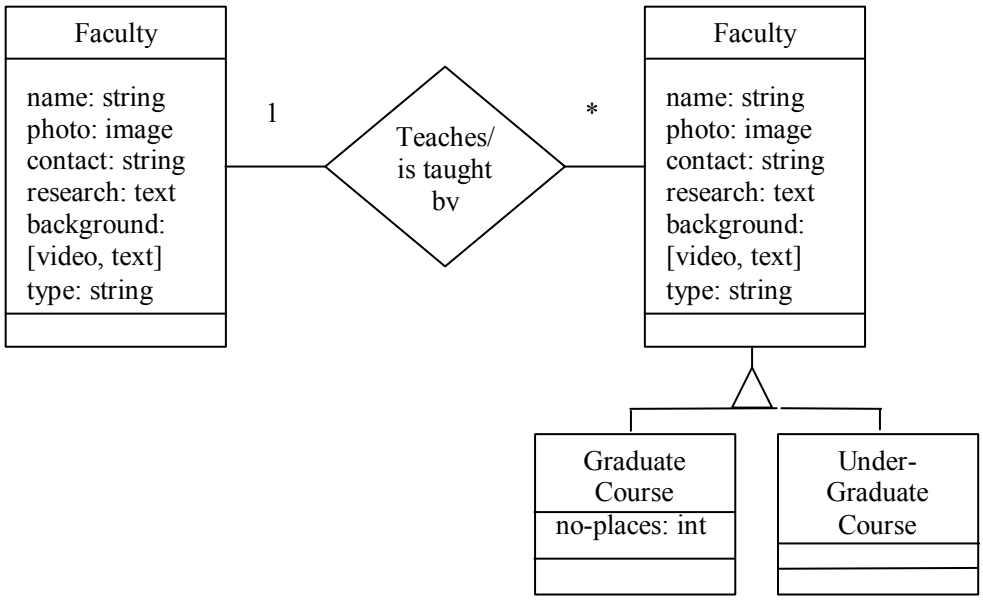


FIGURA 3.34- Parte do modelo conceitual de uma aplicação acadêmica [CAR 98].

As fatias de ligação são aquelas que permitem o relacionamento com outras classes que estão fora do escopo da aplicação. Os relacionamentos entre as fatias são representados por setas. Veja na Figura 3-35 a representação de parte de um modelo de fatias para uma aplicação acadêmica. A entidade *Course* é composta por duas fatias de informação *Main* (principal) e *Bibliography*. A fatia *Main* contém as principais informações sobre o curso. Já a fatia *Bibliography* contém informações adicionais sobre as referências bibliográficas do curso. Veja a seta de ligação entre as duas fatias que representa o relacionamento entre as duas partes de informação.

2.5.2 Modelagem navegacional

Nesta etapa de projeto, um conjunto de contextos navegacionais e estruturas de acesso são definidas, a partir dos modelos gerados na fase de modelagem conceitual. Para isso, leva-se em consideração o perfil do usuário e as tarefas que a aplicação deverá realizar durante uma sessão de navegação. Ao final desta etapa, dois modelos serão produzidos, um modelo navegacional de tipos e um modelo navegacional de instâncias.

O modelo navegacional de tipos descreve uma visão sobre o modelo conceitual de domínio da aplicação, levando-se em consideração os contextos navegacionais e estruturas de acesso identificadas a partir do modelo.

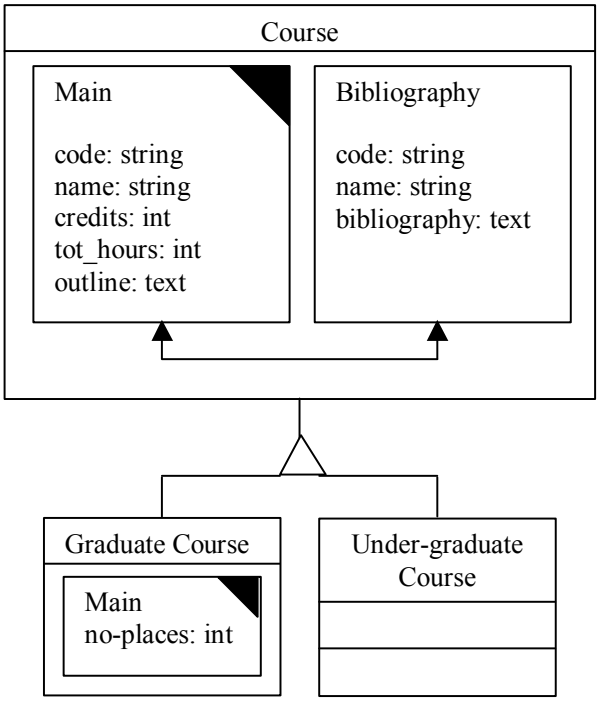


FIGURA 3.35- Parte do modelo de fatias de uma aplicação acadêmica [CAR 98]

No modelo navegacional de instâncias, as classes conceituais modeladas são instanciadas em objetos estruturais, que representam os estados de acordo com as semânticas do modelo HMBS (*Hyperdocument Model Based on Statecharts*), o modelo HMBS é apresentado na seção 3.4.5.

Antes da construção dos modelos navegacionais, deve-se identificar os contextos navegacionais da aplicação. Eles podem ser arbitrários ou genéricos e são inspirados no modelo conceitual. Um contexto navegacional arbitrário pode ser gerado a partir de uma simples classe ou pela combinação de vários atributos de classes. Já um contexto navegacional genérico pode ser derivado de uma classe, atributos de classe, elos, índices ou sessões. Estes contextos podem ser especificados em modelos navegacionais de tipos.

Algumas vezes, como contextos navegacionais arbitrários baseados em uma estrutura ou seleção, podem ser especificados em um modelo navegacional de instâncias.

O modelo navegacional de tipos suporta a definição de contextos navegacionais distintos de acordo com as diferentes estruturas de acesso utilizadas. As estruturas de acesso utilizadas seguem a nomenclatura definida em RMM [IZA 95].

As classes, atributos e fatias são representados de acordo com o modelo de fatias; acrescentado de relacionamentos e mecanismos de acesso, que são representados através de estruturas de acesso.

Os relacionamentos do modelo de fatias são mapeados para elos ou estruturas de acesso no modelo navegacional de tipos. Os relacionamentos que são mapeados em elos representam relacionamentos estruturais que unem estruturas de acesso a outras estruturas de acesso, ou estruturas de acesso a classes ou fatias. Estes relacionamentos aparecem no modelo navegacional de tipos como uma fatia que modela as âncoras. O ponto de entrada para navegação deve ser especificado pelo projetista, assim como a

maneira como as classes podem ser acessadas. Uma representação de um modelo navegacional pode ser observada através da Figura 3.36.

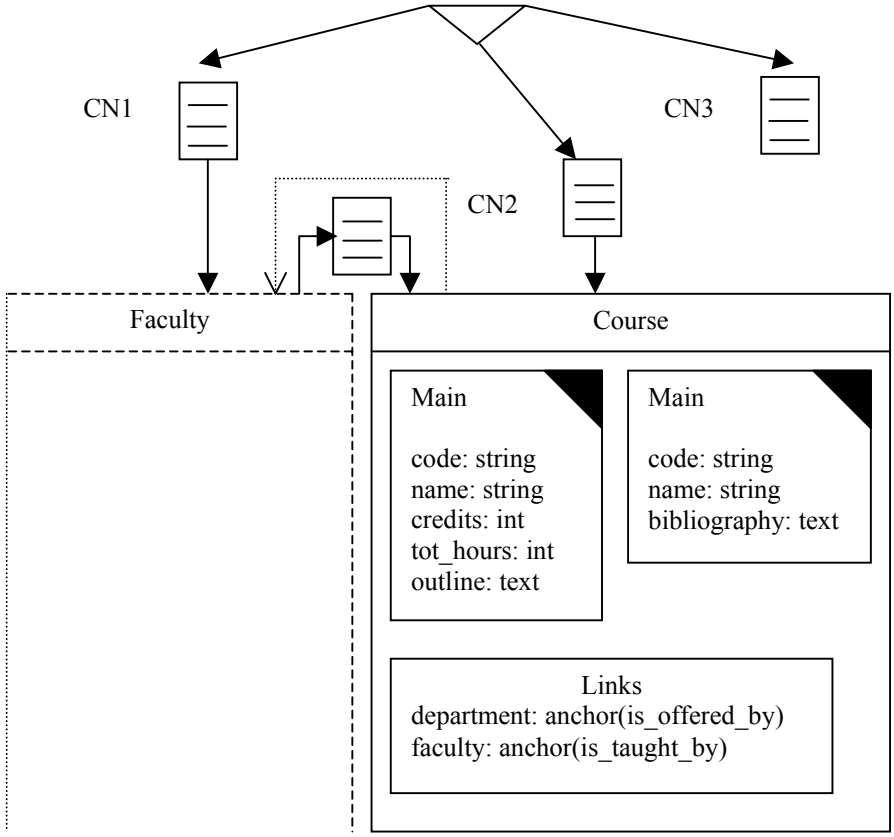


FIGURA 3.36- Modelo navegacional de tipos em uma aplicação acadêmica [CAR 98]

A navegação é iniciada por uma estrutura de agrupamento, especificada por um menu ao topo do diagrama, no formato triangular. Este menu de opções indica caminhos navegacionais distintos, os quais o usuário poderá optar durante uma sessão de navegação. Os caminhos navegacionais determinam diferentes contextos implementados por índices. O índice de acesso CN2 permite uma navegação por todas as instâncias da classe *Course.* Enquanto que os demais índices, CN1 e CN3, permitem o acesso à todas as instâncias de outras classes relacionadas.

Note, através da ilustração, que os relacionamentos estruturais entre duas classes são representados através de setas com linhas cheias, por exemplo o relacionamento estrutural entre *Faculty* e *Course*. O relacionamento citado como exemplo é percorrido através do índice CN2. Relacionamentos de aplicação, aqueles que surgem de visões navegacionais do modelo conceitual, são representados por setas com linhas pontilhadas.

A Tabela 3-2 descreve uma especificação completa para o contexto navegacional CN1. Esta é a forma de documentar cada contexto navegacional identificado, complementando a especificação do diagrama navegacional de tipos.

TABELA 3.2- Especificação de um contexto navegacional de tipos

Nome do Contexto	CN1 Faculty de A até Z	
Tipo do Contexto	Índice	
Atributos do Contexto	-	
Inclui outros contextos, relacionamentos, classes e objetos		
Pontos de entrada	All instances of class Faculty	
Caminho	-	
Comportamento	Controle	Restrição
	Passo a passo e acesso direto	parcial
Importa/Exporta	Importa os links que representam os contextos navegacionais derivados de relacionamentos envolvendo nós originados de contextos navegacionais	
Comentários	-	

Fonte: HMBS/M – An object oriented method for the design and development of hypermedia applications [CAR 98].

O modelo navegacional de instâncias é um modelo HMBS que descreve a organização estrutural e as semânticas de navegação associadas as classes e relacionamentos identificados no modelo navegacional de tipos. Trata-se de um modelo que pode ser adicionado de contextos navegacionais. Um conjunto de regras definem como é realizada a criação do modelo navegacional de instâncias:

- as classes que possuem somente uma fatia de elos e, uma única fatia principal, originam múltiplos objetos que são mapeados em estados;
- as classes que possuem fatias adicionais originam múltiplos objetos que são mapeados em estados compostos. O número de sub-estados depende do número de fatias adicionais;
- superclasses são consideradas classes abstratas, somente suas subclasses originam múltiplos objetos que são mapeados em estados;
- agregações de classes originam múltiplas instâncias de objetos. Cada agregação é mapeada em um superestado que é decomposto em subestados que representam instâncias das classes agregadas;
- índices, indexados de guiados de roteiros, condicionais de guiados de roteiros e grupos são mapeados em estados associados com descrito em [CAR98];
- roteiros guiados podem, ou não, serem mapeados em estados. Caso um roteiro guiado não for diretamente mapeado em um estado, sua ativação ocorre a partir do estado que contém sua âncora de ativação. Os links, que especificam os roteiros guiados por instâncias de uma classe, são representados por transições;
- os relacionamentos estruturais e da aplicação são representados por estruturas de acesso no modelo navegacional de tipos e são mapeados em transições.

Essas regras permitem ao projetista seguir uma linha de projeto a fim de construir o modelo navegacional de instâncias para a aplicação. Após a construção do modelo navegacional de instâncias, o projetista especifica as funções de mapeamento dos elementos estruturais em um modelo navegacional baseado no modelo HMBS. Através do modelo HMBS, os mapeamentos dos estados em páginas de informação e de eventos em âncoras são realizados, assim como o mapeamento de páginas em canais para a apresentação do conteúdo da informação.

O compartilhamento de nós por diferentes contextos navegacionais atua como controle navegacional, pois a escolha de nós destinos depende do contexto que está em andamento. Por exemplo, diferentes roteiros guiados podem produzir diferentes caminhos de navegação através de um certo conjunto de nós. Em HMBS a solução para o controle de desencadeamento de transições está na implementação de *flags* determinadas por variáveis de lógicas condicionais no *statechart*. De acordo com as semânticas do *statechart*, uma transição rotulada *e(v)* é disparada quando o evento *"e"* ocorre e a variável *"v"* é verdadeira. Assim, a navegação é restrita à um contexto navegacional particular, onde somente as transições que satisfaçam a condição são executadas.

2.5.3 Modelagem da interface com o usuário

A interface com o usuário é especificada pela definição das características de visualização da aplicação durante a navegação, independente do ambiente de implementação. A interface é definida através da especificação de canais de apresentação. Os canais de apresentação são dispositivos abstratos que determinam a aparência do hiperdocumento através da definição do aspecto visual dos elementos de interface, de acordo com as definições em [HAD 94]. Quatro tipos de canais são suportados: texto, imagem, áudio e vídeo. O projetista especifica os canais de apresentação de acordo com os tipos de mídia em questão e, também, conforme as características visuais desejadas. Um canal do tipo texto, por exemplo, pode possuir características próprias como tipo, cor e tamanho da fonte utilizada em um certo conteúdo de informação que será apresentado em uma página. As páginas da aplicação são mapeadas em canais apropriados através das funções de mapeamento definidas no modelo HMBS. Os canais podem ser reusados, onde um simples canal pode ser mapeado para múltiplas páginas. Veja um exemplo de uma especificação de um canal do tipo texto na Tabela 3.3.

2.5.4 Implementação e teste

O ambiente de autoria denominado HySCharts [TUR 98] interpreta e implementa aplicações especificadas através do método HMBS/M. HySCharts realiza o mapeamento dos modelos navegacionais e de interface em uma aplicação final. Outra utilidade para o HySCharts é a prototipação de hiperdocumentos que pode ser útil aos projetistas. Uma outra alternativa é a tradução da especificação HMBS/M em elementos de um determinado ambiente de implementação.

TABELA 3.3- Especificação de um canal de apresentação do tipo texto

Nome do canal	Canal texto 1			
Fonte do título	Arial			
Estilo do título	Normal	Bold	Italic	Bold Italic
	X			
Tamanho do título	12			
Posição da título	Centered	Right	Left	
	X			
Cor do título	Automatic			
Fonte do texto	Times New Roman			
Estilo do texto	Normal	Bold	Italic	Bold Italic

	X			
Tamanho do texto	12			
Posição do texto	Centered	Right	Left	
	X			
Cor do texto	Automatic			
Cor do fundo	Light-grey			
Barra de rolagem horizontal	Não			
Barra de rolagem vertical	Sim			

Fonte: HMBS/M [CAR 98].

HySCharts possui um módulo de autoria e um módulo de navegação para a criação e execução de aplicações baseadas no modelo HMBS/M. O módulo de autoria possui uma propriedade adicional que é a verificação da aplicação. Esta propriedade possibilita a verificação de inconsistências no hiperdocumento, como páginas que não foram mapeadas em estados ou âncoras que não foram mapeadas em eventos. Esta verificação é realizada através da implementação de algoritmos de análise de execução de *statecharts*.

Durante a fase de autoria o desenvolvedor cria os modelos navegacionais de instâncias com base na visualização dos estados e eventos constantes no *statechart*, associando-os a páginas e âncoras, respectivamente. Nesta etapa são definidas as variáveis do *statechart*, bem como as propriedades dos objetos estruturais. Outra funcionalidade do módulo de autoria é a possibilidade de definição de canais de apresentação pela associação de páginas de informação.

Na fase de navegação o autor percorre os contextos de navegação criados durante o módulo de autoria. HySCharts implementa um simulador de *statecharts* que monitora as ações de interface com o usuário, tais como: ativação de âncoras e, consequentemente, geração de eventos associados reagindo de acordo com as semânticas operacionais dos *statecharts*. A simulação de um passo de execução leva a uma nova configuração de estados que é reportada ao *browser* interpretador. O *browser* determina o conjunto de páginas que deverão ser apresentadas, de acordo com a nova configuração.

Uma outra abordagem de implementação de modelos HMBS/M é a tradução de modelos navegacionais e de interface em elementos de ambientes de autoria hipermídia, como *Toolbook, Macromedia Director* ou HTML (*Hypertext Markup Language*). Onde as aplicações resultantes podem ser executadas por *browsers* com *Netscape* ou *Internet Explorer*. Este processo requer a definição de um conjunto de regras de conversão de modelos em elementos de uma ferramenta de autoria. Um conjunto de algoritmos de tradução de especificações criadas em *HySCharts* foi implementado em [SHI 98].

2.5.5 HMBS (*Hyperdocument Model Based on Statecharts*)

HMBS é um modelo para a especificação da estrutura e da semântica navegacional de hiperdocumentos que utiliza uma técnica de especificação formal baseada em *Statecharts* de [HAR 87]. Como em modelos implementados nos métodos HDM e OOHDM, HMBS separa a estrutura das informações referentes ao conteúdo do hiperdocumento. Aspectos como concorrência são descritos no modelo HMBS. HMBS não é um método de desenvolvimento de hiperdocumentos, portanto não é destinado à

análise dos requisitos do domínio da aplicação, bem como não é um modelo adequado à captura das semânticas de domínio. A especificação conceitual da aplicação deve ser complementada por um modelo que especifique o esquema conceitual, representando os objetos e os relacionamentos que compõem o domínio da aplicação.

Após a especificação conceitual do problema, pode-se especificar os usos do hiperdocumento e as tarefas particulares a serem realizadas sobre o mesmo, atividades comumente realizadas durante a fase de modelagem navegacional. Desse modo, o HMBS pode ser integrado a uma abordagem de projeto de mais alto nível para o desenvolvimento de hiperdocumentos [TUR 98]. A estrutura semântica do domínio do problema é especificada, utilizando-se para isso o modelo de objetos do Método *Fusion*. Esse modelo é estático, ou seja, não especifica características navegacionais e comportamentais da aplicação.

