**HMBS/M – Um método orientado a objetos
para o projeto e o desenvolvimento
de aplicações hipermídia**

Marcia Regina de Carvalho

Orientação: Prof. Dr. Paulo Cesar Masiero

*Dissertação apresentada ao Instituto de Ciências Matemáticas e de
Computação - USP, como parte dos requisitos para a obtenção do título
de Mestre em Ciências - Área: Ciências de Computação e Matemática
Computacional.*

USP - São Carlos

Abril de 1998

A meus pais, José e Benedita, pelo amor e
educação que me propiciaram

AGRADECIMENTOS

A Deus pela força concedida para vencer mais este desafio.

Ao Prof. Dr. Paulo Cesar Masiero, meu orientador, pela confiança, orientação e pelos ensinamentos.

Aos meus amigos pela paciência e ajuda nas horas difíceis.

Ao CNPq pelo apoio financeiro oferecido durante o desenvolvimento deste trabalho.

Índice

INTRODUÇÃO.....	1
1.1. CONSIDERAÇÃO INICIAIS.....	1
1.2. MOTIVAÇÃO E OBJETIVOS	4
1.3. ORGANIZAÇÃO	5
REVISÃO BIBLIOGRÁFICA.....	6
2.1. CONSIDERAÇÕES INICIAIS	6
2.2. HDM - HYPERTEXT DESIGN MODEL	6
2.2.1. Conceitos	7
2.2.2. Semântica de navegação.....	9
2.3. RMD - RELATIONSHIP MANAGEMENT DESIGN	11
2.3.1. O modelo.....	11
2.3.2. O método.....	12
2.4. RMM - RELATIONSHIP MANAGEMENT METHODOLOGY	15
2.4.1. Conceitos	15
2.4.2. O Método	17
2.4.3. Extensões	20
2.5. OOHDM - OBJECT-ORIENTED HYPERMEDIA DESIGN METHOD	21
2.6. MÉTODO DE PROJETO ORIENTADO A OBJETOS	30
2.6.1. O modelo orientado a objetos e o EORM (<i>Enhanced Object-Relationship Model</i>).....	31
2.6.2. O método.....	32
2.7. AHM - AMSTERDAM HYPERMEDIA MODEL.....	34
2.7.1. O modelo de hipertexto Dexter	34
2.7.2. O modelo multimídia CMIF (<i>CWI Multimedia Interchange Format</i>).....	35
2.7.3. O modelo Amsterdam	36
2.8. HMBS - HYPERTEXT MODEL BASED ON STATECHARTS.....	39
2.8.1. O modelo.....	39
2.8.2. Extensões ao HMBS	41
2.9. COMPARAÇÃO ENTRE OS MÉTODOS.....	50
2.10. CONSIDERAÇÕES FINAIS.....	51
PROPOSTA DE UM MÉTODO PARA O PROJETO DE APLICAÇÕES HIPERMÍDIA - HMBS/M.....	52
3.1. CONSIDERAÇÕES INICIAIS	52
3.2. CARACTERIZAÇÃO DE UM DOMÍNIO DA APLICAÇÃO	52

3.3. MODELOS UTILIZADOS	55
3.3.1 <i>Modelo de objetos (MO)</i>	55
3.3.2 <i>Modelo de Fatias (MF)</i>	57
3.3.3 <i>Contextos de Navegação</i>	58
3.3.4 <i>Modelo Navegacional de Tipos (MNT)</i>	63
3.3.5 <i>Modelo Navegacional de Instâncias (MNI)</i>	66
3.3.6. <i>Canais</i>	69
3.4. O PROCESSO DO MÉTODO HMBS/M	71
3.4.1. <i>Modelagem Conceitual</i>	73
3.4.2. <i>Modelagem Navegacional</i>	75
3.4.3. <i>Modelagem da Interface, Implementação e Teste</i>	81
3.4.3.1. <i>Implementação interpretada</i>	82
3.4.3.2. <i>Implementação Traduzida</i>	87
3.4.3.3. <i>Implementação com tradução livre</i>	89
3.5. CONSIDERAÇÕES FINAIS	90
 ESTUDO DE CASO: ESPECIFICAÇÃO DE UM CATÁLOGO DE CURSOS	91
4.1. CONSIDERAÇÕES INICIAIS	91
4.2. DESCRIÇÃO DO PROBLEMA	91
04.3. AS FASES DO MÉTODO	92
4.3.1. <i>Projeto Conceitual</i>	93
4.3.2. <i>Projeto Navegacional</i>	96
4.3.3. <i>Projeto de Interface</i>	109
4.4. IMPLEMENTAÇÃO	111
4.4.1. <i>Implementação em HyScharts</i>	111
4.4.2. <i>Implementação para a WWW (World Wide Web)</i>	115
4.4.3. <i>Implementação com tradução livre</i>	119
4.5. CONSIDERAÇÕES FINAIS	122
 CONCLUSÃO	125
5.1. CONSIDERAÇÕES INICIAIS	125
5.2. CONTRIBUIÇÕES	126
5.3. TRABALHOS FUTUROS	127
 REFERÊNCIAS	128
 APÊNDICE A	131

Índice de Figuras

FIGURA 2.1	Esquema HDM da aplicação catálogo.....	8
FIGURA 2.2	Transformação de elos abstratos em elos concretos.....	11
FIGURA 2.3	Diagrama RMD da aplicação catálogo.....	13
FIGURA 2.4	Diagrama RMDM da aplicação catálogo.....	16
FIGURA 2.5	Método RMM.....	18
FIGURA 2.6	Notação gráfica o para m-slice.....	21
FIGURA 2.7	Método OOHDM.....	22
FIGURA 2.8	Esquema conceitual do OOHDM para aplicação catálogo.....	23
FIGURA 2.9	Esquema de classes navegacionais da aplicação catálogo.....	25
FIGURA 2.10	Cartão de contexto navegacional.....	26
FIGURA 2.11	ADV da classe Professor e seu respectivo ADV Charts.....	30
FIGURA 2.12	Conceitos básicos do EORM.....	32
FIGURA 2.13	Visão geral do método de projeto.....	33
FIGURA 2.14	Modelo do AHM.....	37
FIGURA 2.15	Parametrização do tipo OR.....	45
FIGURA 2.16	Parametrização do tipo OR com transições associadas.....	46
FIGURA 2.17	Abstração de transições.....	46
FIGURA 2.18	Mapeamento com âncoras e páginas.....	48
FIGURA 2.19	Especificação de um hiperdocumento em alto nível.....	48
FIGURA 3.1	Representação de classes e relacionamentos.....	56
FIGURA 3.2	Representação de agregação.....	56
FIGURA 3.3	Representação de especialização/generalização.....	57
FIGURA 3.4	Representação de especialização múltipla.....	57
FIGURA 3.5	Representação de fatias.....	58
FIGURA 3.6	Modelo de objetos.....	60
FIGURA 3.7	Mapeamento dos relacionamentos do modelo de objetos para o modelo navegacional de tipos.....	65
FIGURA 3.8	Modelo navegacional de tipos.....	66
FIGURA 3.9	Modelo navegacional de instâncias.....	67
FIGURA 3.10	Fases do método para o projeto de aplicações hipermídia.....	72

FIGURA 3.11	Mapeamento de classes.....	76
FIGURA 3.12	Mapeamento de classes divididas em fatias.....	77
FIGURA 3.13	Mapeamento de especialização/generalização.....	77
FIGURA 3.14	Mapeamento de classes com outras fatias.....	77
FIGURA 3.15	Mapeamento de classes agregadas.....	78
FIGURA 3.16	Mapeamento de índices e roteiros guiado indexados.....	78
FIGURA 3.17	Mapeamento de roteiros guiados.....	79
FIGURA 3.18	Mapeamento dos relacionamentos.....	79
FIGURA 3.19	Contexto de navegação arbitrário.....	80
FIGURA 3.20	Manutenção de contextos de navegação.....	80
FIGURA 3.21	Tela inicial do <i>HyScharts</i>	82
FIGURA 3.22	Janela para criação de estados em <i>HyScharts</i>	83
FIGURA 3.23	Janela para criação de transições compostas em <i>HyScharts</i>	84
FIGURA 3.24	Tela para criação de páginas em <i>HyScharts</i>	84
FIGURA 3.25	Exemplo de uma aplicação com $N=0$	85
FIGURA 3.26	Exemplo de uma aplicação com $N=1$	85
FIGURA 3.27	Janela para definição de canais.....	86
FIGURA 3.28	Módulo de verificação do <i>HyScharts</i>	87
FIGURA 3.29	Exemplo de uma aplicação apresentada no <i>browser</i> do <i>HyScharts</i>	88
FIGURA 3.30	Exemplo de uma aplicação em HMTL no <i>Netscape</i>	89
FIGURA 4.1	Modelo de objetos da aplicação catálogo.....	94
FIGURA 4.2	Modelo de objetos da aplicação catálogo.....	95
FIGURA 4.3	Modelo de fatias da aplicação catálogo.....	97
FIGURA 4.4	Modelo navegacional de tipos da aplicação catálogo.....	100
FIGURA 4.5	Modelo navegacional de instâncias da aplicação catálogo.....	101
FIGURA 4.6	Modelo navegacional de instâncias da classe Estrutura Organizacional.....	102
FIGURA 4.7	Modelo navegacional de instâncias da classe Laboratório.....	103
FIGURA 4.8	Modelo navegacional de instâncias da classe Disciplina.....	104
FIGURA 4.9	Modelo navegacional de instâncias da classe Curso.....	105
FIGURA 4.10	Decomposição do estado "dcc".....	105
FIGURA 4.11	Decomposição do estado "bcc".....	106
FIGURA 4.12	Modelo navegacional de instâncias da classe Corpo Docente.....	107
FIGURA 4.13	Módulo de autoria do <i>HyScharts</i>	112

FIGURA 4.14	Exemplo mostrando a situação anterior ao acionamento do botão <i>Hview</i>	113
FIGURA 4.15	Exemplo após o acionamentos do botão <i>Hview</i>	113
FIGURA 4.16	Índice do Corpo Docente no <i>browser</i> do <i>HyScharts</i>	114
FIGURA 4.17	Exemplo de uma das instâncias da classe Corpo Docente (N=0).....	114
FIGURA 4.18	Exemplo de uma das instâncias de classe Corpo Docente (N=1).....	115
FIGURA 4.19	Exemplo de um roteiro guiado indexado em <i>HyScharts</i>	116
FIGURA 4.20	Índice do Corpo Docente.....	117
FIGURA 4.21	Instância da classe Corpo Docente acessada a partir de um roteiro guiado...	118
FIGURA 4.22	Exemplo de um roteiro guiado indexado em HTML.....	119
FIGURA 4.23	Instância da classe Corpo Docente acessada a partir de um índice.....	120
FIGURA 4.24	Objeto visual para fatia Principal da classe Corpo Docente.....	120
FIGURA 4.25	Objeto visual para fatia Áreas de Pesquisa da classe Corpo Docente.....	121
FIGURA 4.26	Objeto visual para a perspectiva vídeo da fatia Formação Acadêmica da classe Corpo Docente.....	121
FIGURA 4.27	Objeto visual para a perspectiva texto da fatia Formação Acadêmica da classe Corpo Docente.....	122
FIGURA 4.28	<i>Statechart</i> para a classe Corpo Docente.....	123
FIGURA 4.29	Diagrama de configuração.....	124

Índice de tabelas

TABELA 2.1	Comparação entre os métodos e modelos para o projeto e o desenvolvimento de aplicações hipermídia.....	51
TABELA 3.1	Tipos de aplicações hipermídia X estruturação e volatilidade de suas informações.....	54
TABELA 3.2	Representação dos contextos de navegação.....	62
TABELA 3.3	Um exemplo de contexto de navegação.....	63
TABELA 3.4	Notação do modelo navegacional de tipos.....	65
TABELA 3.5	Função "m".....	67
TABELA 3.6	Informações referentes às páginas.....	68
TABELA 3.7	Função "ae".....	68
TABELA 3.8	Definição do canal texto	69
TABELA 3.9	Definição do canal imagem.....	70
TABELA 3.10	Definição do canal vídeo.....	70
TABELA 3.11	Definição do canal áudio.....	71
TABELA 3.12	Definição da função "pl".....	71
TABELA 3.13	Notação utilizada.....	76
TABELA 4.1	Contexto de navegação do tipo índice.....	97
TABELA 4.2	Contexto de navegação do tipo classe.....	98
TABELA 4.3	Contexto de navegação derivado de atributo.....	98
TABELA 4.4	Função "m" para a aplicação catálogo.....	107
TABELA 4.5	Informações das páginas (c, t, Anc _p) da aplicação catálogo.....	108
TABELA 4.6	Função "ae" para a aplicação catálogo.....	108
TABELA 4.7	Especificação do canal do tipo texto Texto 1.....	109
TABELA 4.8	Especificação do canal do tipo texto Texto 2.....	109
TABELA 4.9	Especificação do canal do tipo vídeo Vídeo 1.....	110
TABELA 4.10	Função "pl" para a aplicação catálogo.....	110
TABELA 4.11	Operações disparadas pelos eventos da figura 4.28.....	124

HMBS/M: UM MÉTODO ORIENTADO A OBJETOS PARA APOIAR O PROJETO E O DESENVOLVIMENTO DE APLICAÇÕES HIPERMÍDIA

RESUMO

Um novo método orientado a objetos para apoiar o projeto e o desenvolvimento estruturado de aplicações hipermídia, denominado HMBS/M, é proposto. O HMBS/M possui como principal característica o uso do HMBS (*Hypertext Model Based on Statecharts*) que utiliza como modelo formal subjacente a técnica de *Statecharts* para especificar a estrutura organizacional e a semântica de navegação de uma aplicação hipermídia. As quatro etapas que compõe o método - modelagem conceitual, modelagem navegacional, modelagem da interface e implementação - são apresentadas. Em cada fase do método são construídos modelos que podem ser melhorados ou incrementados na fase posterior, permitindo assim um desenvolvimento iterativo. São discutidas três formas de implementação para uma aplicação hipermídia especificada pelo HMBS/M – interpretada, traduzida e de tradução livre – com ênfase para as duas primeiras, que são implementadas usando uma ferramenta que suporta o HMBS, o sistema *HyScharts*, e usando o ambiente WWW (padrão HMTL). Para ilustrar e validar o HMBS/M apresenta-se a especificação de uma aplicação hipermídia baseada nos catálogos de Graduação e Pós-graduação do Instituto de Ciências Matemáticas e de Computação da USP.

HMBS/M: OBJECT ORIENTED METHOD TO SUPPORT DESIGN AND DEVELOPMENT OF THE HYPERMEDIA APPLICATIONS

ABSTRACT

A new object oriented method to support the design and structured development of hypermedia applications called HMBS/M is proposed. HMBS/M supports the use of HMBS (*Hypertext Model Based on Statecharts*) that utilizes as underlying formal model the *Statecharts* technique to specify the organizational structure and the browsing semantics of hypermedia applications. The four phases that comprise the method -- conceptual design, navigational design, interface design and implementation -- are presented. In each phase of the method models are built that can be improved and expanded in the next phase, allowing an interactive development. Three implementation options for a hypermedia application specified with support of HMBS/M are discussed: interpreted, translated and free translation, emphasizing the first two that are implemented using a tool that supports HMBS, the *HyScharts*, and using the WWW environment (HTML standard). A case study based in the graduate and under-graduate course catalogs of the Instituto de Ciências Matemáticas e de Computação of the Universidade de São Paulo is presented to illustrate and validate HMBS/M.

CAPÍTULO 1

INTRODUÇÃO

1.1 CONSIDERAÇÃO INICIAIS

Nos últimos anos a área das aplicações hipermídia¹ tem crescido muito rapidamente. Há uma grande quantidade de informações circulando na *Internet* provenientes de várias partes do mundo e novas áreas de aplicação que necessitam de uma rica variedade de tipos de dados multimídia estão emergindo. Tal acontecimento fez com que os sistemas hipermídia se tornassem o mais recente meio de comunicação mundial, além de aumentar o desenvolvimento, em escala comercial, das aplicações hipermídia. É importante entender que sistema hipermídia é o ambiente que facilita a criação e a transmissão de aplicações hipermídia pelo mundo. Uma breve visita à *Internet* mostra o quanto ela está sendo utilizada como meio de comunicação e transmissão de informações pelas instituições governamentais, empresas e profissionais liberais, permitindo que mais negócios sejam efetuados. Além disso, as *Intranets* surgem como novo meio para compartilhar informações de maneira fácil e barata dentro das empresas, mudando radicalmente a maneira de trabalhar dentro delas. Segundo o Forrester Research, nos Estados Unidos, 16% das empresas já tem *Intranet*, e 50% a terão brevemente, enquanto que no Brasil essa tecnologia começa a ser cada vez mais utilizadas pelas empresas [Aug97]. Conceitualmente falando, a *Intranet* é uma rede corporativa que utiliza os padrões da *Internet* para disponibilizar as informações, sendo essas informações acessíveis somente à própria empresa a que pertencem e às pessoas autorizadas por ela a consultá-las. Sendo assim, a tecnologia hipermídia surge como um meio que permite organizar as informações seguindo a mesma estrutura não linear utilizada pela mente humana para organizá-las e recordá-las. Essa organização é realizada utilizando-se

¹ Neste trabalho o termo aplicação hipermídia corresponde a outros termos encontrados na literatura como hiperdocumento e hipertexto.

de maneira simples e natural os conceitos de multimídia e hipertexto. Enquanto a multimídia fornece poderosos tipos de dados que facilitam a flexibilidade em expressar a informação, o hipertexto provê uma estrutura de controle que suporta uma maneira elegante de navegar através dessa estrutura.

Segundo Garzotto [Gar93], o maior desafio encontrado pelos autores de aplicações hipermídia é capturar e organizar assuntos complexos de maneira adequada, facilitando posteriormente a sua manutenção. Desse modo, uma abordagem sistemática para modelar o conteúdo e definir a organização estrutural é especialmente importante no projeto de aplicações hipermídia grandes e complexas. Nesse caso, o uso de um modelo, ou de um método, ajuda a disciplinar a atividade de autoria, para que [Ros96]:

- seja possível descrever a aplicação independentemente de sua implementação. Desta forma, a definição de um esquema conceitual ajuda a oferecer uma visão mais abstrata do domínio da aplicação do que aquela obtida pela inspeção do seu código, ou pela tentativa de extrair uma semântica das estruturas dos nós e dos elos.
- a evolução da aplicação seja mais facilmente fornecida, desde que as decisões de projetos sejam documentadas no nível correto de abstração.

Garzotto também afirma que uma abordagem estruturada para o desenvolvimento de tais aplicações sugere a noção de *authoring-in-the-large*, que permite a descrição de classes gerais de informação e estruturas navegacionais e de *authoring-in-the-small* que se refere ao desenvolvimento do conteúdo dos nós [Gar91]. Essa visão é baseada na opinião de que as decisões sistemáticas e a estrutura racional da aplicação devem ser tomadas antes dela ser implementada, a fim de gerar uma aplicação coerente e expressiva. Uma parte importante do projeto de aplicações hipermídia lida com aspectos estéticos e cognitivos que não são apoiados pelos atuais ambientes de engenharia de software. Segundo Nanard [Nan95], deve-se considerar o projeto de aplicações hipermídia partindo de dois pontos de vista ou dimensões: um que prescreve técnicas de projeto formais, e outro que lida com a sequência real de estados e ações mentais que ocorrem quando se segue uma técnica de projeto. A atividade mental do projetista sempre compreende uma das seguintes categorias: geração de novo material, estruturação do material e visualização dessa estrutura, atualização, observação e avaliação. Desse modo, um ambiente de projeto de aplicações hipermídia deve apoiar: um ciclo de retroalimentação experimental que facilite a atividade de avaliação e o processo de recursão, a capacidade de usar no modelo os conceitos de generalização e especialização, promovendo atividades de estruturação e atualização e ferramentas de duplicação (*cloning*) fáceis de usar que trabalhem

com exemplos, apoiando a atividade de geração de material.

Segundo Thüring, existem dois tipos de aplicações hipermídia no que se refere ao relacionamento entre estas e o conhecimento [Thü95]. O primeiro tipo encoraja aqueles que querem navegar por grandes volumes de informação, obtendo informação ao longo do caminho. Este tipo é mais apropriado para suportar buscas irrestritas e recuperação de informação. O segundo tipo está diretamente relacionado a problemas e soluções específicas, e é totalmente estruturado e restrito. Este tipo é mais adequado para tarefas que requerem um profundo entendimento e aprendizagem. Desse modo, uma aplicação hipermídia bem projetada deve facilitar a sua própria compreensão, sendo totalmente coerente e evitando a desorientação do leitor. No entanto, tudo o que se refere ao conhecimento e ao processamento humano de informações ainda é muito negligenciado e raramente influencia o projeto de aplicações hipermídia.

Devido às peculiaridades de uma aplicação hipermídia, tais como o papel das ligações, a semântica de navegação, a necessidade de gerenciar um grande volume de informações, a utilização de recursos multimídia que aumentam a complexidade, o envolvimento no projeto de pessoas com diferentes habilidades e formações (escritores, bibliotecários, artistas, músicos e programadores) e a necessidade de prototipagem e teste intensivo com os usuários, seu desenvolvimento difere dos processos tradicionais de desenvolvimento de software. Um dos grandes problemas do projeto de aplicações hipermídia se refere à combinação da navegação controlada pelo usuário com a própria natureza dos dados multimídia. Desse modo, surge a necessidade de um modelo para o projeto de aplicações hipermídia que apresente as seguintes vantagens: fornecer uma linguagem na qual o projetista pode especificar a aplicação, permitindo uma melhor comunicação entre projetistas e usuários; ser utilizado para apoiar a documentação e a manutenção da aplicação; fornecer um *framework* segundo o qual os autores podem desenvolver, analisar e comparar métodos e estilos de autoria; fornecer uma linguagem de modelagem facilitando, se possível, a reutilização parcial da estrutura básica da aplicação; e, descrever precisamente a estrutura e a dinâmica de uma aplicação hipermídia, permitindo realizar uma análise sistemática da aplicação [Gar91, Gar93, Sch92].

Dentre as abordagens existentes, algumas tentam explicitamente modelar a semântica de domínios específicos, adotando estruturas de representação pré-definidas, como o g-IBIS [Gar93]. Outras, como o Dexter Hypertext Reference Model [Gar93], tentam definir de uma maneira não ambígua e rigorosa algumas abstrações importantes encontradas em grande parte dos sistemas existentes. Existem abordagens menos formais que enfatizam estruturas

topológicas como blocos de construção para criar a estrutura da aplicação, como o KMS[Gar93] e o Guide [Gar93]. Outras fornecem modelos que ajudam os projetistas a gerar as estruturas da aplicação mais facilmente e sistematicamente, permitindo aos autores criar múltiplas cópias das estruturas individuais, todas compartilhando certas propriedade; como exemplo pode-se citar o Hypercard [Gar93]. Finalmente, existem aquelas que ajudam a organizar e modularizar o material existente, como o Trigg's Guided Tours e o Tabletops [Gar93]. Outra abordagem bem diferente foi observada em Trellis [Ros94] no qual a estrutura navegacional de uma aplicação hipermídia é descrita utilizando uma Rede de Petri. Finalmente, Zheng modela a estrutura de navegação e de interface da aplicação utilizando *Statecharts* [Zhe92].

Vários modelos formais e métodos para modelar o domínio e descrever o projeto, a organização estrutural e a semântica de navegação de aplicações hipermídia têm sido propostos na literatura, como o RMM - Relationship Management Methodology [Isa95, Dia95], o OOHDM - Object Oriented Hypermedia Design Model [Bar95, Ros96, Ros94, Sch96, Sch94a, Sch94b], o AHM - Amsterdam Hypermedia Model [Har93, Har94] e o EORM - Enhanced Object-Relationship Model [Lan93, Lan94]. Entretanto, tais modelos ainda estão no seu início, sendo que muitos deles não são eficientes e expressivos, dificultando a sua utilização.

1.2. MOTIVAÇÃO E OBJETIVOS

Nos próximo anos a tendência nas empresas será utilizar cada vez mais os recursos da *Internet* interligada às suas *Intranets*. No entanto, as técnicas existentes na indústria tradicional para o desenvolvimento de sistemas não satisfazem os requisitos das aplicações hipermídia, motivando a pesquisa de novos modelos e métodos que forneçam diretrizes para gerenciar de maneira sistemática o projeto e o desenvolvimento desse tipo de aplicação, aumentando a qualidade do projeto e reduzindo o seu custo. Outra motivação para o desenvolvimento de novos métodos e modelos é fornecer diretrizes aos gerentes de projeto para produzir documentação padronizada e dar suporte automatizado aos desenvolvedores. Desse modo, o desenvolvimento de novos métodos para o projeto de aplicações hipermídia é um campo de estudos que crescerá rapidamente nos próximos anos. No entanto, as pesquisas sobre tais métodos ainda são poucas e estão em sua infância, tornando evidente a necessidade de estudos adicionais que permitam a evolução desses métodos de acordo com a necessidade dos seus usuários.

Neste trabalho é apresentado o método orientado a objetos para apoiar o projeto e o desenvolvimento estruturado de aplicações hipermídia denominado HMBS/M. O HMBS/M

utiliza algumas primitivas do método orientado a objetos Fusion [Col94] para representar o domínio da aplicação e o HMBS (*Hypertext Model Based on Statecharts*) [Oli97, Tur97] para a modelagem da navegação na aplicação. O método é composto por quatro etapas: modelagem conceitual, modelagem navegacional, modelagem da interface e implementação. Em cada fase do método são construídos modelos que podem ser melhorados ou incrementados na fase posterior, permitindo assim um desenvolvimento iterativo. O método também pretende permitir a prototipação dos modelos obtido no fim de cada fase para facilitar a verificação dos requisitos.

1.3. ORGANIZAÇÃO

Neste capítulo foram apresentados o estado atual do problema a ser estudado, as motivações e os objetivos deste trabalho. O capítulo dois descreve alguns métodos e modelos para o projeto de aplicações e sistemas hipermídia relevantes para este trabalho, definindo seus principais conceitos e discutindo seus pontos fortes e fracos. No final desse capítulo é apresentada uma comparação entre os métodos, considerando os seus aspectos mais relevantes.

O capítulo três apresenta uma caracterização dos principais domínios das aplicações hipermídia, os modelos utilizados pelo HMBS/M, o processo utilizado por ele para o desenvolvimento de aplicações hipermídia e algumas opções de implementação para esse tipo de aplicação. O capítulo quatro apresenta o estudo de caso utilizado para demonstrar e analisar o HMBS/M. O capítulo cinco apresenta as conclusões e contribuições deste trabalho. Finalmente, é apresentado no apêndice A a definição formal do modelo XHMBS.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

2.1 - CONSIDERAÇÕES INICIAIS

Embora existam muitas pesquisas sobre hipermídia, são poucas as que se referem a métodos para projeto e desenvolvimento de aplicações hipermídia. A necessidade de um método para o projeto e desenvolvimento de aplicações hipermídia está relacionada ao fato de que o projeto deve ser estruturado, deve permitir o reuso e o rastreamento das decisões tomadas, e deve facilitar a construção de ferramentas automatizadas.

Segundo Balasubramanian [Bel94], um bom método de projeto deve ser eficiente para se usar e deve produzir uma aplicação hipermídia coerente, que ajude o usuário a obter a informação desejada. Para que isso seja possível, o método de projeto deve focalizar as informações do domínio da aplicação, as conexões entre elas e como representá-las, manipulá-las e armazená-las. Deve, também, considerar a dinâmica da interação do usuário com a aplicação.

Desse modo, surge a necessidade de métodos mais completos, que utilizem modelos mais gerais para reproduzir todas as características especiais de todos os tipos de aplicação. A seguir são apresentados os métodos estudados, bem como uma comparação/classificação entre eles, terminando com as considerações finais.

