Universidade de São Paulo
Instituto de Física de São Carlos
Departamento de Física e Informática

HMBS: UM MODELO BASEADO EM STATECHARTS PARA A ESPECIFICAÇÃO FORMAL DE HIPERDOCUMENTOS

Marcelo Augusto Santos Turine

USP/IFSC/SBI
8-2-001220

Tese apresentada ao Instituto de Física de São Carlos, da Universidade de São Paulo, para obtenção do título de Doutor em Ciências: Física Aplicada – opção Física Computacional

Orientador: Prof. Dr. Paulo Cesar Masiero
Co-Orientadora: Profª. Drª. Maria Cristina F. de Oliveira

São Carlos
Junho de 1998

IFSC — SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

Turine, Marcelo Augusto Santos
HMBS: Um Modelo Baseado em Statecharts para a Especificação Formal de Hiperdocumentos, São Carlos, 1998
194p

Tese (Doutorado) – Instituto de Física de São Carlos, 1998.

Orientador: Prof. Dr. Paulo Cesar Masiero
Co-Orientadora: Profª. Drª. Maria Cristina F. de Oliveira

1- Hiperdocumento, 2- Especificação Formal, 3- Statecharts, 4- HMBS, 5- Semântica de Navegação

I. Título

UNIVERSIDADE DE SÃO PAULO
Instituto de Física de São Carlos

Av. Dr. Carlos Botelho, 1465
CEP 13560-250 - São Carlos - SP
Brasil

Fone (016) 273-9333
Fax (016) 272-2218

MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE MARCELO AUGUSTO SANTOS TURINE APRESENTADA AO INSTITUTO DE FÍSICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 01 DE JUNHO DE 1998.

COMISSÃO JULGADORA:

Prof. Dr. Paulo Cesar Masiero/ICMC-USP

Prof. Dr. Daniel Schwabe/PUC-RJ

Profa. Dra. Maria das Graças Volpe Nunes/ICMC-USP

Profa. Dra. Marina Teresa Pires Vieira /UFSCar

Prof. Dr. Alvaro Garcia Neto/IFSC-USP

À minha esposa, Joseliza, pelo amor,
pela paciência e pelas infindáveis horas de
estudo roubadas ao seu convívio.

AGRADECIMENTOS

A Deus pelas forças concedidas para enfrentar mais este desafio.

Aos professores Paulo Cesar Masiero e Maria Cristina Ferreira de Oliveira, respectivamente, meu orientador e minha co-orientadora, por terem me confiado este trabalho, pela orientação, dedicação, pelo incentivo e pelos ensinamentos, além da amizade, paciência e atenção sempre concedidas.

Aos meus familiares, em especial, aos meus pais, Maucyr e Esmeralda, pelo apoio constante, pela educação e formação que me proporcionaram.

Ao professor Odelar Leite Linhares por sua constante dedicação ao ensino.

A todos os funcionários do Instituto de Físcia de São Carlos (IFSC-USP) e Instituto de Ciências Matemáticas e Computação (ICMC-USP), que direta ou indiretamente, colaboraram para a realização deste trabalho. Um agradecimento especial à Wladerez, da seção de pós-graduação do IFSC, pelo trabalho irrepreensível.

A todos meus amigos que, direta ou indiretamente, tornaram possível a realização deste trabalho, especialmente ao Mauro, Ricardo, Marquinhos, Fabiano, Márcia Carvalho, Rosemeire, Auri, Ellen, Eliza, Renata Pontin, Graça Nunes, Osvaldo Novaes (Chú), Rosangela Penteado e Beth.

Ao Vanderlei, aluno de iniciação científica, pela ajuda na implementação do sistema HySCharts.

Ao CNPq pelo apoio financeiro fornecido durante todo o desenvolvimento desta tese.

SUMÁRIO

LISTA DE FIGURAS ..... **ix**
LISTA DE TABELAS E QUADROS ..... **xiii**
RESUMO ..... **xv**
ABSTRACT ..... **xvii**

1. INTRODUÇÃO **1**
1.1. Considerações Iniciais 1
1.2. Motivações e Objetivos 4
1.3. Organização da Tese 6

2. MODELOS E MÉTODOS PARA A ESPECIFICAÇÃO E O PROJETO DE HIPERDOCUMENTOS **9**
2.1. Considerações Iniciais 9
2.2. Modelos Orientados ao Sistema 11
2.2.1. Modelo HAM 12
2.2.2. Modelo de Referência Dexter 12
2.2.3. Modelo AHM 15
2.3. Modelos Estruturais 17
2.3.1. Modelo Baseado em Dígrafos 18
2.3.2. Modelo Baseado em Hipergrafos 18
2.3.3. Modelo de Contextos Aninhados 20
2.4. Modelos Comportamentais 23
2.4.1. Modelo Trellis 24
2.4.2. A Proposta de Zheng e Pong 28
2.5. Métodos de Projeto 28

2.5.1. Modelo de Projeto HDM 30
2.5.2. Método RMM 32
2.5.3. Método EORM 34
2.5.4. Método OOHDM 35
2.6. Considerações Finais 39

3. STATECHARTS 41
3.1. Considerações Iniciais 41
3.2. Definição 42
3.3. Descrição Informal 43
3.3.1. Características 43
3.3.2. Parametrização de Estados 48
3.4. Especificação Formal 51
3.4.1. Sintaxe 51
3.4.2. Semântica Operacional 55
3.5. Considerações Finais 57

4. HMBS: UM MODELO BASEADO EM STATECHARTS 59
4.1. Considerações Iniciais 59
4.2. Especificação Formal da Sintaxe 60
4.3. Especificação Informal 63
4.4. Semântica de Navegação 68
4.4.1. Objetos Navegacionais 68
4.4.2. Interpretação da Semântica 72
4.4.3. Exemplos 74
4.5. Visões Hierárquicas 76
4.6. Características do HMBS 81
4.6.1. Estruturação da Informação 81
4.6.2. Separação entre Estrutura e Conteúdo 81
4.6.3. Contextos de Informação 82
4.6.4. Especificação em Níveis de Abstração e de Granularidade 82
4.6.5. Acesso a Informação por Índice 82
4.6.6. Uso de História, de Variáveis e de Ações 84
4.7. Considerações Finais 87

5. ANÁLISE DE ESPECIFICAÇÕES HMBS 89
5.1. Considerações Iniciais 89
5.2. Análise de Propriedades 90
5.2.1. Árvore de Alcançabilidade 90
5.2.2. Alcançabilidade 93
5.2.3. Reiniciabilidade 96
5.2.4. Impasse 97
5.2.5. Vivacidade de Ligações 97
5.2.6. Seqüência Válida de Âncoras 98
5.3. Recursos Adicionais 98
5.3.1. Característica de Visualização 98
5.3.2. Concorrência de Janelas Simultâneas 99
5.3.3. Controle de Acesso 101
5.4. Considerações Finais 103

6. HySCharts: PROTÓTIPO DE UM SISTEMA HIPERDOCUMENTO BASEADO NO HMBS 105
6.1. Considerações Iniciais 105
6.2. Suporte Traduzido X Suporte Interpretado 106
6.3. Arquitetura do Sistema HySCharts 108
6.3.1. Especificação do Sistema 108
6.3.2. Módulo de Autoria 112
6.3.3. Módulo de Navegação 124
6.4. Considerações Finais 129

7. ESTUDO DE CASO: UM EXEMPLO DE ESPECIFICAÇÃO NO CONTEXTO DO PROJETO DE HIPERDOCUMENTOS 131
7.1. Considerações Iniciais 131
7.2. Um Exemplo: Descrição 132
7.3. O Processo de Projeto 133
7.3.1. Projeto Conceitual 135
7.3.2. Projeto Navegacional 139
7.3.3. Projeto de Interface 150
7.3.4. Implementação e Avaliação 151
7.4. Considerações Finais 153

8. TRABALHOS RELACIONADOS: UMA ANÁLISE COMPARATIVA 159
8.1. Considerações Iniciais 159
8.2. Uma Discussão 160
8.3. As Características de Comparação 165
8.4. Vantagens e Desvantagens 167
8.5. XHMBS: Uma Extensão do HMBS 170
8.6. Considerações Finais 171

9. CONCLUSÕES 173
9.1. Considerações Iniciais 173
9.2. Contribuições desta Tese 174
9.3. Sugestões de Novas Pesquisas 175

REFERÊNCIAS BIBLIOGRÁFICAS 179

LISTA DE FIGURAS

Figura 1.1: Exemplo de hiperdocumento 3

Figura 2.1: Arquitetura genérica do HAM 12
Figura 2.2: Arquitetura do modelo Dexter. 13
Figura 2.3: Componentes do modelo Dexter. 15
Figura 2.4: Componentes do modelo AHM. 16
Figura 2.5: Hiperdocumento H especificado por dígrafo. 19
Figura 2.6: Hiperdocumento H especificado por hipergrafo. 20
Figura 2.7: Hierarquia das classes MCA. 21
Figura 2.8: Ligações em nós de composição. 23
Figura 2.9: Rede de Petri PN marcada. 25
Figura 2.10: Statechart que especifica a seleção de um botão. 29
Figura 2.11: Statechart que especifica a abertura e o fechamento de um frame "F_2" usando um botão. 29
Figura 2.12: As fases do método de projeto EORM. 36
Figure 2.13: Exemplo de um ADVchart. 38

Figura 3.1: Exemplo de um Statechart. 43
Figura 3.2: Decomposição do estado "D" do Statechart da Figura 3.1. 45
Figura 3.3: Exemplo genérico da estrutura de uma transição. 46
Figura 3.4: Árvore AND/OR do Statechart da Figura 3.1. 48
Figura 3.5: Parametrização simples do tipo OR. 49
Figura 3.6: Parametrização simples do tipo AND. 49
Figura 3.7: Parametrização do tipo OR com transições associadas. 50
Figura 3.8: Parametrização do tipo OR com reescrita de comportamento. 51
Figura 3.9: Possível configuração de estados para o Statechart da Figura 3.1. 56

Figura 4.1: Mapeamentos "m" e "pl" combinados. 64
Figura 4.2: Estrutura de um hiperdocumento. 66
Figura 4.3: Mapeamento de âncoras 67
Figura 4.4: Exemplo de especificação de uma ligação 70
Figura 4.5: Implementação de ligações múltiplas. 71
Figura 4.6: Configuração (SC_0=$\{B_1, B_2\}$) para o Statechart da Figura 4.2. 74
Figura 4.7: Possíveis leiautes exibidos durante a navegação do hiperdocumento para as configurações dadas e N = 0. 75
Figura 4.8: Possíveis leiautes exibidos durante a navegação do hiperdocumento para as configurações dadas e N = 1. 76
Figura 4.9: Árvore AND/OR para o Statechart da Figura 4.2, com os níveis de visão hierárquica rotulados nos nós 78
Figura 4.10: Ativação da operação "Mostra_VisãoH" durante a navegação do hiperdocumento, SC1 = $\{F_1, F_2\}$ e N = 0 79
Figura 4.11: Resultado da ativação de "ae_4" na janela de visão hierárquica da Figura 4.10 80
Figura 4.12: Estruturas de organização de informação no HMBS. 81
Figura 4.13: Algoritmo para gerar uma configuração de estados válida. 84
Figura 4.14: Possível leiaute exibido após a escolha do título "T_{G21}" durante a navegação do hiperdocumento (SC = $\{E_{11},G_{11a},G_{11b},G_{21}\}$ e N = 1) 85
Figura 4.15: Configuração de contexto $\{P_{E1},P_{E12},P_{G1},P_{G11a},P_{G11b},P_{G2},P_{G22}\}$ e N = 1 86
Figura 4.16: Statechart da Figura 4.2 sem o uso de uma transição com história. 86

Figura 5.1: Estrutura de um hiperdocumento 92
Figura 5.2: Árvore de alcançabilidade para o Statechart da Figura 5.1 94
Figura 5.3: Rede de Petri (sem hierarquia) que descreve concorrência. 100
Figura 5.4: Especificação em Statechart equivalente à Rede de Petri da Figura 5.3 100
Figura 5.5: Statechart com controle de acesso usando variáveis 102

Figura 6.1: Arquitetura do sistema HySCharts 109

Figura 6.2: Modelo de objetos do módulo de autoria do sistema HySCharts. 110

Figura 6.3: Modelo de objetos do módulo de navegação do sistema HySCharts. 111
Figura 6.4: EGS do módulo de autoria do sistema HySCharts. 113
Figura 6.5: Janela de decomposição do estado "Animal". 114
Figura 6.6: Janelas relacionadas ao gerenciamento dos objetos página 116
Figura 6.7: Janelas que gerenciam os objetos âncora 118
Figura 6.8: Janela para criação de âncoras associadas a páginas não textuais 119
Figura 6.9: Janelas para gerenciamento de canais de apresentação. 123
Figura 6.10: Janela do módulo "Verificador dos Objetos HMBS". 124
Figura 6.11: Janela de apresentação do XAnim 126
Figura 6.12: Janela de navegação considerando o Statechart da Figura 6.4, N = 0 e $CC_0=\{P_{Ident\_Texto}, P_{Ident\_Video}\}$ 127
Figura 6.13: Ativação da âncora "Sobre_o_Parque" da configuração de contexto apresentada na Figura 6.12 e $CC=\{P_{Nome}, P_{Logotipo}\}$. 128
Figura 6.14: Resultado da operação "Mostra_VisãoH" quando $CC_0=\{P_{Ident\_Texto}, P_{Ident\_Video}\}$ e N = 0 128
Figura 6.15: Apresentação do hiperdocumento referente à ativação da âncora "Informações" da Figura 6.13 e N = 1 129

Figura 7.1: As fases do processo de projeto. 134
Figura 7.2: Modelo de objetos do PESC. 136
Figura 7.3: Modelo de fatias do PESC 138
Figura 7.4: Legenda para as primitivas do modelo navegacional de tipos 141
Figura 7.5: Modelo navegacional de tipos para o PESC. 143
Figura 7.6: Mapeamento de estruturas de acesso em objetos estruturais HMBS. 145
Figura 7.7: Estratégias alternativas para o mapeamento de estruturas de acesso. 146
Figura 7.8: Modelo navegacional de instância do PESC (alto nível) 147
Figura 7.9: Modelo navegacional de instância do subestado "Animal". 149
Figura 7.10 Modelo navegacional de instância do subestado "Atividade" 149

Figura 7.11: Especificação de um estado compartilhado por contextos navegacionais distintos 150
Figura 7.12: Projeto de Interface do estado "HomePage" 152

Figura 7.13: Implementação da interface do estado "HomePage" da Figura 7.12 154
Figura 7.14: Janela de apresentação resultante da ativação da âncora "Sobre_o_Parque" da Figura 7.13. 155
Figura 7.15: Janela de apresentação resultante da ativação da âncora "Mapa" da Figura 7.14 156
Figura 7.16: Janela resultante da ativação da âncora "Animais" da Figura 7.15 157

Figura 8.1: Rede de Petri (sem hierarquia) que especifica concorrência. 162
Figura 8.2: Rede de Petri (com hierarquia) da Figura 8.1 que especifica concorrência 162
Figura 8.3: Statechart com estados concorrentes, equivalente a Rede de Petri ilustrada na Figura 8.2 163

LISTA DE TABELAS E QUADROS

Tabela 2.1: Esboço das fases do método OOHDM 36
Tabela 4.1: Informações (c,t,Anc_p) das páginas e mapeamento "m" e "pl"..... 65
Tabela 4.2. Mapeamento "ae" 67
Tabela 8.1: Comparação entre os modelos. 167

Quadro I: Implementação da classe "Página" 115
Quadro II: Especificação das âncoras no conteúdo da página "Ident_Texto" 117
Quadro III: Implementação da classe "Âncora" 117
Quadro IV: Implementação da classe "Canal" 120
Quadro V: Implementação da classe "Canal_Texto" 121
Quadro VI: Implementação da classe "Canal_Video" 121
Quadro VII: Implementação da classe "Canal_Áudio" 122

HMBS: UM MODELO BASEADO EM STATECHARTS PARA A ESPECIFICAÇÃO FORMAL DE HIPERDOCUMENTOS

RESUMO

Um novo modelo para a especificação de hiperdocumentos denominado HMBS - ***H**yperdocument **M**odel **B**ased on **S**tatecharts* – é proposto. O HMBS adota como modelo formal subjacente a técnica Statecharts, cuja estrutura e semântica operacional são utilizadas para especificar a estrutura organizacional e a semântica de navegação de hiperdocumentos grandes e complexos. A definição do HMBS, bem como a semântica de navegação adotada, são apresentadas. Na definição apresenta-se como o modelo permite separar as informações referentes à estrutura organizacional e navegacional das representações físicas do hiperdocumento. Também são discutidas características do modelo que possibilitam ao autor analisar a estrutura do hiperdocumento, encorajando a especificação de hiperdocumentos estruturados. Para provar e validar a viabilidade prática do uso do HMBS num contexto real foi desenvolvido um ambiente de autoria e navegação de hiperdocumentos denominado HySCharts - ***Hy**perdocument **S**ystem based on State**Charts***. Esse ambiente fornece facilidades de prototipação rápida e simulação interativa de hiperdocumentos. Para ilustrar como o modelo HMBS e o HySCharts podem ser utilizados no contexto de uma abordagem de projeto sistemática é utilizada como estudo de caso a especificação de um hiperdocumento que apresenta o Parque Ecológico de São Carlos.

HMBS: A STATECHART- BASED MODEL FOR THE FORMAL SPECIFICATION OF HYPERDOCUMENTS

ABSTRACT

A new model for hyperdocument specification called HMBS - ***H****yperdocument* ***M****odel* ***B****ased on* ***S****tatecharts* – is proposed. HMBS uses the Statechart formalism as its underlying model. Statecharts structure and operational semantics are used to specify the organizational structure and the browsing semantics of large and complex hyperdocuments. The definition of HMBS is presented and its browsing semantics is described. It is shown how the model allows the separation of information related to the organizational and navigational structure from the hyperdocument's physical representation. Model features that allow authors to analyze the hyperdocument structure, encouraging the specification of structured hyperdocuments are also discussed. As a proof of concept and also to evaluate the feasibility of using HMBS in real-life applications a system called HySCharts - ***Hy****perdocument* ***S****ystem based on State****Charts*** – was developed. HySCharts is composed by an authoring and a browsing environments, supporting rapid prototyping and interactive simulation of hyperdocuments. A case study is presented that uses the specification of a hyperdocument introducing the Ecological Park of São Carlos to illustrate the use of HMBS and of the HySCharts environment integrated into a systematic design approach.

Capítulo 1

INTRODUÇÃO

1.1. CONSIDERAÇÕES INICIAIS

A demanda por aplicações que utilizam a tecnologia hipermídia encontra-se em contínua expansão devido, principalmente, à disseminação da *World Wide Web*[1] (WWW). Organizações comerciais estão explorando o potencial da WWW e da hipermídia para se apresentar ao público e, ao mesmo tempo, vender seus produtos e serviços com mais rapidez e ampliar o seu universo de consumidores [Bigelow 1988, Bieber; Kacmar 1995, Bieber; Vitali 1997, Ladd et al. 1997, Balasubramanian et al. 1997].

A tecnologia hipermídia consiste em um modelo de organização e recuperação de informações que integra as tecnologias hipertexto e multimídia [Conklin 1987, Begoray 1990, Nielsen 1990a, Hardman et al. 1994]. Herda da tecnologia hipertexto a organização não linear da informação e um novo processo de interação que permite aos seus usuários, denominados leitores, navegar[2] por um grande aglomerado de informações. Segundo Grønbaek e Trigg [1994b], as

---

[1] Um ambiente distribuído disponível na Internet que utiliza a tecnologia hipermídia [Berners-Lee et al. 1994, Marshall 1995].

[2] Outros termos utilizados na literatura para expressar a nova abordagem de leitura proporcionada pela tecnologia hipermídia são surfar e vagar.

ligações formam o coração da tecnologia hipertexto, cujas estruturas navegáveis em forma de rede distinguem-na de outros mecanismos de organização da informação.

Por outro lado, a multimídia permite criar e manipular documentos como objetos estruturados compostos de dados de natureza distinta (texto, imagem, áudio, vídeo ou animação, por exemplo). A integração de informações estáticas e dinâmicas apresentadas em diferentes mídias é vastamente utilizada nos atuais sistemas de gerenciamento de informação, propiciando o desenvolvimento de aplicações com maior flexibilidade e poder de expressão [Smith; Weiss 1988, Nielsen 1990b, Ginige et al. 1995].

Um software ou um sistema que apóia a tecnologia hipermídia é denominado sistema hiperdocumento[3]. Exemplos desses sistemas são Neptune [Delisle; Schwartz 1986], KMS (*Knowledge Management System*) [Akscyn et al. 1988], Hyperties (*Hypertext based on the Interactive Encyclopedia System*) [Marchionini; Shneirderman 1988], NoteCards [Trigg; Irish 1987, Halasz 1988], Intermedia [Yankelovich et al. 1988], HyperCard [Williams 1987], Guide [Brown 1986, Brown 1987], Toolbook [Brown 1991] e Hyper-G [Andrews et al. 1995].

A aplicação, ou o conjunto de documentos on-line gerenciado e manipulado por um sistema hiperdocumento é denominado hiperdocumento[4]. Um documento possui uma dimensão linear, enquanto um hiperdocumento possui várias dimensões, ou seja, contém referências a outros documentos. No hiperdocumento são armazenadas as porções de informação e as meta-informações que determinam as ligações entre elas. A Figura 1.1 ilustra os três componentes básicos que compõem um hiperdocumento:

a) nós: correspondem às unidades básicas de informação e, normalmente, representam a expressão individual de uma idéia ou um conceito;
b) ligações (elos): conectam os nós, formando uma rede de informações que define a estrutura organizacional do hiperdocumento. Segundo Hardman et al. [1994], a função de uma ligação é definir os caminhos de navegação

---

[3] Outros termos encontrados na literatura especializada são sistemas hipertexto e sistemas hipermídia. Nesta tese esses termos são utilizados como sinônimos.

[4] Nesta tese convencionou-se adotar hiperdocumento como um termo que corresponde a denominações diversas encontradas na literatura, como aplicação hipermídia, hipertexto, hipermídia, hiperbase, documentos hipertexto, *web* e hiperespaço de nós e ligações.

no hiperdocumento; e

c) âncoras: representam os pontos de origem das ligações, ou seja, os pontos a partir dos quais os leitores, ativando ligações, poderão navegar no hiperdocumento.

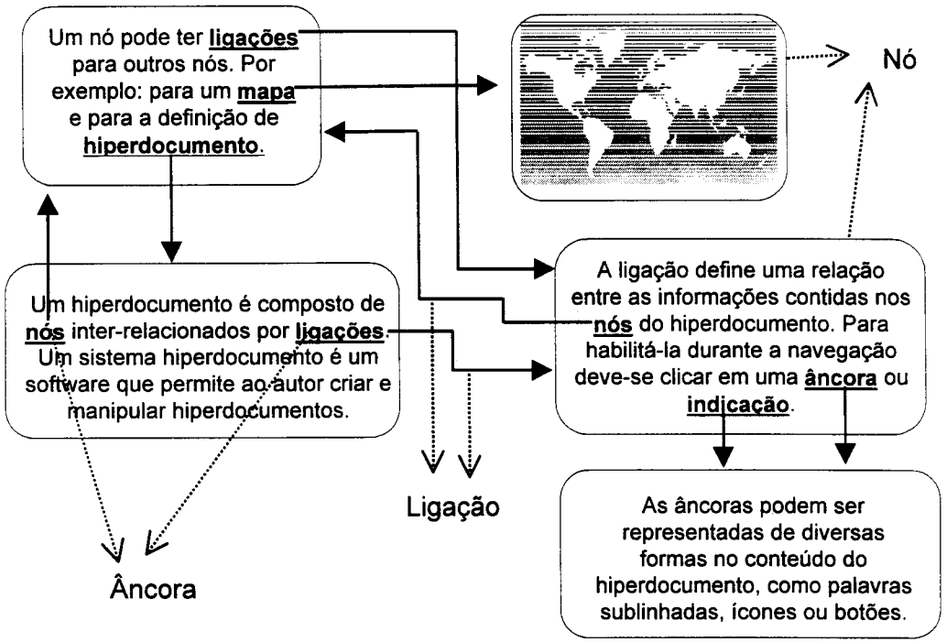


Figura 1.1: Exemplo de hiperdocumento.

Para que o hiperdocumento seja explorado pelos leitores, é necessário especificar, além da sua estrutura organizacional, a semântica de navegação associada (*browsing semantics*) [Garzotto et al. 1993a]. Segundo Stotts e Furuta [1989], a semântica de navegação define as propriedades dinâmicas e de visualização do hiperdocumento em tempo de navegação, permitindo especificar a estrutura navegacional e as transformações associadas ao hiperdocumento quando ligações são ativadas pelo leitor. As tarefas de especificação da estrutura organizacional e da semântica de navegação fazem parte da atividade de autoria que, segundo Garzotto et al. [1991, 1993a], consiste de duas fases. Na primeira, autoria em ponto grande (granularidade grossa), especificam-se a estrutura organizacional e como o leitor poderá navegar por essa estrutura (semântica de navegação). A segunda, autoria em ponto pequeno (granularidade fina), consiste no preenchimento dos nós com a informação desejada e é fortemente dependente das ferramentas disponíveis para implementação, bem como do meio utilizado para armazenar as informações.

Uma vez criado o hiperdocumento na atividade de autoria, ele pode ser interpretado e exibido ao leitor para a navegação. Normalmente o leitor navega no hiperdocumento de acordo com seu grau de interesse e/ou de conhecimento do assunto, e a sua disponibilidade de tempo. Porém, segundo Fraïssé [1997], navegar eficientemente em um espaço de informação não é simplesmente investigar ou explorar livremente um conjunto desarrumado de ligações ou consultar uma base de dados usando caixas de diálogo ou botões. Supõe-se que um esforço prévio de autoria tenha gerado o hiperdocumento de forma a tornar o processo de busca de informações adequado às restrições cognitivas dos diferentes tipos de leitores.

1.2. MOTIVAÇÕES E OBJETIVOS

Pesquisas realizadas ao longo das décadas de 70 e 80 demonstraram o potencial da tecnologia hipermídia como um novo paradigma de organização e recuperação de informações para uma variedade de domínios de aplicação. Porém, são inúmeros os problemas que surgem no projeto de hiperdocumentos. Pode-se citar, entre outros: capturar e organizar a estrutura do domínio de informação, muitas vezes complexa, de maneira a torná-la clara e acessível aos leitores; estabelecer abordagens sistemáticas para definir a estrutura organizacional e a semântica de navegação, bem como para modelar o conteúdo e a interface; definir formas adequadas para organizar e recuperar as informações em sistemas hiperdocumento; implementar mecanismos eficientes para navegar em hiperdocumentos; obter modelos do domínio subjacente que possam ser mapeados de forma simples e direta para um modelo de projeto; especificar os requisitos de informações multimídia; e criar hiperdocumentos que possam ser implementados em plataformas diferentes. Esses são problemas típicos da atividade de autoria, cuja qualidade, de acordo com Brown [1990] e Garzotto et al. [1993a], é o fator determinante no projeto de hiperdocumentos. Tais problemas, aliados àqueles inerentes à utilização da tecnologia hipermídia (desorientação do leitor e sobrecarga cognitiva), têm motivado pesquisas na área.

Os sistemas hiperdocumento e os próprios hiperdocumentos são produtos de software que podem se beneficiar de modelos de especificação e métodos que sistematizem o seu processo de desenvolvimento. No entanto, os métodos e as técnicas tradicionais de desenvolvimento de software não são adequados para o

projeto de hiperdocumentos, que apresentam requisitos próprios, como a necessidade de gerenciar um grande volume de informações, e de combinar a navegação controlada pelo leitor com a própria natureza das informações multimídia [Hardman et al. 1993, Garzotto et al. 1994]. Tais fatos motivam a pesquisa por novos modelos e métodos que forneçam diretrizes para gerenciar de maneira sistemática o projeto de hiperdocumentos, ajudando a disciplinar a atividade de autoria, além de encorajar o desenvolvimento de hiperdocumentos estruturados. O uso de modelos não somente melhora o processo de projeto e desenvolvimento de hiperdocumentos [Cavallaro et al. 1993, Garzotto et al. 1994], mas também facilita a sua análise e avaliação [Garzotto et al. 1995].

Particularmente, a adoção de modelos formais para a descrição de hiperdocumentos pode permitir que estes sejam especificados de forma independente de sua implementação. Segundo Lange [1990], para que os hiperdocumentos sejam "executados" em diferentes sistemas eles devem ser especificados utilizando modelos formais e abstratos. Adicionalmente, modelos formais adequados podem oferecer abordagens sistemáticas e confiáveis para analisar e verificar propriedades estruturais e dinâmicas de hiperdocumentos. Courtiat e Oliveira [1996] destacam que a utilização de técnicas formais para a especificação de hiperdocumentos é uma tendência atual.

Nesta tese postula-se que as características da técnica formal Statecharts podem fornecer um suporte adequado para minimizar diversos problemas associados à especificação de hiperdocumentos. Desse modo, propõe-se um novo modelo denominado Modelo de Hiperdocumento Baseado em Statecharts, ou HMBS (***H****yperdocument* ***M****odel* ***B****ased on* ***S****tatecharts*), que adota o Statecharts como modelo formal subjacente para a especificação de hiperdocumentos. A estrutura e a semântica operacional de Statecharts são utilizadas para especificar a estrutura organizacional e a semântica de navegação de hiperdocumentos grandes e complexos. A semântica de navegação adotada é totalmente compatível com a estrutura definida para o hiperdocumento e é refletida na organização de um Statechart, fornecendo mecanismos de navegação que realçam a estrutura hierárquica típica de muitas aplicações.

O HMBS fornece um modelo estrutural e comportamental preciso e semanticamente rigoroso para a especificação de hiperdocumentos, encorajando o desenvolvimento de hiperdocumentos estruturados. Além da definição, estrutura e

operações do modelo HMBS, nesta tese também é apresentado um protótipo de um sistema hiperdocumento, denominado HySCharts (***Hy****perdocument* ***S****ystem based on State****Charts***), que permite avaliar a possibilidade de uso do HMBS num contexto real, permitindo ao autor especificar um hiperdocumento e ver rapidamente e graficamente o resultado obtido. O HySCharts foi desenvolvido como um sistema de suporte automatizado, oferecendo ambientes de autoria e de navegação que permitem criar e validar especificações de hiperdocumentos segundo o modelo HMBS.

1.3. ORGANIZAÇÃO DA TESE

Esta tese está organizada da seguinte forma: neste capítulo foram apresentados o problema a ser investigado e os objetivos do modelo proposto para a especificação de hiperdocumentos. Para uniformizar a terminologia foram introduzidos os termos básicos de interesse ao tema. Apesar de não existir uma definição formal e exata desses termos, existe um consenso na literatura quanto às suas características principais, o qual foi respeitado neste texto.

O Capítulo 2 contém uma revisão da literatura especializada que versa sobre modelos para a especificação de hiperdocumentos. Também são revisados trabalhos consagrados sobre modelagem de sistemas hiperdocumento, bem como métodos sistemáticos para o projeto e o desenvolvimento de hiperdocumentos.

No Capítulo 3 são apresentados, sucintamente, os conceitos básicos do formalismo Statecharts relevantes para a compreensão do modelo proposto. É apresentada, ainda, a especificação formal de Statecharts, compreendendo sua sintaxe e sua semântica operacional simplificada.

O Capítulo 4 contém a especificação do modelo HMBS, definido com ênfase na formalização do hiperdocumento como um conjunto de objetos estruturais, navegacionais e de apresentação. As características informais do modelo e a sua semântica de navegação são, também, abordadas.

No Capítulo 5 mostra-se como o uso da técnica Statecharts para a especificação de hiperdocumentos possibilita a identificação de inconsistências estruturais, bem como de problemas relacionados à visualização, navegação e análise de propriedades. Apresentam-se, também, propostas sobre como utilizar as propriedades dinâmicas de sistemas reativos no contexto de hiperdocumentos.

O sistema HySCharts para autoria e navegação de hiperdocumentos em

HMBS é apresentado no Capítulo 6. O módulo de autoria permite criar as especificações em HMBS do hiperdocumento. Para tornar disponível aos leitores o conteúdo do hiperdocumento, tais especificações são interpretadas e "executadas" no módulo de navegação, de acordo com o comportamento e com a estrutura navegacional subjacente ao modelo.

Para ilustrar a utilização do modelo HMBS e do sistema HySCharts é apresentado no Capítulo 7 um processo genérico de projeto de hiperdocumentos, e é utilizado como estudo de caso é uma aplicação que apresenta o Parque Ecológico de São Carlos (PESC). Uma descrição da aplicação e das fases que compõem o processo de desenvolvimento de hiperdocumentos, além de exemplos de navegação no HySCharts e no *browser* Netscape, são ilustrados.

Uma análise comparativa dos modelos e dos métodos de projeto abordados no Capítulo 2 com o modelo HMBS é apresentada no Capítulo 8. Finalmente, as conclusões desta tese e as perspectivas para a sua continuidade são apresentadas no Capítulo 9.

Capítulo 2

MODELOS E MÉTODOS PARA A ESPECIFICAÇÃO E O PROJETO DE HIPERDOCUMENTOS

2.1. CONSIDERAÇÕES INICIAIS

Segundo Isakowitz et al. [1997a], o problema de muitos hiperdocumentos é que eles foram projetados e desenvolvidos de maneira *ad hoc*, isto é, os autores não utilizaram técnicas de modelagem, nem seguiram um processo de desenvolvimento detalhado antes de implementar o hiperdocumento em uma plataforma específica.

A tecnologia hipermídia tem evoluído de modo a apoiar hiperdocumentos cada vez mais complexos em termos de conteúdo (diferentes mídias) e de estrutura organizacional e navegacional. A possibilidade de representar referências arbitrárias entre partes do hiperdocumento é uma característica importante na sua manipulação. O problema consiste em combinar essa flexibilidade com mecanismos de organização de forma a evitar que, durante a navegação, o leitor se perca no hiperespaço. Paralelamente ao rápido avanço tecnológico, modelos destinados a fornecer mecanismos de organização para apoiar a especificação de hiperdocumentos e dos sistemas que o gerenciam foram propostos.

Analisando os vários modelos existentes na literatura pode-se identificar cinco abordagens distintas de modelagem:

a) A primeira classe compreende os "modelos fechados", os quais utilizam estruturas topológicas para definir a organização do hiperdocumento. Os sistemas hiperdocumento baseados em tais modelos foram desenvolvidos como aplicações únicas e autocontidas, impossibilitando a reutilização de seus componentes e de sua organização em outras aplicações. Exemplos de abordagens nesta classe são as estruturas lineares usadas em Hypercard [Williams 1987] e Guide [Brown 1987], as estruturas hierárquicas de KMS [Akscyn et al. 1988], o modelo de sistema de informação do sistema gIBIS (*graphical Issue Based Information Systems*) [Conklin; Begeman 1988] e de seu sucessor rIBIS (*real-time Issue Based Information Systems*) [Rein; Ellis 1991]. Os modelos desta classe não são abordados em maiores detalhes nesta tese;

b) A segunda classe inclui os "modelos orientados ao sistema", os quais definem uma arquitetura genérica para sistemas hiperdocumento independentemente da plataforma de implementação e identificam abstrações relevantes nos sistemas existentes e futuros. Entre esses incluem-se o modelo HAM (*Hypertext Abstract Machine*) [Campbell; Goodman 1988], o modelo teórico de Garg [1988], o modelo de referência de hipertexto Dexter [Halasz; Schwartz 1994], o modelo AHM (*Amsterdam Hypermedia Model*) [Hardman et al. 1994] e o modelo do HAS (*Hypermedia Authoring System*) [Vuong; Pereira 1996];

c) A terceira classe compreende os "modelos estruturais", baseados em técnicas cujo objetivo principal é especificar explicitamente a estrutura organizacional do hiperdocumento como uma rede de nós e ligações. Tais técnicas são estáticas, isto é, não possuem uma semântica formalmente definida para especificar o comportamento navegacional inerente aos hiperdocumentos. Exemplos dessa classe são o modelo Gram [Amann; Scholl 1992] baseado em dígrafos, o modelo proposto por Tompa [1989] baseado em hipergrafos, o Modelo de Contextos Aninhados (MCA) [Casanova et al. 1991, Soares et al. 1993, Soares et al. 1994] e o modelo de dados HYDESIGN, que segue uma abordagem orientada a objetos [Marmann; Schlageter 1992];

d) A quarta classe inclui os "modelos comportamentais", que permitem modelar o comportamento dinâmico associado ao hiperdocumento, especificando formalmente uma semântica de navegação e os aspectos pertinentes à interface de sistemas hiperdocumento. Exemplos de modelos nesta classe são o modelo Trellis [Stotts; Furuta 1989] e a proposta de Zheng e Pong [1992]. Como explicitado adiante, o modelo proposto nesta tese pode ser incluído nesta classe devido a sua capacidade de modelar a semântica de navegação, além da estrutura organizacional de hiperdocumentos; e

e) A quinta classe inclui os "métodos de projeto", os quais fornecem um conjunto consistente e sistemático de fases que direcionam os autores no processo de projeto e desenvolvimento de hiperdocumentos. Nesta classe podem ser incluídos o modelo de projeto HDM (*Hypermedia Design Model*) [Garzotto et al. 1991, Schwabe et al. 1992, Garzotto et al. 1993a], o OOHDM (*Object Oriented Hypermedia Design Model*) [Schwabe; Rossi 1995, Schwabe et al. 1996], o RMM (*Relationship Management Methodology*) [Isakowitz et al. 1995] e o EORM (*Enhanced Object-Relationship Model*) [Lange 1994].

O presente capítulo está organizado da seguinte forma. Na Seção 2.2 são apresentados os principais modelos orientados ao sistema. Os modelos estruturais mais discutidos na literatura são mostrados na Seção 2.3. Maior ênfase é dada na Seção 2.4 aos modelos comportamentais, que serviram de base para o modelo proposto nesta tese. Os principais métodos de projeto para apoiar o desenvolvimento de hiperdocumentos são apresentados na Seção 2.5. Um aspecto de interesse no estudo desses métodos é identificar os modelos adotados em cada uma das fases do processo de desenvolvimento.

2.2. MODELOS ORIENTADOS AO SISTEMA

Os sistemas hiperdocumento evoluíram de modo independente, e essa independência refletiu-se nos hiperdocumentos por eles manipulados, de forma que as aplicações produzidas não podem ser utilizadas em outra plataforma. Assim, surgiu a necessidade de definir um conjunto padrão de conceitos e terminologias comuns para o projeto de sistemas hiperdocumento. Nesta seção são apresentados os principais modelos que se tornaram referência e fornecem

abstrações que podem ser utilizadas como um padrão de comparação, interoperabilidade e intercâmbio entre diferentes sistemas hiperdocumento. Os modelos apresentados nesta seção são o modelo HAM (Seção 2.2.1), o modelo de Referência Dexter (Seção 2.2.2) e o modelo AHM (Seção 2.2.3).

2.2.1.Modelo HAM

Campbell e Goodman [1988] foram os precursores da busca por modelos para a especificação de sistemas hiperdocumento, propondo um modelo de armazenamento de dados concebido para representar a estrutura do hiperdocumento em diferentes sistemas. Os criadores do modelo defendem que, para o sucesso desses sistemas, é essencial possuir um modelo que especifique os componentes do hiperdocumento aliado a um método de armazenamento de informações eficiente e independente da aplicação.

A Figura 2.1 esquematiza a arquitetura em níveis proposta pelo modelo. No nível de banco de dados está um servidor baseado em transações que é responsável pelo armazenamento e compartilhamento das informações em um ambiente distribuído. O nível da máquina abstrata é considerado o núcleo central da arquitetura e contém a estrutura organizacional do hiperdocumento, utilizando cinco componentes organizados hierarquicamente: grafo, contexto, nó, ligação e atributo. No nível de apresentação, ou interface com o usuário, especifica-se como as informações são apresentadas aos leitores. Assim, o HAM permite que várias aplicações estabeleçam conexões e manipulem as estruturas e as informações de um hiperdocumento específico.

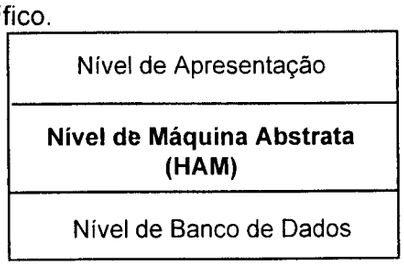


Figura 2.1: Arquitetura genérica do HAM [Campbell; Goodman 1988, p.857].

2.2.2.Modelo de Referência Dexter

O modelo de Referência Dexter (doravante denominado apenas modelo Dexter) propõe uma arquitetura para sistemas hiperdocumento baseada em três camadas (Figura 2.2): camada de tempo de execução, de armazenamento e

interna aos componentes, com interfaces bem definidas entre elas. A arquitetura proposta pelo modelo Dexter é um aprimoramento da arquitetura do modelo HAM, pois acrescenta duas novas interfaces (especificação da apresentação e ancoragem) entre as camadas [Grønbaek 1994, Hardman et al. 1994, Leggett; Schnase 1994; Garzotto et al. 1994].

A camada de armazenamento é considerada o foco do modelo, pois contém a representação da estrutura do hiperdocumento. A abstração básica desta camada é o componente, que pode ser atômico (o mesmo conceito de nó), composto ou de ligação. Os componentes compostos fornecem um mecanismo de estruturação, ou seja, definem uma composição de componentes atômicos inter-relacionados segundo um determinado contexto. Assim, um hiperdocumento, nesta camada, é representado como uma base de dados composta por uma hierarquia de componentes, na qual cada componente comporta-se como repositório genérico de dados.

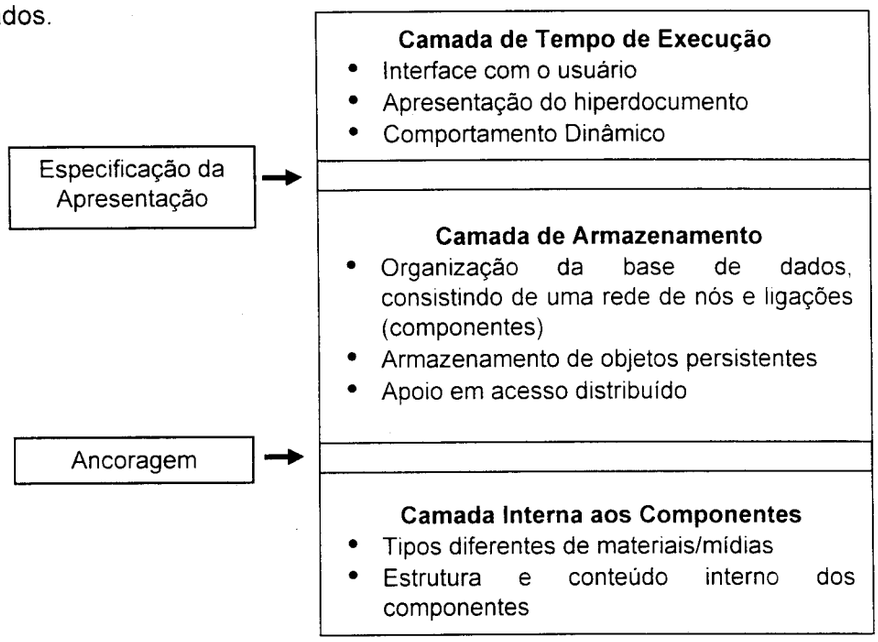


Figura 2.2: Arquitetura do modelo Dexter.

A camada interna aos componentes compreende, especificamente, o conteúdo e a estrutura interna dos componentes do hiperdocumento. Essa camada não é explorada pelo modelo Dexter e outros padrões para representação visando intercâmbio de documentos propostos pela ISO (*International Organization for Standardization*), tais como SGML (*Standard Generalized Markup Language*), HTML (*Hypertext Markup Language*), HyTime (*Hypermedia/Time-Based Structuring*

Language), MHEG (*Multimedia and Hypermedia Information Coding Experts Group*) e ODA (*Office Document Architecture*), podem ser utilizados em conjunto com o modelo Dexter para modelar a estrutura interna dos documentos.

Segundo Halasz e Schwartz [1994], a interface entre a camada de armazenamento e a camada interna ao componente é crítica, pois um sistema hiperdocumento necessita de um mecanismo para referenciar (endereçar) posições ou itens dentro do conteúdo de um componente em particular. O mecanismo utilizado é baseado na noção de âncoras (denominado ancoragem), que representam identificadores relacionados às ligações e a um valor que seleciona a parte ancorada, permitindo o endereçamento de localizações ou itens dentro do conteúdo de um particular componente. Além disso, essa camada permite que as informações dependentes dos dados sejam armazenadas separadamente da hiperestrutura.