Statecharts é uma técnica que utiliza um formalismo baseado em uma linguagem gráfica para a especificação comportamental de sistemas ditos como reativos. Este formalismo foi proposto por Harel [HAR 87] e é classificado como uma tecnologia baseada em uma linguagem visual que permite a especificação de aspectos comportamentais de sistemas, possibilitando a criação, modificação, análise e execução de diagramas de estado. Os *statecharts* estendem o formalismo convencional de Máquinas de Estados Finitos, pois além de estados e transições, foi definida uma hierarquia de estados, concorrência e ortogonalidade entre estados, além de um mecanismo de propagação de eventos denominado *broadcast*, possibilitando a comunicação entre os estados.

Um Statechart é composto por um conjunto de estados, transições e eventos primitivos. A partir desses elementos básicos, definem-se os conjuntos de eventos, condições, expressões, ações e rótulos, além dos inter-relacionamentos entre eles. Um estado é um contexto no qual o sistema se encontra durante a sua execução e são representados graficamente por retângulos com cantos arredondados. Os estados são agrupados em outros estados formando níveis hierárquicos, permitindo a decomposição de estados em níveis. Os *Statecharts* permitem decomposição hierárquica de estados. De maneira que os estados possam ser agrupados em outros estados formando níveis, ver Figura 3.37.

Note que o estado *B* é composto por dois subestados B_1 e B_2, *B* é um subestado de *A*. Portanto, *B* é ancestral direto de B_1 e B_2 e *A* é ancestral direto de *B*. Desta forma, *A* é ancestral indireto de B_1 e B_2.

Os estados são classificados em três tipos: atômicos, OR e AND. Os estados atômicos não podem ser decompostos, formam um estado elementar ou básico. Conforme apresenta a FIGURA 3-37, os estados B_1, B_2, *C, D* e *E* são atômicos. Enquanto que os estados A_1, A_2 e *B* são estados do tipo OR. Estados do tipo OR são exclusivos, onde o sistema somente poderá estar em uma de suas configurações de estado e nunca em todas simultaneamente. Já os estado *A* é um estado do tipo AND, pois as configurações de estado A_1 e A_2 são simultâneas, ou seja, os dois estados estão ativos ao mesmo tempo.

A interação com o *statechart* é realizada através de eventos. Os eventos podem ser externos ou internos. Os eventos externos são conseqüência de uma interação do

usuário, através de um dispositivo de entrada. Já os eventos internos se referem aos efeitos de uma transição.

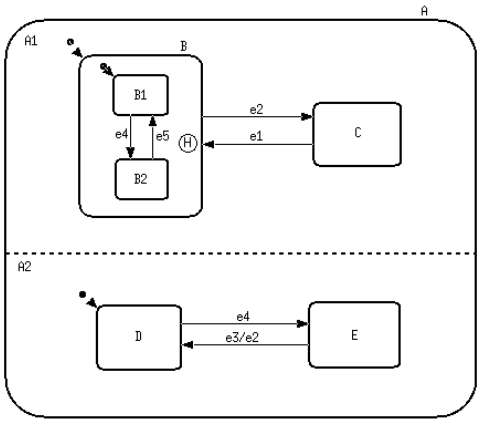


FIGURA 3.37- Um diagrama hierarquia de estados, eventos e transições [TUR 99].

O estado é atingido através de uma transição, sendo esta a única maneira de um determinado estado ser atingido. Um estado está relacionado sequencialmente com outro por meio de uma transição que graficamente é representada por uma seta que une um estado origem a um estado destino. As transições podem ser do tipo (1:1) ou (M:N). Transições do tipo (1:1) são aquelas que possuem somente um estado origem e um estado destino. Enquanto que as transições (M:N) podem Ter vários estados origem e múltiplos estados destino. A Figura 3-38 apresenta um exemplo de transição quando um determinado evento é ativado. Observe que quando o evento e_4 ocorrer, será ativada uma transição do estado *D* para o estado *E*.

Transições *default* são aquelas que não realizam ligações entre estados. Estas transições são executadas automaticamente quando não houver uma outra transição pré-definida. Uma transição *default* é especificada através de um símbolo que marca a transição como *default*. Este símbolo é colocado no canto superior esquerdo do estado. As transições *default* aparecem nos estados *B, B_1* e *D*. Quando o *statechart* for inicializado pelas transições *default*, os estados B_1 e *D* se tornarão ativos, veja FIGURA 3.38.

No modelo HMBS, um hiperdocumento é estruturado em uma hierarquia de estados, definida pelo *statechart*. Esta hierarquia de estados representa a estrutura organizacional e navegacional da aplicação. Cada estado é associado a uma porção de informação do hiperdocumento formando as páginas. Os eventos rotulados nas transições representam as âncoras que irão desencadear as ligações entre as páginas, disponibilizando caminhos de navegação ao usuário da aplicação. A definição do *statechart* é uma simplificação do modelo definido por Harel [HAR 87].

Segundo o formalismo, um hiperdocumento é definido de acordo com a 7-*tupla* H = <*ST, P, m, L, pl, ae, N*>. A Tabela 3-4 define cada elemento de especificação do hiperdocumento.

ST está associado ao *Statechart* referente ao hiperdocumento, sendo definido como uma 11-tupla *ST* = <*S*, ρ, ψ, γ, δ, *V*, *C*, *E*, *A*, *R*, *T*>, onde cada elemento representa, respectivamente: estados, função de hierarquia, função tipo de decomposição, função história, função *default*, conjunto de expressões, conjunto de

condições, conjunto de eventos, conjunto de ações, conjunto de rótulos e conjunto de transições;

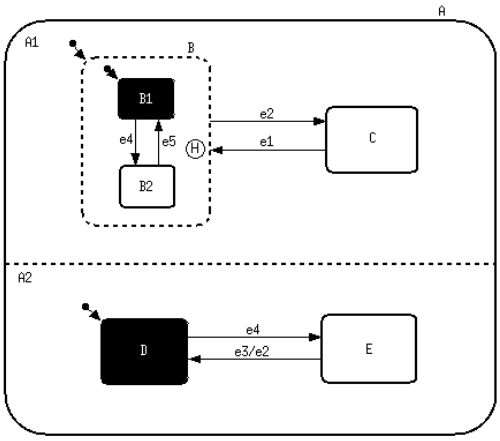


FIGURA 3.38- Transições de estado default [TUR 99]

- *P* forma um conjunto finito de páginas que contém as informações que formam o conteúdo do hiperdocumento. Cada página *p* (nó de informação) $\in$ *P* é definida conceitualmente pela tripla <*c*, *t*, Anc_p>, onde *c* é a porção de informação, ou conteúdo da página. Este conteúdo pode ser composto por mídias estáticas ou dinâmicas. Exemplos de mídias estáticas são textos, imagens e gráficos. Enquanto que mídias dinâmicas podem ser vídeos, áudios ou animações. O elemento *t* representa o título associado à página e Anc_p é uma coleção de âncoras contidas na página. O conjunto de páginas pode incluir uma página nula especial ($P\lambda$), sem qualquer conteúdo, título ou âncora, a qual pode ser associada a estados que não modelam a apresentação da informação;

- m: $S_S \rightarrow P$ denota uma função que mapeia estados de um subconjunto S_S ($S_S \subset S$) em páginas $p \in P$ do hiperdocumento. S_S é o conjunto formado pelos estados compostos do tipo OR e pelos estados atômicos do *Statechart*; estados AND não são mapeados em páginas, sendo utilizados unicamente para especificar concorrência de informações na apresentação;

TABELA 3.4- Elementos de estruturação e organização do Hiperdocumento

Estrutura do Hiperdocumento 7-*tupla*	
ST	Statechart
P	Conjunto de páginas
m	Função de mapeamento de estados
L	Conjunto de canais de apresentação
pl	Função de visualização
ae	Função de mapeamento de eventos
N	Nível de visibilidade

Fonte: Hyscharts: a hyperdocument authoring and browsing environment based on statecharts [TUR 99].

- *L* define o conjunto de canais de apresentação (ou leitores) invocados para interpretar e visualizar as informações contidas nas páginas durante a navegação.

Permitem ao autor especificar a coordenação espacial das informações contidas nas páginas, além de controlar atributos globais da apresentação do hiperdocumento. O uso de canais é inspirado no *Amsterdam Hypermedia Model* [HAD 94] e no Modelo de Contextos Aninhados [CAS 91];

- *pl*: $P \rightarrow L$ define a função de visualização que associa cada página $p \in P$ a um único canal $l \in L$ capaz de interpretá-la;

- *ae*: $Anc_p \rightarrow E$ define a função que associa elementos a_p ($a_p \in Anc_p$) a eventos do *Statechart* que, por sua vez, definem as transições a serem disparadas. Segundo o HMBS, os elementos do conjunto Anc_p da página P só ativam as ligações do hiperdocumento quando estiverem associados a um único evento pela função *ae*. O comportamento do mapeamento *ae* segue a abordagem de programação orientada a eventos, de modo que a ação do leitor de selecionar uma âncora é um evento externo a ser tratado pelo sistema hiperdocumento.

N ($N \geq 0$) define o nível de visibilidade do hiperdocumento. O autor pode utilizar o valor de N para definir a profundidade hierárquica a ser apresentada durante a navegação. Trata-se de um novo mecanismo de navegação, tal que ao invés de ativar as âncoras, o leitor pode navegar pelas páginas associadas a estados acima ou abaixo do nível de hierarquia definido pelo autor, contribuindo para minimizar a sua eventual desorientação.

O HMBS possui na sua definição três tipos de objetos: estruturais, navegacionais e de apresentação. A estrutura organizacional do hiperdocumento definida pelo *Statechart* subjacente permite especificar os objetos estruturais, que são os estados, as transições e os eventos do *Statechart*. Os objetos navegacionais são as páginas, as ligações e as âncoras e formam a estrutura navegacional da aplicação. Os canais definem os objetos relativos à apresentação física do hiperdocumento.

Os objetos navegacionais definidos em HMBS são as páginas, ligações e as âncoras que são implementadas no hiperdocumento.

As páginas representam objetos estruturais do tipo estados, que são modelados no *statechart* da aplicação. O mapeamento dos estados do modelo em páginas da aplicação é realizado pela função *m*. Cada estado é mapeado em somente um objeto navegacional do tipo página. Toda página possui os seguintes atributos: título, conteúdo e âncoras. O título é uma palavra-chave que atua como identificador único para a página. O conteúdo é a porção de informação contida na página, podendo ser do tipo texto, imagem áudio ou vídeo.

O termo janela lógica, utilizado durante a navegação, é uma analogia à uma página da aplicação. Cada página é convertida em uma janela lógica durante a navegação. A utilização de janelas lógicas permite que os atributos de uma mesma página possam ser visualizados de diversas formas, desvinculando o modelo dos detalhes do ambiente de implementação.

Os elos, que formam o conjunto de ligações de ativações de páginas do hiperdocumento, são representados por transições *T* do *statechart*. As transições modelam os relacionamentos entre os estados (páginas) em uma estrutura organizacional do hiperdocumento, modelada através do *statechart*. As ligações relacionam as páginas da aplicação através da definição de âncoras e são a representação física das transições.

O destino de uma ligação pode ser uma página ou um conjunto de páginas, dependendo do tipo do estado destino ativado pela transição subjacente à ligação. Deve ser considerado o fato de que no modelo HMBS não existe a possibilidade do destino de uma ligação ser uma porção de informação dentro da mesma página de origem. Ligações são armazenadas separadamente das páginas que referenciam, ou seja, não são inseridas no conteúdo das páginas, como acontece, por exemplo, com os documentos que usam a linguagem HTML. Ligações múltiplas, ou seja, ligações com múltiplas páginas destino podem ser definidas especificando transições M:N de s*tatecharts* ou por uma ligação unidirecional para um página intermediária que contém ligações para as diferentes páginas destino. O HMBS não permite definir ligações temporais, que são ligações ativadas automaticamente por intervalos de tempo pré-definidos.

Uma âncora é um objeto navegacional que representa a origem de uma ligação (elo). A ativação de uma âncora desencadeia os eventos associados à ela, de acordo com a função de mapeamento de âncoras *ae*. A propagação dos eventos é realizada pelo mecanismo *broadcast*, segundo a semântica de operacional dos *statecharts*.

A semântica de navegação do HMBS define qual página é apresentada, quais âncoras são habilitadas e quais são as transformações navegacionais que ocorrem quando da interação com o leitor. O comportamento dinâmico do hiperdocumento é baseado na semântica operacional de *Statecharts* a partir de alguma configuração de estados.

A apresentação do hiperdocumento compreende um conjunto de páginas interpretadas pelos canais, que são apresentadas em janelas lógicas. As características físicas relacionadas à implementação das janelas lógicas ou à visualização das âncoras dependem do sistema de hiperdocumento (ambiente de apresentação).

Segundo a semântica de navegação, uma página é resultado do mapeamento de um objeto estrutural do tipo estado, constante no *statechart* subjacente, pela função de mapeamento *m*. Esta função realiza o mapeamento através dos atributos título, conteúdo e âncoras de cada página, que são definidos pela tripla <c*, t, Anc*$_p$>. O título (*t*) é uma palavra-chave de identificação de uma página como única. O conteúdo (*c*) é a porção de informação contida na página, que pode ser do tipo texto, imagem, áudio ou vídeo. Pode-se estabelecer uma analogia entre páginas e janelas lógicas, pois durante a navegação cada página é convertida em uma janela lógica e, dependendo do sistema hiperdocumento e do canal associado, os atributos da página podem ser visualizados de diferentes maneiras na janela lógica, podendo apresentar o título e o conteúdo da página, ou somente o conteúdo.

A semântica de navegação define, ainda, um conjunto de transições *T* do s*tatechart*, representando as ligações de ativação do hiperdocumento, que são os elos de ligação (*links*). As transições representam o inter-relacionamento de estados na estrutura organizacional do hiperdocumento, enquanto as ligações relacionam os objetos página durante a navegação e são definidas a partir da criação das âncoras. O destino de uma ligação pode ser uma página ou um conjunto de páginas, dependendo do tipo do estado destino ativado pela transição subjacente à ligação. Entretanto, o HMBS não permite que o destino de uma ligação seja uma porção de informação dentro da mesma página de origem. Ligações são armazenadas separadamente das páginas que referenciam, ou seja, não são inseridas no conteúdo das páginas, como acontece, por exemplo, com os documentos que usam a linguagem HTML. Ligações múltiplas, ou seja, ligações com

múltiplas páginas destino podem ser definidas especificando transições *M:N* de *Statecharts,* ou por uma ligação unidirecional para um página intermediária que contém ligações para as diferentes páginas destino. O HMBS não permite definir ligações temporais, ou seja, ligações ativadas automaticamente (sem intervenção do usuário).

O objeto navegacional âncora representa a origem de uma ligação e a sua ativação faz com que todos os eventos associados a ele (via função *ae*) sejam disparados de acordo com a semântica operacional do *Statechart*. No HMBS, quando o leitor seleciona uma âncora, o sistema realiza as seguintes ações:

- gera o evento correspondente à âncora, este evento é obtido pelo mapeamento *ae*. Após, dispara a ligação correspondente, que é interpretada internamente pela transição do *Statechart* a qual o evento está associado;
- ativa todos os estados que fazem parte do conjunto destino da transição disparada, gerando a próxima configuração de estados (*state configuration - SC*) e desabilitando a configuração anterior;
- por fim, define a nova configuração de contexto (*context configuration - CC*) do hiperdocumento como sendo composta pelas páginas associadas aos estados da nova configuração de estados, além das páginas que obedecem ao escopo estabelecido pelo nível de visibilidade *N* definido pelo autor. Invoca, em seguida, os respectivos canais de apresentação para cada página da configuração de contexto.

Na Figura 3-39 é apresentado um esquema de mapeamentos *m* e *pl* do modelo HMBS em que três objetos navegacionais do tipo página são definidos e associados a estados do *Statechart* subjacente (função *m*). Duas das páginas armazenam informações do tipo texto e são associadas a um mesmo canal (função *pl*), a outra página armazena uma imagem e é associada a um canal que a reconhece.

Os canais são classificados de acordo com o tipo da mídia que manipulam (texto, imagem, áudio ou vídeo, por exemplo) e pelos atributos de apresentação de alto nível (tipo e tamanho do caractere fonte para um canal texto, volume para um canal áudio, entre outros) e de coordenação espacial (largura e altura das janelas lógicas). O termo genérico "janela lógica" é usado para designar o conceito que os sistemas hiperdocumento implementam como repositório das informações na navegação. Por exemplo, no contexto do modelo HMBS, o conteúdo de cada página é apresentado na navegação em uma janela lógica. Uma apresentação do hiperdocumento em um sistema é formada por uma ou mais janelas lógicas.

O exemplo da Figura 3-39 enfatiza a possibilidade em reutilizar canais para interpretar páginas com o mesmo tipo de mídia, permitindo que a apresentação das páginas do hiperdocumento durante a navegação seja padronizada e consistente. Isso não impede, entretanto, a criação de novos canais de apresentação para interpretar páginas de uma mesma mídia. Assim, um único modelo de representação pode ser usado para gerar diferentes versões, ou diferentes apresentações, de um mesmo hiperdocumento.

2.5.6 XHMBS (*Extended Hyperdocument Model Based on Statecharts*)

O modelo XHMBS é uma extensão do modelo original HMBS. Trata-se de um modelo para a especificação da estrutura organizacional, navegacional e de sincronização de aplicações baseadas em hiperdocumentos. XHMBS utiliza as mesmas semânticas do modelo HMBS, acrescidas de características para a sincronização de sequências de tempo durante a apresentação da informação, requisitos típicos de aplicações que utilizam recursos multimídia.

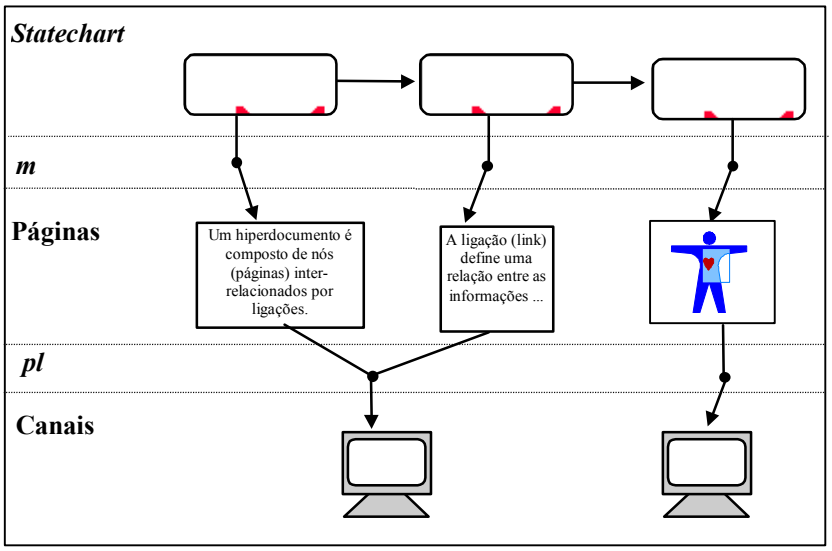


FIGURA 3.39- Mapeamentos m e pl Combinados [TUR 99]

Um novo formalismo foi implementado no modelo denominado *Hypercharts*. Estes são uma extensão dos *statecharts* que acrescentam características para a especificação de sincronização de mídias. As principais características adicionadas ao modelo XHMBS são baseadas nos modelos de redes de *Petri*, utilizados na especificação de aplicações multimídia. Foram acrescentadas facilidades para a definição da estrutura de nós e elos da aplicação, bem como a descrição do comportamento temporal dos dados dinâmicos contidos em nós de informação.