2.2 - HDM - HYPERTEXT DESIGN MODEL

O desenvolvimento do HDM [Gar91, Gar93] se deu como parte do projeto *Hytea*, cujo objetivo era produzir um conjunto de ferramentas de autoria para o desenvolvimento de aplicações hipermídia. O HDM é um modelo que propõe a especificação de um esquema da aplicação, no qual se descrevem as classes gerais de informação em termos de sua apresentação, sua estrutura interna de organização e de suas interconexões. Sendo assim, um esquema captura a semântica e as regularidades organizacionais na representação das estruturas de uma

dada classe de aplicações. A partir desse esquema o modelo permite definir uma particular aplicação, fornecendo primitivas para descrever uma instância do esquema, isto é, um conjunto particular de classes de informação e suas conexões. A diferença entre um esquema e uma instância do esquema permite que, algumas vezes, o mesmo esquema seja utilizado por diferentes aplicações no mesmo domínio. Essas aplicações compartilham a mesma estrutura global e diferem nas instâncias particulares em que são apresentadas.

Um esquema HDM é mostrado na figura 2.1, no qual os retângulos com bordas arredondadas representam as entidades e seus componentes, como é o caso de professor, disciplina e evento, e as setas representam os relacionamentos. Nesse exemplo não foram representados os atributos e as perspectivas, para facilitar o entendimento da aplicação catálogo, que serve de veículo para exemplificar os vários métodos.

O HDM pode ser usado de diferentes maneiras. Como um dispositivo de modelagem ele permite a produção de especificações em alto nível de aplicações existentes ou a serem desenvolvidas; como dispositivo de implementação ele é a base para o projeto de ferramentas que apoiam o desenvolvimento dessas aplicações. No entanto, o HDM é utilizado principalmente como dispositivo de modelagem, pois além de fornecer maneiras para descrever aplicações hipermídia de modo conciso e independente da plataforma de implementação, ele ajuda o autor a criar um modelo conceitual de uma dada aplicação sem muita preocupação com os detalhes de implementação.

2.2.1 - CONCEITOS

O HDM tem várias peculiaridades que o torna diferente de outros modelos, mas ao mesmo tempo ele empresta dos modelos existentes na área de modelagem de dados e de hipertextos muitos dos detalhes de suas características.

Uma aplicação HDM consiste de estruturas de informação chamadas entidades, que representam algum objeto do mundo real do domínio da aplicação. Uma entidade é o menor pedaço de informação autônoma que pode ser introduzido ou removido da aplicação e pode ser decomposta em uma hierarquia de componentes, que serão posteriormente transformados em unidades. As estruturas de informação podem ser interconectadas por diferentes tipos de elos, bem como por estruturas de acesso (*outlines*) e uma semântica de navegação *default*.

As entidades são agrupadas em tipos de entidades, que correspondem às classes de objetos do mundo real. Todas as entidades pertencentes ao mesmo tipo têm certas características

em comum: o nome de seu tipo de entidade; o conjunto de perspectivas sob as quais seu conteúdo é apresentado e os tipos de elos da aplicação que saem ou chegam a ela.

Figura 2.1 - Esquema HDM da aplicação catálogo

Em aplicações hipermídia é frequentemente necessário que um mesmo tópico seja apresentado de maneiras alternativas. A noção de diferentes apresentações para o mesmo conteúdo é capturada no HDM pelo conceito de perspectiva. O conjunto de perspectivas é compartilhado por todas as entidades de um dado tipo. Sendo assim, se um dado tipo de entidade possui duas perspectivas, todas as componentes pertencentes àquele tipo também têm duas possíveis perspectivas.

Uma unidade corresponde a um componente associado a uma perspectiva específica e é caracterizada por um nome e por um corpo (texto, imagem, som, etc). Em HDM, diferentes unidades podem compartilhar um mesmo corpo, mas isto não é muito utilizado, pois pode provocar desorientação no usuário.

Elos em hipertexto são estruturas que interconectam pedaços de informação, e que possuem uma regra de representação e uma de navegação. A primeira regra captura as relações existentes no domínio da aplicação, e a segunda captura os padrões de navegação. No entanto, o principal sentido de um elo é tornar explícito para o leitor um relacionamento do domínio da

aplicação. O HDM propõe três tipos de elos: elos estruturais, elos de perspectiva e elos da aplicação. A distinção entre tais elos simplifica o trabalho dos projetistas da aplicação, induz a um uso mais consistente dos elos, cria padrões de navegação mais previsíveis e torna possível introduzir características adicionais, tais como derivação de elos e a definição de uma semântica de navegação *default*. Os elos de perspectiva conectam as diferentes perspectivas do mesmo componente; são navegacionalmente muito simples, já que ao ativar um elo desse tipo não há mudança de contexto de informação. Os elos estruturais conectam componentes pertencentes a uma mesma entidade e correspondem a relacionamentos induzidos pela estrutura de árvore da entidade. Os elos estruturais são um pouco mais complexos que os elos de perspectiva mas ainda deixam o leitor dentro do mesmo contexto de informação. Finalmente, os elos da aplicação representam os relacionamentos entre entidades do domínio da aplicação. Um tipo de elo da aplicação é especificado por um nome, um conjunto de tipos de entidades origem e destino, e um atributo de simetria, que define se cada elo de um determinado tipo possui um elo inverso. Do ponto de vista navegacional, esse tipo de elo é o mais incômodo, pois atravessar um elo desse tipo implica em uma mudança de contexto.

De acordo com a terminologia do HDM, uma aplicação hipermídia pode ser dividida em duas porções: uma hiperbase e um conjunto de estruturas de acesso. A hiperbase consiste de todos os elementos definidos anteriormente, enquanto as estruturas de acesso permitem ao leitor selecionar pontos de entrada para a navegação futura. Os *outlines* são um tipo de estrutura de acesso formado por componentes ordenados segundo uma árvore. Cada componente é conectado aos seus pais, a seus filhos e a seus irmãos via elos estruturais.

No HDM pode-se deixar implícita a definição de um número significativo de elos, sendo esses induzidos a partir de propriedades estruturais do modelo. Todos os elos implícitos ou deriváveis podem ser gerados automaticamente a partir da especificação HDM, usando o esquema da aplicação. Os elos de perspectiva podem ser automaticamente derivados a partir da definição dos componentes e unidades. Já os elos estruturais e da aplicação podem ser automaticamente computados a partir de elos básicos, pela execução das correspondentes cláusulas de derivação.

2.2.2 SEMÂNTICA DE NAVEGAÇÃO

O uso atual de hipertextos é definido por sua semântica de navegação, que determina quais objetos de informação e elos entre esses objetos serão percebidos pelo leitor, e qual deve

ser o comportamento quando os elos são ativados. O HDM não prescreve, a priori, qualquer semântica de navegação específica, nem inclui qualquer primitiva de alto nível para definição dessa semântica. Entretanto, ele define uma semântica de navegação padrão, compatível com as estruturas nós-elos encontradas na maioria dos hipertextos.

A semântica de navegação padrão do HDM assume que somente as unidades podem ser percebidas pelo leitor como nós, e que somente um nó é ativado de cada vez. Desse modo, somente elos entre unidades são percebidos. Como no HDM somente os elos de perspectiva conectam unidades, os elos estruturais e da aplicação devem ser transformados em tais elos. Sendo assim, os elos estruturais e da aplicação são chamados de abstratos, enquanto os elos de perspectiva são chamados de concretos.

As regras do HDM para transformar elos abstratos em elos concretos são baseadas na idéia de ter uma representação padrão para cada objeto abstrato. A representação padrão para um componente é sua unidade sob a perspectiva padrão, e para uma entidade é a representação padrão do seu componente raiz. A transformação de elos abstratos em elos concretos é feita da seguinte maneira: cada elo da aplicação é transformado em um conjunto de elos concretos, conectando cada unidade do componente origem à representação padrão do componente destino. Já os elos estruturais seguem a seguinte regra: se um componente C_1 tem um elo estrutural para um componente C_2 , então, para cada perspectiva P , cada unidade de C_1 que tem esta perspectiva é ligada à unidade de C_2 com a mesma perspectiva. Na figura 2.2 é mostrado um exemplo da transformação de elos abstratos em elos concretos.

Em hipertexto, conexões entre pedaços de informação são frequentemente percebidas por meio de âncoras. Âncoras são elementos gráficos da interface que mostram a existência de conexão, e podem ser selecionadas pelo leitor para ir a outro nó. No HDM a semântica de navegação padrão tem a noção de ponto-de-escolha para âncoras que se referem a vários possíveis nós destinos. A estrutura é associada a uma âncora e permite selecionar via menus um dos múltiplos destinos da âncora.

O uso do HDM resultou em melhorias significativas na qualidade e velocidade do processo de desenvolvimento de aplicações hipermedia [Gar93]. Em particular, tem sido especialmente útil para aplicações nas quais regularidade, organização, modularização e consistência são fatores importantes.

A introdução de novos conceitos, tais como perspectiva, diferentes tipos de elos, distinção entre hiperbase e estruturas de acesso, e a possibilidade de integrar mais facilmente a

estrutura de uma aplicação hipermídia com sua semântica de navegação caracterizam grandes vantagens na utilização do HDM. A derivação automática de elos a partir da descrição do projeto conceitual é uma das vantagens mais marcantes. Entretanto, uma análise mais cuidadosa do modelo mostra alguns pontos que podem ser melhorados, tais como: introdução de mecanismos de acesso mais sofisticados, agregação de entidades, compartilhamento de informações e utilização de mídia ativa.

Figura 2.2 - Transformação de elos abstratos em elos concretos

2.3 - RMD - RELATIONSHIP MANAGEMENT DESIGN

O RMD [Bel94] é, de certa forma, uma expansão do modelo de dados do HDM com o objetivo de refletir a natureza dinâmica dos elos. As peculiaridades inerentes às aplicações hipermídia resultam em certos problemas a serem resolvidos durante o projeto. Tais problemas envolvem decidir como dividir o domínio de informações em pedaços, como os pedaços resultantes serão ligados e como o usuário deve interagir com o sistema. Desse modo, os projetistas de aplicações hipermídia devem se concentrar em duas tarefas principais. A primeira delas considera a definição das classes gerais de informação e das estruturas de navegação, enquanto a segunda leva em conta a instanciiação dessas classes. No entanto, tais tarefas devem ser realizadas de maneira consistente e eficiente, a fim de gerar um bom projeto final.

2.3.1 - O MODELO

A estrutura básica do RMD é um modelo Entidade-Relacionamento (E-R), em que o diagrama E-R modela as informações sobre a estrutura do domínio da aplicação, mas não especifica todos os mecanismos de navegação e de acesso.

O RMD utiliza alguns conceitos definidos no HDM. Como tal, as entidades são definidas

como estruturas hierárquicas de componentes, nas quais o nó raiz representa o maior nível da entidade e cada componente tem um relacionamento é-parte-de com seus pais. O nó raiz e seus componentes têm perspectivas associadas a eles, e uma unidade é a menor quantidade de informação à qual o usuário tem acesso, correspondendo a um nó na rede nós-elos.

Existem dois tipos de conexões no modelo. As conexões estruturais definem a estrutura de um tipo de entidade ligando seus componentes e suas perspectivas. Já as conexões aplicativas descrevem relacionamentos entre os tipos de entidades, conectando os componentes pertencentes a diferentes tipos de entidade.

Como estrutura de acesso, os índices são mecanismos que permitem ao usuário escolher o nó destino. Já os roteiros são listas ordenadas de unidades de informação criadas pelo autor. Além dessas, o RMD utiliza índices condicionais, roteiros condicionais, roteiros condicionais indexados e grupos. Os índices condicionais correspondem a índices para um subconjunto de instâncias que satisfaçam a um determinado predicado. Os roteiros condicionais especificam uma sequência de nós que satisfaçam a um determinado predicado, enquanto os roteiros condicionais indexados são similares a estes e possuem um índice em cada nó da sequência. Finalmente, os grupos são similares a menus e fornecem um meio para acessar outras estruturas de acesso. Um diagrama RMD é mostrado na figura 2.3.

2.3.2 - O MÉTODO

O método apresentado é um procedimento sistemático para o projeto e a construção de aplicações hipermídia, já que esse é um processo que consome muito tempo e, freqüentemente, requer a ligação de muitos pedaços de informação residentes em diferentes mídias e conectadas de modo arbitrário e complexo. O método é composto dos seguintes passos:

Projeto Entidade-Relacionamento

Durante este passo os aspectos do domínio que serão apoiados pela aplicação são representados por um diagrama E-R. O diagrama E-R não indica a maneira como o usuário deve navegar através do espaço de informação, mas mostra as entidades e os relacionamentos entre elas, de modo a ajudar a determinar as entidades e os elos aplicativos a serem apresentados na aplicação.

Projeto de Entidade

Neste estágio descreve-se como o conteúdo das entidades do domínio será dividido em unidades de informação. Isto resulta em um diagrama de entidades que representa os relacionamentos internos das entidades da aplicação.

Figura 2.3 - Diagrama RMD da aplicação catálogo

Projeto Navegacional

Durante este estágio o diagrama RMD é obtido pela composição do diagrama E-R, com o diagrama de entidade substituindo os relacionamentos aplicativos por índices condicionais.

Projeto do protocolo de conversão

Neste passo especifica-se como o diagrama RMD é transformado em uma aplicação

hipermídia na plataforma destino. Para isso utilizam-se regras de conversão que determinam a implementação dos relacionamentos e das estruturas de acesso em nós e elos da *web*. Algumas dessas regras são apresentadas a seguir: 1) relacionamentos um-para-um resultam na criação de dois elos indicando a bidireção; 2) relacionamentos um-para-muitos resultam na criação de um nó intermediário, geralmente um índice; 3) índices resultam na criação de um nó especial com elos para a raiz de cada nó; 4) roteiros resultam na criação de um nó introdutório e a partir desse nó há um elo para o 1º nó, do 1º para o 2º e assim sucessivamente; 5) grupos resultam em um nó separado com elos para os nós introdutórios dos índices e roteiros.

A estrutura do RMD também suporta elos derivados automaticamente. A regra geral para os elos derivados é propagar os relacionamentos ao longo dos componentes estruturalmente relacionados. Desse modo, cada nó é ligado somente aos seus pais e a seus irmãos.

Projeto da interface do usuário

Neste estágio é realizado o projeto do esquema das telas de todos os objetos do modelo RMD. Isto envolve o esquema dos botões, a aparência dos nós e índices e a localização dos botões de navegação.

Projeto do comportamento em tempo de execução

Este estágio envolve o projeto do comportamento da aplicação, especificando a semântica para atravessar um elo e como obter a informação a ser apresentada. Sendo assim, os projetistas devem determinar a natureza estática ou dinâmica dos elos e nós.

Construção e Teste

Durante este estágio a aplicação hipermídia é construída e os nós e elos do modelo de dados da aplicação são preenchidos com os dados do domínio, seguindo o protocolo de conversão e o projeto do comportamento em tempo de execução.

O método determina algumas regras para estabelecer, manter e usar os relacionamentos suportados por ele, especialmente os relacionamentos entre aplicações independentes. A implementação de relacionamentos um-para-muitos e muitos-para-muitos pode ser determinada de três maneiras: 1) estática - os índices são inseridos na aplicação pelo desenvolvedor e não podem ser mudados; 2) dinâmica (semi-interoperável) - os índices são criados dinamicamente mas o banco de dados utilizado é interno ao ambiente de desenvolvimento; e 3) dinâmica (totalmente-interoperável) - os índices são criados em tempo de execução acessando informações

de bancos de dados externos.

A utilização do método RMD facilita o processo de desenvolvimento e ajuda a produzir um projeto mais completo e consistente. No entanto, este ainda não trata de mídias ativas e nem do compartilhamento de informações.

2.4 - RMM - RELATIONSHIP MANAGEMENT METHODOLOGY

O RMM é uma expansão do modelo RMD - *Relationship Management Design* proposto por Balasubramanian, Isakowitz e Sthor e serve como base para o projeto e o desenvolvimento de aplicações hipermídia [Isa95]. Os autores o consideram mais útil para aplicações que tenham uma estrutura regular e com freqüente necessidade de atualizações das informações. Dado que muitas aplicações hipermídia necessitam atualizar suas informações freqüentemente, torna-se necessário algum meio para automatizar e tornar rotineiro o desenvolvimento inicial e o processo de atualização dessas aplicações.

O RMM utiliza o RMDM - *Relationship Management Data Model* como modelo de dados para expressar o projeto de aplicações hipermídia. Um modelo de dados é um conjunto de objetos lógicos usados para fornecer uma abstração de uma porção do mundo real. Como tal, o RMDM fornece uma linguagem para descrever os objetos de informação e os mecanismos de navegação em aplicações hipermídia. Entretanto, é necessário deixar claro que o RMM é aplicado ao desenvolvimento de aplicações hipermídia, e não ao desenvolvimento de sistemas hipermídia, que são os ambiente que facilitam a criação de tais aplicações. Na figura 2.4 é apresentado um diagrama RMDM para a aplicação catálogo.

2.4.1 - CONCEITOS

O modelo de dados RMDM é a base para o RMM e utiliza algumas primitivas que são mostradas a seguir.

Primitivas do domínio

As primitivas do domínio modelam a informação relativa ao domínio da aplicação. Os tipos de entidade e seus atributos representam os objetos físicos ou abstratos do mundo real. Os relacionamentos associativos um-para-muitos ou um-para-um descrevem as associações entre os diferentes tipos de entidades. Utilizando os princípios de banco de dados, os relacionamentos um-para-muitos são fatorados em dois relacionamentos um-para-um. Como muitas entidades

apresentam um grande número de atributos e pode ser impraticável ou indesejável apresentá-los todos juntos, o RMDM utiliza a noção de *slice* (fatia) para agrupar atributos de uma mesma entidade.

Figura 2.4 - Diagrama RMDM da aplicação catálogo

Primitivas de acesso

As primitivas de acesso permitem a navegação no RMDM que, no geral, possui seis primitivas de acesso. Os elos uni e bidirecionais são usados para especificar o acesso entre *fatias* de uma entidade e não ultrapassam as fronteiras desta. A navegação por diferentes entidades pode ser especificada por índices, roteiros e grupos. Os índices agem como uma lista de instâncias de entidades, fornecendo acesso direto a cada item da lista. Os roteiros implementam um caminho linear por uma coleção de itens, permitindo ao usuário mover-se para frente e para trás. Os grupos são mecanismos, como menus, que permitem o acesso a outras partes do documento hipermídia. As condições ou predicados lógicos podem ser utilizadas juntamente

com os índices e os roteiros para determinar quais instâncias de uma entidade podem ser acessadas a partir daquele construtor. Como último mecanismo tem-se os roteiros indexados condicionais que combinam índices e roteiros para fornecer uma estrutura de acesso mais poderosa.

2.4.2 - O MÉTODO

O método RMM é mostrado graficamente na figura 2.5, no contexto de um ciclo completo de desenvolvimento de software. O foco principal do método é o projeto dos mecanismos de acesso, conduzido pelos seguintes passos: projeto E-R, projeto de entidade e projeto navegacional.

Projeto E-R

No primeiro passo do projeto representa-se o domínio de informação da aplicação por um diagrama Entidade-Relacionamento (E-R). Neste estágio o processo de projeto representa o estudo das entidades e relacionamentos relevantes do domínio da aplicação. Essas entidades e relacionamentos são a base para a aplicação hipermídia e muitos deles aparecerão na aplicação final como nós e elos. O objetivo é utilizar um diagrama E-R para analisar o domínio da aplicação e ajudar a identificar relacionamentos para apoio à navegação.

Projeto de Fatias

Este passo é único para aplicações hipermídia e determina como a informação numa entidade escolhida será apresentada ao usuário e como eles devem acessá-la. Tal projeto envolve dividir uma entidade em fatias (*slices*) significativas e organizá-las em uma rede hipertexto. Cada fatia agrupa um ou mais atributos de uma entidade e isso é representado em um diagrama de fatias. Cada entidade tem uma fatia principal que é usada como *default* para os elos que chegam na entidade.

O diagrama de entidades também modela a navegação entre fatias via elos uni e bidirecionais. Os elos que representam conexões entre fatias são chamados elos estruturais e se diferenciam dos relacionamentos associativos, uma vez que eles conectam pedaços de informação dentro da mesma instância de entidade, enquanto os relacionamentos associativos interconectam diferentes instâncias de entidades pertencentes a diferentes classes de entidades.

Do ponto de vista navegacional há uma razão importante para diferenciar esses dois tipos

de conexão, pois ao atravessar um elo estrutural o contexto da informação não muda, o que não acontece ao atravessar um relacionamento associativo.

Figura 2.5 - Método RMM

No fim deste passo é obtido o diagrama E-R₊, que é o diagrama E-R acrescido de cada entidade com o seu diagrama de fatias. É importante levar em consideração a divisão de uma

entidade em um número adequado de fatias; a escolha de uma fatia para ser a principal; a conexão correta dessas fatias e o rótulo dos elos dessas conexões.

Projeto Navegacional

Durante o projeto navegacional são projetados os caminhos que permitem a navegação no hipertexto. Cada relacionamento associativo que aparece no diagrama E-R+ é analisado de acordo com os requisitos da aplicação, para então ser substituído por uma ou mais das estruturas de acesso citadas anteriormente. Como o método RMM é mais aplicável a domínios que requerem atualizações freqüentes na base de dados, todos os caminhos navegacionais são especificados de forma genérica, referindo-se às propriedades das entidades e dos relacionamentos. Para isso, utilizam-se índices, roteiros, e roteiros indexados com condições.

Durante este passo os projetistas devem identificar os componentes de informação e os relacionamentos que serão acessados; os grupos que serão apresentados; e as estruturas de acesso apresentadas em cada caso. É importante salientar que as decisões sobre os componentes, relacionamentos e grupos devem ser baseadas nos requisitos do sistema obtidos na fase de análise. Sobre as estruturas de acesso, os grupos fornecem pontos de entrada hierárquicos; no entanto, sempre que possível, tenta-se reduzir a profundidade das hierarquias para evitar desorientação do usuário. Nos relacionamentos um-para-muitos a escolha é mais complexa. Geralmente utiliza-se um roteiro quando há poucas instâncias participando ou quando não há chaves para o índice. Por outro lado, quando há um grande número de instâncias a melhor opção é um índice. Os roteiros indexados são freqüentemente usados quando há uma boa chave para o índice ou alguma navegação local. No fim deste passo o diagrama ER+ é transformado em um diagrama RMDM.

Projeto do protocolo de conversão

Durante este passo utiliza-se um conjunto de regras de conversão para transformar cada elemento do diagrama RMDM em um objeto na plataforma destino. Este passo é atualmente realizado manualmente por programadores.

Projeto da interface do usuário

Este passo envolve o projeto dos leiautes das telas para cada objeto que aparece no diagrama RMDM.

Projeto do comportamento em tempo de execução

Durante este passo as decisões de projeto sobre como atravessar um elo, história, regresso e mecanismos de navegação são implementadas. Além disso, os desenvolvedores devem também considerar a volatilidade e o tamanho do domínio de informações para decidir se o conteúdo dos nós e elos é construído durante o desenvolvimento da aplicação ou dinamicamente em tempo de execução.

Construção

Este passo consiste na construção e teste da aplicação como nos projetos tradicionais de engenharia de software.

2.4.3. EXTENSÕES

Com base nas diversos projetos de aplicações hipermídia realizados com o RMM foram identificadas certas limitações que prejudicam a eficiência do método [Isa97b]. Entre as principais limitações estão:

- a inabilidade para especificar qual informação é para ser apresentada como conteúdo de uma âncora, isto é, a origem para o texto ou imagem que aparece como hiperlink em unidade de apresentação;
- fato de que as entidades só podem agregar atributos;
- fato de que um *slice* não pode conter atributos ou estruturas de acesso.

Desse modo, o RMM introduziu um novo tipo de *slice* chamado *m-slice* [Isa97a] e um novo conceito chamado de diagrama de aplicação [Isa97b]. O *m-slice* é um construtor usado para agrupar informações em unidades significativas de informações. Cada *m-slice* possui uma classe proprietária específica a fim de ser considerado como um elemento daquela classe. Desse modo, um *m-slice* pode ser reusado tantas vezes quanto necessário, por ele próprio ou como parte de um outro *m-slice*, sem a necessidade de redefini-lo. *M-slices* encapsulam informações de várias origens, tais como: atributos da classe proprietária, atributos de classes relacionadas e estruturas de acesso. Além disso, eles podem ser agregados e aninhados para formar *m-slices* de mais alto nível. A notação gráfica dos *m-slices* é apresentada na figura 2.6, na qual os elementos que aparecem dentro dos limites são os que constituem os elementos do *m-slices*.

É importante salientar que *m-slices* descrevem somente qual informação faz parte de um construtor e onde obtê-la. Os *m-slices* fornecem o poder necessário para o RMM representar

arbitrariamente unidades de informação complexas, suportando uma abordagem estruturada, reusável, gerenciável e programável para o projeto e o desenvolvimento de aplicações hipermídia.

Figura 2.6 – Notação gráfica para o m-slice

O diagrama da aplicação é um novo conceito introduzido pelo RMM e tem como objetivo fornecer uma visão global da aplicação hipermídia. O diagrama da aplicação pode ser usado na negociação sobre o projeto da aplicação com os usuários e outras partes interessadas. Ele também pode ser usado para encontrar erros e omissões e para refinar a aplicação. O diagrama da aplicação pode ser construído seguindo os seguintes passos:

- examine todos os m-slices usados na aplicação e anote todos os *hyperlinks* que estão ancorados nesses m-slices. Coloque todos esses m-slices no diagrama da aplicação;
- identifique todas as interações entre os m-slices colocados no passo anterior. Examine cada um desses elos e os coloque no diagrama da aplicação representados por setas. Examine se há m-slices aninhados e coloque os elos encontrados nesses m-slices no diagrama da aplicação;
- finalmente, coloque no diagrama da aplicação os elos que existem entre as entidades.

O método RMM apresenta como vantagens a seqüência de passos para o projeto, uma maior quantidade de representações gráficas e estruturas de acesso mais poderosas. No entanto, não é totalmente aplicável a alguns tipos de aplicações hipermídia, restringindo, assim, a sua utilização de modo mais geral.

2.5 - OOHDM - OBJECT-ORIENTED HYPERMEDIA DESIGN METHOD

O OOHDM - *Object-oriented hypermedia design method* é uma extensão do HDM que apresenta uma natureza orientada a objetos e inclui primitivas especiais de modelagem para o projeto navegacional e de interface [Sch96]. A utilização de modelos orientados a objetos

permite a descrição de uma aplicação hipermídia usando construtores de alto nível e a formulação de projetos complexos com uma notação concisa e expressiva.

O OOHDM é composto de quatro atividades, como mostra a figura 2.7: projeto conceitual, projeto navegacional, projeto de interface abstrata e implementação. Tais atividades são realizadas num estilo de desenvolvimento incremental, interativo e baseado em protótipo, sendo que durante cada uma delas, exceto na última, um conjunto de modelos orientados a objetos descrevendo um particular projeto é construído ou enriquecido a partir do anterior. Cartões são utilizados para especificar e documentar as classes, objetos, sub-sistemas e contextos navegacionais [Sch96].

Passos	Produtos	Mecanismos	Interesses do Projeto
	Classes, sub-sistemas atributos, relacionamentos perspectivas, objetos	Classificação, composição, generalização, especialização.	Modela a semântica do domínio da aplicação
Projeto navegacional	Nós, elos, estruturas de acesso, contextos e transformações navegacionais.	Mapeamento entre objetos conceituais e navegacionais.	Modela o perfil e as tarefas do usuário. Ênfase nos aspectos cognitivos.
Projeto da interface abstrata	Objetos da interface abstrata, respostas a eventos externos transformações da interface	Mapeamento entre objetos de navegação e perceptíveis.	Modela os objetos perceptíveis, implementação da metáfora escolhida
Implementação	Aplicação executando.	Aqueles fornecidos pelo ambiente destinos	Aspectos de integridade e de desempenho

Figura 2.7 - Método OOHDM

Projeto Conceitual

Um modelo conceitual do domínio da aplicação é construído durante esta atividade, usando princípios de modelagem orientada a objetos e algumas primitivas adicionais. A principal preocupação nesta fase é capturar a semântica do domínio de forma tão neutra quanto possível.