A Figura 2.3 ilustra a organização geral da camada de armazenamento de um hiperdocumento. Os cinco componentes apresentados incluem três componentes atômicos (identificadores #3346, #332, #409), um componente composto (identificador #4112) e um componente de ligação (Identificador #9981) que representa o relacionamento de uma âncora do componente atômico identificado por #3346 para o componente composto #4112.

A camada de tempo de execução é responsável pela manipulação das ligações, âncoras e componentes em tempo de execução, proporcionando a funcionalidade para o uso, visualização e manipulação do hiperdocumento. O conceito fundamental nessa camada é o de instanciação de um componente, ou seja, apresentação do mesmo para o leitor.

A interface entre a camada de tempo de execução e a camada de armazenamento é denominada especificação da apresentação, e permite definir como um componente deve ser apresentado, quais informações serão visíveis e o tamanho de uma janela de apresentação, além de obter informações específicas do componente na camada de armazenamento. Portanto, segundo o modelo, a forma de apresentação de um componente pode não ser uma função exclusiva do sistema, mas também ser uma propriedade do próprio componente.

Dentre os exemplos de sistemas hiperdocumento que utilizam os conceitos propostos pelo modelo Dexter encontram-se o sistema DHM (*DeVise Hypermedia*) [Grønbaek et al. 1994, Grønbaek; Trigg 1994a] cuja ênfase é na autoria

cooperativa; o sistema DHM/WWW (*DeVise Hypermedia for World Wide Web*) [Grønbaek et al. 1997] que integra os serviços fornecidos pelo DHM à WWW; o sistema HyperProp [Soares et al. 1993, Colcher et al. 1995, Soares et al. 1995a], que usa o modelo de contextos aninhados e propõe uma arquitetura aberta para permitir portabilidade entre plataformas; o sistema SEPIA (*Structured Elicitation and Processing of Ideas for Authoring*) [Schütt; Streitz 1990], cujo objetivo é apoiar a atividade de autoria; o sistema baseado no modelo de dados GHMI (*General Hypertext data Model supporting Integration*) [Wan; Bieber 1996], o qual permite a integração de hiperdocumentos e sistemas de informação; e o KMEDIA-UNIX (*KAIST Hypermedia for Unix*) [Park et al. 1996], entre outros.

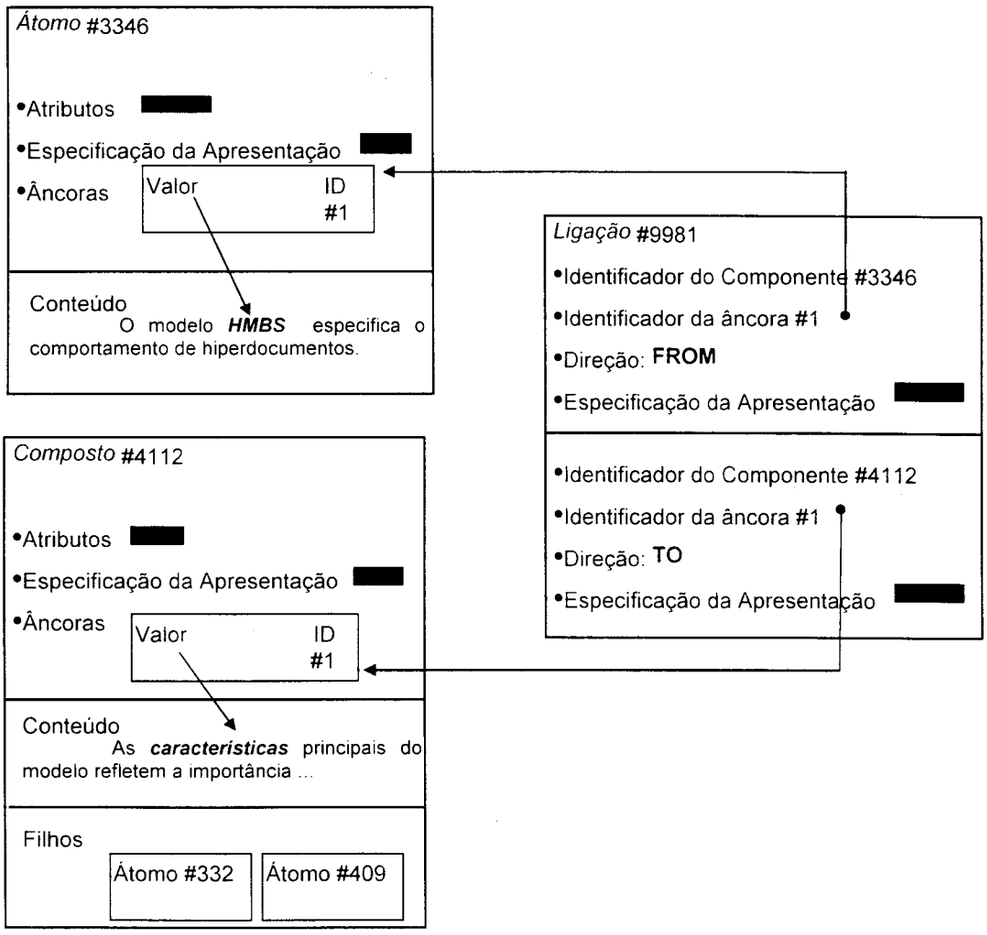


Figura 2.3: Componentes do modelo Dexter [Halasz; Schwartz 1994, p.35].

2.2.3.Modelo AHM

Hardman et al. [1994] desenvolveram o modelo AHM como uma extensão ao modelo Dexter, adicionando características para a modelagem do comportamento

temporal, sincronização de dados multimídia e atributos de especificação de apresentação de alto nível para auxiliar a composição de nós com conteúdo multimídia e contexto de ligações. Grande parte dessas extensões é baseada no modelo multimídia denominado CMIF (*CWI Multimedia Interchange Format*) [Bulterman 1993]. Para validar os conceitos propostos pelo modelo AHM foi desenvolvido um ambiente de autoria e apresentação denominado CMIFed (*CWI Multimedia Interchange Format editor*) [Van Rossum et al. 1993]. A seguir, as principais extensões do modelo AHM em relação ao modelo Dexter são apresentadas.

Os componentes AHM possuem a mesma nomenclatura dos componentes do modelo Dexter (atômicos, compostos e ligação). O componente atômico contém, na seção de especificação da apresentação, requisitos para a modelagem temporal e atributos de apresentação de alto nível (via canais) (Figura 2.4). Os componentes compostos permitem a inclusão de referências a componentes atômicos ou compostos e não têm mais conteúdo associado diretamente, permitindo especificar a estrutura de apresentação ao invés de simplesmente colecionar componentes relacionados para o propósito de navegação.

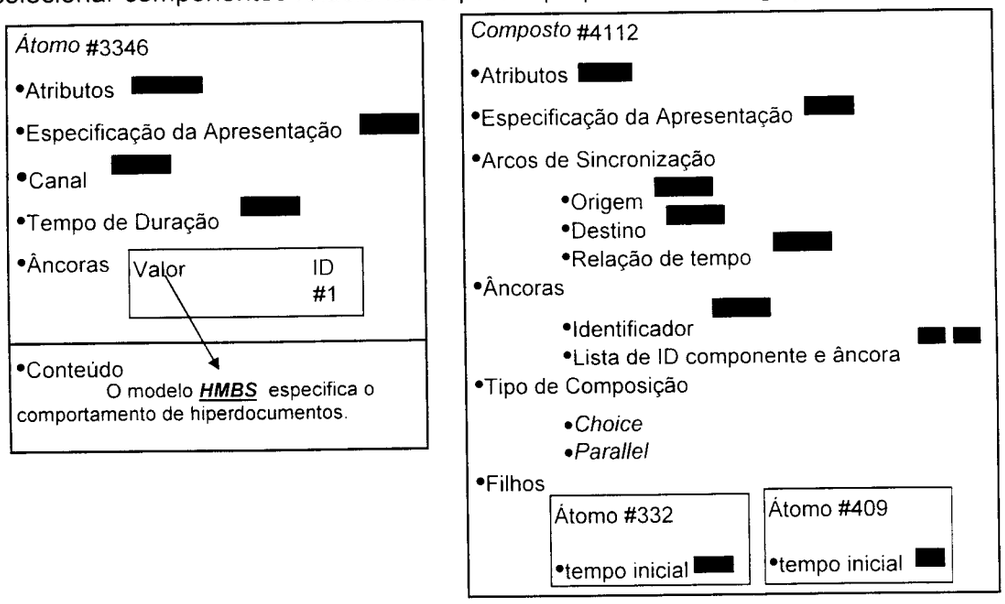


Figura 2.4: Componentes do modelo AHM [Hardman et al. 1994].

O componente composto permite referenciar os componentes atômicos associados a ele usando o tipo de composição *choice* ou *parallel*, isto é, as cadeias de dados dos componentes atômicos podem ser, respectivamente, apresentadas de modo exclusivo ou concorrente. Para tratar da sincronização entre cadeias, o

modelo usa arcos de sincronização, que são estruturas que definem a ordem da informação na especificação da apresentação dos componentes compostos. Esses arcos são usados para indicar se duas cadeias devem sincronizar suas apresentações em algum ponto, permitindo a especificação de requisitos de sincronização, como o atraso[5] (*delay*).

Para especificar o comportamento da apresentação multimídia quando uma ligação é seguida, foi criada a noção de contextos de ligação ou contextos de informação, que são representados por componentes compostos que encapsulam os componentes envolvidos na origem (contexto corrente) e no destino de uma ligação. Dessa forma, o uso de contextos torna explícita a porção da apresentação que é afetada quando uma ligação é seguida a partir de uma âncora, permitindo ao autor especificar as características desejadas da apresentação ao seguir uma ligação, ou seja, se alguns dados irão persistir ou se suas apresentações serão encerradas.

Tanto o Dexter quanto o AHM permitem que atributos de apresentação sejam especificados para componentes atômicos ou compostos. Esses atributos são suficientes para o controle local da apresentação, mas são inadequados para especificar características globais do hiperdocumento. Para tal finalidade, o AHM usa o conceito de canais, dispositivos lógicos que definem o estilo de apresentação do conteúdo associado ao componente. Os canais são definidos globalmente e referenciados para cada componente atômico. Com esse recurso, o modelo permite especificar atributos de apresentação de alto nível para cada tipo de mídia apresentada pelo canal, como o volume do áudio, a velocidade da apresentação e preferências de cor e fonte.

2.3. MODELOS ESTRUTURAIS

Os modelos baseados em dígrafos e hipergrafos são os mais tradicionais para especificar a estrutura organizacional de hiperdocumentos. Um modelo mais robusto e completo dessa classe é o modelo conceitual de dados utilizado no sistema HyperProp [Soares et al. 1993], proposto como um modelo genérico para documentos hipermídia. O modelo de dados Canto [Nicholas; Rosenberg 1993] permite estruturar hierarquicamente hiperdocumentos utilizando nós e ligações organizados em esquemas de dados. Nesta seção é apresentada uma visão geral

[5] Indica o intervalo de tempo entre a entrega das cadeias de dados no local de destino.

de algumas propostas de modelagem de hiperdocumentos baseadas em dígrafos (Seção 2.3.1) e hipergrafos (Seção 2.3.2), além do modelo MCA (Seção 2.3.3).

2.3.1. Modelo Baseado em Dígrafos

Segundo Delisle e Schwartz [1987] e Amann e Scholl [1992], um hiperdocumento H, modelado como um dígrafo, é definido como uma 3-tupla H=<N,L,E>, em que:

a) N é um conjunto finito de nós, cada um dos quais está associado a uma unidade de informação;

b) L é um conjunto de rótulos (*labels*), que correspondem às âncoras, e

c) E: $N \times L \rightarrow N$ é um conjunto de arcos dirigidos rotulados (*edges*), que especificam as ligações entre os nós.

Para definir a posição corrente ou estado do usuário durante a navegação utiliza-se um marcador para o nó (representado graficamente por $\oplus$). Uma semântica de navegação simples é definida pelo modelo usando o comportamento dos marcadores. Quando um marcador está em um nó, as seguintes ações são realizadas [Raymond; Tompa 1988]:

a) o conteúdo associado ao nó é exibido;

b) os rótulos associados aos arcos cuja origem é o nó marcado são disponíveis para a navegação, por exemplo, em um menu; e

c) quando um rótulo é selecionado pelo usuário, o marcador é movido para o nó destino.

A Figura 2.5 ilustra a estrutura de um hiperdocumento H com cinco nós de informação. O estado do usuário é definido pelo nó de nome "2" e, de acordo com a semântica de navegação adotada, a informação associada ao nó corrente é apresentada e as âncoras que podem ser ativadas são *help*, *cont* e *note*.

2.3.2. Modelo Baseado em Hipergrafos

Tompa [1989] propôs o uso do formalismo hipergrafos para representar a estrutura geral de hiperdocumentos, permitindo especificar explicitamente os componentes (nós, ligações e rótulos) comuns da estrutura, bem como referenciar diretamente "grupo de nós" (hipernós) por ligações semânticas comuns (hiperarcos). Um hiperdocumento H, modelado como um hipergrafo rotulado dirigido, é definido por uma 6-tupla H=<N, P, R,V ,L ,E>, na qual:

a) N é um conjunto finito de nós;

b) P é um conjunto de páginas, que são as unidades de informação;

c) R é um conjunto de interpretadores utilizados para ler e apresentar as páginas de informação utilizando algum dispositivo físico;

d) V: N→P é uma função que mapeia cada nó da estrutura em uma página de informação;

e) L é um conjunto de rótulos que definem as âncoras do hiperdocumento. Um rótulo é um valor atômico que pode ser lido, escrito ou comparado com um outro rótulo. A habilidade de comparar rótulos fornece o mecanismo de navegação na base de dados definida pelos nós e arcos; e

f) E: $2^N \times 2^L \rightarrow 2^N$ é um conjunto de hiperarcos rotulados dirigidos.

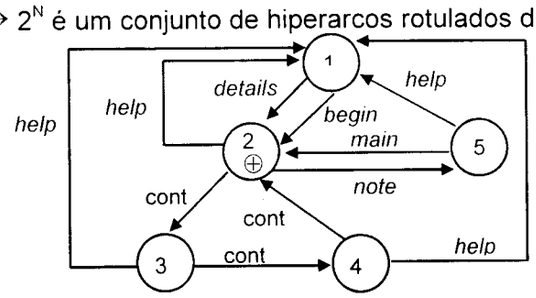


Figura 2.5: Hiperdocumento H especificado por dígrafo [Tompa 1989, p.86].

No hipergrafo subjacente ao hiperdocumento, os usuários, em um determinado instante, estão posicionados em um número arbitrário de nós (marcados pelo símbolo ⊕), e dessa forma têm acesso concorrente a várias páginas de informação. O comportamento dinâmico é definido pela noção de marcadores. Um estado do usuário compreende o estado do hipergrafo instanciado com um conjunto $M \subset N$ que designa o conjunto de nós marcados.

Uma semântica de navegação orientada a conjuntos pode ser definida da seguinte maneira: o sistema hiperdocumento deve interpretar todas as páginas associadas aos nós marcados, examinar os arcos cuja origem inclui esses nós e marcar todos os nós destino quando algum rótulo pertencente a um arco for selecionado. A Figura 2.6 retrata a estrutura em hipergrafo do mesmo hiperdocumento ilustrado na Figura 2.5. De acordo com a semântica de navegação, o conteúdo das duas páginas (*Páginas 1 e 5*) associadas aos nós "1" e "5" do hiperdocumento é apresentado simultaneamente. Se o rótulo "details" for selecionado, os nós "2", "3", "4" e "5" serão marcados e, conseqüentemente, o conteúdo de suas páginas será apresentado, iniciando um processo de navegação

concorrente.

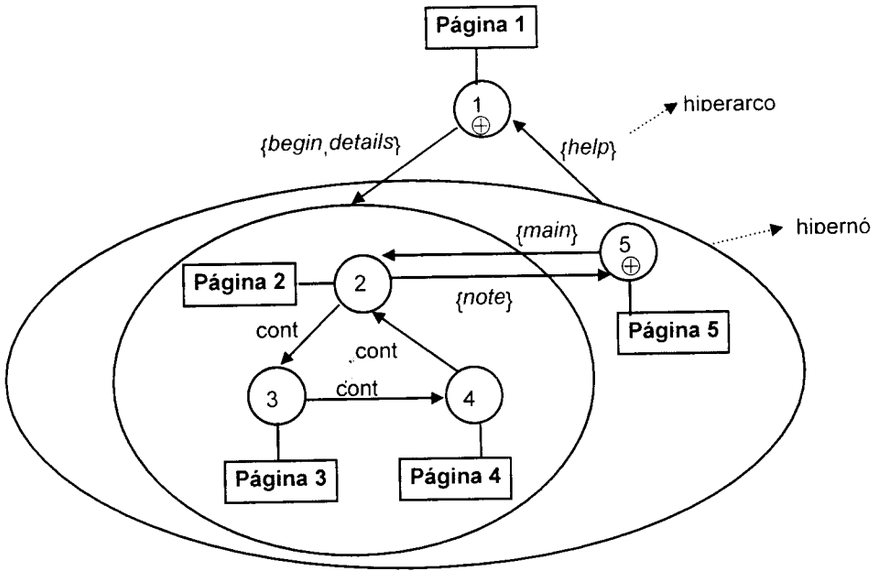


Figura 2.6: Hiperdocumento H especificado por hipergrafo [Tompa 1989, p.89].

2.3.3.Modelo de Contextos Aninhados

O modelo conceitual de dados utilizado no sistema HyperProp [Soares et al. 1993, Colcher et al. 1995, Soares et al. 1995a] é denominado Modelo de Contextos Aninhados (MCA). Trata-se de um modelo hipermídia genérico que, entre outros recursos, permite composições aninhadas de nós e ligações, separa a estrutura dos documentos contida nas ligações e nas âncoras das informações contidas nos nós, permite a especificação estrutural e de apresentação de dados em conformidade com o padrão MHEG, permite a especificação de relações temporais e espaciais entre nós, fornece controle de versões, mantém histórico dos documentos e permite apoiar o trabalho cooperativo [Soares et al. 1994, Muchaluat; Soares 1995, Soares et al. 1995b, Costa et al. 1996, Soares; Rodrigues 1997].

A especificação de hiperdocumentos no MCA é baseada nos conceitos convencionais de nós e ligações subjacentes à teoria dos grafos, com a adição das classes de nós terminais e de composição, sendo que estes últimos constituem a característica central do modelo. Assim, o modelo estende o formalismo de grafos permitindo especificar nós aninhados, formando uma hierarquia de nós.

A Figura 2.7 ilustra a hierarquia de classes proposta pelo modelo. Uma entidade é um objeto genérico do modelo e possui como atributos um identificador

único, uma lista de controle de acesso que associa um usuário ou um grupo de usuários aos seus respectivos direitos de acesso ao atributo, e um descritor que contém informações que determinam como a entidade deve ser apresentada ao leitor.

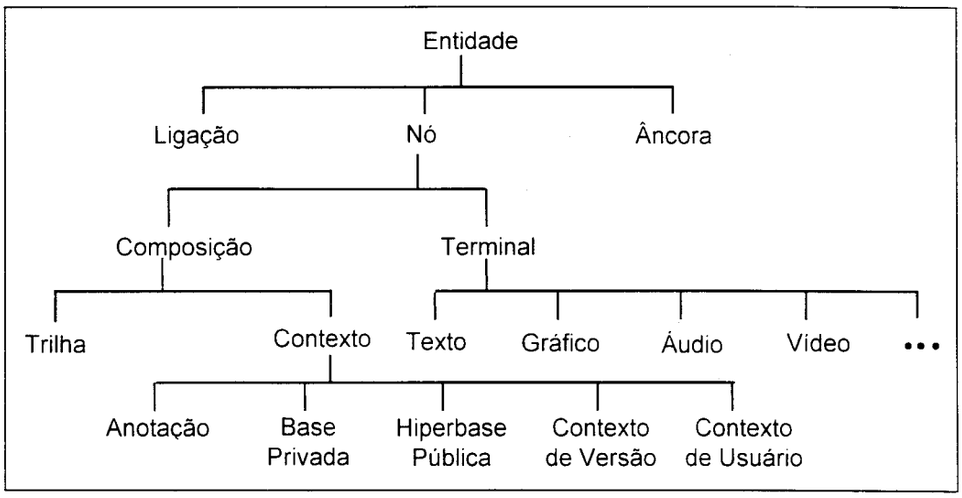


Figura 2.7: Hierarquia das classes MCA [Casanova et al. 1991].

Um nó é uma entidade que tem como atributos adicionais um conteúdo e uma lista de âncoras. Um nó pode ser terminal ou de composição. O nó terminal possui um conteúdo e uma lista de âncoras que dependem da aplicação, isto é, a estrutura interna desse nó é tratada pela aplicação e não é, portanto, modelada no MCA. Por exemplo, os nós terminais podem ser modelados como objetos multimídia especificados no padrão MHEG, e podem ser especializados em outras classes, como texto, voz, imagem, conforme requerido pela aplicação.

Um nó de composição é um nó cujo conteúdo é um conjunto de nós e ligações, permitindo organizar hierarquicamente o hiperdocumento. A classe nó de composição pode ser especializada nas classes nós contexto e trilha. Uma trilha contém uma lista ordenada de nós que representam caminhos navegacionais pré-definidos. Um nó contexto contém ligações, nós terminais, trilhas e nós de contexto, com nenhuma entrada repetida. Essa recursividade na definição permite que os contextos sejam aninhados em qualquer profundidade, e um mesmo nó possa estar contido em mais de um contexto. Segundo Batista [1994], o nó contexto permite especificar mecanismos de agregação, hierarquia e visão, além de propiciar que os documentos sejam organizados hierarquicamente e que sejam definidas diferentes visões do mesmo hiperdocumento. A classe nós de contexto pode ser especializada em outras classes, como apresentado na Figura 2.7, que

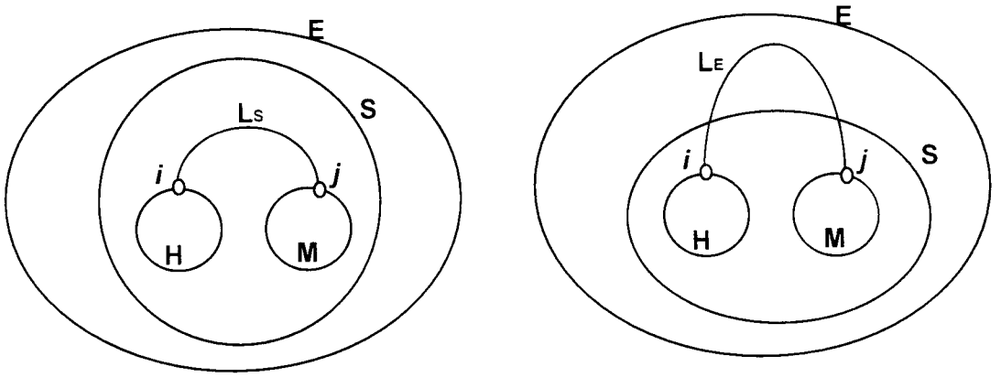
não são de interesse para a discussão em questão.

Tanto um nó terminal quanto um nó de composição contêm como atributo uma lista de âncoras. Uma âncora é uma entidade que possui como atributo adicional uma região, que especifica as entradas utilizadas por ligações com extremidade no nó. Uma região de um nó terminal pode ser um segmento (unidade de informação) do seu conteúdo ou o símbolo especial λ, que representa o conteúdo do nó terminal como inteiro. Em um nó de composição, a região associada a uma âncora deve ser o símbolo λ (novamente representa todo o conteúdo do nó) ou um subconjunto contendo apenas nós. Dessa forma, os nós contêm, além do seu conteúdo, uma máscara que atua como interface com as informações externas ao nó.

Uma ligação é a entidade usada para conectar nós, podendo ancorar no nó como um todo ou em alguma região situada no conteúdo do nó. Uma ligação é sempre direcional e pode possuir múltiplos destinos, o que permite a definição de conexões um-para-muitos. Os três atributos adicionais que especificam uma ligação são: o conjunto de extremidades de origem, o conjunto de extremidades de destino e o ponto de encontro. Os valores dos dois primeiros atributos definem o contexto da ligação e são conjuntos cujos elementos são pares nós/âncora. O atributo ponto de encontro define as condições que devem ser satisfeitas nas extremidades de origem (por exemplo, a extremidade foi selecionada ou apresentada) para que sejam executadas ações (por exemplo, exibir, suspender exibição, retomar exibição, etc) nas extremidades de destino da ligação [Souza Filho; Soares 1996].

A Figura 2.8 ilustra um exemplo de modelagem utilizando o MCA, focalizando o uso de ligações em nós de composição. Considere um hiperdocumento sobre "Poesia Inglesa do Século XVI", no qual trabalha uma equipe de teatrólogos. Suponha que tal aplicação seja modelada com um nó de composição "E" que contém um nó de composição "S", o qual agrupa as peças de *Shakespeare*. O nó "S" pode conter os nós terminais texto "H" e "M", representando as peças *Hamlet* e *Macbeth*. De acordo com a Figura 2.8(a), uma ligação conectando os nós "H" e "M" pode, em princípio, ser definida em "S" como L_S=(Origem, Destino, ponto_de_encontro), tal que Origem:<(H),i>, Destino:<(M),j> e ponto_de_encontro: Selection(Origem)→Run(Destino). As âncoras válidas em "H" e "M" são "i" e "j", e o ponto_de_encontro especifica o par condição/ação da ligação. Uma alternativa

para essa especificação seria criar uma ligação definida em "E" como L_E=(<(S,H),i>,<(S,M),j>, ponto_de_encontro), conforme ilustra a Figura 2.8(b). Note que a ligação L_S será vista por todo o documento que incluir "S" (o nó de composição agrupando as peças de Shakespeare provavelmente será compartilhado por diversos documentos), ao passo que L_E somente será vista em "S" pelos leitores que alcançarem "S" passando por "E" [Casanova et al. 1995].



(a) Ligação no contexto S. (b) Ligação no contexto E.

Figura 2.8: Ligações em nós de composição [Casanova et al. 1995, p.6].

No modelo conceitual MCA o hiperdocumento é tratado como uma estrutura de dados essencialmente passiva. Entretanto, um sistema hiperdocumento deve fornecer ferramentas para o leitor ter acesso, visualizar e manipular a estrutura organizacional do hiperdocumento. Esses aspectos são de responsabilidade do modelo de apresentação do MCA, que se encarrega de adicionar às entidades que serão exibidas as informações necessárias para a sua apresentação [Souza Filho; Soares 1995, Souza Filho; Soares 1996]. Os mecanismos de navegação oferecidos pelo MCA são: em profundidade (ação de seguir o aninhamento dos nós de composição, permitindo ao leitor mover-se para frente e para trás na composição hierárquica), pelas ligações (ação de percorrer as ligações), por consultas (o leitor pode atingir um determinado nó descrevendo as propriedades satisfeitas por esse nó), *browsers* para contextos (mostra uma visão pictórica do hiperdocumento) e por trilhas (permite aos leitores seguir trajetórias pré-estabelecidas).

2.4. MODELOS COMPORTAMENTAIS

Nesta seção são apresentados em detalhes o modelo Trellis, baseado em Redes de Petri (Seção 2.4.1), e a proposta de Zheng e Pong, baseada em Statecharts (Seção 2.4.2).

2.4.1.Modelo Trellis

O modelo Trellis, proposto por Stotts e Furuta [1989, 1990], utiliza o formalismo Redes de Petri [Peterson 1977] para especificar a estrutura de nós e ligações do hiperdocumento, além de sua semântica de navegação. O propósito do modelo é aproveitar as propriedades estáticas de dígrafos para especificar a estrutura dos documentos e as propriedades dinâmicas de Redes de Petri para definir a semântica de navegação inerente ao hiperdocumento.

Antes de descrever o modelo Trellis, são apresentadas as características principais do formalismo Redes de Petri, que constitui uma técnica eficiente para especificar e analisar o fluxo de controle em sistemas caracterizados por atividades concorrentes e de sincronização. De maneira geral, uma Rede de Petri possui dois tipos de nós: os lugares (representados graficamente por círculos) e as transições (representadas por barras). Os nós são conectados por arcos, que são representados por setas direcionadas dos lugares para as transições e vice-versa.

Formalmente, uma Rede de Petri *PN* é definida como uma 3-tupla

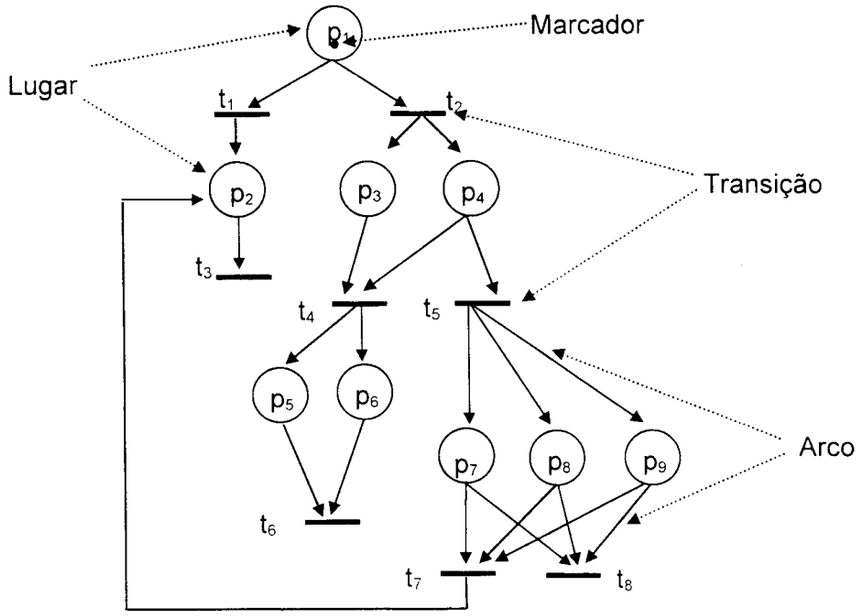
$$PN = <P, T, A>, \text{ na qual:}$$

a) $P = \{p_1, p_2, \ldots, p_n\}$ é um conjunto finito de lugares ($n \geq 0$);

b) $T = \{t_1, t_2, \ldots, t_m\}$ é um conjunto finito de transições ($m \geq 0$), e

c) $A: (P \times T) \cup (T \times P) \rightarrow \{0,1\}$ é uma função que representa os arcos que vão dos lugares para as transições e das transições para os lugares.

A execução de uma Rede de Petri é controlada pela posição e movimentação de marcadores (representados por círculos cheios) nos lugares da rede. Os marcadores definem o comportamento dinâmico da Rede, mostrando os lugares ativos em um determinado momento e as transições que podem ser ativadas. O conjunto de todos os lugares ativos em um instante define uma marcação na Rede de Petri, ou seja, representa um dos estados que o sistema pode alcançar durante a sua execução. Uma marcação μ em uma Rede de Petri é uma função $\mu: P \rightarrow N$, na qual P é o conjunto de lugares e N é o conjunto dos números naturais. Assim, uma Rede de Petri marcada possui um vetor (μ) que associa a cada lugar da rede o seu número de marcadores. Um exemplo de uma Rede de Petri com a marcação $\mu = (1,0,0,0,0,0,0,0)$ é ilustrado graficamente na Figura 2.9.



PN = <P, T, A> e μ = (1,0,0,0,0,0,0,0,0) é uma marcação de PN, em que:

P = {p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9}

T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8}

A ={ (p_1, t_1), (p_1, t_2), (p_2, t_3), (p_3, t_4), (p_5, t_6), (p_6, t_6), (p_7, t_7), (p_7, t_8), (p_8, t_7), (p_8, t_8), (p_9, t_7), (p_9, t_8), (t_1, p_2), (t_2, p_3), (t_2, p_4), (t_4, p_5), (t_4, p_6), (t_5, p_7), (t_5, p_8), (t_5,p_9), (t_7, p_2) }

Figura 2.9: Rede de Petri PN marcada.

Basicamente, o número de marcadores e sua distribuição definem a semântica de execução do formalismo Redes de Petri, segundo as seguintes regras:

a) Uma transição "t" está habilitada quando todos os seus lugares de entrada estiverem ocupados por marcadores;

b) Qualquer transição habilitada pode ser disparada;

c) Disparar uma transição "t" significa remover os marcadores de seus lugares de entrada e colocá-los nos seus lugares de saída; e

d) O procedimento para disparar transições continua enquanto houver transições habilitadas, caso contrário a execução é finalizada.

No exemplo da Figura 2.9, as transições "t_1" e "t_2" estão habilitadas e, quando

"t_2" for disparada a próxima marcação será μ = (0,0,1,1,0,0,0,0,0), que por sua vez habilita outras transições.

De maneira geral, o modelo Trellis é a primeira proposta reconhecida na literatura que utiliza o formalismo Redes de Petri para modelar hiperdocumentos. Segundo o Trellis, um hiperdocumento *H* é definido formalmente como:

$H = <PN, C, W, B, P_L, P_D>$, em que:

a) PN = <P, T, A> é uma Rede de Petri, como definida anteriormente;

b) C é um conjunto de conteúdos do documento;

c) W é um conjunto de janelas;

d) B é um conjunto de botões;

e) P_L é uma projeção lógica para o hiperdocumento que fornece um leiaute lógico para a estrutura navegacional. Tal projeção é definida por uma 3-tupla

$P_L = <C_L, W_L, B_L>$, na qual:

i) $C_L : P \rightarrow C \cup \phi$ é uma função que associa um conteúdo a cada lugar;

ii) $W_L : P \rightarrow W \cup \phi$ é uma função que associa uma janela a cada lugar;

iii) $B_L : T \rightarrow B \cup \phi$ é uma função que associa um botão a cada transição;

f) P_D é uma projeção física para o hiperdocumento que simplesmente associa as janelas (W_L) e os botões lógicos (B_L) aos dispositivos de apresentação do sistema destino.

Um hiperdocumento marcado é definido aumentando a Rede de Petri subjacente com uma marcação (μ), e representa o estado do hiperdocumento durante o processo de navegação. Além disso, especifica quais componentes (conteúdo, janela, botão) são apresentados e quais caminhos de navegação podem ser percorridos pelo leitor. Desse modo, a semântica de execução da Rede de Petri subjacente ao hiperdocumento define a semântica de navegação do mesmo, que é apresentada a seguir:

a) A navegação no hiperdocumento inicia-se com a execução da Rede de Petri subjacente na marcação inicial (μ_0), que define o estado inicial do hiperdocumento;

b) Para cada lugar que possui marcador insere-se o conteúdo associado ao lugar na janela lógica correspondente. Usando os mapeamentos físicos, os componentes lógicos são interpretados pelos dispositivos de apresentação;

c) As transições habilitadas na marcação atual determinam os botões (B_L) visíveis ou selecionáveis nas janelas lógicas (W_L) do hiperdocumento. Os botões são mecanismos que relacionam a navegação com a execução da Rede de Petri;

d) A seleção de um botão pelo leitor ativa as transições correspondentes, gerando um novo estado do hiperdocumento na navegação; e

e) O processo de navegação termina quando nenhum botão é selecionável.

O modelo Trellis permite a especificação de apresentações e caminhos de navegação concorrentes. Por exemplo, supondo o estado do hiperdocumento definido na Figura 2.9, a ativação da transição "t_2" cria dois caminhos de navegação concorrentes, mostrando os componentes $C_L(p_3)$ e $C_L(p_4)$ simultaneamente. A ativação da transição "t_5" causa na apresentação a substituição do componente $C_L(p_4)$ pelos componentes $C_L(p_7)$, $C_L(p_8)$ e $C_L(p_9)$, enquanto o componente $C_L(p_3)$ permanece visível. Esse exemplo ilustra a apresentação de componentes concorrentes e a sincronização na apresentação.

O Trellis permite separar a estrutura organizacional (representada pela Rede de Petri) dos componentes visíveis ao leitor, permitindo que componentes lógicos sejam representados de diferentes formas, dependendo do estado do hiperdocumento e da projeção de apresentação P_d. Além disso, permite resolver, elegantemente, vários problemas associados à tecnologia hipermídia, como análise de complexidade da apresentação, alcançabilidade de um nó, caminhos de navegação concorrentes, sincronização, controle de acesso a determinadas porções de informação e versões personalizadas de hiperdocumentos [Stotts; Furuta 1989, Stotts et al. 1992].

Para validar o modelo Trellis foi desenvolvido um ambiente de autoria e navegação denominado αTrellis que permite a especificação e exploração de hiperdocumentos. O modelo não possui mecanismos para especificar os relacionamentos temporais que podem existir quando informações multimídia são tratadas, limitando a especificação a informações estáticas. Novos modelos baseados no Trellis [Stotts; Furuta 1991] e em Redes de Petri foram propostos para especificar os requisitos relativos à sincronização de documentos multimídia, como o modelo MORENA (*Multimedia ORganization Employing a Network Approach*) [Botafogo; Mossé 1995], o modelo OCPN (*Object Composition Petri Net*) [Little; Ghafoor 1990] e suas derivações: XOCPN (*Extended Object Composition Petri Net*)

[Woo et al. 1994], DTPN (*Dynamic Timed Petri Nets*) [Prabhakaran; Raghavan 1993], TSPN (*Time Stream Petri Net*) [Díaz; Sénac 1993], $TSPN_{UI}$ (*Time Stream Petri Net with User Interaction*) [Cooper 1995] e MHPN (*Multimedia Hypermedia Petri Net*) [Wang; Wu 1995].

2.4.2. A Proposta de Zheng e Pong

Zheng e Pong [1992] utilizam a técnica Statecharts[6] para especificar a estrutura hierárquica e o comportamento dinâmico da interface em sistemas hiperdocumento. Segundo Zheng e Pong, a interface pode ser baseada em *frames* (por exemplo, HyperCard e InterMedia) ou em *scrolling* (por exemplo, o sistema Guide). O objetivo principal dessa proposta é modelar as transformações de interface resultantes da interação do leitor com o sistema.

Para ilustrar as transformações de interface são apresentados, a seguir, dois exemplos de situações que podem ocorrer na especificação da interface de sistemas hiperdocumento. A Figura 2.10 ilustra um Statechart que especifica o comportamento de um botão "B" (representado pelo estado s_B) em um sistema hiperdocumento baseado em *scrolling*. Inicialmente, o botão está em um estado "$S_{B\_FECHADO}$", que visualmente exibe o botão na interface. Se o leitor clicar no botão, um evento "b_{ATIVAR}" é disparado e o comportamento do botão é modificado para o estado "S_{B_ABERTO}" e, conseqüentemente, outras informações poderiam aparecer na interface. Para voltar ao estado inicial, o leitor deve clicar no botão novamente e o evento "$b_{DESATIVAR}$" é disparado. A Figura 2.11 ilustra um outro exemplo que modela o comportamento da interface de um sistema baseado em *frames*, de modo que cada *frame*, denotado por "S_{F1}", "S_{F2}", entre outros, contém um fragmento de informação do hiperdocumento. Um mesmo botão "B" (denotado por s_B) no *frame* "F_1" (estado S_{F1}) é utilizado para abrir e fechar outro *frame* "F_2" (denotado por S_{F2}) pela ativação do evento "b_{ATIVAR}", que por sua vez dispara uma ação ("F_{2_ABRIR}").

2.5. MÉTODOS DE PROJETO

Os modelos anteriores diferem dos métodos desta classe por não fornecerem uma abordagem sistemática e prática capaz de direcionar os autores nas diferentes fases ao longo do projeto, desenvolvimento e implementação de hiperdocumentos.

---

6 Caso o leitor tenha dúvidas sobre o formalismo Statecharts, recomenda-se ler o Capítulo 3 antes dessa seção.

Em cada uma das fases, modelos de dados são produzidos e/ou enriquecidos. A abordagem adotada na maioria dos métodos compreende a modelagem conceitual da aplicação, a modelagem navegacional e da interface, e uma fase de implementação seguida por avaliação e teste [Nanard; Nanard 1995].

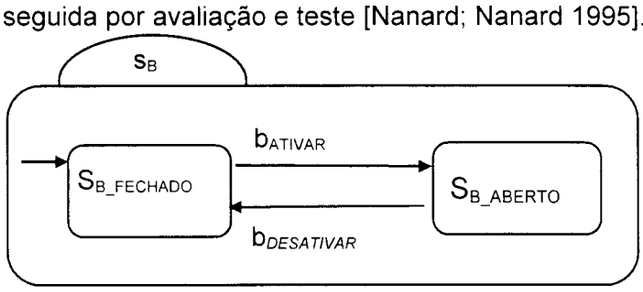


Figura 2.10: Statechart que especifica a seleção de um botão [Zheng; Pong 1992, p.245].

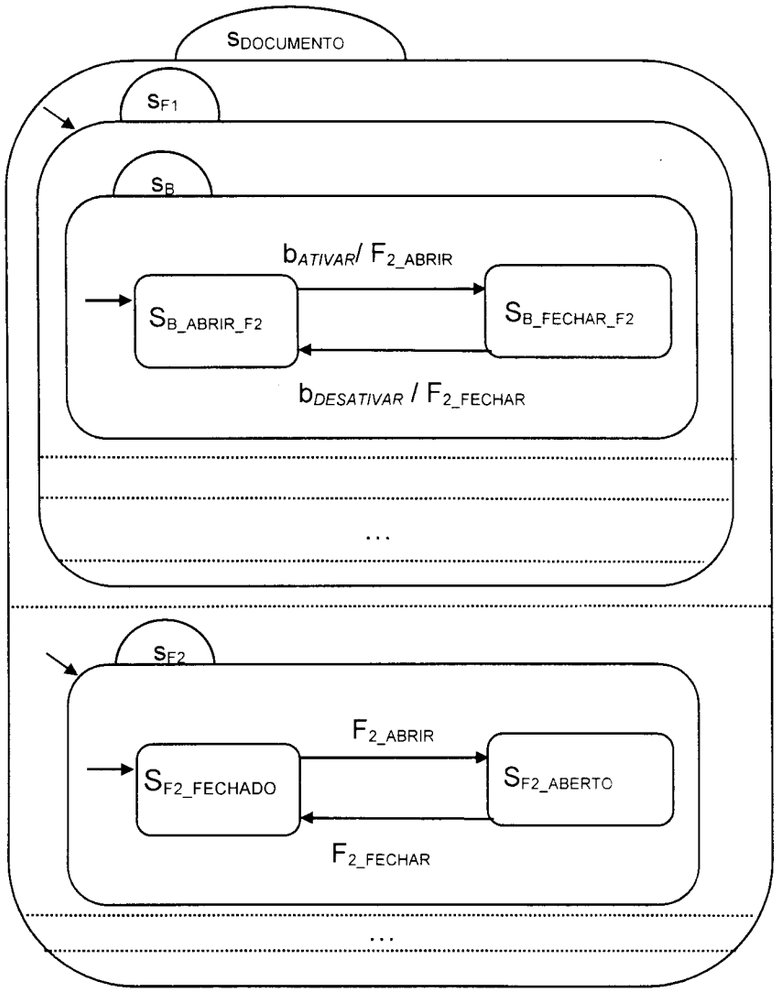


Figura 2.11: Statechart que especifica a abertura e o fechamento de um *frame* "F_2" usando um botão [Zheng; Pong 1992, p.245].

Os principais métodos dessa classe são o modelo de projeto HDM, o método orientado a objetos para o projeto de aplicações hipermídia OOHDM, o método de gestão de relacionamentos RMM, o método de projeto baseado no modelo de relacionamento de objetos enriquecido EORM, o método de projeto de hiperdocumentos de Signore [1995], o método TADANAGET (*TAsk-DAta-NAvigation-GEneration-PresenTation*) [Fraïssé 1997], que enfatiza a independência entre o modelo de dados e o modelo de navegação do hiperdocumento, e a abordagem de modelagem semântica HSM (*Hypertext Semantic Modelling*) [Nemetz et al. 1995], baseada no modelo GSM (*Generic Semantic Model*) [Hull; King 1987]. Nesta seção é apresentada uma visão geral dos modelos de especificação utilizados pelos métodos orientados à aplicação mais conhecidos: o HDM (Seção 2.5.1), o RMM (Seção 2.5.2), o EORM (Seção 2.5.3) e o OOHDM (Seção 2.5.4).