Novas notações foram implementadas no modelo para permitir a especificação de comportamento temporal e sincronização multimídia. Uma história de temporizações, transições temporizadas e um mecanismo de sincronização são as três notações incluídas no modelo. Algumas notações foram adicionalmente incluídas no modelo para a criação de estados parametrizados e abstrações de transições.

A estrutura do *Hyperchart* segue a mesma estrutura formal definida para os *statecharts,* adicionada dos mecanismos para especificação de comportamento temporal e sincronização. A estrutura é composta por uma 14-tupla, de acordo com a seguinte notação: HYP = <S, ρ, ψ, δ, γ, **V, C, E, T, Ac,** τ, LSC, T_hist, T_s>.

O modelo segue as mesmas características do modelo original HMBS, sendo que o conjunto de ações (*A)* em HMBS foi substituído pelo conjunto de ações (***Ac***) em XHMBS. As demais inclusões no modelo são as seguintes:

τ: **T** $\rightarrow$ {*true, false*} que implementa uma função para história de tempo. A função retorna *true* ou *false.* Se τ (t) retorna *true,* indica que uma transição *t* possui um símbolo de história de tempo associado, permitindo a implementação de recursividade em uma transição. A representação desta notação no modelo é realizada através de um ícone na forma de um relógio, colocado próximo à transição *t*;

LSC: **S** $\rightarrow$**N** é uma função que implementa um contador de passos locais de um estado. A função LSC registra todos os passos de execução de um determinado estado *s.* O contador é incrementado a cada passo de execução do *statechart* pelo relógio global do sistema;

T_hist: **S** $\rightarrow$ **N** é um registrador que mantém o valor do contador LSC no momento em que o estado se encontra, caso for interrompido prematuramente;

T_s mecanismo de sincronização de transições, onde cada tipo de sincronização está no conjunto de múltiplas transições sincronizadas (M:N). Cada transição possui um rótulo que controla o tempo de sincronização, que é indicado em cada evento gerador da transição.

Para retomar uma determinada atividade (apresentação multimídia) que foi interrompida durante sua execução, foi implementado um mecanismo que mantém um controle sobre os passos de execução de mídias. O mecanismo permite que a apresentação de uma mídia seja retomada a partir do ponto de interrupção, geralmente ocasionado por interação do usuário.

O mecanismo de temporização é representado por um ícone no formato de um relógio que é colocado junto à transição representada no modelo do *hyperchart*. XHMBS permite a especificação de transições que são disparadas por sincronizações de tempo. A sincronização é realizada de acordo com os tempos de execução dos estados fonte da transição. As transições temporizadas permitem a especificação de comportamento temporal das atividades de apresentação multimídia associadas aos estados do hyperchart. Esta característica é poderosa para a especificação do comportamento funcional de uma aplicação dependente de tempo e não de interações do usuário. Permitem, desta forma, a especificação de requisitos multimídia baseados em *delay* e *jitter*.

As transições são associadas à eventos temporizados, onde cada evento possui a forma "[α, η, β]". A notação α corresponde ao limite mínimo de tempo e β corresponde ao limite máximo de tempo da apresentação da mídia, enquanto que η representa o tempo ideal da apresentação. Diferente dos eventos de um *statechart* convencional, os eventos de um *hyperchart* não podem ser associados com outros eventos ou condições rotuladas em uma transição. Por outro lado, os eventos de um *hyperchart* possuem uma ação associada que indica os procedimentos necessários quando da ativação da transição. A notação final para um rótulo de transição segue a forma "[α, η, β]/ `a`", onde `a` representa a ação.

A apresentação de uma sequência de imagens associada a uma sequência de áudios pode ser um exemplo típico de transições temporizadas. Digamos que um determinado texto de introdução seja composto por dois quadros que são apresentados em sequência, onde cada quadro dura 5 segundos. Estes quadros são acompanhados por quatro faixas de áudio.

A sincronização de múltiplos tipos de dados são requisitos típicos de aplicações que utilizam os recursos da hipermídia, tal como hiperdocumentos. Os mecanismos de sincronização propostos pelo formalismo *hypercharts* especificam transições do tipo M:N. Este tipo de transições são agrupadas em cinco componentes: estados fontes, arcos fontes, estados destinos, rótulos para cada arco e um tipo de transição. O tipo de transição indica um controlador particular de transições, como por exemplo: para restringir transições pela associação de eventos temporizados.

A Figura 3-40 ilustra um exemplo de uma sincronização do tipo M:N, onde M = 3 e N = 1. Segundo o exemplo, o modelo deverá especificar a sincronização de três elementos de mídia que convergem em um estado Y. Os três elementos de mídia são um fluxo de texto (X_1), uma sequência de áudio (X_2) e uma sequência de vídeo (X_3) que são estados ortogonais e compõem o estado X. Um critério de sincronização deverá ser estabelecido para que o término destas mídias ocorra antes da apresentação da atividade especificada pelo estado Y. Este critério de sincronização depende do controlador de transições adotado. Ou seja, o controlador deverá gerenciar as atividades de cada estado (X_1, X_2, X_3) a fim de garantir o término de suas atividades em conjunto.

O controlador de transições suportado por *Hypercharts* é o mesmo adotado nos formalismos TSPN e $TSPN_{UI,}$ que é composto por cinco tipos básicos: *strong-or, weak-and, master, or* e *and.* E ainda, quatro tipos compostos derivados dos anteriores: *or-master, strong-master, weak-master* e *and-master*. Uma apresentação sincronizada é composta por um conjunto de componentes concorrentes, especificados em um *hyperchart* que é parte de uma transição sincronizada M:N.

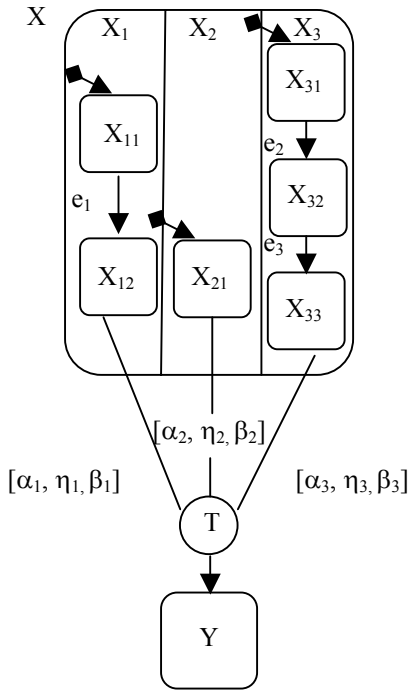


FIGURA 3.40- Uma notação para a transição sincronizada [TUR 97]

Em uma apresentação sincronizada deve aparecer no mínimo dois fluxos de dados sincronizados entre dois pontos (ponto de início e ponto de fim de sincronismo). Cada um destes fluxos de dados é representado por um componente OR de um *hyperchart*.

3 Análise comparativa entre as abordagens de projeto

Nesta seção será realizada uma comparação entre as abordagens de projeto de hiperdocumentos apresentadas neste trabalho, onde serão analisadas as características comuns ao processo de especificação de aplicações, tais como: estruturação da informação, estruturas de navegação, projeto de interface e implementação. Tais características são as mais discutidas nos diversos trabalhos de pesquisa encontrados na literatura como [IZA 95][ROS 96][TUR 97], constituindo-se no principal foco de preocupação durante o processo de desenvolvimento de hiperdocumentos.

3.1 Organização da informação

A identificação do domínio e organização da informação é a primeira etapa do processo de projeto de aplicações de software, portanto é de suma importância durante a especificação de um hiperdocumento. Em todas as abordagens apresentadas neste trabalho, verificou-se que a etapa de definição do domínio da aplicação é bastante evidenciada, pois a partir desta etapa é que será derivada as estruturas de navegação da aplicação final.

3.1.1 Modelo de domínio

A especificação de domínio representa um modelo de dados que especifica as informações conceituais da aplicação, bem como os relacionamentos entre estas informações conceituais.

O modelo para a especificação de domínio proposto por HDM é constituído por entidades e elos. As entidades formam os tipos de informações conceituais de domínio da aplicação e os elos representam relacionamentos entre elas. As entidades são formadas por componentes estruturados hierarquicamente, formando porções de informação interrelacionadas. Cada componente do modelo conceitual pode ser visto por perspectivas diferentes através da implementação de unidades. Os elos estruturais formam os relacionamentos entre os componentes de uma mesma entidade, ou entre as diferentes perspectivas de uma unidade. Já os elos aplicativos especificam as semânticas de relacionamentos de domínio da aplicação. O modelo conceitual do hiperdocumento é representado por um diagrama denominado modelo da aplicação. A partir do diagrama de modelo da aplicação o projetista poderá criar as principais estruturas de acesso às informações do hiperdocumento.

Em HDM há poucas primitivas de modelagem, desta forma o modelo gerado é bastante conciso, onde muitas decisões de projeto não são documentadas durante a modelagem do hiperdocumento. Não existe uma separação entre o projeto conceitual e o navegacional, portanto um mesmo esquema conceitual não pode ser utilizado para perfis de usuários distintos.

OOHDM é um descendente direto do modelo HDM, portanto mantém as principais características do modelo original, acrescentadas de novas primitivas de modelagem. O modelo conceitual é formado por classes, relacionamentos e subsistemas. A especificação das classes segue o mesmo perfil de modelos orientados a objetos, sendo que seus atributos podem ter mais de um tipo. Esta característica foi

acrescentada em OOHDM para representar múltiplas perspectivas de uma mesma entidade real.

O modelo proposto em OOHDM enriquece HDM no sentido de que incorpora primitivas orientadas a objeto para a descrição não somente de agregações (como os componentes de HDM), mas também as noções de especialização/generalização com herança. Em OOHDM existe uma separação clara entre a modelagem conceitual e navegacional, permitindo que uma mesma especificação conceitual seja utilizada para diferentes perfis de usuário. Complementando a documentação durante a modelagem conceitual foram implementados cartões que auxiliam na tomada de decisões, as quais irão afetar o projeto da aplicação tanto para frente como para trás.

Assim como em OOHDM e HMBS/M, RMM possui etapas de projeto bem definidas. O domínio da aplicação é construído durante o projeto do modelo entidade relacionamento, onde cada entidade é descrita. O conceito de fatia é definido para dividir uma entidade em porções menores de informação para determinar como a informação em cada entidade será apresentada aos usuários. Cada fatia pode ser vista como uma janela e conter um ou mais atributos de entidades. As fatias são modeladas e representadas através de um diagrama de fatias.

RMM é semelhante a outras abordagens de projeto de hiperdocumentos por dividir as etapas de especificação em atividades distintas, onde cada uma implementa um conjunto de primitivas bem definidas.

O modelo de dados apresentado em RMM é baseado no modelo entidade relacionamento o que possibilita a implementação de poucos mecanismos de abstração. Diferente de outras abordagens de projeto como OOHDM e HMBS/M que implementam um modelo de dados orientado a objetos que é mais enriquecido.

A técnica introduzida no modelo de especificação HMT propõe etapas para o tratamento do processo de especificação de hiperdocumentos. A especificação envolve a determinação do domínio da informação, bem como o tratamento das conexões entre as informações e interação do usuário.

Para a modelagem de domínio da aplicação HMT utiliza um modelo baseado nos conceitos e notações do modelo de objetos definido em OMT. Portanto, as estruturas de domínio são baseadas em objetos que são representados através de entidades e relacionamentos. Semelhante a HMBS/M e OOHDM, mais uma vez modelos para especificação de objetos foram utilizados no formalismo proposto por HMT.

HMBS/M é um método de projeto de hiperdocumento que divide o processo de projeto em etapas como OOHDM e RMM. No método HMBS/M são descritas as classes e relacionamentos de domínio da aplicação. Os modelos gerados são de objetos e de fatias. O modelo de objetos reflete informações sobre classes, atributos e relacionamentos, sendo que os atributos podem possuir diferentes perspectivas, assim como em HDM e OOHDM. Semelhante ao método RMM, as informações podem ser divididas em fatias e estas são representadas através do modelo de fatias.

3.1.2 Separação entre estrutura e conteúdo

O modelo do projeto deve ser independente da implementação, pois assim não será condicionado a um único ambiente de implementação [ROS 96][TUR 97]. Embora o modelo possa considerar algumas configurações de implementação, deve-se abstrair detalhes. A separação entre a estrutura organizacional do hiperdocumento de seu conteúdo é evidenciada nas abordagens descritas neste trabalho.

Em HDM não foi definida um semântica de mapeamento entre o modelos especificados em objetos de implementação. Os modelos HDM descrevem mais a estrutura organizacional da aplicação, bem como seus relacionamentos.

No método OOHDM, a definição do conteúdo do hiperdocumento é realizada durante a fase de projeto de interface abstrata, que é posterior as fases de modelagem conceitual e projeto de navegação que garantem a estruturação organizacional e navegacional da aplicação. Durante esta fase é especificada a aparência interfacial de cada objeto navegacional que será percebido pelo usuário.

Já no método RMM a separação entre estrutura e conteúdo é definida durante a fase de projeto do protocolo de conversão, onde é utilizado um conjunto de regras de conversão para transformar cada elemento do diagrama RMDM em um objeto na plataforma destino.

Em HMT a definição do conteúdo da aplicação é realizada a partir do modelo de *HyperObjects*. Este modelo é um refinamento do modelo de objetos, que descreve o domínio da informação, onde são acrescentadas as mídias que serão utilizadas na aplicação. Verifica-se neste modelo que existe a preocupação na separação entre estrutura e conteúdo, evidenciada através do modelo de *HyperObjects*.

Na abordagem HMBS/M a separação entre a estrutura e o conteúdo do hiperdocumento está bem especificada através dos mapeamentos definidos no modelo. A definição dos mapeamentos entre estados e páginas (*m*) e entre âncoras e eventos (*ae*) garantem que toda a estrutura esteja descrita em um nível distinto daquele no qual se encontram as porções de conteúdo da informação, como proposto pelo modelo *Dexter* [HAL 94].

3.1.3 Abstração e granularidade

Nesta seção será realizada uma comparação entre a possibilidade de decomposição do modelo conceitual em níveis de abstração e granularidade. Na medida em que os níveis de abstração aumentam, pode-se explorar diferentes níveis de granularidade. Normalmente, os detalhes da especificação do hiperdocumento devem aparecer no nível mais baixo de abstração, no qual o nível de granularidade é mais fino.

A abstração e granularidade em HDM é realizada através do modelo hierárquico de componentes. Como um componente é parte de uma entidade, o modelo pode subdividir uma entidade em porções menores para melhor representar a informação. Estas porções menores são os componentes que são organizados conforme uma hierarquia estrutural que acrescenta maiores detalhes de acordo com o nível de decomposição.

OOHDM acrescenta o conceito de subsistemas para permitir o mecanismo de abstração durante o processo de modelagem de domínio. Um subsistema é uma abstração de um sistema conceitual completo, podendo relacionar-se com outros subsistemas. Os subsistemas possuem um ou mais pontos de entrada que são compostos por classes que podem ser acessadas por outros subsistemas. A granularidade é observada de acordo com os níveis de detalhe dos subsistemas.

O mecanismo de abstração proposto por RMM são os diagramas entidade-relacionamento e de fatias. O diagrama de fatias modela os níveis de informação de uma entidade. Uma entidade é vista em RMM como porções de informações, as quais contém agrupamentos de atributos.

No modelo HMT a abstração é representada pelos níveis de decomposição de objetos do modelo conceitual. A decomposição de objetos é observada no modelo de objetos, primeira etapa do processo.

Em HMBS/M a possibilidade de decomposição de estados permite ao autor abordar o processo de especificação de hiperdocumentos em vários níveis de abstração e granularidade. Este aspecto permite uma análise em níveis de complexidade, facilitando a compreensão do domínio e especificação do modelo.

3.2 Estruturas de navegação

Através das estruturas de navegação é que o usuário irá interagir com o hiperdocumento, realizando a leitura das informações. As estruturas de navegação em hiperdocumentos são definidas por nós, elos e estruturas de acesso, que permitem a definição da maneira como o usuário irá percorrer as informações da aplicação.

3.2.1 Nós, elos, estruturas de acesso e contexto navegacionais

A seguir, veremos como as abordagens de especificação estudadas modelam os nós, elos e estruturas de acesso da aplicação, realizando uma comparação entre os modelos descritos neste trabalho.

Os elementos de informação de HDM são constituídos por unidades. Uma unidade HDM corresponde a uma visão de perspectiva sobre uma entidade do modelo conceitual, representando um nó de informação. Os elos em HDM são definidos em conceituais e navegacionais. Os elos navegacionais são derivados das ligações de domínio e representam os caminhos de navegação da aplicação. As estruturas de acesso em HDM são definidas em unidades que representam os únicos objetos navegacionais do padrão HDM, um único nó de informação pode ser ativado por vez.

Os nós, elos e estruturas de acesso são denominados objetos navegacionais em OOHDM e são representados no modelo através de classes navegacionais. As classes navegacionais representam uma visão do esquema conceitual de domínio da aplicação. Os nós são objetos de visualização de informações de classes conceituais. Estas visões são formadas por uma linguagem de consulta que combina atributos de uma ou um conjunto de classes do esquema conceitual. Os elos são materializações dos relacionamentos identificados no modelo conceitual e são representados por classes de elos. Os elos podem ser demonstrados durante a navegação desde que sejam definidos

objetos de interface para eles. A principal estrutura navegacional em OOHDM é o contexto navegacional, composto por um conjunto de nós elos e classes de contexto, ou outros contextos aninhados. O contexto navegacional complementa o conceito de classe navegacional delimitando o espaço navegacional do usuário e indicando quais objetos navegacionais serão disponibilizados.

O diagrama entidade-relacionamento é o modelo que representa a estrutura lógica de domínio da informação de uma determinada aplicação em RMM. Deste diagrama serão derivados os nós, elos e estruturas de acesso da aplicação. Os nós serão compostos a partir de entidades do diagrama conceitual. Já os elos serão derivados dos relacionamentos e servirão como base para a identificação de estruturas de navegação. Muitos relacionamentos identificados no modelo entidade-relacionamento irão se transformar em estruturas de acesso, tais como: índices, roteiros guiados e agrupamentos na aplicação final.