O esquema conceitual consiste de um conjunto de objetos e classes conectadas por relacionamentos; os objetos são instâncias das classes, e as classes podem estar relacionadas a subsistemas (um esquema hipermídia completo que é usado como parte de outro esquema

perspectivas originarão uma classe de elos hipermídia não especificados explicitamente como relacionamentos no esquema, correspondente à conexão das diferentes perspectivas do mesmo atributo.

Dois construtores de abstração são fornecidos para lidar com complexidade: agregação e o par generalização/especialização. O primeiro descreve classes complexas como agregação de classes mais simples e o segundo constrói hierarquias de classes usando herança como mecanismo de compartilhamento.

Projeto Navegacional

Nesta fase a informação apresentada no modelo conceitual é reorganizada a fim de se ter uma aplicação que possa ser usada por um conjunto de usuários tentando realizar um conjunto de tarefas. O projeto navegacional é representado por dois esquemas, o esquema de classes navegacionais e o esquema de contexto de navegação.

O OOHDM utiliza um conjunto de tipos de classes navegacionais pré-definidas, tais como nós, elos e estruturas de acesso. Os nós são os recipientes básicos de informação em aplicações hipermídia e suas estruturas dependem da semântica da aplicação. Uma classe de nó é caracterizada especificando-se a(s) classe(s) conceitual(ais) a partir da qual ela é derivada, seus atributos e as âncoras que ela contém. As classes do esquema conceitual que possuem atributos com múltiplas perspectivas devem ser transformadas em nós com atributos diferentes ou em nós diferentes, um para cada perspectiva. As âncoras são consideradas como uma classe primitiva, cujo parâmetro é o tipo de elo que está associado ao nó. O esquema de classes navegacionais da aplicação catálogo é mostrado na figura 2.9.

Os elos implementam os relacionamentos definidos no esquema conceitual. As classes de elo são definidas pela especificação de seus atributos, de seu comportamento, dos objetos origem e destino e da cardinalidade. Um elo pode ser definido com um-para-um, que requer múltiplas âncoras (uma para cada instância) ou um-para-muitos, que requer um ponto-de-escolha ou índice como objeto intermediário. Quando um elo é utilizado para representar o relacionamento entre os membros de uma composição ou agregação, ele é frequentemente chamado de elo estrutural, como no HDM. Por outro lado, elos definidos entre duas classes de nós derivadas do mesmo objeto conceitual, mas com diferentes tipos para o mesmo atributo, são equivalentes aos elos de perspectiva no HDM.

As estruturas de acesso agem como índices ou dicionários e são úteis para ajudar o usuário final a encontrar a informação desejada. As estruturas de acesso também são modeladas

como classes e caracterizadas por um conjunto de seletores, um conjunto de objetos destino e um predicado. Elas podem ser acessadas a partir dos nós, usando âncoras específicas, ou podem elas próprias fazer parte de um nó de composição.

Figura 2.9 - Esquema de classes navegacionais da aplicação catálogo

A estrutura navegacional de uma aplicação é definida no OOHDM em termos de contextos navegacionais. Um contexto navegacional é definido como uma estrutura de dado que compreende nós, elos, classes de contexto e outros contextos de navegação. Para definir um

contexto navegacional deve-se identificar os pedaços de informação do domínio que estão relacionados segundo algum critério, a fim de garantir a coerência do material apresentado. Os contextos devem ter atributos, usados para representar a informação sobre o próprio contexto; um ou mais pontos de entrada; um caminho conectando seus elementos; uma especificação do comportamento; uma especificação de quais elementos são visíveis em contextos aninhados, e se eles devem estar relacionados a outros contextos navegacionais. A especificação do caminho descreve como os elementos do contexto são conectados ao se navegar dentro do contexto; e a especificação do comportamento descreve o fluxo de controle durante a navegação. Os contextos navegacionais são especificados por intermédio de cartões como o da figura 2.10.

Nome do contexto navegacional	
Atributos { incluindo a meta-informação)	
Objetos, Classes e outros contextos navegacionais inclusos	
Contextos Navegacionais/Classes/Objetos	Relacionamentos
Relacionado a	
Pontos de Entrada	
Caminho	
Comportamento	
Importa/Exporta	
Comentários	

Figura 2.10 - Cartão de contexto navegacional

Há alguns tipos de contextos navegacionais que são induzidos a partir dos vários elementos do esquema navegacional de classes. O contexto de navegação arbitrário é o tipo mais geral de contexto. Seus elementos são definidos de uma maneira arbitrária pelo projetista, que os escolhe segundo algum critério. Um exemplo típico de tais contextos são os roteiros guiados (*Guided Tours*). O contexto de navegação de classe é induzido pela definição de classe. Ele é formado pela coleção de todos os nós que pertencem a uma determinada classe. O contexto de navegação derivado de elos é induzido pela presença de elos um-para-muitos gerados por um elo conceitual. O contexto de navegação derivado de estrutura coleciona todos os nós que pertencem ao mesmo objeto de composição; os elos nesse contexto permitem navegação estrutural dentro da composição. O contexto de navegação de índice é derivado da estrutura índice e contém os elementos apontados pelo índice. Este tipo de contexto navegacional tem um comportamento padrão que permite navegar do índice para qualquer

elemento, e de qualquer elemento de volta para o índice. Finalmente, o contexto de navegação de sessão é construído dinamicamente, e seus elementos são adicionados pela avaliação de alguma condição em tempo de execução [Bar95].

O OOHDM utiliza a noção de classes de contexto para completar uma classe navegacional com a adição de atributos, dependendo do contexto. Desse modo um objeto de contexto adiciona atributos ao objeto navegacional que somente são visíveis dentro de um ou mais contextos. A definição de uma classe de contexto inclui a definição de seu escopo – o conjunto de contextos navegacionais em que ela é válida, e a definição de seus participantes – o conjunto de objetos naquele contexto que ela modificará.

Como em todo projeto de aplicação hipermídia, alguns aspectos do comportamento dinâmico da aplicação devem ser especificados na fase de projeto navegacional. Tais aspectos definem como o espaço navegacional muda durante a navegação e quais ações de controle o usuário possui. Como citado anteriormente, a especificação de um contexto navegacional contém a descrição do caminho e do comportamento das classes que pertencem a esse contexto. Segundo a semântica de navegação do OOHDM, o caminho pode ser de três tipos: sequencial, ramificado e condicional. O caminho sequencial é o tipo mais simples, no qual os elementos do contexto são organizados em seqüência. A especificação deve incluir uma ordem de acordo com alguma condição nos atributos dos elementos, ou uma ordem arbitrária. O caminho ramificado estabelece um caminho hierárquico conectando os elementos no contexto. Este tipo de organização é modelada usando contextos aninhados. Finalmente, o caminho condicional inclui pontos de ramificação nos quais condições são avaliadas e o resultado determina o caminho a ser seguido. Caminhos condicionais incluem caminhos procedurais, em que os caminhos devem ser uma parte de outro caminho; caminhos variáveis, em que o próximo elo a ser seguido é computado dinamicamente; e caminhos paralelos, em que muitos elos devem ser seguidos simultaneamente, possivelmente com alguma sincronização entre eles.

O comportamento de um contexto de navegação é definido segundo dois aspectos: controle e restrição. O controle de navegação dentro do contexto pode ser automático ou passo a passo. Quando o controle é automático, o usuário é guiado automaticamente pela aplicação; quando o controle é passo a passo o usuário controla quando o próximo passo navegacional será tomado. O controle passo a passo é subdividido em acesso direto e sequencial. No primeiro os elos do contexto definem um índice de contexto de navegação que permite ao usuário escolher diretamente o nó que deseja visitar. No segundo, os elos de contexto definem um caminho sequencial a ser percorrido pelo usuário [Bar95].

A navegação em um contexto de navegação pode ser limitada segundo três restrições de navegação: navegação livre, exclusiva ou parcial. A navegação de acesso direto livre permite ao usuário acessar diretamente qualquer elemento daquele contexto de navegação. Caso a navegação seja de acesso direto exclusivo, o índice não inclui todos os elementos daquele contexto, e somente os elementos contidos no índice são acessíveis diretamente. Se a navegação de acesso direto for parcial, o contexto de navegação importa elos de outro contexto de navegação. Para a navegação seqüencial, caso esta seja livre, o usuário pode percorrer qualquer elo de cada elemento do contexto de navegação. Caso a navegação seqüencial seja exclusiva, o usuário só pode percorrer elos do contexto. Finalmente, para a navegação seqüencial parcial, o usuário pode percorrer elos importados de outros contextos de navegação [Bar95].

Há uma outra dimensão que caracteriza o estilo passo a passo de navegação. Dois estilos básicos são relatados na literatura, os mais comuns são o baseado em grafo, encontrado em muitas aplicações hipermídia, e o baseado em conjunto.

Projeto de interface abstrata

A fim de especificar o modelo de interface abstrata de uma aplicação hipermídia é necessário definir a aparência da interface e descrever suas propriedades estáticas e dinâmicas e seus relacionamentos com o modelo navegacional. Para isso, é preciso especificar [Ros94]:

- a aparência na interface de cada objeto navegacional visto pelo usuário. O mesmo objeto navegacional pode ter diferentes representações na interface em diferentes situações,
- outros objetos de interface para fornecer várias funções da aplicação, tais como barras de menu e botões de controle,
- os relacionamentos entre objetos da interface e navegacionais, tal como a maneira pela qual um evento externo afetará a navegação,
- as transformações da interface que ocorrem devido ao efeito da navegação ou de um evento externo no comportamento dos diferentes objetos da interface,
- a sincronização de alguns objetos da interface, principalmente quando mídias dinâmicas estão envolvidas.

O OOHDM utiliza ADVs (*Abstract Data Views*) para descrever a interface com o usuário de uma aplicação hipermídia. Os ADVs especificam a maneira como os elementos da interface são estruturados usando agregação e generalização/especialização como mecanismos de abstração, e a maneira como os elementos da interface são estaticamente relacionados aos

objetos de navegação, utilizando Diagramas de Configuração para expressar tais relacionamentos. Além disso, eles mostram, por intermédio de *ADV-Charts*, como os elementos da interface comportam-se quando reagem a eventos externos, em particular como eles acionam a navegação e quais transformações da interface ocorrem quando o usuário interage com a aplicação. A figura 2.11 mostra o ADV e seu respectivo *ADV-Charts* para a classe “Professor” da aplicação catálogo.

A definição da maneira como cada objeto navegacional será percebido pelo usuário final geralmente é realizada em hipermídia e em outras aplicações interativas pela definição da aparência da interface. Isso implica em determinar a estrutura geral da interface da aplicação hipermídia e em especificar os objetos da interface que estarão disponíveis. Sendo assim, para cada classe navegacional necessita-se definir seu ADV correspondente. No caso de nós, a estrutura do ADV deve ser facilmente determinada pela estrutura do mesmo. No caso dos elos não se especificam os aspectos da interface quando estes são um-para-um porque a sua presença será indicada nos ADVs correspondentes pelas âncoras. Entretanto, pode-se definir aspectos perceptíveis de elos um-para-muitos por intermédio de ações.

As estruturas de acesso devem ser percebidas como um menu, um conjunto de ícones ou outras representações mais complexas. Entretanto, em todos os casos elas são especificadas como uma coleção de ADVs, cujos proprietários são os nós destino.

Em geral, o comportamento da aplicação é completamente especificado, definindo-se a maneira como eventos externos afetam a navegação e a interface da aplicação. Isso é realizado pela semântica de navegação que especifica o “estado interno” da aplicação e como ela muda durante o processo de navegação. No entanto, deve-se mostrar o efeito de cada evento externo em termos de transformações que ocorrem na interface, usando-se *ADV Charts* para mostrar os possíveis estados e correspondentes transições de cada ADV, a fim de entender o comportamento dos componentes individuais da interface ao reagir a eventos externos.

Implementação

Durante esta atividade o projetista deve mapear os modelos navegacional e de interface abstrata em objetos concretos utilizados no ambiente de implementação escolhido.

A orientação a objetos permite ao OOHDM utilizar mecanismos que fornecem alto nível de abstração e composição com uma semântica bem definida, além de permitir o reuso de tais mecanismos. O uso de contextos navegacionais ajuda a reduzir o problema de desorientação do

usuário e a minimizar a carga cognitiva imposta pela navegação através de hiperdocumentos complexos. Além disso, o OOHDM é um método que estrutura o seu processo de projeto de maneira clara, o que torna fácil o seu uso.

Figura 2.11 - ADV da classe professor (a) e seu respectivo *ADV Charts* (b)

2.6 - MÉTODO DE PROJETO ORIENTADO A OBJETOS

O método de projeto orientado a objetos proposto por Lange [Lan93,Lan94] tenta integrar a funcionalidade de hipertextos com os sistemas de informação, utilizando o método OMT proposto por *Rumbaugh* e o Modelo Objeto-Relacionamento Estendido. Ele baseia-se em um processo de modelagem que captura a estrutura, o comportamento e a interação dos objetos do domínio da aplicação, apresentando as seguintes vantagens: fornece um construtor para capturar

a semântica complexa da interação dos objetos que caracteriza as aplicações hipermídia; traz algumas das vantagens das abstrações orientadas a objetos para o projeto de aplicações hipermídia; simplifica a integração de aplicações hipermídia com sistemas de informação desenvolvidos com orientação a objetos e suporta ferramentas e construtores de interfaces gráficas.

2.6.1 - O MODELO ORIENTADO A OBJETOS E O EORM (*ENHANCED OBJECT-RELATIONSHIP MODEL*)

O modelo orientado a objetos fornece aos analistas de sistemas e projetistas a habilidade para expressar um domínio da aplicação. Os objetos são usados para modelar entidades do mundo real e possuem uma semântica estrutural e comportamental. Dois mecanismos de estruturação são usados na abordagem orientada a objetos: classificação e generalização. O primeiro permite que os objetos com a mesma semântica estrutural e comportamental sejam reunidos em classes para modelar conjuntos de entidades homogêneas. O segundo representa situações nas quais classes são combinadas para formar novas classes que definem o conjunto de semântica estrutural e comportamental compartilhadas. Entretanto, o modelo orientado a objetos não suporta os relacionamentos tão bem quanto o modelo de dados relacional. Em um modelo orientado a objetos uma relação binária e bidirecional entre dois objetos é representada pela definição de um atributo que aponta para o outro objeto participante da relação, mas essa representação falha em capturar a semântica da relação. Para solucionar tal falha, o modelo objeto-relacionamento foi refinado para desenvolver um modelo estendido baseado na integração do modelo orientado a objetos e na semântica entre objetos, tendo como idéia central representar explicitamente a semântica das relações no modelo orientado a objetos. A semântica das relações é definida como uma classe, permitindo que as semânticas estrutural e navegacional sejam mantidas em um único lugar. As principais vantagens dessa representação são: relações podem participar de outras relações; as relações tornam-se semanticamente ricas e bibliotecas reusáveis podem ser criadas para representar a semântica. No armazenamento, as relações entre duas classes são representadas de forma explícita por elos entre os membros dessas classes, permitindo que os elos representem como objeto os estados não triviais e o comportamento das relações, participem de outros elos e sejam tratados como objetos. A figura 2.12 mostra tal representação.

2.6.2 - O MÉTODO

O método de projeto orientado a objetos fornece um meio para capturar a semântica de integração dos objetos em um sistema orientado a objetos. O método consiste de três componentes: a estrutura de classe, a estrutura de composição e a estrutura de Interface Gráfica do Usuário, mostrados na figura 2.13.

Figura 2.12 - Conceitos básicos do EORM

A estrutura de classe consiste de uma biblioteca reusável de definições de classes e possui duas atividades relacionadas a ela: identificação e refinamento de classes. A primeira atividade consiste em identificar as classes relevantes a partir do domínio do problema. Classes irrelevantes, não específicas e redundantes, bem como relacionamentos, atributos e operações devem ser evitados nesta fase. A segunda atividade consiste no detalhamento da informação de cada classe pela definição de seus atributos, operações e relacionamentos de herança. Nessa atividade os atributos são adicionados representando as propriedades da classe, sendo que atributos derivados, identificadores, objetos e detalhes são considerados desnecessários e devem ser evitados. As operações também são adicionadas durante essa atividade e definem os aspectos funcionais da classe, isto é, o comportamento local da classe, excluindo a interação do comportamento que é capturada pela composição. Finalmente, deve ocorrer a introdução de herança, que representa situações nas quais classes distintas são combinadas para formar uma nova classe virtual. Ela pode ser considerada como uma generalização de classes existentes em

novas super-classes ou como uma especialização de classes existentes em novas sub-classes.

Figura 2.13 - Visão geral do método de projeto

A estrutura de composição consiste de uma biblioteca reusável de definições da classe elo e possui duas atividades relacionadas a ela: identificação e refinamento da composição. Na primeira atividade identificam-se as composições de classes dentro do domínio da aplicação. A definição de uma composição inclui o relacionamento, as classes participantes do relacionamento e suas cardinalidades, e as semânticas do relacionamento, sendo que os relacionamentos derivados e irrelevantes devem ser evitados. Na segunda atividade refinam-se as composições pela definição dos domínios e cardinalidades dos participantes, dos invariantes (restrições) que permitem determinar a integridade dos dados e pela associação das semânticas da classe elo para cada composição. Os invariantes podem ser classificados como invariantes relacionados a elos individuais ou invariantes relacionados à composição como um todo. A definição das semânticas da classe elo envolve considerar para o relacionamento em cada composição as seguintes questões: o tipo do relacionamento (estrutura representacional ou caminho de acesso relacionado à interface do usuário); uso de direção; frequência de uso; propriedades estáticas/dinâmicas; densidade de população e comportamento. Algumas questões extras, como compartilhamento e herança, devem ser consideradas com respeito ao uso de elos.

A estrutura da interface gráfica (GUI) consiste de uma biblioteca reusável de definições de apresentação e possui duas atividades relacionadas: apresentação e identificação da janela e

mapeamento de classes e composições em apresentações. Uma apresentação consiste de um ou mais campos em que um objeto pode mostrar seu conteúdo. Uma janela pode mostrar uma apresentação e pode também conter menus que são usados para associar o comportamento à janela, fazendo-a parecer e agir como um programa independente da aplicação. A primeira atividade consiste em determinar as janelas, isto é, aplicações e navegadores do domínio e em determinar qual apresentação deve aparecer em cada janela. A segunda atividade deve considerar como atributos e operações das classes são mapeados em apresentações, como os eventos são fixados às operações e como a funcionalidade é associada aos menus das janelas.

O método proposto é mais geral e expressivo que alguns métodos existentes e por ser um processo iterativo permite que se volte aos estágios anteriores para corrigir e refinar o projeto.

2.7 - AHM - AMSTERDAM HYPERMEDIA MODEL

O AHM é uma extensão do *Dexter Hypertext Reference Model* (DHM) que utiliza o modelo *CWI Multimedia Interchange Format* (CMIF) para modelagem de dados multimídia. Segundo Hardman, Bulterman, e Rossum [Har94], o AHM fornece uma estrutura geral que pode ser usada para descrever construtores básicos e ações que são comuns em uma grande variedade de sistemas hipermídia. Desse modo, o AHM adiciona ao Dexter a noção de tempo, de apresentação de alto nível e de elo de contexto, permitindo a criação de seqüências multimídia identificáveis, especificação do estilo de apresentação para multimídia, composição de seqüências multimídia existentes em apresentações maiores, separação da informação estrutural da informação de apresentação e a especificação de contextos. A seguir é apresentada uma breve revisão dos modelos que compõe o AHM.

2.7.1 - O MODELO DE HIPERTEXTO DEXTER

O modelo Dexter [Hal94] separa um sistema hipertexto em três camadas com interfaces bem definidas entre elas. A camada de armazenamento contém os objetos persistentes e armazenáveis que formam o hipertexto. A camada interna do componente preocupa-se com o conteúdo e a estrutura dentro dos componentes da rede hipertexto. A camada em tempo de execução é responsável por manipular elos, âncoras e componentes em tempo de execução.

A camada de armazenamento descreve a estrutura do hipertexto como um conjunto finito de componentes, juntamente com duas funções: *resolver* e *acessor*, responsáveis pela obtenção

de componentes. A unidade básica da camada de armazenamento é um componente atômico, ou um elo ou uma composição formada de outros componentes. O componente atômico é uma primitiva, similar a um nó em sistemas hipertexto; elo é um componente que representa relacionamentos entre outros componentes; e composição é um componente construído a partir de outros componentes. Todo componente tem uma identidade única, dada por seu identificador (UID).

A interface entre a camada de armazenamento e a camada interna do componente utiliza uma entidade chamada âncora, que permite o endereçamento de localizações ou itens dentro do conteúdo de um particular componente. A interface entre a camada de armazenamento e a de tempo de execução utiliza o mecanismo de especificação da apresentação, que codifica as informações sobre como um componente deve ser apresentado ao usuário em uma rede hipertexto na camada de armazenamento.

A camada interna do componente é tratada no modelo Dexter como estando fora do modelo em si. É assumido que outros modelos para projetar a estrutura das aplicações, ou documentos ou tipo de dados, sejam utilizados em conjunção com o Dexter.

A camada de tempo de execução inclui o conceito de instanciação de um componente, que é a apresentação do componente para o usuário, e uma entidade sessão, que serve para manter, de tempos em tempos, o mapeamento entre os componentes e suas instanciações.

O modelo apresenta como limitações a falta de noção de tempo, nenhuma forma para especificar as informações da apresentação em alto nível e nenhuma noção de contexto para âncoras.

2.7.2 - O MODELO MULTIMÍDIA CMIF (*CWI MULTIMEDIA INTERCHANGE FORMAT*)

O CMIF é um modelo para representar e manipular documentos multimídia. Ele concentra-se na organização da informação baseada em tempo e na construção de documentos maiores a partir de documentos menores. Os objetos de mídia no modelo possuem conteúdo e restrições de sincronização. Além disso, o modelo inclui blocos de dados, nos quais os dados de um agente atômico são armazenados; canais como dispositivos de saída abstratos para mostrar eventos; arcos de sincronização para especificar as restrições de tempo entre os blocos de dados; descritores de dados para descrever a semântica do bloco de dados; e descritores de eventos para descrever a apresentação de uma instância do bloco de dados.

O modelo oferece duas maneiras para visualizar um documento que está sendo escrito: a visão hierárquica, que mostra o documento como uma coleção estruturada de blocos de dados a

ser mostrada em série ou em paralelo, e a visão em canal, que mostra blocos de dados mapeados para canais. A principal limitação do modelo é que ele não inclui elos [Har93].

2.7.3 - O MODELO AMSTERDAM

O AHM utiliza, como o Dexter, a noção de componente atômico para armazenar a meta-informação que se refere a um particular bloco de dados, enquanto utiliza o componente composição para definir a informação para uma coleção de blocos atômicos ou composições. Desse modo, o modelo localiza a informação em tempo e a apresentação de um componente atômico; e concentra a informação do componente composição na sua estrutura de apresentação. Além disso, o modelo promove a reusabilidade de dados e modela mais estritamente a maneira como o volume de informações multimídia será armazenado, utilizando banco de dados externos ou sistemas de arquivo.

Como extensão ao Dexter, o AHM utiliza um bloco atômico na sessão de especificação da apresentação de um componente atômico para modelar aspectos relacionados ao tempo e codificar, em alto nível, atributos da apresentação. Já o componente composição é usado, no AHM, para construir a estrutura da apresentação ao invés de simplesmente colecionar componentes relacionados para o propósito da navegação. Uma das diferenças em relação ao Dexter é a presença de arcos de sincronização como estruturas que definem a ordem da informação na especificação da apresentação dos componentes de composição. Além disso, a especificação dos componentes de uma composição foi expandida para incluir o tempo de compensação entre os sub-componentes da composição e o seu tipo. O tipo de uma composição pode ser paralelo, no qual todos os seus componentes são mostrados; ou escolha, no qual a composição mostra pelo menos um de seus sub-componentes. Os dois componentes são mostrados na figura 2.14.

Um modelo hipermídia deve permitir agrupamento de itens dentro da apresentação e a expressão de restrições de tempo entre esses itens. Ao agrupar componentes dentro de uma apresentação, as restrições de tempo entre os sub-componentes da composição precisam ser especificadas, e isso pode ser feito com respeito ao componente pai ou com respeito a outro componente irmão. Sendo assim, uma pequena mudança é necessária no modelo Dexter, que é a remoção do conteúdo de um componente de composição, restringindo a presença deste somente aos componentes atômicos. Com isso, as informações de tempo para cada sub-componente são dadas pela relação de sincronização com o pai ou com outro componente irmão.

Especificação da apresentação	Nome de canal
	Duração - especificada ou implícita
	Outros componentes - informação da especificação da apresentação
Atributos	Semântica da informação
Âncoras	ID da âncora
	Valor
Conteúdo	Bloco de dados ou ponteiros para dados

Especificação da apresentação	Componentes - informação da especificação da apresentação
	De - ID do componente
	Para - ID do componente
Arcos de sincronização	Relação de tempo
Atributos	Semântica da informação
Âncoras	ID da âncora
	Lista de (ID do componente, ID da âncora)
Sub-componentes	ID do componente
	Hora de início

Figura 2.14 - Modelo do AHM

O modelo utiliza canais para definir um estilo de apresentação *default* para componentes atômicos que são mostrados via aquele canal. Os canais são definidos globalmente e referenciados por cada componente atômico. A utilização de canais permite à apresentação ser

dependente da estrutura e, desse modo, o mesmo documento pode ser apresentado em diferentes formas pela especificação de diferentes estilos de apresentação globais. Um uso típico de canais é combinar algum deles dentro de um leiaute reusável.

Além disso, o modelo permite que a informação requerida seja uma combinação de apresentações menores dentro de apresentações maiores. Ao combinar componentes existentes, o componente de composição resultante deve ser apresentado usando recursos disponíveis. No entanto, isso pode ser impossível devido ao uso de canais incompatíveis ou o uso excessivo de canais individuais.

No AHM, a estrutura hierárquica e as informações de tempo são expressas por componentes de composição. A estrutura não hierárquica da informação é expressa através de elos e âncoras e a informação de tempo não hierárquica é expressa por arcos de sincronização armazenados no ancestral comum dos itens relacionados. As relações baseadas em tempo devem ser tratadas como uma informação da apresentação e mantidas separadas da estrutura de informação baseada em elos. As restrições de tempo podem ser impostas entre dois descendentes de um componente composição, entretanto, é mais claro distinguir entre as restrições de tempo definidas entre os sub-componentes de um componente de composição, e aquelas entre quaisquer dois descendentes irmãos do componente de composição.

O AHM utiliza contextos para especificar se uma nova apresentação é mostrada ou se parte da apresentação atual é substituída quando um leitor segue um elo. Os contextos origem ou destino para um elo são definidos como a parte da estrutura do documento que é afetada quando o elo de (ou para) uma âncora é seguido. Um benefício da especificação de contexto é que somente parte do documento mostrado na tela precisa ser afetado quando o elo é seguido. Os nós de nível mais alto permanecem, enquanto os de nível mais baixo são substituídos. Para cada âncora associada a um elo é necessário especificar se esta é origem ou destino do elo, uma definição da âncora e o contexto - parte da apresentação que será afetada quando o elo é seguido.

As principais vantagens do AHM são sua habilidade para descrever relações temporais, bem como a utilização de composição e o uso de canais para a especificação da apresentação de um componente. No entanto, o fato do modelo não tratar da estrutura interna dos componentes limita o seu uso no processo de projeto de aplicações hipermídia.

2.8 - HMBS - HYPERTEXT MODEL BASED ON STATECHARTS

Statecharts constituem um formalismo visual para descrever estados e transições em um padrão modular. O formalismo de *statechart* estende o formalismo clássico de máquinas de estados finitos e diagramas de transições de estado pela incorporação de notações para hierarquia, concorrência, mecanismo de difusão (*broadcasting*) para comunicação entre componentes concorrentes, composição, agregação e refinamento de estados [Hae87].