2.5.1.Modelo de Projeto HDM

O HDM é o primeiro modelo amplamente conhecido para o projeto de hiperdocumentos, voltado especialmente para modelar o domínio da aplicação (autoria em ponto grande) [Garzotto et al. 1991, Garzotto et al. 1993a]. As três fases que compreendem tal projeto são:

a) Especificar um esquema HDM que define os objetos conceituais que modelam uma classe de aplicação;
b) Especificar um hiperdocumento instanciando e satisfazendo as restrições definidas no esquema HDM construído no item acima. Essa instância apresenta somente os aspectos estáticos do hiperdocumento; e
c) Definir uma semântica de navegação que determina como os objetos são apresentados ao leitor (propriedades de visualização) e qual o comportamento dinâmico do hiperdocumento.

Do ponto de vista de notação e terminologia, o HDM compartilha similaridades com MER [Chen 1976]. Inicialmente, o modelo propõe a definição de um esquema que é posteriormente instanciado para classes de aplicações do mesmo tipo. As noções de esquema e instância de esquema permitem a reutilização de um mesmo esquema para aplicações diferentes em um mesmo domínio, compartilhando a mesma estrutura global e diferenciando apenas nas instâncias. Um esquema HDM é descrito como um conjunto de tipos de entidades e

de ligações. As entidades são hierarquicamente formadas por componentes que, por sua vez, correspondem a uma abstração para um conjunto de unidades. Uma unidade (similar ao conceito de nó) é o menor fragmento de informação que pode ser visualizado em uma aplicação HDM. Cada unidade corresponde ao conteúdo de um componente visto de acordo com uma perspectiva em particular, ou seja, a perspectiva estabelece a aparência do fragmento de informação na unidade.

As ligações em HDM podem ser estruturais, de perspectiva e de aplicação. As ligações estruturais são utilizadas para relacionar os diferentes componentes pertencentes a uma mesma entidade. As ligações de perspectivas conectam as diferentes unidades de um mesmo componente. As ligações de aplicação especificam a semântica dos relacionamentos no domínio da aplicação, e podem conectar componentes e entidades do mesmo tipo ou de tipos diferentes.

A primitiva *web* permite a representação de relações n-árias no domínio. Além disso, pode ser usada para representar estruturas de acesso, que definem os pontos de entrada para o leitor iniciar o processo de navegação. Em HDM, tais estruturas podem ser os *outlines* (estruturas tipo índice) e os roteiros guiados [Zellweger 1989, Marshall; Irish 1989, Garzotto et al. 1993b].

O HDM define uma semântica de navegação *default* que, por um processo de compilação pode-se transformar a especificação da instância do esquema HDM em um modelo intermediário baseado em grafos (modelo de nós e ligações). Em seguida, os nós e as ligações devem ser mapeados diretamente para objetos do sistema hiperdocumento destino, o qual é responsável por especificar o comportamento dinâmico e as propriedades de visualização das informações. Assim, para uma mesma especificação estática HDM pode-se definir diferentes semânticas de navegação, permitindo que o mesmo hiperdocumento tenha comportamentos dinâmicos diferentes ou que seja implementado em diferentes sistemas hiperdocumento.

Schwabe et al. [1992] descrevem o protótipo de um compilador denominado HF, que permite traduzir uma instância de um esquema HDM em uma aplicação executável em HyperCard, usando a representação de uma base de dados relacional. Caloini [1992] apresenta o HCT (*Hypertext Compiling Tools*), sucessor de HF, que contém um conjunto de ferramentas para traduzir especificações HDM em aplicações Toolbook.

Vilain e Heuser [1994] utilizaram o HDM no projeto de uma aplicação jurídica

e apresentaram várias extensões levando em consideração os requisitos da aplicação desenvolvida, tais como, cardinalidade das ligações, obrigatoriedade de uma ligação e associação de regras às ligações cuja origem é a primitiva *web*. Cavallaro et al. [1993] e Garzotto et al. [1995] propuseram o modelo HDM+, uma extensão do HDM que fornece novas primitivas de modelagem, principalmente para a autoria em ponto pequeno. Com o objetivo de organizar melhor a estrutura navegacional do hiperdocumento propõe-se o uso de coleções como uma nova primitiva de modelagem para autoria em ponto grande [Garzotto et al. 1994]. Na autoria em ponto pequeno, um componente tem sua própria estrutura interna, consistindo de *frames*. Cada *frame* é um agregado de *slots* e representa uma perspectiva. Além disso, o HDM+ fornece uma interpretação navegacional para o esquema utilizando quatro tipos de navegação: estrutural, de aplicação, de acesso e de perspectiva. Com tais extensões, as fases para a especificação de hiperdocumentos consistem em: projeto estático em ponto grande, projeto estático em ponto pequeno, projeto dinâmico em ponto grande, projeto dinâmico em ponto pequeno, leiaute em ponto grande e leiaute em ponto pequeno.

Pesquisas atuais no contexto do modelo HDM estão direcionadas para as fases de avaliação e análise de hiperdocumentos [Garzotto et al. 1995, Garzotto et al. 1997]. Garzotto et al. [1996] analisam diferentes aspectos de reuso de componentes de software e de informação no contexto do projeto de hiperdocumentos. Enfatizam que o reuso é fundamental, pois promove os critérios de consistência e previsibilidade.

2.5.2.Método RMM

Os conceitos do Método de Gestão de Relacionamentos, ou RMM, foram propostos inicialmente por Balasubramanian et al. [1994], e depois consolidados por Isakowitz et al. [1995], Balasubramanian et al. [1996] e Isakowitz et al. [1997a]. O núcleo do método RMM é o modelo de dados RMDM (*Relationship Management Data Model*), baseado no MER e nos modelos de projeto HDM e HDM+. O RMDM fornece uma linguagem gráfica para especificar a semântica do domínio (primitivas do domínio) e os mecanismos de navegação apoiados pelo hiperdocumento (primitivas de acesso).

As primitivas do domínio são divididas em primitivas do domínio E-R (entidades, atributos, relacionamentos associativos) e primitivas do domínio RMD

(fatias ou *slices*). As entidades representam os elementos básicos do domínio e são caracterizadas por um conjunto de atributos. Os relacionamentos especificam associações entre os diferentes tipos de entidades e podem ter cardinalidade "um para um" (1-1) ou "um para muitos" (1-N).

Os atributos de cada entidade são agrupados em fatias inter-relacionadas, que modelam como a informação será apresentada ao leitor. Cada entidade tem uma fatia mais representativa (*default*) e uma fatia mínima, que é uma coleção mínima de atributos utilizados como âncora para as instâncias da entidade [Balasubramanian et al. 1996]. Além disso, as fatias podem ser híbridas (*hybrid slices*), isto é, podem conter atributos de entidades diferentes e estruturas de acesso. Isakowitz et al. [1997a] estenderam o conceito de fatias para "m-fatias" (*m-slices*), permitindo agrupar e aninhar as fatias de uma entidade, formando uma hierarquia de fatias. A primitiva "m-fatia" permite descrever "qual" informação faz parte dos elementos RMDM e "como" obtê-la, permitindo uma definição precisa dos elementos de informação apresentados para o leitor.

As primitivas de acesso permitem modelar como será o processo de navegação no hiperdocumento. A navegação é apoiada em RMDM por seis primitivas de acesso: ligação uni e bidirecional, índice, roteiro guiado, roteiro guiado indexado e agrupamento. As ligações uni e bidirecionais são utilizadas para especificar o acesso entre as fatias de uma mesma entidade, podendo ser definidas dentro dos limites de uma entidade. As outras primitivas permitem modelar a navegação entre as diferentes entidades do modelo conceitual.

O método RMM divide o processo de projeto de hiperdocumentos em sete fases: projeto entidade-relacionamento (E-R), projeto de fatias, projeto navegacional, projeto de interface com o usuário, projeto de protocolos de conversão, projeto de comportamento em tempo de execução e construção/teste. Embora apresentado como um modelo cascata, o método é flexível e apóia um processo iterativo.

Durante o projeto E-R, um ME-R da aplicação é desenvolvido. Em seguida, os atributos das entidades são subdivididos em fatias representativas, produzindo um diagrama de E-R estendido (DE-R +), que compreende o DE-R gerado na fase anterior, mais o diagrama de fatias originado para cada entidade. No projeto de navegação o esquema E-R é enriquecido com as estruturas de acesso que especificam como os leitores podem navegar pelas informações. No final dessa

fase, o DE-R + se transforma em um diagrama RMDM completo. Em contraste com o DE-R, que representa o modelo conceitual, o diagrama RMDM especifica como os leitores navegam no hiperdocumento.

No projeto de interface com o usuário são especificadas como as informações serão apresentadas ao leitor. No projeto de protocolos de conversão é produzido um conjunto de regras para tradução dos componentes do diagrama RMDM em objetos físicos de um sistema hiperdocumento destino. O projeto de comportamento em tempo de execução determina como as informações do hiperdocumento são recuperadas e como é realizada a interação do leitor com o sistema. A fase de construção e teste é responsável por implementar o projeto obtido nas fases anteriores, por criar uma base de dados com as informações do domínio e por realizar testes para verificar se os requisitos funcionais, navegacionais e de usabilidade foram satisfeitos.

O RMM destina-se prioritariamente ao projeto de hiperdocumentos que possuem informações estruturadas e que requerem freqüentes atualizações (alta volatilidade). Segundo Isakowitz et al. [1995], esses critérios configuram a necessidade de meios para sistematizar e automatizar o desenvolvimento e os subseqüentes processos de atualização do hiperdocumento. Balasubramanian et al. [1996] e Isakowitz et al. [1997a] discutem algumas limitações, vantagens e desvantagens do método RMM.

A ferramenta RMCase (*Relationship Management Case Tool*) foi desenvolvida para apoiar o projeto e o desenvolvimento de hiperdocumentos baseados no método RMM [Díaz; Isakowitz 1995; Díaz et al. 1995, Isakowitz et al. 1997b]. Várias aplicações utilizando essa ferramenta e o método RMM foram projetadas, e podem mostrar a potencialidade do método: "LINKBase" apresenta informações bibliográficas da ACM SIGLINK extraídas de uma base de dados relacional (versão em WWW - http://www.acm.org/siglink/) [Balasubramanian et al. 1995, Balasubramanian et al. 1996], "*JMIS WebSite*" apresenta as informações da revista "*Journal of Management Information Systems*" [Isakowitz et al. 1997a], e o catalogo WWW do Departamento de Sistemas de Informação da Escola de Administração Leonard Stern (ISweb) [Isakowitz et al. 1995].

2.5.3.Método EORM

Lange [1994] propôs um método de projeto orientado a objetos baseado na

técnica de modelagem OMT (*Object Modeling Technique*) [Rumbaugh et al. 1991] e no modelo de dados EORM (*Enhanced Object-Relationship Model*) [Lange 1990]. O objetivo do método é modelar a semântica de interação entre os objetos de sistemas de informação baseados na tecnologia hipermídia, capturando a estrutura e o comportamento de tais objetos.

O modelo EORM permite especificar a estrutura organizacional de hiperdocumentos usando as primitivas *nós* (unidades de informação), as quais são estruturadas internamente por *slots* (*template* para o conteúdo do nó); ligações direcionadas entre os nós; *buttons* (âncoras), que são a origem das ligações no conteúdo dos nós; *fields*, que são o destino das ligações; e finalmente, a primitiva estrutura, que organiza a rede de nós e ligações do hiperdocumento (por exemplo, usando estruturas lineares ou hierárquicas). O modelo EORM não define uma semântica de navegação para interpretar o hiperdocumento.

A Figura 2.12 ilustra as três fases do método: estrutura de classe, estrutura de relacionamento e estrutura de interface com o usuário. Na fase estrutura de classe o autor deve realizar uma análise orientada a objetos do domínio da aplicação utilizando a abordagem OMT para capturar a estrutura e o comportamento dos objetos, bem como os relacionamentos semânticos entre eles. Na fase estrutura de relacionamento deve-se especificar a estrutura organizacional do hiperdocumento, integrando o modelo de análise com o modelo EORM. Por último, na fase de interface com o usuário define-se a apresentação do hiperdocumento, identificando as diferentes formas para apresentar as informações.

2.5.4.Método OOHDM

O método de projeto OOHDM (*Object Oriented HDM*) é uma extensão orientada a objetos do HDM, de modo que o processo de desenvolvimento é dividido em quatro fases, realizadas em uma combinação de estilos iterativos e incrementais de desenvolvimento: modelagem conceitual, projeto navegacional, projeto da interface abstrata e implementação (Tabela 2.1) [Schwabe; Rossi 1994, Schwabe; Barbosa 1994, Schwabe; Rossi 1995, Schwabe et al. 1996]. Os produtos gerados em cada uma das fases, os mecanismos de abstração utilizados ao longo do processo e os interesses de projeto também são ilustrados na Tabela 2.1. Tanto o OOHDM quanto o RMM compartilham da perspectiva de que o método deve

permitir uma evolução do projeto e a possibilidade de laços de realimentação, como mencionada por Isakowitz et al. [1995] e Nanard e Nanard [1995].

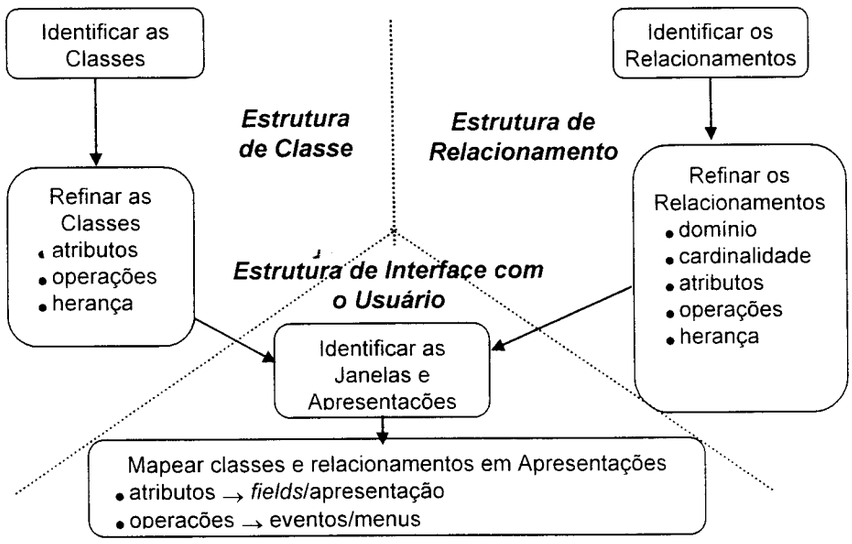


Figura 2.12: As fases do método de projeto EORM.

Tabela 2.1: Esboço das fases do método OOHDM [Schwabe; Rossi 1995, p.45].

Fases	Modelagem Conceitual	Projeto de Navegação	Projeto de Interface Abstrata	Implementação
Produtos	Classes, subsistemas, relacionamentos, e perspectivas de atributos	Nós, ligações, estruturas de acesso, contextos de navegação, e transformações navegacionais	Objetos de interface abstrata, reações a eventos externos, e transformações de interface	Aplicação executável
Mecanismos	Classificação, composição, generalização, e especialização	Mapeamento entre objetos conceituais e de navegação, e padrões de navegação para a descrição da estrutura geral da aplicação	Mapeamento entre objetos de navegação e objetos de interface	Os fornecidos pela plataforma destino
Interesses do Projeto	Modelagem semântica do domínio de aplicação	Considera o perfil do leitor, e enfatiza os aspetos cognitivos e arquiteturais	Modela os objetos perceptíveis, implementa metáforas, descreve a interface para objetos navegacionais	Desempenho e completitude

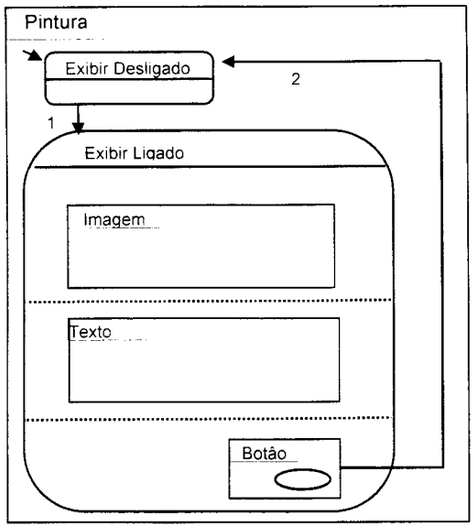
Durante a modelagem conceitual, um esquema da aplicação é produzido utilizando os princípios de modelagem orientada a objetos do método OMT [Rumbaugh et al. 1991]. Esse esquema permite especificar uma classe de hiperdocumentos em um determinado domínio com base nas primitivas: classes, relacionamentos e subsistemas. Os mecanismos agregação e generalização/ especialização podem ser usados para especificar as classes conceituais. Nesta atividade os autores não se preocupam com os tipos de leitores e os usos do hiperdocumento, apenas com a semântica do domínio da aplicação. Desse modo, as primitivas de modelagem do OOHDM são mais ricas do que as do HDM.

No OOHDM um hiperdocumento é visto como uma visão navegacional do modelo conceitual. Esta visão é construída durante o projeto navegacional levando-se em consideração os tipos de leitores aos quais a aplicação se destina, e o conjunto de tarefas a serem desempenhadas. Dessa forma, diferentes modelos navegacionais podem ser construídos para o mesmo esquema conceitual, expressando diferentes visões do mesmo domínio. Cada visão define um conjunto de contextos e classes navegacionais. Os contextos navegacionais expressam a estrutura navegacional do hiperdocumento, enquanto as classes navegacionais (tipos pré-definidos são nós, ligações, âncoras e estruturas de acesso) especificam os objetos que serão visualizados pelo leitor. Os contextos navegacionais têm função similar às coleções (*collections*) [Garzotto et al. 1994] e foram inspirados no conceito de contextos aninhados do MCA [Casanova et al. 1991], sendo definidos como um conjunto de nós, ligações e outros contextos navegacionais (aninhados) que auxiliam na organização dos objetos navegacionais [Schwabe; Barbosa 1994]. Por exemplo, um roteiro guiado apresentando todas as obras de um mesmo pintor é um contexto navegacional.

As classes e os contextos navegacionais definem a estrutura estática do hiperdocumento. O comportamento da navegação é especificado por Diagramas de Navegação, que podem ser facilmente mapeados para o formalismo Statecharts. Esses diagramas permitem especificar também as transformações navegacionais como, por exemplo, expressar quais nós de informação são apresentados para o leitor, quais se mantêm abertos cada vez que uma ligação é percorrida, quais nós podem ser abertos simultaneamente e quais nós são mutuamente exclusivos, ou seja, nunca estão acessíveis ao mesmo tempo.

Para especificar o modelo de interface abstrata, o método OOHDM usa o modelo de "Visões de Dados Abstratas" ou ADVs (*Abstract Data Views*), criado originalmente para especificar formalmente a separação entre a interface e os componentes de software. O ADV oferece um método de projeto independente da implementação, proporcionando uma maior reusabilidade de componentes de projeto e de interface [Rossi et al. 1995, Cowan; Lucena 1995, Rossi 1996]. Para especificar os padrões dinâmicos da interface de hiperdocumentos, o OOHDM usa o formalismo visual ADVcharts, que é uma generalização dos Statecharts [Harel 1987] e dos ObjectCharts [Coleman et al. 1992], apoiando o aninhamento de estados e de ADVs, além de permitir a associação entre eventos externos e ADVs. Um ADVchart é composto por ADVs, estados, atributos e transições. O aninhamento de ADVs permite mostrar a estrutura de agregação dos objetos de interface.

Na Figura 2.13 o ADV "Pintura" pode estar no estado "Exibir Desligado" ou "Exibir Ligado", e a transição "1" será ativada pelo evento externo "Exibir". O estado AND "Exibir Ligado" é formado por três subestados, correspondendo aos ADVs Imagem, Texto e Botão. A transição "2", cuja origem é o ADV "Botão" e o destino é "Exibir Desligado", é ativada quando o evento "MouseClicado" for gerado pelo leitor no ADV "Botão". Essa transição é descrita utilizando-se a função foco (o que significa que o cursor está localizado no ADV correspondente), fazendo com que a pintura corrente deixe de ser visualizada (contextoPerceptivo = ContextoPerceptivo - Pintura) e a mensagem âncoraSelec(pintor) seja enviada.



Transições

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

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

Figura 2.13: Exemplo de um ADVchart [Rossi 1996, p.102].

Assim, durante o projeto de interface abstrata, um modelo é construído e deve especificar quais objetos de interface serão vistos pelo leitor e, particularmente, a forma de apresentação dos diferentes objetos navegacionais, as transformações causadas pela ativação desses objetos, quais objetos ativarão a navegação, a maneira como os objetos multimídia serão sincronizados e quais transformações ocorrerão na interface. A separação entre a interface abstrata e o modelo navegacional permite construir interfaces diferentes para o mesmo modelo navegacional adequando-se, portanto, às variações de necessidades e preferências dos leitores.

Finalmente, os modelos navegacionais e de interface gerados usando o método OOHDM devem ser traduzidos para um ambiente de implementação. Os modelos produzidos podem ser implementados de forma direta em plataformas hipermídia disponíveis, tais como Hypercard, Toolbook, Director e HTML [Schwabe et al. 1996]. Algumas estratégias para tradução de um artefato de projeto gerado com o OOHDM para um programa Toolbook são descritas por Rossi [1996].

Publicações mais recentes sobre o OOHDM enfatizam a necessidade de padrões de projeto no processo de desenvolvimento de hiperdocumentos, promovendo a reusabilidade de estilos de navegação e interface. Além do reuso de componentes de software e de informação no contexto do projeto de hiperdocumentos proposto por Garzotto et al. [1996], Rossi et al. [1997] propõem reutilizar a experiência de projeto ou de estruturas de projeto. Eles apresentam padrões de projeto (*design patterns*) para o desenvolvimento de sistemas hiperdocumentos e para o projeto de hiperdocumentos (projeto navegacional e de interface, mais especificamente).

2.6. CONSIDERAÇÕES FINAIS

Este capítulo apresentou o estado da arte na área de modelos e métodos para a especificação e o projeto de hiperdocumentos, além de modelos de referência para sistemas hiperdocumento. Inicialmente, foi proposta uma classificação que identifica cinco abordagens distintas de modelagem na área de tecnologia hipermídia: modelos fechados, modelos orientados ao sistema, modelos estruturais, modelos comportamentais e métodos de projeto.

As classes de maior interesse para esta tese são as de modelos estruturais e de modelos comportamentais. Os modelos comportamentais, por natureza,

especificam tanto a estrutura organizacional quanto formalmente a semântica de navegação, característica não explorada nos modelos estruturais.

Essa revisão permitiu constatar que as características da técnica Statecharts, um formalismo semelhante a Redes de Petri, podem ser utilizadas para modelar a estrutura organizacional e a semântica de navegação de hiperdocumentos. Tal fato garante o embasamento teórico necessário à definição de um novo modelo, objeto de estudo e pesquisa desta tese. Os próximos capítulos apresentam, respectivamente, uma visão geral do formalismo Statecharts e o modelo proposto como uma das contribuições desta tese.

Capítulo 3

STATECHARTS

3.1. CONSIDERAÇÕES INICIAIS

A principal característica do modelo proposto nesta tese é a utilização de Statecharts como técnica subjacente para a especificação estrutural e comportamental de hiperdocumentos, de maneira similar ao uso de Redes de Petri pelo modelo Trellis [Stotts; Furuta 1989]. O objetivo deste capítulo é apresentar sucintamente os principais conceitos de Statecharts relevantes para a compreensão do modelo HMBS.

Está além do escopo deste capítulo apresentar uma especificação completa de Statecharts e exemplos do seu uso, que podem ser encontrados em outras publicações [Harel 1987, Harel et al. 1987, Harel 1988, Harel et al. 1990, Huizing; Roever 1991, Harel; Naamad 1996, Harel; Gery 1997]. Na Seção 3.2 apresenta-se a definição da técnica Statecharts e comenta-se sobre algumas de suas aplicações. As características dessa técnica são descritas informalmente na Seção 3.3. A especificação formal de Statecharts compreendendo sua sintaxe e sua semântica operacional simplificada é apresentada na Seção 3.4. Finalmente, as conclusões são apresentadas na Seção 3.5.

3.2. DEFINIÇÃO

A técnica gráfica conhecida como Statecharts foi proposta por Harel [1987] originalmente para a especificação comportamental de uma classe particular de sistemas denominada sistemas reativos. Os Statecharts podem ser caracterizados como uma linguagem de especificação, como um formalismo e como uma tecnologia. Como linguagem, os Statecharts são visuais, podendo ser descritos e compreendidos graficamente usando uma notação visual. Como formalismo possuem sintaxe e semântica definidas formalmente, possibilitando que a especificação do problema modelado seja analisada e validada automaticamente. Como tecnologia, podem existir ferramentas que permitem a criação, modificação, análise e execução de Statecharts. Há atualmente várias ferramentas que utilizam Statecharts para o desenvolvimento de sistemas com comportamentos complexos, tais como Statemate [Harel et al. 1990, Harel; Naamad 1996], Argonaute [Maraninchi 1989], StatSim [Masiero et al. 1991] e BetterState [Ali 1995].

Os Statecharts estendem o formalismo convencional das Máquinas de Estados Finitos (MEFs) de forma que, além da noção de estados e transições, definem-se hierarquia de estados, concorrência ou ortogonalidade entre estados, e um mecanismo de propagação de eventos (*broadcast*) que possibilita a comunicação entre estados. Com tais extensões, o desenvolvimento *top-down* ou *bottom-up* de sistemas é favorecido e torna-se possível a "fatoração" de transições disparadas por um mesmo evento, permitindo criar diagramas concisos, claros e intuitivos.

Algumas aplicações que empregam Statecharts são, por exemplo, a especificação de diálogos em interfaces usuário-computador [Harel 1987, Lucena 1997], de sistemas de tempo real, de linguagens de programação para sistemas concorrentes [Jahanian; Mok 1994], de hardware e de protocolos de comunicação. Recentemente, a técnica Statecharts foi proposta para ser utilizada no método de modelagem orientado a objetos UML (*Unified Modeling Language*) [Harel; Gery 1997]. UML é uma coleção de técnicas voltadas para a especificação, visualização, construção e documentação de software. Zheng e Pong [1992], como descrito no Capítulo 2, utilizam Statecharts para especificar o comportamento da interface de sistemas hiperdocumento.

Várias extensões de Statecharts, tais como sobreposição de estados [Harel; Kahana 1992], Statecharts temporais (*Timed Statecharts*) [Kensten; Pnuelli 1992],

abstrações de transições e estados parametrizados [Paulo 1997] foram propostas na literatura, cada uma procurando satisfazer a determinados requisitos, originando diversas variações de Statecharts. Beeck [1994] apresenta uma breve comparação de aproximadamente vinte variações, propostas por diferentes autores. Nesta tese empregam-se os conceitos e as características de Statecharts propostas originalmente por Harel [1987]. Entretanto, é importante destacar que nem todas as características da definição original são utilizadas na especificação de hiperdocumentos proposta pelo modelo HMBS. Para exemplificar as características de Statecharts ao longo deste capítulo utiliza-se o exemplo apresentado na Figura 3.1.

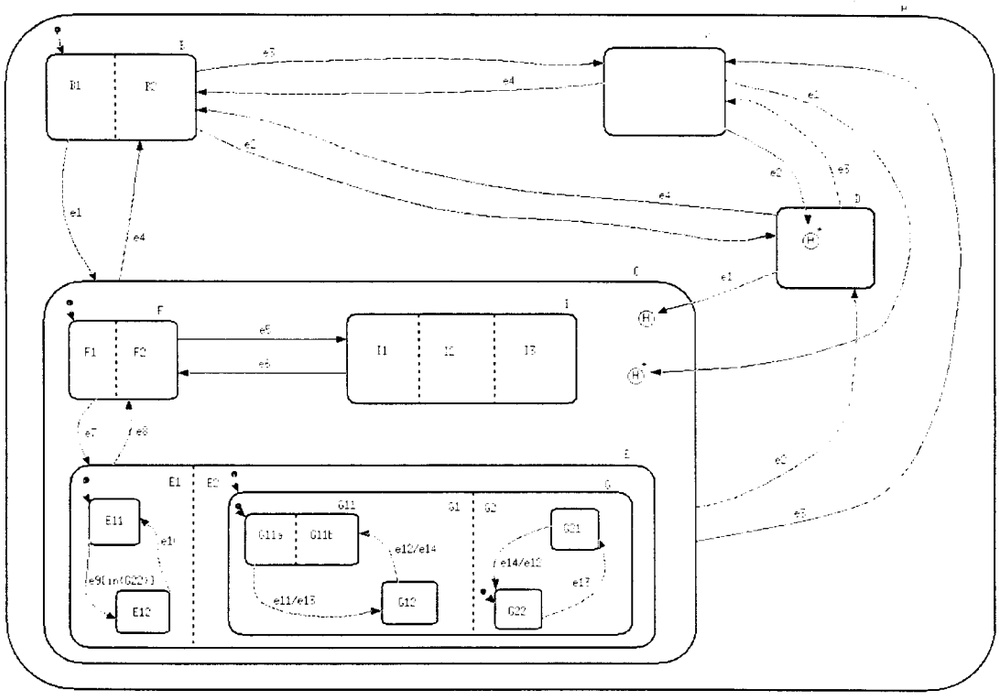


Figura 3.1: Exemplo de um Statechart.

3.3. DESCRIÇÃO INFORMAL

3.3.1. Características

A definição de Statecharts é baseada em um conjunto de estados, transições e eventos primitivos, condições primitivas e variáveis. A partir desses conjuntos de elementos básicos, definem-se os conjuntos de eventos, condições, expressões, ações e rótulos, além dos inter-relacionamentos entre eles.

Os estados permitem ao projetista especificar um contexto no qual o sistema deverá se encontrar durante a sua execução e são representados graficamente por retângulos com cantos arredondados. Há três tipos de estados: atômicos (ou básicos), OR e AND. Assumindo como exemplo o Statechart da Figura 3.1, os estados "B_1", "B_2", "E_{11}", "E_{12}", "F_1", "F_2", "G_{11a}", "G_{11b}", "G_{12}", "G_{21}", "G_{22}", "I_1", "I_2" e "I_3" são estados atômicos, pois não se decompõem em subestados. Como os Statecharts permitem decomposição hierárquica de estados (isto é, estados podem ser agrupados em outros estados formando níveis), diz-se que um estado "s_1" é subestado de "S" quando este for ancestral direto ou indireto de "s_1" ao longo da hierarquia.

Se um estado não for atômico, a sua decomposição em subestados pode ser do tipo OR ou AND. Os estados "A", "C", "D", "E_1", "E_2", "G_1", "G_2" e "K" são decomposições do tipo OR. O estado "A", por exemplo, é formado por uma abstração ou agregação de "B", "C", "D" e "K" e o estado "C" contém os subestados "E", "F" e "I". Assim, diz-se que "E", "F" e "I" são estados-filhos de "C" e, reciprocamente, que "C" é estado-pai de "E", "F" e "I". Todo Statechart possui um estado que não tem qualquer estado-pai, denominado raiz do Statechart. No Statechart ilustrado na Figura 3.1 o estado raiz é "A".

Diz-se que o sistema modelado pelo Statechart está em um estado quando se encontra em uma certa situação em um instante considerado. Assim, se o sistema está em "C" então está exclusivamente em apenas um de seus subestados, "E", "F" e "I", nunca em todos simultaneamente. Por outro lado, se está em "B", um estado do tipo AND, então está em todos os seus subestados simultaneamente, ou seja, em "B_1" e "B_2".

Quando a especificação em Statechart torna-se muito complexa, o que acontece com modelagens de problemas reais, pode-se decompor o Statechart em diagramas separados. Tal característica é denominada *unclustering* ou *zooming*, e permite a visualização em níveis de abstração diferentes, conservando a mesma hierarquia. Por exemplo, os estados "D" e "K" (tipo OR) do Statechart da Figura 3.1 são decompostos em outros diagramas. Na Figura 3.2 é apresentado o diagrama de decomposição do estado "D".

Os eventos, sejam eles externos ou internos, permitem que as especificações em Statecharts interajam com o ambiente externo. Fisicamente, os eventos correspondem a ações do usuário sobre dispositivos de entrada ou são gerados

pela aplicação como conseqüência de uma transição. Os eventos primitivos são aqueles que ocorrem no ambiente do sistema especificado. Além desses eventos, os Statecharts permitem especificar o evento nulo[7] (λ) e expressões do tipo *true*(*condição*)[8], *false*(*condição*)[9], *exit*(*estado*)[10] e *entered*(*estado*)[11]. Combinações de quaisquer das expressões de eventos com operadores lógicos são consideradas eventos, como também as expressões do tipo "*evento*[*condição*]".

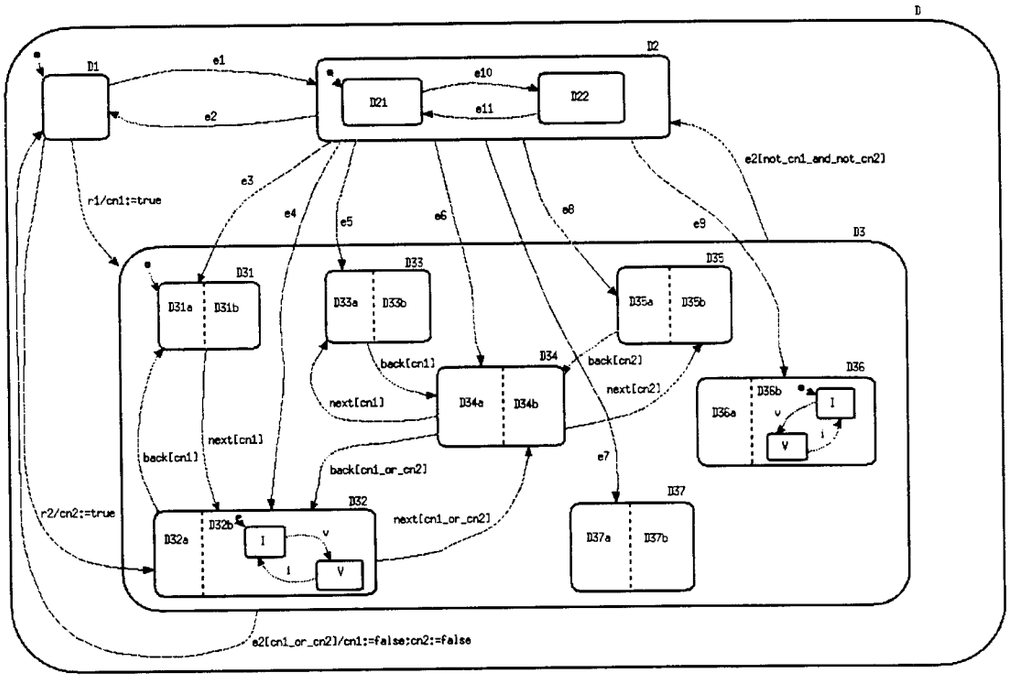


Figura 3.2: Decomposição do estado "D" do Statechart da Figura 3.1.

As ações em Statecharts são utilizadas, principalmente, para expressar saída de eventos para o ambiente externo, permitindo afetar componentes ortogonais. Conceitualmente, a ação não consome tempo e gera uma "reação em cadeia" (*broadcasting*).

As transições indicam o relacionamento seqüencial entre estados e são consideradas o único meio pelo qual um estado pode ser atingido, explícita ou implicitamente. As transições são representadas por setas unidirecionais que ligam os estados de origem aos de destino, e podem ser do tipo um para um (1:1), que

---

7 Evento gerado a cada passo de execução ou simulação do sistema.

8 Evento que ocorre quando o valor da *condição* se torna verdadeiro.

9 Evento que ocorre quando o valor da *condição* se torna falso.

10 Evento que ocorre quando o sistema sai do *estado*.

11 Evento que ocorre quando o sistema entra no *estado*.

têm somente um estado origem e um destino, e do tipo M para N (M:N), que podem ter múltiplos estados origem e/ou múltiplos estados destino. Porém, as transições denominadas *default*, representadas pelo símbolo "↴" na borda de estados, não definem ligação entre estados e são percorridas durante um processo de ativação quando nenhuma outra alternativa é definida. Na Figura 3.3, transições *default* chegam à borda de "A" e de "C", que são considerados estados *default*.

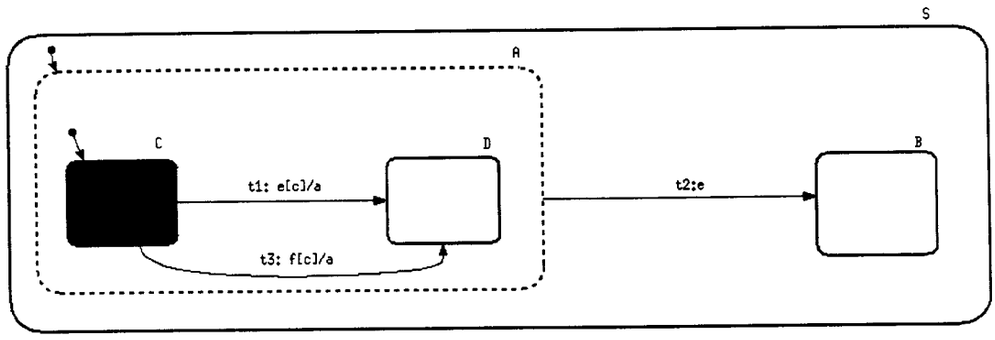


Figura 3.3: Exemplo genérico da estrutura de uma transição.

Genericamente, a sintaxe do rótulo de uma transição é "$\alpha[\delta]/\beta$", na qual "α" é a expressão contendo os eventos necessários para que a transição ocorra; "δ" é uma expressão condicional aplicada sobre a transição que, caso seja falsa, a inibe; e "β" é uma expressão que indica as ações a serem disparadas quando a transição ocorrer. Todos esses elementos de um rótulo podem ser opcionais, e as transições *default* só podem ter ações associadas ao rótulo.

Na Figura 3.3 é ilustrado um Statechart que possui três transições ("t_1", "t_2" e "t_3") e cinco estados ("S", "A, "B", "C" e "D"). Graficamente, o estado raiz, os estados hachurados e os que possuem borda pontilhada representam os estados ativos, isto é, que fazem parte da configuração de estados na qual o sistema se encontra em um determinado momento. Neste exemplo, o estado atômico "C" está ativo e, quando a transição "t_3: f[c]/a" for disparada, ou seja, se a expressão condicional "c" for verdadeira e o evento "f" ocorrer, a ação "a" será executada e o estado "D" será ativado.

O conflito entre transições diferentes associadas ao mesmo evento pode levar a uma situação de não-determinismo no Statechart, como apresentado na Figura 3.3. De acordo com a semântica definida por Harel [1987], não se define se "t_1" ou "t_2" será executada quando o estado "C" estiver ativo e o evento "e" ocorrer com a condição "c" verdadeira. Algumas situações não-determinísticas podem

desaparecer com a introdução de prioridades para a execução de transições. Por exemplo, o conceito de escopo de uma transição pode ser utilizado para estabelecer prioridades. Define-se escopo de uma transição como o estado do tipo OR no nível mais baixo da hierarquia que é o ancestral comum a ambos os estados origem e destino da transição. Pode-se estabelecer que têm prioridade as transições com escopo em nível mais alto. Na Figura 3.3, o escopo de "t_1" é o estado "A" e o escopo de "t_2" é "S". Desse modo, define-se que "t_2" tem prioridade sobre "t_1", pois seu escopo (estado "S") se encontra em um nível mais alto da hierarquia do que o estado "A", subestado de "S".

Um conceito adicional em Statecharts é o de história, que permite descrever um modo alternativo de ativação de subestados. Graficamente, a história é representada pela letra "H" inserida em um pequeno círculo. O Statechart ilustrado na Figura 3.1 contém os dois tipos possíveis de história: simples (H) e recursiva (H^*). A entrada por história em um estado "X" do tipo OR conduz ao subestado mais recentemente visitado, o que pode, potencialmente, diferir da configuração determinada pelas transições *default*. No mecanismo de história simples considera-se apenas o estado mais recentemente visitado no mesmo nível de entrada da história, e nos níveis mais baixos prossegue-se com as transições *default*. Caso contrário (história recursiva), considera-se sempre o estado mais recentemente visitado em todos os níveis hierárquicos do estado destino da transição.

No Statechart da Figura 3.1 tem-se uma entrada no estado "C" por história simples e uma outra por história recursiva. O disparo do evento "e_1" quando o estado "D" está ativo causa a ativação de "C", cujo subestado ativado será o mais recentemente visitado entre "E", "F" e "I". Se o subestado mais recentemente visitado foi "E", seguindo as transições *default* ativam-se seus subestados "E_{11}", "G_{11a}", "G_{11b}" e "G_{22}". Entretanto, se o evento "e_1" ocorrer quando o estado "K" está ativo, a entrada em "C" será por história recursiva (H^*) tornando ativos por história os subestados de C em todos os níveis hierárquicos. Por exemplo, se o subestado de C mais recentemente ativado foi "E", seus subestados mais recentemente visitados serão ativados.

A estrutura geral de um Statechart pode ser representada por uma árvore AND/OR com profundidade "d", cujas folhas denotam os estados atômicos. Os estados podem ser repetidamente decompostos em componentes OR ou componentes AND, e estados origem e destino de uma transição residem em

UFSC — SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

qualquer nível da árvore. A Figura 3.4 esquematiza a árvore AND/OR que representa a hierarquia de estados do Statechart da Figura 3.1 na qual, para simplificar, não foram representadas as decomposições dos estados "D" e "K". Nesse exemplo, os nós da árvore estão associados a estados do Statechart e cada nó contém representado o nome do estado e o seu tipo de decomposição, que pode ser "^" (AND) ou "v" (OR).

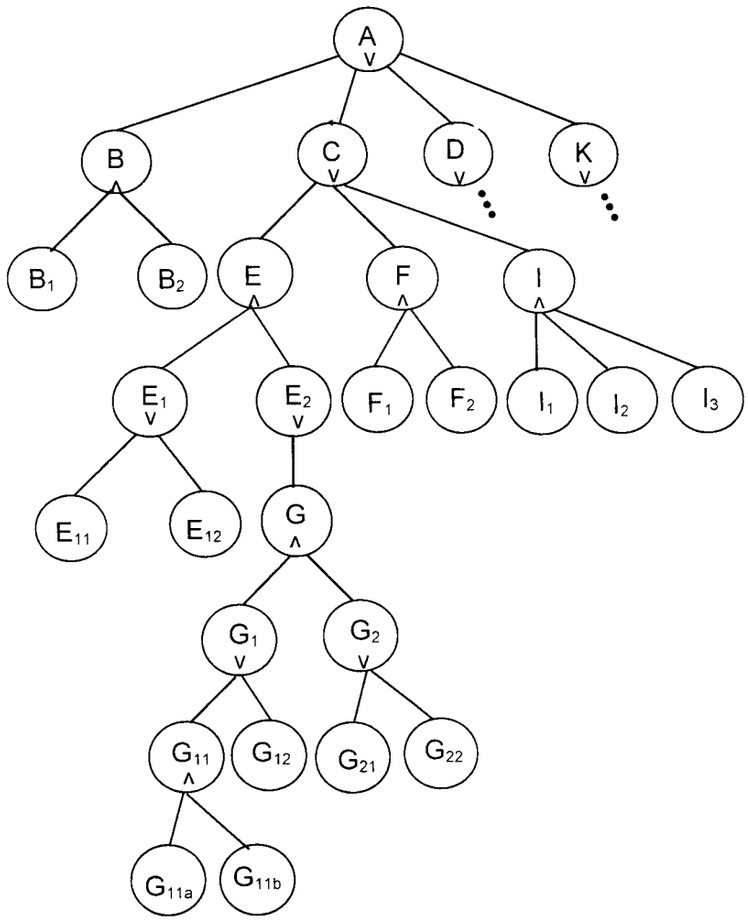


Figura 3.4: Árvore AND/OR do Statechart da Figura 3.1.

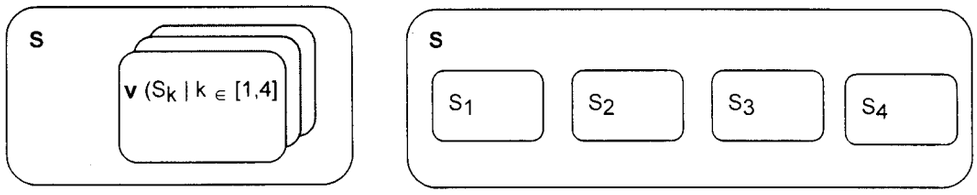
3.3.2. Parametrização de estados

A notação de estados parametrizados, como sugerida por Harel [1987], permite que estados e suas respectivas transições que definem comportamentos semelhantes sejam representados uma única vez por um estado e um conjunto de transições acompanhados de uma notação de indexação, a qual determina um intervalo de parametrização sobre o qual se definem os estados. Por exemplo, a definição de um intervalo de parametrização [1,10] é equivalente a dez estados e

mais um conjunto de transições em uma notação não parametrizada.