Em HMT, o modelo de objetos especifica as principais entidades e relacionamentos de domínio da aplicação. As entidades modeladas no domínio formarão os nós de informação, bem como seus relacionamentos formarão os elos de ligação ou estruturas de acesso da aplicação. Durante a modelagem navegacional é que o autor irá decidir como serão os caminhos de acesso aos objetos do modelo. Nesta fase os relacionamentos entre os objetos de domínio é que definirão a estratégia de projeto dos caminhos de acesso. Estes relacionamentos poderão ser transformados em um simples elo ou em uma estrutura de acesso como um índice, por exemplo. Contextos de navegação são incluídos em HMT para melhorar a especificação das semânticas dos elos de ligação e como base para o modelo de interface. Segundo o modelo, um contexto contém objetos de contexto padrão onde a navegação por estes objetos é sensitiva ao contexto. Sendo assim, a navegação precisa respeita os limites do contexto padrão.

Segundo o método HMBS/M, a modelagem navegacional é a etapa de projeto de hiperdocumentos. Nesta etapa, os contextos navegacionais e as estruturas de acesso são definidas tendo como base o modelo conceitual da aplicação. O modelo navegacional é uma visão do domínio da aplicação. Neste modelo são identificadas as estruturas de acesso da aplicação. O modelo navegacional de instâncias permite o mapeamento de classes conceituais em objetos navegacionais que são os estados. Um estado representa um nó de informação. Os relacionamentos definidos no modelo de fatias são mapeados em elos ou estruturas de acesso, onde o projetista precisa definir o ponto de entrada para uma navegação de usuário, bem como as possíveis restrições de navegação.

3.2.2 Modelagem de transformações navegacionais

Nesta seção veremos como as transformações navegacionais ocorrem durante a navegação do usuário. A especificação de transformações tem como objetivo especificar as alterações no espaço navegacional do usuário que ocorrem quando da transição entre os nós de informação, representando assim os padrões dinâmicos da navegação de acordo com os objetos navegacionais acessíveis em determinado momento.

Em OOHDM é utilizado um diagrama de navegação para a especificação do comportamento dinâmico da navegação. O diagrama de navegação é baseado no formalismo *statecharts* [HAR 87]. Um diagrama de navegação é uma adaptação de um *statechart* para o campo da hipermídia, mas com o mesmo poder computacional do

formalismo original. O diagrama de navegação proposto por OOHDM modela estados e transições modelando as mudanças no espaço navegacional.

No método HMBS/M a semântica de navegação é que define qual a página que será apresentada e quais âncoras serão habilitadas. Desta forma, as transformações navegacionais são especificadas para modelar o comportamento dinâmico da aplicação quando o usuário iniciar uma navegação. De acordo com a semântica de navegação HMBS/M, uma determinada página é resultado do mapeamento de um estado constante no *statechart*.

3.3 Projeto de interface

O modelo de interface do hiperdocumento permite a especificação de como a informação será apresentada ao usuário, através da definição dos *layouts* das telas e aparência dos objetos navegacionais. O projeto de interface tem por objetivo modelar a aparência visual dos objetos de interface, bem como dividir o material que será apresentado em contextos apropriados de acordo com o perfil do usuário.

Em OOHDM são utilizados os modelos de diagrama de configuração e ADV-Charts. Os diagramas de configuração modelam a aparência dos objetos navegacionais, bem como seus relacionamentos. A troca de mensagens entre os objetos navegacionais, assim como a reação aos eventos do usuário e internos da aplicação, devem ser modelados através de diagramas de configuração. Já os ADV-Charts modelam o comportamento dinâmico dos objetos de interface, demonstrando como ocorrerão as transformações no espaço navegacional.

No modelo HMT, a interface é elaborada tendo como base o modelo de HyperObject e navegacional. A aparência de cada objeto navegacional, bem como seu comportamento são modelados durante a especificação de interface.

HMBS/M especifica um modelo de interface através de canais de apresentação. Os canais são dispositivos abstratos que determinam as características visuais dos elementos de interface. Quatro tipos de canais são definidos: texto, imagem, áudio e vídeo. Os tipos de canais são determinados de acordo como o tipo de mídia que será apresentada e, também, de acordo com as características visuais desejadas. As páginas de apresentação do hiperdocumento são mapeadas em canais por funções de mapeamento definidas nos modelos HMBS e XHMBS.

Na maioria das abordagens de especificação de hiperdocumentos estudas, verifica-se a preocupação na separação entre o projeto de interface abstrata e o ambiente de implementação. Assim, um projeto de interface pode ser implementado em qualquer ambiente, sem o compromisso de estar vinculada a um determinado ambiente.

4 Ferramenta para especificação de hiperdocumentos

A ferramenta visual para especificação de hiperdocumentos apresentada neste trabalho utiliza as primitivas do método OOHDM para a modelagem da aplicação, desde a definição conceitual, passando pelas etapas de navegação e interface até a obtenção do produto final. Para a implementação do modelo de apresentação final são geradas bases de dados de onde serão extraídas as informações. A partir dessas bases de dados serão gerados documentos em XML [W3C 98a], baseados nos modelos de navegação e interface especificados. Para a formatação do hiperdocumento serão unidas folhas de estilo em XSL [W3C 98b], que irão flexibilizar as formas de apresentação das informações, permitindo-se assim, a apresentação do conteúdo em um *web-browser* compatível.

Nas próximas seções a arquitetura do software será apresentada e, em seguida, será descrito o ambiente gráfico de especificação e implementação de hiperdocumentos proposto pelo software.

4.1 Arquitetura do software

A arquitetura da ferramenta foi dividida em duas camadas: uma camada de aplicação onde estão os modelos de interface com o usuário e, uma camada de armazenamento, onde os modelos construídos através da interface visual são mapeados e armazenados em bases de dados para, posteriormente, serem instanciados. A Figura 4-1 ilustra a arquitetura do software.

4.1.1 Camada de aplicação

Na camada de aplicação está o editor gráfico de modelos. Trata-se do ambiente gráfico onde o usuário irá especificar as características abstratas do hiperdocumento, através dos modelos: conceitual, navegacional e de interface. Este ambiente, disponibiliza objetos visuais para a elaboração do projeto de acordo com as primitivas envolvidas em cada etapa descrita no método OOHDM.

As Funções de mapeamento têm por finalidade, a tradução dos modelos gráficos, especificados pelo projetista, em estruturas que serão armazenadas nas bases de dados de modelos conceituais, modelos navegacionais e de interface. Estas estruturas definem as características dos elementos envolvidos em cada diagrama criado na aplicação. Informações como posicionamento, características e relacionamento entre os elementos gráficos são mantidas, para que o projetista possa realizar a manutenção das especificações criadas.

O Gerador de código é um módulo que tem por finalidade capturar as instâncias dos modelos conceituais e de navegação para gerar o hiperdocumento final. Duas tarefas são realizadas pelo módulo gerador de código. Uma tarefa é a criação das bases de dados que guardarão os modelos abstratos da aplicação. E a outra, trata-se da implementação das bases de dados referentes ao conteúdo de informações dos hiperdocumentos gerados pela aplicação.

Os modelos de apresentação serão definidos através de *scripts* em XML, obtidos através de visões navegacionais da aplicação, associados a folhas de estilo XSLT [W3C 99a], que transformarão os arquivos XML em HTML [W3C 99b]. Assim, o hiperdocumento poderá ser visualizado através de um *browser* compatível com XML.

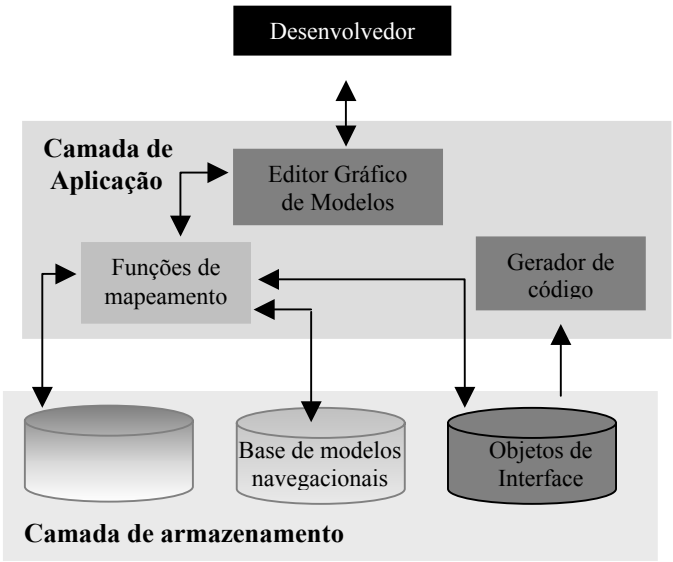


FIGURA 4.1- Arquitetura da ferramenta.

4.1.2 Camada de armazenamento

Na Camada de armazenamento estão as bases de dados onde são mantidas as descrições dos modelos gráficos abstratos, através de registros. A base de modelos conceituais armazena o modelo de domínio da aplicação, representado através de um diagrama de classes e seus relacionamentos. Para cada classe do modelo será gerado um registro em uma tabela de classes conceituais. Os atributos de cada classe conceitual são mantidos em uma tabela de atributos e, suas perspectivas em uma outra tabela de perspectivas. Assim como os atributos, os métodos e relacionamentos também são mantidos em tabelas.

A base de modelos navegacionais, onde é armazenado o modelo de navegação da aplicação, representado através de nós, elos e estruturas de acesso, contextos e classes em contexto. Cada nó representado no diagrama navegacional é armazenado em registros de uma tabela de nós. Assim como seus atributos são armazenados em tabelas separadas. Sendo que cada atributo possui somente um único tipo, uma tabela de perspectivas é desnecessária nesta etapa. Os métodos e relacionamentos também são armazenados em suas respectivas tabelas.

Já na base de objetos de interface estão armazenados os modelos de interface abstrata da aplicação, que definem a apresentação do hiperdocumento. Por fim, estão os documentos que serão gerados na forma de modelos de apresentação.

4.2 O ambiente gráfico de especificação

O ambiente gráfico de especificação é constituído de módulos de interface que permitem a especificação das características do hiperdocumento que está sendo

modelado, em um nível abstrato. Cada módulo de interface representa uma etapa do processo de especificação, segundo a notação do método OOHDM [VIL 99].

Um determinado módulo de interface é selecionado através de uma barra de seleção de modelos da ferramenta, localizada na base inferior da interface principal do software, veja a Figura 4.2.

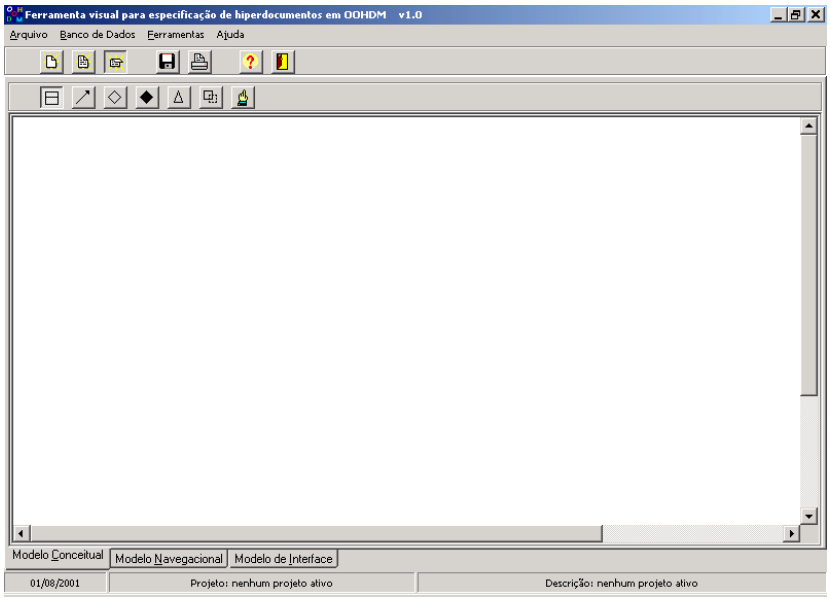


FIGURA 4.2- Interface principal do software.

A Figura 4-3 mostra uma tela de advertência ao usuário. Ela informa ao projetista que um modelo conceitual deve ser definido, anteriormente, para que visões navegacionais possam ser originadas. Outras caixas de advertências são implementadas, auxiliando o usuário no processo de desenvolvimento evolutivo do método OOHDM. Esta característica se torna importante, principalmente, quando usuários inexperientes estão projetando uma determinada aplicação.

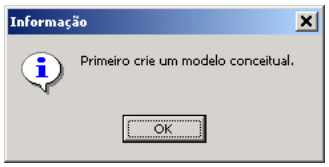


FIGURA 4.3- Mensagem de advertência.

4.2.1 Módulo de especificação do esquema conceitual

O primeiro módulo permite a definição das características conceituais da aplicação. Um modelo conceitual é composto por entidades, atributos, perspectivas de atributos e relacionamentos. Estas primitivas de modelagem são utilizadas através de componentes visuais da ferramenta.

Na barra de ferramentas de modelagem, pode-se observar os botões que representam as primitivas de modelagem do método. Ferramentas como classes, relacionamentos, composições, agregações e especializações foram implementadas. Uma ferramenta para deslocamento de classes na área de desenho e um botão para

edição de propriedades foram inseridos na barra de ferramentas. A primeira, para que o projetista possa deslocar uma determinada classe desenhada na área de especificação, acomodando melhor o diagrama conceitual. E, a segunda, para permitir a edição de propriedades de uma determinada classe modelada, tais como: atributos, métodos e o conjunto de relacionamentos.

A Figura 4-4 apresenta a interface de modelagem conceitual da aplicação, através da modelagem de uma aplicação exemplo. Neste exemplo, observe as classes "Autor", "Livro", "Editora" e "Gêneros" modeladas. Assim como, os seus relacionamentos com as classes do modelo. A classe "Livro" possui um auto-relacionamento, denominado "Referencia". Note que a classe "Autor" possui duas perspectivas para o atributo "E-mail", são elas: *String* e *Image*. A perspectiva *String* é a *default*, a qual poderá ser nomeada, ou não. Quando esta classe for mapeada para o modelo de navegação, cada perspectiva deverá ser aplicada a um atributo distinto em um nó navegacional. Na ilustração, pode-se observar que as propriedades de uma determinada classe, podem ser acessadas por um submenu de opções. Este submenu é selecionado através da seleção do botão de edição de propriedades das classes, a partir da barra de ferramentas de modelagem. Após selecionar esta opção, o projetista, pressionando o botão direito do *mouse* sobre uma classe do modelo, poderá acessar as opções do menu. Existem opções para editar os atributos, métodos e relacionamentos da classe. Bem como, opções como a criação de um auto-relacionamento, deslocamento da classe no modelo e exclusão da mesma. As caixas de diálogos para edição dos atributos, métodos e relacionamentos de uma classe são ilustradas na Figura 4.5.

Os atributos de uma classe são adicionados ou removidos, através da manipulação da caixa de diálogo de edição de atributos. Um atributo é especificado através de sua descrição, conjunto de perspectivas e tipos de dados de cada perspectiva. A definição do atributo *default* é realizada através de um *checkbox*. Várias perspectivas podem ser adicionadas, ou removidas, antes de um atributo ser adicionado a lista de atributos de uma classe, ver Figura 4-5 (a).

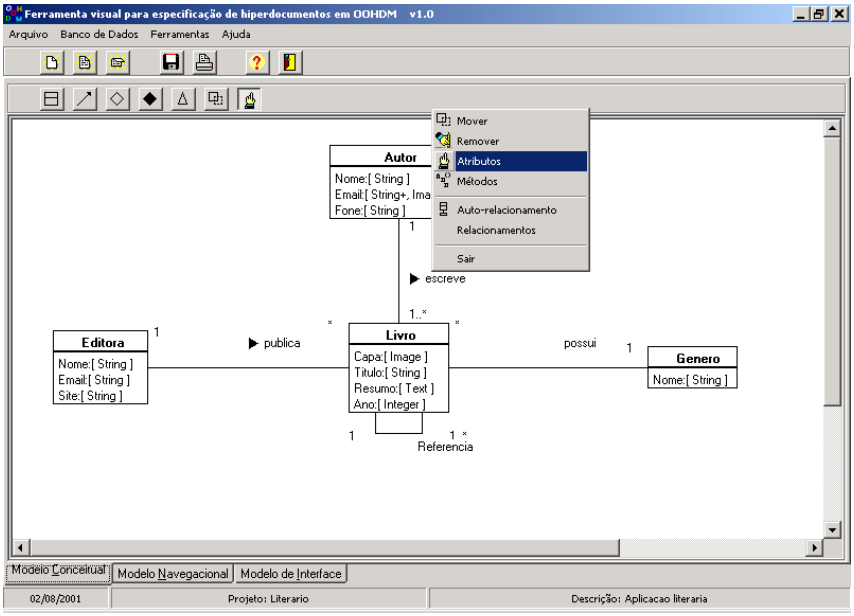
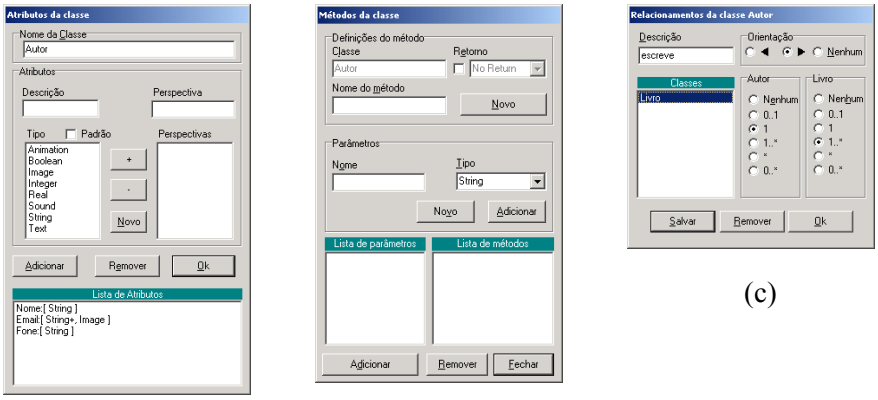


FIGURA 4.4- Interface de modelagem conceitual.

Os métodos da classe são editados através da caixa de diálogos apropriada, ilustrada na Figura 4-5 (b). Nesta interface o usuário poderá definir a descrição do método, parâmetros e retornos.

Após o projetista criar um relacionamento entre as classes do modelo, selecionando a ferramenta de relacionamento e, pressionando o mouse sobre a classe origem, deslocando até a classe alvo. Pode-se, então, editar as propriedades do relacionamento. A caixa de diálogo de edição de relacionamentos permite a definição das propriedades dos relacionamentos, tais como: descrição, cardinalidades e sentido do relacionamento, Figura 4-5 (c).