O HMBS utiliza a técnica de *Statecharts* como modelo de especificação formal adjacente para especificar a organização estrutural e a semântica de navegação de um hipertexto [Oli97]. O modelo também permite a descrição de aspectos de concorrência de um hipertexto, além de ser bastante intuitivo. A estrutura lógica utilizada pelo modelo fornece um mecanismo que permite separar a estrutura organizacional das representações físicas ou conteúdo do hiperdocumento.

2.8.1 - O MODELO

No HMBS, um hiperdocumento H é definido como uma 7-tupla $H = \langle ST, P, m, L, pl, ae, N \rangle$ na qual:

- $ST = \langle S, \rho, \psi, \gamma, \delta, V, C, E, A, R, T \rangle$ representa o *Statechart* que especifica a estrutura organizacional e navegacional subjacente ao hiperdocumento;
- $P = \{P_1, P_2, \dots, P_n\}$, $n > 0$, é o conjunto finito de páginas de informação que define o conteúdo do hiperdocumento. Cada página é definida conceitualmente pela tripla $\langle c, t, Anc_p \rangle$, na qual "c" é o conteúdo da página, "t" é o título da página e "Anc_p" é uma coleção de âncoras associadas à página. O conjunto de página pode incluir um página nula especial P_λ que pode ser associada a estados que não modelam a apresentação da informação;
- $m: S_s \rightarrow P$ é uma função que mapeia os estados "s" de um subconjunto S_s a páginas $p \in P$ do hiperdocumento;
- $L = \{l_1, l_2, \dots, l_k\}$ $k > 0$, define o conjunto de canais de apresentação pelos quais é possível especificar os requisitos relativos à apresentação da informação contida nas páginas do hiperdocumento;
- $pl: P \rightarrow L$ é uma função de visualização que associa cada página $p \in P$ a um único canal $l \in L$ capaz de interpretá-la;

- $ae: Anc_p \rightarrow E$ é uma função que associa as âncoras pertencentes a uma página P a eventos do *Statechart*, que por sua vez definem as transições que serão disparadas;
- N ($N \geq 0$) define o nível de visibilidade ou de navegação do hiperdocumento.

O HMBS possui na sua definição três tipos de objetos: estruturais, navegacionais e de apresentação. Os objetos estruturais são os estados, as transições e os eventos especificados no *Statechart* subjacente. Os objetos navegacionais são as páginas, as ligações e as âncoras e os objetos relativos à apresentação são definidos pelo canais.

A estrutura navegacional do hiperdocumento é especificada pelo *Statechart* subjacente, que representa o documento em termos de objetos estruturais, pelos componentes navegacionais, além dos mecanismos de apresentação da informação. 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 da semântica é baseado na semântica operacional de *Statecharts* a partir de uma configuração de estados inicial. Os estados do *Statechart* são mapeados, segundo a função " m ", em um único objeto navegacional do tipo página. As possíveis ligações de ativação do hiperdocumento são representadas pelo conjunto de transições do *Statechart*. 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 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). As âncoras indicam os pontos de ancoragem no hiperdocumento. Elas representam a origem de uma ligação e a sua ativação faz com que todos os eventos associados a ele sejam disparados de acordo com a semântica operacional do *Statechart*.

A semântica de navegação de um hiperdocumento é a maneira como a informação a ser visitada é apresentada ao leitor que está navegando pelo hiperdocumento. Para visualizar as informações do hiperdocumento, os canais são invocados pelo sistema para interpretar um número arbitrário de páginas durante a navegação, sendo que diversas páginas podem ser exibidas concorrentemente. No HMBS, quando um usuário seleciona uma âncora, o sistema realiza as seguintes ações:

- gera o evento correspondente à âncora, disparando 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 e desabilitando a configuração anterior;
- define a nova configuração de contexto do hiperdocumento como sendo composto 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 um canal de apresentação para cada página da configuração de contexto.

A hierarquia de estados em um *statechart* pode ser descrita por um árvore E/OU com profundidade d. Os estados podem ser repetidamente decompostos em componentes OU ou em componentes E, e os estados origem e destino de um transição podem residir em qualquer nível da árvore. O modelo define uma operação chamada *Show-hview* que permite a visualização de páginas associadas aos estados imediatamente acima daqueles no nível N. A execução dessa operação não modifica a configuração do *statechart* atual mas somente fornece acesso aos nós que estão no nível acima ou abaixo na hierarquia.

A utilização do HMBS permite a determinação de alguns parâmetros relevantes para a apresentação física do hiperdocumento e para análise da alcançabilidade dos nós no hiperdocumento. Por meio da construção da árvore de alcançabilidade descrevem-se todas as possíveis seqüências de computação para o *statechart*, podendo, assim, determinar se uma parte de um hiperdocumento pode ser alcançada durante a navegação, ou se há nós que nunca serão alcançados; detectar as configurações finais dos estados; detectar a presença de caminhos de navegação cíclicos, checar a ordem temporal dos relacionamentos entre as páginas visitadas; etc. O HMBS também fornece um modelo computacional adequado para expressar concorrência, utilizando mecanismos úteis para representar a apresentação concorrente de múltiplas páginas e a navegação concorrente no hiperdocumento, além de fornecer ao autor suporte para especificar sincronização de atividades concorrentes. O conjunto de variáveis booleanas associadas ao HMBS pode ser usado para fornecer um mecanismo simples e efetivo para forçar as restrições de navegação nos leitores de um hiperdocumento, além de permitir que diferentes caminhos de navegação compartilhem um ou mais nós mas tenham intercaladas seções mutuamente exclusivas.

2.8.2 - EXTENSÕES AO HMBS

Hypercharts

Os *hypercharts* são uma proposta de linguagem de alto nível baseada em *statecharts* para a especificação de hiperdocumentos, com o objetivo de oferecer um conjunto de notações que permita especificar concisamente o comportamento de hiperdocumentos complexos, principalmente aqueles que envolvem requisitos de sincronização de dados multimídia em ambientes distribuídos. Eles podem ser considerados um superconjunto dos *statecharts*, de forma que, além de possuírem todas as notações permitidas em *statecharts* convencionais, introduzem novas notações que permitem a especificação do comportamento temporal de uma aplicação hipermídia [Pau97].

Os *hypercharts* possuem mecanismos específicos para a manipulação do tempo na especificação. A opção de se utilizar *statecharts* convencionais para a definição da semântica dos *hypercharts* se deve ao fato de que *statecharts* temporais Pnuelli apud Paulo¹, que também permitem especificação temporal, são restritos no que diz respeito à utilização de transições interníveis, referências a nomes de estados e mecanismos de história, além de permitirem o uso de estados instantâneos, ou seja, estados que são ativados e desativados dentro de um mesmo passo de execução Beeck apud Paulo². A semântica dos *hypercharts* é caracterizada pelos passos que são compostos de vários micro-passos, dentro dos quais somente uma transição é avaliada, permitindo a evolução das várias máquinas de estado concorrentemente. Sendo assim, nesta extensão, para cada estado do *statechart* são definidos dois mapeamentos para contadores, representando respectivamente, um relógio local ao estado e um registrador de história temporal desse estado. O relógio local registra o número de passos durante os quais o estado permanece ativo, sendo incrementado a cada passo. O registrador de história temporal mantém o último tempo de ativação de um estado, complementando o mecanismo de história nos *hypercharts*.

Os *hypercharts*, assim como *statecharts* temporais, possuem recursos de notação temporal chamados eventos temporais que permitem a especificação de transições cujos disparos são determinados pela passagem do tempo, enquanto seus estados de origem estão ativos. Sendo assim, é possível especificar requisitos de sincronização e comunicação como atraso e “jitter”,

¹ Pnuelli, A. e Kensten, Y.; “Timed and Hybrid Statechart and their textual representation”. Formal Techniques in Real-Time and Fault-Tolerant Systems - Second International Symposium, Nijmegen, Holanda. Lecture Notes in Computer Science, volume 571, pg. 591-620. Springer Verlag. Janeiro 1992. Paulo, Fabiano Borges, 1997.

² Beeck, M. V. D.; “A comparison of statecharts variants”. Formal Techniques in Real-Time and Fault-Tolerant Systems. Lecture Notes in Computer Science, volume 863, pg. 128-148. Springer Verlag. Setembro 1994. Paulo, Fabiano Borges, 1997.

além de viabilizar o mecanismo que modela a apresentação da informação no modelo XHMBs. A transformação de um *hyperchart* em um *statechart* equivalente à sua semântica obedece a três estratégias: uma voltada para estados do tipo AND; uma voltada para estados do tipo OR; e, finalmente, outra voltada para estados básicos. Em todas essas estratégias de transformação, o objetivo é incorporar a um estado um *TCS* (“*Time Controller Statechart*”), capaz de controlar o comportamento temporal de uma transição t com origem nesse estado.

Mecanismos de sincronização são necessários para permitir a comunicação quando vários *hypercharts* estão ativos concorrentemente. Os mecanismos de sincronização podem ser suportados pela difusão de eventos (*broadcasting*), geração de eventos internos e pelas transições M:N. Como as transições M:N apresentam algumas limitações, os *hypercharts* disponibilizam a notação de transições M:N sincronizadas, que permitem que os arcos que partem dos estados origem sejam rotulados com eventos temporais e que um tipo de sincronização seja associado à transição. Uma transição M:N sincronizada possui cinco grupos de componentes: estados origem, arcos origem, estados destino, rótulos (associados a cada arco de origem) e um tipo de sincronização. O tipo anotado em uma transição M:N sincronizada determina como as restrições impostas pelos eventos temporais associadas a cada arco de origem devem ser tratadas em termos da geração de uma restrição temporal única para toda a transição. A sincronização de transições pode ter os seguintes tipos:

- *Strong-or* (SO): garante a corretude temporal da cadeia de dados mais avançada, isto é, trata de considerar aquele estado origem que tiver sua atividade de apresentação encerrada antes dos demais;
- *Weak-and* (WA): garante a corretude temporal da cadeia de dados mais atrasada, isto é, trata de considerar o último estado origem a ter sua atividade de apresentação encerrada;
- *Master* (M(s)): garante a corretude temporal da cadeia de dados definida como *master*, isto é, trata de considerar o estado origem cujo nome é colocado entre parênteses junto com a notação do tipo de transição;
- *Or* (OR): considera a corretude temporal de qualquer uma das cadeias que terminar sua atividade de apresentação antes das demais, isto é, trata de considerar cada estado de origem em termos de seu término devido ao final da atividade de apresentação;
- *And* (A): garante a corretude temporal de todo o cenário de sincronização, isto é, trata de considerar a intersecção, se não for vazia, entre os intervalos definidos pelos eventos temporais associados a cada arco de origem;

- *Or-Master* (OM(s)): introduz a noção de não-determinismo de sincronização, limitado superiormente pela restrição temporal associada à cadeia de dados master, sendo assim, este tipo de transição será disparada se um dos estados origem for desativado pelo término normal de sua atividade de apresentação ou se o estado definido como master alcançar seu limite temporal máximo;
- *Strong-Master* (SM(s)): considera as restrições temporais relativas à cadeia de dados master e à cadeia de dados mais avançada, isto é, a transição irá disparar se o estado master atingir o seu limite temporal mínimo e algum dos estados for desativado, seja pelo término da sua atividade de apresentação ou por ter atingido seu limite temporal máximo;
- *Weak-Master* (WM(s)): introduz a noção de não-determinismo de sincronização limitado inferiormente pela restrição temporal associada à cadeia de dados master, sendo assim, este tipo de transição será disparada se o estado de origem master atingir seu limite temporal máximo e um dos estados de origem for desativado pelo término normal de sua atividade de apresentação;
- *And-Master* (AM(s)): neste tipo de sincronização o limite superior do intervalo dinâmico de disparo da transição é determinado pela cadeia master, enquanto que o limite inferior fica determinado pela cadeia mais atrasada. Sendo assim, o disparo de uma transição desse tipo ocorre nas seguintes situações: se todos os estados de origem atingirem os seus limites temporais mínimos e se o estado “master” atingir seu limite temporal máximo ou se todos os estados de origem terminarem suas atividades de apresentação.

Devido a algumas peculiaridades das aplicações hipermídia, tais como níveis de abstração, granularidade da especificação e repetição de certos padrões de comportamento, há a necessidade de que o modelo de especificação suporte essas características de forma conveniente. No caso dos padrões de comportamento repetitivos a especificação fica demasiadamente grande, tornando interessante que tais padrões possam ser especificados uma só vez, acompanhados da especificação formal de quantas vezes se repetem. A solução adotada pelos *hypercharts* é a utilização de estados parametrizados e o de abstração de transições. Os estados parametrizados permitem que os estados e respectivas transições com comportamento iguais sejam representados uma única vez. Essa representação é realizada por intermédio de um estado e um conjunto de transições acompanhadas de uma notação de indexação. A parametrização de estados do tipo OR é apresentada nas figuras 2.15. A abstração de transições permite que um estado seja decomposto isoladamente, independentemente das transições que

cruzam a sua borda, chegando aos seus subestados ou partindo deles.

(a) Estado Parametrizado

(b) Decomposição da Parametrização

Figura 2.15 - Parametrização do tipo OR

A possibilidade de a parametrização envolver transições associadas aos subestados parametrizados gera a necessidade de uma notação apropriada que permita especificar qual é exatamente o estado envolvido na origem ou no destino de uma determinada transição. Na maioria dos casos, porém, a identificação de um estado só pode ser especificada de forma relativa, ou seja, não é possível determinar o valor exato da variável de parametrização, mas esse valor pode ser determinado se o valor da variável que determina o estado anterior ou posterior for conhecido. Na figura 2.16 é apresentada a parametrização de um estado do tipo OR com transições associadas.

A abstração de transições define um nível de abstração pelo qual é possível representar apenas o contexto em que determinada transição se encontra, sem que seja necessário explicitar seus estados origem e destino. Trata-se de um mecanismo que abstrai transições em termos de superestados de origem e de destino, como apresentado na figura 2.17. O uso combinado dos mecanismos de parametrização de estados e abstração de transições constitui uma técnica de especificação bastante poderosa, permitindo especificar estruturas repetitivas complexas por intermédio de diagramas bastante concisos.

Figura 2.16 - Parametrização do tipo OR com transições associadas

Figura 2.17 - Abstração de transições

As soluções notacionais providas pelos *hypercharts* visam, principalmente, a especificação temporal em um nível mais alto do que aquele oferecido por *statecharts*. Os mecanismos de parametrização de estados e de abstração de transições, no entanto, não constituem, por si só, facilidades para esse tipo de especificação. É possível, porém, que a combinação de todos os recursos notacionais oferecidos por *hypercharts*, incluindo parametrização de estados e abstração de transições, constitua uma forma poderosa de

especificação de cenários complexos de sincronização hipermídia, satisfazendo aos requisitos de expressividade exigidos de um modelo com esse propósito [Pau97].

O XHMBS é uma extensão do modelo HMBS cuja principal característica é a capacidade de especificação de requisitos relativos à sincronização em hiperdocumentos. Além disso, outros requisitos relativos às aplicações hipermídia passam a ser passíveis de especificação pelo XHMBS devido às inovações incorporadas a esse modelo. Entre as inovações apresentadas pelo XHMBS estão a possibilidade de especificação da coordenação espacial e da distribuição da informação por *canais de apresentação e comunicação*, a separação total entre o conteúdo da informação e a sua estrutura através de um melhor uso de *âncoras* e, como modificação de maior impacto, a especificação navegacional por intermédio de *hypercharts*, que passam a substituir os *statecharts* convencionais como técnica subjacente ao modelo. O XHMBS continua a ser, como o HMBS, um modelo de projeto de hiperdocumentos. Dessa forma, para se especificar uma aplicação de forma completa, há a necessidade de um processo mais abrangente, partindo da modelagem semântica da informação e incluindo o XHMBS como modelo da estrutura e da semântica de navegação de um hiperdocumento [Paul97].

XHMBS - eXtended Hyperdocument Model Based on Statecharts

Um hiperdocumento é definido pelo XHMBS como uma 11-tupla $H = \langle Hyp, P, Anc, M, A, L, Pch, Cch, V_p, V_a, Ch \rangle$. Com relação à definição de um hiperdocumento pelo HMBS, alguns componentes foram modificados ou adicionados. O XHMBS utiliza *hypercharts* - *Hyp* para definir a estrutura do hiperdocumento, define o conjunto *Anc* de âncoras do hiperdocumento, associa por intermédio da função *A* uma âncora a cada evento contido em uma transição, define o conjunto *Pch* de Pchannels e o conjunto *Cch* de Cchannels, associa por intermédio da função *V_p* as páginas do hiperdocumento e estados associados a elas aos Pchannels, associa por intermédio da função *V_a* uma âncora e a página associada a ela a um Pchannel e associa por intermédio da função *Ch* um Cchannel a cada Pchannel utilizado pelo hiperdocumento. Os mapeamentos realizados pelas funções *M*, *A*, *V_p* e *V_a* são apresentados na figura 2.18. A definição completa de um hiperdocumento pelo XHMBS é mostrada no apêndice A.

Figura 2.18 - Mapeamento com âncoras e páginas

Na figura 2.19 é apresentada a especificação de uma apresentação multimídia no contexto de um hiperdocumento em XHMBS. O hiperdocumento é composto por cinco grandes porções: Intro, MM, Menu, Foto e Sobre. Ao iniciar a navegação são apresentados dois textos, um após o outro, ao mesmo tempo que uma trilha sonora. O segundo texto deve terminar junto com o áudio e é seguido de uma página de texto (menu) com ligações: para a introdução, para uma galeria de fotos e para a apresentação multimídia.

Figura 2-19 - Especificação de um hiperdocumento em alto nível

O XHMBS apresenta as seguintes características em relação ao HMBS:

- melhor definição de páginas: a definição de páginas utilizada pelo XHMBS torna mais clara a noção de como uma página é estruturada, permitindo, por exemplo, que

algoritmos de acesso direto às páginas sejam especificados sobre uma base formal bem definida;

- várias páginas associadas ao mesmo estado: a associação de várias páginas ao mesmo estado permite que, caso haja várias versões da mesma informação a serem disponibilizadas em uma mesma seção de navegação, a estrutura do hiperdocumento não precisará ser alterada, ficando por conta do sistema de apresentação, representado no modelo pelos *Pchannels*, a tarefa de apresentar a página correta.

- âncoras como unidades independentes da estrutura: utiliza um conjunto de âncoras *Anc* e um mapeamento *A* que as associa aos eventos definidos no *hyperchart* para evitar que o autor se preocupe com conteúdo da informação enquanto define a estrutura da aplicação;

- *Pchannels* e *Cchannels*: são abstrações mais robustas para os *readers* do HMBS. Um *Pchannel*, ou canal de apresentação, é uma abstração abrangente de um dispositivo de apresentação que deve incluir atributos de apresentação de alto nível e mecanismos de coordenação espacial, entre outros. Se o hiperdocumento é distribuído e, portanto, é necessária a utilização de uma rede de computadores, a especificação dos requisitos de comunicação é realizada através de *Cchannels*, ou canais de comunicação, que abstraem a interface da aplicação com o sistema de comunicação subjacente;

- relações de visibilidade: é constituída de dois mapeamentos V_a e V_p . O mapeamento V_p define as associações entre páginas e estados e possui restrições que garantem a corretude da utilização da nova definição de *M*. O mapeamento V_a permite que uma âncora seja associada a um *Pchannel* e provê independência de dispositivo para as âncoras, assim como ocorre com as páginas através do mapeamento V_p ;

- semântica de navegação do XHMBS: a semântica de navegação do XHMBS utiliza *hypercharts* em vez de *statecharts*, mas mantém a idéia de que a informação é apresentada conforme as configurações assumidas pelo modelo subjacente (neste caso, um *hyperchart*). A principal mudança está na forma com que a informação associada ao *hyperchart* é apresentada e isso se deve às novas definições de elementos e de mapeamentos do modelo XHMBS. Dessa forma, a semântica de navegação aqui apresentada é baseada no comportamento da aplicação quando uma configuração de estados do *hyperchart* é fornecida.

Em termos de notação, o XHMBS conta com os recursos gráficos dos *hypercharts*, da mesma forma que ocorre com os *statecharts* no HMBS, além de tabelas que representam os

mapeamentos definidos para o hiperdocumento (M, A, V_P, V_a, Ch).

As interações com o usuário no XHMBS podem ocorrer de duas formas: através das ligações do hiperdocumento, permitindo a navegação, e através das interfaces multimídia oferecidas pelo sistema de apresentação através dos *pchannels*. A modelagem das interações com o usuário pode ser realizada livremente com a notação de *hypercharts*, acrescentando-se uma ação que manipula o relógio global do *hyperchart* subjacente ao hiperdocumento.

O XHMBS satisfaz à maioria dos requisitos das aplicações hipermídia tais como: modelagem do domínio da aplicação, estruturação da informação, separação entre a estrutura e o conteúdo da informação, várias visões da mesma informação, versões sob medida do mesmo hiperdocumento, especificação em vários níveis de abstração e de granularidade, local de origem e destino da cadeia de dados, intervalos de apresentação, tipos de fontes, dependência temporal, atraso, jitter, taxas de throughput, sensibilidade a erros, interação usuário-computador, coordenação espacial, controle de atributos globais, tipo de mídia da cadeia de dados, noção de visita a um nó, estruturas alternativas de acesso à informação, ligações temporais, ligações dentro de componentes, contextos de informação e visão hierárquica do hiperdocumento. No entanto, o modelo não suporta a derivação de elementos da aplicação e ligações dinâmicas.

2.9 - COMPARAÇÃO ENTRE OS MÉTODOS

A seguir, a tabela 2.1 apresenta uma comparação entre os métodos e modelos estudados voltados para o projeto de aplicações hipermídia, segundo alguns requisitos considerados relevantes. Alguns requisitos foram obtidos das características dos próprios métodos e modelos, e outros de artigos específicos sobre o assunto. Os mecanismos de abstração permitem o reuso e o compartilhamento das estruturas do domínio da aplicação. No caso do OOHDM essa característica é satisfeita porque ele utiliza agregações e generalizações/especializações. A estruturação da informação permite que hierarquias sejam criadas e, conseqüentemente, propriedades sejam herdadas. Todos os métodos e modelos satisfazem a essa característica porque possuem primitivas tais como: componentes, unidades, partes, agregações e composições. As estruturas de acesso permitem acessar e consultar as informações. O RMD e o RMM satisfazem a essa característica pelo uso de índices, roteiros e grupos, enquanto que o OOHDM a satisfaz pela definição de contextos navegacionais. Uma semântica de navegação específica permite determinar todos os aspectos relacionados à navegação do hiperdocumento. O RMD, RMM e o OOHDM satisfazem essa característica pela definição de regras de

conversão. A utilização de mídia dinâmica permite que a informação percebida pelo leitor seja modificada sem sua intervenção. O AHM não foi comparado aos outros métodos por se tratar de um modelo para especificação de sistemas hipermídia e não de aplicações hipermídia.

Como o HMBS e o XHMBS não são métodos para o projeto de aplicações hipermídia este trabalho propõe-se a desenvolver um método para que incorpore ambos os modelos.

Tabela 2.1 - Comparação entre os métodos e modelos para o projeto e o desenvolvimento de aplicações hipermídia

Métodos e Modelos para o projeto e o desenvolvimento de aplicações hipermídia						
Requisitos	HDM	RMD	RMM	OOHDM	EORM	XHMBS
Mecanismos de abstração				X	X	X
Estruturação da informação	X	X	X	X	X	X
Estruturas de acesso		X	X	X		
Semântica de navegação bem definida		X	X	X		X
Mídia ativa						X

2.10 - CONSIDERAÇÕES FINAIS

O uso dos métodos e modelos estudados tem propiciado melhoras significativas na qualidade e velocidade do processo de desenvolvimento de aplicações hipermídia, além de produzir projetos mais completos e consistentes. No entanto, nenhum dos métodos estudados apresenta todos os requisitos considerados necessários, exigindo que em muitas situações sejam utilizados outros métodos ou modelos complementares. Outra consideração importante é que nenhum dos métodos estudados é totalmente aplicável a todas as classes de aplicações hipermídia, acarretando, muitas vezes, uma subutilização dos recursos oferecidos pelos métodos ou produção de aplicações mal-projetadas. No próximo capítulo são apresentados os modelos utilizados pelo HMBS/M e processo seguido por ele.

CAPÍTULO 3

PROPOSTA DE UM MÉTODO PARA O PROJETO DE APLICAÇÕES HIPERMÍDIA - HMBS/M

3.1. CONSIDERAÇÕES INICIAIS

O método para o projeto de aplicações hipermídia proposto neste trabalho tem como característica principal a utilização do HMBS em um das suas fases. Além disso, o método pretende satisfazer todos os requisitos apresentados na capítulo dois, utilizando algumas características de outros métodos encontrados na literatura e introduzindo novos conceitos.

O capítulo está organizado da seguinte forma: na seção 3.2 é apresentado a caracterização de um domínio da aplicação no qual são mostrados os tipos mais comuns de aplicações hipermídia encontrados atualmente. Na seção 3.3 são apresentados os modelos utilizados pelo método, suas características e representação. Para facilitar o entendimento dos modelos são apresentados alguns exemplos. Na seção 3.4 é apresentado o processo de desenvolvimento do HMBS/M, focalizando cada fase do método e seus objetivos. Na mesma seção é apresentado os possíveis tipos de implementação de aplicações hipermídia identificados neste trabalho.

3.2. CARACTERIZAÇÃO DE UM DOMÍNIO DA APLICAÇÃO

Hipermídia pode ser definida como sendo hipertextos que utilizam diferentes tipos de mídia, tais como gráficos, áudio, vídeo, animações, etc. Desse modo, o uso da hipermídia permite desenvolver complexas aplicações de gerenciamento de informação de modo mais efetivo e útil.

As aplicações baseadas no conceito de hipermídia são muitas e diferem entre si pela estruturação e volatilidade de suas informações. Hoje em dia as aplicações hipermídia mais comuns são [Bel97]:

- Dicionários e enciclopédias- esse tipo de aplicação geralmente apresenta uma grande quantidade de informações bem estruturadas e pouco volátil. O uso de

hipermídia facilita as pesquisas realizadas pelo usuário bem como ajuda a gerenciar o aprendizado de uma melhor forma;

- Aplicações médicas - esse tipo de aplicação envolve uma grande quantidade de informações obtidas de várias fontes, tais como livros, registros de pacientes, publicações em jornais e revistas técnicas, anotações de outros colegas, etc. Essas informações nem sempre estão bem estruturadas e sofrem atualizações freqüentemente. A utilização de hipermídia permite associar essas informações, tornando mais fácil o seu acesso e ajudando o trabalho colaborativo na medicina;
- Assistência on-line - esse tipo de aplicação apresenta informações bem estruturadas e pouco voláteis. O uso de hipermídia torna mais fácil e flexível acessar essas informações, fazendo com que o usuário encontre a informação desejada mais rapidamente;
- Sistemas de aprendizado - esse tipo de aplicação envolve informações estruturadas e um pouco volátil. O uso de hipermídia nesse tipo de aplicação facilita o aprendizado, estimulando o usuário a aprender explorando a aplicação;
- Quiosques - esse tipo de aplicação armazena uma grande quantidade de informações bem estruturadas e pouco voláteis. O uso de hipermídia permite criar quiosques interativos que servem como guias em museus, shoppings, agências de turismo, etc, além de ajudar o usuário a encontrar a informação desejada mais facilmente;
- Trabalho corporativo - esse tipo de aplicação envolve uma grande quantidade de informações não estruturadas, altamente ligadas e em constante atualização. O uso de hipermídia é apropriado para representar as informações e a discussão sobre essas informações, bem como o seu compartilhamento;
- Sistemas de apoio a decisão (SAD) - esse tipo de aplicação geralmente apresenta informações estruturadas e um pouco voláteis. O uso de hipermídia pode melhorar o processo de tomar decisões, facilitando a navegação pelos vários modelos, dados e relatórios da aplicação SAD;
- Aplicações legais - esse tipo de aplicação apresenta informações estruturadas e pouco voláteis. O uso de hipermídia é um poderoso meio para reunir informações sobre os casos, cruzar informações de modo apropriado, recuperar informações rapidamente e representar as idéias dos advogados de modo eficiente;
- World Wide Web (W3) - a WWW é um mecanismo de recuperação de informações baseado em hipertextos que fornece acesso a informações em plataformas

heterogêneas conectadas à Internet. Geralmente as aplicações disponíveis na WWW podem ser estruturadas ou desestruturadas, pouco ou muito volátil, dependendo do domínio da aplicação. Um exemplo de aplicação disponível na WWW são os catálogos de venda de produtos.