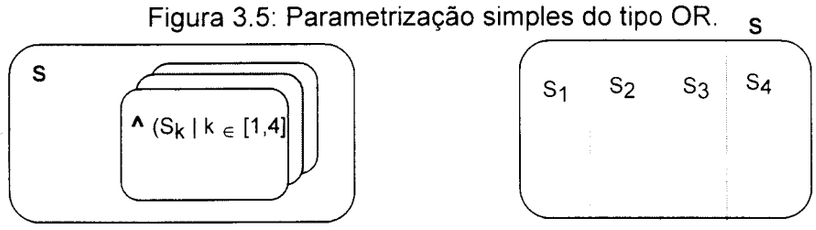
Como a proposta de parametrização de Harel não é completamente definida, Paulo et al. [1997] modificaram a notação de forma a aumentar o seu poder de especificação. A parametrização proposta é sempre relativa a um superestado cujos subestados são descritos pela notação parametrizada e usa os operadores matemáticos OR (v) e AND ($\wedge$).

A declaração da forma de parametrização de um estado inclui, além do seu tipo (OR ou AND), o nome do superestado, o nome genérico dos subestados que serão parametrizados, a variável de parametrização (ou índice) e a definição do intervalo de parametrização. A Figura 3.5 (a) apresenta um exemplo de parametrização simples do tipo OR. O conjunto de subestados é determinado pelo índice, que varia dentro do intervalo definido. A Figura 3.5 (b) mostra como seria o Statechart equivalente à Figura 3.5 (a) em notação expandida. Uma parametrização simples do tipo AND, assim como o Statechart equivalente, são mostrados na Figura 3.6.



(a) Estado parametrizado. (b) Decomposição da parametrização.

Figura 3.5: Parametrização simples do tipo OR.



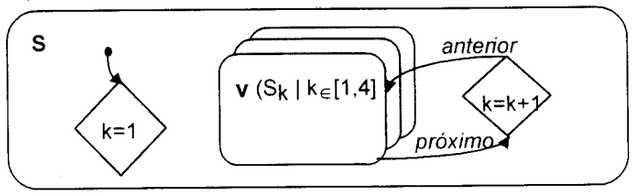
(a) Estado parametrizado. (b) Decomposição da parametrização.

Figura 3.6: Parametrização simples do tipo AND [Paulo 1997, p.81].

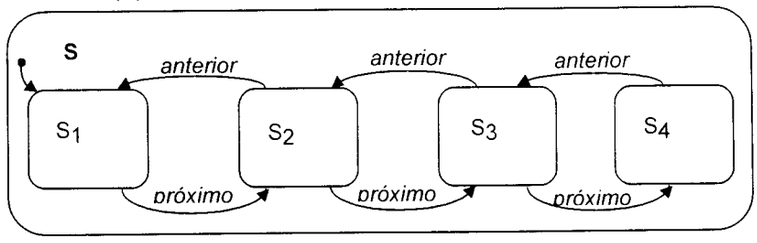
O uso de transições associadas aos subestados parametrizados requer uma notação apropriada que permita especificar qual é exatamente o estado envolvido na origem ou no destino de uma determinada transição. Assim, é necessário especificar o valor da variável de parametrização que está sendo referenciada pela transição. A Figura 3.7 apresenta o exemplo da Figura 3.5 com transições associadas à parametrização. A representação gráfica de um losango é adotada para incluir a sentença que especifica a transição parametrizada. Tal sentença,

chamada sentença de valoração, é uma expressão aritmética que determina o valor da variável de parametrização. Quando a sentença de valoração resulta em um valor não definido no intervalo de parametrização, a transição associada não é definida.

No exemplo da Figura 3.7 (a), as sentenças de valoração são utilizadas para três definições de parametrização. A primeira delas ("k=1") define o estado "S_1" como o subestado *default* do estado "S". As outras duas permitem especificar as transições entre os estados, uma associada ao evento "próximo" e outra associada ao evento "anterior". Como mostra a Figura 3.7 (b), quando essa parametrização é expandida, duas exceções ocorrem: a ausência de uma transição associada ao evento "anterior" partindo do estado "S_1" e de uma transição associada ao evento "próximo" partindo de "S_4". Isso porque, nesses casos, o valor da variável de parametrização não está definido (k=5), não sendo possível definir o estado de destino dessas transições, o que as torna igualmente indefinidas.



(a) Estado parametrizado com transição.

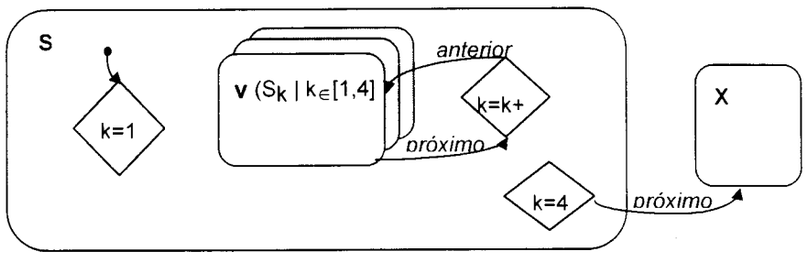


(b) Decomposição da parametrização.

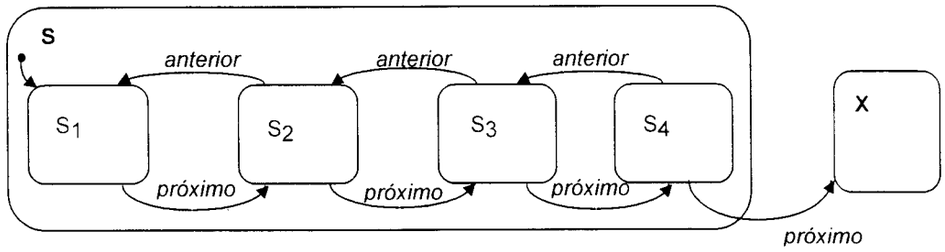
Figura 3.7: Parametrização do tipo OR com transições associadas.

É comum, no entanto, definir exceções à parametrização, principalmente para o primeiro e o último estados de uma seqüência encadeada por transições. A solução para esse problema é utilizar a reescrita de comportamento que, em termos de notação, é a definição de uma sentença de valoração com valor constante, representando explicitamente um estado que pode ser origem ou destino de transições associadas a estados externos à notação parametrizada [Paulo et al. 1997]. A Figura 3.8 apresenta uma modificação do exemplo da Figura

3.7 no qual o comportamento do estado "S_4" foi reescrito e possui todas as características definidas nos outros termos da parametrização, além de ser origem da transição definida pela reescrita de seu comportamento.



(a) Estado parametrizado.



(b) Decomposição da parametrização.

Figura 3.8: Parametrização do tipo OR com reescrita de comportamento.

3.4. ESPECIFICAÇÃO FORMAL

3.4.1. Sintaxe

A sintaxe da técnica Statecharts, conforme descrita por Harel et al. [1987], é apresentada nesta seção. O modelo HMBS apresentado no Capítulo 4 utiliza um subconjunto dessa sintaxe. As características de Statecharts não utilizadas pelo modelo são destacadas nesta seção pelo uso de fonte diferenciada e de estilo itálico.

A estrutura de um Statechart (ST) é uma 11-tupla

$$ST = <S,\rho,\psi,\gamma,\delta,V,C,E,A,R,T>, \text{ na qual:}$$

a) $S = \{s_1, s_2, \ldots, s_n\}$ é o conjunto finito formado pelos estados do Statechart, com $n > 0$;

b) ρ: $S \rightarrow 2^S$ é a função de hierarquia, responsável pelo mapeamento de cada estado $s \in S$ a um conjunto $Z \subset 2^S$, definindo, assim, os subestados de cada estado na hierarquia do Statechart. Se $x \in S$, $y \in S$ e $\rho(x) = \rho(y)$, então $x = y$, isto é, não existem dois estados distintos com o mesmo

conjunto de subestados. Um estado $s \in S$ é básico quando $\rho(s) = \phi$, ou seja, quando não possui subestados. Existe um único estado $r \in S$ tal que $\forall s \in S, r \notin \rho(s)$, ou seja, "r" é um estado que não possui um ancestral na hierarquia e é denominado estado raiz do Statechart. As funções ρ^* e ρ^+ são extensões de ρ, definidas por:

$$\rho^+(s) = \bigcup_{i \geq 1} \rho^i(s) \qquad \rho^*(s) = \bigcup_{i \geq 0} \rho^i(s)$$

Para $s \in S$, $\rho^i(s)$ representa o conjunto dos estados descendentes diretos de "s" no nível de hierarquia "i". Por definição $\rho^0(s) = \{s\}$. Assim, $\rho^*(s)$ é o conjunto formado por "s" e por todos os seus descendentes na hierarquia, e $\rho^+(s)$ é o conjunto formado por todos os estados descendentes de "s" na hierarquia;

c) ψ: $S \rightarrow$ {AND,OR} é a função tipo de decomposição, que define o tipo de decomposição hierárquica para os estados não básicos, ou seja, para cada estado $s \in S \mid \rho(s) \neq \phi$;

d) γ: $H \rightarrow S$ é a função história, responsável pelo mapeamento dos símbolos de história aos estados do Statechart. Somente estados OR podem ser associados a símbolos de história, isto é, $\gamma(H) = y$ se $y \in S$ e $\psi(y) = \{OR\}$. Se $h_1 \in H$ e $h_2 \in H$ e $\gamma(h_1) = \gamma(h_2)$ então $h_1 = h_2$;

e) δ: $S \rightarrow 2^S$ é a função default, que define, para um estado $s \in S$, um conjunto de estados iniciais ou default. Se $x \in \delta(s)$, então para $x \in S$, $x \in \rho^+(s)$. Diz-se que $\delta(s)$ é o conjunto default de "s", ou seja, $\delta(s)$ define os subestados de "s" que serão ativados quando "s" for ativado (exceto quando houver algum símbolo de história associado a "s");

f) V é o conjunto de expressões do Statechart, o qual contém números e variáveis, e é definido indutivamente, como descrito a seguir:

- *Se k é um número, então $k \in V$, isto é, um número é uma expressão;*
- Seja V_p o conjunto de variáveis primitivas do Statechart. Se $v \in V_p$, então $v \in V$, isto é, uma variável é uma expressão;
- *Se $v \in V$, então current(v) $\in V$, isto é, o resultado da função que testa o valor corrente de uma variável é uma expressão. A expressão current(v) pode ser abreviada por cr(v);*

- *Se $v_1 \in V$, $v_2 \in V$ e $O \in \{+,-,*,/\}$ então v_1 O $v_2 \in V$, isto é, uma expressão algébrica é uma expressão;*

g) C é o conjunto de condições do Statechart, definido indutivamente a seguir:

- *true* $\in$ C e *false* $\in$ C são condições;
- Seja C_p o conjunto de condições primitivas do Statechart. Se c $\in$ C_p, então c $\in$ C, isto é, uma condição primitiva é uma condição;
- Se s $\in$ S, então *in*(s) $\in$ C, isto é, o teste se o sistema está em um estado "s" é uma condição;
- *Se $s \in S$, então not_yet(s) $\in C$, isto é, o teste se o sistema não está em um estado "s" é uma condição;*
- Se u $\in$ C, v $\in$ C então u *or* v $\in$ C, u *and* v $\in$ C e *not* u $\in$ C, isto é, uma condição pode ser composta pela combinação lógica de outras condições segundo os operadores lógicos *and*, *or*, e *not*;
- *Se $u \in V$, $v \in V$ e $R \in \{=, >, <, \neq, \leq, \geq\}$ então u R v $\in C$, isto é, uma expressão relacional é uma condição;*
- *Se $c \in C$ então current(c) $\in C$, isto é, o valor corrente de uma condição é uma condição;*

h) E = {e_1, e_2, ..., e_n} é o conjunto de eventos do Statechart. Pode ser definidc indutivamente como segue:

- $\lambda \in$ E, λ é o evento nulo;
- Seja E_p o conjunto de eventos primitivos do Statechart. Se e $\in$ E_p, então e $\in$ E, isto é, um evento primitivo é um evento;
- *Se $c \in C$, então true(c) $\in E$ e false(c) $\in E$, isto é, a mudança do valor lógico de uma condição é um evento;*
- *Se $v \in V$, então changed(v) $\in E$, isto é, a mudança do valor de uma expressão é um evento;*
- *Se $s \in S$, então exit(s) $\in E$ e entered(s) $\in E$, isto é, ocorre um evento no momento em que o sistema desativa ou ativa um estado;*
- *Se $e_1 \in E$, $e_2 \in E$ então e_1 and $e_2 \in E$, e_1 or $e_2 \in E$, isto é, um*

evento pode ser composto pela combinação lógica de dois eventos por intermédio dos operadores lógicos and e or:

- Se $e \in E$ e $c \in C$ então $e[c] \in E$, isto é, um evento acompanhado de uma condição é um evento;

i) $A = \{a_1, a_2, ..., a_n\}$ é o conjunto de ações e pode ser definido indutivamente como segue:

- $\mu \in A$, é a ação nula;
- Se $c \in C_p$, $d \in C$, então $c:=d \in A$, isto é, a atribuição do valor de uma condição "d" a uma condição primitiva "c" é uma ação;
- *Se $v \in V_p$, $u \in V$, então $v:=u \in A$, isto é, a atribuição do valor de uma expressão "u" a uma variável "v" é uma ação;*
- Se $e \in E_p$, então $e \in A$, isto é, um evento primitivo é uma ação;
- Se $a_i \in A \mid i=0,...,n$, então $a_0;...;a_n \in A$, isto é, uma seqüência de ações separadas por ";" é uma ação;

j) $R \subset E \times A$ é o conjunto de rótulos formado pelos pares ordenados $(e,a) \mid e \in E$ e $a \in A$. Com essa definição pode-se dizer, informalmente, que a forma geral do rótulo de uma transição é "*e*[*c*]/*a*", na qual $e \in E$, $c \in C$ e $a \in A$; e

k) $T \subset 2^S \times R \times 2^{S \cup H}$ é o conjunto de transições do Statechart. Uma transição $t = (X,r,Y)$ é composta por um conjunto de estados origem "X", um conjunto de estados destino ou símbolos de história "Y" e um rótulo $r \in R$.

Para o Statechart da Figura 3.1 o conjunto de estados é S = {A, B, B_1, B_2, C, D, E, E_1, E_{11}, E_{12}, E_2, F, F_1, F_2, G, G_1, G_{11}, G_{11a}, G_{11b}, G_{12}, G_2, G_{21}, G_{22}, I, I_1, I_2, I_3, K}, "A" é o estado raiz e o conjunto de estados atômicos é {B_1, B_2, E_{11}, E_{12}, F_1, F_2, G_{11a}, G_{11b}, G_{12}, G_{21}, G_{22}, I_1, I_2, I_3}. A função de hierarquia para os estados "A" e "E" é, respectivamente, $\rho(A) = \{A,B,C,D,K\}$, $\rho(E) = \{E,E_1,E_2\}$. A função ψ para os mesmos é $\psi(A) = \{OR\}$ e $\psi(E) = \{AND\}$. A função história é $\gamma(H)=C$. A função *default* para o estado "A" e "E" é, respectivamente, $\delta(A) = \{B\}$ e $\delta(E) = \{E_{11}, G, G_{11}, G_{22}\}$. O conjunto de expressões de evento é $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9[in(G_{22})], e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$ e o conjunto de transições é dado por T = {({B},e_1,{C}), ({B,C,K},e_2,{D}), ({B,C,D},e_3,{K}), ({C,D,K},e_4,{B}), ({F},e_5,{I}), ({I},e_6,{F}), ({F},e_7,{E}),

$(\{E\},e_8,\{F\})$, $(\{E_{11}\},e_9[in(G_{22})],\{E_{12}\})$, $(\{E_{12}\},e_{10},\{E_{11}\})$, $(\{G_{11}\},e_{11}/e_{13},\{G_{12}\})$, $(\{G_{12}\},e_{12}/e_{14},\{G_{11}\})$, $(\{G_{22}\},e_{13},\{G_{21}\})$, $(\{G_{21}\},e_{14}/e_{12},\{G_{22}\})$, $(\{D\},e_1,\{C,H\})$, $(\{K\},e_1,\{C,H^*\})$, $(\{K\},e_2,\{D,H^*\})$ }.

3.4.2. Semântica Operacional

Utilizando as características apresentadas na Seção 3.4.1 é possível especificar o comportamento de um Statechart. Essas definições permitem que se tenha uma noção intuitiva do funcionamento da técnica, mas usos mais sofisticados exigem um conhecimento de sua semântica de execução ou operacional. Uma introdução informal à semântica operacional de Statecharts baseada na definição de Harel [1987] é apresentada sucintamente nesta seção.

A semântica operacional dos Statecharts é baseada em um modelo discreto de tempo, segundo o qual os estados de um sistema são duráveis, enquanto que as transições são instantâneas. O termo "passo de execução" ou "passo de simulação" tem origem nesse modelo de tempo. Um passo é composto de vários micro-passos, dentro dos quais somente uma transição é avaliada permitindo, assim, a evolução das várias máquinas de estado concorrentemente ativas. Em cada passo de execução o Statechart assume uma configuração legal de estados ou, resumidamente, configuração de estados (SC - *Statechart Configuration*), a qual corresponde ao conjunto dos estados básicos ativos naquele passo. Dessa forma, a cada passo uma nova configuração é atingida.

Define-se o menor ancestral comum (*LCA- Lowest Common Ancestor*) de um conjunto de estados "X", denotado por LCA(X), como um estado $y \in S \mid X \subseteq \rho^*(y)$ (isto é, "y" é um ancestral comum) e $\forall s \in \rho^+(y) \rightarrow X \not\subset \rho^*(s)$. Dois estados distintos "x" e "y" são ortogonais, denotados por $x \perp y$, se $x = y$ (isto é, um estado "x" é considerado ortogonal a si mesmo) ou se $\psi(LCA(\{x,y\})) = AND$, isto é, se o menor ancestral comum a "x" e "y" for do tipo AND. Um conjunto de estados é ortogonal se seus estados são ortogonais dois a dois. Um conjunto ortogonal maximal é um conjunto ortogonal que não pode ser estendido (adicionando estados) para um conjunto ortogonal maior. Dessa forma, uma configuração de estados é um conjunto ortogonal maximal de estados básicos nos quais o sistema especificado pode estar simultaneamente, representando um conjunto de estados ativos do Statechart.

A configuração de estados inicial de Statecharts é obtida pela ativação de seu estado raiz e, recursivamente, de todos os subestados *default*. A configuração inicial para o Statechart da Figura 3.1 é $SC_0=\{B_1, B_2\}$, ou seja, um subconjunto de estados atômicos formado a partir dos estados ativos {A, B, B_1, B_2}. Outras configurações possíveis são $SC_1=\{E_{11}, G_{11a}, G_{11b}, G_{22}\}$, $SC_2=\{F_1, F_2\}$ e $SC_3=\{E_{11}, G_{12}, G_{21}\}$. Na Figura 3.9 apresenta-se a configuração de estados SC_1 durante a execução do Statechart da Figura 3.1. A ocorrência do evento "e_{11}" dispara a transição "($\{G_{11}\}$, e_{11}/e_{13}, $\{G_{12}\}$)", a ação associada, que gera o evento "e_{13}" é executada e, conseqüentemente, pelo mecanismo de propagação de eventos a transição "($\{G_{22}\}$, e_{13}, $\{G_{21}\}$)" também é disparada no mesmo passo, levando o sistema à configuração SC_3.

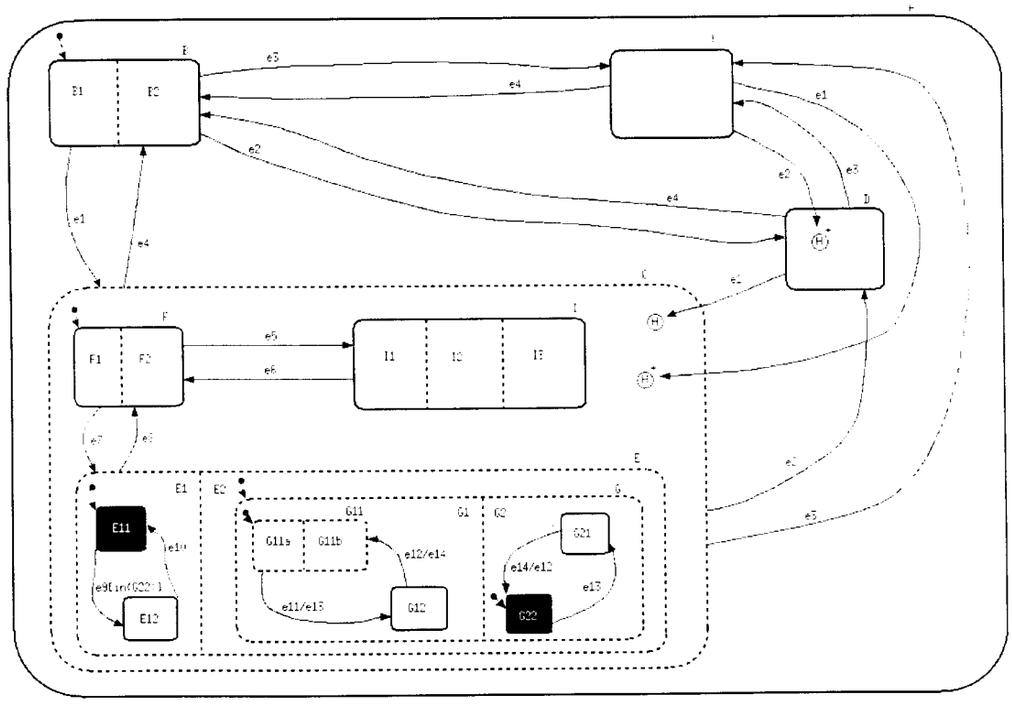


Figura 3.9: Possível configuração de estados para o Statechart da Figura 3.1.

Harel [1988] apresenta uma discussão detalhada de algumas questões que a semântica formal de Statecharts deve resolver. Huizing e Roever [1991] comentam sutilezas relacionadas à semântica de Statecharts, por exemplo, a necessidade de uma reação em cadeia não consumir tempo (conceitualmente) e a definição dos micro-passos. Para complementar a especificação da semântica operacional

definida por Harel ou adequá-la a determinados tipos de aplicações, novas propostas surgiram [Huizing et al. 1988, Kensten; Pnuelli 1992, Hooman et al. 1992, Harel; Naamad 1996].

3.5. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado, sucintamente, os conceitos e as características gerais da técnica Statecharts, sem a intenção de ser uma introdução completa. Certamente, outros recursos de notação e outras formas de utilização das características de Statecharts aqui apresentadas poderiam ser exploradas no contexto da modelagem de hiperdocumentos. Entretanto, foi registrado o essencial para a compreensão do modelo HMBS.

Os próximos capítulos apresentam a definição do modelo HMBS, suas principais características e um protótipo de sistema hiperdocumento desenvolvido para validar o modelo proposto.

Capítulo 4

HMBS: UM MODELO BASEADO EM STATECHARTS

4.1. CONSIDERAÇÕES INICIAIS

A proposta de tese do presente trabalho consiste na definição de um modelo para a especificação da estrutura e da semântica navegacional de hiperdocumentos usando a técnica Statecharts como modelo de especificação formal subjacente. O modelo proposto é denominado ***H**yperdocument **M**odel **B**ased on **S**tatecharts*, ou HMBS. De acordo com a classificação de modelos para hiperdocumentos proposta no Capítulo 2, o HMBS pode ser incluído na classe de modelos comportamentais, devido à sua capacidade e flexibilidade em modelar a estrutura organizacional e a semântica de navegação associada a hiperdocumentos. Como outros modelos, por exemplo, o Trellis [Stotts; Furuta 1989], o HMBS permite separar as informações referentes à estrutura das representações físicas ou conteúdo do hiperdocumento. Aspectos selecionados do modelo HMBS podem ser encontrados em várias publicações [Oliveira et al. 1995, Oliveira et al. 1996, Turine et al. 1996, Turine et al. 1997a, Turine et al. 1997b, Turine et al. 1998, Paulo et al. 1998, Oliveira et al. 1998].

Para ilustrar as características e potencialidades do modelo utiliza-se neste capítulo a estrutura do hiperdocumento ilustrado na Figura 3.1. O capítulo está organizado da seguinte forma: Na Seção 4.2 apresenta-se a definição formal do modelo HMBS, com ênfase na formalização do hiperdocumento como um conjunto de objetos estruturais, navegacionais e de apresentação. Na Seção 4.3 as características do modelo são explicitadas informalmente e, para facilitar o entendimento dessas características, um hiperdocumento é instanciado com os objetos do modelo. Na Seção 4.4 é definida a semântica de navegação associada ao modelo e também são apresentados exemplos que ilustram o seu uso.

Com o objetivo de explorar a hierarquia do hiperdocumento como um recurso capaz de minimizar a desorientação, que é uma desvantagem potencial associada a hiperdocumentos, propõe-se o conceito de visões hierárquicas na Seção 4.5. Para efeito de comparação com outros modelos afins, algumas características e propriedades satisfeitas pelo HMBS são apresentadas na Seção 4.6 e as conclusões deste capítulo são apresentadas na Seção 4.7.

4.2. ESPECIFICAÇÃO FORMAL DA SINTAXE

Um hiperdocumento é definido no HMBS segundo uma organização hierárquica dos estados do Statechart que representa a sua estrutura organizacional e navegacional, os quais são associados a porções de informação ou páginas. Os eventos contidos nos rótulos das transições representam, respectivamente, as âncoras que disparam as possíveis ligações entre as páginas, definindo, assim, os caminhos de navegação disponíveis para o leitor do hiperdocumento.

Segundo o modelo HMBS, um hiperdocumento H é definido como uma 7-tupla

$$H = <ST, P, m, L, pl, ae, N>, \text{ na qual:}$$

a) ST representa o Statechart que especifica a estrutura organizacional e navegacional subjacente ao hiperdocumento. Como apresentado no Capítulo 3 (Seção 3.4.1), utiliza-se um subconjunto da sintaxe da técnica Statecharts descrita por Harel et al. [1987]. O Statechart é definido como uma 11-tupla, ST = <S, ρ, ψ, γ, δ, V, C, E, A, R, T>, representando, respectivamente, estados, função de hierarquia, função tipo de

decomposição, função história, função *default*, conjunto de expressões, conjunto de condições, conjunto de eventos, conjunto de ações, conjunto de rótulos e conjunto de transições. As simplificações introduzidas na sintaxe de Statecharts foram destacadas na Seção 3.4.1. É interessante ressaltar algumas simplificações importantes: um símbolo de história não pode ser *default* de um estado, o evento *time_out*[12] não é permitido, e os eventos não podem ser formados por uma combinação lógica de outros eventos, ou seja, se $f \in E$ e $g \in E$ então (f *or* g) $\notin E$ e (f *and* g) $\notin E$;

b) $P = \{p_1, p_2, ..., p_n\}$, $n > 0$, é o conjunto finito de páginas de informação que define o conteúdo do hiperdocumento. Cada página $p \in P$, comumente denominada nó de informação, contém uma porção de informação relevante em um determinado contexto do hiperdocumento e é definida conceitualmente pela tripla <c, t, Anc_p>, tal que "c" é a porção do conteúdo de informação, que pode ser composta por mídias estáticas ou independentes de tempo (texto, gráfico ou imagem, por exemplo), ou por mídias dinâmicas ou dependentes de tempo (vídeo, áudio ou animação); "t" representa o título associado à página, o qual deve identificar o conceito que melhor descreve a informação contida na página a fim de auxiliar na busca por uma informação específica durante o processo de navegação; $Anc_p = \{a_1, a_2, ..., a_m\}$, $m > 0$ define uma coleção de âncoras associadas à página. O conjunto de páginas pode incluir uma página nula especial ($P\lambda$), sem qualquer conteúdo, título e/ou âncora, a qual pode ser associada a estados que não modelam apresentação de informação;

c) m: $S_S \rightarrow P$ denota uma função que mapeia estados "s" de um subconjunto S_S ($S_S \subset S$) a páginas $p \in P$ do hiperdocumento. O subconjunto "S_S" é definido por:

S_S: $\{x \in S \mid \psi(x) = OR \vee \rho(x) = \phi\}$, isto é, "$S_S$" é o conjunto formado pelos estados compostos do tipo OR e pelos estados atômicos do Statechart. Estados do tipo AND não são mapeados para páginas, pois são utilizados unicamente para especificar concorrência de informações na apresentação;

---

[12] *time_out(e,m)*: determina a ocorrência de um evento "e" no instante em que "m" passos transcorreram desde o momento em que o estado origem da transição foi ativado.

d) $L = \{l_1, l_2, ..., l_k\}$, $k > 0$, define o conjunto de canais de apresentação (ou leitores), que são abstrações de dispositivos pelos quais é possível especificar os requisitos relativos à apresentação da informação contida nas páginas do hiperdocumento. Os canais são invocados para visualizar as informações contidas nas páginas durante a navegação, atuando como interpretadores da página. Dependendo do tipo da mídia, os canais invocam um aplicativo para visualizar as informações. Por exemplo, editores de texto podem ser utilizados como dispositivos para visualizar informações textuais, editores gráficos como dispositivos para imagens estáticas e manipuladores de áudio e vídeo como dispositivos para animação.

Os canais permitem ao autor especificar a coordenação espacial das informações pertencentes às páginas, além de controlar atributos globais da apresentação do hiperdocumento. As características de apresentação disponíveis em cada canal dependem dos recursos disponíveis no sistema hiperdocumento a ser utilizado. O uso de canais é inspirado nos modelos AHM [Hardman et al. 1994] e MCA [Casanova et al. 1991], mas o conceito adotado no HMBS é mais simples, pois o hiperdocumento não é definido de forma distribuída e, conseqüentemente, não requer a especificação de atributos relativos à comunicação;

e) $pl: P \rightarrow L$ define a função de visualização que associa cada página $p \in P$ a um único canal $l \in L$ capaz de interpretá-la. Deste modo, os canais são aplicados no contexto de uma página;

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

Seja $t \in T$ uma transição do Statechart na forma "t = (X,r,Y)", $rot: T \times E \rightarrow \{true, false\}$ uma função que verifica se um evento "e_r" 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 $rot(t_m,e_n) = false$ caso contrário. Assim, a função "ae" pode ser definida como:

$$ae(a_p) = e \leftrightarrow (\exists\ p \in P \mid p = <c,t,Anc_p> \wedge a_p \in Anc_p) \wedge e \in E \wedge$$
$$(\exists\ t \in T \mid rot(t,e) = true) \wedge (\exists\ s \in S \mid m(s) = p) \wedge (\exists\ s_1 \in S \mid s \in \rho^{*}(s_1))$$

Segundo tais restrições, para que as âncoras pertencentes às páginas possam ser associadas a um evento de uma transição, é necessário que tal evento seja único no rótulo da transição. Além disso, a página que contém a âncora deve ser mapeada para um estado pertencente ao conjunto de estados origem da transição especificada ou, ainda, pode ser mapeada para um subestado de um outro estado pertencente a esse conjunto. Assim, como Dexter e AHM, o HMBS permite a reutilização de componentes de ligação, de forma que as transições com seus respectivos eventos definidos na estrutura do hiperdocumento podem ser reutilizadas para definir âncoras em diferentes contextos de navegação; e

g) N ($N \geq 0$) define o nível de visibilidade ou de navegação do hiperdocumento. O autor do hiperdocumento pode utilizar o valor de "N" para definir a profundidade da hierarquia na estrutura especificada pelo Statechart subjacente a ser apresentada durante a navegação. Um hiperdocumento com valor de "N" igual a zero ($N = 0$) tem exibidas, durante a navegação, todas as páginas associadas aos estados atômicos que estiverem ativos na configuração válida de estados. Com "N" igual a um ($N = 1$), as páginas exibidas são as definidas por $N = 0$, além das associadas aos estados ancestrais (do tipo OR) imediatos da configuração de estados atual, e assim por diante para $N > 1$. Usando a hierarquia de estados do Statechart, o HMBS permite explorar a hierarquia do hiperdocumento. Trata-se de um novo mecanismo de navegação, tal que ao invés de ativar as âncoras, o leitor pode navegar pelas páginas associadas a estados acima ou abaixo do nível de hierarquia definido pelo autor, contribuindo para minimizar a sua eventual desorientação.

4.3. ESPECIFICAÇÃO INFORMAL

O HMBS possui na sua definição três tipos de objetos: estruturais,

navegacionais e de apresentação. A estrutura organizacional do hiperdocumento definida pelo Statechart subjacente permite especificar os objetos estruturais, que são os estados, as transições e os eventos do Statechart. Os objetos navegacionais definem a estrutura navegacional e são as páginas, as ligações (ainda que não explicitamente formalizadas no modelo) e as âncoras. Os canais definem os objetos relativos à apresentação do hiperdocumento.

Na Figura 4.1 é apresentado um esquema de mapeamentos "m" e "pl" do modelo HMBS em que três objetos navegacionais do tipo página são definidos e associados a estados do Statechart subjacente (função "m"). Duas das páginas armazenam informações do tipo texto e são associadas a um mesmo canal (função "pl"), a outra página armazena uma imagem e é associada a um canal que a reconhece.

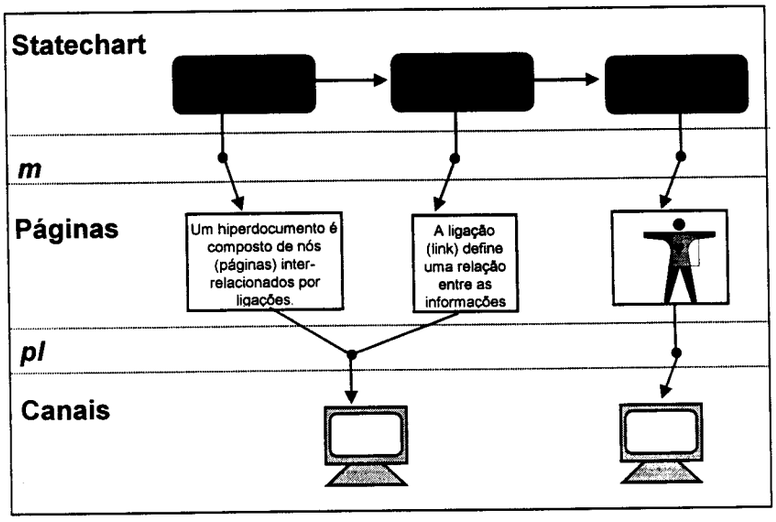


Figura 4.1: Mapeamentos "m" e "pl" combinados.

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

O exemplo da Figura 4.1 enfatiza a possibilidade em reutilizar canais para

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

Para exemplificar as características do modelo, é apresentada na Figura 4.2 a estrutura organizacional de um hiperdocumento. Tal Figura é idêntica à Figura 3.1 do Capítulo 3, tendo sido replicada apenas para facilitar a leitura. O autor define os objetos estruturais com o Statechart e na Tabela 4.1 são apresentadas as definições dos objetos navegacionais e de apresentação. Nesse exemplo, o conjunto de canais é definido por $L = \{L_T, L_I, L_A, L_V\}$, cujos elementos definem, respectivamente, um canal do tipo texto, imagem, áudio e vídeo; e o nível de visibilidade especificado é igual a zero (N = 0). A definição das informações $<c, t, Anc_p>$ de cada página, o mapeamento de estados em páginas ("m") e o mapeamento de páginas em canais ("pl") são ilustrados na Tabela 4.1. A Figura 4.3 ilustra a associação de um objeto navegacional âncora, inserido em uma página do tipo texto, ao evento "e" contido no rótulo de uma transição do Statechart (função "ae"). Tal abordagem envolve duas tarefas distintas na atividade de autoria: a especificação das transições (ligações estruturais) e a definição das âncoras que são parte do conteúdo das páginas e especificam os pontos de ancoragem. A Tabela 4.2 ilustra, para o conjunto de páginas do hiperdocumento definido na Tabela 4.1, o mapeamento "ae" de todas as âncoras especificadas em cada página.

Tabela 4.1: Informações (c,t,Anc_p) das páginas e mapeamento "m" e "pl".

S	P	C	t	Anc_p	L
A	P_A	C_A	T_A	Ø	L_T
B_1	P_{B1}	C_{B1}	T_{B1}	$\{ae_1,ae_2,ae_3\}$	L_T
B_2	P_{B2}	C_{B2}	T_{B2}	Ø	L_V
C	P_C	C_C	T_C	$\{ae_4\}$	L_T
D	P_D	C_D	T_D	$\{ae_1,ae_3,ae_4\}$	L_T
E_1	P_{E1}	C_{E1}	T_{E1}	Ø	L_T
E_2	P_{E2}	C_{E2}	T_{E2}	Ø	L_T
F_1	P_{F1}	C_{F1}	T_{F1}	$\{ae_2,ae_3,ae_4,ae_5,ae_7\}$	L_T

S	P	C	t	Anc_p	L
F_2	P_{F2}	C_{F2}	T_{F2}	Ø	L_I
I_1	P_{I1}	C_{I1}	T_{I1}	$\{ae_2,ae_3,ae_4,ae_6\}$	L_T
I_2	P_{I2}	C_{I2}	T_{I2}	Ø	L_I
I_3	P_{I3}	C_{I3}	T_{I3}	Ø	L_A
E_{11}	P_{E11}	C_{E11}	T_{E11}	$\{ae_2,ae_3,ae_4,ae_8,ae_9\}$	L_T
E_{12}	P_{E12}	C_{E12}	T_{E12}	$\{ae_2,ae_3,ae_4,ae_8,ae_{10}\}$	L_T
G_1	Ø	Ø	Ø	Ø	Ø
G_2	Ø	Ø	Ø	Ø	Ø
G_{11a}	P_{G11a}	C_{G11a}	T_{G11a}	$\{ae_2,ae_3,ae_4,ae_8,ae_{11}\}$	L_T
G_{11b}	P_{G11b}	C_{G11b}	T_{G11b}	$\{ae_{11}\}$	L_I
G_{12}	P_{G12}	C_{G12}	T_{G12}	$\{ae_2,ae_3,ae_4,ae_8,ae_{12}\}$	L_T
G_{21}	P_{G21}	C_{G21}	T_{G21}	$\{ae_2,ae_3,ae_4,ae_8,ae_{14}\}$	L_T
G_{22}	P_{G22}	C_{G22}	T_{G22}	$\{ae_{13}\}$	L_V
K	P_K	C_K	T_K	$\{ae_1,ae_2,ae_4\}$	L_T

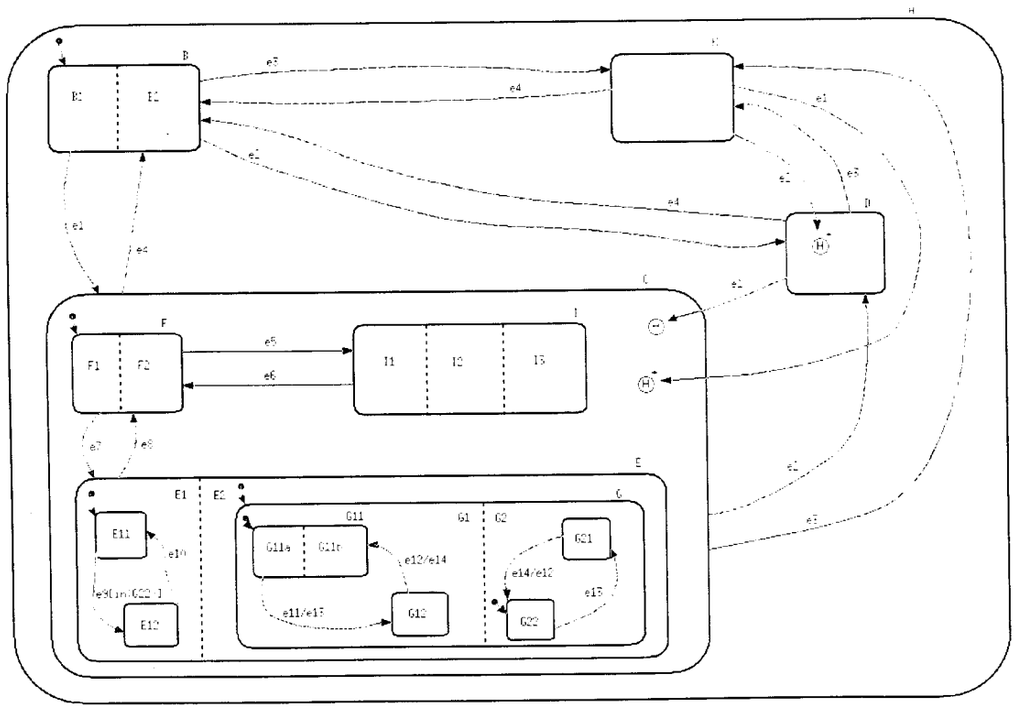


Figura 4.2: Estrutura de um hiperdocumento.

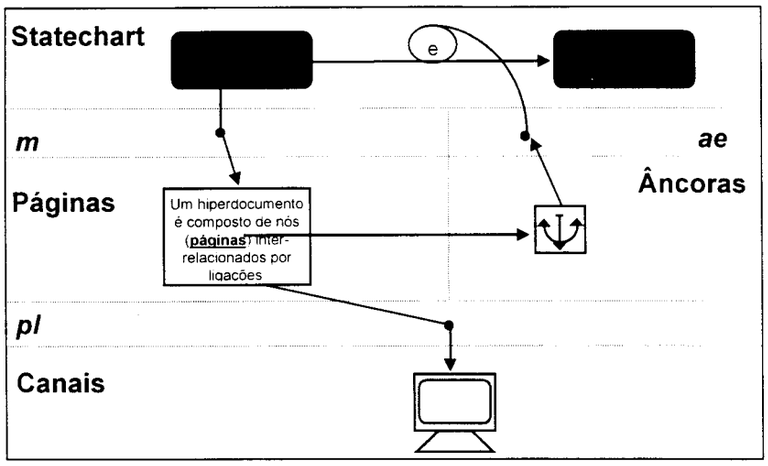


Figura 4.3: Mapeamento de âncoras.

Tabela 4.2. Mapeamento "ae".

P	Anc_p	
	$ae_i \rightarrow e_i$	
P_{B1}	ae_1	e_1
P_{B1}	ae_2	e_2
P_{B1}	ae_3	e_3
P_C	ae_4	e_4
P_D	ae_1	e_1
P_D	ae_3	e_3
P_D	ae_4	e_4
P_{F1}	ae_2	e_2
P_{F1}	ae_3	e_3
P_{F1}	ae_4	e_4
P_{F1}	ae_5	e_5
P_{F1}	ae_7	e_7
P_{I1}	ae_2	e_2
P_{I1}	ae_3	e_3
P_{I1}	ae_4	e_4
P_{I1}	ae_6	e_6
P_{E11}	ae_2	e_2
P_{E11}	ae_3	e_3
P_{E11}	ae_4	e_4

P	Anc_p	
	$ae_i \rightarrow e_i$	
P_{E11}	ae_8	e_8
P_{E11}	ae_9	e_9
P_{E12}	ae_2	e_2
P_{E12}	ae_3	e_3
P_{E12}	ae_4	e_4
P_{E12}	ae_8	e_8
P_{E12}	ae_{10}	e_{10}
P_{G11a}	ae_2	e_2
P_{G11a}	ae_3	e_3
P_{G11a}	ae_4	e_4
P_{G11a}	ae_8	e_8
P_{G11a}	ae_{11}	e_{11}
P_{G11b}	ae_{11}	e_{11}
P_{G12}	ae_2	e_2
P_{G12}	ae_3	e_3
P_{G12}	ae_4	e_4
P_{G12}	ae_8	e_8
P_{G12}	ae_{12}	e_{12}
P_{G21}	ae_2	e_2

P	Anc_p	
	$ae_i \rightarrow e_i$	
P_{G21}	ae_3	e_3
P_{G21}	ae_4	e_4
P_{G21}	ae_8	e_8
P_{G21}	ae_{14}	e_{14}

P	Anc_p	
	$ae_i \rightarrow e_i$	
P_{G22}	ae_{13}	e_{13}
P_K	ae_1	e_1
P_K	ae_2	e_2
P_K	ae_4	e_4

4.4. SEMÂNTICA DE NAVEGAÇÃO

Nesta seção apresentam-se a definição da semântica de navegação associada ao modelo HMBS e alguns exemplos que ilustram sua flexibilidade para especificar as transformações navegacionais que podem ser necessárias para manipular hiperdocumentos.

4.4.1. Objetos Navegacionais

A estrutura navegacional do hiperdocumento é especificada pelo Statechart subjacente, que representa o documento em termos de objetos estruturais (estados, transições e eventos), pelos componentes navegacionais (páginas, âncoras e ligações), além dos mecanismos de apresentação da informação (canais). 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 alguma configuração de estados.