(a) (b) (c)

FIGURA 4.5- Interfaces de edição de atributos (a), métodos (b) e relacionamentos (c).

4.2.2 Módulo de especificação do esquema navegacional

Elaborado o diagrama conceitual, parte-se para a modelagem navegacional. Um modelo de navegação é expresso em termos de nós, elos, contextos navegacionais e classes em contexto. Os nós representam os atributos de uma classe conceitual, ou a combinação de atributos de classes do modelo conceitual. Enquanto que os elos representam os relacionamentos entre essas classes.

No modelo navegacional o projetista elabora visões navegacionais, a partir de classes e relacionamentos encontrados no modelo conceitual. Definindo assim, como o usuário final irá manipular a estrutura do hiperdocumento. Um mesmo modelo conceitual abstrato poderá derivar várias visões navegacionais, de acordo com o perfil do usuário alvo e seus interesses navegacionais. A ferramenta permite que o projetista possa definir visões distintas para diferentes perfis de usuários.

Assim como na fase de modelagem conceitual, as ferramentas de especificação de classes, relacionamentos, composições, agregações e especializações são implementadas. O projetista poderá especificar características derivadas do modelo de domínio, criado na fase anterior. Para isso, algumas consistências são necessárias, para que o modelo seja coerente com as características conceituais da aplicação. A criação de elos derivados de relacionamentos estruturais devem ser verificados. Somente relacionamentos e classes constantes no modelo conceitual poderão ser derivados nesta

etapa. Assim como, somente interessa aqueles relacionamentos que permitem a navegação entre os objetos do modelo.

Outro aspecto a ser considerado é a consistência necessária entre as alterações efetuadas no modelo conceitual, após uma determinada visão navegacional ser instanciada. Nenhuma classe ou relacionamento conceitual pode ser excluído após a criação de uma visão navegacional. Assim como atributos e métodos de uma classe conceitual devem ser mantidos. Somente a inclusão de novos atributos, métodos e relacionamentos são permitidos no modelo de domínio da aplicação.

Durante a definição de um perfil de navegação, somente um valor por atributo será visualizado em cada nó, embora as entidades conceituais possuam diversas perspectivas de valores para seus atributos, conforme [ROS 96]. A Figura 4-6 apresenta a interface de modelagem navegacional da aplicação. Nesta ilustração, nota-se que a aplicação possui duas visões navegacionais modeladas. A visão do editor e a visão do autor. Uma determinada visão é selecionada através de um menu e sua representação gráfica é apresentada na área de modelagem. Através da interface de criação de esquemas de classes navegacionais, o projetista poderá inserir, alterar ou excluir visões navegacionais do modelo. A visão apresentada na ilustração é a visão do autor.

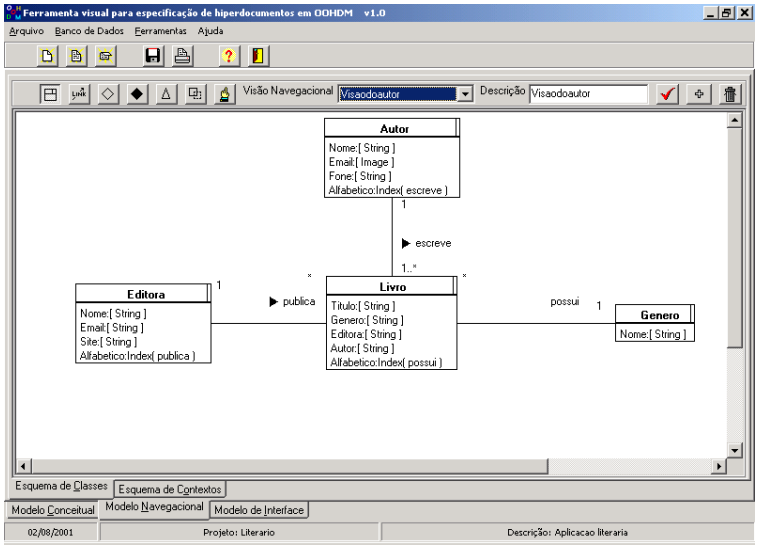


FIGURA 4.6- Modelagem Navegacional (visão do autor).

A visão do autor apresenta dois nós modelados: o nó “Autor” e o nó “Livro”. Um autor é representado através dos atributos “nome”, “E-mail”. Esses são derivados da classe conceitual “Autor” do modelo conceitual da aplicação. Observe que nem todos os atributos foram utilizados nesta visão. Assim como o atributo “E-mail” foi definido através de sua perspectiva “Image”. A perspectiva “String” não foi utilizada. Já o nó “Livro” é definido através da combinação de atributos de quatro classes conceituais: “Livro”, “Autor”, “Editora” e “Gênero”. Observa-se, ainda, que o nó “Autor”, possui um atributo que representa um índice, originado do relacionamento conceitual “Autor escreve Livro”. Um relacionamento conceitual pode originar dois tipos de atributos em um nó: um índice ou uma âncora. O atributo do tipo índice está associado a classe que possui cardinalidade “1”. Neste caso, deve-se incluir o atributo índice na classe. Já o atributo do tipo âncora está associado a classe de cardinalidade “*”. Então, deve-se incluir um atributo do tipo âncora nesta classe.

Na Figura 4-7, foi representada a visão do editor. Esta visão é composta pelos nós “Editora”, “Livro” e “Autor”. Nesta visão o interesse está em determinar que livros são publicados pela editora. Um índice de livros publicados é acessado através da estrutura de acesso “índice de livros”, modelada no nó editora, através do atributo “livro” associado ao índice “publica”.

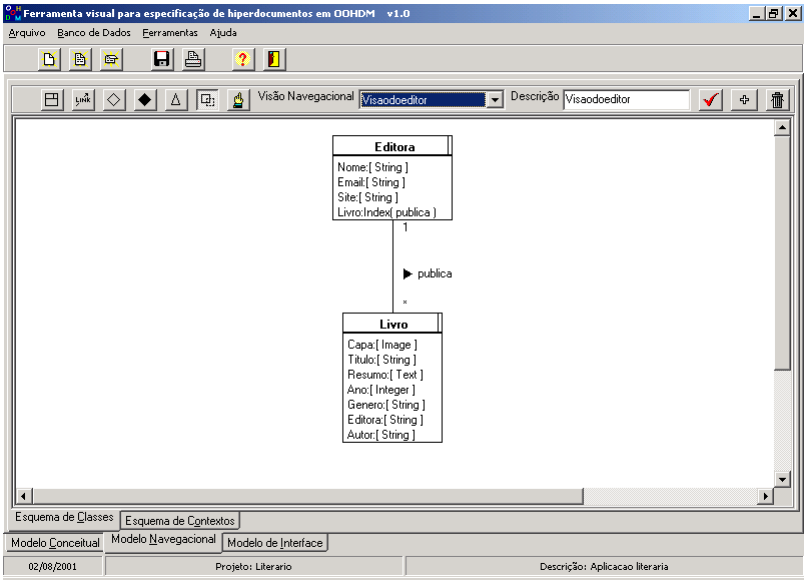


FIGURA 4.7- Modelagem Navegacional (visão do editor).

A Figura 4-8 apresenta as interfaces de especificação dos atributos e elos de um nó navegacional. Os atributos são selecionados a partir de classes constantes no modelo conceitual. Para isso, uma determinada classe de origem deve ser selecionada. O nó recebe, inicialmente, o nome da classe que o originou. O nome do nó pode ser alterado através de uma caixa de edição. Cada atributo é incluído na lista de atributos, através da definição sua perspectiva e descrição. A classe de onde o nó é derivado é apresentada na interface. Um determinado atributo pode ser adicionado, alterado ou removido a cada instante – Figura 4-8 (a).

A definição dos elos é realizada através da seleção do nó do modelo e edição de suas propriedades através da caixa de diálogo de especificação de elos, ver a Figura 4-8 (b). Nesta interface, define-se o tipo de atributo que irá ser derivado do elo, ou seja, se é um índice, uma âncora, uma lista, ou combinação de elementos. O nome do contexto é determinado, assim como o sentido da navegação. No exemplo apresentado, a navegação ocorre somente do nó “Autor” ao nó “Livro”. Nesta interface, também, é apresentado o nome do relacionamento conceitual origem do elo.

Uma consistência entre as características dos nós navegacionais, existentes no modelo de navegação, com as definições constantes no diagrama de classes conceituais se faz necessária. Visto que, um determinado nó combina atributos das classes do modelo conceitual. Para cada visão navegacional modelada, deve ser definido um diagrama de contextos navegacionais. Esta característica, proposta em OOHDM, tem a finalidade de complementar um determinado esquema navegacional.

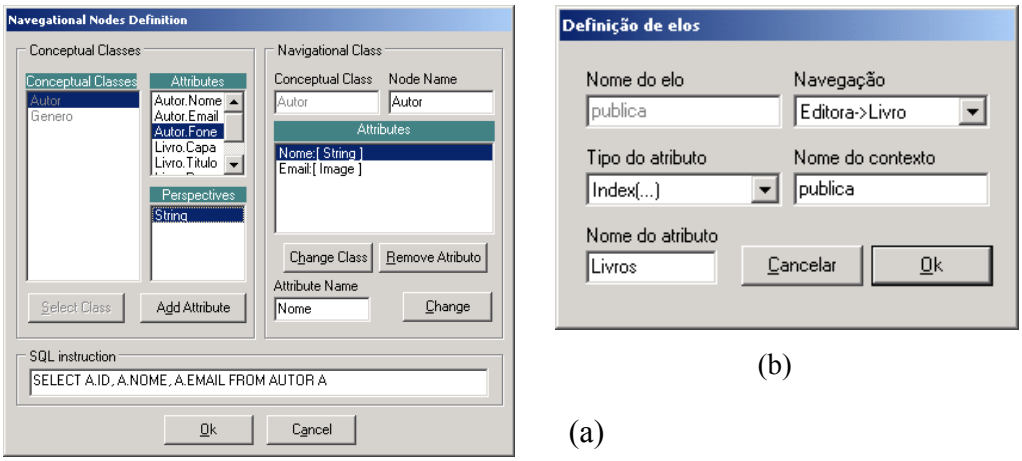


FIGURA 4.8- Interfaces de especificação de nós (a) e elos (b).

No diagrama de contextos navegacionais são definidas as principais estruturas de acesso da aplicação, na forma de índices. E, também, o conjunto de contextos para cada nó modelado. Este conjunto de contextos define como a informação a respeito de um determinado nó será apresentada a cada instante, ou seja, o conjunto de instâncias e formas de ordenação dos elementos. Diferentes contextos podem ser definidos para serem utilizados em distintas situações, de acordo com o perfil de usuário alvo. Os índices indicam qual o contexto, ou o conjunto de contextos que será acessado. A Figura 4-9 apresenta o esquema de contexto navegacional para a visão do autor.

Alguns nós poderão ser apresentados com características distintas, dentro de um contexto particular da aplicação. Determinados atributos, que não foram definidos em um nó navegacional podem ser apresentados somente em um contexto específico. Veja que, neste caso, um atributo ou conjunto de atributos não pode ser modelado no esquema navegacional, pois as características modeladas no diagrama de classes navegacionais são apresentadas em todos os contextos que referenciam o nó. Portanto, para casos particulares, como esses, são definidas classes em contexto.

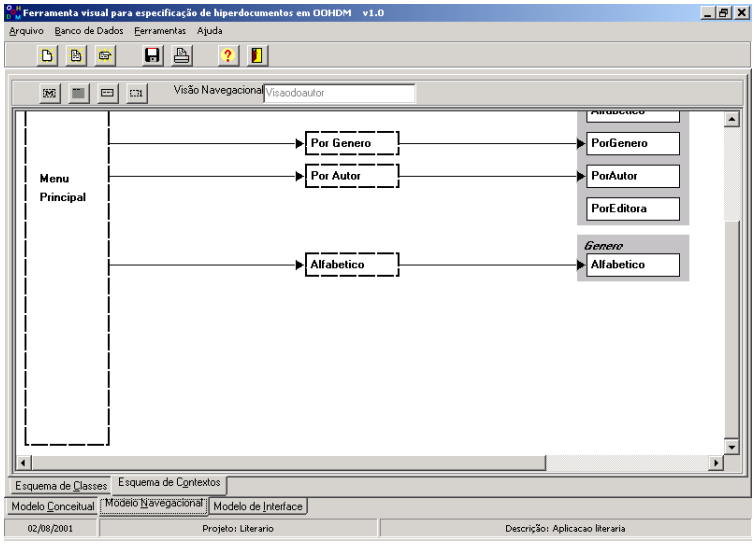


FIGURA 4.9- Interface de contexto navegacional (visão do autor).

4.2.3 Módulo de definição de interface

Após a definição das características conceituais do hiperdocumento e dos perfis de navegação, o projetista deverá definir as características da interface da aplicação. A ferramenta disponibiliza um modelo de interface formado por um menu principal e uma área de apresentação. A região de apresentação do *browser* é delimitado por dois *frames*. No *frame* à esquerda será apresentado o menu principal da aplicação. Já no *frame* mais à direita será mostrado o conteúdo selecionado através do menu principal.

Através do módulo de definição de interface o projetista poderá definir as características de apresentação do hiperdocumento, tais como: tipos de fontes, disposições e cores dos objetos de interface apresentados.

Como a aplicação irá gerar um modelo de documento que será transformado em HTML, para posterior apresentação no *browser*, a aplicação deverá estar intimamente relacionada a um determinado ambiente de implementação. O ambiente de implementação adotado é o baseado em HTML. Sendo assim, o modelo de projeto de interface adotado no método OOHDM não foi utilizado no projeto da aplicação.

Uma determinada interface abstrata instancia objetos de navegação abstratos, definidos na fase de navegação, tais como: nós e elos, em objetos de representação visual real. Para isso, deve-se associar mídias aos atributos dos nós navegacionais. As mídias serão encontradas em uma base de armazenamento de objetos de apresentação, definida na arquitetura da ferramenta.

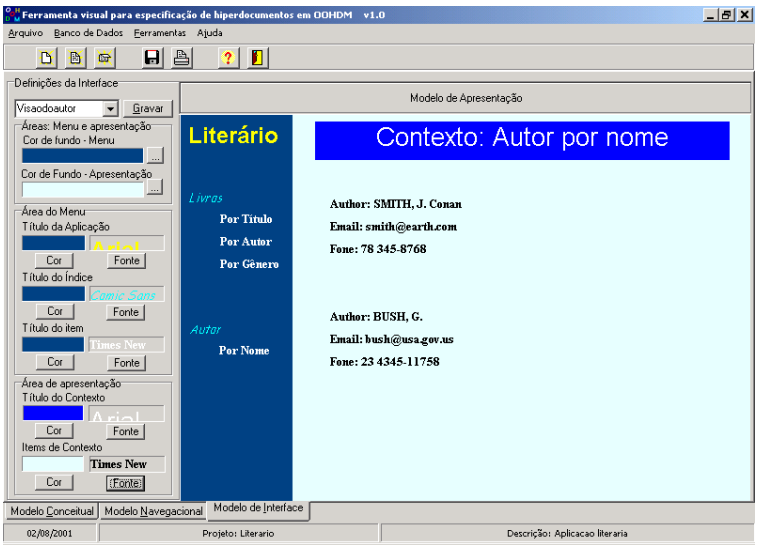


FIGURA 4.10- Interface de especificação das características de Interface.

4.3 Mantendo a aplicação

A qualquer momento, o projetista poderá salvar o modelo criado. Para isso, deverá acessar o menu principal do software e selecionar a opção "salvar como", a aplicação será gravada para posterior manutenção. Caso a aplicação já tenha sido gravada, anteriormente, basta selecionar "salvar". Na barra de atalhos para as opções do menu existe um botão que permite o acesso rápido a estas funções. Ele é representado através da ilustração de um disquete. Veja a Figura 4.11.

Selecionando-se a opção de gravação, abrirá uma caixa de diálogo, onde o projetista irá informar os dados de identificação da aplicação. Informações como nome do projeto, uma breve descrição, autor, versão e data serão requeridas.

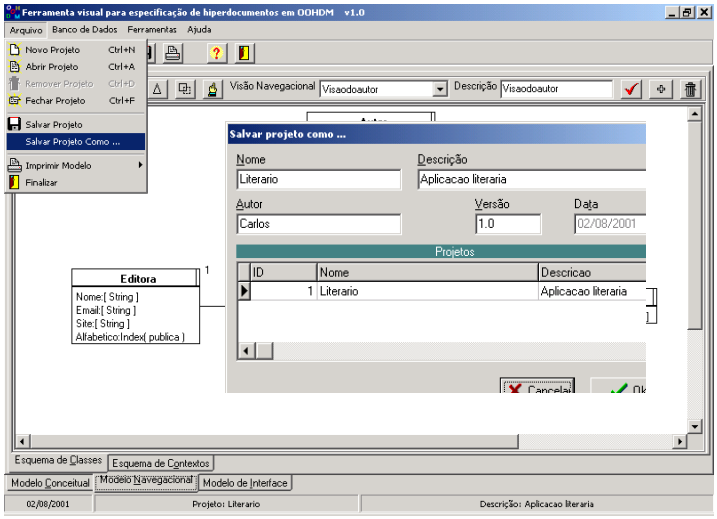


FIGURA 4.11- Gravando os modelos gráficos.

Um projeto, previamente criado, poderá ser aberto. Para que o projetista possa realizar manutenções. A opção do menu principal para abertura de um projeto é apresentada através da Figura 4.12.

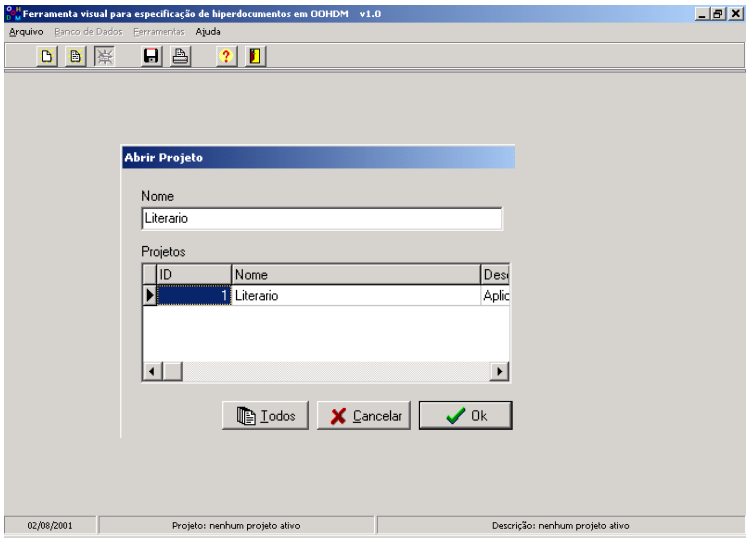


FIGURA 4.12- Abrindo o projeto de uma aplicação.