O método para o projeto de aplicações hipermídia proposto, como qualquer outro método de projeto não é útil para todos os tipos de aplicações. O método apresentado é mais apropriado para aplicações que possam ser representadas por classes, relacionamentos entre essas classes e objetos das classes. Dentre as aplicações apresentadas as que mais se encaixam no perfil traçado são as aplicações médicas e legais, os quiosques e aplicações para WWW. Como algumas dessas aplicações apresentam freqüentes atualizações a solução é automatizar as fases do método proposto e utilizar banco de dados para armazenar as informações.

Aplicações que apresentam elos dinâmicos, isto é, elos que são criados em tempo de execução segundo a avaliação de alguma condição pré-definida não podem ser especificadas utilizando o método proposto. A característica dinâmica desse tipo de aplicação não pode ser modelada pelo modelo de navegação utilizado já que *Statecharts* são estáticos. Todos os tipos de aplicações podem apresentar essa característica, no entanto os sistemas de aprendizado, os SAD e trabalho corporativo são as aplicações que mais freqüentemente utilizam desse artifício. A tabela 3.1 apresenta os tipos de aplicações apresentados em relação a estruturação e volatilidade de suas informações.

Tabela 3.1 - Tipos de aplicações hipermídia X estruturação e volatilidade de suas informações

Tipos de aplicação	Estruturação da informação	Volatilidade da informação
Dicionários e Enciclopédias	Baixa	Baixa
Aplicações Médicas	Baixa	Média
Assistência on-line	Alta	Baixa
Sistemas de aprendizado	Alta	Baixa
Quiosques	Alta	Baixa
Trabalho Corporativo	Média	Alta
Sistemas de apoio a decisão	Alta	Baixa
Aplicações Legais	Alta	Baixa
World Wide Web	Depende da aplicação	Depende da aplicação

3.3. MODELOS UTILIZADOS

3.3.1 MODELO DE OBJETOS (MO)

O modelo de objetos é o primeiro modelo definido na fase de modelagem conceitual. Ele é composto pelos seguintes elementos primitivos: classes, atributos e relacionamentos.

As classes são estruturas de informação que representam algum objeto do mundo real do domínio da aplicação. Classes possuem atributos que representam abstrações das características dos objetos do mundo real e são tipados conforme tipos pré-definidos: inteiro, cadeia, texto, data, vídeo, imagem, som, animação e gráfico. O tipo de um atributo deve representar ou um relacionamento implícito, ou o tipo de mídia usado, ou sua aparência na aplicação final. Um atributo pode ter mais de uma aparência. Cada possível aparência do atributo é chamada de "perspectiva" e uma delas deve ser escolhida como *default*. A perspectiva *default* é a perspectiva sob a qual um atributo é inicialmente mostrado e é especificada por um sinal de +. As várias perspectivas de um atributo dão origem a um conjunto de ligações que conectam essas perspectivas.

Os relacionamentos representam as ligações existentes entre as classes do domínio da aplicação e podem possuir atributos próprios. Regras podem ser definidas para representar o papel de uma classe em um relacionamento. A definição dessas regras é útil para deixar claro a ordem de leitura dos relacionamentos, principalmente nos auto-relacionamentos.

Invariantes podem ser definidos para afirmar que uma propriedade é sempre válida. No caso de aplicações hipermídia a definição de invariantes pode ser irrelevante desde que não produza mudanças substanciais na aplicação final.

A cardinalidade C dos relacionamentos tem as seguintes possibilidades:

- se C não existir, significa que podem existir zero ou mais objetos, esse é o *default*;
- se C for *, significa que podem existir zero ou mais objetos relacionados;
- se C for +, significa que podem existir um ou mais objetos relacionados;
- se C for N, significa o número exato de objetos da classe relacionados;
- se C for M..N, significa o número mínimo e máximo de objetos relacionados entre as classes.

A representação de que todos os objetos da classe participam do relacionamento, denominada de marcador de totalidade, é denotada por um pequeno retângulo cheio no arco que conecta o relacionamento à classe.

Na figura 3.1 são apresentadas duas classes (Classe 1 e Classe 2) conectadas por um

relacionamento. As classes e o relacionamento possuem atributos próprios. Duas regras (Regra1 e Regra2) representam o papel da Classe1 e da Classe2 no relacionamento e a cardinalidade é representada por C1 e por C2 para as classes Classe1 e Classe2 respectivamente. A Classe2 apresenta o marcador de totalidade, informando que todos os objetos da classe participam do relacionamento.

Figura 3.1 - Representação de classes e relacionamentos

Dois mecanismos de estruturação são usados no modelo de objetos: agregação e especialização/generalização. A agregação de classes permite definir classes complexas como agregação de classes mais simples. A notação para a agregação de classes é mostrada na figura 3.2, com as classes componentes e as suas cardinalidades contidas dentro da classe agregada.

Figura 3.2 - Representação de Agregação

A generalização permite que uma classe, chamada supertipo, seja formada pelas propriedades comuns de várias classes, chamadas subtipos. A generalização de classes é representada por um triângulo conectando uma classe supertipo às classes subtipos. Um triângulo cheio significa que as classes subtipos são disjuntas (figura 3.3 (a)), caso contrário, significa que as classes não são disjuntas (figura 3.3 (b)). A especialização múltipla permite que um subtipo seja definido como especialização de mais de uma classe supertipo (figura 3.4). Qualquer que seja o tipo de especialização, a classe subtipo herda os atributos e relacionamentos da classe supertipo. Nesse tipo de aplicação não há interesse em especificar o comportamento das classes, embora este também seja herdado da classe supertipo.

Figura 3.3 - Representação de especialização/generalização

Figura 3.4 - Representação de especialização múltipla

3.3.2 MODELO DE FATIAS (MF)

O modelo de fatias determina como a informação numa classe escolhida é agrupada para ser apresentada ao usuário. Este modelo utiliza os mesmos conceitos do modelo de objetos e introduz um novo conceito, fatias. Uma fatia agrupa um ou mais atributos de uma dada classe. Há dois tipos fundamentais de fatias: fatia Principal e fatia de Ligações.

A fatia Principal é aquela a ser inicialmente apresentada quando uma determinada classe for acessada. Uma fatia Principal deve ser escolhida para cada classe e sua notação é um triângulo cheio na borda superior direita da classe. A fatia de Ligações agrupa os relacionamentos de cada classe. Outras fatias podem ser criadas conforme os atributos que cada classe possui.

Neste modelo são especificados os relacionamentos entre fatias, que permitem navegar para outras fatias da classe a partir da fatia Principal. Os relacionamentos que levam a uma classe que está fora do escopo da aplicação não aparecem como relacionamentos, sendo representados na fatia de Ligações como âncoras, seguindo a seguinte notação: âncora(relacionamento) → {nó-destino}. A figura 3.5 apresenta a notação para os conceitos definidos nesta seção.

Figura 3. 5 - Representação de fatias

3.3.3 CONTEXTOS DE NAVEGAÇÃO

Um contexto de navegação permite reunir informações do domínio da aplicação que estão relacionados segundo algum critério. A noção de contexto de navegação visa a reduzir a carga cognitiva imposta pela navegação em hiperdocumentos, minimizando o problema da desorientação do usuário. Isso é realizado propondo-se caminhos com estratégias de navegação bem definidas e informações contextuais que auxiliam o usuário a percorrer os nós de seu interesse.

A definição de um contexto de navegação é feita pelo seu nome, seu tipo, seus atributos, os objetos que o constituem, os pontos de entrada, o tipo de caminho a percorrer, o tipo de comportamento, os elos que ele pode exportar e importar e comentários sobre o próprio contexto.

O nome do contexto de navegação é qualquer nome escolhido pelo projetista. Existem alguns tipos de contextos de navegação que são induzidos a partir dos vários elementos do modelo conceitual. Esses contextos de navegação podem ser agrupados conforme os modelos em que eles podem ser especificados. Sendo assim, têm-se dois grupos de contextos de navegação:

- contextos de navegação especificados no modelo navegacional de tipos e no modelo navegacional de instâncias:
 - derivado de atributo,
 - derivado de classe,

- derivado de elo,
- índice,
- sessão;
- contextos de navegação especificados no modelo navegacional de instâncias:
 - arbitrário,
 - derivado de estrutura,
 - seleção.

O contexto de atributo permite navegar entre as instâncias de uma dada classe ou relacionamento segundo o valor específico de um determinado atributo.

O contexto derivado de classe permite navegar por todas as instâncias de uma determinada classe.

O contexto derivado de elo é obtido dos relacionamentos do modelo conceitual e permite navegar pelas instâncias desses relacionamentos.

O contexto de índice contém as instâncias apontadas pelo índice. Esse tipo de contexto de navegação tem um comportamento padrão que permite navegar do índice para qualquer elemento e vice-versa.

O contexto de sessão é construído dinamicamente e seus elementos são adicionados conforme alguma condição avaliada em tempo de execução.

O contexto arbitrário permite navegar por instâncias escolhidas arbitrariamente pelo projetista, segundo algum critério.

O contexto derivado de estrutura é obtido de classes agregadas e permite navegar dentro dessa estrutura.

O contexto de seleção permite ao usuário especificar sua própria consulta à aplicação usando uma linguagem de consulta pré-definida. Nesse caso as informações devem estar armazenadas em uma base de dados para que o usuário possa escolher sobre qual classe ou relacionamento da aplicação ele deseja fazer a consulta. A consulta é especificada por um predicado lógico definido usando a notação formal de *Backus e Naur* - BNF apresentada a seguir:

```

predicado ::= <cláusula-1> (<cláusula-2> | <cláusula-3>)
<cláusula-1> ::= "∀" <variável> "∈" (<classe> | <relacionamento>) " | "
<cláusula-2> ::= <valor-atributo> <símbolo-1> <variável>
               { <símbolo-2> <cláusula-2> }
<cláusula-3> ::= (<variável>, <variável>) "∈" <relacionamento>
  
```

$\langle \text{símbolo-2} \rangle \langle \text{cláusula-2} \rangle \langle \text{símbolo-2} \rangle$
 $\langle \text{variável} \rangle " \in " \langle \text{classe} \rangle$
 $\langle \text{variável} \rangle ::= \text{identificador} \mid (\text{identificador} " . " \text{identificador})$
 $\langle \text{classe} \rangle ::= \text{identificador}$
 $\langle \text{relacionamento} \rangle ::= \text{identificador}$
 $\langle \text{valor-atributo} \rangle ::= \text{identificador}$
 $\langle \text{símbolo-1} \rangle ::= ["(" " \neg ") "] " = " \mid " < " \mid " > " \mid " > = " \mid " < " \mid " < = " \mid " \leq " \mid " \subset "$
 $\langle \text{símbolo-2} \rangle ::= " ^ " \mid " \vee "$

Na figura 3.6 é apresentado o modelo de objetos para a relação entre Cliente e Conta Bancária. Com base no modelo de objetos apresentado o predicado lógico para especificar todos os clientes que nasceram em 1974 e cujo estado civil seja solteiro pode ser definido da seguinte forma: $\forall c \in \text{Cliente} \mid 1974 \in c.\text{data-nascimento} \wedge \text{estado-civil} = \text{solteiro}$. O predicado lógico para especificar o nome e o número da conta dos clientes que tenham em seu nome Souza pode ser definido da seguinte forma: $\forall x \in \text{cliente} \mid (x, x1) \in \text{tem} \wedge \text{Souza} \in x.\text{nome} \wedge x1 \in \text{conta}$.

Figura 3.6 - Modelo de objetos

Os atributos do contexto de navegação representam informações sobre o próprio contexto. Os objetos que constituem um contexto de navegação são definidos na cláusula inclui e representam outros contextos, objetos, classes e relacionamentos que constituem o contexto especificado. No caso de classes e relacionamentos é possível definir esta cláusula por um predicado lógico definido segundo a BNF utilizada para o contexto de seleção. Os pontos de entrada de um contexto de navegação especificam as instâncias do contexto que podem ser acessadas de fora do contexto.

O caminho de um contexto de navegação pode ser de três tipos:

- sequencial,
- ramificado,
- condicional.

No caminho sequencial as instâncias do contexto são apresentadas sequencialmente

segundo uma ordem pré-definida. O caminho ramificado contém ramos que o usuário deve escolher explicitamente para continuar a navegação. O caminho condicional possui ramos nos quais a aplicação avalia condições pré-definidas pelo autor para saber para onde navegar. O caminho condicional pode ser:

- procedural,
- programável,
- variável,
- paralelo.

O caminho condicional procedural pode aparecer como item de um outro caminho. O caminho condicional programável pode armazenar informações para uso futuro. O caminho condicional variável possui elos variáveis e o próximo elo é computado dinamicamente. Finalmente, o caminho condicional paralelo pode ser executado simultaneamente e sincronizado com atividades em outros ramos.

O comportamento de um contexto de navegação é definido segundo dois aspectos: controle e restrição. O controle de navegação dentro do contexto pode ser:

- automático,
- passo a passo .

No controle automático o usuário é guiado automaticamente pela aplicação. No controle passo a passo o usuário escolhe quando dar o próximo passo. O controle passo a passo é subdividido em:

- acesso direto,
- acesso seqüencial,
- acesso direto seqüencial.

No caso de acesso direto, os elos de contexto definem um índice que permite ao usuário escolher diretamente a instância que deseja visitar. No seqüencial os elos de contexto definem um caminho seqüencial a ser percorrido pelo usuário. Finalmente, o acesso direto seqüencial combina os dois anteriores, permitindo ao usuário escolher diretamente a instância que se deseja visitar ou um nó para começar a navegação por um caminho já definido.

Em um contexto, a navegação pode ser limitada segundo três restrições:

- livre,
- exclusiva,
- parcial.

Quando a restrição é livre o usuário pode acessar diretamente qualquer elemento do contexto em questão ou pode percorrer qualquer elo de cada elemento do contexto de navegação, mesmo que saia da sequência definida pelos elos de contexto. Na restrição exclusiva somente os elementos contidos no índice do contexto de navegação podem ser acessados ou somente o caminho pré-definido pelo autor pode ser seguido. Finalmente, na restrição parcial, o contexto de navegação importa elos de outros contextos de navegação.

A cláusula importa/exporta serve para definir os elos que o contexto de navegação pode importar de outros contextos e/ou exportar para outros contextos quando há interseção. Sendo assim, pode-se definir quais elos são visíveis a partir de um elemento de interseção. Os comentários especificam informações adicionais sobre o próprio contexto de navegação. A tabela 3.2 apresenta a sintaxe para a definição dos contextos de navegação.

Tabela 3.2 - Representação dos contextos de navegação

Nome do Contexto	<nome-contexto>
Tipo do Contexto:	<ul style="list-style-type: none"> ◆ Modelo navegacional de tipo e de instâncias: <ul style="list-style-type: none"> ◆ Derivado de atributo ◆ Derivado de classe ◆ Derivado de elo ◆ Índice ◆ Sessão ◆ Modelo navegacional de instâncias: <ul style="list-style-type: none"> ◆ Arbitrário ◆ Derivado de estrutura ◆ Seleção
Atributos de contexto:	<nome-atributo: tipo atributo>
Inclui { outros contextos, objetos, classes, relacionamentos}:	<nome-contexto, nome-objeto, nome-classe, nome-relacionamento, consulta>
Pontos de entrada:	<instâncias>
Caminho:	<ul style="list-style-type: none"> ◆ Sequencial ◆ Ramificado ◆ Condicional
Comportamento:	<ul style="list-style-type: none"> ◆ Tipos de controle: <ul style="list-style-type: none"> ◆ Automático ◆ Passo a passo: <ul style="list-style-type: none"> ◆ acesso direto ◆ acesso sequencial ◆ acesso direto sequencial ◆ Tipos de restrição: <ul style="list-style-type: none"> ◆ livre

Cont. tabela 3.2 - Representação dos Contextos de Navegação

♦ exclusiva
♦ parcial
Importa/Exporta:
Comentários:

Um exemplo de definição de um contexto de navegação é apresentado na tabela 3.3. Esse exemplo tem como base o modelo de objetos apresentado na figura 3.6. O contexto definido permite navegar por todos os objetos da classe Cliente ordenados alfabeticamente. Sendo assim, o contexto é do tipo Classe e inclui a classe Cliente. O contexto não apresenta atributos próprios e o ponto de entrada para o contexto podem ser qualquer objeto da classe Cliente. Não há um caminho pré-definido para o usuário seguir. O usuário é quem controla a navegação e pode acessar arbitrariamente qualquer objeto da classe Cliente, percorrendo qualquer elo do objeto mesmo que saia do contexto de navegação. Nenhum elo é importado ou exportado, pois a restrição de navegação é livre e nenhum comentário é feito sobre o contexto.

Tabela 3.3 - Um exemplo de contexto de navegação

Nome do Contexto	CN1 Cliente de A .. Z	
Tipo do Contexto	Classe	
Atributos de contexto	-	
Inclui{ outros contextos, relacionamentos, classes e objetos}	Cliente	
Pontos de entrada	Todas as instâncias da classe Cliente	
Caminho	-	
Comportamento	Controle Passo a passo e acesso direto	Restrição Livre
Importa/Exporta	-	
Comentários	-	

3.3.4 MODELO NAVEGACIONAL DE TIPOS (MNT)

O modelo navegacional de tipos modela a navegação na aplicação baseando-se nos contextos de navegação definidos. Este modelo mistura os conceitos dos modelos da fase anterior e as estruturas de acesso. Sendo assim os conceito de classe, atributo, agregação, especialização/generalização e fatia são utilizados do mesmo modo que nos modelos anteriores e representados pela mesma notação. Os relacionamentos e a maneira de acessar inicialmente as classes do domínio da aplicação são representados por estruturas de acesso com base nos contextos de navegação definidos. As estruturas de acesso utilizadas são: índices, roteiros

guiados, roteiros guiados indexados, roteiros guiados condicionais, índices condicionais e grupos. Um índice é uma lista de instâncias das classes. Um roteiro guiado é um caminho sequencial, circular pelas instâncias de uma ou mais classes, permitindo ao usuário ir para frente e voltar no caminho. O roteiro guiado indexado é uma mistura de roteiro guiado e índice, fornecendo acesso direto a cada item de uma lista e permitindo mover-se para frente e para trás. Os roteiros guiados e índices condicionais permitem escolher com base em condições e predicados lógicos quais instâncias de uma dada classe podem ser acessadas. Os grupos são menus que permitem o acesso a outras partes da aplicação hipermídia.

A partir desse modelo os relacionamentos são classificados em dois tipos: relacionamentos da aplicação e relacionamentos estruturais. Os relacionamentos da aplicação conectam informações de diferentes classes, permitindo mudar o contexto da navegação. Já os relacionamentos estruturais conectam as fatias de uma mesma classe e as estruturas de acesso a essas classes. Deste modo, os relacionamentos estruturais representam como acessar inicialmente as classes da aplicação e não permitem mudar o contexto da navegação.

Os relacionamentos com cardinalidade 1:1 são representados por âncoras na fatia de Ligações das classes envolvidas e por uma linha unidirecional. Os relacionamentos com cardinalidade M:N são divididos em dois relacionamentos 1:N. Estes relacionamentos são representado por âncoras na fatia de Ligações das classes envolvidas, por uma linha unidirecional no sentido da classe cuja cardinalidade é 1 e por uma das estruturas de acesso no sentido da classe cuja cardinalidade é N. Um exemplo é apresentado na figura 3.7.

As âncoras representam a existência de um relacionamento da aplicação ou de um relacionamento para uma classe fora do escopo da aplicação, como no modelo de fatias.

A tabela 3.4 apresenta a notação para o modelo navegacional de tipos e a figura 3.8 apresenta um exemplo. O exemplo apresenta as classes Disciplina e Corpo Docente, os relacionamentos entre elas e as estruturas de acesso que permitem acessá-las no início da navegação: CN1, CN2, CN3 e CN4. A classe Corpo Docente está dividida em cinco fatias: Principal, Ligações, Formação-acadêmica-t, Formação-acadêmica-v, Áreas-pesquisa. O relacionamento com a classe Disciplina é representado pelo roteiro guiado CN5 e pela respectiva âncora na fatia de Ligações. A classe Disciplina é uma generalização das classes Disciplina-grad e Disciplina-pós-grad e está dividida em três fatias: Principal, Ligações e Bibliografia. O relacionamento com a classe Corpo Docente é representado pela linha tracejada e pela respectiva âncora na fatia de Ligações. As classes subtipos apresentam cada uma informações exclusivas. A classe Disciplina-grad apresenta um auto-relacionamento

representado pelo índice CN6 e pela respectiva âncora na Fatia de Ligações. A classe Disciplina-pós-grad apresenta o atributo nvagas na fatia Principal.

Figura 3.7 - Mapeamento dos relacionamentos do modelo de objetos para o modelo navegacional de tipos

Tabela 3.4 - Notação do modelo navegacional de tipos

Índice
Roteiro Guiado
Roteiro Guiado Indexado
Roteiro Guiado Condicional
Roteiro Guiado Indexado Condicional

Cont. tabela 3.4 - Notação do modelo navegacional de Tipos

Grupo
âncora(relacionamento)	Relacionamento que será implementado como uma âncora
Relacionamento estrutural
Relacionamento da aplicação

Figura 3.8 - Modelo navegacional de tipos

3.3.5 MODELO NAVEGACIONAL DE INSTÂNCIAS (MNI)

O modelo navegacional de instâncias permite modelar as instâncias das classes e os relacionamentos do modelo navegacional de tipos em termos de estados e transições. Ele utiliza *Statecharts*, seguindo a definição do *Hypertext Model Based on Statecharts* HMBS - para

especificar a estrutura e a semântica de navegação da aplicação hipermídia. O modelo navegacional de instâncias baseia-se no modelo navegacional de tipos e nos contextos de navegação já especificados.

No modelo navegacional de instâncias de uma aplicação hipermídia é representada por estados, transições e eventos. Na figura 3.9 é apresentado, parcialmente, o modelo navegacional de instâncias para o modelo navegacional de tipos da figura 3.8.

Figura 3.9 - Modelo navegacional de instâncias

Seguindo a definição do HMBS, os estados do *Statechart* são mapeados para páginas segundo uma função "m". Os estados do tipo AND não são mapeados para páginas pois tais estados têm como objetivo modelar a apresentação de páginas concorrentes. A função "m" para o modelo da figura 3.9 é apresentada na tabela 3.5.

Tabela 3.5 - Função "m"

Estado	Página
CN1	P ₁
CN2	P ₂
CN3	P ₃

Cont. tabela 3.5 - Função "m"

CN4	P ₄
Antonio-principal-foto	P ₅
Antonio-principal-inf	P ₆
ES-principal	P ₇
ES-bibliografia	P ₈
ES-ligacoes	P ₉

As páginas possuem um conteúdo, um título e âncoras associados a elas. O conteúdo associado a uma página pode ser um texto, um vídeo, uma imagem ou um áudio. As informações referentes a cada página são apresentadas na tabela 3.6.

Tabela 3.6 - Informações referentes às páginas

Página	c - Informação	t - Título	Anc _p - Âncora
P ₁	-	Menu Principal	{Índice Corpo Docente, Roteiro Corpo Docente, Índice Disciplinas}
P ₂	-	Índice do Corpo Docente	{Antonio, Pedro, Sair}
P ₃	-	Roteiro Guiado pelo Corpo Docente	{Ativa, Sair}
P ₄	-	Índice de Disciplinas	{CG, ES, ICC, Sair}
P ₅	Imagem-1	Foto-Antonio	Ø
P ₆	Texto-1	Informações Gerais	Ø
P ₇	Texto-2	Informações Gerais	Ø
P ₈	Texto-3	Bibliografia	Ø
P ₉	-	Âncoras	{Próximo, Anterior, Professor, Sair}

As âncoras de cada página são associadas aos eventos de uma transição segundo uma função "ae". A função "ae" é apresentada na tabela 3.7. As transições no HMBS são unidirecionais, sendo assim os relacionamentos bi-direcionais dão origem a duas transições unidirecionais. O HMBS não restringe o mapeamento de âncoras somente às transições cuja origem é o estado associado à página que contém a âncora, mas permite que o mesmo evento no rótulo de uma mesma transição seja utilizado para representar diferentes âncoras, dependendo do contexto em que se encontrar. As âncoras também podem ser associadas a eventos de uma transição cujo estado origem seja ancestral do estado associado à página que contém a âncora sendo definida.

Tabela 3.7 - Função "ae"

Página - P	Âncora - Anc _p	Evento - E
P ₁	Índice Corpo Docente	e2
P ₁	Roteiro Corpo Docente	e3
P ₁	Índice Disciplinas	e4

Cont. tabela 3.7 - Função "ae"

P ₂	Antonio	e2a
P ₂	Pedro	e2b
P ₂	Sair	v2
P ₃	Ativar	e3
P ₃	Sair	v3
P ₉	Próximo	p
P ₉	Anterior	v
P ₉	Professor	e6
P ₉	Sair	s

3.3.6. CANAIS

Os canais são mecanismos utilizados para apresentar as informações de uma aplicação hipermídia. Os canais podem ser de quatro tipos: texto, imagem, áudio e vídeo. A definição dos canais é feita por uma tabela, considerando as características específicas de cada tipo.

Todos os tipos de canais apresentam como característica básicas um nome e informações sobre a formatação do título. Tais informações são: a fonte utilizada, o estilo, o tamanho e a cor da fonte e o posicionamento do título. A fonte do título pode ser dos tipos freqüentemente utilizados como: *Times New Roman*, *Arial*, *Roman*, etc. O estilo pode ser: normal, negrito, itálico ou negrito e itálico. O tamanho e a cor variam conforme a fonte escolhida e o posicionamento pode ser: centralizado, à direita ou à esquerda.

O canal do tipo texto é definido segundo a tabela 3.8. Este tipo de canal apresenta como características específicas a formatação do texto a ser apresentado e a opção de serem apresentadas as barras de rolagens à direita e na base da página. As informações sobre a formatação do texto a ser apresentado são as mesmas utilizadas para a formatação do título.

Tabela 3.8 - Definição do canal texto

Nome-canal	Canal-texto-1			
Fonte-título	Arial			
Estilo-título	Normal	Negrito	Itálico	Negrito Itálico
		X		
Tamanho-fonte-título	16			
Posição-título	Centralizado	Direita	Esquerda	
	X			
Cor-fonte-título	Automática			
Fonte-texto	Times New Roman			
Estilo-texto	Normal	Negrito	Itálico	Negrito Itálico
	X			
Tamanho-fonte-texto	12			

Cont. tabela 3.8 - Definição do canal texto

Posição-texto	Centralizado	Direita	Esquerda
	X		
Cor-fonte	Automática		
Cor-fundo	Cinza-claro		
Scroll-base	Não		
Scroll-direita	Sim		

O canal do tipo imagem é definido na tabela 3.9. Este tipo de canal possui somente as características básicas citadas anteriormente.

Tabela 3.9 - Definição do canal imagem

Nome-canal	Canal-imagem-1			
Fonte-título	Arial			
Estilo-título	Normal	Negrito	Itálico	Negrito Itálico
		X		
Tamanho-fonte-título	16			
Posição-título	Centralizado	Direita	Esquerda	
	X			
Cor-fonte-título	Automática			

O canal do tipo vídeo é definido na tabela 3.10. Este tipo de canal apresenta como características específicas o *status* do vídeo, a opção de se exibir ou não o controlador de vídeo e a velocidade do vídeo. O *status* do vídeo indica se o vídeo é ativado automaticamente (ativado) ou se é necessária alguma ação do usuário (desativado) para ativá-lo. O controlador de vídeo pode ser implementado como uma barra de funções que permite ao usuário voltar, ir para frente, parar e recomençar o vídeo. A velocidade do vídeo pode ser: normal, rápida ou lenta.