A apresentação do hiperdocumento compreende um conjunto de páginas interpretadas pelos canais e apresentadas em janelas lógicas. Características físicas relacionadas à implementação das janelas lógicas ou à visualização das âncoras dependem do sistema hiperdocumento. Nesta seção discute-se como mapear os objetos estruturais em objetos navegacionais e como a semântica de navegação pode ser interpretada.

Páginas

Como apresentado anteriormente, um objeto estrutural "estado" do Statechart é mapeado, segundo a função "m", para um único objeto navegacional do tipo página. Uma página é especificada pelos atributos título, conteúdo e âncoras. Um

título é uma palavra-chave (ou expressão) que identifica a página de maneira única. O conteúdo refere-se à porção de informação contida na página, que pode ser do tipo texto, imagem, áudio ou vídeo, por exemplo. A definição do atributo âncora será apresentada posteriormente nesta seção.

Pode-se estabelecer uma analogia entre páginas e janelas lógicas, pois durante a navegação cada página é convertida em uma janela lógica e, dependendo do sistema hiperdocumento e do canal associado, os atributos da página podem ser visualizados de diferentes maneiras na janela lógica, podendo apresentar o título e o conteúdo da página, ou somente o conteúdo.

Ligações

O conjunto de possíveis ligações de ativação do hiperdocumento (comumente conhecidas como *links*, ou elos) é representado pelo conjunto de transições (T) do Statechart. A relação entre transições e ligações é que os objetos estruturais "transição" representam os relacionamentos estruturais, ou seja, o inter-relacionamento de estados na estrutura organizacional do hiperdocumento, enquanto as ligações relacionam os objetos "página" durante a navegação e são definidas a partir da criação das âncoras.

O objeto navegacional "ligação" não é explicitamente definido no HMBS, mas surge quando da criação das âncoras. A origem de uma ligação é uma âncora relacionada a uma página, a qual, por sua vez, deve estar associada a um estado. 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. A Figura 4.4 ilustra o objeto navegacional "ligação" no nível de apresentação, o qual foi habilitado a partir da criação da âncora rotulada "páginas".

As ligações são armazenadas separadamente das páginas que referenciam, ou seja, não são inseridas no conteúdo das páginas, como acontece, por exemplo, com os documentos que usam a linguagem HTML. Para cada transição existente no Statechart podem existir uma ou mais ligações, dependendo do número de âncoras criadas e relacionadas com os eventos contidos nas transições. Por exemplo, para a transição com rótulo "e_4" da Figura 4.2 (no contexto do estado "C" para o estado "B") existem oito ligações relacionadas. Tais ligações são definidas

pela criação da âncora "ae_4" pertencente ao conjunto de âncoras (Anc_p) das páginas "C", "F_1", "I_1", "E_{11}", "E_{12}", "G_{11a}", "G_{12}" e "G_{21}" (Tabela 4.2).

O HMBS permite somente especificar ligações unidirecionais, oriundas das transições unidirecionais do Statechart. Duas transições unidirecionais podem ser criadas para cada ligação bidirecional, dobrando o número de transições na especificação. As ligações podem ser de cardinalidade um para um, muitos para um e um para muitos, ou seja, podem ter uma ou mais páginas origem e uma ou mais páginas destino. O comportamento ou a transformação navegacional associada à ativação de uma ligação é definido pela semântica de navegação do modelo.

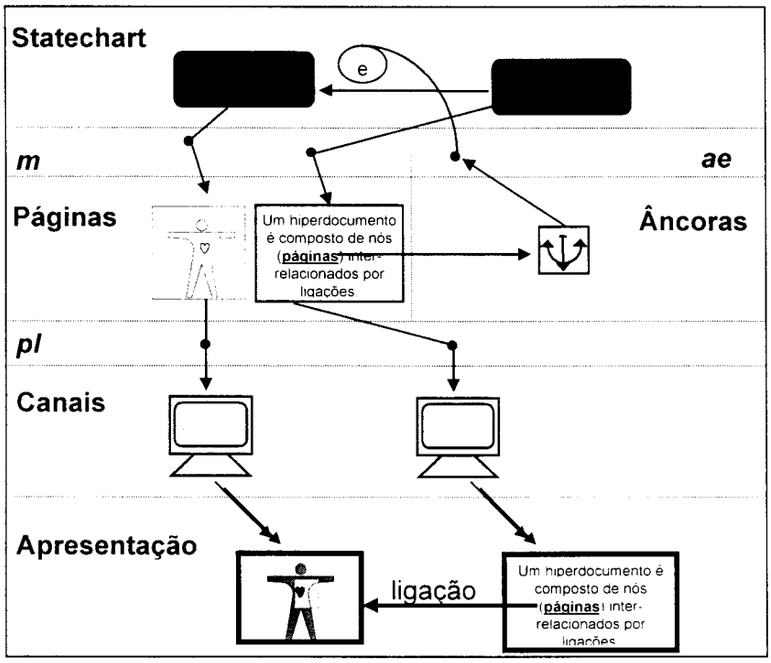
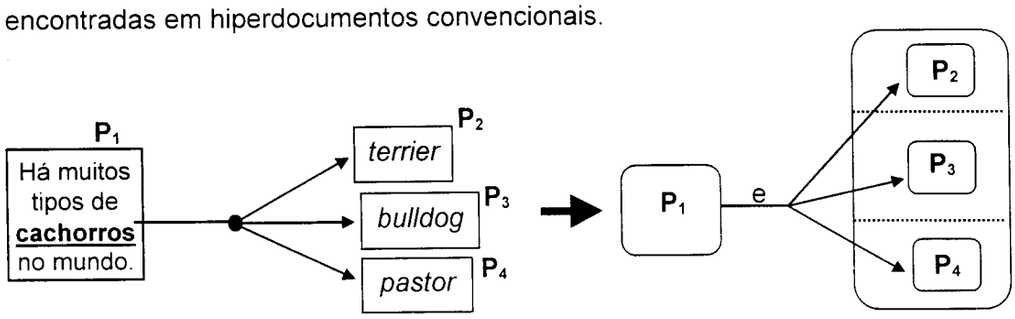


Figura 4.4: Exemplo de especificação de uma ligação.

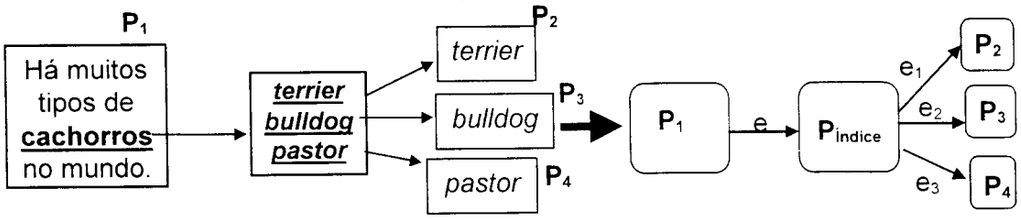
Ligações múltiplas, ou seja, ligações com múltiplas páginas destino podem ser definidas de duas maneiras. Na primeira pode-se obter tais ligações especificando transições M:N de Statecharts, conforme apresentado na Figura 4.5 (a). Nessa situação, os conteúdos de todas as páginas destino são apresentados simultaneamente. A segunda maneira consiste em definir uma ligação unidirecional para uma página intermediária do tipo índice, que contém ligações para as diferentes páginas destino, cujos conteúdos são apresentados exclusivamente, como na Figura 4.5 (b).

Visto que o HMBS não especifica notação temporal, não é possível definir

ligações temporais, ou seja, ligações ativadas automaticamente (sem a intervenção do leitor). As únicas ligações permitidas no modelo são estáticas, como as encontradas em hiperdocumentos convencionais.



(a) Ligações multidestinos.



(b) Ligações com página intermediária.

Figura 4.5: Implementação de ligações múltiplas.

Âncoras

O objeto navegacional "âncora" é contextualizado por uma página e indica os pontos de ancoragem no hiperdocumento. Representa a origem de uma ligação e a sua ativação faz com que todos os eventos associados a ele (via função "ae") sejam disparados de acordo com a semântica operacional do Statechart. Como em OOHDM [Schwabe et al.1996], as âncoras são tratadas como gatilhos navegacionais, isto é, são palavras-chave que, quando selecionadas, ativam as ligações correspondentes do hiperdocumento e uma navegação ocorre. Esta interpretação sustenta a visão tradicional pela qual as âncoras são apresentadas como objetos de interface e definem a origem de ligações.

O HMBS não restringe o mapeamento de âncoras (função "ae") somente a eventos cujas transições têm como origem 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. Por exemplo, pode-se definir no conteúdo de uma mesma página três âncoras em regiões diferentes associadas a um mesmo evento. Essa flexibilidade também permite ao autor especificar âncoras associadas a

eventos de uma transição cujo estado origem seja ancestral do estado associado à página que contém a âncora. Por exemplo, analisando o Statechart da Figura 4.2 e a Tabela 4.2 que ilustra o mapeamento "ae", nota-se que a página associada ao estado "F_1" (P_{F1}) inclui em sua definição um conjunto formado por cinco âncoras {ae_2, ae_3, ae_4, ae_5, ae_7}, ainda que na estrutura navegacional existam somente duas transições, ({F},e_5,{I}) e ({F},e_7,{E}), com origem no estado "F". Pode-se associar a âncora rotulada "ae_2" da página "P_{F1}" ao evento "e_2", pois "C" é um estado ancestral de "F". Tal abordagem ilustra a possibilidade de reutilizar as transições para criar novos caminhos de navegação que exploram a hierarquia definida pelo Statechart subjacente ao hiperdocumento.

Fisicamente, dependendo da plataforma escolhida para "executar" o hiperdocumento, as âncoras podem ser representadas por botões, menus ou palavras-chave (*hotkeys*) no interior do documento. Desse modo, a operação de ativar uma âncora resulta no disparo de um evento no modelo Statechart subjacente que, conseqüentemente, ativa todas as transições possivelmente habilitadas, modificando o conjunto de janelas lógicas apresentado.

4.4.2. Interpretação da Semântica

A noção mais elementar da semântica de navegação do HMBS é a de visitar uma página, de modo que a apresentação das informações contidas no conteúdo das páginas do hiperdocumento é determinada unicamente pela decisão do leitor em seguir uma ligação pela ativação de uma âncora. Assim, o conceito convencional de navegação em hiperdocumentos é satisfeito.

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. O conjunto de páginas exibidas durante um determinado momento da navegação define uma "configuração de contexto do hiperdocumento" (CC). Um conceito similar à configuração de contexto é utilizado por Stotts e Furuta [1989] quando se referem a um hipertexto marcado (*marked hypertext*).

A configuração de contexto corrente (CC) é formada pelas páginas associadas aos estados da configuração de estados corrente do Statechart, além das páginas que satisfazem o nível de visibilidade "N" especificado pelo autor. Para N igual a zero (N = 0), a configuração de contexto inicial (CC_0) é definida pela

configuração de estados *default* do Statechart. Formalmente, a definição da configuração de contexto é descrita abaixo:

Se $N = 0$, então

Todas as páginas *p* do conjunto CC_0 são visualizadas. tal que

$$CC_0 = \{p \mid x \in S_S \wedge m(x) = p \wedge \rho^0(x) \cap SC \neq \phi\}$$

Se $N = 1$, então

Todas as páginas *p* do conjunto $CC_0 \cup CC_1$ são visualizadas, tal que

$$CC_1 = \{p \mid x \in S_S \wedge m(x) = p \wedge \rho^1(x) \cap SC \neq \phi\}$$

Se $N = 2$, então

Todas as páginas *p* do conjunto $CC_0 \cup CC_1 \cup CC_2$ são visualizadas, tal que

$$CC_2 = \{p \mid x \in S_S \wedge m(x) = p \wedge \rho(\rho(x)) = \rho^2(x) \cap SC \neq \phi\}.$$

Generalizando,

Se $N = n$, então

Todas as páginas *p* do conjunto CC são visualizadas. tal que

$$CC = \bigcup_{i=0,\ldots,n} CC_i \text{ , tal que } CC_i = \{p \mid x \in S_S \wedge m(x) = p \wedge \rho^i(x) \cap SC \neq \phi\}$$

Baseando-se na semântica operacional dos Statecharts, a semântica de navegação do hiperdocumento definida pelo HMBS é interpretada como segue. Quando o leitor seleciona uma âncora, o sistema hiperdocumento realiza as seguintes ações:

a) Gera o evento correspondente à âncora (informação obtida pelo mapeamento "ae"), disparando a ligação correspondente à transição do Statechart à qual o evento está associado;

b) Ativa todos os estados que fazem parte do conjunto destino da transição disparada, gerando a próxima configuração de estados e desabilitando a configuração anterior. Em seguida remove da apresentação as janelas lógicas associadas às páginas correspondentes aos estados da configuração e seus estados ancestrais, segundo o nível "N"; e

c) Define a nova configuração de contexto do hiperdocumento como sendo formada pelas páginas associadas aos estados da nova configuração de estados, além das páginas que obedecem ao escopo estabelecido pelo nível de visibilidade "N" definido pelo autor. Invoca, em seguida, os respectivos canais de apresentação (função "pl") para cada página da

configuração de contexto.

4.4.3. Exemplos

O Statechart da Figura 4.2 é utilizado nesta seção para exemplificar a semântica de navegação do HMBS. Assume-se que a configuração de contexto inicial $CC_0 = \{P_{B1}, P_{B2}\}$ é determinada pela configuração de estados inicial do Statechart apresentada na Figura 4.6 e que os mapeamentos e definições do HMBS para esse hiperdocumento são os representados na Seção 4.3. A seguir, são apresentadas algumas configurações de contexto que podem ser alcançadas durante a navegação, bem como o leiaute das janelas lógicas apresentado por um sistema hiperdocumento hipotético, o qual implementa a semântica de navegação definida pelo HMBS.

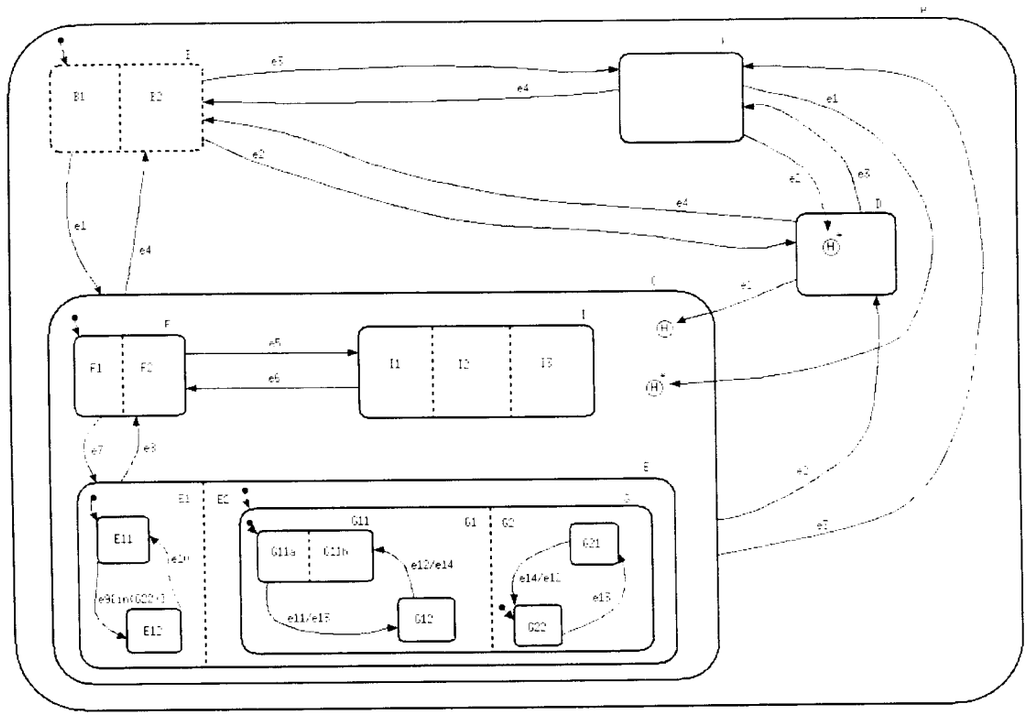
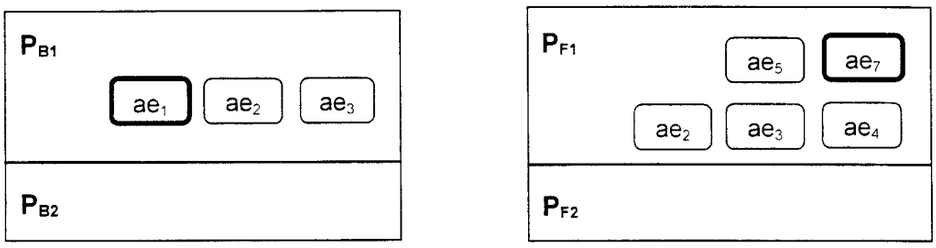


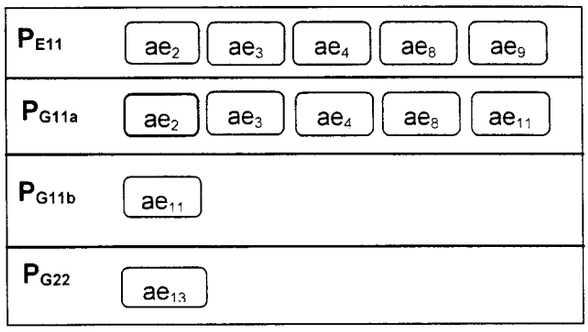
Figura 4.6: Configuração (SC_0={B_1, B_2}) para o Statechart da Figura 4.2.

Considerando o nível de visibilidade N = 0, um leiaute esquemático apresentado ao leitor no estágio inicial de navegação é apresentado na Figura 4.7 (a). O sistema apresenta o conteúdo das páginas "P_{B1}" e "P_{B2}" associadas aos estados "B_1" e "B_2", bem como o conjunto de âncoras (Anc_p) especificadas em cada uma das duas páginas (ver Tabela 4.1). Nesse caso, somente a página "P_{B1}" tem âncoras associadas, representadas por "ae_1", "ae_2" e "ae_3". A ação de clicar na

âncora rotulada "ae_1" na página "P_{B1}" ativa o evento "e_1" contido no rótulo da transição ({B},e_1,{C}) (Tabela 4.2), conduzindo à configuração de estados $SC_1 = \{F_1,F_2\}$ e sua correspondente configuração de contexto $CC_1 = \{P_{F_1},P_{F_2}\}$. O novo leiaute das janelas lógicas é ilustrado na Figura 4.7 (b). A partir desta configuração de contexto, se o leitor clicar na âncora rotulada "ae_7", o Statechart subjacente assume a configuração de estados $SC_2 = \{E_{11}, G_{11a}, G_{11b}, G_{22}\}$ e o novo leiaute das janelas lógicas é ilustrado na Figura 4.7 (c). Esses exemplos de navegação ilustraram que considerando N = 0, o sistema apresenta somente as páginas associadas aos estados da configuração de estados do Statechart.



(a) $CC_0 = \{P_{B1}, P_{B2}\}$. (b) $CC_1 = \{P_{F1}, P_{F2}\}$.

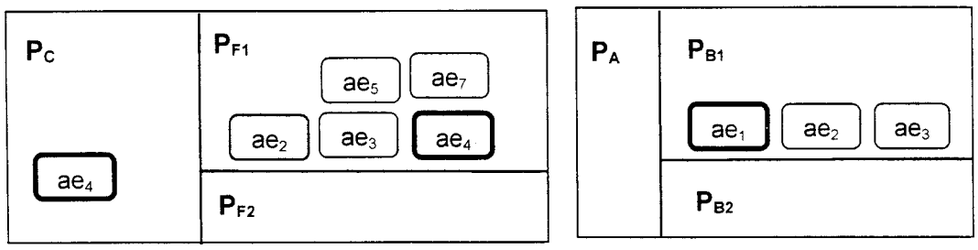


(c) $CC_2 = \{P_{E11}, P_{G11a}, P_{G11b}, P_{G22}\}$.

Figura 4.7: Possíveis leiautes exibidos durante a navegação do hiperdocumento para as configurações dadas e N = 0.

Segundo o HMBS, as páginas associadas a estados não pertencentes a uma configuração de estados do Statechart podem fazer parte de uma configuração de contexto do hiperdocumento. Explorando o nível de visibilidade do hiperdocumento o leitor pode visualizar, em conjunção com as páginas associadas a estados da configuração de estados corrente, as páginas associadas a seus estados ancestrais. Tal facilidade permite aos leitores visualizarem simultaneamente páginas em diferentes níveis da hierarquia do hiperdocumento. Por exemplo, iniciando o nível de visibilidade N = 1 é possível mostrar as páginas associadas aos

estados atômicos e também as páginas associadas a seus estados ancestrais imediatos que estão no conjunto "S_S" (Seção 4.2). Assim, se N = 1 e a configuração de estados corrente é dada por $SC_1 = \{F_1, F_2\}$ (configuração exibida na Figura 4.7 (b)), um leiaute hipotético para as janelas lógicas da configuração de contexto CC = $\{P_C, P_{F1}, P_{F2}\}$ é mostrado na Figura 4.8 (a). A página "P_C" associada ao estado ancestral imediato é exibida, bem como suas âncoras associadas ("ae_4"). A seleção da âncora rotulada "ae_4" nas páginas "P_C" ou "P_{F1}" resulta em abandonar a apresentação das páginas "P_C", "P_{F1}" e "P_{F2}" simultaneamente, voltando para a configuração de contexto $CC_0=\{P_A, P_{B1}, P_{B2}\}$. Um possível leiaute da apresentação desse exemplo é retratado na Figura 4.8 (b).



(a) $CC_1 = \{P_C, P_{F1}, P_{F2}\}$. (b) $CC_0 = \{P_A, P_{B1}, P_{B2}\}$.

Figura 4.8: Possíveis leiautes exibidos durante a navegação do hiperdocumento para as configurações dadas e N = 1.

4.5. VISÕES HIERÁRQUICAS

Thüring et al. [1995] apresentaram oito princípios de projeto com o objetivo de aumentar a legibilidade de hiperdocumentos e, conseqüentemente, a compreensibilidade dos mesmos. Tais princípios concentram-se, basicamente, em propor mecanismos para aumentar a coerência[13] do hiperdocumento e reduzir a sua sobrecarga cognitiva. Um dos oito princípios (*P_5: Visualizar a estrutura do documento*) propõe uma representação visual da estrutura do hiperdocumento, a qual deve fornecer uma visão geral do mesmo, ajudando o leitor na identificação dos tópicos de interesse e, conseqüentemente, na sua orientação. Desse modo, mecanismos para navegar de forma estruturada na hierarquia do hiperdocumento são altamente desejáveis. Esse tipo de "navegação estruturada" é fortemente apoiada pelo HMBS, sendo que nesta seção são apresentadas operações para explorar naturalmente a hierarquia de hiperdocumentos. Assim, a especificação do

[13] Segundo Thüring et al. [1995], um documento é coerente se um leitor consegue construir uma representação mental do mesmo que corresponda a fatos e relações do mundo real.

hiperdocumento em HMBS é um passo para satisfazer o princípio "P_5" de Thüring et al. [1995], permitindo obter uma visão geral adequada do hiperdocumento no nível de abstração desejado.

Adicionalmente, a representação visual do Statechart pode ser combinada com o princípio "P_6" (*Incluir pistas na estrutura de visualização que mostram a posição atual do leitor, o caminho percorrido até esta posição e as possíveis opções de navegação que podem ser escolhidas*). Em HMBS, a estrutura navegacional do hiperdocumento pode ser apresentada durante a navegação como um mapa, clicável ou não, para orientar os leitores, ajudando-os a encontrar o caminho desejado no hiperdocumento, exibindo futuras opções de navegação e também ajudando a construir uma representação mental coerente da estrutura organizacional do hiperdocumento [Thuering et al. 1991, Nanard; Nanard 1995].

Como apresentado no Capítulo 3 (Seção 3.3.1), a hierarquia de estados pode ser descrita por uma árvore AND/OR com profundidade "d". Considerando a estrutura do hiperdocumento ilustrada na Figura 4.2, a sua árvore AND/OR com os níveis de visão hierárquica é apresentada na Figura 4.9. Cada nó da árvore corresponde a um estado do Statechart e é rotulado por um conjunto de possíveis níveis-ascendentes associados a ele, dependendo da composição aninhada de estados. Estados do tipo AND não são associados a níveis-ascendentes, pois não são mapeados para páginas. O nível-ascendente para um nó folha da árvore é formado pelo conjunto {0}. Para os nós não folha o conjunto de níveis é a soma do nível-ascendente de cada um de seus filhos diretos mais um. Por exemplo, o nó associado ao estado "C" possui três níveis-ascendentes definidos pelo conjunto {1,2,3}: o nível 1 é do caminho de navegação para "F" ou "I", o nível 2 é do caminho para "E" e "E_1", e o nível 3 para "E" e "E_2". Deste modo, os níveis-ascendentes de um nó definem os níveis da hierarquia a partir da qual as páginas associadas aos estados serão visíveis na navegação. Por exemplo, a página associada ao estado "C" pode ser visualizada nos níveis de hierarquia 1, 2 e 3.

Para melhor explorar a organização hierárquica do hiperdocumento, o modelo define a operação denominada "Mostra_VisãoH[14]". No início da seção de navegação é definida uma variável auxiliar "h", com h = N+1, e

---

[14] Operação "Mostra Visão Hierárquica".

Se $h \leq d^{15}$, então

o sistema deve exibir todas as páginas *p* do conjunto CC_h, tal que

$CC_h = \{p \mid x \in S_S \wedge m(x) = p \wedge \rho^h(x) \cap SC \neq \phi\}$

Considerando a Figura 4.7 (b) com N = 0, se o leitor executar a operação "Mostra_VisãoH", o sistema deverá exibir o leiaute das janelas lógicas ilustrado na Figura 4.10, na qual a janela de visão hierárquica aparece parcialmente sobreposta às outras associadas ao conteúdo das páginas "P_{F1}" e "P_{F2}".

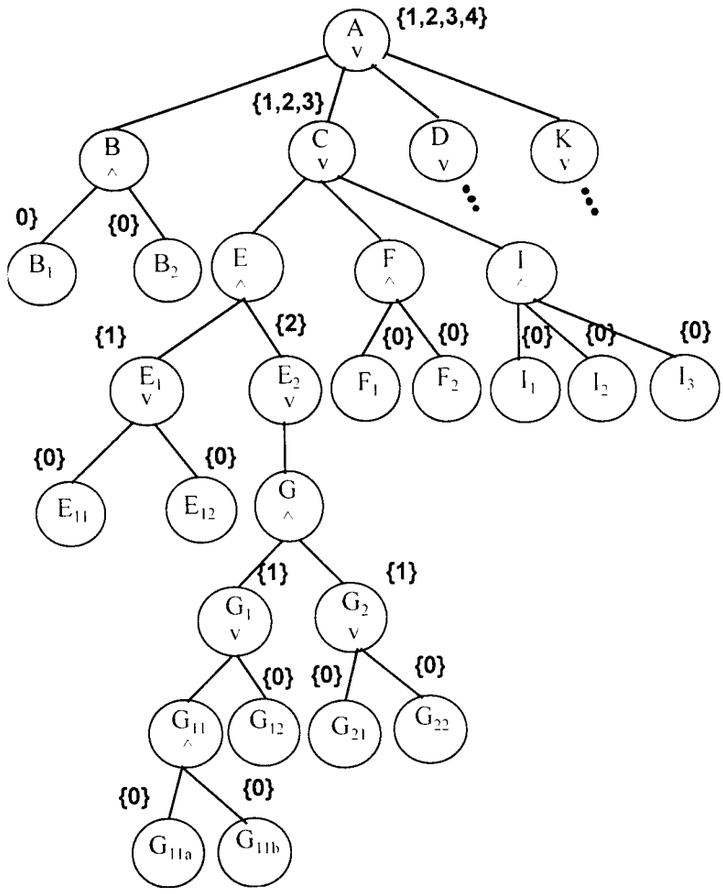


Figura 4.9: Árvore AND/OR para o Statechart da Figura 4.2, com os níveis de visão hierárquica rotulados nos nós.

A janela de visão hierárquica habilitada pela operação "Mostra_VisãoH" possui três ligações estruturais adicionais, representadas pelas operações: *UP*, DOWN e DISMISS. A ação associada à operação DISMISS finaliza a operação "Mostra_VisãoH". As operações UP e DOWN alteram o valor atual de "h". subindo

---

15 Altura da árvore AND/OR.

ou descendo, respectivamente, um nível na hierarquia e, conseqüentemente, modificam as páginas apresentadas como resultado da operação de visão hierárquica:

UP: Se $h < d$, então $h = h+1$; ativar "Mostra_VisãoH".

DOWN: Se $h > N$, então $h = h-1$; ativar "Mostra_VisãoH".

A execução de "Mostra_VisãoH" não altera a configuração de estados corrente do Statechart, apenas fornece um modo alternativo de acesso às informações contidas em páginas associadas a estados que estão em níveis acima (UP) ou abaixo (DOWN) na hierarquia. Se não existe nó na árvore AND/OR associado a um estado do tipo OR ou atômico acima ou abaixo na hierarquia, o *browser* não altera as páginas apresentadas até então. Essa situação é ilustrada a partir da configuração retratada na Figura 4.10, assumindo que o botão UP foi pressionado. Como resultado dessa seleção a página "P_A" associada ao estado "A" será mostrada na janela de visão hierárquica. Se o botão UP for pressionado novamente, a janela de visão hierárquica não é modificada, pois o nó "A" é raiz da árvore AND/OR.

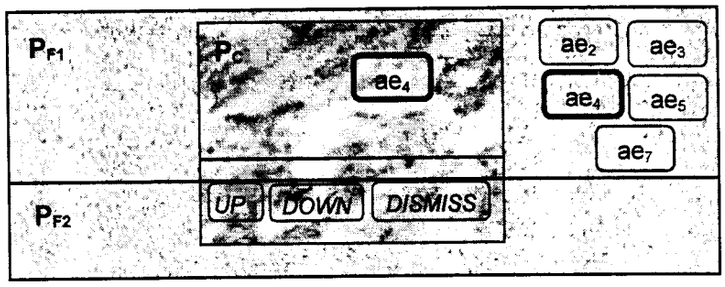


Figura 4.10: Ativação da operação "Mostra_VisãoH" durante a navegação do hiperdocumento, $SC_1 = \{F_1, F_2\}$ e $N = 0$.

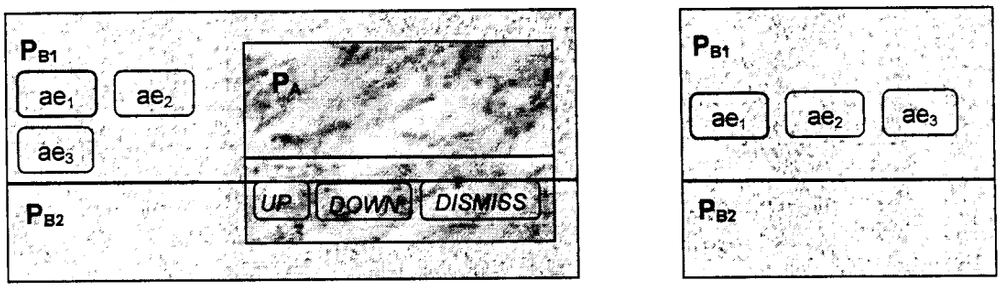
Um exemplo prático para essa operação é a navegação por um hiperdocumento que representa um livro. Com o mecanismo de visões hierárquicas é possível ao leitor subir na hierarquia para descobrir qual seção e/ou capítulo está sendo explorado. Adicionalmente, ainda que o leitor esteja navegando hierarquicamente no hiperdocumento pelas operações UP e DOWN, é possível ativar as âncoras habilitadas pela exibição das páginas correspondentes. Quando uma âncora é ativada, três interpretações podem ser adotadas pelo sistema:

a) Buscar o evento especificado, via mapeamento "ae", ativar a transição subjacente e gerar uma nova configuração de estados do Statechart. Em

seguida, as janelas lógicas associadas às páginas da nova configuração de contexto, além das páginas resultantes da aplicação da operação "Mostra_VisãoH" sobre tal configuração, são apresentadas. O valor de "h", modificado pela operação "Mostra_VisãoH", é mantido. Por exemplo, a Figura 4.11 (a) ilustra as janelas apresentadas depois da ativação da âncora "ae_4" de "P_{F1}" ou "P_C" da Figura 4.10 (N = 0), bem como a nova configuração corrente de estados SC = {B_1, B_2};

b) Ativar a transição correspondente e o *browser* apresenta as janelas lógicas associadas às páginas da nova configuração do Statechart. As páginas resultantes da operação "Mostra_VisãoH" anterior são inibidas, a operação "Mostra_VisãoH" é encerrada e o valor de "h" é atualizado com o valor original de N especificado pelo autor. Partindo do exemplo que resultou na configuração exibida na Figura 4.11 (a), a janela lógica apresentada pelo sistema hiperdocumento, neste segundo caso, é ilustrada na Figura 4.11 (b); e

c) Não habilitar as âncoras exibidas pela janela resultante da operação "Mostra_VisãoH", permitindo ativar somente as âncoras associadas com as páginas da configuração de contexto.



(a) SC = {B_1, B_2}; N = 0 e h=1. (b) SC = {B_1, B_2}; N = 0 e h=0.

Figura 4.11: Resultado da ativação de "ae_4" na janela de visão hierárquica da Figura 4.10.

A primeira alternativa pode ser interessante se o hiperdocumento é composto por uma estrutura hierárquica uniforme (por exemplo, um livro que é subdividido em capítulos, seções e subseções). Entretanto, essa abordagem pode provocar desorientação para o leitor, pois as janelas lógicas apresentadas são modificadas bruscamente. A terceira alternativa é muito restritiva no processo de navegação. Desse modo, a segunda alternativa pode oferecer um meio-termo adequado.

4.6. CARACTERÍSTICAS DO HMBS

Nesta seção são apresentadas algumas características e propriedades do HMBS relevantes no contexto do projeto de hiperdocumentos. Tais características estabelecem parâmetros para compará-lo com outros modelos (estruturais e comportamentais) afins na literatura.

4.6.1. Estruturação da Informação

Segundo Ginige et al. [1995], os hiperdocumentos podem ser organizados em três tipos de estruturas básicas: hierárquica, linear e rede (grafo). O HMBS permite que a informação seja estruturada nesses três tipos, podendo inclusive combiná-los. A Figura 4.12 apresenta as estruturas de organização hierárquica, linear e em rede, segundo a notação HMBS.

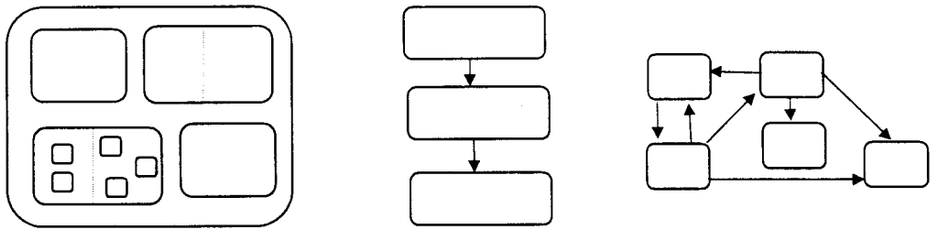


Figura 4.12: Estruturas de organização de informação no HMBS.

4.6.2. Separação entre Estrutura e Conteúdo

A separação entre a estrutura e o conteúdo do hiperdocumento é garantida pelos mapeamentos definidos no modelo. A definição dos mapeamentos estados/páginas ("m") e âncoras/eventos ("ae") garantem que toda a estrutura esteja descrita em um nível distinto daquele no qual se encontram as porções de conteúdo da informação, como proposto pelo modelo Dexter. Adicionalmente, o mecanismo de definição dos mapeamentos no HMBS garante flexibilidade na especificação, possibilitando que objetos, tanto da estrutura quanto do conteúdo, sejam reutilizados de maneiras distintas. Por exemplo, um mesmo evento pode estar associado a várias âncoras, dependendo do contexto de informação utilizado e o conteúdo de uma página pode estar associado a outra página do hiperdocumento.

4.6.3. Contextos de Informação

O conceito de superestados dos Statecharts pode ser usado para definir os contextos de informação existentes no hiperdocumento. Um estado-pai determina o contexto dos seus subestados e inter-relacionamentos entre eles, definindo assim os nós (ou páginas) de composição que admitem aninhamento [Soares et al. 1994].

O conceito de contextos de navegação, proposto por Schwabe e Barbosa [1994], pode ser especificado no HMBS como um conjunto de páginas e ligações constituindo um subgrafo do hiperdocumento. As páginas no contexto recuperam um determinado assunto segundo uma certa perspectiva e em um certo nível de conhecimento. Em geral, inclui um caminho no qual é proposto o próximo passo, mas deixa o leitor confortável para explorar o material sem sentir-se desorientado ou perdido. Os contextos de navegação podem ser implementados pelas estruturas de acesso roteiro guiado e índice.

4.6.4. Especificação em Níveis de Abstração e de Granularidade

A possibilidade de decomposição de estados permite ao autor abordar o processo de especificação de hiperdocumentos em vários níveis de abstração e granularidade. Na medida em que os níveis de abstração aumentam, pode-se explorar diferentes níveis de granularidade, tanto para a especificação da estrutura dos estados quanto para os mecanismos de concorrência e sincronização. Normalmente, os detalhes da especificação do hiperdocumento devem aparecer no nível mais baixo de abstração, no qual o nível de granularidade é mais fino.

4.6.5. Acesso a Informação por Índice

O HMBS permite que a navegação seja feita da forma tradicional, seguindo as ligações habilitadas pelas âncoras ou por estruturas alternativas de acesso. Um recurso útil ao leitor durante a navegação é a pesquisa por páginas relacionadas a uma palavra-chave. O acesso direto a páginas pode ser implementado por índices, que são estruturas de acesso alternativas apoiadas pelo modelo.

Os títulos associados às páginas podem ser utilizados como chaves de busca para compor o índice de acesso direto. Em uma primeira alternativa, os títulos apresentados no índice são restritos aos títulos de páginas associadas a estados que satisfazem a condição, $s \in S_S$ e $N \in$ nível_ascendente(s), tal que a página associada ao estado "s" seja visível para o nível de visibilidade "N". Dessa maneira,

o índice contém somente os títulos das páginas visíveis no nível de visibilidade "N".

No momento em que o leitor escolhe um título, o sistema hiperdocumento deve identificar a página "P_S" associada ao título e o estado "s" associado à página. Além disso, o sistema deve escolher os estados que, com "s", definem uma configuração de contexto adequada. Assim, o sistema apresenta a página "P_S" juntamente com as páginas correspondentes aos estados que, com "s", definem uma configuração de estados válida, escolhida segundo uma função heurística. O HMBS não estabelece uma função específica para ser implementada no sistema, mas possíveis opções seriam: escolher os estados *default* da especificação em Statechart, ou os estados do tipo OR com menos subestados, ou ainda os estados do tipo OR não *default*.

Para gerar uma configuração válida de estados a partir de um estado específico (denominado estado inicial) desenvolveu-se um algoritmo (ilustrado na Figura 4.13) que usa a árvore AND/OR do Statechart e um procedimento recursivo que visita todos os estados descendentes do estado inicial, adicionando seus descendentes *default* ao conjunto de estados ativos (CE).

A título de ilustração, considere o hiperdocumento especificado pelo Statechart da Figura 4.2, a função de mapeamento "m" apresentada na Tabela 4.1, N = 1 e a árvore AND/OR apresentada na Figura 4.9. Supondo que, durante a navegação, um índice de títulos foi proposto para iniciar a navegação e o leitor tenha selecionado o título "T_{G21}" relativo à página "P_{G21}" associada ao estado "G_{21}". Para que o sistema apresente a página correspondente é necessário obter uma configuração de estados que contenha o estado "G_{21}". Aplicando o algoritmo nesse contexto, gera-se o conjunto de estados CE = {A, C, E, E_1, E_2, E_{11}, G, G_1, G_2, G_{11}, G_{11a}, G_{11b}, G_{21}}, e a configuração de estados atômica válida é SC = {E_{11},G_{11a},G_{11b},G_{21}}. Em conseqüência, o sistema hiperdocumento apresenta as janelas lógicas da configuração de contexto CC = {P_{E11}, P_{G11a}, P_{G11b}, P_{E1}, P_{G1}, P_{G2}, P_{G21}}, pois N = 1 (Figura 4.14).

Uma abordagem alternativa para a construção de índices é tornar o processo de busca mais geral, garantindo ao leitor acesso direto a todas as páginas do hiperdocumento, independentemente de serem potencialmente visíveis ou não no nível de visibilidade "N". Suponha que a navegação é iniciada pelo título "T_K" da página "P_K" associada ao estado "K". Se a página "P_K" é visível no nível "N", ou

seja, min(nível_ascendente(k)) $\leq$ N, então essa situação é idêntica à primeira abordagem. Caso contrário, o nível de visibilidade recebe momentaneamente o valor min(nível_ascendente(k)) e desse modo pode-se obter o conjunto de estados ativos utilizando os algoritmos anteriores passando como parâmetro o estado "K".

Algoritmo Cria_Configuração (T: Estado)

{ T está associado a uma página e representa um nó da árvore AND/OR }
Início
(1) Iniciar um CE com um conjunto vazio;
(2) Adicionar T em CE;
(3) Se T não é um estado atômico (nó folha), então
(3.1) Ativar o procedimento Adiciona_Descendente_Default(T,CE);
(4) Para cada ancestral M de T na árvore AND/OR faça
(4.1) Adicionar M em CE;
(4.2) Se M é um estado do tipo AND então
Para cada filho direto K de M (excluindo o caminho ancestral) faça:
- Adicionar K em CE;
- Ativar o procedimento Adiciona_Descendente_Default(K,CE);

(5) Exibir todas as páginas associadas aos estados x $\in$ CE de acordo com semântica de navegação do HMBS e o nível de visibilidade N;
(6) SC é a configuração de estados contendo os estados atômicos de CE;
Fim.

Procedimento Adiciona_Descendente_Default(X: Estado, CE: Conjunto de Estados)

Início
(1) Se X é um estado atômico
Então return;
(2) Senão Se X é um estado OR
Então
(2.1) Escolher X' como default de X (uso da função heurística)
(2.2) Adicionar X' em CE;
(2.3) *Ativar o procedimento* Adiciona_Descendente_Default (X',CE);
(3) Senão Se X é um estado do tipo AND
Então
(3.1) Para cada filho direto X'' de X faça:
(3.1.1) Adicionar X'' em CE;
(3.1.2) Adiciona_Descendente_Default (X'', CE);
Fim.

Figura 4.13: Algoritmo para gerar uma configuração de estados válida.

4.6.6. Uso de História, de Variáveis e de Ações

O uso de história pode ser explorado para especificar situações nas quais

contextos anteriormente visitados precisam ser recuperados. Um caso típico em que ocorre tal situação é em hiperdocumentos do tipo ajuda on-line, por exemplo, em tutoriais. O uso de história permite uma considerável redução no número de transições na especificação de hiperdocumentos e, conseqüentemente, pode reduzir sua complexidade.

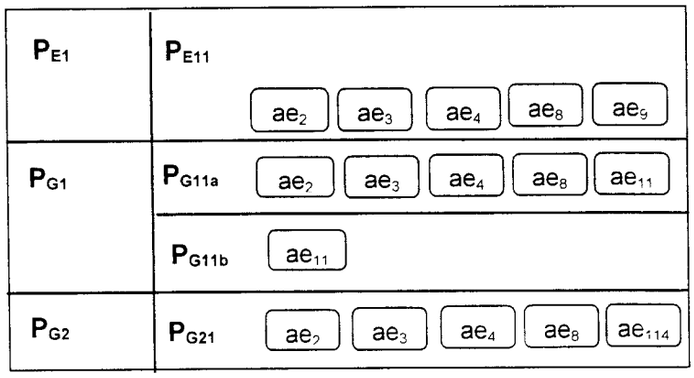


Figura 4.14: Possível leiaute exibido após a escolha do título "T_{G21}" durante a navegação do hiperdocumento (SC = $\{E_{11}, G_{11a}, G_{11b}, G_{21}\}$ e N = 1).