4.4 Implementação do hiperdocumento

A implementação do hiperdocumento final será realizada através de um módulo gerador de código. Este módulo realiza duas etapas distintas. Em uma primeira etapa o módulo irá realizar o mapeamento do modelo de visões navegacionais abstratas em tabelas em um modelo relacional. Estas tabelas guardam as informações que serão apresentadas no hiperdocumento. Ferramentas de manutenção de dados permitirão que o usuário alimente e mantenha essas tabelas. Em uma segunda etapa, o gerador de

código irá gerar arquivos XML, a partir das definições de navegação e contexto. Este módulo irá agregar os dados constantes nas tabelas.

Os documentos XML gerados utilizarão folhas de estilo associadas para a transformação e formatação de modelos de apresentação, através da criação de arquivos HTML. Assim, pretende-se dar maior flexibilidade de apresentação do hiperdocumento, gerando distintos modelos de apresentação, para diferentes árvores navegacionais de um mesmo modelo conceitual abstrato.

4.4.1 Gerando o banco de dados relacional

O modelos abstratos, elaborados através da interface gráfica da ferramenta, são transformados em tabelas relacionais. Dessa forma, possibilitando a inserção dos dados que irão compor os hiperdocumentos gerados.

Cada classe do modelo conceitual será mapeada em uma tabela. O nome da tabela gerada será o mesmo descrito para a classe conceitual. As definições para cada atributo da classe, constantes no modelo abstrato, tais como: nome do atributo, tipo, tamanho e perspectiva, serão traduzidos em um campo da tabela. A Figura 4-13, apresenta parte de uma aplicação literária e é utilizada como um exemplo para demonstração de como ocorrem os mapeamentos do modelo abstrato. Na ilustração é apresentado o relacionamento da classe "Autor" com a classe "Livro". Um autor pode escrever vários livros, assim como um determinado livro pode ser escrito por vários autores, assim, o relacionamento escreve possui uma cardinalidade "N:N".

FIGURA 4.13- Relacionamento conceitual Autor escreve Livro.

A Tabela 4-1, reflete o mapeamento da classe abstrata "Autor", representada no diagrama, em tabelas do banco de dados relacional. A coluna esquerda da tabela contém a estrutura de dados que define a classe conceitual "Autor". Esta estrutura é representada por uma classe que contém as propriedades: um campo chave de identificação da classe (ID), a descrição da classe, um vetor contendo o conjunto de atributos, um vetor contendo um conjunto de relações que são originadas a partir da classe e um vetor contendo um conjunto de relações que têm destino na classe.

Por sua vez, cada atributo da classe possui uma ou mais perspectivas, que são representadas através de uma estrutura de dados que guarda as informações referentes a cada perspectiva. São elas: a identificação da classe origem da perspectiva, o atributo que a perspectiva está relacionada e o nome da perspectiva.

As relações que são originadas da classe, também, são descritas em uma estrutura de dados adequada, com suas características como: a descrição que identifica a relação, o tipo da relação (relacionamento, agregação, composição ou especialização), as cardinalidades origem e destino e as entidades fonte e destino da relação.

Assim como as relações origem, existem as relações que tem destino na classe. Portanto, uma estrutura de dados para a definição das relações destino se faz necessária. Nela estão os dados que definem a descrição da classe de origem, a relação da classe de origem ligada à classe em questão e o tipo da relação.

Já na coluna da direita, estão as tabelas geradas a partir do modelo de dados anterior. Uma tabela é criada com o nome da classe conceitual. Nesta tabela são introduzidos o conjunto de atributos modelados, com seus respectivos tipos e tamanhos de dados. Para cada perspectiva é criado um novo atributo na tabela.

Caso existam relacionamentos que partem da classe conceitual. Deve ser verificada a cardinalidade deste relacionamento. Se a cardinalidade for do tipo "N:1", um novo atributo é inserido na tabela para referenciar a tabela origem do relacionamento. Mas, se a cardinalidade for do tipo "N:N", uma nova tabela é criada, cujo nome é a combinação do nome da tabela origem, a descrição do relacionamento e o nome da tabela destino.

TABELA 4.1- Modelo de mapeamento em tabelas relacionais.

Classes conceituais e estruturas de dados	Tabelas em Paradox (exemplo classe Autor)

Estrutura da Classe
EntidadesMC = class
public
ID: integer;
Descricao: string[10];
Atributos: array of AtributosMC;
Relacoes: array of RelacoesMC;
RelacoesDestino: array of RelacoesDestinoMC;
end;
 	
Conjunto de atributos da classe
AtributosMC = record
Nome: string;
Tipo: string;
Perspectivas: array of PerspectivasMC;
end;
 	
Conjunto de perspectivas de cada atributo
PerspectivasMC = record
IDClasse: integer;
IDAtributo: integer;
Perspectiva: string;
Nome: string;
end;
 	
Conjunto de relações origem da classe
RelacoesMC = record
Descricao: string;
Tipo: byte;
EntidadeFonte: integer;
EntidadeAlvo: integer;
CardinalidadeOrigem: integer;
CardinalidadeDestino: integer;
end;
 	
Conjunto de relações destino da classe
RelacoesDestinoMC = record
Classe: integer;
Relacao: integer;
Tipo: byte;
End;
	Nome da tabela: Autor.db
	Campos Tipo Tamanho
	(código do autor)
	IDAutor Number
	(Conj. de atrib. da classe, para cada perspectiva modelada)
	AtribNome AtribTipo AtribTam
	(Conj. de atrib. originados de um relacionamento destino, para cardinalidades N:1, definidos no conjunto de relações destino da classe conceitual)
	AtribRel AtribRelTipo AtribRelTam
	
	Nome da tabela: AutorEscreveLivro.db
	(tabela originada do relacionamento Autor escreve Livro, cardinalidade N:N, definido no conjunto de relações origem da classe conceitual)
	Campos Tipo Tamanho
	(Chave do relacionamento Autor escreve Livro)
	ID
	(código do autor)
	IDAutor Number
	(código do livro)
	IDLivro Number

4.4.2 Inserindo os dados no banco

Após a criação do banco de dados de informações, os dados devem ser alimentados nas tabelas, para que se possa originar os hiperdocumentos. Para isso, foram inseridas ferramentas de manutenção de dados no ambiente da ferramenta. A Figura 4-14 apresenta o menu para criação do banco de dados relacional da aplicação.

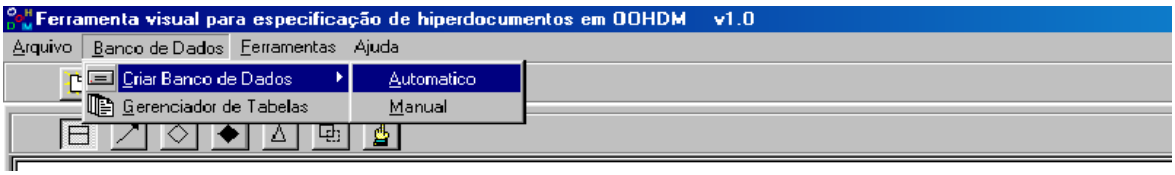


FIGURA 4.14- Menu de criação do banco de dados da aplicação

O projetista poderá optar por gerar o banco de dados automaticamente, ou então definir as propriedades para cada tabela manualmente. O software irá aplicar algumas regras para a criação das tabelas e seus relacionamentos, tais como:

Será gerada uma tabela para cada classe definida no modelo conceitual;

para relacionamentos do tipo 1 para n, será incluído um atributo chave na tabela que possui a cardinalidade n no relacionamento;

para relacionamentos do tipo n para n, será criada uma nova tabela resultante.

Após a criação do banco de dados de informações, os dados devem ser alimentados nas tabelas para que os hiperdocumentos sejam gerados. Para isso, foram inseridas ferramentas de manutenção de dados que permitem esta operação.

As tabelas previamente criadas, são apresentadas na interfaces, as quais podem ser selecionadas para edição pelo usuário. Após selecionar a tabela o usuário alimenta os dados que servirão de base de informação da aplicação final. A interface permite a inclusão, edição e exclusão de dados.

A Figura 4-15 apresenta a interface de manutenção das tabelas criadas através do mapeamento das classes conceituais.

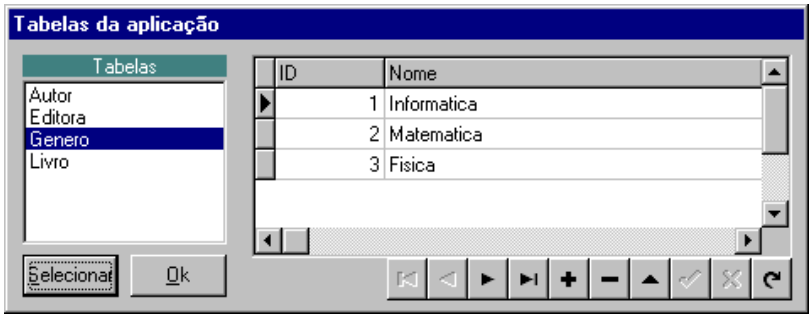


FIGURA 4.15- Interface de manutenção das tabelas.

4.4.3 Gerando os arquivos XML

Com o mapeamento do modelo conceitual em tabelas de um banco de dados relacional e com as informações já alimentadas, o projetista poderá gerar os arquivos no formato XML. Através desses arquivos, o software utilizará folhas de estilo em XSL para modelar o hiperdocumento em HTML que apresentará as informações no *browser*. Para isso, instruções XSLT serão incorporadas na folha de estilo para selecionar, ordenar e formatar as informações que serão apresentadas.

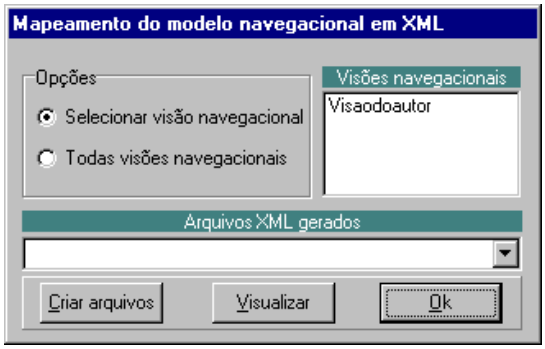


FIGURA 4.16- Interface de mapeamento das tabelas em XML.

Veja, através da Figura 4-17, um exemplo de um arquivo XML resultante do mapeamento das informações constantes da tabela de autores.

```
<Autores>
  <Autor>
    <ID>3</ID>
    <Nome>Beltrano</Nome>
    <Email></Email>
    <Fone>2421345</Fone>
  </Autor>
  <Autor>
    <ID>2</ID>
    <Nome>Ciclano</Nome>
    <Email></Email>
    <Fone>2426472</Fone>
  </Autor>
  <Autor>
    <ID>1</ID>
    <Nome>Fulano</Nome>
    <Email></Email>
    <Fone>2425897</Fone>
  </Autor>
</Autores>
```

FIGURA 4-17. Arquivo XML contendo as informações sobre autores.

4.4.4 Gerando os modelos de apresentação com XSLT e XSL

A Segunda etapa do processo de implementação da aplicação, após o mapeamento do conteúdo das tabelas em arquivos XML, trata-se da associação de folhas de estilo para a transformação e formatação das informações em HTML. Com este processo, torna-se possível apresentar o conteúdo e testar a aplicação em um

browser compatível com XML. O *browser* utilizado para testar a aplicação é o Internet Explorer 5 da Microsoft.

A Figura 4-18 apresenta a folha de estilo para transformação e formatação do arquivo de autores apresentado na Figura 4.17.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
    <html>
     <head>
       <title>Application: Literario</title>
     </head>
     <body bgcolor="FFFFFF">
<center><table><th bgcolor="FFFFFF">
        <font color="080000" face="MS Sans Serif" size="8">
          Contexto: Autor Alfabetico
        </font>
</th></table></center>
     <br/><br/><br/>
     <xsl:apply-templates/>
     </body>
   </html>
 </xsl:template>
 <xsl:template match="Autor-XML">
  <table border="1" align="center">
   <th bgcolor="black"><font face="Times" size="4" color="white">ID</font></th>
   <th bgcolor="black"><font face="Times" size="4" color="white">Nome</font></th>
   <th bgcolor="black"><font face="Times" size="4" color="white">Email</font></th>
   <th bgcolor="black"><font face="Times" size="4" color="white">Fone</font></th>
   <xsl:for-each order-by="+ Nome" select="Autor" xmlns:xsl="http://www.w3.org/TR/WD-xsl" >
     <tr>
       <td bgcolor="white"><font face="Times" size="2" color="black"><xsl:value-of select="ID"/></font></td>
       <td bgcolor="white"><font face="Times" size="2" color="black"><xsl:value-of select="Nome"/></font></td>
       <td bgcolor="white"><font face="Times" size="2" color="black"><xsl:value-of select="Email"/></font></td>
       <td bgcolor="white"><font face="Times" size="2" color="black"><xsl:value-of select="Fone"/></font></td>
     </tr>
   </xsl:for-each>
  </table>
 </xsl:template>
</xsl:stylesheet>
```

FIGURA 4-18. Folha de estilo para transformação e formatação do documento em HTML.

4.4.5 Gerando a aplicação

A última etapa do processo de implementação é a criação e execução da aplicação final. Para isso, foi inserido um módulo que irá gerar o modelo do hiperdocumento. Este módulo irá criar o menu principal da aplicação que conterá os índices para cada contexto definidos no modelo de contextos navegacionais. Através da opção para executar a aplicação, o projetista poderá testar e avaliar o seu projeto. Para a execução da aplicação é necessário que o *browser* Internet Explorer da Microsoft esteja instalado. O ferramenta visual para especificação de hiperdocumentos faz uma chamada ao *browser*, passando como parâmetro o arquivo HTML que constitui o módulo

principal da aplicação. A partir do módulo principal da aplicação são acessadas as opções do menu. A Figura 4-19 apresenta a interface de criação da aplicação.

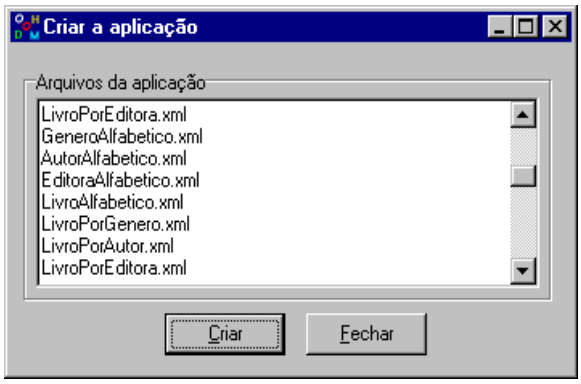


FIGURA 4-19. Interface de criação da aplicação.

4.5 Exportação da especificação para OOHDM-ML

A ferramenta será desenvolvida em linguagem DELPHI, que possui uma arquitetura de banco de dados proprietária, cujo formato de seus dados é fechado. Para contornar esta limitação, a ferramenta irá gerar um formato de dados aberto, permitindo a exportação dos modelos de projeto criados a partir de sua interface. O formato a ser adotado é o XML, contendo uma DTD com informações sobre a estrutura do hiperdocumento modelado com a ferramenta, de acordo com a especificação OOHDM-ML [PON 98]. Tal característica, irá possibilitar a integração da ferramenta com o ambiente de implementação de projetos denominado OOHDMWeb [MOU 99]. Este ambiente implementa hiperdocumentos, a partir de tabelas em *Access*. Atualmente, está sendo desenvolvida uma extensão deste ambiente, para a criação de modelos de apresentação em XML.

4.5.1 A DTD OOHDM-ML

O objetivo da criação de uma DTD para OOHDM-ML é a definição de um conjunto de regras que ditam como um hiperdocumento XML deve ser projetado [MED 01]. O modelo de definição do tipo de documento está no arquivo denominado OOHDM.DTD. A especificação OOHDM-ML apresenta cada elemento, através de sua descrição e definição na DTD e exemplos de marcação em um documento XML. A FIGURA 4-13 apresenta a estrutura de uma especificação de hiperdocumentos, de acordo com a especificação OOHDM-ML. Os principais elementos, com seus principais atributos, são apresentados. Para que uma estrutura mínima de um documento possa ser gerada. O elemento <OOHDM> é o elemento raiz da estrutura. Este elemento contém todos os demais elementos do documento. Possui os atributos *name* e *version*, que identificam, respectivamente: o nome da aplicação e a versão. Como este elemento é o raiz da estrutura, sua definição é obrigatória, bem como a definição de seus dois atributos. Este elemento é formado por um, ou mais, elementos do tipo <OOHDM_BASE> e, zero ou mais elementos do tipo <OOHDM_WEB>.

Observe que a definição de pelo menos um elemento do tipo <OOHDM_BASE> é obrigatória. Pois, a partir deste elemento, serão geradas implementações do hiperdocumento em OOHDMWeb. O elemento <OOHDM_BASE> possui toda a

especificação do projeto. Para isso, ele é constituído de elementos do tipo <conceptual_model>, <navigational_model> e <interface_model>. Esses elementos permitem a definição das características conceituais, navegacionais e de interface da aplicação. Por sua vez, cada modelo do método, também possui um subconjunto de elementos. Para a definição das informações de domínio da aplicação, deve-se utilizar uma marcação contendo um elemento do tipo <conceptual_model>. Esta marcação irá conter todas as definições para as classes e relacionamentos modelados, segundo as primitivas do método OOHDM. Após a marcação do modelo conceitual da aplicação, parte-se para as definições navegacionais, através de um elemento do tipo <navigational_model>. Este elemento especifica as classes navegacionais modeladas na especificação, tais como: conjunto de nós, elos, contextos navegacionais e classes em contexto. Por fim, as definições de interface abstrata são especificadas pelo elemento <interface_model>. Esta marcação ainda está sendo elaborada, portanto ainda não existe um modelo de especificação de interface em OOHDM-ML.

4.5.2 Template para exportação do hiperdocumento

Na especificação OOHDM-ML [PON 98] foram propostos *templates*, que permitem a estruturação do projeto de uma aplicação de forma modular. Separando em níveis os hiperdocumentos de especificação. Há um módulo principal do hiperdocumento, que contém a estrutura básica do projeto. A partir deste módulo, são carregados os módulos de especificação conceitual, navegacional e de interface. Esses módulos são referenciados como entidades externas. A Figura 4-14, apresenta o *template* proposto para o módulo principal da especificação.