Tabela 3.10 - Definição do canal vídeo

Nome-canal	Canal-vídeo-1			
Fonte-título	Arial			
Estilo-título	Normal	Negrito	Itálico	Negrito Itálico
		X		
Tamanho-fonte-título	16			
Posição-título	Centralizado	Direita	Esquerda	
	X			
Cor-fonte-título	Automática			
Status-vídeo	Ativado			
Exibir-controlador	Sim			
Velocidade-vídeo	Normal			

O canal do tipo áudio é definido na tabela 3.11. Este tipo de canal apresenta como

características específicas o *status* do áudio, a opção de se exibir ou não o controlador de áudio e o volume inicial do áudio. As noções de *status* do áudio e de controlador de áudio são as mesmas utilizadas para vídeo. O volume inicial do áudio pode ser: normal, alto ou baixo.

Tabela 3.11 - Definição do canal áudio

Nome-canal	Canal-áudio-1			
Fonte-título	Arial			
Estilo-título	Normal	Negrito	Itálico	Negrito Itálico
		X		
Tamanho-fonte-título	16			
Posição-título	Centralizado	Direita	Esquerda	
	X			
Cor-fonte-título	Automática			
Status-áudio	Ativado			
Exibir-controlador	Sim			
Volume-inicial-áudio	Normal			

As páginas são associadas aos canais pela função "pl". Esta função é apresentada na tabela 3.12.

Tabela 3.12 - Função "pl"

Página - P		Canais - L
P	1	Canal-texto-1
P	2	Canal-texto-1
P	3	Canal-texto-1
P	4	Canal-texto-1
P	5	Canal-imagem-1
P	6	Canal-texto-2
P	7	Canal-texto-2
P	8	Canal-texto-2
P	9	Canal-texto-1

3.4. O PROCESSO DO MÉTODO HMBS/M

O método proposto tem como principal objetivo apoiar o projeto e o desenvolvimento estruturado de aplicações hipermídia. Ele é composto por quatro fases em que o produto de cada fase é incrementado ou melhorado na fase posterior. As melhorias introduzidas no(s) modelo(s) de uma das fases devem ser especificadas no(s) modelo(s) da(s) fase(s) anterior(es), permitindo assim um desenvolvimento iterativo. O método também pretende permitir a prototipação dos

modelos obtidos no fim de cada fase para facilitar a verificação dos requisitos.

A fase de especificação dos requisitos não faz parte do método mas é imprescindível. Essa fase consiste em obter do usuário todas as metas e objetivos que ele deseja que a aplicação satisfaça. O produto da fase de especificação é o documento de requisitos que servirá como base concreta para as fases posteriores. As fases do método são apresentadas na figura 3.10.

Figura 3.10 - Fases do método para o projeto de aplicações hipermídia

3.4.1. MODELAGEM CONCEITUAL

Esta fase tem como objetivo analisar o domínio da aplicação e representar a aplicação em modelos que ajudem a identificar as informações relevantes da aplicação hipermídia. O modelo conceitual é composto de dois modelos: o modelo de objetos e o modelo de fatias. O modelo de objetos permite modelar as informações do domínio da aplicação e os relacionamentos entre elas. As principais primitivas utilizadas no modelo de objetos são: classes, relacionamentos e atributos.

As classes do modelo de objetos geralmente surgem dos substantivos encontrados no documento de requisitos. No entanto, para que um substantivo dê origem a uma classe ele deve ter um conceito que seja importante para entender o domínio da aplicação. Objetos físicos, pessoas, organizações e abstrações, são todos possíveis candidatos a classes.

Os relacionamentos podem representar comunicação, associação física e ações. Geralmente os relacionamentos surgem dos verbos do documento de requisitos. A cardinalidade dos relacionamentos deve ser decidida após a definição do relacionamento. Uma maneira de se decidir a cardinalidade dos relacionamentos é esboçar um exemplo em termos de tuplas. Deve-se levar em consideração se um objeto deve aparecer no relacionamento e se pode haver um número fixo ou arbitrário de objetos correspondentes. Quanto a totalidade é necessário verificar se todos os objetos de uma certa classe devem participar do relacionamento.

Os atributos devem ser deixados para o final da construção do modelo de objetos. A representação de algo como um atributo exclui a adição de outras estruturas no modelo. Classes que não tenham atributo ou que sejam relacionadas somente com uma outra classe podem ser transformadas em atributos.

O modelo de objetos utiliza dois mecanismos de estruturação para simplificar a modelagem de aplicações hipermídia complexas. A agregação é utilizada para modelar os relacionamentos do tipo "parte de" ou o relacionamento "tem um". Geralmente a agregação pode ser usada para modelar restrição física e hierarquia. A agregação também pode ser usada para agregar um relacionamento, indicando que só os objetos do relacionamento fazem parte da agregação e para reduzir a complexidade de relacionamentos ternários.

A generalização é utilizada para modelar relacionamentos do tipo "tipo de" ou o relacionamento "é um". A generalização deve ser introduzida próximo ao final da construção do modelo de objetos. O uso prematuro de generalizações pode distorcer o modelo, introduzindo classes desnecessárias. Para se introduzir uma generalização num modelo de objetos deve-se examinar o modelo para identificar classes com propriedades comuns, tais como atributos

comuns ou participação em relacionamentos comuns. Os atributos e relacionamentos comuns podem ser abstraídos em objetos generalizados dos quais as classes originais são especializações.

Com base na análise do documento de requisitos é construído o modelo de objetos seguindo alguns critérios para a sua construção. Uma forma de se iniciar a construção do modelo de objetos é organizar listas de substantivos e verbos encontrados no documento de requisitos. Após a construção das listas, elas devem ser analisadas considerando somente os componentes que são realmente relevantes para a apresentação e explicação do domínio da aplicação. Esse processo de análise deve ser realizado até que a lista fique estável. Dependendo do domínio da aplicação pode-se dividir o domínio em partes coesivas e independentes para facilitar a análise.

Após a construção do modelo de objetos a última tarefa é determinar o escopo da aplicação a ser desenvolvida. Sendo assim, com base no documento de requisitos e em decisões tomadas durante a construção do modelo de objetos, é traçada a linha que engloba as classes, relacionamentos, generalizações e agregações que serão realmente implementadas nas fases seguintes.

Com o escopo da aplicação determinado o próximo passo é construir o modelo de fatias. O modelo de fatias modela somente as informações e os relacionamentos do domínio da aplicação que serão realmente implementados. As fatias agrupam um ou mais atributos de uma classe, segundo um determinado critério, como por exemplo:

- atributos que fazem sentido serem visitados/apresentados juntos;
- atributos muito longos;
- atributos apresentados por um mesmo tipo de mídia.

Sendo assim, atributos que se referem as características específicas da classe, tal como nome, código, idade, endereço, etc, geralmente são agrupados na fatia Principal. Atributos que são textos muito extensos ou gráficos e tabelas muito grandes devem ser agrupados em uma nova fatia. Já os atributos com mais de uma perspectiva devem dar origem a várias fatias, uma para cada perspectiva do atributo.

No modelo de fatias, a fatia Ligações agrupa os relacionamentos que ligam uma classe dentro do escopo da aplicação a uma classe fora do escopo. Deste modo, deve ser realizada uma análise do modelo de objetos e do documento de requisitos para determinar se alguns desses relacionamentos deve ser especificado e qual(ais) dele(s). Geralmente, os relacionamentos especificados são aqueles cuja classe fora do escopo da aplicação já foi implementada em outra aplicação. A fatia de Ligações pode ou não estar conectada as outras fatias da classe. Caso ela

não esteja conectada a outras fatias, suas informações são apresentadas em conjunto com as informações dessas fatias. Caso contrário suas informações são apresentadas individualmente.

3.4.2. MODELAGEM NAVEGACIONAL

Esta fase tem como objetivo reorganizar os modelos da fase anterior, considerando o perfil do usuário e as tarefas que o autor pretende apoiar. Durante esta fase são especificados os contextos de navegação e criados o modelo navegacional de tipos e o modelo navegacional de instâncias.

Os contextos de navegação definem várias formas de se navegar pela aplicação minimizando a desorientação do usuário. Geralmente é especificado um contexto do tipo índice para ser o ponto inicial da aplicação. Esse contexto é implementado como um menu que permite acessar os outros contextos da aplicação.

Geralmente as classes mais importantes devem dar origem a vários contextos que permitem acessar os seus objetos de formas diferentes. Tais contextos podem ser do tipo índice ou do tipo classe. Atributos que permitem selecionar um determinado número de objetos de uma classe segundo alguma condição pré-estabelecida devem dar origem a contextos do tipo derivado de atributo. Os relacionamentos devem dar origem a contextos do tipo derivado de elo. No caso de relacionamentos M:N o projetista deve decidir se o relacionamentos será implementado nos dois sentidos. Em caso afirmativo, como os relacionamentos desse tipo são divididos em dois relacionamentos 1:N, um contexto derivado de elo deve ser criado para cada um dos relacionamentos.

As agregações geralmente dão origem a contextos derivado de estruturas. Nesse caso deve-se considerar algumas opções com relação a sua navegação. No caso de agregações com cardinalidade definida pode-se navegar por elas especificando uma âncora que determina um caminho entre as classes agregadas ou várias âncoras, conforme a cardinalidade dessas classes. No caso de agregações com cardinalidade indefinida (* ou +), pode-se criar um novo nó contendo todas as âncoras que levam as classes agregadas ou essas âncoras podem ser especificadas no próprio nó em questão. Outra opção é especificar tudo em um nó só, eliminando assim a hierarquia imposta pela agregação.

A navegação em contextos de navegação aninhados pode ser um pouco mais complexa do que nos contextos normais, pois pode-se permitir que os contextos inclusos herdem elos dos contextos que os contêm. No caso de nós que participam de mais de um contexto de navegação, os elos que podem ser seguidos a partir de um nó dependem do caminho especificado no

contexto de navegação atual. Caso se deseje que os elos referentes a outros contextos possam ser seguidos, deve-se especificar isso na cláusula Importa/Exporta.

Alguns contextos de navegação só são definidos no modelo navegacional de instâncias devido à necessidade de se saber quais instâncias farão parte dele, como por exemplo o contexto arbitrário.

O modelo navegacional de tipos permite modelar os vários tipos de contextos de navegação definidos por estruturas de acesso. Neste modelo os relacionamentos são divididos em relacionamentos da aplicação e relacionamentos estruturais. Os relacionamentos da aplicação são os próprios relacionamentos do modelo de fatias. Tais relacionamentos, além de terem uma representação gráfica por estruturas de acesso, aparecem na fatia de Ligações como âncoras. Deste modo, fica mais fácil visualizar como esses relacionamentos aparecem nas páginas finais.

Os relacionamentos estruturais conectam as estruturas de acesso entre si, estas estruturas a classes e uma fatia a outra. Tais relacionamentos não aparecem na fatia de Ligações, pois apesar de serem uma âncora eles não permitem navegar para outra classe.

O modelo navegacional de tipos não difere muito dos modelos da fase anterior. No entanto, a partir deste modelo o projetista já tem especificado o ponto inicial para navegação na aplicação, como acessar as classes e como navegar de uma classe para outra.

O modelo navegacional de instâncias modela as instâncias das classes e dos relacionamentos do modelo navegacional de tipos em termos de estados e transições. A notação utilizada é apresentada na tabela 3.13. Utilizando *Statecharts*, como definido pelo HMBS, podem-se definir algumas regras para o mapeamento entre os dois modelos:

Tabela 3.13 - Notação utilizada

Estado
Transição
Estado Inicial

- uma classe dá origem a vários objetos que são mapeados para estados, como apresentado na figura 3.11;

Figura 3.11 - Mapeamento de classe

- as classes que possuem somente as fatias de Ligações e Principal dão origem a vários objetos que são mapeados para estados como apresentado na figura 3.12;

Figura 3.12 - Mapeamento de classes divididas em fatias

- no caso de generalizações/especializações, as classes especializadas dão origem a vários objetos que são mapeados para estados como apresentado na figura 3.13;

Figura 3.13 - Mapeamento de especialização/generalização

- as classes que possuem outras fatias além das fatias Principal e de Ligações dão origem a vários objetos que são mapeados para estados compostos por outros estados, conforme a quantidade de fatias. Essa situação é ilustrada na figura 3.14;

Figura 3.14 - Mapeamento de classes com outras fatias

- as classes formadas pela agregação de outras classes dão origem a vários objetos, que são mapeados para estados. Esses estados são decomposto em outros estados que representam as classes agregadas e assim sucessivamente. Em cada estado decomposto é criado um estado extra, representado o estado que foi decomposto. O estado extra é

especificado como estado inicial pois os estados que o compõem só são alcançados pela ativação de elos. Esse artifício também é usado para representar a ativação de um elo com origem no estado decomposto cujo destino é um dos estados que o compõem. Esse mapeamento é apresentado na figura 3.15;

Figura 3.15 - Mapeamento de classes agregadas

- índices, roteiros guiados indexados, roteiros guiados condicionais, índices condicionais e grupos são mapeados para estados como apresentado na figura 3.16;

Figura 3.16 - Mapeamento de índices e roteiros guiados indexados

- os roteiros guiados podem ou não ser mapeados para estados, ficando a critério do projetista. Quando o roteiro guiado não é mapeado para estado sua ativação parte do estado que contém a âncora que o ativa, como apresentado na figura 3.17;

Figura 3.17 - Mapeamento de roteiros guiados

- os relacionamentos da aplicação e relacionamentos estruturais representados no modelo navegacional de tipos pelas estruturas de acesso são mapeados para transições, como mostrado na figura 3.18.

Figura 3.18 - Mapeamento dos relacionamentos

Os contextos arbitrários são definidos no modelo navegacional de instâncias pela

especificação dos objetos que compõem. Os objetos podem pertencer a uma mesma classe ou a classes diferentes, como apresentado na figura 3.19.

Figura 3.19 - Contexto de navegação arbitrário

As ligações que especificam um roteiro guiado entre as instâncias de uma mesma classe surgem a partir deste modelo e são modeladas por intermédio de transições. O fato de que um nó pode participar de vários roteiros guiados e assim ter mais de um nó como destino acarreta um grande problema: o controle de navegação. Como a escolha do nó destino depende do roteiro guiado que se está seguindo, uma solução é criar *flags*, utilizando condições e variáveis lógicas. Desse modo, a manutenção do contexto de navegação é realizada pela criação de várias transições associadas ao mesmo evento P (próximo) e V (anterior). Sendo assim somente a transição associada ao *flag* ativo será disparada. Na figura 3.20 é apresentado um exemplo de dois roteiros guiados nos quais há a manutenção do contexto de navegação por intermédio de dois flags: c1 e c2. Nesses roteiros os caminhos são sequenciais e circulares, isto é, o usuário só sai do roteiro após acessar o último nó e a partir desse nó ele pode voltar ao nó inicial.

Figura 3.20 - Manutenção de contextos de navegação

3.4.3. MODELAGEM DA INTERFACE, IMPLEMENTAÇÃO E TESTE

A maioria dos métodos existentes para o projeto de aplicações hipermídia não utiliza modelos formais para especificar a interface homem-computador, utilizando ferramentas dependentes da implementação e do ambiente para realizar essa tarefa. Dessa forma é necessário que um modelo formal genérico seja utilizado para especificar a interface do usuário, garantindo uma maior independência e ajudando no reuso de componentes da interface. O uso de um modelo formal facilita os testes e a manutenção da aplicação.

Para especificar a interface é necessário definir suas propriedades estáticas e dinâmicas, tais como:

- a aparência interfacial de cada objeto navegacional que será percebido pelo usuário, isto é, a representação de seus atributos incluindo as âncoras;
- outros objetos de interface para oferecer as diversas funções da aplicação, como barras de menus, botões de controle e menus;
- os relacionamentos entre os objetos de interface e os objetos navegacionais, tal como o modo como um evento externo afeta a navegação;
- as transformações de interface ocorridas pelo efeito da navegação ou de eventos externos no comportamento de diferentes objetos da interface;
- a sincronização de alguns objetos de interface, principalmente quando vários tipos de mídia estão envolvidos.

O método proposto pretende definir somente as características genéricas da interface, isto é, as características que independem do ambiente de implementação. Desse modo, a definição de canais serve como base para determinar certas características sobre a aparência dos objetos na aplicação final. Essas características estão relacionadas à formatação de títulos e textos e a certos artefatos utilizados na interface, tais como: barras de rolagens e barras de controle para áudio e vídeo. O modelo navegacional de instâncias especifica, de certo modo, como a interação do usuário com a interface pode afetar a navegação. No entanto, as transformações de interface dependem dos objetos de interface (botões, textos e imagens ancorados, etc) disponíveis no ambiente de implementação e não são tratadas pelo método. A sincronização de determinados objetos da interface que utilizam diferentes tipos de mídia não são apresentadas neste trabalho mas pode ser definida pelo modelo XHMBS apresentado no capítulo 2.

Durante a implementação os modelos navegacional e de interface são mapeados em objetos concretos de um determinado ambiente, cujo resultado é a aplicação hipermídia

propriamente dita. Hoje em dia há muitos ambientes de implementação dos mais variados tipos mas não se pode utilizar o mesmo conjunto de regras para traduzir um projeto realizado no método proposto para uma aplicação em um desses ambientes. O primeiro passo a ser realizado nessa fase é escolher o ambiente de implementação. Uma vez escolhido o ambiente, o modelo navegacional de instâncias e as informações referentes à interface devem ser mapeados para artefatos do ambiente de implementação e todas as suas classes devem ser instanciadas, sendo associadas aos seus respectivos conteúdos. Para que o mapeamento seja realizado um conjunto de regras deve ser definido. Durante este trabalho foram identificados três possíveis tipos de implementação: interpretada, traduzida e de tradução livre. Esses tipos de implementação são apresentados mais detalhadamente nas próximas seções.

3.4.3.1. Implementação interpretada

A implementação interpretada requer um sistema capaz de interpretar e executar as especificações da aplicação sendo desenvolvida. Neste trabalho a implementação interpretada foi realizada com apoio do protótipo do sistema de autoria e navegação de hiperdocumentos chamada *HySCharts* (*Hyperdocument System based on Statecharts*). A figura 3.21 apresenta a tela inicial do *HySCharts*.

Figura 3.21 - Tela inicial do *HySCharts*

O sistema engloba um módulo de autoria no qual o *Statechart* que representa o modelo navegacional de instâncias é criado. A figura 3.22 apresenta a janela para criação de estados e a figura 3.23 apresenta a janela para criação de transições compostas que utilizam *flags* para controlar a navegação.

Figura 3.22 - Janela para criação de estados em HyScharts

Nesse mesmo módulo os estados e transições do modelo navegacional de instâncias são associados, de forma automatizada, às páginas e âncoras da aplicação. Essas associações são realizadas seguindo as funções de mapeamento apresentadas na seção 3.3.5. A criação de páginas envolve a determinação de um título para a página, do canal que a interpretará e do arquivo que contém o conteúdo que deve ser apresentado. A figura 3.24 apresenta a tela para a criação de páginas no módulo de autoria do *HySCharts*. Após a criação de uma página é realizada a associação dessa página ao seu respectivo estado.

No módulo de autoria é definido o nível de visibilidade (N) durante a navegação através da aplicação hipermídia. A figura 3.25 apresenta um exemplo no qual o nível de visibilidade é igual a zero. Na figura 3.26 o mesmo exemplo é apresentado com nível de visibilidade igual a um. Nesse último caso a página associada ao estado imediatamente acima na hierarquia, "laboratorio", é apresentada simultaneamente com as páginas associadas ao estado AND "intermedia".

Figura 3.23 - Janela para criação de transições compostas no *HyScharts*

Figura 3.24 - Tela para criação de páginas em *HyScharts*

Figura 3.25 - Exemplo de uma aplicação com N=0

Figura 3.26 - Exemplo de uma aplicação com N=1

Em *HySCharts* o controle de navegação e as estruturas hierárquicas foram facilmente implementados já que estas questões foram totalmente definidas no modelo navegacional de instâncias. No entanto, a especificação da interface da aplicação não apresenta muitas opções. A apresentação da aplicação no sistema é realizada por páginas e as âncoras apresentam uma única forma de apresentação, não sendo possível, por exemplo, associá-las a botões. A liberdade que o autor possui com relação à definição da interface está na especificação dos canais de apresentação. A figura 3.27 apresenta a janela para especificação dos canais e deve seguir as tabelas apresentadas na seção 3.3.6.

Create Channel

Channel Name : _____

Media Type ☒ Text ☐ Image ☐ Audio ☐ Video

Title Position ☒ Top

Height : 0 0 900

Width : 0 0 1152

Position X: 0 0 1152

Position Y: 0 0 900

Fonts

Font Name ☒ Default

Style ☒ FONT_STYLE_DEFAULT

Size ☒ Small

Fg/Bg ☒ Foreground ☐ Background

Foreground : _____

Background : _____

Colors ☒ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

☐ Scroll Base ☐ Scroll Right ☐ Resize

Status Audio ☒ On ☐ Off

Audio Output Port ☒ Main Speaker ☐ Headphones ☐ Line Out

Status Video ☒ On ☐ Off

☐ Controller ☐ Loop Presentation

Figura 3.27 - Janela para definição de canais

O sistema inclui um módulo de verificação no qual é possível realizar testes pela simulação da aplicação sendo desenvolvida. A tela principal do módulo de verificação é apresentada na figura 3.28 no qual o *Statechart* subjacente é apresentado simultaneamente com as páginas da aplicação. Assim, pode-se verificar se a semântica de navegação está correta e se todos os estados e transições estão associados às páginas e âncoras corretas.

Figura 3.28 - Módulo de verificação do HyCharts

A navegação em *HyCharts* é realizada por um *browser* que permite navegar pela aplicação hipermídia seguindo a semântica de navegação definida no modelo navegacional de instâncias e a configuração atual. Na figura 3.29 é apresentado o exemplo de uma aplicação no *browser* do *HyCharts*. A principal diferença do *browser* com relação ao módulo de verificação é que no *browser* são apresentadas somente as páginas com as informações e âncoras.

3.4.3.2. Implementação Traduzida

A implementação traduzida consiste em traduzir os modelos navegacional e de interface em artefatos de algum software de autoria de aplicações hipermídia como *Toolbook*, *Macromedia Director*, HTML (*Hypertext Language Markup*) e apresentar a aplicação final utilizando um *browser* como o *Netscape* e o *Internet Explorer*. A tela principal do *Netscape* é apresentada na figura 3.30. A utilização de um *browser* oferece algumas vantagens, tais como: armazenamento em tempo de execução das páginas visitadas pelo usuário durante a navegação,

armazenamento das páginas que um usuário acessa frequentemente, etc. Nesse tipo de implementação devem ser definidas regras para converter os objetos da especificação da aplicação em objetos do software de apresentação escolhido. A implementação traduzida pode ser automática ou manual.

Figura 3.29 - Exemplo de uma aplicação apresentada no *browser* do *HyCharts*

Nesse caso, a implementação da interface oferece um pouco mais de liberdade ao autor da aplicação. Dependendo do software de autoria escolhido pode-se utilizar botões, textos e imagens ancoradas, artefatos que permitem dividir a tela em várias janelas independentes, etc. As tabelas especificadas na seção 3.3.6 também são utilizadas na definição de certas características da interface. Os testes nesse tipo de implementação são geralmente executados navegando-se pela aplicação para verificar se a semântica de navegação foi implementada corretamente e se todas as páginas e âncoras foram implementadas. A figura 3.30 apresenta o exemplo de uma aplicação em HMTL visualizadas no *Netscape*.

Um dos problemas desse tipo de implementação é que o conjunto de regras de conversão nem sempre é o mesmo para todos os tipos de *software* de autoria. Além disso, certas características, como o controle de navegação e a organização hierárquica, não são facilmente implementadas.

Figura 3.30 - Exemplo de uma aplicação em HMTL no Netscape

3.4.3.3 - Implementação com tradução livre

A implementação com tradução livre não utiliza nenhum *software* de apresentação de aplicações hipermídia ou sistema de autoria de hiperdocumento. Esse tipo de implementação apresenta um interface própria, isto é, não utiliza nenhum *browser* para apresentação e navegação das informações. Um exemplo de aplicação hipermídia que utiliza esse tipo de implementação é o CD da *National Gallery* de Londres. Nesse caso, o próprio autor determina por quais artefatos de interface as informações serão apresentadas ao usuário e define para cada artefato da interface qual é a sua função e qual é a ação que ele deve disparar. Uma das opções para especificar esse tipo de interface é utilizar *Statechart* como apresentado em [Zhe92]. Outra opção é especificar as características da apresentação de cada entidade do domínio da aplicação por objetos visuais. Para cada objeto visual é definido um *Statechart* que especifica todas as ações associadas a cada artefato da interface e as transformações que ele causará. Uma abordagem para implementação da interface parecida com esta é apresentada em [Ros96]. Os testes nesse tipo de implementação são mais complexos e devem ter uma atenção especial do

projetista.

3.5 - CONSIDERAÇÕES FINAIS

O HMBS/M tem como principal característica a utilização do HMBS para especificar a semântica de navegação e a organização estrutural de uma aplicação hipermídia. A utilização do paradigma de orientação a objetos permite a especificação de aplicações hipermídia mais complexas de forma concisa e expressiva.

Neste capítulo foram apresentados inicialmente os principais tipos de aplicações hipermídia encontradas na atualidade. A seguir foram apresentados os modelos utilizados pelo método proposto, mostrando suas características, conceitos e representações. As fases do método são apresentadas a seguir com as diretrizes sobre como proceder na construção de cada modelo. Finalmente, foram discutidas as três opções de implementação identificadas neste trabalho para esse tipo de aplicação.

No capítulo seguinte é apresentado o estudo de caso utilizado para avaliar o método proposto.

CAPÍTULO 4

ESTUDO DE CASO: ESPECIFICAÇÃO DE UM CATÁLOGO DE CURSOS

4.1. CONSIDERAÇÕES INICIAIS

Neste capítulo é apresentado o estudo de caso utilizado para analisar o HMBS/M e a sua viabilidade para projetar aplicações hipermídia. O exemplo escolhido é bastante simples mas apresenta as características básicas desse tipo de aplicação.

O capítulo está organizado da seguinte forma: na seção 4.2 é apresentada uma descrição da aplicação utilizada como exemplo e na seção 4.3 é apresentada, fase a fase, todas as considerações feitas com relação ao projeto da aplicação e os modelos construídos. A seção 4.4 apresenta os dois tipos de implementação realizados neste estudo de caso.

4.2. DESCRIÇÃO DO PROBLEMA

A aplicação exemplo usada para análise e desenvolvimento do método proposto é baseada nos catálogos da Graduação e Pós-graduação do Instituto de Ciências Matemáticas de São Carlos - ICMSC. Ambos os catálogos têm como objetivo fornecer informações básicas relativas ao corpo docente, estrutura curricular dos cursos, conteúdos, formas de ingresso, disciplinas oferecidas, etc. Além disso, espera-se que os catálogos sirvam para orientação dos estudantes que ingressem em um dos cursos de graduação ou pós-graduação do ICMSC.

O catálogo da graduação é anual e fornece várias informações sobre:

- o ICMSC;
- o trancamento de matrícula;
- o ingresso no ICMSC;
- a avaliação do rendimento escolar e frequência;

- o sistema de recuperação;
- bolsas de estudo;
- estágios;
- o departamento de Computação e Estatística (SCE);
- o departamento de Matemática (SMA);
- a biblioteca Prof. Achille Bassi;
- a Comissão de Graduação (CG);
- a Comissão de Pós-Graduação (CPG);
- o Centro de Educação Física Esportes e Recreação (CEFER);
- a Universidade de São Paulo (USP);
- os principais laboratórios do ICMSC;
- os cursos oferecidos e a sua estrutura curricular;
- as disciplinas e o seu conteúdo programático;
- o corpo docente.

O catálogo da pós-graduação é semestral e apresenta informações sobre:

- os cursos de pós-graduação do ICMSC;
- as disciplinas da pós-graduação que serão oferecidas no semestre e o seu conteúdo programático.