A Figura 4.15 apresenta uma configuração de contexto CC = $\{P_{E1}, P_{E12}, P_{G1}, P_{G11a}, P_{G11b}, P_{G2}, P_{G22}\}$ do hiperdocumento especificado pelo Statechart da Figura 4.2, considerando N = 1. Para exemplificar o uso de história, suponha que o leitor necessite de uma informação que pode ser encontrada em um contexto que inclui o estado "K" e, para obtê-la, ativa a âncora "ae_3" em "P_{E12}" ou "P_{G11a}". que dispara a ligação correspondente. A estrutura navegacional do hiperdocumento deveria permitir ao leitor voltar à configuração de contexto anterior após encontrar a informação no contexto que contém "K". Na especificação original do hiperdocumento (Figura 4.2), tal situação é implementada pela transição com rótulo "e_1", que tem como origem o estado "K" e destino o símbolo de história H* associado ao estado "C". Uma simulação dessa mesma especificação sem o uso da história H* é apresentada na Figura 4.16, na qual a transição "e_1" foi substituída por oito novas transições, tornando a especificação mais complexa.

Além do mecanismo história, o uso de variáveis e de ações também pode enriquecer as especificações HMBS. A alteração dos valores das variáveis durante a navegação permite personalizar e controlar o caminho a ser percorrido pelo leitor. Adicionalmente, permite especificar, de forma simplificada, outras aplicações interessantes, tal como aprendizagem programada baseada na execução de ações associadas a transições. Um sistema destinado a auxiliar o processo de

ensino/aprendizagem poderia definir em tempo de execução os caminhos mais adequados para um dado leitor considerando seu conhecimento e perfil sobre o assunto, o qual pode ser inferido com base nas escolhas do leitor durante as seções de navegação ou, alternativamente, com base em pesos/notas atribuídas aos caminhos de navegação escolhidos.

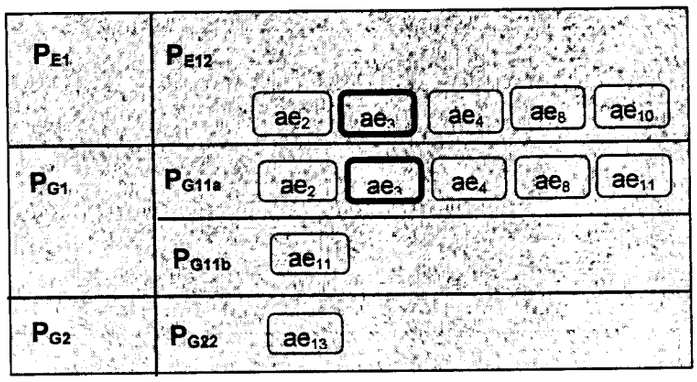


Figura 4.15: Configuração de contexto $\{P_{E1},P_{E12},P_{G1},P_{G11a},P_{G11b},P_{G2},P_{G22}\}$ e N = 1.

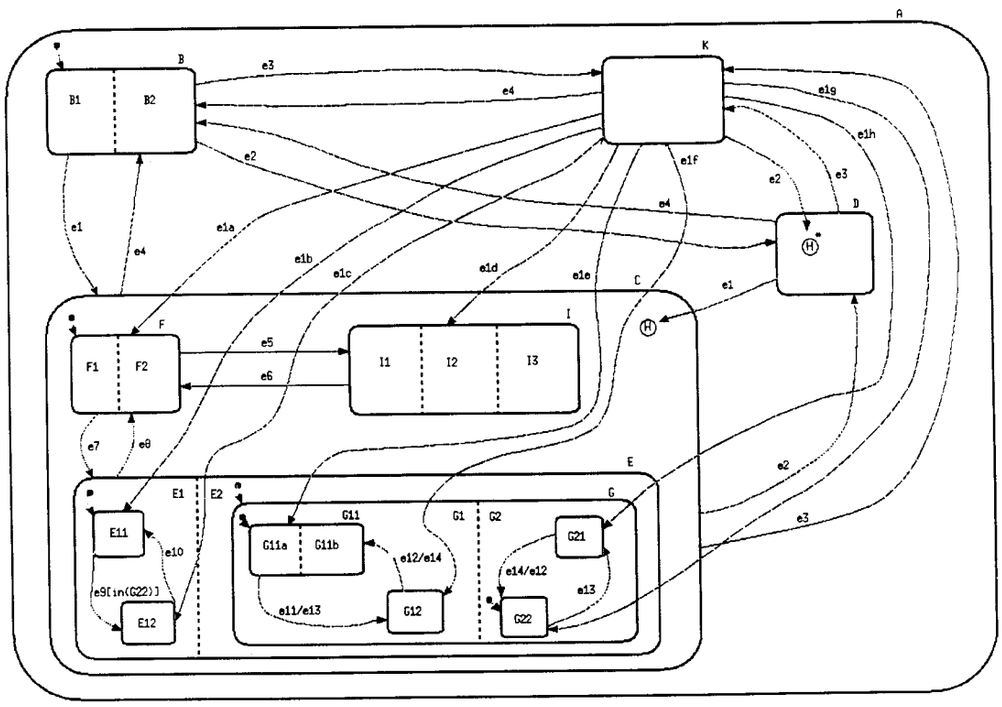


Figura 4.16: Statechart da Figura 4.2 sem o uso de uma transição com história.

O mecanismo de *broadcast* proporcionado pelo uso de ações pode ser explorado, potencialmente, para fornecer suporte à sincronização em diversos tipos de aplicações multimídia. Tal mecanismo é decorrente da semântica de execução

das ações, que geram eventos que podem ser detectados por componentes ortogonais. Por exemplo, um vídeo que termina antes de seu áudio poderia gerar um sinal para finalizar a apresentação do áudio e avançar para outras partes da apresentação.

4.7. CONSIDERAÇÕES FINAIS

Este capítulo apresentou a especificação formal do modelo HMBS, que utiliza Statecharts como técnica subjacente para a especificação da estrutura organizacional e navegacional de hiperdocumentos em tempo de autoria, e do seu comportamento dinâmico em tempo de navegação. Utilizando a especificação de Statecharts discutida no Capítulo 3, as características relacionadas à sintaxe do modelo foram apresentadas e contextualizadas por exemplos. Além disso, foi apresentada a semântica de navegação utilizada pelo modelo para explorar o hiperdocumento em tempo de navegação.

A capacidade de especificar concorrência entre estados dos Statecharts possibilita ao HMBS especificar a exibição simultânea de páginas. Pelo mesmo motivo, o modelo permite a definição de caminhos concorrentes de navegação, ou seja, ligações independentes podem ser definidas para causar a modificação de apenas uma parte da apresentação. Tais características enriquecem o modelo, tornando-o bastante flexível para a especificação de vários domínios de aplicação.

A possibilidade de analisar os hiperdocumentos com o objetivo de identificar inconsistências estruturais, bem como problemas relacionados à visualização e à navegação é uma propriedade importante do HMBS. No próximo capítulo tais propriedades são discutidas objetivando identificar possíveis problemas navegacionais antes de o hiperdocumento ser explorado pelo leitor.

Capítulo 5

ANÁLISE DE ESPECIFICAÇÕES HMBS

5.1. CONSIDERAÇÕES INICIAIS

O uso de técnicas e/ou métodos de especificação formal no processo de desenvolvimento de software é sugerido por vários autores [Buchmann; Sunshine 1980, Wing 1990, Harel et al. 1990, Fraser et al. 1991]. Postula-se nesta tese que o uso de modelos de especificação formal pode conduzir a um aumento de qualidade[16] e confiabilidade de hiperdocumentos, permitindo realizar análises rigorosas na especificação e, conseqüentemente, na sua implementação em uma linguagem de programação; permite, também, checar a completitude da especificação e, ainda, verificar se uma implementação satisfaz aos requisitos de análise estabelecidos. A especificação pode ser analisada empregando operadores matemáticos, uma vez que argumentos de prova formal podem ser usados para testar e provar a consistência e a corretitude da sintaxe da especificação.

---

[16] Nesta tese o termo qualidade de hiperdocumento é adotado como sinônimo de "adequação para uso" (*fitness for use*), ou seja, a característica que contempla as expectativas e necessidades do leitor.

O HMBS oferece a possibilidade de análise de hiperdocumentos com o objetivo de identificar inconsistências estruturais, bem como problemas relacionados à visualização, navegação e análise de propriedades. Na Seção 5.2 algumas propriedades dinâmicas para sistemas reativos são analisadas no contexto de hiperdocumentos. Na Seção 5.3 são descritas possíveis abordagens para identificar problemas de especificação e/ou para modelar situações típicas em hiperdocumentos. As conclusões deste capítulo são apresentadas na Seção 5.4.

5.2. ANÁLISE DE PROPRIEDADES

Os formalismos Statechart e Redes de Petri possuem um comportamento semelhante no que diz respeito à análise de propriedades dinâmicas para sistemas reativos. Boaventura [1992] e Masiero et al. [1994] definiram para Statecharts algumas propriedades usadas em Redes de Petri, tais como: alcançabilidade, reiniciabilidade, impasse, seqüência de disparos e uso de transições. A maioria dessas propriedades é verificada em Statecharts utilizando técnicas de análise do espaço de estados e, em especial, tais técnicas são derivadas a partir da árvore de alcançabilidade, uma estrutura que representa as possíveis configurações de estado alcançáveis durante a sua execução.

Nesta seção é ilustrado como construir a árvore de alcançabilidade para especificações em Statechart. A partir da árvore gerada, várias análises podem ser aplicadas em tempo de autoria para validar o hiperdocumento, permitindo ao autor detectar anomalias. As propriedades verificadas para Statecharts são também aplicadas no contexto de hiperdocumentos. Tal abordagem é interessante, principalmente, em documentos que crescem sem uma organização cuidadosa, como os disponíveis nas páginas da WWW. Em particular, são abordados vários problemas típicos em hiperdocumentos, discutidos, também, por Stotts e Furuta [1989], que utilizam a árvore de alcançabilidade de Redes de Petri para determinar características do hiperdocumento especificado. De maneira semelhante, Courtiat e Oliveira [1996] realizaram uma análise na árvore de alcançabilidade da técnica formal RT-LOTOS para identificar inconsistências temporais em especificações multimídia.

5.2.1. Árvore de Alcançabilidade

5.2.1.1. Definição e Construção

A árvore de alcançabilidade definida para Statecharts é semelhante à árvore

de Redes de Petri, com algumas características próprias, tais como a notação adotada, o mecanismo de representação da hierarquia de estados e o tratamento de história [Masiero et al. 1994]. Para construir a árvore de alcançabilidade consideram-se todas as possíveis seqüências de computação para o Statechart. A computação envolve uma seqüência de passos de tempo e, a cada passo, assume-se que as expressões de evento são avaliadas como verdadeiras, e as transições habilitadas levam a uma nova configuração de estados. A seqüência de configurações de estado é representada pelo conjunto

$\{(SC_i, \gamma_i, \Pi_i^*)\}$, $i \geq 0$, na qual:

a) $SC_0=(X_0, \Pi_0, \Theta_0)$ tal que "X_0" é a configuração inicial, "Π_0" é o conjunto de eventos externos gerados pelo ambiente (por exemplo, seleção de uma âncora) e "Θ_0" é o conjunto de condições primitivas cujo valor é verdadeiro;

b) γ_i é o passo realizado em SC_i, no instante de tempo σ_{i+1}, $i \geq 0$; e

c) Π_i^* é o conjunto de eventos gerados por γ_i, para $i \geq 0$.

Assim, obtém-se a seqüência de possíveis configurações, tal que "SC_{i+1}" é a configuração do sistema alcançada a partir de "SC_i" quando se ativam as transições em γ_i, $i \geq 0$.

A árvore de alcançabilidade é construída com base nas configurações de estado, de modo que cada nó da árvore, com exceção dos nós história, está associado a uma configuração. Os arcos da árvore são direcionados e rotulados pelos eventos contidos nas transições. Os nós história representam todas as configurações alcançáveis a partir de transições associadas a símbolos de história. Os símbolos "H" e "*" aparecem nos nós da árvore em substituição às configurações alcançáveis cujos estados são ativados, respectivamente, por história simples "H" e recursiva "H*". Ao percorrer a árvore, após a sua criação, quando um nó história é encontrado, os símbolos "H" e "*" são substituídos pela última configuração alcançada por aquele estado.

A seguinte notação é adotada para representar um nó da árvore de alcançabilidade: parênteses denotam um estado composto do tipo OR, colchetes denotam superestados AND, e os valores 1 e 0 representam, respectivamente, os estados atômicos ativos e não ativos. Tal notação preserva a hierarquia e o tipo de decomposição, embora somente os estados básicos sejam explicitamente

representados. Para o Statechart apresentado na Figura 5.1[17], os nós da árvore assumem a seguinte forma genérica:

([B_1, B_2], ([F_1, F_2], [I_1, I_2, I_3], [(E_{11}, E_{12}), [([G_{11a}, G_{11b}], G_{12}), (G_{21}, G_{22})]]), D, K).

A configuração de estados inicial é o nó raiz da árvore de alcançabilidade e para o Statechart da Figura 5.1, SC_0 = {B_1,B_2} e o nó raiz da árvore é, então, C_0=([1,1],([0,0],[0,0,0],[(0,0),[([0,0],0),(0,0)]]),0,0). A árvore de alcançabilidade do Statechart da Figura 5.1 é apresentada na Figura 5.2.

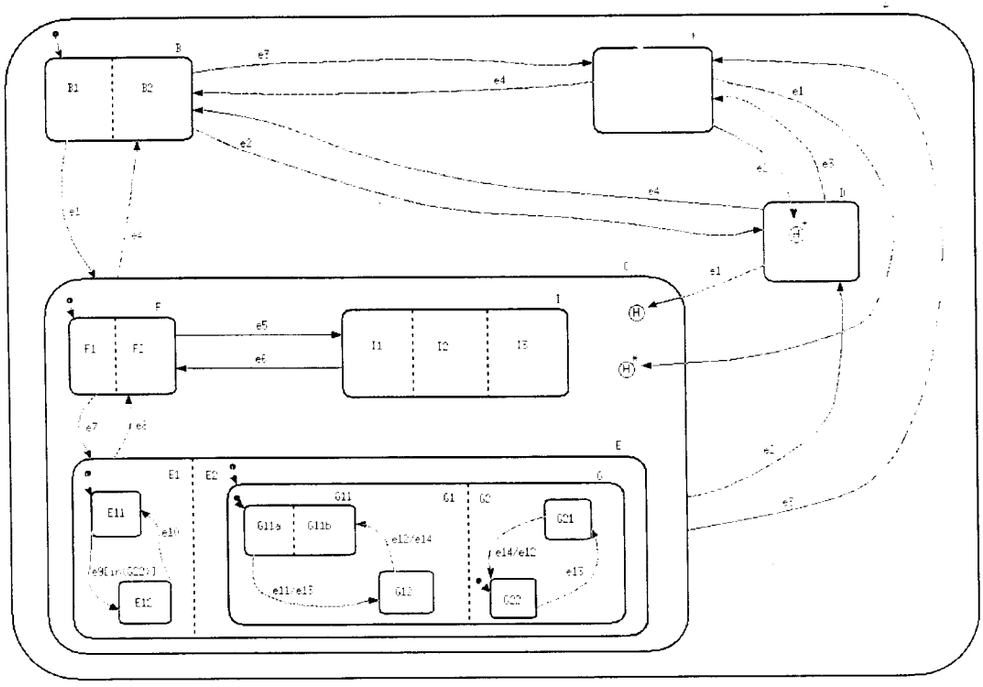


Figura 5.1: Estrutura de um hiperdocumento.

5.2.1.2. Problemas

O processo de construção da árvore de alcançabilidade pode ser problemático no caso de Statecharts complexos. Cabe salientar que, como a árvore de alcançabilidade é finita, o processo de sua construção também é finito. Um Statechart possui um número finito de estados e transições e, conseqüentemente, o número de configurações de estados possíveis é finito. Por outro lado, a execução do Statechart pode ser infinita, pois é possível repetir infinitamente as possíveis configurações de estados alcançáveis.

---

17 Para simplificar a árvore de alcançabilidade, os estados OR "D" e "K", decompostos em outro nível, conforme apresentado no Capítulo 4, são considerados neste capítulo como estados atômicos.

Uma questão a ser considerada é a explosão de estados da árvore, cujo número máximo de nós é limitado pelo espaço de memória da máquina. O tamanho da árvore gerada pode ser crítico para hiperdocumentos com um grande número de páginas de informação, uma situação que exige tratamento especial. As vantagens proporcionadas pela análise da árvore de alcançabilidade na detecção de inconsistências e anomalias no hiperdocumento são essenciais, e tais análises não precisam necessariamente ser feitas on-line, o que minimiza o problema de explosão.

Várias propostas para minimizar o problema da explosão de estados foram estudadas na literatura e podem ser adaptadas nesse contexto. Alguns autores, como Buchmann e Sunshine [1980] e Leveson e Stolzy [1987], sugerem validações parciais do sistema. Um hiperdocumento complexo especificado em vários níveis hierárquicos pode ser analisado por partes, minimizando o problema. O algoritmo de construção é usado para construir uma árvore para cada um desses níveis hierárquicos e a análise pode ser feita parcialmente. Outras abordagens limitam o escopo dos testes, isto é, propõem a construção da árvore para as partes críticas e isoladas do sistema. Quando partes maiores do sistema necessitam ser testadas, uma possível solução pode ser eliminar algumas transições, gerando uma árvore com um menor número de configurações. Essas restrições devem ser introduzidas com muito cuidado para garantir que os testes sejam realizados sem perda de informações relevantes.

5.2.2. Alcançabilidade

A árvore de alcançabilidade pode ser usada para identificar páginas do hiperdocumento que não são alcançáveis durante a navegação. Pode-se analisar a alcançabilidade de uma página a partir da configuração de contexto inicial ou a partir de uma configuração qualquer. Dado o título "T_1", que identifica unicamente a página "P_1", o objetivo é verificar, a partir da configuração de estados inicial "SC_0", se o estado "s_1" associado à página "P_1" (via $m(s_1) = P_1$) faz parte de alguma configuração de estados alcançável. Em caso afirmativo, pode-se obter uma seqüência de âncoras (associadas a eventos pela função "ae") que leva da configuração de contexto inicial até uma configuração que contenha a página "P_1".

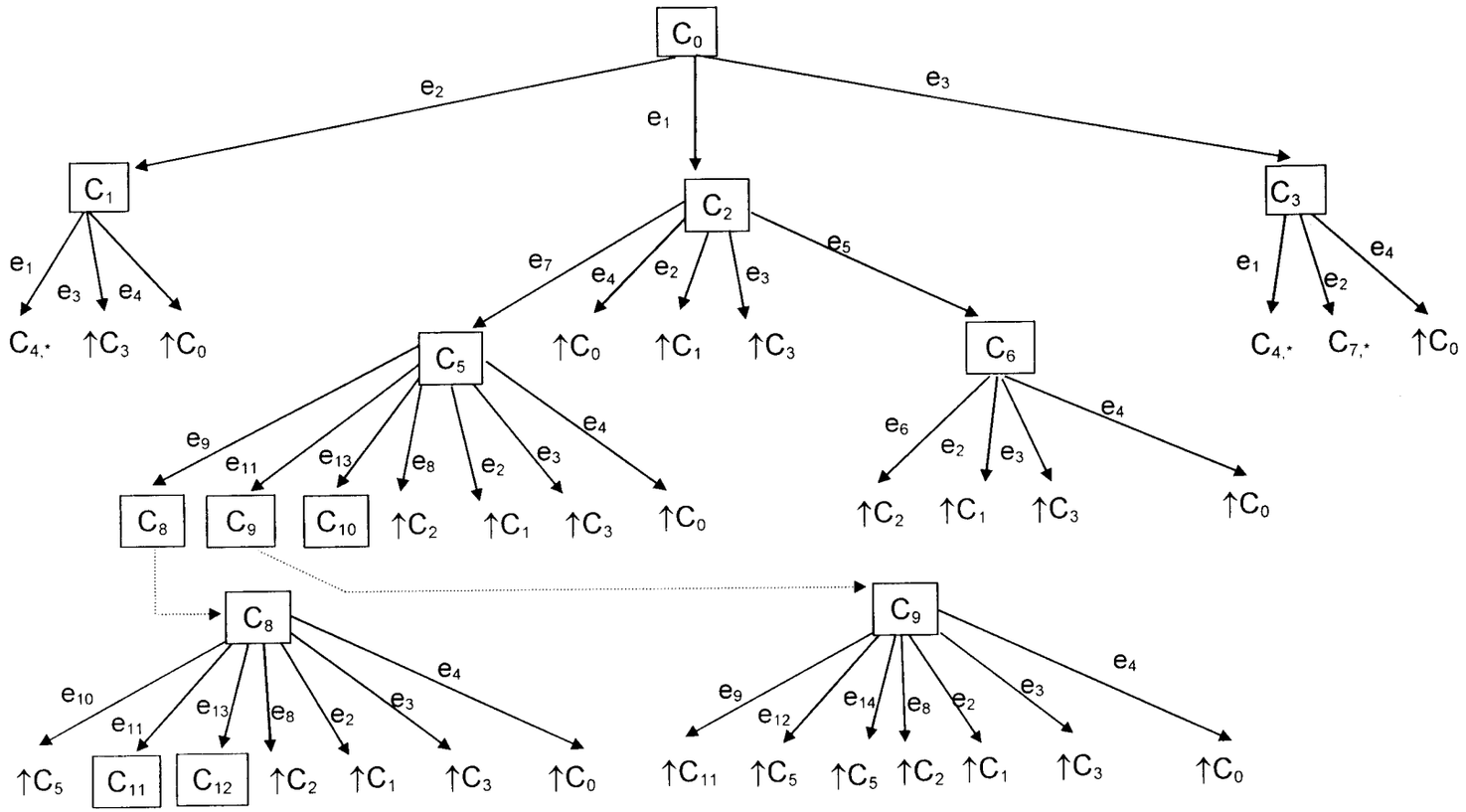
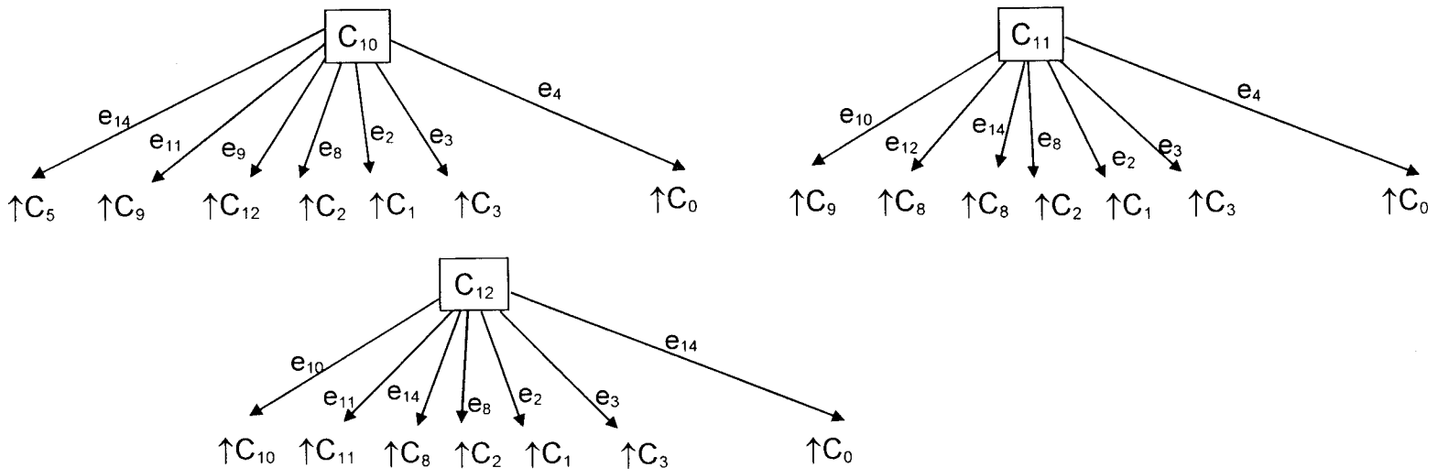


Figura 5.2: Árvore de alcançabilidade para o Statechart da Figura 5.1 (continuação na página seguinte).



$([B_1,B_2],([F_1,F_2],[I_1,I_2,I_3],[(E_{11},E_{12}),[([G_{11a},G_{11b}],G_{12}),(G_{21},G_{22})]]),D,K)$

C_0=([1,1],([0,0],[0,0,0],[(0,0),[([0,0],0),(0,0)]]),0,0)
C_1=([0,0],([0,0],[0,0,0],[(0,0),[([0,0],0),(0,0)]]),1,0)
C_2=([0,0],([1,1],[0,0,0],[(0,0),[([0,0],0),(0,0)]]),0,0)
C_3=([0,0],([0,0],[0,0,0],[(0,0),[([0,0],0),(0,0)]]),0,1)
C_4=([0,0],([*,*],[*,*,*],[(*,*),[([*,*],*),(*,*)]]),0,0)
C_5=([0,0],([0,0],[0,0,0],[(1,0),[([1,1],0),(0,1)]]),0,0)
C_6=([0,0],([0,0],[1,1,1],[(0,0),[([0,0],0),(0,0)]]),0,0)
C_7=([0,0],([0,0],[0,0,0],[(0,0),[([0,0],0),(0,0)]]),*,0)
C_8=([0,0],([0,0],[0,0,0],[(0,1),[([1,1],0),(0,1)]]),0,0)
C_9=([0,0],([0,0],[0,0,0],[(1,0),[([0,0],1),(1,0)]]),0,0)
C_{10}=([0,0],([0,0],[0,0,0],[(1,0),[([1,1],0),(1,0)]]),0,0)
C_{11}=([0,0],([0,0],[0,0,0],[(0,1),[([0,0],1),(1,0)]]),0,0)
C_{12}=([0,0],([0,0],[0,0,0],[(0,1),[([1,1],0),(1,0)]]),0,0)

Figura 5.2: Árvore de alcançabilidade para o Statechart da Figura 5.1.

Geralmente há vários caminhos que partem de uma determinada configuração de contexto e atingem outra. O caminho mais curto pode ser encontrado fazendo-se uma busca em largura na árvore de alcançabilidade. Na Figura 5.2, observa-se que a página "P_{G11a}", por exemplo, é alcançável, pois o estado "G_{11a}" está ativo nas configurações definidas pelos nós "C_5", "C_8", "C_{10}" e "C_{12}". O caminho mais curto é obtido pelo nó "C_5" e a seqüência de âncoras para atingir tal configuração é "ae_1" e "ae_7".

Pode-se considerar que existe uma anomalia no hiperdocumento quando uma página não é alcançável a partir da configuração de contexto inicial. É de extrema importância para o autor do hiperdocumento determinar se existem páginas que nunca serão alcançadas, pois isso pode indicar um erro na especificação. Tal propriedade pode ser verificada da seguinte maneira: Dado um hiperdocumento "H" e uma configuração de estados inicial "SC_0", uma maneira simples de determinar se uma página particular "P_S", associada a um estado "s", pode ser visualizada durante a navegação é computar a árvore de alcançabilidade e verificar se o estado "s" aparece em alguma configuração gerada. Se não aparecer, as informações da página nunca serão apresentadas quando o hiperdocumento for navegado. Porém, se a página for buscada por acesso direto (via um índice geral), as informações da página são apresentadas. Similarmente, é possível determinar se páginas podem ser apresentadas em conjunto com outras analisando se os estados associados aparecem simultaneamente em algum nó da árvore. Por exemplo, pela configuração "C_6" da árvore de alcançabilidade da Figura 5.2, observa-se que as páginas "P_{I1}", "P_{I2}" e "P_{I3}" são apresentadas simultaneamente.

5.2.3. Reiniciabilidade

Um hiperdocumento é reiniciável se, para toda configuração de contextos "CC", alcançada a partir de "CC_0", existe uma seqüência de âncoras que permite retornar à configuração inicial "CC_0". Se o hiperdocumento é reiniciável, então o Statechart subjacente também o é, mas o inverso não é verdadeiro, pois o autor pode não especificar todas as associações âncoras/eventos permitidas pelo HMBS. Desse modo, para garantir que o hiperdocumento seja reiniciável, deve-se garantir que sempre existe um caminho navegacional que permita ao leitor retornar à configuração de contexto inicial. A garantia de tal propriedade pode ajudar a diminuir a desorientação do leitor durante a navegação.

O hiperdocumento cuja árvore de alcançabilidade é apresentada na Figura 5.2 é reiniciável, pois a partir de todas as configurações de contexto é possível alcançar a configuração inicial $CC_0 = \{P_{B1}, P_{B2}\}$, definida pelo nó "$C_0$" da árvore. Uma outra propriedade interessante a ser observada na árvore de alcançabilidade é a presença de caminhos de navegação cíclicos, cuja ocorrência indica que uma mesma configuração de contexto pode ser visitada várias vezes durante a navegação.

5.2.4. Impasse

A árvore de alcançabilidade também possibilita a detecção de situações de impasse (*deadlock*) no hiperdocumento. A ausência de impasse garante que o leitor, durante a navegação, nunca atingirá uma configuração de contexto de inatividade total, na qual possa permanecer por tempo indefinido. Essa situação ocorre quando, por exemplo, o leitor visita uma página terminal, ou seja, uma página que não tem âncoras para serem ativadas.

O hiperdocumento poderá ter situações de impasse se o Statechart subjacente atingir configurações de estados terminais (configurações das quais nenhuma transição pode ser disparada), caso em que existirão nós terminais na sua árvore de alcançabilidade. Esses nós correspondem aos nós folha da árvore que não são réplicas de outros nós não folhas. Por exemplo, a árvore de alcançabilidade da Figura 5.2 não possui nó folha terminal, pois todos os nós já ocorreram anteriormente.

A presença de configurações de estados terminais no Statechart ou configurações de contexto terminais no hiperdocumento, entretanto, não implica necessariamente em uma situação de impasse. Por exemplo, hiperdocumentos direcionados ao processo de aprendizagem oferecem ao leitor/estudante um roteiro guiado do início ao fim de uma determinada experiência, ou curso. Nessa abordagem, as páginas terminais podem indicar o fim do roteiro guiado, não representando uma situação de impasse no hiperdocumento.

5.2.5. Vivacidade de Ligações

Uma outra propriedade aplicável na análise de hiperdocumentos é identificar se existem transições especificadas no Statechart que nunca disparam, ou seja, se existem ligações no hiperdocumento que não podem ser ativadas pelo leitor. Tal propriedade proporciona ao autor verificar possíveis inconsistências e anomalias na

especificação de ligações.

No hiperdocumento da Figura 5.1, uma possível configuração de contexto $\{P_{E11}, P_{G11a}, P_{G11b}, P_{G21}\}$ é definida pelo nó "C_{10}" da árvore de alcançabilidade (Figura 5.2). Nessa configuração, o sistema hiperdocumento apresenta as quatro páginas "P_{E11}", "P_{G11a}", "P_{G11b}" e "P_{G21}" e a âncora rotulada "ae_9", localizada na página "P_{E11}", não poderá ser ativada nesse contexto, pois a expressão condicional "$in(G_{22})$" associada não é satisfeita. Situações semelhantes a essa ilustram possíveis anomalias na especificação de ligações e compete ao autor analisar tais casos.

5.2.6. Seqüência Válida de Âncoras

No contexto de hiperdocumentos, é interessante analisar as possíveis seqüências de ativação de âncoras a partir de uma página inicial, ou seja, as relações de ordem temporal entre as páginas visitadas. Como a semântica de Statecharts possibilita determinar o conjunto de todas as seqüências de eventos/transições que podem ser ativadas durante a execução do Statechart a partir de um estado inicial, é possível verificar quaisquer restrições que o autor deseja impor na seqüência de navegação.

Exemplos de possíveis restrições em uma seção de navegação seriam: para visitar uma página "P_y" deve-se, anteriormente, ter visitado uma página "P_x"; se uma página "P_y" foi visitada então não se deve permitir ao leitor visitar uma página "P_x". Ambas as restrições podem ser analisadas e checadas pela árvore de alcançabilidade do Statechart subjacente ao hiperdocumento.

5.3. RECURSOS ADICIONAIS

5.3.1. Característica de Visualização

Um sistema hiperdocumento pode usar a árvore de alcançabilidade para determinar o número máximo de janelas lógicas ou subdivisões de uma mesma janela, dependendo da política de visualização adotada, que serão apresentadas simultaneamente durante a navegação. Tal característica pode ser empregada pelo módulo de interface que determina os leiautes de apresentação durante a navegação.

O número máximo de janelas na navegação, representado por Max_N, depende do nível de visibilidade (N) especificado para navegação no

hiperdocumento. Se o nível de visibilidade for zero (N = 0), então Max_0 é o número máximo de estados ativos (marcados por 1) nas configurações de estados do Statechart obtidas pela árvore de alcançabilidade. Para o hiperdocumento da Figura 5.1, cuja árvore de alcançabilidade é mostrada na Figura 5.2, $Max_0 = 4$, pois é o número máximo de estados ativos nas configurações de estados representadas pelos nós "C_5", "C_8", "C_{10}" e "C_{12}" da Figura 5.2. Dessa forma, o sistema pode prever e já dividir a janela de apresentação em quatro *frames* ou em quatro regiões bem definidas. Porém, se o nível de visibilidade for diferente de zero, então $Max_{N\neq0}$ é o número máximo de estados ativos, considerando a soma de todos os estados das configurações da árvore de alcançabilidade com seus respectivos estados ancestrais, dependendo de N. Considerando N = 1 e o hiperdocumento da Figura 5.1, $Max_1=7$, isto é, o número máximo de janelas lógicas apresentadas para explorar o hiperdocumento é sete.

5.3.2. Concorrência de Janelas Simultâneas

Segundo Ladd et al. [1997], a onipresença da WWW é fruto da simplicidade do modelo de grafo subjacente. Infelizmente, tal formalismo não representa as características de hierarquia, concorrência e sincronização. O uso de *frames* permite um certo nível de apresentação concorrente, ainda que rudimentar, pois não permite a sincronização entre as páginas de informação. A concorrência na apresentação é altamente desejável em muitas situações: o autor deseja apresentar trechos de documentos em paralelo com um resumo, análise ou comentário; um educador deseja apresentar visões diferentes do mesmo material tais como um gráfico e uma descrição textual.

Uma vez representada a concorrência na apresentação das informações, é importante habilitar a sincronização entre elas, por exemplo: o autor deseja manter uma relação "veja-antes" (pré-requisito) entre diferentes porções de informação do hiperdocumento; um educador deseja que um estudante pare e faça uma avaliação antes de iniciar o curso ou um determinado tópico.

Usando estados ortogonais, o HMBS permite especificar mecanismos apropriados para representar a concorrência na apresentação de múltiplas páginas no hiperdocumento, além de fornecer aos autores suporte para a especificação de sincronização de atividades concorrentes. Atividades concorrentes podem ocorrer, por exemplo, quando várias páginas (supostamente relacionadas) são

apresentadas simultaneamente em uma janela, ou em múltiplas janelas, ou ainda via diferentes mídias. Uma situação de sincronismo é representada por Stotts e Furuta [1989] na Rede de Petri ilustrada na Figura 5.3. Se os três lugares "s_7", "s_8" e "s_9" estão com fichas e a transição "t_7" for disparada, as fichas de cada lugar de entrada são removidas e colocadas no lugar de saída "s_2". Nesta abordagem, o sistema deve apresentar o conteúdo associado aos três lugares: um texto, uma imagem relacionada ao texto e também um áudio. A ativação da transição "t_7" deve causar o encerramento da apresentação dos três conteúdos. Na Figura 5.4 ilustra-se a especificação correspondente em Statechart.

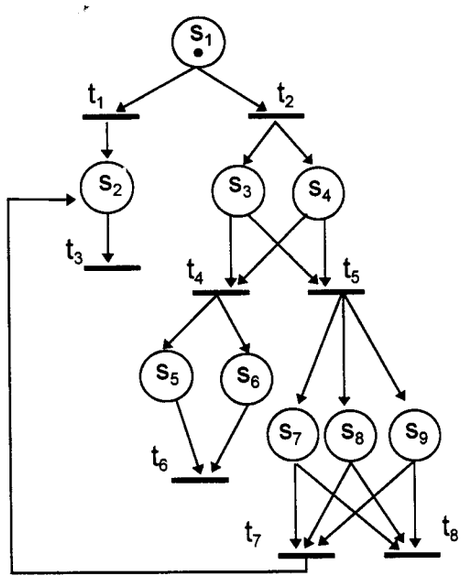


Figura 5.3: Rede de Petri (sem hierarquia) que descreve concorrência [Stotts; Furuta 1989, p.18].

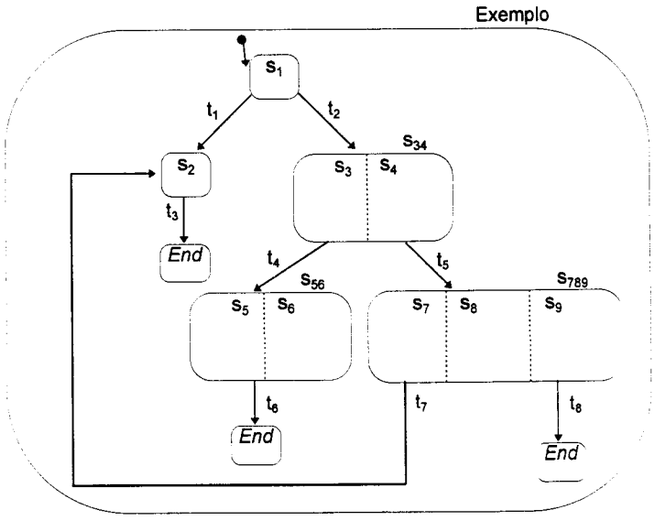


Figura 5.4: Especificação em Statechart equivalente à Rede de Petri da Figura 5.3.

Em Statechart, este tipo de atividade pode ser especificado por duas transições: uma transição pode ativar vários estados paralelos (decomposição AND), e uma segunda transição faz com que todos os estados paralelos sejam simultaneamente abandonados. Na Figura 5.4, os três estados AND "s_{34}", "s_{56}" e "s_{789}" contêm múltiplos subestados associados a páginas a serem apresentadas concorrentemente. Os estados de nome "End" indicam estados finais e têm uma página nula associada. A ativação do evento "t_4", na configuração de estados $SC_1=\{s_3,s_4\}$, causará a desativação dos estados "s_3" e "s_4", conduzindo a uma nova configuração $SC_2=\{s_5,s_6\}$. Por outro lado, se o evento "t_5" for habilitado no lugar de "t_4", os estados "s_3" e "s_4" serão abandonados e a nova configuração será $SC_3=\{s_7,s_8,s_9\}$. De acordo com a semântica de Statecharts, as informações das três páginas da configuração de contexto $\{P_{S_7},P_{S_8},P_{S_9}\}$ associada à configuração SC_3 tornar-se-ão visíveis. As janelas lógicas das três páginas serão, então, removidas da apresentação quando as âncoras associadas aos eventos "t_7" ou "t_8" forem ativadas. Assim, os estados do tipo AND fornecem um mecanismo natural para especificar, explicitamente, o paralelismo inerente ao hiperdocumento, o que não ocorre com o formalismo Redes de Petri.

5.3.3. Controle de Acesso

Os conjuntos de condições (C) e de expressões (V) definidos pelo Statechart podem ser usados para impor restrições aos leitores durante a navegação. Adicionalmente, permitem que diferentes caminhos de navegação compartilhem uma ou mais páginas, mantendo seções mutuamente exclusivas, superando as limitações de grafos direcionados.

Nesta seção ilustra-se como especificar mecanismos de controle de acesso no HMBS utilizando variáveis do Statechart. Para exemplificar, adota-se o exemplo apresentado por Stotts e Furuta [1989], o qual é utilizado como um hiperdocumento hipotético que permite gerenciar informações dos registros dos funcionários em uma empresa. O hiperdocumento pode ser acessado por duas classes de leitores: uma classe privilegiada (PR) e uma classe restrita (RR). Leitores da classe "PR" podem navegar pelo hiperdocumento inteiro, mas leitores da classe "RR" não podem visualizar as seções contendo avaliações de desempenho dos funcionários. Uma especificação em Statechart do hiperdocumento que trata deste cenário é

apresentada na Figura 5.5. As páginas associadas aos estados "s_3", "s_4" e "s_5" contêm as avaliações dos funcionários e são exibidas concorrentemente. As páginas associadas aos estados "s_{12}" e "s_{13}" também contêm informações privilegiadas e têm seus conteúdos exibidos simultaneamente. As outras informações dos registros dos funcionários (páginas associadas aos estados "s_2", "s_6" até "s_9", e "s_{11}") contêm informações de acesso irrestrito.

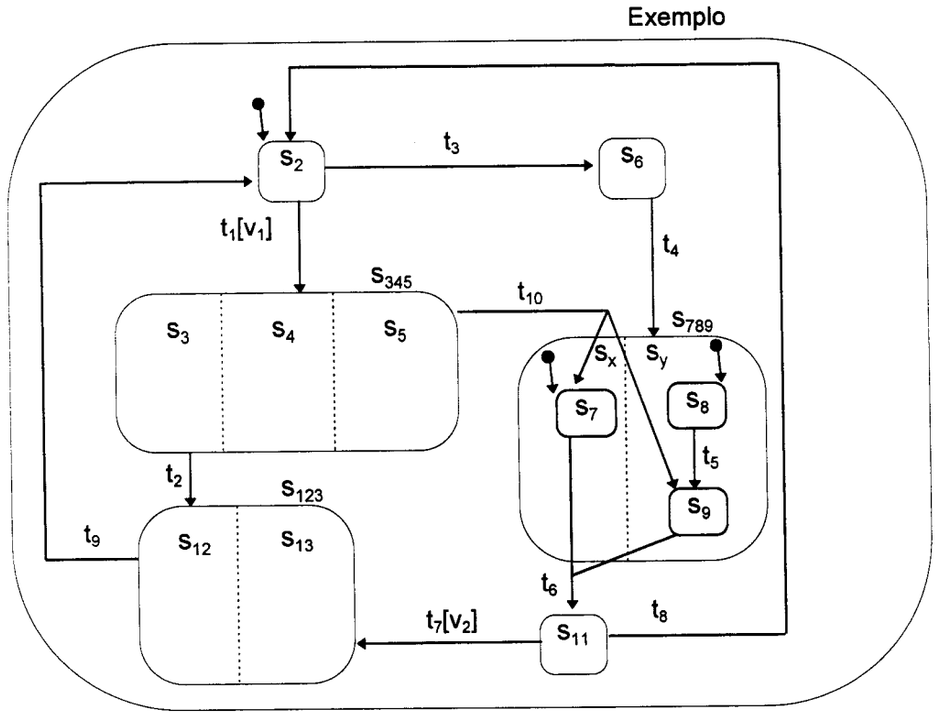


Figura 5.5: Statechart com controle de acesso usando variáveis.

Na especificação, o controle de acesso às páginas restritas é governado pelas duas variáveis lógicas "v_1" e "v_2". Para implementar um mecanismo equivalente, Stotts e Furuta [1989] valem-se de lugares especiais de controle de acesso. Segundo a semântica dos Statecharts, uma transição rotulada pela expressão "$t_1[v_1]$" é ativada somente se o evento "t_1" for ativado e a condição "v_1" for verdadeira. Desse modo, para a classe de usuários "PR" os valores iniciais de "v_1" e "v_2" são verdadeiros e, por conseguinte, o acesso às páginas restritas é habilitado. Para a classe de leitores "RR", as variáveis lógicas "v_1" e "v_2" recebem valores iniciais falsos, impedindo a visualização das páginas "P_{S3}", "P_{S4}", "P_{S5}", "P_{S12}" e "P_{S13}". Assim, as âncoras associadas aos eventos "t_1" e "t_7" não serão habilitadas para tais usuários e, conseqüentemente, as transições rotuladas pelos eventos "t_2", "t_9" e "t_{10}" nunca serão ativadas. Para verificar as propriedades do

hiperdocumento quando se utilizam variáveis de controle de acesso é necessário gerar várias árvores de alcançabilidade, uma para cada conjunto inicial de variáveis.

Uma terceira classe de leitor poderia compreender os semi-restritos, que podem navegar pelas páginas com informações salariais associadas aos estados "s_{12}" e "s_{13}", mas não podem navegar pela avaliação de desempenho contida nas páginas "P_{S3}", "P_{S4}" e "P_{S5}". Essa nova classe é criada atribuindo o valor verdade para a variável "v_2" e falso para "v_1". Examinando a árvore de alcançabilidade desse hiperdocumento, nota-se que um leitor pode selecionar a âncora associada ao evento "t_7", mas a ligação associada à transição rotulada pelo evento "t_1" nunca será habilitada e a configuração de contexto $\{P_{S3},P_{S4},P_{S5}\}$ nunca será apresentada. Caso o autor precise modificar o valor das variáveis durante a seção de navegação pode-se utilizar as ações, que proporcionam uma navegação adaptativa e, conseqüentemente, personalizada para o leitor.