```
<OOHDM name=  version= >
  <OOHDM_BASE>
    <conceptual_model>
      <conceptual_class  id=  name= >
            <attrib name=  type=  size= >
             <perspective name=  type= size=  default= />
             <perspective name=  type= />
            </attrib>
            <operation name=  property=  visibility= >
              <parameter>
              </parameter>
              <return_type>
              </return_type>
            </operation>
       </conceptual_class>
       <relationship id= name= source_class=  target_class= source_cardinality= target_cardinality= >
       </relationship>
    </conceptual_model>
    <navigational_model>
      <navigational_class id= name=>
            <nav_attrib name= conceptual_class=/>
            <nav_attrib name= conceptual_class=/>
            <nav_attrib name= conceptual_class=/>
            <nav_attrib name= conceptual_class=/>
            <nav_attrib name= conceptual_class=/>
               <perspective_attrib name= perspective=  conceptual_attrib=  conceptual_class=  optional=/>
               <perspective_attrib name= perspective=  conceptual_attrib= conceptual_class=/>
      </navigational_class>
    </navigational_model>
    <interface_model/>
  </OOHDM_BASE>
  <OOHDM_WEB/>
</OOHDM>
```

FIGURA 4.13- Estrutura de um documento OOHDM-ML.

A ferramenta visual para espeficicação de hiperdocumentos utiliza o modelo de *template*, para a exportação do modelo gráfico em OOHDM-ML. O módulo principal do projeto, definido através da figura anterior, é utilizado como documento principal da especificação. Note que o modelo de projeto segue as definições contidas em OOHDM.DTD. O nome do documento principal recebe o nome da aplicação. As entidades externas "modelo_conceitual" e " modelo_navegacional" referenciam os respectivos documentos para especificação das características conceituais e navegacionais.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OOHDM SYSTEM "../OOHDM.dtd"
[
  <!ENTITY modelo_conceitual  SYSTEM "modelo_conceitual.xml">
  <!ENTITY modelo_navegacao   SYSTEM "modelo_navegacao.xml">
]>
<OOHDM name="Nome da Aplicação" version="1.0">
  <!-- Projeto OOHDM da aplicacao -->
  <OOHDM_BASE>
    <!-- Definicao do Modelo Conceitual -->
    <conceptual_model>
          &modelo_conceitual;
    </conceptual_model>
    <!-- Definicao do Modelo de Navegacao -->
    <navigational_model>
          &modelo_navegacao;
    </navigational_model>
    <!-- Definicao do projeto da Interface Abstrata -->
    <interface_model/>
  </OOHDM_BASE>
  <!-- Mapeamento dos modelos da aplicacao para o ambiente OOHDM-WEB -->
  <OOHDM_WEB/>
</OOHDM>
```

FIGURA 4.14- Módulo principal da especificação OOHDM-ML.

O projetista, após modelar o hiperdocumento, graficamente através da interface do software, deverá selecionar a opção de exportação da especificação. No menu principal da ferramenta há opções para a criação dos documentos de especificação. O projetista poderá gerar a qualquer momento toda a especificação, ou um módulo específico. Veja a Figura 4.15.

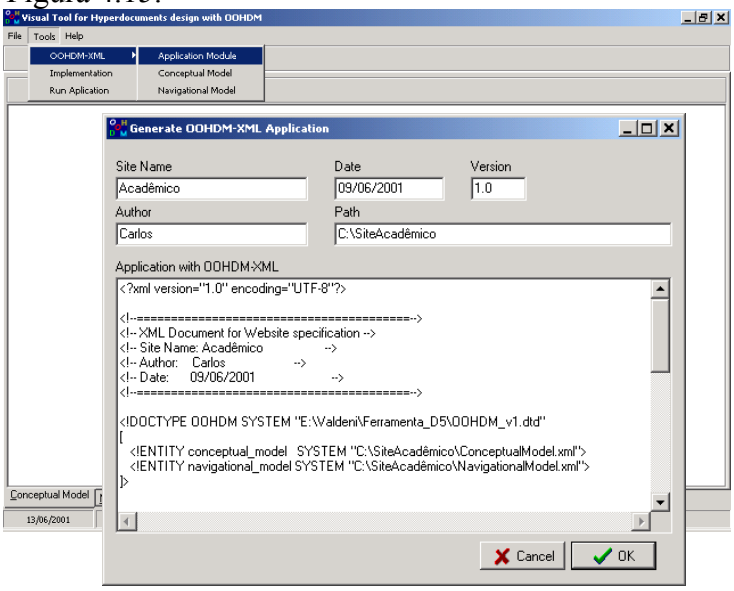


FIGURA 4.15- Módulo de exportação dos dados para OOHDM-ML.

4.5.3 Template para exportação do modelo conceitual

O modelo conceitual gráfico da aplicação é exportado em um *template* OOHDM-ML. Neste *template* estão todas as classes conceituais definidas no modelo, bem como seus relacionamentos. As propriedades e métodos de cada classe do modelo são especificadas através do elemento <*conceptual_class*>, que é composto por um conjunto de elementos do tipo <*attrib*> e <*perspective*>. Para os relacionamentos do modelo também é definido um elemento o <*relationship*>, onde as características do relacionamento entre as classes do modelo são definidas. A Figura 4-16 apresenta um modelo de template para um diagrama de classes conceituais de uma aplicação literária.

```
<!--==================-->
<!-- Conceptual Model -->
<!--==================-->

<!-- Conceptual class: "Autor" -->

<conceptual_class id="ID" name="Autor" property="Classe Autor" >
  <attrib name="Nome" Type="String" size="255" >
  </attrib>
  <attrib name="Email"  >
    <perspective name="Email1" Type="Image" > </perspective>
    <perspective name="Email2" Type="String" > </perspective>
  </attrib>
  <attrib name="Fone" Type="String" size="255" >
  </attrib>
</conceptual_class>

<!-- Relationship Editora publica Livro -->

<relationship id="IDpublica" name="publica"
          source_class="Editora" target_class="Livro"
          source_cardinality="1"  target_cardinality="n" >
</relationship>
```

FIGURA 4.16- *Template* para um diagrama de classes conceituais.

4.5.4 Template para exportação do modelo navegacional

Assim como o *template* do modelo conceitual, também é criado um *template* para exportação do modelo navegacional da aplicação em OOHDM-ML. Nele estão as definições dos nós e links da aplicação através de seus respectivos elementos.

5 Estudo de Caso

Para aferição e avaliação do software proposto foi sugerido o projeto de uma aplicação exemplo. A aplicação escolhida foi o projeto de um hiperdocumento literário. O projeto literário é composto pelas seguintes entidades: obras, autores, biografia, gêneros e editoras. Através deste exemplo prático, pretende-se dar uma visão geral do processo de especificação abstrata e de implementação de aplicações com a ferramenta, bem como demonstrar suas funcionalidades.

5.1 Especificação do hiperdocumento literário

O projeto da aplicação segue as etapas propostas no método OOHDM e que são implementadas na interface da ferramenta visual. A primeira etapa é a delimitação de domínio, seguida pela especificação dos objetos navegacionais e de contextos. A seguir, será elaborada a interface da aplicação.

Ao término da especificação abstrata, o projetista poderá realizar o mapeamento das estruturas gráficas em tabelas de um banco de dados relacional, de onde serão capturadas as informações que serão apresentadas no hiperdocumento da aplicação.

5.1.1 Especificando o modelo conceitual

Ao iniciar um novo projeto, clicando em novo projeto, no menu principal da interface do software, será apresentada ao usuário a área de modelagem conceitual. A partir desta interface será modelado o esquema de classes conceituais composto de mecanismos de classificação, generalização/especialização, agregação/composição e relacionamentos. Este é o ponto de partida do projeto, portanto o software irá realizar um controle na evolução de etapas do projeto da aplicação, não permitindo que o projetista omita etapas durante o projeto da aplicação. Esta é uma característica estabelecida no método OOHDM, ou seja, todo projeto é construído de acordo com uma sequência evolutiva de etapas.

A aplicação literária possui em seu domínio as seguintes características:

Classes conceituais: Obras, Autores, Editoras, Gêneros, Biografias;

Relacionamentos: Autores com Obras, Editoras com Obras, Obras com Gêneros, Autores com Biografias, Obras com Obras.

O diagrama de classes conceituais é apresentado através da Figura 5.1.

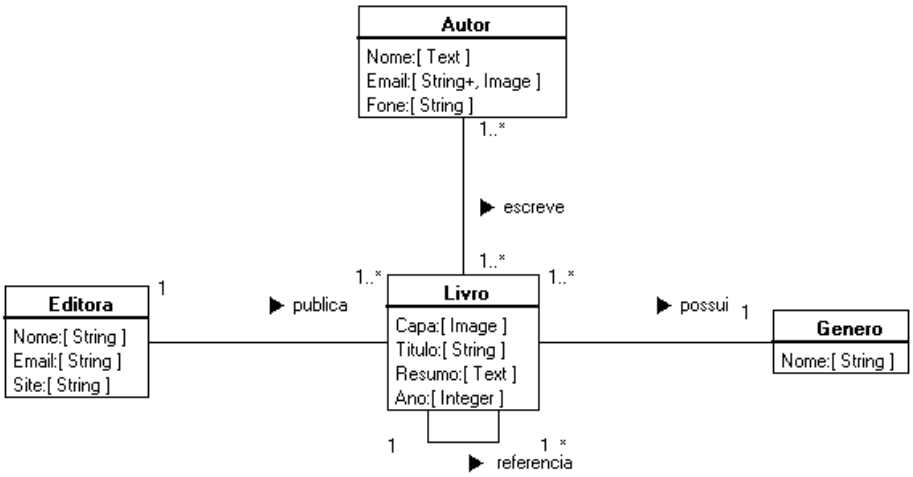


FIGURA 5.1- Diagrama de classes conceituais.

5.1.2 Especificando a navegação e contextos

Após a definição conceitual da aplicação, através da modelagem conceitual do hiperdocumento, o projetista deverá especificar modelos de navegação. Estes modelos são representados através dos diagramas de classes navegacionais, onde diferentes perfis podem ser traçados, de acordo com as preferências do usuário final.

Nesta etapa, o projetista poderá adicionar, editar ou remover as visões navegacionais que está criando a qualquer momento. Uma visão navegacional é elaborada através do mapeamento de classes e relacionamentos envolvidos no diagrama de classes conceituais. Portanto, uma consistência é necessária nesta etapa, somente as classes e relacionamentos constantes no diagrama de classes conceituais poderão ser utilizados no modelo navegacional. Bem como, a alteração ou exclusão de elementos do modelo conceitual não será permitida se já existir uma visão navegacional modelada.

Uma outra consistência realizada pelo software é a verificação da existência ou não de um modelo conceitual. O projetista somente poderá especificar uma navegação se um modelo conceitual, contendo pelo menos uma classe conceitual, já tenha sido modelado. A Figura 5.2 apresenta o diagrama navegacional, segundo a visão do autor.

Foram modelados dois nós navegacionais, baseados nas classes conceituais “Autor e Livro”. A partir do nó “Autor” é criado um elo para o nó “Livros”. Desta forma, o usuário poderá realizar uma navegação para os livros associados a um determinado autor.

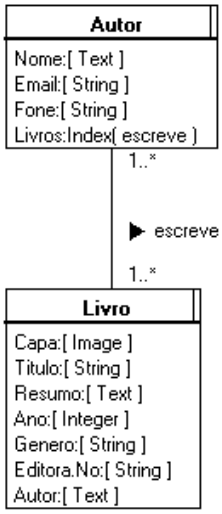


FIGURA 5.2- Modelo navegacional, visão do autor.

O projetista utiliza um diagrama de esquema de contextos de navegação para complementar a especificação navegacional,. Este diagrama demonstra como ocorrerá a navegação. Para isso, utiliza contextos de navegação associados a um índice. Veja a Figura 5.3.

No menu principal há entradas para cada índice associado a um contexto específico dentro de um nó navegacional. Cada nó pode possuir vários contextos associados.

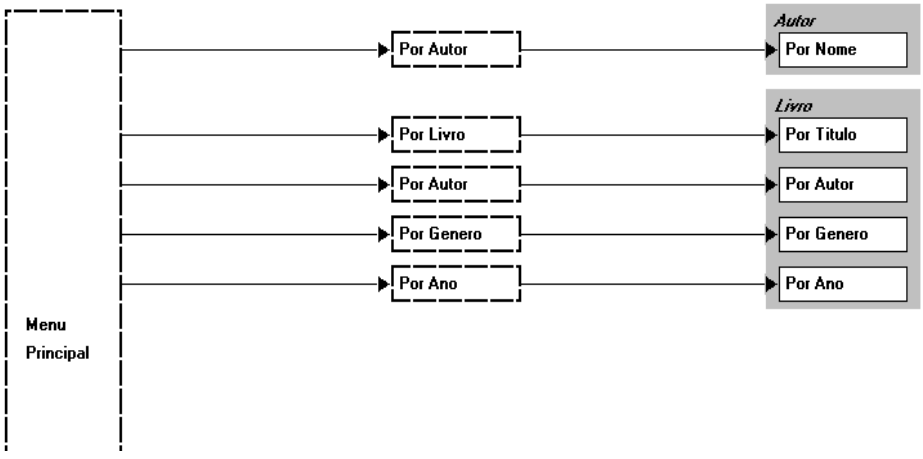


FIGURA 5.3- Esquema de contextos, visão do autor.

Na aplicação Literária, segundo a visão do autor, foram definidos dois nós navegacionais, conforme relatado anteriormente. O nó “Autor”, pode ser visitado de acordo com o contexto “Por Nome”, onde todos os autores são apresentados em ordem alfabética. Já o nó “Livro” possui os seguintes contextos: “Por Título”, “Por Autor”, “Por Gênero” e “Por Ano”, onde as informações sobre os livros podem ser consultadas segundo os critérios definidos por esses contextos.

5.1.3 Especificando a interface

Ao finalizar a modelagem do hiperdocumento, através das primitivas envolvidas em cada etapa do método OOHDM, o projetista poderá criar uma aplicação. A ferramenta disponibiliza módulos para a implementação e avaliação de aplicações modeladas.

Primeiramente, o projetista deverá definir a aparência dos elementos de interface da aplicação. Esta etapa é cumprida através do módulo de especificação de interface, onde as características visuais do hiperdocumento final serão definidas. Através do módulo de interface o projetista irá definir a aparência do menu principal, índices e títulos. A Figura 5-4 apresenta a tela de definição de interface do software.

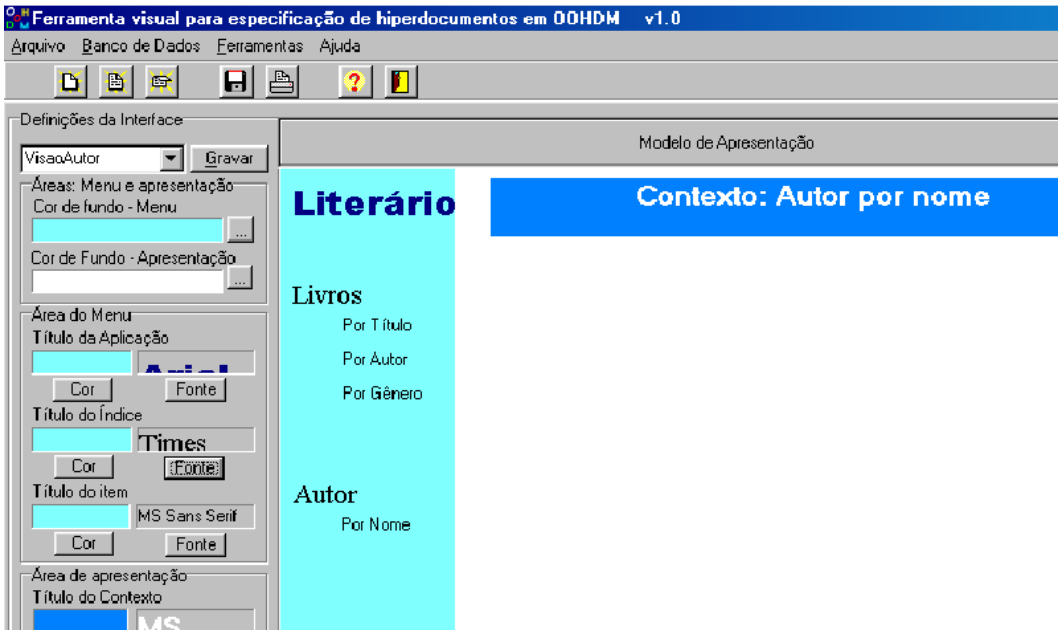


FIGURA 5.4- Tela de definição de interface.

5.2 Implementação do hiperdocumento literário

A próxima etapa do processo de desenvolvimento do hiperdocumento é o mapeamento do modelo abstrato da aplicação em um ambiente de implementação específico. A implementação é realizada em XML, pois trata-se de um formato de dados aberto.

Para implementar a especificação da aplicação são definidas algumas etapas:

Mapeamento das classes conceituais em tabelas de um banco de dados relacional.

Onde cada classe do modelo é mapeada para uma tabela. Os relacionamentos entre as classes do modelo também são mapeados em atributos em cada tabela, ou até mesmo em novas tabelas conforme a cardinalidade com que ocorrem. Na aplicação Literária são criadas as seguintes tabelas: “Autor”, “Livro”, “Editora”, “Gênero” e “Escreve”. Esta última tabela citada é proveniente do relacionamento múltiplo entre as classes “Autor” e “Livro” do modelo conceitual.

Inserção de informações nas tabelas.

Após a criação do bando de dados, as tabelas precisam ser alimentadas com dados que servirão de base de informações para o conteúdo do hiperdocumento. Esta etapa é realizada através do módulo de manutenção de tabelas disponibilizado pela ferramenta. Através deste módulo, o usuário poderá inserir, alterar ou excluir informações nas tabelas da aplicação.

Ao concluir as etapas anteriores, a aplicação final já poderá ser gerada. Para isso, o projetista deverá executar a opção de mapeamento de dados para arquivos XML. Ou seja, os dados constantes nas tabelas do banco de dados relacional serão exportados para o formato XML. Esta exportação de dados se faz necessária, devido ao formato de dados proprietário utilizado pelo banco de dados. Com a exportação dos dados em XML, pretende-se gerar uma aplicação para a *Web.*

A seguir, as folhas de estilo em XSL deverão ser unidas aos seus respectivos arquivos XML a fim de transformar e formatar os hiperdocumentos da aplicação. A implementação final da aplicação é realizada através da opção de menu "Criar Hiperdocumento". As opções de menu para mapeamento de dados em arquivos XML e criação do hiperdocumento final estão no menu "Ferramentas", "Implementação".

Os arquivos gerados pela aplicação estão representados na Tabela 5.1. Onde um módulo principal da aplicação é criado e, a partir dele, os contextos são acessados. A partir do menu principal da aplicação o usuário terá acesso aos índices da aplicação.

TABELA 5.1- Arquivos gerados pelo software.