Com base nos dois catálogos foram determinados os relacionamentos entre as informações disponíveis. Além disso, algumas informações que não constam da versão impressa dos catálogos da graduação e pós-graduação foram adicionadas para enriquecer a implementação, tais como:

- eventos e projetos de pesquisa do SCE, SMA e do ICMSC;
- publicações e projetos de pesquisa dos membros do corpo docente;
- alunos de graduação e pós-graduação que participam de projetos de pesquisa.

Na próxima seção a aplicação catálogo é apresentada seguindo cada fase do método proposto.

4.3. AS FASES DO MÉTODO

4.3.1. PROJETO CONCEITUAL

O projeto conceitual envolve a construção do modelo de objetos e do modelo de fatias com base nas informações do domínio da aplicação. O modelo de objetos da aplicação catálogo apresenta as informações citadas na seção 4.2 em termos de classes, relacionamentos e atributos.

Com base nessas informações o modelo de objetos foi desenvolvido contendo classes obtidas das próprias informações do catálogo, tais como: "Corpo Docente", "Laboratório", "Disciplina" e "Curso". Outras classes foram criadas para representar a reunião de certas informações apresentadas separadamente como a classe "Estrutura Organizacional" que reúne as informações sobre os departamentos, as comissões, a biblioteca, o CEFER, o ICMSC e a USP e a classe "Regras" que reúne as informações referentes a trancamento de matrícula, ingresso, avaliação, recuperação, etc. O modelo de objetos da aplicação apresenta algumas generalizações como, por exemplo, a classe "Disciplina" que é uma generalização das classes "Disciplina_graduação" e "Disciplina_pós", e as classes "Curso" e "Alunos". Nas generalizações especificadas nesse exemplo a classe super-tipo é uma classe abstrata, isto é, ela é especificada somente para reunir os atributos em comum que as classes sub-tipos apresentam e não possui nenhuma instância. Agregações foram utilizadas no modelo de objetos para representar a organização hierárquica de certas informações. A classe "CGraduação", por exemplo, é organizada em "Currículos", os "Currículos" são organizados em "Períodos" e os "Períodos" em "Disciplinas". Cada classe da aplicação catálogo possui os seus próprios atributos sendo que alguns deles apresentam várias perspectivas como o atributo "formação_acadêmica" da classe "Corpo Docente" e o atributo "descrição" da classe "Laboratório". Os relacionamentos foram especificados com base nas possíveis relações entre as classes criadas.

Após a construção do modelo de objetos é determinado o escopo da aplicação por uma linha tracejada. As classes "Aluno", "ProjPesquisa", "Evento" e "Publicações" estão fora do escopo da aplicação por já terem sido implementadas em outras aplicações. No entanto é possível acessar essas informações a partir da aplicação catálogo pois a aplicação apresentará elos que levam a essas informações. O modelo de objetos da aplicação catálogo é apresentado nas figuras 4.1 e 4.2 para facilitar a sua apresentação.

O modelo de fatias representa em termos de classes, relacionamentos e atributos somente as informações dentro do escopo da aplicação e agrupa os atributos de cada classe em fatias seguindo os critérios apresentados no capítulo 3.

Figura 4.1 – Modelo de objetos da aplicação catálogo

Considerando somente as classes que estão dentro do escopo definido e os seus atributos foram criadas as seguintes fatias:

- classe Corpo Docente: Principal, Formação-acadêmica-t, Formação-acadêmica-v, Áreas-pesquisa e Ligações;
- classe Disciplina: Principal e Bibliografia;
- classe Disciplina_pós: Principal;
- classe Estrutura Organizacional: Principal e Ligações;
- classe Laboratório: Principal, Principal-vídeo e Ligações;
- classe Curso: Principal;
- classe CGraduação: Principal;
- classe Currículo: Principal;

Figura 4.2 - Modelo de objetos da aplicação catálogo

- classe Período: Principal;
- classe Sub-área: Principal;
- classe Regra: Principal.

Para os atributos que possuem mais de uma perspectiva foi criada uma fatia para cada uma das perspectivas e alguns atributos foram repetidos em algumas das fatias. A fatia de Ligações agrupa os relacionamentos que ficaram fora do escopo definido e não aparece em todas as classes da aplicação. Os relacionamentos entre as fatias é mostrado no modelo por setas uni e bidirecionais. No modelo de fatias os relacionamentos que levam a classes fora do escopo determinado são apresentados na fatia de "Ligações" por uma âncora e serão na fase de implementação associados ao URL(*Uniform Resource Locator*) que contém essas informações. O modelo de fatias da aplicação catálogo é apresentado na figura 4.3.

4.3.2. PROJETO NAVEGACIONAL

O projeto navegacional envolve a definição dos contextos de navegação e a construção dos modelos navegacional de tipos e navegacional de instâncias. Foram definidos quarenta e nove contextos de navegação para a aplicação catálogo dos tipos índice, classe, derivado de atributo, derivado de elo e de seleção. Os contextos do tipo índice foram definidos principalmente como ponto inicial da navegação e para acessar as classes da aplicação. Contextos do tipo classe foram definidos para as principais classes da aplicação e contextos derivado de atributos foram definidos para as classes "Corpo Docente" e "CpósGraduação". Contextos derivado de elo foram definidos para a maioria dos relacionamentos encontrados na aplicação.

Os contextos derivados de estrutura só são especificados no modelo navegacional de instâncias. O contexto de seleção definido permite acessar as informações referentes a todas as classes e relacionamentos de uma aplicação e serve, em princípio, para qualquer tipo de aplicação. No entanto, sua implementação só é possível quando as informações do domínio da aplicação estão armazenadas em uma base de dados. A especificação de alguns contextos de navegação são apresentados nas tabelas 4.1, 4.2 e 4.3.

Cont. tabela 4.1 - Contexto de navegação do tipo índice

Inclui {outros contextos, relacionamentos, classes e objetos}	Corpo Docente	
Pontos de entrada	Todas as instâncias da classe Corpo Docente	
Caminho	-	
Comportamento	Controle passo a passo e acesso direto	Restrição parcial
Importa/Exporta	Importa os elos que representam os contextos de navegação derivados dos relacionamentos que envolvem os nós do contexto de navegação em questão.	
Comentários	-	

Tabela 4.2 - Contexto de navegação do tipo classe

Nome do Contexto	CN6 Corpo Docente de A .. Z	
Tipo do Contexto	Classe	
Atributos do Contexto	-	
Inclui {outros contextos, relacionamentos, classes e objetos}	Corpo Docente	
Pontos de entrada	Todas as instâncias de Corpo Docente ou a instância escolhida como 1ª	
Caminho	Sequencial (ordem c.tipo (c.nome, ascendente) $c \in \text{Corpo Docente}$), circular	
Comportamento	Controle passo a passo e acesso direto sequencial	Restrição parcial
Importa/Exporta	Importa os elos que representam os contextos de navegação derivados dos relacionamentos que envolvem os nós do contexto de navegação em questão.	
Comentários	Este contexto permite navegar por todos os docentes, pelo tipo de cada um ("Titular", "Associado", "Doutores" e "Assistentes") e pelo nome.	

Tabela 4.3 - Contexto de navegação derivado de atributo

Nome do Contexto	CN10a Corpo Docente "BD"
Tipo do Contexto	Derivado de Atributo
Atributos do Contexto	-
Inclui {outros contextos, relacionamentos, classes e objetos}	$\forall c \in \text{Corpo Docente} \mid \text{"BD"} \in c.\text{áreas_pesquisa}$
Pontos de entrada	Instância escolhida como 1ª
Caminho	Sequencial (ordem ascendente c.nome, $c \in \text{Corpo Docente}$), circular

Cont. Tabela 4.3 - Contexto de navegação derivado de atributo

Comportamento	Controle Passo a passo e seqüencial	Restrição parcial
Importa/Exporta	Importa os elos que representam os contextos de navegação derivados dos relacionamentos que envolvem os nós do contexto de navegação em questão.	
Comentários	-	

Após a especificação dos contextos de navegação o próximo passo é construir o modelo navegacional de tipos. O modelo navegacional de tipos mantém as classes, fatias, agregações e generalizações do modelo de fatias, entretanto, outros relacionamentos foram adicionados à fatia de "Ligações" como âncoras e os contextos de navegação foram representados pelas estruturas de acesso. A principal mudança do modelo navegacional de tipos em relação ao modelo de fatias é a representação dos contextos de navegação, os quais permite começar a navegação pela aplicação. Os contextos de seleção e derivado de estrutura não foram representados no modelo navegacional de tipos, assim como os contextos arbitrários. No caso da aplicação catálogo nem todos os relacionamentos do modelo de fatias foram representados em ambos os sentidos no modelo navegacional de tipos. A figura 4.4 apresenta o modelo navegacional de tipos para a aplicação catálogo.

O modelo navegacional de instâncias é representado por estados e transições. Como a aplicação apresenta uma quantidade muito grande de informações não foi possível representá-la em somente um *Statechart* devido a limitações impostas pela ferramenta *HyScharts*. Desse modo, foram criados vários *Statecharts* aninhados, um para cada classe da aplicação, mostrando as possíveis maneiras de se iniciar a navegação naquela classe tendo como origem outras classes e estruturas de acesso. A navegação na aplicação como um todo deve ser visualizada como a integração desses *Statecharts*. O modelo navegacional de instâncias da aplicação catálogo é apresentado na figura 4.5, no qual os estados "corpodocente", "disciplina", "curso", "lab" e "estruturaorganizacional" representam os *Statecharts* de cada uma dessas classes.

Nesse modelo são representados os relacionamentos que no modelo navegacional de tipos aparecem com setas unidirecionais. As instâncias das classes que podem ser acessadas por roteiros guiados foram conectadas pelas transições associadas aos eventos "p" e "v", determinando assim o caminho para cada roteiro guiado. As transições associadas ao evento "s" permitem que a partir de qualquer uma das instâncias de uma classe seja possível voltar ao índice

ou roteiro guiado pelo qual se começou a navegação. Tanto as transições associadas ao evento "s" como aquelas associadas aos eventos "p" e "v" utilizam *flags* condicionais para determinar qual é o próximo estado a ser alcançado. As transições associadas ao evento "v" que estão numerados (v1, v2, etc) permitem voltar ao estado pelo qual se acessou o estado atual.

Figura 4.5 - Modelo navegacional de instâncias da aplicação catálogo

O modelo navegacional de instâncias da classe "Estrutura organizacional" é apresentado na figura 4.6 e apresenta todas as instâncias dessa classe. Foram representadas todas as possíveis maneiras de se acessar as instâncias da classe em questão, seja por estruturas de acesso ou por outras classes. Os estados "c-entrada1", "c-entrada2", "c-entrada3" e "c-entrada4" foram criados para representar os possíveis contextos de navegação pelos quais se pode acessar as instâncias das outras classes que acessam as instâncias da classe em questão. Na figura 4.6 um exemplo dessa situação ocorre quando o estado "sma" é acessado a partir do estado "lem" pela transição "d". Nesse *Statechart* não foram utilizadas os eventos "p" e "v", pois não há nenhum roteiro guiado especificado para essa classe.

Figura 4.6 - Modelo navegacional de instâncias para a classe Estrutura Organizacional

A classe "Laboratório" segue o mesmo exemplo da classe "Estrutura Organizacional" sendo que para as instâncias dessa classe foram especificados dois caminhos para os roteiros guiados cn22a e cn22b. O componente "p" dos estados "inter", "labes", "labic", "lasd", "ldcc" e "lem" foram decompostos em dois estados "texto" e "video". O modelo navegacional de instâncias para a classe "Laboratório" é apresentado na figura 4.7.

Para o modelo navegacional de instâncias da classe "Disciplina" foram representadas somente as instâncias das classes sub-tipos "Disciplina-graduação" e "Disciplina-pós" já que a classe Disciplina é uma classe abstrata. As fatias "Principal" e de "Ligações" foram representadas pelos componentes "f" e "l" dos estados AND que representam as disciplinas. O componente "f" foi decomposto em dois estados "principal" e "bibliografia". O modelo navegacional de instâncias para a classe "Disciplina" é apresentado na figura 4.8 e apresenta as transições e20 e 20h, que representam os relacionamentos unidirecionais do modelo navegacional de tipos.

A classe curso segue o mesmo exemplo da classe "Disciplina" e somente as classe sub-tipos "Cgraduação" e "CpósGraduação" foram instanciadas. Como essas classes são formadas pela agregação de outras classes cada estado que representa as instâncias das classes "Cgraduação" e "CpósGraduação" foi decomposto em outros estados. Desse modo, o estado "bcc" foi decomposto em um estado extra e em mais dois estados "obrigatórias" e "optativas" que representam as instâncias da classe agregada "Currículo". O estado obrigatórias foi decomposto em um estado extra e em mais três estados "período1", "período2" e "períodon" que representam as instâncias da classe agregada "Período". O estado "período1" foi decomposto em um estado extra e em mais quatro estados "disci1", "disci2", "disci3" e "discin" que representam as instâncias da classe agregada "Disciplina-graduação". As decomposições realizadas no estado "obrigatórias" devem ser realizadas no estado "optativas" assim como as decomposições realizadas no estado "período1" devem ser realizadas em todos os estados que representam os períodos. Os outros estados "bmat" e "lmat" devem ser da mesma forma que o estado "bcc".

O estado "dcc" foi decomposto em um estado extra e em mais dois estados "computação" e "mat-comp", que representam as instâncias da classe agregada "Sub-area". O estado

"computação" foi decomposto em um estado extra e em mais três estados "disci1", "disci2" e "discin" que representam algumas instâncias da classe agregada "Disciplina-pós". O estado extra criado em cada decomposição é um artifício utilizado para que a ativação de uma transição no estado decomposto possa alcançar os estados que o compõe. O modelo navegacional de instâncias para a classe "Curso" é apresentado na figura 4.9 e as decomposições realizadas nos estados "dcc" e "bcc" são apresentadas na figura 4.10 e 4.11 respectivamente.

Figura 4.8 - Modelo navegacional de instâncias para a classe Disciplina

O modelo navegacional de instâncias para a classe "Corpo Docente" segue o mesmo exemplo dos modelos já apresentados. Foram representadas instâncias da classe "Corpo Docente", as estruturas de acesso que permitem acessar essas instâncias e estabelecidos os caminhos a serem percorridos para cada roteiro guiado definido para a classe. As fatias da classe foram mapeadas para estados, sendo que a fatia principal foi mapeada para um estado AND, pois possui um atributo do tipo imagem que deve ser apresentado concorrentemente com as outras informações da fatia. O estado "fa" que representa a fatia "formação acadêmica" foi

Figura 4.11 – Decomposição do estado “bcc”

Pode-se notar que os *Statecharts* que representam o modelo navegacional de instâncias de determinadas classes ficaram demasiadamente complexos e difíceis de se entender. Essa situação frequentemente ocorre na especificação de aplicações hipermídia devido a certas peculiaridades desse tipo de aplicação. Uma solução é utilizar estados parametrizados como em *hyperscharts*. Desse modo, os estados que apresentam repetição de certos padrões de comportamento são representados uma só vez. Um bom exemplo é o modelo navegacional de instâncias da classe “Corpo Docente” no qual os estados que representam as instâncias dessa classe são representados uma única vez e as transições que chegam ou partem desses estados utilizam variáveis de parametrização para determinar os estados origem ou destino.

Figura 4.12 - Modelo navegacional de instâncias para a classe Corpo Docente

Após a construção do modelo navegacional de instâncias foram especificadas as tabelas que representam: a associação dos estados dos *Statecharts* a páginas pela função "m", a associação das âncoras de cada página a eventos pela função "ae" e as informações de cada página. Mesmo que o modelo navegacional da aplicação catálogo tenha sido construído em vários *Statecharts* a especificação das tabelas foi realizada considerando o *Statechart* da aplicação catálogo como a integração de todos os *Statecharts*. Na tabela 4.4 é apresentada parte da função "m" para os estados da figura 4.5 e 4.6, na tabela 4.5 são apresentadas as informações (c, t, Anc_p) das páginas associadas na tabela 4.4 e na tabela 6 é apresentada parte da função "ae" para as âncoras das páginas apresentadas na tabela 4.5.

Tabela 4.4 - Função "m" para a aplicação catálogo

Estado	Página
Catalogo	$\emptyset$
cn17	P ₁

Estado	Página
catalogo-relacionamento	$\emptyset$
cn3	P ₂

Cont. tabela 4.4 - Função "m" para a aplicação catálogo

Corpodocente	Ø
Disciplina	Ø
Graduacao	Ø
Posgraduacao	Ø
Curso	Ø
Graduação	Ø
posgraduacao	Ø
laboratorio	Ø
Estruturaorganizacional	Ø

Estruturas	Ø
biblioteca	P ₃
cefer	P ₄
cg	P ₅
cpg	P ₆
icmsc	P ₇
sce	P ₈
sma	P ₉
usp	P ₁₀

Tabela 4.5 - Informações das páginas (c,t,Anc_p) da aplicação catálogo

Página	t	c	Anc _p
P ₁	-	Catálogo da Graduação e Pós graduação do ICMSC	{Índice de Corpo Docente, Índice de Cursos, Índice de Disciplinas, Índice de Estruturas, Índice de Laboratórios}
P ₂	-	Índice de Estruturas	{Biblioteca, Cefer, CG, CPG, ICMSC, SCE, SMA, USP, Sair}
P ₃	texto1	Biblioteca	{Sair}
P ₄	texto2	Cefer	{Sair}
P ₅	texto3	CG	{Sair}
P ₆	texto4	CPG	{Sair}
P ₇	texto5	ICMSC	{Componentes, Sair}
P ₈	texto6	SCE	{Cursos, Corpo Docente, Disciplinas, Laboratórios, Eventos, Projetos Pesq, Sair}
P ₉	texto7	SMA	{Cursos, Corpo Docente, Disciplinas, Laboratórios, Eventos, Projetos Pesq, Sair}
P ₁₀	texto8	USP	{Sair}

Tabela 4.6 - Função "ae" para a aplicação catálogo

Página	Âncora	Evento
P ₁	Índice do Corpo Docente	e1
P ₁	Índice de Cursos	e4
P ₁	Índice de Disciplinas	e2
P ₁	Índice de Estruturas	e3
P ₁	Índice de Laboratórios	e5
P ₂	Biblioteca	e3a
P ₂	Cefer	e3b
P ₂	CG	e3c
P ₂	CPG	e3d
P ₂	ICMSC	e3e
P ₂	SCE	e3f
P ₂	SMA	e3g
P ₂	USP	e3h

Página	Âncora	Evento
P ₇	Componentes	e26
P ₇	Sair	s
P ₈	Cursos	e24a
P ₈	Corpo Docente	e18a
P ₈	Disciplinas	e21a
P ₈	Laboratórios	e22a
P ₈	Eventos	-
P ₈	Projetos Pesq.	-
P ₈	Sair	s
P ₉	Cursos	e24b
P ₉	Corpo Docente	e18b
P ₉	Disciplinas	e21b
P ₉	Laboratórios	e22b

Cont. tabela 4.6 - Função "ae" para a aplicação catálogo

P ₂	Sair	v3
P ₃	Sair	s
P ₄	Sair	s
P ₅	Sair	s
P ₆	Sair	s

P ₉	Eventos	-
P ₉	Projetos Pesq.	-
P ₉	Sair	s
P ₁₀	Sair	s

4.3.3. PROJETO DE INTERFACE

O projeto de interface envolve a definição dos canais. Para a aplicação catálogo foram definidos um canal do tipo vídeo e dois canais do tipo texto por se tratar de uma aplicação na qual a maioria das informações apresentadas são textos. A especificação dos canais utilizados na aplicação catálogo são apresentadas nas tabelas 4.7, 4.8 e 4.9 respectivamente.

Tabela 4.7 - Especificação do canal texto Texto-1

Nome-canal	Texto-1			
Fonte-título	Arial			
Estilo-título	Normal	Negrito	Itálico	Negrito Itálico
		X		
Tamanho-fonte-título	14			
Posição-título	Centralizado	Direita	Esquerda	
	X			
Cor-fonte-título	Vermelho Escuro			
Fonte-texto	Times New Roman			
Estilo-texto	Normal	Negrito	Itálico	Negrito Itálico
	X			
Tamanho-fonte-texto	12			
Posição-texto	Centralizado	Direita	Esquerda	
	X			
Cor-fonte	Azul			
Cor-fundo	Cinza			
Scroll-base	Não			
Scroll-direita	Sim			

Tabela 4.8 - Especificação do canal texto Texto-2

Nome-canal	Texto-2			
Fonte-título	Times New Roman			
Estilo-título	Normal	Negrito	Itálico	Negrito Itálico
		X		
Tamanho-fonte-título	14			
Posição-título	Centralizado	Direita	Esquerda	
	X			
Cor-fonte-título	Vermelho Escuro			

Cont. tabela 4.8 - Especificação do canal texto Texto-2

Fonte-texto	Times New Roman			
Estilo-texto	Normal	Negrito	Itálico	Negrito Itálico
	X			
Tamanho-fonte-texto	12			
Posição-texto	Centralizado	Direita	Esquerda	
	X			
Cor-fonte	Automática			
Cor-fundo	Cinza			
Scroll-base	Não			
Scroll-direita	Sim			

Tabela 4.9 - Especificação do canal vídeo Vídeo-1

Nome-canal	Vídeo-1			
Fonte-título	Arial			
Estilo-título	Normal	Negrito	Itálico	Negrito Itálico
		X		
Tamanho-fonte-título	14			
Posição-título	Centralizado	Direita	Esquerda	
	X			
Cor-fonte-título	Automática			
Status-vídeo	Ativado			
Exibir-controlador	Sim			
Velocidade-vídeo	Normal			

Após a especificação dos canais, cada página é associada ao canal que irá interpretá-la pela função "pl". Parte da função "pl" para a aplicação catálogo é apresentada na tabela 4.1.

Tabela 4.10 - Função "pl" para a aplicação catálogo

Página	Canal
P ₁	texto-1
P ₂	texto-1
P ₃	texto-2
P ₄	texto-2
P ₅	texto-2
P ₆	texto-2
P ₇	texto-2
P ₈	texto-2
P ₉	texto-2
P ₁₀	texto-2

A aparência final da interface da aplicação depende da plataforma destino e só será definida durante a fase de implementação.

4.4. IMPLEMENTAÇÃO

Nesta fase o modelo navegacional de instâncias e as características da interface da aplicação definidas pelo canais foram implementados em *HyScharts* (implementação interpretada) e HTML (interpretação traduzida). A implementação com tradução livre não foi realizada para a aplicação catálogo. Nas seções seguintes são apresentadas, mais detalhadamente, as duas opções de implementação realizadas.

4.4.1. IMPLEMENTAÇÃO EM *HYSCARTS*

O sistema *HyScharts* apresenta dois módulos, o de autoria e o de verificação, como apresentado no capítulo 3. No módulo de autoria a primeira ação realizada foi especificar os canais que são utilizados para interpretar as páginas. Definidos os canais, foram criadas as páginas que são apresentadas na aplicação final e estas associadas aos estados que as representam no modelo navegacional de instâncias. Certos estados como os do tipo AND e aqueles utilizados para agrupar outros estados não foram associados a nenhum página.

Após todas as páginas necessárias terem sido criadas e associadas aos seus respectivos estados foram criadas as âncoras de cada página. A criação das âncoras envolve escolher no conteúdo de uma página a palavra ou texto ancorado que representa a âncora e associá-la a um evento que irá disparar uma determinada transição. Um exemplo do módulo de autoria do sistema é apresentado na figura 4.13.

A navegação através da aplicação é realizada no *browser* do *HyScharts*. O início da navegação começa pela página associada ao estado inicial e a escolha de uma das âncoras dessa página permite acessar outra(s) página(s) seguindo uma determinada transição. O *browser* apresenta uma janela de botões que permite ao usuário interagir com a aplicação, recebendo eventos externos. Esses eventos, ao serem interpretados, modificam a configuração atual do modelo navegacional de instâncias apresentado novas páginas e disponibilizando outras âncoras. Nessa janela estão disponíveis os botões:

- BACK: volta a página anteriormente visitada;
- UP: apresenta a página superior na hierarquia caso esta exista;
- DOWN: apresenta a página inferior na hierarquia caso esta exista;
- DISMISS: fecha a janela de botões;

- HVIEW: dispara a operação *Show-hview*.

Figura 4.13 - Módulo de autoria do HyScharts

A operação *Show-hview* permite a visualização da página imediatamente acima da página sendo visualizada. Após o botão *Hview* ser acionado os botões UP e DOWN tornam-se ativos, sendo possível descer ou subir na hierarquia. O exemplo utilizado para apresentar a execução da operação *Show-hview* apresenta um estado "bcc" que é decomposto nos estados "extra", "obrigatorias" e "optativas" e que apresenta nível de visibilidade igual a zero. Ao se acessar o estado "bcc" é apresentada a página associada ao estado "extra" como mostra a figura 4.14. Se a operação *Show-hview* for disparada a página associada ao estado "bcc" é apresentada em conjunto com uma das páginas associadas aos estados decompostos, como mostra a figura 4.15.

A figura 4.16 apresenta a página referente ao índice de Corpo Docente da aplicação catálogo. A figura 4.17 apresenta as páginas iniciais de uma das instâncias da classe "Corpo docente" no *browser* do HyScharts, com N=0, e a figura 4.18 apresenta o mesmo exemplo com

State: 'estra' Title: 'bcc_estra'

Bacharelado em Ciencia da Computacao	
Disciplinas <u>Obrigatorias</u>	178
Disciplinas <u>Optativas</u>	71
Disciplinas Excluidas	9
Total	256

Sair

State: 'bcc' Title: 'bcc'

Bacharelado em Ciencia da Computacao

Para o curso de Bacharelado em Ciencia da Computacao a estrutura curricular tambem e composta de disciplinas de responsabilidade dos departamentos acima citados, compreendidos em 10 semestres, sendo os dois ultimos semestres reservados para estagio. Sao oferecidas anualmente 45 vagas. A coordenacao didatica dos cursos e feita pela Comissao de Graduação (CG).

Decomposition of State 'bcc'

Unfold Level:

```

graph TD
    bac((bac)) --> ob[obrigatorias]
    bac --> op[optativas]
    ob --> op
    
```

Figura 4.15 - Exemplo após o acionamento do botão *Hview*

Figura 4.16 - Índice do Corpo Docente no *browser* do HyScharts

Figura 4.17 - Exemplo de uma das instâncias da classe Corpo Docente (N=0)

Figura 4.18 - Exemplo de uma das instâncias da classe Corpo Docente (N=1)

Na figura 4.19 é apresentada a página referente a um dos roteiros guiados definidos para a classe "Corpo Docente". Como apresentado no capítulo 3, o controle de navegação é realizado pelo uso de *flags*. No exemplo da figura 4.19 quando a âncora "ATIVAR" é selecionada, a variável *boolean* "cn6" torna-se verdadeira. Desse modo, quando as âncoras "Próximo" ou "Anterior" forem selecionadas somente as transições "p" ou "v" cujo *cn6*=true são disparadas, levando à próxima instância daquele contexto de navegação.

4.4.2 . IMPLEMENTAÇÃO PARA A WWW (WORLD WIDE WEB)

A implementação para a WWW foi realizada em HTML com nível de visibilidade $N=0$. Hoje em dia existem muitos softwares que permitem desenvolver páginas em HTML, tais como: WORD97, Netscape Composer, FrontPage, AOLpress, etc. Para a implementação da aplicação catálogo foi utilizado o WORD97 e o AOLpress, como *browser* foi utilizado o *Netscape Communicator*.

Figura 4.19 - Exemplo de um roteiro guiado indexado em HyScharts

Com base no modelo navegacional de instâncias foi criada uma página HTML para cada estado desse modelo que representa alguma informação do domínio da aplicação. A figura 4.20 apresenta a página referente ao índice de Corpo Docente da aplicação catálogo.