Finalizando, para construir hiperdocumentos com classes de controle de acesso, o autor deve realizar as seguintes tarefas: identificar as classes de controle de acesso, decidir quais páginas associadas a estados são restritas para cada classe; associar uma variável de controle para cada classe e, finalmente, definir os valores iniciais das variáveis de cada classe.

5.4. CONSIDERAÇÕES FINAIS

Este capítulo apresentou algumas propriedades que podem ser analisadas pelos autores com o objetivo de identificar inconsistências estruturais e problemas relacionados à navegação. A maioria das análises é baseada na árvore de alcançabilidade de Statecharts, e as seguintes propriedades dinâmicas de Statecharts foram definidas e introduzidas no contexto de hiperdocumentos: alcançabilidade de uma página a partir de uma configuração de contexto qualquer, reiniciabilidade, *deadlock* durante a navegação, vivacidade de ligações navegacionais e seqüência válida de âncoras.

Além dessas propriedades, neste capítulo foram apresentados recursos adicionais que podem ser utilizados para identificar problemas relacionados à visualização e navegação, tais como: características físicas de visualização das janelas lógicas pelo sistema hiperdocumento, representação de concorrência na apresentação de múltiplas páginas e controle de acesso a leitores do hiperdocumento.

Desse modo, objetivou-se mostrar, neste capítulo, que a rica base formal matemática inerente aos Statecharts pode ser utilizada pelo modelo HMBS para apoiar a atividade de autoria, explorando diferentes maneiras de melhorar a organização estrutural e navegacional do hiperdocumento, encorajando o desenvolvimento de hiperdocumentos estruturados.

Para validar o modelo HMBS e mostrar como as características discutidas neste capítulo poderiam ser implementadas foi desenvolvido um protótipo de sistema hiperdocumento denominado HySCharts, que será apresentado no próximo capítulo.

Capítulo 6

HySCharts: PROTÓTIPO DE UM SISTEMA HIPERDOCUMENTO BASEADO NO HMBS

6.1. CONSIDERAÇÕES INICIAIS

Um hiperdocumento especificado em HMBS pode ser disponível para navegação de duas maneiras: suporte traduzido e suporte interpretado. Neste capítulo essas abordagens são definidas, com ênfase no suporte interpretado, adotado nesta tese para validar o HMBS e ilustrar um ambiente automatizado que apóia o processo de especificação de hiperdocumentos em HMBS.

Para viabilizar a geração de especificações em HMBS com base no suporte interpretado foi desenvolvido um protótipo de sistema de autoria e navegação de hiperdocumentos denominado HySCharts (***Hy****perdocument* ***S****ystem based on state****Charts***). Este capítulo tem por objetivo apresentar as características e a funcionalidade do sistema HySCharts, e está organizado da seguinte forma. Na Seção 6.2 definem-se as características do suporte traduzido e do suporte interpretado. Também é apresentada uma descrição geral do sistema HySCharts, focalizando os principais aspectos do suporte interpretado. Na Seção 6.3 é

apresentada a arquitetura básica do sistema, explicitando o seu aspecto metodológico de desenvolvimento, os principais módulos funcionais (de autoria e de navegação) e os seus inter-relacionamentos, além de detalhes de implementação dos objetos navegacionais e de apresentação do modelo HMBS.

No decorrer deste capítulo é utilizado como exemplo um hiperdocumento que apresenta o Parque Ecológico de São Carlos (PESC), o qual é discutido mais detalhadamente no Capítulo 7 (a estrutura organizacional em HMBS desse mesmo hiperdocumento já foi apresentada nos capítulos anteriores). As considerações finais deste capítulo são apresentadas na Seção 6.4.

6.2. SUPORTE TRADUZIDO X SUPORTE INTERPRETADO

Na abordagem de suporte traduzido os objetos HMBS da especificação do hiperdocumento devem ser transformados em objetos de algum software de apresentação ou sistema hiperdocumento existente, como o *Toolbook*™, o *MacroMedia Director*® ou *browsers* HTML. As primitivas especificadas no projeto devem ser mapeadas em artefatos de implementação e todos os objetos do modelo (páginas, ligações, âncoras e canais) devem ser instanciados. Para realizar tal tradução, regras devem ser elaboradas para guiar o processo de conversão de padrões de especificação em código fonte de alguma linguagem que possa ser compilada ou interpretada pelos aplicativos apropriados para a apresentação de hiperdocumentos.

A equipe de desenvolvimento pode enfrentar problemas no suporte traduzido devido à grande variedade e heterogeneidade dos ambientes de implementação existentes. Por exemplo, a operação "Mostra_VisãoH" não seria facilmente implementada com os recursos oferecidos pela linguagem HTML ou pelo *Toolbook*. Além disso, não é possível usar um mesmo conjunto de regras na tradução de uma especificação HMBS para um documento HTML e para um programa do *MacroMedia Director*®, por exemplo. Mais ainda, a implementação da estrutura hierárquica e decisões de projeto, como preservar ou não a semântica de navegação definida pelo HMBS, podem dificultar o processo de tradução. Nesta tese essa abordagem não foi explorada.

A outra alternativa, o suporte interpretado, requer um sistema capaz de criar, interpretar e executar especificações de hiperdocumentos em HMBS, permitindo ao autor definir a estrutura organizacional e navegacional de hiperdocumentos, além de verificar sua validade pela execução interativa. Essa foi a abordagem adotada

nesta tese, tendo sido desenvolvido o protótipo de um sistema mono-usuário, denominado HySCharts, para apoiar a especificação formal e a execução de hiperdocumentos modelados em HMBS.

O sistema engloba duas ferramentas: uma de autoria e uma de navegação. A primeira é apoiada nos conceitos e propriedades do modelo, que permite ao projetista definir o comportamento navegacional do hiperdocumento de forma automatizada. A segunda permite executar as especificações, apresentando as informações para o leitor em um ambiente que usa a semântica de navegação proposta pelo modelo. O ambiente inclui um *browser* que permite ao leitor explorar a estrutura navegacional do hiperdocumento e sua configuração de contexto em um estágio qualquer da navegação. O *browser* também permite ao autor visualizar, a priori, o comportamento dinâmico do hiperdocumento. Adicionalmente, o sistema permite que algumas das propriedades discutidas no Capítulo 5 sejam analisadas pelo autor, podendo apoiá-lo na tarefa de inspecionar se a especificação proposta corresponde a um hiperdocumento estruturado e se satisfaz aos requisitos definidos pelos leitores e pelo próprio autor.

O protótipo HySCharts foi desenvolvido como uma extensão do sistema StatSim (*Statechart Simulator*), um ambiente gráfico automatizado cujo principal objetivo é apoiar a validação de especificações em Statechart. O StatSim foi desenvolvido como um protótipo ao longo de cerca de cinco anos (1989-1993) pelo grupo de Engenharia de Software do ICMC-USP[18], envolvendo diversos projetos de mestrado e de iniciação científica. Para a sua implementação foram adotadas a linguagem de programação C e o *toolkit* X-View. Descrições sobre o projeto podem ser encontradas em várias dissertações, relatórios e artigos [Fortes 1991, Batista 1991, Masiero et al. 1991, Cangussu 1993, Penteado et al. 1995]. Assim, o HySCharts dispõe de uma máquina de Statecharts (reutilizada do ambiente StatSim) que executa, de forma interpretada, uma especificação descrita em Statecharts. O *browser* do HySCharts recebe os eventos externos (ações do leitor) e os interpreta, modificando a apresentação do hiperdocumento.

A especificação do modelo HMBS foi implementada e acoplada ao ambiente Statsim utilizando uma abordagem orientada a objetos. Como plataforma de programação adotou-se estações de trabalho *Sun* rodando *Solaris*, a linguagem C++ e o *toolkit* X-View, que permite implementar interfaces gráficas em ambiente

Unix. A versão atual do sistema, operacional desde Novembro de 1997, possui aproximadamente 55.000 linhas de código, das quais cerca de 30.000 correspondem ao ambiente StatSim.

6.3. ARQUITETURA DO SISTEMA HySCharts

A arquitetura do sistema HySCharts é composta por três camadas principais, denominadas camadas de aplicação, de estruturação e de armazenamento, respectivamente (Figura 6.1). A camada de aplicação permite ao autor criar a estrutura do hiperdocumento e, ao leitor, visualizar e navegar pelo seu conteúdo. Essa camada está subdividida nos módulos de autoria e de navegação, em adição a um ambiente de edição e simulação de Statecharts. A camada de estruturação é considerada o núcleo do sistema, pois compreende a estrutura interna subjacente à formalização do hiperdocumento em HMBS como base nos objetos estruturais, navegacionais e de apresentação. Todos esses objetos são armazenados em bases de dados gerenciadas por funções da camada de armazenamento.

Nesta seção apresentam-se o modelo de objetos do módulo de autoria e de navegação, bem como as características e as ferramentas utilizadas em cada um deles. Também são apresentados a estrutura e os objetos armazenáveis em cada uma das três camadas da arquitetura do HySCharts, a interface entre elas e como os objetos foram implementados.

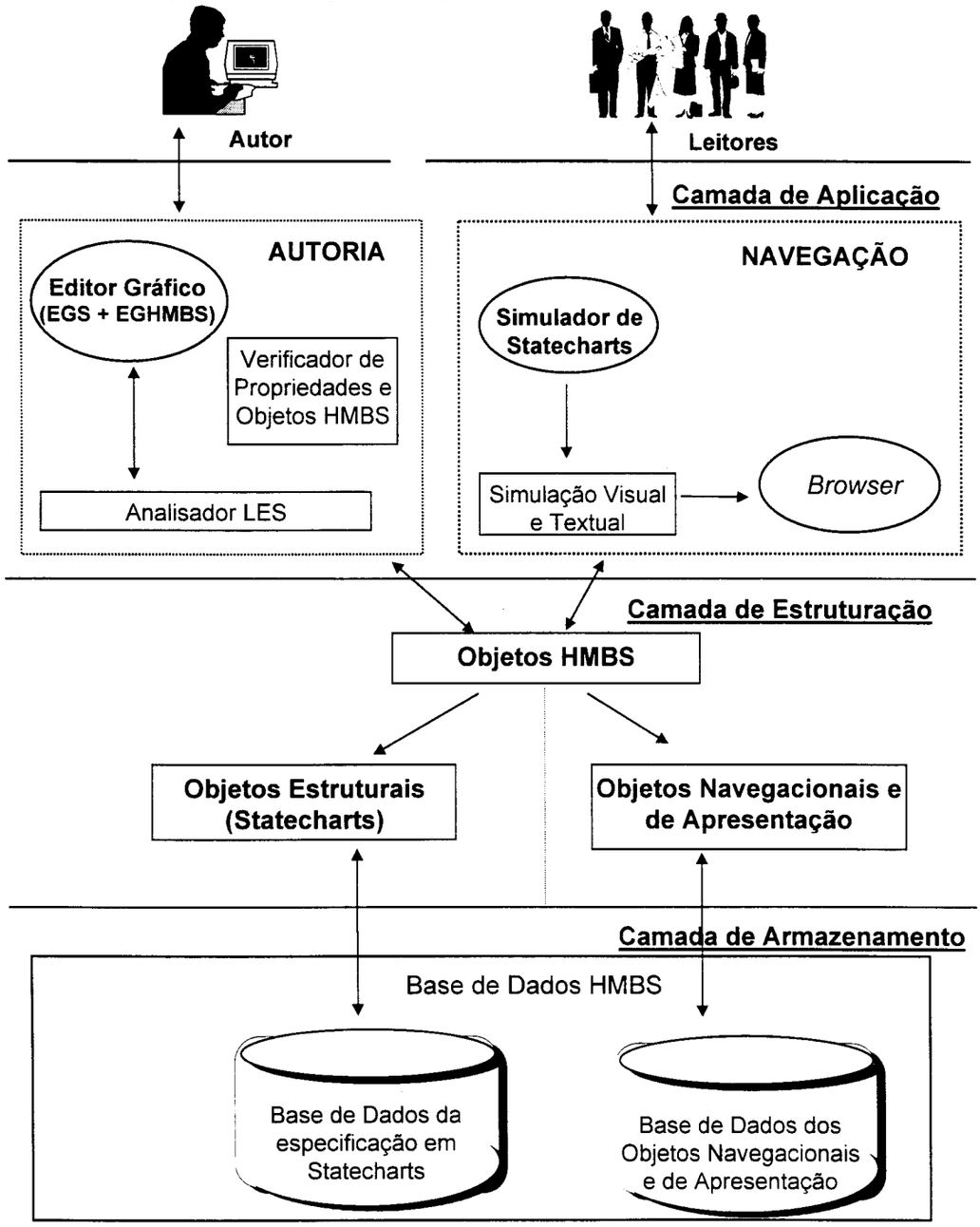
6.3.1. Especificação do Sistema

Nesta seção é apresentada uma visão geral da abordagem utilizada para implementar o sistema HySCharts. Os modelos de objetos relativos aos módulos de autoria e de navegação são ilustrados e foram desenvolvidos utilizando o método de análise e projeto orientado a objetos *Fusion* [Coleman et al. 1994]. Turine et al. [1997b] descrevem a especificação completa desses dois módulos.

O modelo de objetos apresentado na Figura 6.2 especifica o módulo de autoria de hiperdocumentos e inclui os principais relacionamentos entre o autor e o sistema para definir os objetos HMBS. Inicialmente, o autor deve criar a estrutura organizacional do hiperdocumento em Statecharts. Os objetos responsáveis pela criação e manipulação de Statecharts foram reutilizados do modelo de objetos do ambiente StatSim, recuperado por um processo de engenharia reversa com base

[18] Instituto de Ciências Matemáticas e de Computação, USP, São Carlos.

no método *Fusion*, como descrito por Penteado [1996]. Uma versão simplificada desse modelo é ilustrada pela classe "Máquina_Statecharts", que oferece ferramentas para edição (modelo de objetos de autoria - Figura 6.2) e simulação (modelo de objetos de navegação - Figura 6.3) de Statecharts.



Legendas:

LES - Linguagem de Especificação de Statecharts

EGS - Editor Gráfico de Statecharts

EGHMBS - Editor Gráfico dos Objetos HMBS

Figura 6.1: Arquitetura do sistema HySCharts.

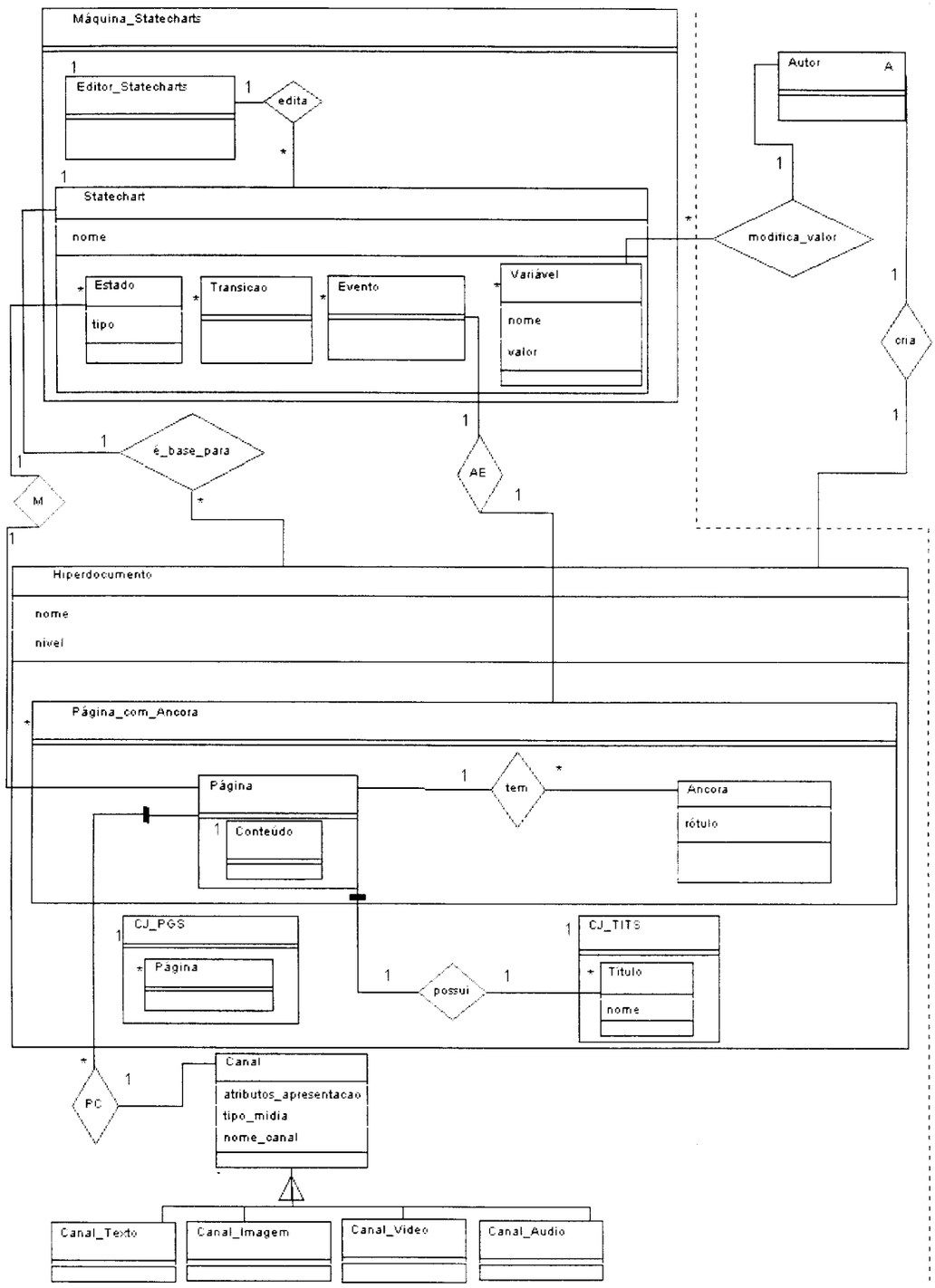


Figura 6.2: Modelo de objetos do módulo de autoria do sistema HySCharts.

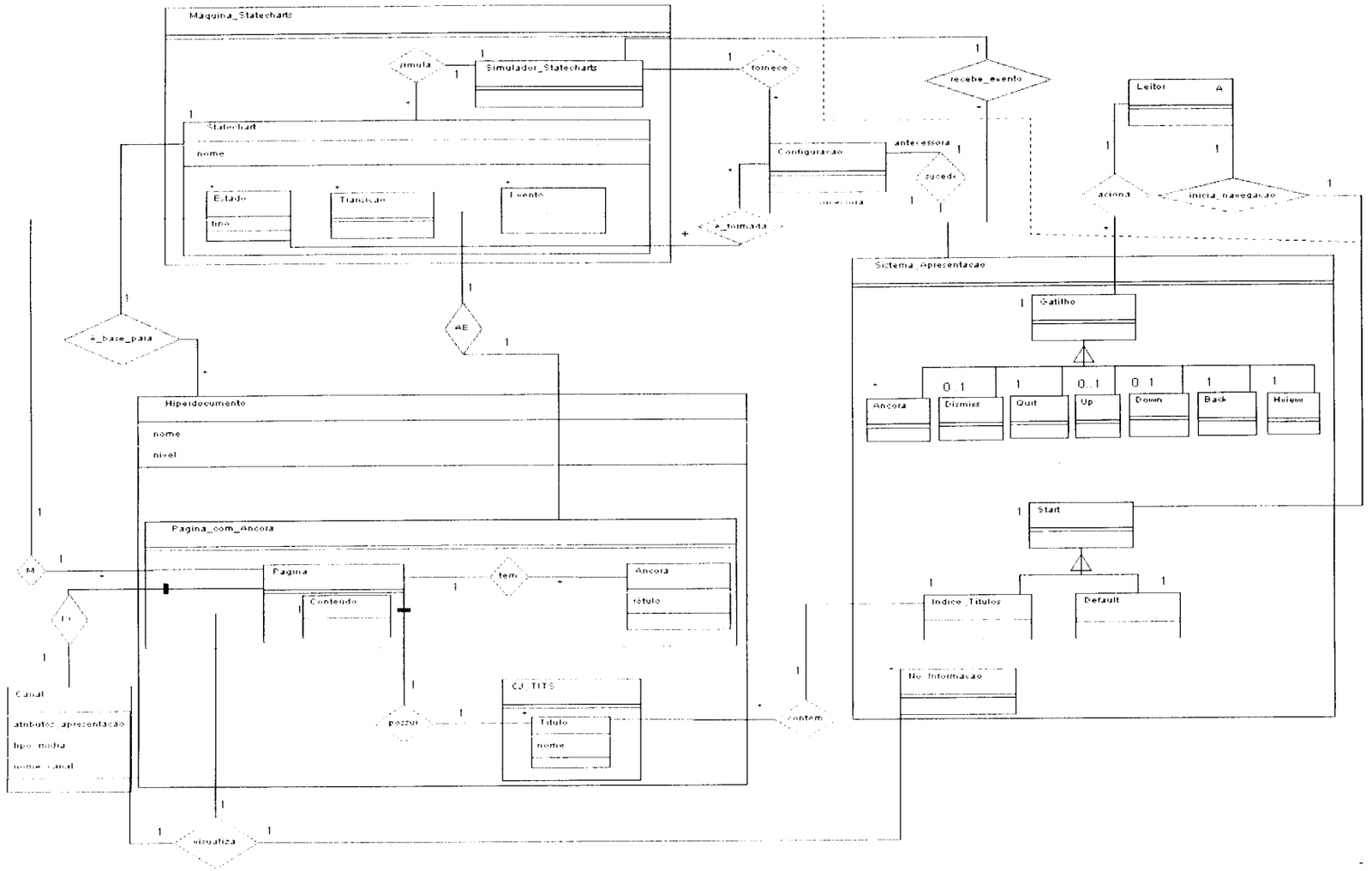


Figura 6.3: Modelo de objetos do módulo de navegação do sistema HySCharts.

Criada a estrutura organizacional, o autor deve associá-la ao hiperdocumento, como indicado no relacionamento "é_base_para". Em seguida, deve criar os objetos de apresentação (canais), os objetos navegacionais (páginas, âncoras, ligações) e realizar as associações pertinentes entre os objetos, conforme definido no modelo HMBS. Por exemplo, o relacionamento "M" da Figura 6.2 permite associar estados a páginas de informação, e o "AE" permite associar uma âncora pertencente ao conteúdo de uma página a um evento do Statechart. Esses dois relacionamentos correspondem, respectivamente, às funções "m" e "ae" da modelo HMBS.

O modelo de objetos da Figura 6.3 especifica as classes e os relacionamentos do módulo de navegação do sistema HySCharts. Inicialmente, o leitor inicia a navegação (relacionamento "inicia_navegação") no hiperdocumento a partir de um índice (classe "Índice_Títulos") formado pelos títulos das páginas ou pela configuração de estados inicial (classe "*Default*") do Statechart subjacente. O mecanismo de apresentação das unidades de informação ou conteúdo das páginas (classe "Nó_Informação") é definido pela classe "Sistema_Apresentação", cuja principal característica é receber os eventos gerados pelo leitor (classe "Gatilho"), disparar o simulador de Statecharts e visualizar o conteúdo das páginas, invocando os canais de apresentação (classe "Canal") correspondentes.

6.3.2. Módulo de Autoria

As três funções básicas do módulo de autoria são descritas nas subseções seguintes: especificação de objetos estruturais, especificação de objetos navegacionais e de apresentação, e análise da estrutura navegacional.

6.3.2.1. Especificação de Objetos Estruturais

Os objetos estruturais são os estados, as transições e os eventos que formam o Statechart subjacente ao hiperdocumento especificado em HMBS. O Statechart é criado graficamente com base em um editor gráfico (EGS - Editor Gráfico de Statecharts). Para validar a especificação, o sistema converte a representação gráfica em descrições de uma linguagem interna denominada LES (Linguagem de Especificação de Statecharts) e tais descrições são passadas por um analisador (Analisador LES). Se erros não são detectados, cria-se uma estrutura de dados na camada de estruturação contendo a especificação em termos dos objetos estruturais. Deve-se ressaltar que a estrutura dos objetos

estruturais foi reutilizada do ambiente StatSim e sua implementação não é comentada nesta tese. Uma descrição de sua estrutura de dados e da gramática que especifica a LES é apresentada por Fortes e Masiero [1991].

Na Figura 6.4 é ilustrada a janela do EGS, que contém ferramentas que permitem ao autor criar e manipular os objetos estruturais e as variáveis do Statechart. A janela com título "QUERY STATE: Animal", sobreposta à janela principal, apresenta informações estruturais sobre o estado "Animal", um estado do tipo OR cuja decomposição é especificada em um outro nível hierárquico. Para editar e visualizar a representação em Statechart do estado decomposto em outro nível utiliza-se a ferramenta de decomposição de estados que ativa uma nova janela (seleção do botão *Decompose* do menu *States*). A Figura 6.5 apresenta a janela que contém o contexto de decomposição do estado "Animal". Algumas modificações foram introduzidas no EGS original, reutilizado do StatSim, para possibilitar a interação com os módulos que tratam da especificação de objetos navegacionais e de apresentação do HMBS.

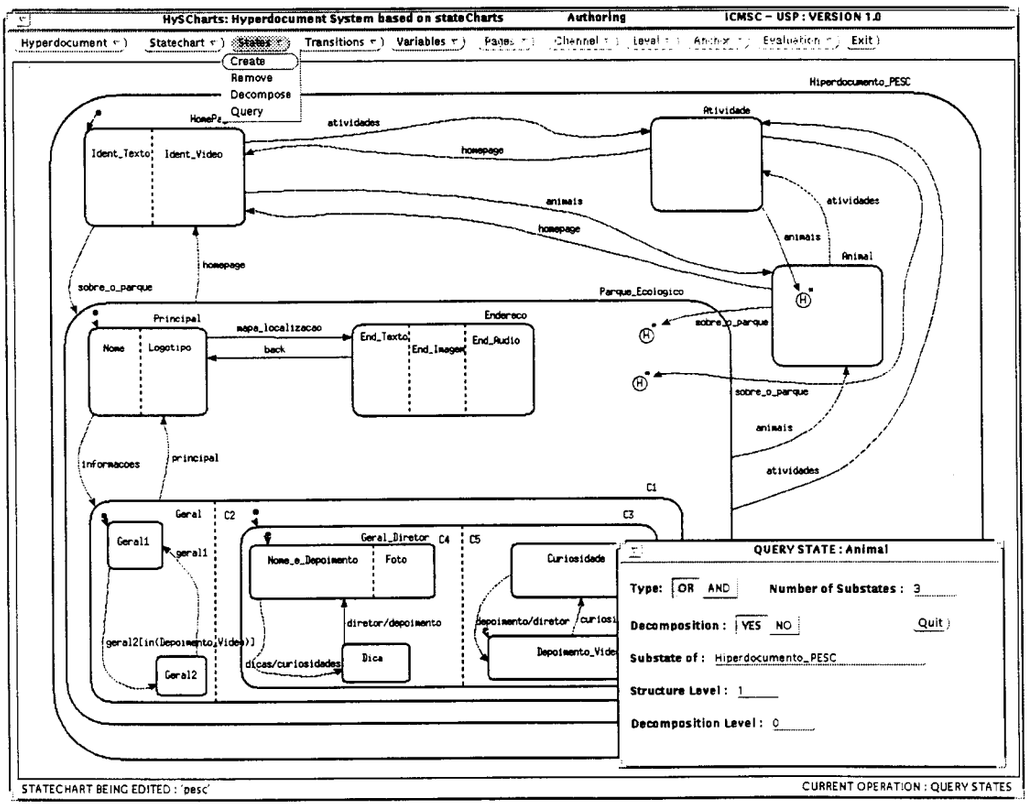


Figura 6.4: EGS do módulo de autoria do sistema HySCharts.

6.3.2.2. Especificação de Objetos Navegacionais e de Apresentação

A estrutura organizacional do hiperdocumento em Statechart é utilizada como base para a criação dos objetos navegacionais (páginas, ligações, âncoras) e dos objetos de apresentação (canais). O autor, usando o editor gráfico do modelo HMBS, denominado EGHMBS, cria todos os objetos navegacionais e de apresentação, e associa-os aos objetos estruturais correspondentes. Em nível de implementação, os objetos HMBS foram definidos como classes de objetos, segundo a sintaxe da linguagem C++. Os detalhes de implementação dessas classes são ilustrados a seguir. Nas Figuras 6.6 a 6.10 são ilustrados os leiautes de algumas janelas do HySCharts que permitem ao autor gerenciar cada um desses objetos.

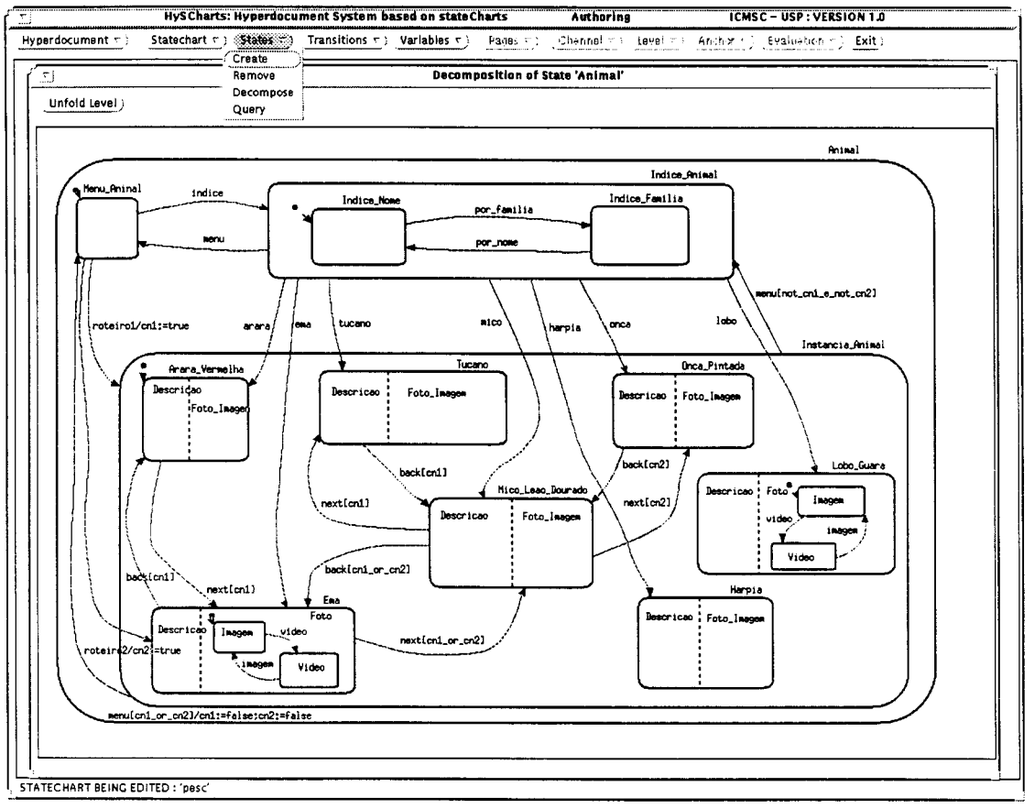


Figura 6.5: Janela de decomposição do estado "Animal".

O Quadro I contém a descrição da classe "Página" que implementa o objeto página segundo o HMBS. Os atributos privados "nome_arq", "t" e "LAnc" representam, respectivamente, o nome do arquivo que contém o conteúdo da página, o título da página e a lista de âncoras associadas à página.

A Figura 6.6 apresenta três janelas relacionadas aos objetos página, com

títulos "*Create Page*", "*Query Page of the State Ident_Texto*" e "*Pages Created*". A janela "Create Page" permite especificar uma página pelos atributos título, conteúdo, tipo da mídia e canal de apresentação que a reconhece. O conteúdo da página é identificado pelo nome de um arquivo ("File"). Se tal conteúdo consistir de informação descrita pela mídia do tipo texto, o autor pode invocar um editor de textos pela opção "Edit". Para outras mídias, o conteúdo do arquivo deve ser criado em aplicativos externos ao sistema. Para interpretar e visualizar o conteúdo de uma página é necessário associá-la a um único canal de apresentação, também especificado pelo autor. Basicamente, os canais definem o tipo de mídia associado à página e características de visualização. Os tipos de mídia válidos no sistema são: texto, imagem, áudio e vídeo. Assim, na janela "Create Page" o autor especifica o nome do canal e, como conseqüência, o sistema habilita a opção "Media Type" correspondente. Após a especificação dessas informações, o autor pode associar tal página ao objeto estrutural (estado) correspondente, criando uma instância do mapeamento "m" entre estados e páginas.

Quadro I: Implementação da classe "Página".

```
class Pagina
{
    private:

                    // nome do arquivo que contém o conteúdo "c" da Página.
            char nome_arq[MAXNOME];
                    // título "t" da Página.
            class Titulo* t;
                    // coleção de âncoras associadas a Página.
            class Lista * LAnc;
    public:

            Pagina (char*);    // Construtor da classe.
            ~Pagina ( );       // Destrutor da classe.
                    // retorna um ponteiro para o título da Página.
            Titulo* Retornar_Titulo(void);
                    // retorna o nome do arquivo da Página.
            char* Retornar_Nome_Arq( );
                    // atualiza o nome e o título da Página.
            void Atualizar_Nome_Arq(char*);
            void Atualizar_Titulo(Titulo*);
}
```

A Figura 6.6 também ilustra a operação de consulta a páginas associadas a estados. Ao selecionar um estado, caso este tenha uma página associada, as informações relativas à página e ao estado são apresentadas. Por exemplo, na janela "*Query Page of the State Ident_Texto*" são apresentadas as informações relativas ao estado "Ident_Texto" subestado do estado "HomePage" (tipo de decomposição e número de subestados) e à página associada (título, nome do

canal, tipo da mídia). A janela "*Pages Created*" contém as informações (título, tipo da mídia e nome do estado associado) sobre todas as páginas criadas para o hiperdocumento especificado.

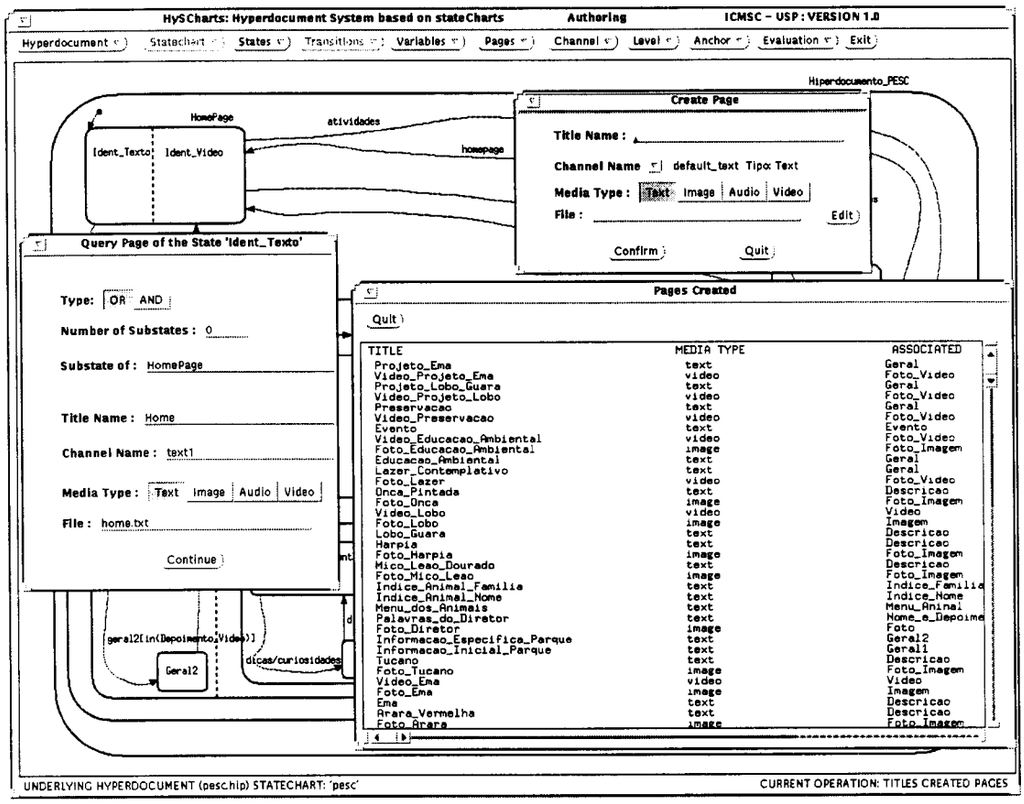


Figura 6.6: Janelas relacionadas ao gerenciamento dos objetos página.

Tendo criado as páginas e associado-as aos estados correspondentes, o autor pode especificar os objetos navegacionais âncora que habilitam as ligações do hiperdocumento. As âncoras pertencentes a páginas do tipo texto são indicadas no conteúdo "c" da página pelos marcadores de início "<<" e fim ">>" mas, durante a navegação, as âncoras são visualmente apresentadas sublinhadas e em negrito. Por exemplo, o conteúdo da página "Ident_Texto" (apresentado no Quadro II) possui três âncoras, rotuladas como "Sobre_o_Parque", "Atividades", e "Animais".

A classe "Âncora", apresentada no Quadro III, implementa os objetos âncora. Para criar um objeto âncora pertencente a uma página do tipo texto basta selecionar o estado associado e o canal de apresentação correspondente exibe o conteúdo de sua página em uma janela lógica. O autor escolhe a palavra desejada e, em seguida, seleciona o evento associado pertencente ao rótulo de uma transição. Dessa maneira, uma instância do relacionamento "AE" (ver modelo de objetos da autoria - Figura 6.2) é criada e os objetos âncora e ligação são

definidos. Na Figura 6.7 o conteúdo da página texto "P_{Ident_Texto}" é exibido na janela com o título "*Viewer of the page with title Home and state Ident_Texto*".

Quadro II: Especificação das âncoras no conteúdo da página "Ident_Texto".

```
-------------------------------------------------------
        P A R Q U E   E C O L O G I C O
          "DR. ANTONIO T. VIANNA"
                SAO CARLOS - SP
-------------------------------------------------------

        Menu Inicial
        ----------------
        1- <<Sobre_o_Parque>>
        2- <<Atividades>> desenvolvidas no parque
        3- <<Animais>> abrigados no parque
```

Na Figura 6.7 é ilustrada, também, a janela "*Query Anchor-> Event*", que apresenta o resultado da operação de consulta sobre a âncora "Sobre_o_Parque" da página "P_{Ident_Texto}". Tal âncora está associada ao evento rotulado "sobre_o_parque", cuja transição tem como nome de estado origem "HomePage" e como destino "Parque_Ecológico".

Quadro III: Implementação da classe "Âncora".

```
class Ancora
{
    private:
                    // posção inicial e final da âncora no conteúdo da página.
            int ini, fim;
                    // rótulo ou nome da âncora.
            char rotulo[MAXNOME];
                    /* posição gráfica da âncora quando visualizada na janela lógica durante
                    o processo de navegação. */
            class Posicao_palavra *pos_palavra;
    public:
            Ancora(char*,int,int,int);      // construtor da classe.
            ~Ancora();              // destrutor da classe.
                                    // retorna as posições início e fim da âncora.
            int Retornar_Inicio( );
            int Retornar_Fim( );
                                    // retorna o rótulo da âncora.
            char* Retornar_Nome( );
                                    /* métodos para verificar se existe um objeto âncora
                                    especificado na posição clicada pelo leitor na janela lógica.*/
            int existe_palavra_na_posicao(int, int);
            void Atualizar_Posicao_Ancora(int, int, int, int);
            class Posicao_palavra* Retornar_Posicao_Ancora();
}
```

O sistema HySCharts não permite ao autor definir um trecho de um vídeo ou uma região de uma imagem como uma âncora. Para minimizar essa limitação do sistema, o autor pode criar âncoras associadas à página como um todo. Por

exemplo, de acordo com a Tabela 4.1, a página "P_{G11b}" corresponde à página "P_{FOTO}" da Figura 6.8 e está associada a um canal do tipo imagem. A janela de título "*State: Foto*" da Figura 6.8 mostra o conteúdo da página "P_{FOTO}", como a âncora rotulada "Dicas" é visualizada em tempo de navegação, e também como essa âncora é exibida na janela lógica (por um botão).

Na Figura 6.8 pode-se observar também a janela "Hyperdocument Level", na qual o autor define o nível de visibilidade "N" para o hiperdocumento. Nesse caso, o autor especificou que somente as páginas associadas aos estados básicos serão visíveis durante o processo de navegação (N = 0).

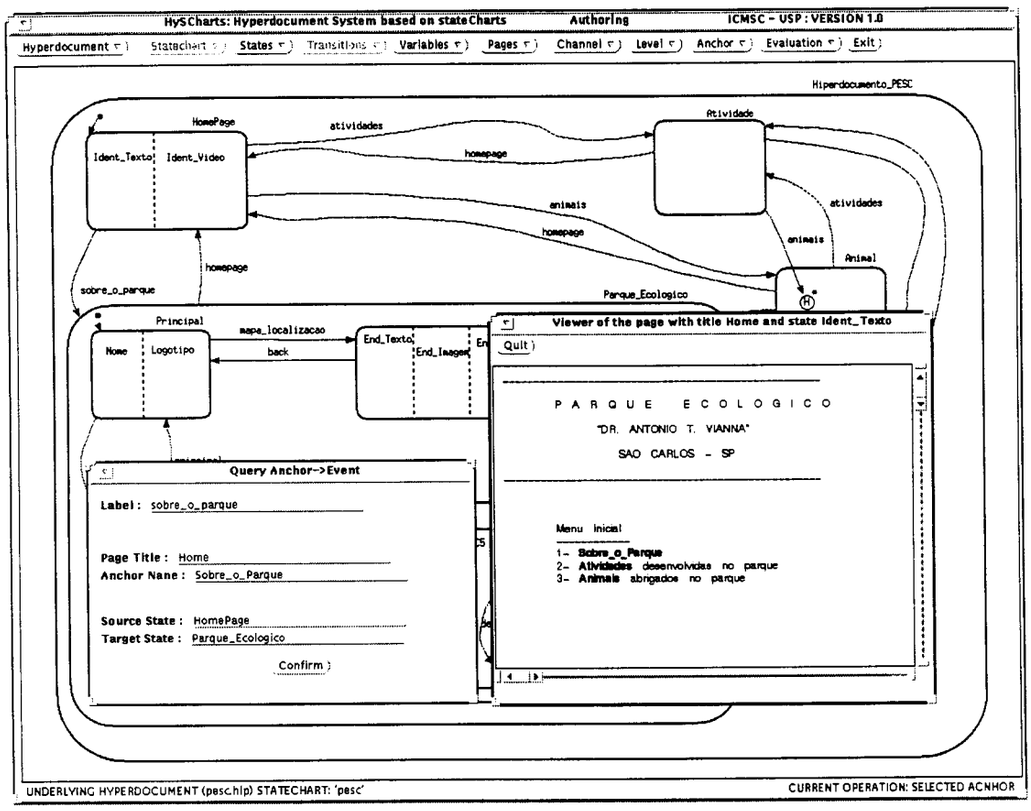


Figura 6.7: Janelas que gerenciam os objetos âncora.

A camada de estruturação armazena também o objeto canal, uma abstração de um dispositivo que inclui atributos de apresentação de alto nível e mecanismos de coordenação espacial das páginas. O sistema HySCharts trata esses elementos de especificação segundo uma abordagem orientada a objetos, no sentido de que existem classes de canais *default* que podem ser instanciadas, conforme as necessidades do autor, usando os mecanismos de herança e especialização/generalização. Em princípio, podem existir tantos canais de

apresentação quantos forem necessários para a implementação do hiperdocumento. Por exemplo, pode-se ter dois canais diferentes do tipo texto: um utilizando a fonte "Times New Roman" e o outro "Arial". Como no sistema CMIfed [Van Rossum et al. 1993], o uso de canais permite que um mesmo documento seja apresentado de diferentes maneiras, reespecificando os atributos de apresentação ao invés de modificar a estrutura navegacional.