ARQUIVO	DESCRIÇÃO
Literário.html	Módulo principal da aplicação que contém os *frames* HTML.
Menu.html	Módulo do menu principal da aplicação contém as entradas para os menus.
AutorPorNome.html	Arquivo do contexto autor por nome.
LivroPorTitulo.html	Arquivo do contexto livro por título.
LivroPorGenero.html	Arquivo do contexto livro por gênero.
LivroPorAutor.html	Arquivo do contexto livro por autor.
LivroPorAno.html	Arquivo do contexto livro por ano.
AutorPorNome.xsl	Arquivo de formatação do contexto autor por nome.

e.xsl	
LivroPorTitulo.xsl	Arquivo de formatação do contexto livro por título.
LivroPorGenero.xsl	Arquivo de formatação do contexto livro por gênero.
LivroPorAutor.xsl	Arquivo de formatação do contexto livro por autor.
LivroPorAno.xsl	Arquivo de formatação do contexto livro por ano.

5.3 Testes e avaliação

Para testar e avaliar a aplicação gerada pela ferramenta foi utilizado o *browser Internet Explorer 5* da *Microsoft*. O motivo da escolha do *browser* citado é devido ao fato do mesmo ser compatível com a tecnologia XML.

A tela principal da aplicação é apresentada na Figura 5.5. O menu principal é disponibilizado em um *frame* colocado à esquerda da interface. No menu principal é apresentado o título da aplicação, a descrição dos nós navegacionais modelados no esquema de contextos de navegação e, o conjunto de índices associados aos contextos definidos na modelagem navegacional.

Quando o usuário clicar sobre um determinado índice, as informações referentes ao contexto associado serão apresentadas na forma de uma tabela, na área de apresentação da aplicação, localizada no *frame* mais à direita. Cada contexto apresentado possui um título para identificação, centralizado na parte superior da área de apresentação.

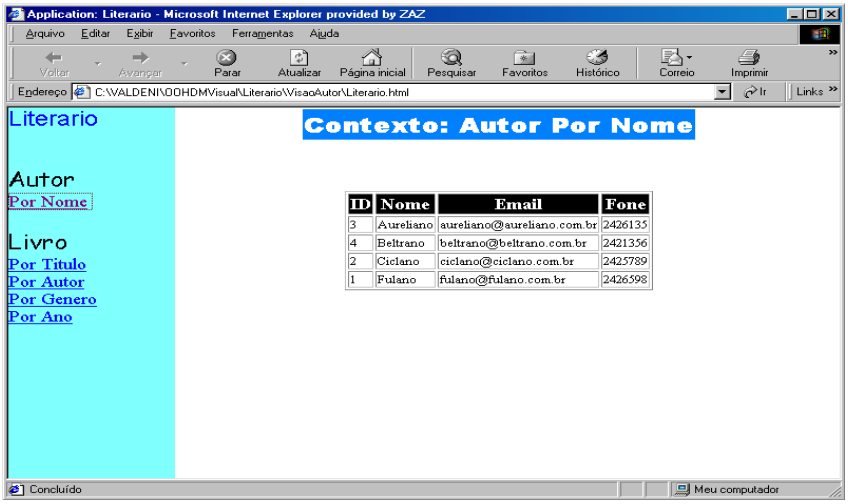


FIGURA 5.5- Tela principal da aplicação gerada pelo software.

6 Conclusões

Neste trabalho foi apresentada uma ferramenta para apoio ao processo de especificação e desenvolvimento de hiperdocumentos. Com a utilização da ferramenta é possível criar uma aplicação baseada em hiperdocumentos a partir da especificação abstrata de requisitos, através do método de projeto OOHDM. A utilização de uma abordagem de especificação de hiperdocumentos deverá se tornar uma prática usual pelos desenvolvedores, pois possibilita a organização e estruturação do processo de projeto de aplicações de forma evolutiva e natural, que vai desde a definição até a implementação da aplicação final. Através da utilização de uma ferramenta o projetista poderá validar sua aplicação antes de colocá-la em produção. A especificação de grandes aplicações sem a utilização de um método formal pode ser um empreendimento pouco produtivo, confuso e gerar um produto ineficiente.

Em estudos preliminares, pode-se observar, que em todas as abordagens de projeto de hiperdocumentos analisadas (HDM [GAR 93], OOHDM [SCH 95], RMM [IZA 95], HMT [NEM 98] e HMBS [TUR 98]) existem as seguintes preocupações:

divisão do processo de especificação de hiperdocumentos em fases distintas, onde cada fase trata um determinado problema de projeto;

em cada fase são utilizadas primitivas de modelagem que permitem a especificação de requisitos em alto nível de abstração, independente do conteúdo e da implementação;

a utilização do paradigma de orientação a objetos é amplamente discutido na grande maioria das abordagens tratadas neste trabalho. Talvez por se tratar de uma abordagem que permite um raciocínio sobre entidades do mundo real de forma natural, objeto de interesse das aplicações hipermídia, conforme cita Rossi [ROS 96];

cada modelo gerado em uma fase de projeto é enriquecido com novas primitivas em uma fase seguinte, tornando a especificação do hiperdocumento um processo incremental;

em todas as abordagens consultadas existe um grande enfoque na modelagem de estruturas de navegação. A desorientação do usuário durante a navegação de um hiperdocumento é um problema relevante quando se fala a respeito de aplicações baseadas em hipermídia. A redução da carga cognitiva de informação, bem como contextos de navegação são amplamente discutidos nas abordagens analisadas.

A utilização de ferramentas para a especificação das características de hiperdocumentos possibilita uma maior agilidade e flexibilidade no processo. Ferramentas automatizadas permitem a especificação do hiperdocumento através de primitivas gráficas de alto nível, desde as fases iniciais até a obtenção do hiperdocumento final.

Pode-se citar alguns trabalhos já realizados na área de simulação e prototipação de especificações formais de aplicações baseadas em hiperdocumentos, como o ambiente *Hyscharts* [TUR 99]. Esta ferramenta interpreta especificações de hiperdocumentos através da abordagem HMBS, validando modelos baseados em

statecharts. Já o ambiente OOHDM-Web é uma ferramenta de prototipação de aplicações hipermídia que manipulam banco de dados na Web [PON 98], gerando CGIs para a integração com banco de dados.

Com o presente trabalho, pretende-se disponibilizar uma ferramenta capaz de permitir a elaboração de modelos de apresentação de hipedocumentos, segundo o método OOHDM, e gerar várias formas de representação, através de um código aberto e portável a diversas plataformas e ambientes computacionais.

6.1 Contribuições

Com o desenvolvimento deste trabalho, pretende-se disponibilizar uma ferramenta para auxiliar o projetista de aplicações durante o processo de desenvolvimento. Como principais contribuições deste trabalho, pode-se citar:

A disponibilização de um ambiente gráfico que implementa as primitivas de um método de projeto consolidado. Permitindo, desta maneira, o controle, documentação e consistência entre as etapas de projeto, produzindo uma especificação coerente em todas as etapas do processo.

Um modelo de especificação formal das características conceituais da aplicação, exportado em um formato aberto e compatível com outro ambiente de implementação de aplicações projetadas com o método OOHDM para a *Web*, denominado OOHDM-WEB [PON 98].

Um ambiente de criação e manutenção dos dados da aplicação, que irão ser produzidos a partir de modelos de navegação definidos no ambiente de especificação gráfica da ferramenta.

Modelos de apresentação das informações, flexibilizando a forma de apresentação dos hiperdocumentos gerados a partir da ferramenta.

6.2 Trabalhos relacionados

A ferramenta visual para especificação de hiperdocumentos, segundo o método OOHDM, está fortemente relacionada com a especificação OOHDM-ML. No trabalho citado é desenvolvida uma extensão ao ambiente de implementação de hiperdocumentos denominado OOHDM-Web, que possibilita a criação de aplicações *Web* baseadas em CGIs. A ferramenta visual para especificação de hiperdocumentos gera uma descrição textual, no formato XML, dos modelos gráficos criados a partir da sua interface. As fases de especificação conceitual e navegacional são exportadas para o formato OOHDM-ML, permitindo que a especificação seja implementada em outro ambiente.

6.3 Trabalhos futuros

No *software* apresentado neste trabalho foi elaborada uma alternativa para exportação de modelos gráficos criados no ambiente de desenvolvimento. Esta alternativa é a criação de uma especificação textual, baseada em XML, de acordo com os modelos gráficos abstratos do método OOHDM.

Uma inversão do processo poderia ser interessante, onde uma opção de importação de especificações textuais em XML para o ambiente gráfico da ferramenta poderia ser implementada.

Desta maneira, não somente especificações em OOHDM-ML poderiam ser importadas, mas também especificações textuais baseadas em outros métodos de especificação de aplicações, como *workflow* por exemplo. O ambiente da ferramenta poderia ser adaptável ao método que está sendo importado. Desta forma, disponibilizando as primitivas de modelagem adequadas a cada método.

Bibliografia

[BAN 87] BANERJEE, J. et al. Data model issues for object oriented applications. In: ACM TOOIS, 5., 1987. **Proceedings...** San Antonio: ACM Press, 1987.

[BUC 92] BUCHANAN M.C.; ZELLWEGER P. Specifying Temporal Behaviour in Hypermedia Documents. In: ACM ECHT, 1992. **Proceedings...** Milano: ACM Press, 1992.

[CAR 94a] CARNEIRO, L.M.F. et al. ADVcharts: a Visual Formalism for Highly Interactive Systems. In: HARRISON, M.D.; JOHNSON, C. (Ed.). **Software Engineering in Human-Computer Interaction.** Cambridge: University Press, 1994.

[CAR 94b] CARNEIRO L.M.F. **Jasminum**: Joining ADVs and State Machines in a Notation for User-Interface Modeling. PHD Thesis, University of Waterloo, 1994.

[CAR 98] CARVALHO, M. R. **HMBS/M – An object oriented method for the design and development of hypermedia applications**. 1998. Dissertação (Mestrado em Ciência da Computação) - ICMC-USP, São Carlos.

[CAS 93] CASANOVA, M.A. et al. The Nested Context Model for Hyperdocuments. In: ACM ECHT, 1993. **Proceedings...** San Antonio: ACM Press, 1993.

[COL 92] COLEMAN, D. et al. Introducing Object-charts or how to use Statecharts in Object-Oriented Design. **IEEE Transactions on Software Engineering**, New York, v. 18, n. 1, p. 9-18, Jan. 1992.

[COW 93] COWAN, D. D. et al. Abstract Data Views Structured Programming. **IEEE Transactions on Software Engineering**, New York, v. 14, n. 1, p. 1-13, Jan. 1993.

[COW 95] COWAN, D. D. et al. Abstract Data Views: An Interface Specification Concept to Enhance Design for Reuse. **IEEE Transactions on Software Engineering**, New York, v. 21, n. 3, Mar. 1995.

[GAR 96a] GARRIDO, A.; ROSSI, G. A Framework for extending object-oriented applications with hypermedia functionality. In: RESEARCH AND DEVELOPMENT IN HYPERMEDIA AND MULTIMEDIA, 1996. **Proceedings** [S.l.;s.n.], 1996.

[GAR 96b] GARRIDO, A. et al. A Tool for extending object-oriented applications with hypermedia functionality. In: ACM ECHT, 1996. **Proceedings...** Washington: ACM Press, 1996.

[GAR 91] GARZZOTO, F. et al. HDM- A Model for the Design of Hypertext Applications. In: ACM TIS, 1991. **Proceedings...** San Antonio: ACM Press, 1991.

[GAR 93] GARZZOTO, F. et al. HDM- A Model Based Approach to Hypermedia Application Design. In: ACM TIS, 1993. [S.l.]: ACM Press, 1993.

[GAR 94] GARZZOTO, F. et al. Adding Multimedia Collections do the Dexter Model. In: ACM ECHT, 1994. **Proceedings...** Edinburgh: ACM Press, 1994.

[GAR 95] GARZZOTO, F. Hypermedia Design, Analysis and Evaluation Issues. **Communications of the ACM**, [S.l.], p. 74-86, Aug. 1995.

[HAR 93] HARDMAN, L. et al. Links in Hypermedia: the Requirements for Context. In: ACM TOOIS. **Proceedings...** [S.l.;s.n.], 1993.

[HAD 94] HARDMAN, L. et al. The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. **Communications of the ACM**, [S.l.], v. 37, n. 2, p. 50-63, Feb. 1994.

[HAR 87] HAREL, D. et al. On the formal semantics of statecharts. **IEEE Symposium on Logic in Computer Science**. New York: [s.n.], Jun. 1987.

[IZA 95] IZAKOWITZ. T. et al. RMM: A methodology for structured hypermedia design. **Communications of the ACM**, [S.l.], p. 34-44, Oct. 1995.

[JAC 92] JACOBSON, I. et al. **Object-Oriented Software Engineering**: a use case Driven Approach. [S.l.]: Addison-Wesley, 1992.

[JAC 95] JACOBSON, I. The Use-Case Construct in Object-Oriented Software Engineering. In: SCENARIO-BASED DESIGN; CARROLL, J. (Ed.). **ACM Press**. [S.l.], 1995.

[KIM 91] KIM, W. **Introduction to Object-Oriented Databases**. [S.l.]: ACM Press, 1991.

[KIM 94] KIM, W. **Modern Databases**. [S.l.]: ACM Press, 1994.

[LAN 94] LANGE, D. An Object-Oriented design method for hypermedia information systems. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCE, 1994. **Proceedings...** Hawaii: [s.n.], 1994.

[MAS 94] MASIERO, P. A reachability tree for statecharts and analysis of some properties. **Information and Software Technology**, [S.l.], v. 36, n. 10, p. 615-624, 1994.

[MED 2001] MEDEIROS, A.; SCHWABE, D. **Especificação OOHDM-ML**. Rio de Janeiro: Departamento de Informática, PUC, 2001.

[MOU 99] MOURA, I. C. R. **OOHDM-Web, manual do usuário**. Rio de Janeiro: Departamento de Informática, PUC, 1999.

[NAN 91] NANARD, J.; NANARD, M. Using Structured Types to Incorporate Knowledge in Hypertext. In: ACM TIS, 1991. **Proceedings...** San Antonio: ACM Press, 1991.

[NAN 93] NANARD, J.; NANARD, M. Should Anchors be typed too? An experiment with MacWeb. In: ACM CONFERENCE ON HYPERTEXT, 5., 1993. **Proceedings...** [S.l.]: ACM Press, 1993.

[NAN 95] NANARD, J.; NANARD, M. Hypertext Design Environments and the Hypertext Design Process. **Communications of the ACM**, [S.l.], p. 49-56, Aug 1995.

[NEM 98] NEMETZ, F. **Hypermedia Modeling Technique:** an object-oriented design model for hypermedia applications. 1998. Disponível em: http://www.cutsys.com/CHI97/Nemetz.html. Acesso em: 25 mar. 2000.

[PON 97] PONTES, R.; SCHWABE, D. **Ambiente para Desenvolvimento de Aplicações WEB**. Rio de Janeiro: Departamento de Infomática, PUC, 1997.

[ROS 96] ROSSI, G. **An Object-Oriented method for the design of hypermedia applications**. 1996. D. Sc. Thesis. Departamento de Informática, PUC, Rio de Janeiro.

[RUM 91] RUMBAUGH, J. et al. **Object Oriented Modeling and Design**, [S.l.]: Prentice Hall Inc. 1991.

[SCH 94a] SCHULER, W. A Design Space for Hypermedia Interface. In: WORKSHOP ON METHODOLOGIES FOR DESIGNING AND DEVELOPING HYPERMEDIA APPLICATIONS, 1994. **Proceedings...** Edinburgh: [s.n.], Sep. 1994.

[SCH 94b] SCHWABE, D.; ROSSI, G. From Domain Models to Hypermedia Applications: an object-oriented approach. In: INTERNATIONAL WORKSHOP ON METHODOLOGIES FOR DESIGNING AND DEVELOPING HYPERMEDIA APPLICATIONS, 1994. **Proceedings...** Edinburgh: [s.n.], Sep. 1994.

[SCH 95] SCHWABE, D.; ROSSI, G. The Object-Oriented Hypermedia Design Model. **Communications of the ACM**, [S.l.;s.n.], p. 45, Aug. 1995.

[SCH 96] SCHWABE, D.; ROSSI, G.; BARBOSA, S. Systematic Hypermedia Design with OOHDM. In: ACM INTERNATIONAL CONFERENCE ON HYPERTEXT, 1996. **Proceedings...** Washington: ACM Press, 1996.

[SHI 98] SHIBUYA, R. **Geração automática de especificações HMBS em HTML**. 1998. Dissertação (Mestrado em Ciência da Computação), ICMSC-USP, São Carlos.

[THU 91] THURING, M.; HAAKE, J. M.; HANNEMANN, J. What's ELIZA doing in the Chinese room? Incoherent hyperdocuments and how to avoid them. In: ACM TIS, 1991. **Proceedings...** New York: ACM Press, p. 161-177, 1991.

[TRI 88] TRIGG, R. Guided Tours and Tabletops: Tools for communicating in a hypertext environment. **ACM Transactions on Office Information Systems**, v. 6, n. 4, p. 398-414, Oct. 1988.

[TUR 97] TURINE, M. A. S.; OLIVEIRA, M.C.F.; MASIERO, P. C. A Navigation-oriented hypertext model based on Statecharts. In: ACM TOOIS, 1997. **Proceedings...** Southampton: ACM Press, Apr. 1997.

[TUR 98] TURINE M. A. S. **HMBS:** a statechart-based method for the formal specification of hyperdocuments. 1998. D. Sc. Thesis. IFSC-USP, São Carlos.

[TUR 99] TURINE, M. A. S.; OLIVEIRA, M. C. F.; MASIERO, P. C. **Hyscharts:** a hyperdocument authoring and browsing environment based on statecharts. In: MULTIMEDIA TOOLS AND APPLICATIONS, 1999. [S.l]: Kluwer Academic Publishers, 1999.

[ZHE 92] ZHENG, Y; PONG, M.C. Using Statecharts to Model Hypertext. In: EUROPEAN CONFERENCE ON HYPERTEXT, 1992. **Proceedings...** Milano: ACM Press, Dec. 1992.

[VIL 99] VILAIN, P; SCHWABE, D. **Notação da Metodologia OOHDM, Versão 1.1**. Rio de Janeiro: Departamento de Informática, PUC, 1999.

[W3C 98a] W3C. **Extensible Markup Language (XML) 1.0, W3C Recommendation**. 1998. Disponível em: <http://www.w3.org/TR/1998/REC-xml-19980210>. Acesso em: 15 abr. 2001.

[W3C 98b] W3C. **Extensible Stylesheet Language (XSL) 1.0, W3C Recommendation**. 1998. Disponível em: <http://www.w3.org/TR/2000/CR-xsl-20001121/>. Acesso em: 15 abr. 2001.

[W3C 99a] W3C. **XSL Transformations (XSLT) 1.0, W3C Recommendation**. 1999. Disponível em: <http://www.w3.org/TR/1999/REC-xslt-19991116>. Acesso em 01 abr. 2001.

[W3C 99b] W3C. **HTML 4.01 specification, W3C Recommendation**. 1999. Disponível em: <http://www.w3.org/TR/1999/REC-html401-19991224>. Acesso em 01 abr. 2001.