Os estados do tipo AND foram mapeados para páginas divididas em *frames*. *Frames* são janelas individuais nas quais se pode carregar informações de diferentes arquivos ou URL's. No caso da aplicação catálogo foram utilizados *frames* para as classes que possuem outras fatias além das fatias "Principal" e de "Ligações". Sendo assim, foi criado um *frame* no qual é carregada a fatia "Principal" e as outras fatias da classe e um outro *frame* para carregar a fatia de "Ligações". Desse modo, as âncoras da classe sempre estão disponíveis independentemente da fatia onde se esteja. Os estados decompostos foram implementados como páginas normais e nesse caso não foi necessário a implementação dos estados "extra" criados no modelo navegacional de instâncias. As âncoras em HTML foram implementadas como botões ou textos e imagens ancoradas. As imagens, bem como as fotos, foram somente adicionadas à página da

qual fazem parte, não sendo necessário criar outra página só para sua apresentação como no *HySCharts*. Um exemplo é apresentado na figura 4.21.

Figura 4.20 - Índice do Corpo Docente

O controle de navegação não foi implementado em HMTL, pois a linguagem não oferece artefatos para isso. Sendo assim, foram utilizados programas desenvolvidos em C e CGI (*Commun Gateway Interface*). CGI é a forma de comunicação que um servidor Web emprega para enviar informações entre o *browser* cliente e um programa de computador em um servidor Web. Na aplicação catálogo foi utilizado CGI nas páginas que participam de roteiros guiados predefinidos. A figura 4.22 apresenta a página em HMTL do roteiro guiado indexado definido pelo contexto de navegação CN6 para a classe "Corpo Docente". Quando o usuário acessa um roteiro guiado, indexado ou não, é gerado um formulário que contém os dados referentes àquele contexto de navegação. No exemplo apresentado na figura 4.22 tal situação ocorre quando aquela página é acessada. Quando o botão "ATIVAR ROTEIRO" é selecionado o programa CGI utiliza os dados do formulário para saber qual página deve ser construída. Para atualização e manutenção do caminho que será seguido pelo usuário naquele contexto de navegação foram

utilizados *cookies* (variáveis de memória). Um *cookie* contém os códigos de todas as páginas acessadas durante o roteiro guiado, ordenadas pela ordem de acesso. Sendo assim, quando o botão "Próximo" é selecionado, a URL da próxima página é adicionada ao *cookie*. Se o botão "Anterior" é selecionado, a URL da página atual é retirada da *cookie*.

Figura 4.21 - Instância da classe Corpo Docente acessada a partir de um roteiro guiado

Como as páginas acessadas por roteiros guiados também podem ser acessadas por índices foi utilizado o símbolo "\$" para especificar quando ela foi acessada por um índice. Desse modo, o programa CGI sabe que não necessita utilizar *cookies* para armazenar as informações referentes às páginas acessadas. A figura 4.23 apresenta a mesma página apresentada na figura 4.21 quando esta é acessada por um índice. Nesse caso os botões "Próximo" e "Anterior" não aparecem na página.

Como neste tipo de implementação o nível de visibilidade N é fixo, isto é, para cada valor de N deve ser feita uma implementação diferente, a função *Show-hview* não foi implementada. Tal situação apresenta uma desvantagem da implementação traduzida com relação a

implementação interpretada, na qual a definição de N pode mudar de acordo com a vontade do usuário.

Figura 4.22 - Exemplo de um roteiro guiado indexado em HTML

4.4.3 - IMPLEMENTAÇÃO COM TRADUÇÃO LIVRE

A implementação com tradução livre não foi realizada para este estudo de caso. No entanto, um exemplo deste tipo de implementação é apresentado a seguir. Considerando a classe "Corpo Docente" da aplicação Catálogo propõem-se nas figuras 4.24, 4.25, 4.26 e 4.27 uma possível apresentação para os objetos visuais dessa classe. Os objetos visuais definem como cada informação da classe em questão é apresentada, isto é, em qual posição da tela as informações são mostrada e como elas são mostradas (botões, textos ancorados, etc). No caso da classe "Corpo Docente" foi definido um objeto visual para cada fatia da classe e para cada perspectiva de um certo atributo. Definidos os objetos visuais da classe, o próximo passo deve ser definir o *Statechart* que especifica todas as ações associadas a cada artefato da interface e as transformações que ele causará. O *Statechart* para a classe "Corpo Docente" é apresentado na figura 4.28. A tabela 4.11 apresenta os métodos disparados pelos eventos da figura 4.28. O

diagrama de configuração da figura 4.29 apresenta a interação dos objetos visuais da classe "Corpo Docente" com o próprio objeto Corpo Docente. Os métodos dos objetos visuais mudam a configuração atual da interface enquanto os métodos do objeto da aplicação obtêm as informações necessárias para essa nova configuração.

Figura 4.23 - Instância da classe Corpo Docente acessada a partir de um índice

Figura 4.24 - Objeto visual para fatia Principal da classe Corpo Docente

Corpo Docente		
<u>Formação Acadêmica</u> <u>Informações Gerais</u>	<u>Áreas de pesquisa</u> Nome: cadeia Áreas de pesquisa: texto	
<u>Publicações</u>	<u>Projetos de Pesquisa</u>	<u>Disciplinas</u>
<u>Departamento</u>		
Sair	Próximo	Anterior

Figura 4.25 - Objeto visual da fatia Áreas de Pesquisa da classe Corpo Docente

Corpo Docente		
<u>Áreas de Pesquisa</u> <u>Informações Gerais</u>	<u>Formação acadêmica vídeo</u> Nome: cadeia Formação acadêmica: vídeo	Controle do vídeo
<u>Publicações</u>	<u>Projetos de Pesquisa</u>	<u>Disciplinas</u>
<u>Departamento</u>		
Sair	Próximo	Anterior

Figura 4.26 - Objeto visual para a perspectiva vídeo da fatia Formação Acadêmica da classe Corpo Docente

Corpo Docente		
<u>Áreas de Pesquisa</u> <u>Informações Gerais</u>	Formação acadêmica texto Nome: cadeia Formação acadêmica: texto	
<u>Publicações</u>	<u>Projetos de Pesquisa</u>	<u>Disciplinas</u>
<u>Departamento</u>		
Sair	Próximo	Anterior

Figura 4.27 - Objeto visual para a perspectiva texto da fatia Formação Acadêmica da classe Corpo Docente

4.5. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o estudo de caso utilizado para ilustrar e avaliar o HMBS/M. A aplicação usada para o estudo de caso é baseada nos catálogos da Graduação e Pós-graduação do Instituto de Ciências Matemáticas de São Carlos - ICMSC. Apesar da aplicação catálogo ser relativamente simples, foi possível explorar todas as características oferecidas pelo método. Além disso, o uso do método mostrou que ele é fácil de ser utilizado e bastante intuitivo, facilitando a comunicação entre o autor da aplicação e seus usuários. Algumas dificuldades foram encontradas no decorrer da implementação. Na implementação em *HyCharts* a maior dificuldade encontrada foi com relação a impossibilidade de representar toda a aplicação catálogo em somente um *Statechart*. Devido a essa limitação do protótipo não foi possível implementar de maneira satisfatória os relacionamentos entre as instâncias de classes diferentes. Além disso, algumas instâncias de certas classes tiveram que ser repetidas em *Statecharts* diferentes. Em HMTL a maior dificuldade encontrada foi com relação à implementação dos contextos de navegação que definem roteiros guiados. Como HTML é uma linguagem que oferece TAGS somente para apresentação de dados foi necessário o uso de CGI e programas em C para implementar essa característica da aplicação. O uso de CGI torna a aplicação um pouco mais complexa mas no caso da aplicação catálogo não interferiu no seu desempenho. No próximo capítulo são apresentadas as contribuições deste trabalho e as possibilidades de trabalhos futuros.

Figura 4.28 - Statechart para a classe Corpo Docente

Tabela 4.11 – Métodos disparados pelos eventos da figura 4.28

Métodos
fa, fal: mostrar-página-formaçãoacadêmica
ap, apl: mostrar-página-areaspesquisa
p, pl: mostrar-página-infogerais
v: tornar-disponível-vídeo-formaçãoacadêmica
l: executar-vídeo-formaçãoacadêmica
d: desligar-vídeo-formaçãoacadêmica
t: mostra-texto-formaçãoacadêmica
s: mostrar-página-inicial
p: mostrar-próxima-página-roteiro
v: mostrar-página-anterior-roteiro
on: ativar-botão-próximo/anterior
off: desativar-botão-próximo/anterior
departamento: mostrar-página-departamento
disciplina: mostrar-índice-disciplinas
projetos: mostrar-índice-projetos
publicações: mostrar-índice-publicações

Figura 4.29 - Diagrama de Configuração

CAPÍTULO 5

CONCLUSÃO

5.1. CONSIDERAÇÕES INICIAIS

Neste trabalho foi apresentado um método orientado a objetos para apoiar o projeto e o desenvolvimento de aplicações hipermídia. O HMBS/M possui como principal característica o uso do HMBS (*Hypertext Model Based on Statecharts*) para especificar a estrutura organizacional e a semântica de navegação de um aplicação hipermídia. Certos conceitos utilizados em outros métodos e modelos foram incorporados ao HMBS/M de modo a facilitar o projeto e o desenvolvimento de aplicações hipermídia mais complexas. Neste capítulo são apresentadas as conclusões gerais deste trabalho, mostrando suas principais contribuições e as sugestões para trabalhos futuros.

Este trabalho apresentou a definição de um método para o projeto de aplicações hipermídia que utiliza de forma prática o HMBS. O HMBS é um modelo formal baseado em *Statecharts* que permite especificar a estrutura e o comportamento de aplicações hipermídia (Capítulo 2). No entanto, o modelo não especifica uma aplicação hipermídia de forma completa pois não modela a semântica do domínio da aplicação. Sendo assim, foi necessária a definição de um processo mais abrangente no qual ele possa ser incluído.

O HMBS/M é composto por quatro fases: modelagem conceitual, modelagem navegacional, modelagem de interface e implementação. Em cada fase os modelos da fase anterior podem ser incrementados ou melhorados, permitindo assim um desenvolvimento iterativo. O método também pretende permitir a prototipação dos modelos obtidos no fim de cada fase para facilitar a verificação dos requisitos. Durante a modelagem conceitual o domínio da aplicação é analisado e representado por dois modelos: o modelo de objetos e o modelo de fatias. Ambos os modelos utilizam alguns conceitos da orientação a objetos, o que facilita a estruturação de aplicações mais complexas. A modelagem navegacional tem como objetivo reorganizar os modelos da fase anterior, considerando o perfil do usuário e as tarefas que o autor pretende apoiar. Durante esta fase são definidos os contextos de navegação e construídos dois

modelos: o modelo navegacional de tipos e o modelo navegacional de instâncias. A definição dos contextos de navegação ajuda a minimizar a desorientação do usuário, que é um dos grandes problemas encontrados no projeto de aplicações hipermídia. Além disso, o uso do HMBS para construção do modelo navegacional de instâncias permite especificar de modo mais completo e consistente a semântica de navegação da aplicação sendo desenvolvida. Regras foram definidas ajudar o projetista a mapear as primitivas utilizadas no modelo navegacional de tipos para as primitivas utilizadas no modelo navegacional de instâncias. A modelagem da interface tem como objetivo definir as propriedades estáticas e dinâmicas da interface. No HMBS/M somente as características genéricas da interface, isto é, aquelas que independem do ambiente de implementação escolhido, são definidas. O HMBS/M utiliza canais para definição dessas características. Na implementação os modelos navegacionais e de interface são mapeados para artefatos do ambiente de implementação escolhido, resultando na própria aplicação hipermídia. Neste trabalho foram identificadas três tipos de implementação: interpretada, traduzida e de tradução livre. Na implementação interpretada é necessário um sistema capaz de interpretar e executar as especificações da aplicação sendo desenvolvida. A implementação traduzida consiste em traduzir os modelos de navegação e de interface em artefatos do ambiente de implementação escolhido. A implementação com tradução livre não utiliza nenhum *software* de apresentação de aplicações hipermídia ou sistema de autoria de hiperdocumento. Esse tipo de implementação apresenta um interface própria, isto é, não utiliza nenhum *browser* para apresentação e navegação das informações (Capítulo 2).

Para ilustrar e validar o HMBS/M foi utilizado um estudo de caso apresentado no capítulo 4. A aplicação exemplo foi implementada de modo interpretado e de modo traduzido. No modo interpretado a aplicação foi implementada em HyCharts, o protótipo do sistema de autoria e navegação de hiperdocumentos desenvolvido por pesquisadores do ICMSC [Tur98]. No modo traduzido a aplicação foi implementada em HMTL para disponibilização na WWW. Foi possível verificar que os modelos utilizados pelo HMBS/M são mais gerais e facilitam a implementação da aplicação hipermídia independentemente do ambiente escolhido.

5.2. CONTRIBUIÇÕES

O método proposto é uma contribuição original e tem como relevância o uso do modelo HMBS de forma prática para projetar aplicações hipermídia. Com o HMBS incluso dentro de um processo mais abrangente para apoiar o projeto e o desenvolvimento de aplicações

hipermídia foi possível especificar todas as características desse tipo de aplicação de forma completa e concisa.

Em especial, o uso de *statecharts* para modelar a navegação (MNI) em aplicações hipermídia é pioneira na área. Com o uso de *statecharts* foi possível modelar concorrência na apresentação de múltiplas unidades de informação e hierarquia, além de controlar a navegação pelo uso de *flags*. As regras de conversão do MNT para o MNI também são uma contribuição para facilitar a atividade de modelagem navegacional de aplicações hipermídia.

5.3. TRABALHOS FUTUROS

O método proposto pode ser utilizado para o projeto de outros tipos de aplicações hipermídia, como sistemas de aprendizado e de suporte a decisão. Além disso, deve-se considerar o uso de banco de dados para armazenamento das informações da aplicação hipermídia. O uso de banco de dados facilitaria a implementação dos contextos de seleção no qual o usuário especifica sua própria consulta. Nesse caso é necessário o estudo de como representar elos dinâmicos, isto é, elos criados em tempo de execução, pois o HMBS não permite a especificação desse tipo de elo, já que *Statecharts* são estáticos.

Outro ponto a ser considerado, é a criação de uma ferramenta automatizada para apoiar a construção de todos os modelos definidos no HMBS/M. Atualmente, o modelo de objeto e o modelo navegacional de instâncias podem ser definidos com o apoio do computador. O modelo de objetos pode ser definido na ferramenta *FusionCase* já que ele utiliza alguns conceitos do método *Fusion*. Já o modelo navegacional de instâncias pode ser definido no *HyScharts*. A definição de regras para o mapeamento automatizado ou manual de especificações em HMBS para linguagens como HMTL e Java ou sistemas como Director e Toolbook são propostas de trabalhos que podem ser realizados no futuro. Atualmente, regras para a conversão para HTML vêm sendo definidas por Shibuya [Shi98].

Finalmente, o uso do XHMBS para especificação de requisitos temporais e de sincronização de informações multimídia deve ser estudado. A incorporação do XHMBS tornaria o método proposto mais poderoso e completo, pois seria possível especificar todas as características inerentes às aplicações hipermídia.

REFERÊNCIAS

- [Aug97] Augusto, A. 97, O ano da Intranet. *Byte Brasil*, p. 31-34, Março 1997.
- [Bar95] Barbosa S.D.J. Modelagem e especificação da navegação em aplicações hipermídia. *Dissertação de Mestrado*, Departamento de Informática Pontifícia Universidade Católica do Rio de Janeiro, 110p., Fevereiro 1995.
- [Bel97] Belasubramanian, P. *State of the Art Review on Hypermedia Issues and Applications*, (http://www.isg.sfu.ca/~duchier/misc/hypertext_review), 1997
- [Bel94] Belasubramanian, P., Isakowitz T., Stohr E. A. designing hypermedia applications, In: *Proceedings of the 27th International Conference on System Sciences*, Maui, Hawaii, January 4-7, New York, IEEE Press, p.354-365, 1994.
- [Cas97] Castro, M. A S. Tutorial HTML, *ICMSC*, Abril 1997.
- [Col94] Coleman, D.; Arnold, P.; Bodoff, S.; Dollin, C.; Gilchirst, H.; Hayes, F.; Jeremaes, P. *Object-oriented development: The Fusion Method*, New Jersey, USA, Prentice Hall, Englewood Cliffs, 313p., 1994.
- [Dia95] Diaz, A.; Isakowitz, T. Rmcase: computer-aided support for hypermedia design and development. In: *Proceedings of the Second International Workshop on Hypermedia Design*. Montpellier, France, Springer Verlag, June 1995.
- [Gar91] Garzotto, F.; Paolini, P.; Schwabe, D. HDM: a model for the design of hypertext applications. In: *Proceedings Hypertext'91, III ACM Hypertext Conference*, San Antonio, TX, December 15-18, p.313-328, 1991.
- [Gar93] Garzotto, F.; Paolini, P.; Schwabe, D. HDM: a model-based approach to hypertext application design. *ACM Transactions on Information Systems*, v.11, n.1, p.1-26, January 1993.
- [Hal94] Halasz, F.G.; Schwartz, M. The dexter hypertext reference model. *Communications of the ACM*, v.37, n.2, p.30-39, February 1994.
- [Har93] Hardman L.; Bulterman D.C.A.; Rossum G.V. The Amsterdam hypermedia model: extending hypertext to support real multimedia, *Technical Report*, 1993.

- [Har94] Hardman L.; Bulterman D.C.A.; Rossum G.V. The Amsterdam hypermedia model: adding time and context to the Dexter model. *Communications of the ACM*, vol. 37, n. 2, p. 50-62, February 1994.
- [Hae87] Harel, D. *Statecharts: A visual formalism for complex systems*. *Science of Computer Programming*, v.8, n.3, p. 231-274, July 1987.
- [Isa97a] Isakowitz, T., Kamis, A., Koufaris, M. Extending the capabilities of RMM: Russian Dolls and hypertext. In: *Proceedings of the 30th Annual Hawaii International Conference on System Sciences*, HICSS'97, 1997.
- [Isa97b] Isakowitz, T., Kamis, A., Koufaris, M. Reconciling top-down and bottom-up design approaches in RMM. Technical Report, (<http://eldorado.stern.nyu.edu/~tisakowi/rmm/>), 1997
- [Isa95] Isakowitz, T.; Stohr, E.A.; Balasubramanian, P. RMM: a methodology for structured hypermedia design. *Communications of the ACM*, v.38, n.8, p.34-44, August 1995.
- [Lan93] Lange D.B. Object-oriented hypermodeling of hypertext supported information systems. *IEEE Computer*, p. 380-389, 1993.
- [Lan94] Lange D.B. An object-oriented design method for hypermedia information systems, In: *Proceedings of 27th International Conference on System Sciences*, Maui, Hawaii, New York, IEEE Press, p.366-375, 1994.
- [Nan95] Nanard, J.; Nanard, M. Hypertext design environments and the hypertext design process. *Communications of the ACM*, v.38, n.8, p.49-56, August 1995.
- [Oli97] Oliveira, M.C.F.; Turine, M.A.S.; Masiero, P.C. A *Statechart*-based model for hypertext. Relatório Técnico. Instituto de Ciências Matemáticas de São Carlos, Fevereiro 1997, 38p.
- [Pau97] Paulo, F.B. Especificação de aplicações hipermídia baseada em *Statecharts*, Dissertação de Mestrado, ICMSC, São Carlos, 1997.
- [Ram97] Ramalho, J.A. Iniciando em HTML, *Makron Books*, São Paulo 1997.
- [Ros94] Rossi, G.; Schwabe, D.; Lucena C.J.P.; Cowan D.D. An object-oriented model for designing the human-computer interface of hypermedia applications, Relatório Técnico, Departamento de Informática Pontifícia Universidade Católica, Rio de Janeiro, 19p., 1994.
- [Ros96] Rossi, G. Um método orientado a objetos para o projeto de aplicações hipermídia. Tese de Doutorado, Departamento de Informática Pontifícia Universidade

Católica, Rio de Janeiro, 1996.

- [Sch92] Schwabe, D; Caloini, A.; Garzotto, F.; Paolini, P. Hypertext development using a model-based approach. *Software-Practice and Experience*, v.22, n.11, p.937-962, November 1992.
- [Sch94a] Schwabe, D.; Rossi G. From domain models to hypermedia applications: an object-oriented approach. Relatório Técnico, Departamento de Informática Pontifícia Universidade Católica, Rio de Janeiro, 17p., 1994.
- [Sch94b] Schwabe, D; Barbosa, S.D.J. Navigation modelling in hypermedia applications, Relatório Técnico, Departamento de Informática Pontifícia Universidade Católica, Rio de Janeiro, 21p., 1994.
- [Sch96] Schwabe, D; Rossi, G.; Barbosa, S.D.J. Systematic hypermedia applications design with OOHDM. In: *Proceedings of Hypertext'96, VII International ACM Hypertext Conference*, Washington, DC, p.116-128, March 1996.
- [Shi98] Shibuya, R. Geração automática de especificações HMBS em HTML. Dissertação de Mestrado. ICMSC-USP, São Carlos, 1998 (em adamento).
- [Thü95] Thüring, M.; Hannemann, J.; Haake, J.M. Hypermedia and cognition: designing for comprehension. *Communications of the ACM*, v.38, n.8, p.57-66, August 1995.
- [Tom97] Tomas, M. et al Programando em Java para Internet, *Makron Books*, São Paulo, 1997.
- [Tur97]] Turine, M.A.S.; Oliveira, M.C.F; Masiero, P.C. *A Navigation-oriented hypertext model based on Statecharts*. In: *Proceedings of Hypertext 97*, Southampton, UK, p. 102-111, April 1997.
- [Tur98] Turine, M.A.S. HMBS: Um modelo baseado em *Statecharts* para a especificação formal de hiperdocumentos. Tese de Doutorado. IFSC-USP, São Carlos, 1998.
- [Zhe92] Zheng, Y.; Pong, M. Using *Statecharts* to model hypertext. In: *Proceedings of the European Conference on Hypertext Technology, ACM ECHT Conference*, Milano, November 30 - December 4, p. 242-250, 1992.
- [Wei97] Weinman, W.E. Manual do CGI, *Makron Books*, São Paulo, 1997.

APÊNDICE A

Definição Formal do Modelo XHMBS

A definição completa de um hiperdocumento pelo XHMBS é definida como segue:

Seja *Info* o conjunto das porções básicas do conteúdo da informação de um hiperdocumento. Dessa forma, um hiperdocumento é definido como uma 11-tupla $H = \langle Hyp, P, Anc, M, A, L, Pch, Cch, V_p, V_a, Ch \rangle$ em que:

- *Hyp* é um *hyperchart* definido como uma 12-tupla $\langle S, \rho, \Psi, \delta, H, \tau, LSC, T_{hist}, V, C, E, T, Ac, T_s \rangle$, representando, respectivamente, estados, função de hierarquia, função de tipo de decomposição, função “default”, conjunto de símbolos de história, função de *história temporal*, função de *relógios locais*, função de *registradores de história temporal*, conjunto de expressões, conjunto de condições, conjunto de eventos, conjunto de transições, conjunto de ações e conjunto de *transições M:N sincronizadas*.
- $P: Info^2$ denota o conjunto de páginas da forma $\langle c, t \rangle \mid c, t \in Info$ representando, respectivamente, o conteúdo e o título de uma página. Como definido no modelo HMBS, o conjunto *P* inclui a página nula, sem qualquer conteúdo ou título. Essa página pode ser utilizada com estados que não modelam a apresentação da informação, funcionando como artifícios de especificação, modelando a passagem do tempo, por exemplo.
- $Anc \subset Info$ é o conjunto de âncoras do hiperdocumento. Uma âncora é uma unidade de informação que difere de uma página por, normalmente, conter uma porção reduzida de informação, como uma palavra, por exemplo.
- $M: S_s \leftrightarrow P$ é uma relação que associa estados do subconjunto S_s , definido a seguir, a páginas do hiperdocumento. $S_s: \{x \in S \mid \psi(x) = OR \vee \rho(x) = \emptyset\}$

A definição da relação *M* da forma $S_s \leftrightarrow P$ é baseada na notação do método Z [Spi92].

A mesma definição pode ser escrita como $M: 2^{S_s} \times P$

- $A: E \times T \times P \rightarrow Anc$ é a função que permite associar uma âncora a cada evento contido no rótulo de uma determinada transição de um *hyperchart*. Além disso, a âncora deve ser contextualizada por uma página. As restrições sobre a função A são definidas a seguir: Seja $t \in T$ uma transição do *hyperchart* na forma $t = (X, r, Y)$, onde X é o conjunto de nós de origem, Y é o conjunto de nós de destino e r é o rótulo de t .

Seja $rot: T \times E \rightarrow \{true, false\}$ uma função que verifica se um evento primitivo e_n faz parte do rótulo de uma transição t_m , tal que $rot(t_m, e_n) = true$ se e_n faz parte do rótulo de t_m e $rot(t_m, e_n) = false$ se e_n não faz parte do rótulo de t_m . então: $A(e, t, p) = anc \mid anc \in Anc \Rightarrow rot(t, e) = true \wedge \exists \langle s, p \rangle \in M \wedge (\exists subs \in X \mid s \in \rho^*(subs))$, ou seja, se um evento, uma transição e uma página são mapeadas para uma âncora, então é necessário que esse evento esteja presente no rótulo da transição e que exista um estado mapeado para uma página, de forma que esse estado pertença ao conjunto de origem da transição ou seja subestado de um outro que pertence a esse conjunto.

- L é o nível de visibilidade do hiperdocumento, exatamente como definido pelo modelo HMBS.
- Pch é o conjunto de *Pchannels* (“Presentation Channels”), ou canais de apresentação, que são abstrações através das quais é possível especificar os requisitos relativos à apresentação da informação contida nas páginas.
- Cch é o conjunto de *Cchannels* (“Communication Channels”), ou canais de comunicação, que são abstrações da interface com o subsistema de comunicação subjacente, para o caso de se estar especificando uma aplicação distribuída.
- $V_p: P \times S_S \rightarrow Pch$ é uma função que associa pares formados pelas páginas do hiperdocumento e pelos estados a que estão associadas a *Pchannels* que especificam suas apresentações. A função V_p possui a seguinte restrição:

$$\forall p_1, p_2, \dots, p_n \in P, \forall s \in S_S, \forall pch_1, pch_2, \dots, pch_n \in Pch,$$

$$V_p(p_1, s) = pch_1 \wedge V_p(p_2, s) = pch_2 \wedge \dots \wedge V_p(p_n, s) = pch_n \Leftrightarrow pch_1 = pch_2 = \dots = pch_n,$$
ou seja, não é permitido que vários pares página-estado envolvendo o mesmo estado sejam associados, simultaneamente, a *Pchannels* diferentes.
- $V_a: Anc \times P \rightarrow Pch$ é uma função que associa um par formado por uma âncora e a página a que ela está associada a *Pchannels* que as apresentam e coletam os estímulos sobre elas. Essa função é restrita pelas seguintes relações:

$$\forall a \in Anc, \forall p \in P, \forall pch \in Pch$$

$V_a(a, p) = pch \Leftrightarrow \exists e \in E \wedge \exists t \in T \mid A(e, t, p) = a$, ou seja, se uma âncora e uma página estão associadas a um *Pchannel*, então essa âncora deve estar associada a um evento, a uma transição e a essa página pela relação *A*.

$\forall a_1, a_2, \dots, a_n \in Anc, \forall p_1, p_2, \dots, p_n \in P, \forall pch_1, pch_2, \dots, pch_n \in Pch$

$\forall t \in T, \forall e \in E, (V_a(a_1, p_1) = pch_1 \mid A(e, t, p_1) = a_1) \wedge (V_a(a_2, p_2) = pch_2 \mid A(e, t, p_2) = a_2) \wedge \dots \wedge (V_a(a_n, p_n) = pch_n \mid A(e, t, p_n) = a_n) \Leftrightarrow pch_1 = pch_2 = \dots = pch_n$, ou seja, todas as âncoras relacionadas a páginas diferentes mas ao mesmo par evento-transição devem ser associadas ao mesmo *pchannel*.

- **Ch**: $Pch \rightarrow Cch \cup \emptyset$ é a função que associa um *Cchannel* a cada *Pchannel* utilizado pelo hiperdocumento. Se a aplicação é local, o *Pchannel* pode ser associado ao *cchannel* nulo, ou seja, $Ch(pch) = \emptyset$.