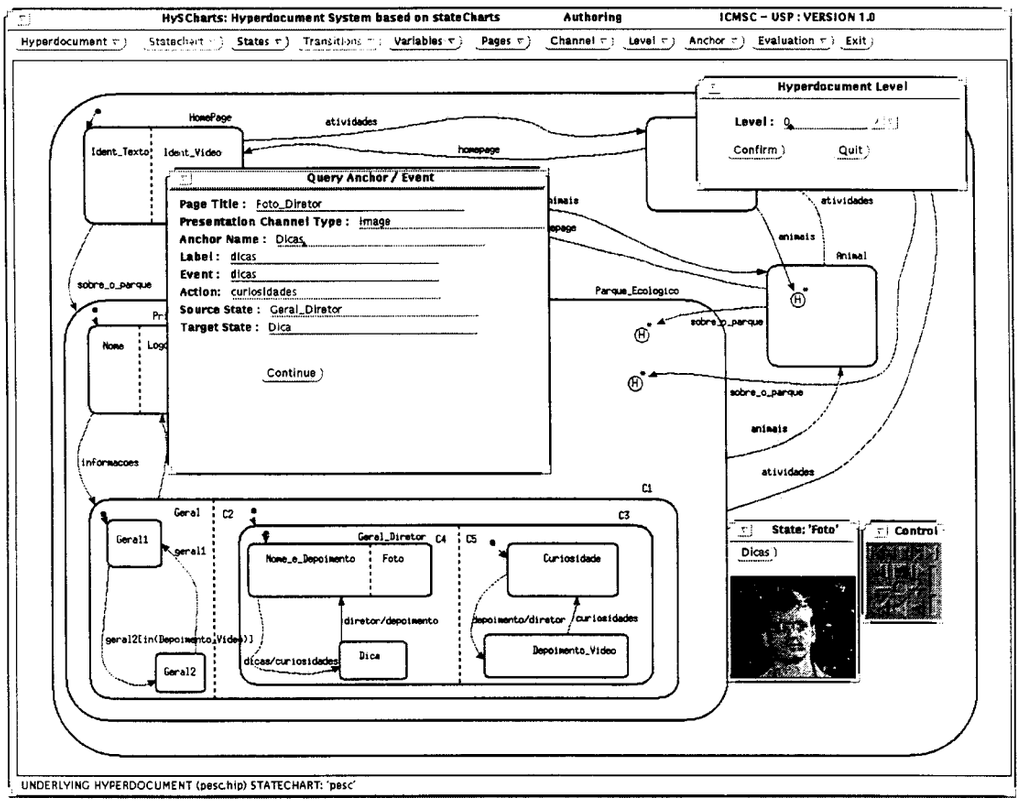


Figura 6.8: Janela para criação de âncoras associadas a páginas não textuais.

No HySCharts foram implementadas quatro classes de canais, uma para cada tipo de mídia (texto, imagem, áudio e vídeo). Os atributos genéricos dos canais estão implementados na classe de nome "Canal", apresentada no Quadro IV. Os atributos "posição_x", "posição_y", "largura" e "altura" da classe "Canal" definem a coordenação espacial da janela lógica criada pelo canal. Os atributos dependentes da mídia são especificados em suas classes específicas. Assim, além das características comuns herdadas da classe "Canal", os quatro canais podem possuir características adicionais ou mesmo redefinir certos atributos. A seguir, são apresentadas as características dos quatro canais específicos implementados no sistema.

Canal Texto

A classe "Canal_Texto" contém atributos específicos para a apresentação de informações textuais. A implementação da classe "Canal_Texto" é apresentada no Quadro V. Os atributos de apresentação de alto nível de um canal do tipo texto são, por exemplo, o estilo e o tamanho da fonte, e a cor de fundo das janelas lógicas.

Quadro IV: Implementação da classe "Canal".

```
class Canal
{
    private:
                    // nome do canal de apresentação.
            char nome_canal[MAXNOME];
                    // tipo da mídia do canal (vídeo, áudio, imagem ou texto).
            char tipo_midia[MAXNOME];
                    /* define a localização da janela lógica criada pelo canal com relação ao
                    dispositivo de apresentação. Trata-se da definição de um retângulo
                    (<posição_x, posição_y>;< posição_x + largura, posição_y + altura>). */
            int posicao_x, posicao_y, largura, altura;
                    /* posição do título de uma página na janela lógica (direita, esquerda,
                    topo, base, invisível). */
            int posicao_titulo;

    public:
            Canal(char *, char *, int, int, int, int, int);     // construtor da classe.
            ~Canal();                                           // destrutor da classe.
                            // retorna o nome do canal.
            char * Retornar_Nome( );
                            // retorna o tipo da mídia do canal.
            char * Retornar_Tipo();
                            // retorna a posição da janela lógica no dispositivo de saída.
            int Retornar_Posicao_X( );
            int Retornar_Posicao_Y( );
            int Retornar_Largura( );
            int Retornar_Altura( );
                            // retorna a posição do título na janela lógica.
            int Retornar_Posicao_Titulo( );
}
```

Canal Imagem

A classe "Canal_Imagem" contém atributos para a interpretação de informações do tipo imagem, os quais correspondem aos mesmos atributos da classe "Canal" para a apresentação de informações (Quadro IV).

Canal Vídeo

A classe "Canal_Vídeo" contém atributos para a interpretação de informações do tipo vídeo (Quadro VI). O atributo "status_video" permite especificar se o canal está ativo em tempo de execução, definindo, assim, uma apresentação com instâncias particulares.

Canal Áudio

A classe "Canal_Áudio" contém atributos globais para a apresentação de

áudio. As características particulares dessa classe são apresentadas no Quadro VII.

Quadro V: Implementação da classe "Canal_Texto".

```
class Canal_Texto : public Canal
{
    private:
                    /*define a fonte (Roman, Arial, Courier, Times New Roman, etc ) e o
                    estilo para o texto apresentado por este canal em particular. */
          int nome_fonte;
          int estilo;
          int tamanho_fonte; // define o tamanho do fonte de letra.
          int foreground; // define as cores de fundo e de letra para o texto.
          int background;
                    /* define a existência de barras de rolagem para o texto, tanto verticais (a
                    esquerda ou a direita), quanto horizontais (acima ou abaixo do texto). */
          int scroll_base;
          int scroll_direita;
          int resize;

    public:
                    // construtor da classe.
          Canal_Texto(char *, char *, int, int, int, int, int, int, int, int, int, int, int, int);
          ~Canal_Texto ();                    // destrutor da classe.
          int Retornar_Nome_Fonte( );  // retorna o nome e estilo da fonte.
          int Retornar_Estilo_Fonte();
                    // retorna o tamanho da fonte.
          int Retornar_Tamanho_Fonte( );
                    // retorna as cores de fundo e de letra para o texto.
          int Retornar_Foreground();
          int Retornar_Background();
                    // retorna a existência ou não de barras de rolagem.
          int Retornar_Scroll_Base();
          int Retornar_Scroll_Direita();
          int Retornar_Resize();
}
```

Quadro VI: Implementação da classe "Canal_Video".

```
class Canal_Video : public Canal
{
    private:
                    // define o status do canal, ou seja, se está ativo ou não.
          int status_video;
                    // exibir o controlador de dados multimídia (Stop, Pause, etc).
          int exibir_controlador;
          int porta_de_saida;
                    // apresentação do vídeo termina ou não depois do último quadro.
          int apresentacao_ciclica;

    public:
                    // construtor da classe.
          Canal_Video(char *, char *, int, int, int, int, int, int, int, int);
          ~Canal_Video();                // destrutor da classe.
                    // retorna status do vídeo e os atributos privados da classe.
          int Retornar_Status_Video( );
          int Retornar_Exibir_Controlador( );
          int Retornar_Forma_Apresentacao( );
          int Retornar_Porta_de_Saida( );
}
```

Quadro VII: Implementação da classe "Canal_Áudio".

```
class Canal_Audio : public Canal
{
        private:
            int status_audio;
            int exibir_controlador;
            int porta_de_saida;
            int apresentacao_ciclica;
        public:
            Canal_Audio(char *, char *, int, int, int, int, int, int, int, int, int);
            ~Canal_Audio ();
            int Retornar_Status_Audio( );
            int Retornar_Exibir_Controlador( );
            int Retornar_Forma_Apresentacao( );
            int Retornar_Porta_de_Saida( );
}
```

Para manipular os canais de apresentação, o sistema HySCharts possui ferramentas para criação, modificação e eliminação (disponível no menu "Channel"). Na Figura 6.9 é apresentada a janela com título "*Create Channel*", que apresenta os atributos de apresentação personalizados para a mídia texto, selecionada pelo autor na opção "Media Type". Para modificar as informações de um canal, o autor deve selecionar o canal desejado e modificar os atributos necessários (janela de título "*Modify Channel*" da Figura 6.9).

A camada de armazenamento é responsável pela representação e armazenamento de todos os objetos estruturais, navegacionais e de apresentação do HMBS utilizados na especificação do hiperdocumento. Essa camada não foi extensivamente explorada pelo sistema HySCharts. Por exemplo, para armazenar a estrutura interna do conteúdo das páginas poderia ter sido usado algum modelo de referência projetado especificamente para modelar a estrutura de documentos, como os padrões ODA e SGML.

Todas as informações da especificação do hiperdocumento definida pelo autor são armazenadas de forma simples pelo sistema HySCharts. A estrutura navegacional do hiperdocumento representada em Statecharts é armazenada em vários arquivos binários. As informações associadas aos objetos navegacionais e de apresentação são armazenadas em um arquivo texto com a extensão ".hip". Detalhes das estruturas de implementação podem ser encontrados em Turine et al. [1997b].

6.3.2.3. Análise da Estrutura Navegacional

A terceira e última função do módulo de autoria é oferecer ferramentas para análise e validação das especificações de hiperdocumentos segundo o modelo

HMBS. O módulo responsável por tal funcionalidade, em tempo de autoria, é o "Verificador de Propriedades e Objetos HMBS", que é responsável por oferecer o *status* dos objetos do formalismo. Por exemplo, apresenta o nome do arquivo que descreve o Statechart subjacente ao hiperdocumento, o nível de visibilidade e um histórico de todos os objetos navegacionais com os seus respectivos mapeamentos para os objetos estruturais. Na Figura 6.10 é ilustrada uma janela com título "Report" (obtida pela seleção do botão "Evaluation" do ambiente de autoria), que apresenta tal análise para o hiperdocumento de nome "pesc.hip".

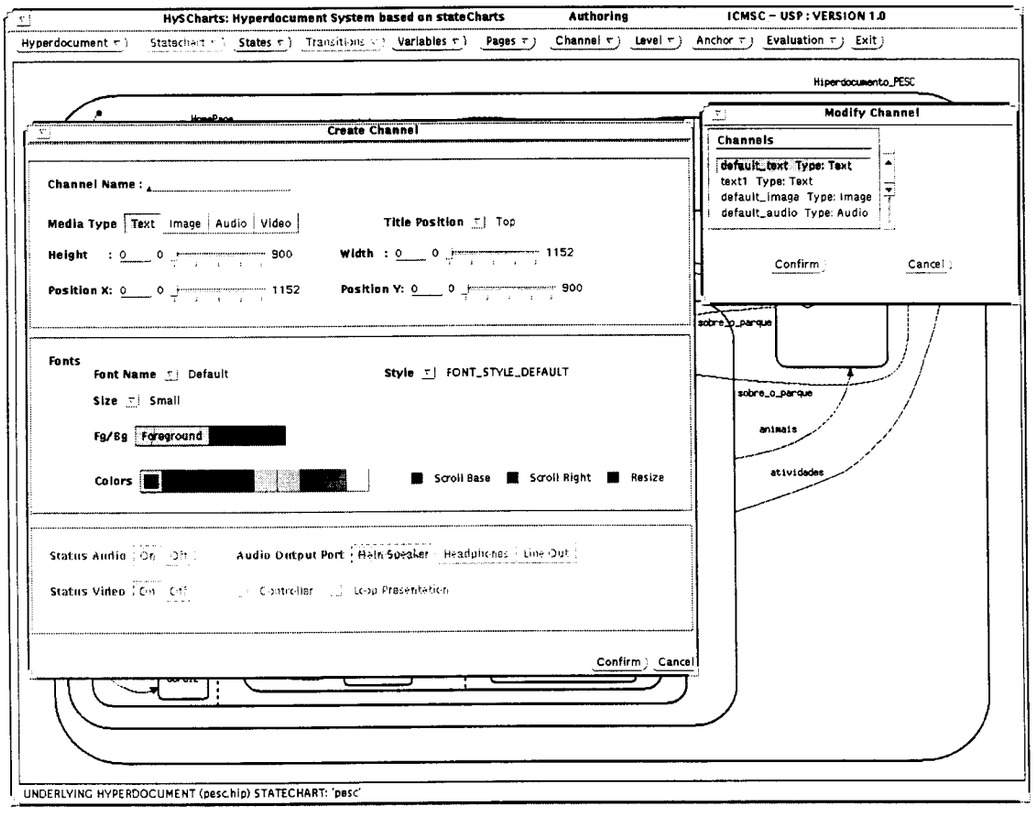


Figura 6.9: Janelas para gerenciamento de canais de apresentação.

Como discutido no Capítulo 5, o HMBS permite realizar algumas análises no hiperdocumento com o objetivo de identificar inconsistências estruturais, bem como problemas relacionados à visualização, navegação e análise de propriedades. Essas análises são baseadas essencialmente na geração da árvore de alcançabilidade, que fornece informações para a análise e detecção de anomalias na especificação. Entretanto, o HySCharts ainda não inclui um módulo de geração e análise da árvore de alcançabilidade, limitando as análises que podem ser feitas atualmente. A implementação desse módulo é proposta como trabalho futuro.

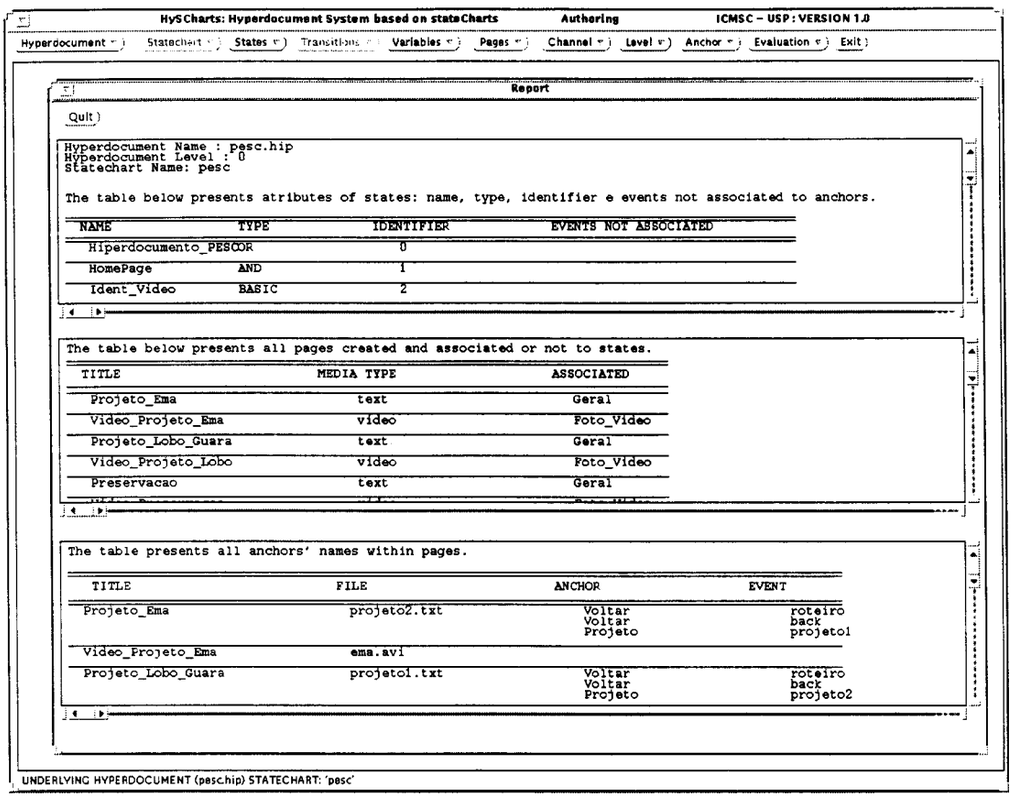


Figura 6.10: Janela do módulo "Verificador dos Objetos HMBS".

6.3.3. Módulo de Navegação

Esse módulo proporciona à camada de aplicação a funcionalidade de apresentar as informações do hiperdocumento, isto é, permite aos leitores visualizar e manipular o conteúdo do hiperdocumento. A camada de aplicação é responsável pela interação com o leitor e, desse modo, houve a preocupação em desenvolver uma interface interativa e simples. As duas ferramentas principais desse módulo são o Simulador de Statecharts e o *Browser*.

O Simulador é um componente do sistema que aceita os eventos externos gerados pelo b*rowser* a partir das ações dos leitores (ativação de uma âncora, por exemplo) e reage de acordo com a semântica operacional de Statecharts, reconfigurando a estrutura navegacional do hiperdocumento a cada passo de simulação. A simulação pode ser feita de forma interativa e ainda visual ou textual. As ações do leitor são interpretadas pelo *browser* e, conseqüentemente, eventos são gerados e enviados ao simulador. Os resultados da simulação são enviados automaticamente para o *browser* que os interpreta e modifica a apresentação do hiperdocumento.

O *browser*, no contexto do sistema HySCharts, corresponde à interface entre o leitor e o simulador da especificação do hiperdocumento. Uma de suas funções é apresentar a estrutura navegacional do hiperdocumento graficamente, proporcionando uma importante medida de contexto e espaço. Além de apresentar a estrutura navegacional subjacente ao hiperdocumento, o *browser* é responsável pela interpretação das configurações de contexto geradas pelo simulador. Para cada página da configuração de contexto o *browser* invoca os canais de apresentação associados, os quais interpretam as informações da página e as exibe em uma janela lógica. Dessa maneira, a visualização das informações do hiperdocumento é realizada pela apresentação de múltiplas janelas lógicas independentes.

Para exibir páginas do tipo texto foi desenvolvido um visualizador de texto, pois os objetos âncora deveriam ficar explícitos. Os objetos navegacionais do tipo âncora são sublinhados e em negrito, e podem ser ativados diretamente pelo leitor. Sua seleção é reconhecida pelo *browser*, que dispara os eventos correspondentes para o simulador de Statecharts.

Para exibir informações não textuais é utilizado o programa XAnim, que permite a visualização de informações em uma grande variedade de mídias (imagem, animação, vídeo, áudio) e formatos (*avi*, *wav*, *gif*, *quicktime*, *mpeg*, *au*, entre outros). A Figura 6.11 ilustra um exemplo de uma apresentação em XAnim, na qual duas janelas são apresentadas: uma corresponde à apresentação de um vídeo (janela de título "Xanim") e uma outra contém os comandos de controle da apresentação (encerrar apresentação, aumentar/diminuir volume, pausa, entre outros). O XAnim permite ao autor personalizar a apresentação, oferecendo um grande número de opções relacionadas à manipulação de áudio, mapeamento de cores, redefinição de tamanho (escala), apresentação em janelas remotas e comandos específicos para controle da animação (*pause*, *play*, *stop*, volume, etc). Por exemplo, os atributos "exibir_controlador" e "apresentação_cíclica" da classe "Canal_Vídeo" (Quadro IV) controlam, respectivamente, a apresentação ou não da janela de comandos de controle e o encerramento ou não do vídeo após a apresentação do último quadro.

A navegação no sistema HySCharts pode ser iniciada pela configuração de contexto inicial ou pela escolha de um título associado a uma página específica (acesso direto à informação usando Índice - Seção 4.6.5). A primeira forma de

navegação é iniciada com a apresentação das janelas lógicas das páginas da configuração "CC_0" que, por sua vez, estão associadas aos estados da configuração inicial "SC_0" do Statechart subjacente. A Figura 6.12 retrata o leiaute que corresponde à configuração de contexto $CC_0 = \{P_{Ident\_Texto}, P_{Ident\_Video}\}$ da Figura 6.4, considerando N = 0 (situação análoga à Figura 4.7(a)). O leitor visualiza duas janelas lógicas: uma janela do tipo texto (título "*State: Ident_Texto Title: Home*") e outra do tipo vídeo (título "*State: Ident_Video*"), à qual sempre está associada a uma janela de comandos de controle. A janela de título "Show_Hview" sempre é apresentada durante a navegação e será definida a seguir. A janela texto associada à "P_{Ident_Texto}" contém as seguintes âncoras, sublinhadas e em negrito: "Sobre_o_Parque", "Atividades" e "Animais". Ao ativar a âncora "Sobre_o_Parque", a ligação associada à transição rotulada pelo evento "sobre_o_parque" da Figura 6.4 é percorrida e o *browser* apresenta as páginas associadas à nova configuração de contexto $CC_1 = \{P_{Nome}, P_{Logotipo}\}$ (Figura 6.13).

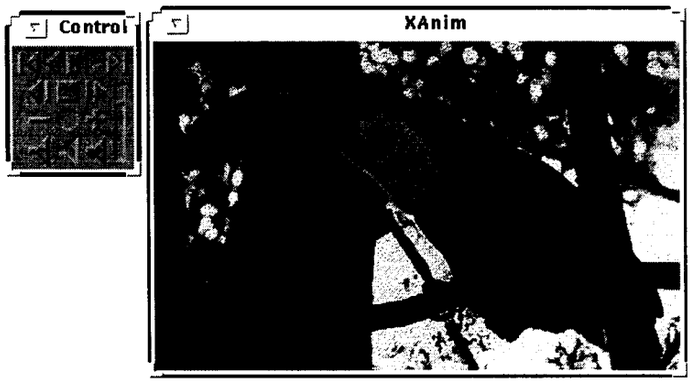


Figura 6.11: Janela de apresentação do XAnim.

Além das janelas lógicas que compõem a apresentação do hiperdocumento, uma outra janela com nome "Show_Hview" é apresentada ao leitor no ambiente de navegação (Figuras 6.12 e 6.13). Tal janela implementa a operação "Mostra_VisãoH" definida no HMBS. Uma vez que a opção "Hview" pertencente à janela "Show_Hview" é ativada pelo leitor, os botões "UP", "DOWN" e "DISMISS" tornam-se ativos e uma nova apresentação é realizada. Se a opção "Hview" é escolhida considerando a configuração de contexto da Figura 6.12, a nova apresentação do hiperdocumento é ilustrada na Figura 6.14, a qual mostra o conteúdo das páginas da configuração de contexto $CC_1 = \{P_{Ident\_Texto}, P_{Ident\_Video}, P_{Hiperdocumento\_PESC}\}$, que inclui a página "$P_{Hiperdocumento\_PESC}$" associada ao estado

"Hiperdocumento_PESC", que é o ancestral da configuração de estados atual SC=={Ident_Texto, Ident_Video} (ver situação análoga retratada na Figura 4.11 (a)).

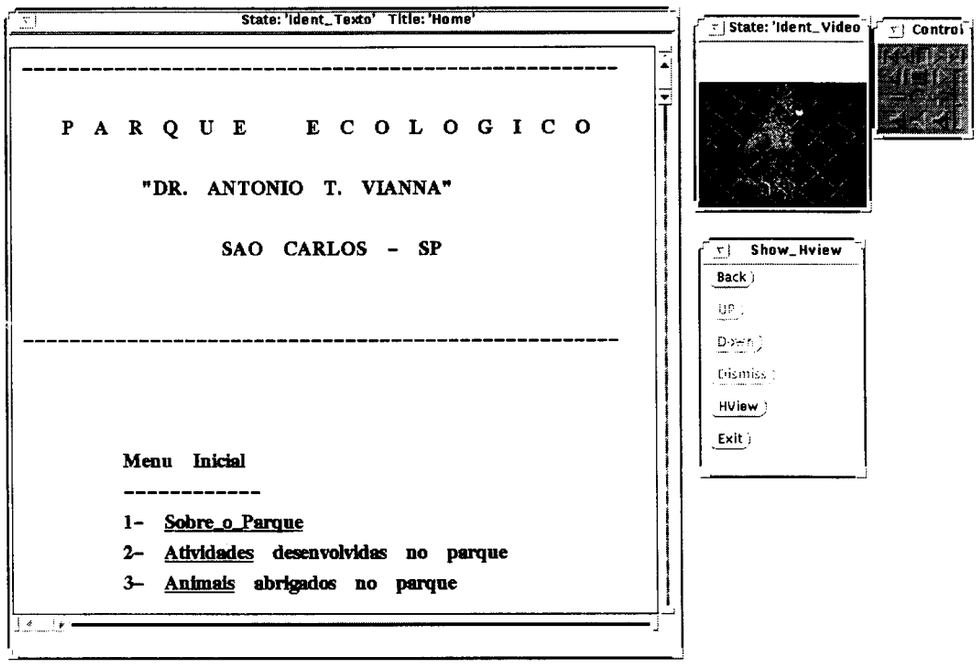


Figura 6.12: Janela de navegação considerando o Statechart da Figura 6.4, N=0 e $CC_0=\{P_{Ident\_Texto}, P_{Ident\_Video}\}$.

Na janela "Show_Hview" também existe a operação "back", que permite ao leitor, durante a navegação, voltar à configuração de contexto imediatamente anterior. Vale ressaltar que a estrutura organizacional do hiperdocumento não representa explicitamente as ligações referentes a essa operação, implementada no sistema HySCharts como um recurso adicional para o leitor.

O resultado da ativação da âncora "Informações" da janela "*State:* Nome *Title:* Sobre_o_Parque" da Figura 6.13, considerando o nível de visibilidade N = 1, é ilustrado na Figura 6.15. A configuração de contexto corrente é CC = $\{P_{Geral1}$, $P_{Nome\_e\_Depoimento}$, P_{Foto}, $P_{Depoimento\_Video}$, P_{Geral} }, de modo que quatro páginas estão associadas diretamente a estados da configuração de estados SC_2={Geral1, Nome_e_Depoimento, Foto, Depoimento_Video}, e a página P_{Geral} representada pela janela lógica "Contexto_Geral" corresponde ao estado ancestral imediato dos estados em "SC_2". Assim, a apresentação do hiperdocumento inclui cinco janelas lógicas. Ainda na Figura 6.15 é exibida a representação gráfica da estrutura navegacional do hiperdocumento em simulação. Dessa forma, o autor pode "executar" o hiperdocumento e observar, graficamente, o efeito na apresentação.

Tal abordagem pode ser útil para simular interativamente a apresentação do hiperdocumento, permitindo analisar seu comportamento e, conseqüentemente, identificar a necessidade de eventuais mudanças na especificação, além de reduzir a desorientação e aumentar a compreensão do hiperdocumento pelo leitor [Thuering et al. 1991].

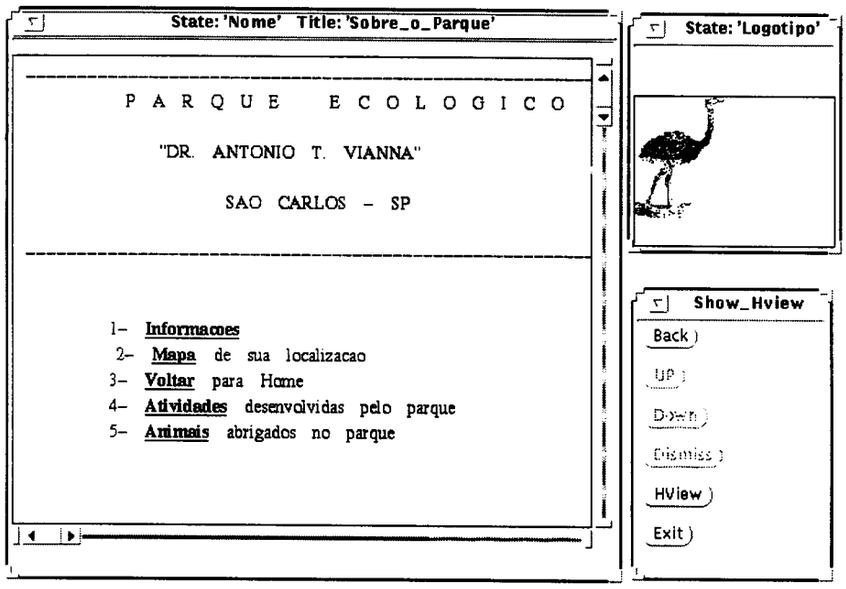


Figura 6.13: Ativação da âncora "Sobre_o_Parque" da configuração de contexto apresentada na Figura 6.12 e CC={P_{Nome}, $P_{Logotipo}$}.

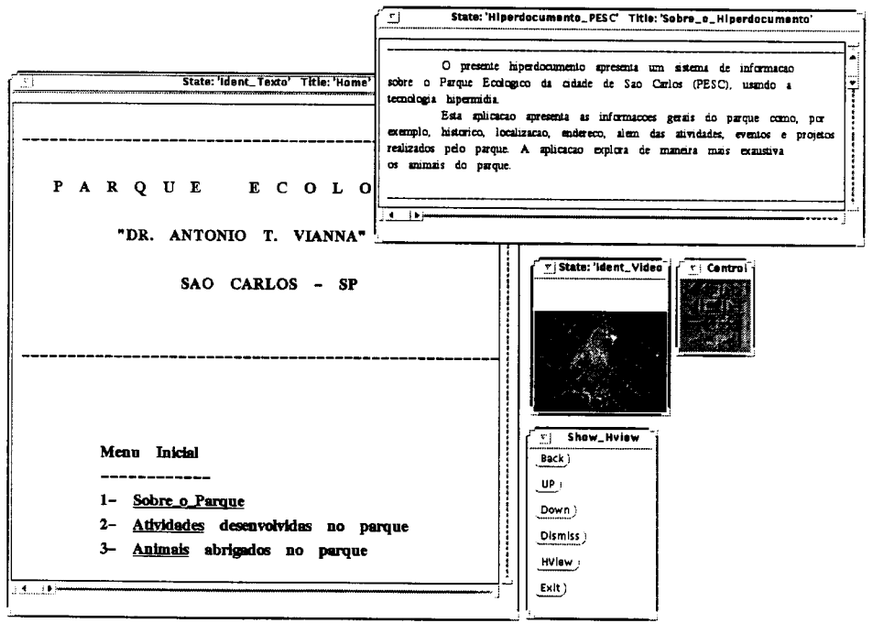


Figura 6.14: Resultado da operação "Mostra_VisãoH" quando CC_0={P_{Ident_Texto},P_{Ident_Video}} e N = 0.

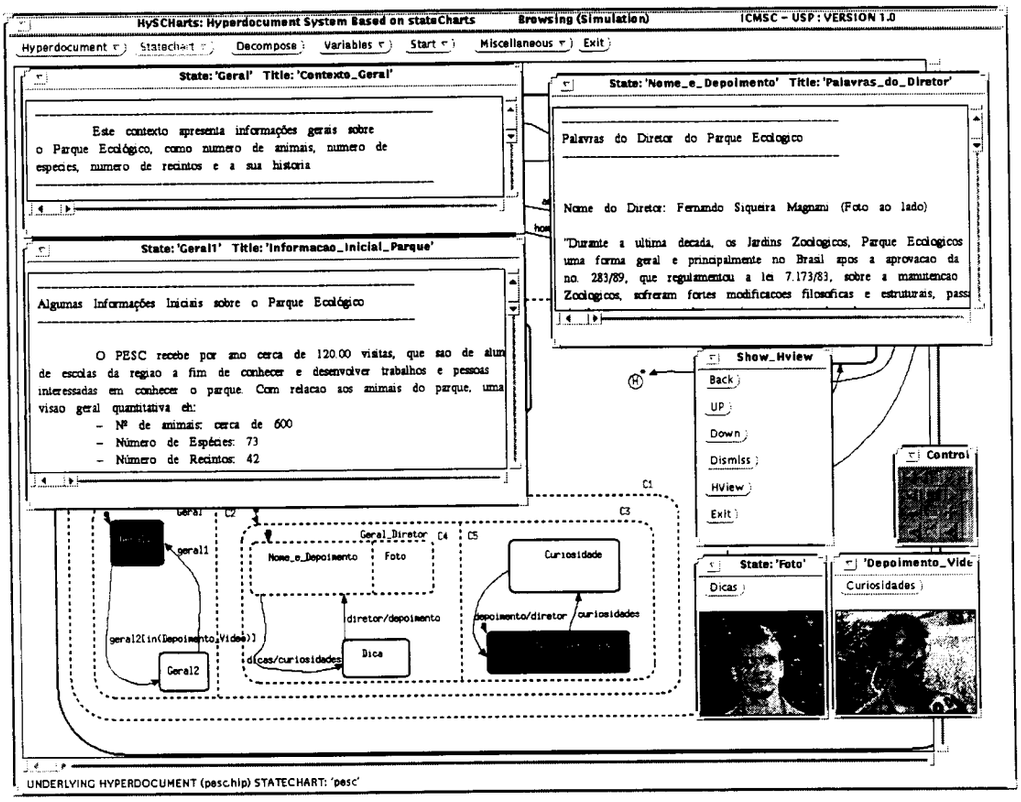


Figura 6.15: Apresentação do hiperdocumento referente à ativação da âncora "Informações" da Figura 6.13 e N = 1.

6.4. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas a arquitetura e as características do sistema HySCharts. Tal sistema foi desenvolvido para implementar o suporte interpretado, que permite criar, interpretar e executar especificações HMBS, além de fornecer facilidades de prototipação e simulação do Statechart subjacente ao hiperdocumento. O objetivo principal desse sistema é validar a proposta do modelo HMBS e permitir aos desenvolvedores verificarem a estrutura organizacional, navegacional e comportamental do mesmo.

A arquitetura do HySCharts é composta pelas camadas de aplicação, de estruturação e de armazenamento. A camada de aplicação é subdividida nos módulos de autoria e de navegação. Na autoria o autor cria o hiperdocumento, e no módulo de navegação o leitor visualiza e navega pelo seu conteúdo. A camada de estruturação contém a estrutura interna dos objetos estruturais, navegacionais e de apresentação do HMBS. Todos esses objetos são armazenados em bases de dados gerenciadas por funções da camada de armazenamento.

No próximo capítulo é apresentada a modelagem de um exemplo prático que utiliza o modelo HMBS e o sistema HySCharts para especificar um hiperdocumento. Adicionalmente, é apresentado um processo de projeto sistemático que insere o formalismo HMBS, principalmente, na fase de projeto navegacional de hiperdocumentos.

Capítulo 7

ESTUDO DE CASO: UM EXEMPLO DE ESPECIFICAÇÃO NO CONTEXTO DO PROJETO DE HIPERDOCUMENTOS

7.1. CONSIDERAÇÕES INICIAIS

A especificação da estrutura organizacional e navegacional do hiperdocumento em HMBS e, conseqüentemente, o mapeamento dessa estrutura em objetos concretos de uma plataforma (WWW ou outro sistema hiperdocumento) são tarefas realizadas durante a atividade de autoria. Na medida em que as aplicações se tornam complexas, é impossível realizar tais tarefas sem o auxílio de uma abordagem de projeto sistemática, eventualmente automatizada. O objetivo deste capítulo é ilustrar como o modelo HMBS e o sistema HySCharts podem ser utilizados no contexto de uma abordagem de projeto sistemática.

O HMBS não é um método de desenvolvimento de hiperdocumentos, e tampouco, um modelo destinado a analisar os requisitos do domínio da aplicação e capturar a sua semântica. Para que o autor obtenha a especificação do hiperdocumento em HMBS é desejável partir de um modelo ou um esquema conceitual que represente os objetos e os relacionamentos do mundo real. Após a

especificação conceitual do problema, pode-se especificar os usos do hiperdocumento e as tarefas particulares a serem realizadas sobre o mesmo, atividades comumente realizadas durante uma fase de modelagem navegacional.

Assim, pretende-se mostrar neste capítulo que o HMBS pode ser integrado a uma abordagem de projeto de mais alto nível para o desenvolvimento de hiperdocumentos. Ressalta-se que a concepção inicial dessa abordagem é uma das contribuições desta tese, mas sua descrição completa está no escopo de um projeto de mestrado em desenvolvimento no ICMC [Carvalho 1997, Carvalho 1998].

O veículo utilizado para ilustrar como a especificação navegacional em HMBS pode ser obtida em uma abordagem de projeto, foi desenvolver uma aplicação hipermídia para o Parque Ecológico de São Carlos (PESC). Uma descrição dessa aplicação é apresentada na Seção 7.3. Na Seção 7.4 as quatro fases do processo de projeto (projeto conceitual, projeto navegacional, projeto de interface e implementação/avaliação) são apresentadas no contexto do PESC. Tal hiperdocumento foi totalmente modelado como parte desta tese e implementado no sistema HySCharts. Posteriormente, implementou-se uma versão do PESC em HTML, que pode ser executada no *browser* Netscape e comparada com a versão do sistema HySCharts. As conclusões são apresentadas na Seção 7.5.

7.2. UM EXEMPLO: DESCRIÇÃO

Apesar de simples, a aplicação utilizada como estudo de caso ilustra o uso do modelo HMBS e como a estrutura navegacional em HMBS pode ser obtida no contexto de um processo de projeto de mais alto nível. O hiperdocumento em questão deve apresentar as informações sobre o Parque Ecológico "Dr. Antônio T. Vianna", localizado na cidade de São Carlos. Potenciais leitores dessa aplicação são os estudantes de nível secundário, que buscam informações para trabalhos escolares; pesquisadores interessados em investigar os projetos de preservação de espécies em andamento, como o da preservação do "Mico Leão Dourado", espécie ameaçada de extinção; e o público de visitantes do parque em geral.

A aplicação deve incluir informações sobre o parque e os animais nele abrigados e descrições das atividades promovidas pelo parque, bem como um histórico do PESC, um depoimento do diretor e dicas para os visitantes que desejam conhecer o parque. O foco do hiperdocumento são os animais,

aproximadamente 600 de um total de 73 espécies, cujas características populares e algumas científicas devem ser apresentadas para o leitor.

7.3. O PROCESSO DE PROJETO

Segundo Nanard e Nanard [1995], o processo de desenvolvimento de hiperdocumentos não é intrinsecamente diferente daquele utilizado na construção de softwares convencionais. Existe um consenso crescente acerca das atividades que devem ser desempenhadas em cada uma das fases do processo, que compreendem a análise, o projeto, a implementação, o teste e a manutenção. O OOHDM [Schwabe et al. 1996], o RMM [Isakowitz 1995, Isakowitz et al. 1997a] e o EORM [Lange 1994] são exemplos representativos de métodos publicados que se apóiam em uma abordagem sistemática para o desenvolvimento de hiperdocumentos. O processo de projeto adotado no desenvolvimento do aplicativo PESC emprega conceitos e modelos já definidos por outros métodos, principalmente o RMM e o OOHDM.

Rossi [1996] aponta que, no contexto de hiperdocumentos, já existe um amplo reconhecimento de que há problemas relacionados às atividades desenvolvidas nas diferentes fases. Por exemplo, a definição de formalismos adequados para especificar os produtos gerados em cada fase, a compreensão de como um modelo evolui durante o processo, a tradução do modelo de análise em um modelo de projeto, a integração da atividade de teste e a reutilização de produtos e estratégias de projeto.

Isakowitz et al. [1995] salientam a necessidade de prototipação e testes intensivos com leitores durante o desenvolvimento do hiperdocumento, uma vez que a tolerância a erros é bastante baixa nesse tipo de aplicação. Nielsen [1990a, 1990b] também reforça a necessidade de constantes avaliações, mencionando que a estrutura geral de um hiperdocumento sofre, muitas vezes, a interferência das decisões tomadas durante a construção de cada página de informação e de cada ligação individualmente. Dessa forma, uma abordagem de prototipação no processo de projeto de hiperdocumentos é recomendada, pois fornece *feedback* para os autores e permite que os leitores e a equipe de desenvolvimento esclareçam e determinem os requisitos e as suas formas de interpretação.

O processo de projeto adotado foi subdividido em quatro diferentes fases, denominadas projeto conceitual, projeto navegacional, projeto de interface e

implementação/avaliação (Figura 7.1). Essas fases foram realizadas segundo uma abordagem mista de desenvolvimento incremental e iterativa, de modo que em cada fase um modelo foi construído ou enriquecido. Ademais, o processo enfoca que as decisões de autoria em ponto pequeno (como as que se referem à estrutura e ao conteúdo de uma página de informação) não devem ser tomadas durante a fase de projeto conceitual, mas na fase de projeto de interface. Protótipos podem ser desenvolvidos a partir da fase de projeto navegacional. As características e os modelos produzidos em cada fase são apresentados sucintamente a seguir.

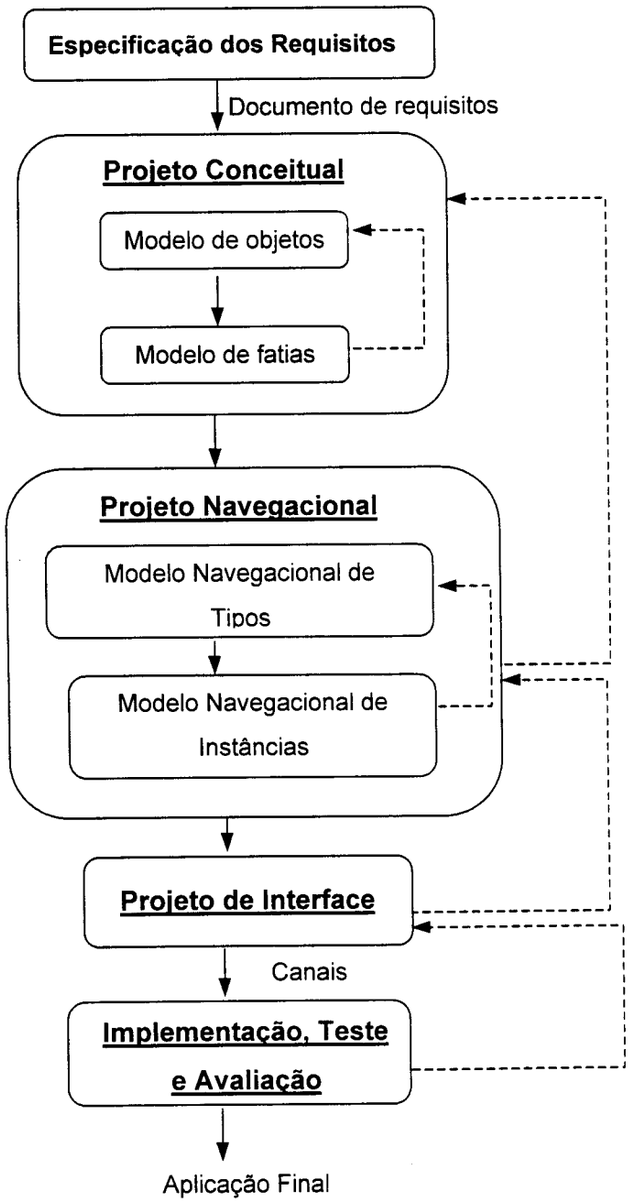


Figura 7.1: As fases do processo de projeto.

7.3.1. Projeto Conceitual

O objetivo principal dessa fase é captar a semântica do domínio do problema, que é o ponto de partida para qualquer desenvolvimento. O autor do hiperdocumento especifica os aspectos gerais da aplicação, iniciando a autoria em ponto grande sem se preocupar com os tipos de leitores, os usos e as tarefas a serem realizadas no hiperdocumento. Durante essa fase, a estrutura semântica do domínio é especificada pelo modelo de objetos e pelo modelo de fatias. Esses dois modelos são estáticos, ou seja, não especificam características navegacionais e comportamentais da aplicação.

Modelo de Objetos

O modelo de objetos utilizado adota os princípios do método de desenvolvimento de software orientado a objetos *Fusion* [Coleman et al. 1994]. As classes são usadas para representar os objetos do domínio, os atributos representam as propriedades intrínsecas ou conceituais dos objetos, e os relacionamentos associativos representam as ligações entre eles. Somente o conceito original "atributo" do *Fusion* foi estendido para fornecer suporte a atributos multi-tipados, ou seja, um mesmo atributo pode ter vários tipos de dados associados (por exemplo, String, Número, Caractere, Texto, Imagem, Vídeo, Áudio, Data), que especificam as diferentes formas de apresentação (aparência visual ou retórica). Cada aparência de um atributo é denominada "perspectiva do atributo" (conceito derivado do HDM e do OOHDM). No caso de existirem múltiplas perspectivas, uma delas é escolhida como perspectiva *default* (assinalada com um sinal de "+"), que é a perspectiva sob a qual o atributo será inicialmente mostrado.

A Figura 7.2 ilustra o modelo de objetos para o PESC, que possui nove classes ("Parque_Ecológico", "Animal", "Atividade", "Diretor", "Projeto", "Educação_Ambiental", "Lazer_Contemplativo", "Evento", "Preservação") e dois relacionamentos associativos ("promove/é_promovida", "abriga/é_abrigado"). A classe "Parque_Ecológico" é uma agregação da classe "Diretor" (classes complexas são especificadas como agregações de classes mais simples) e contém as informações gerais sobre o PESC, além de informações sobre seu diretor. A classe "Animal" contém informações sobre todos os animais abrigados no parque. A classe "Atividade" contém informações sobre as atividades realizadas no PESC e é especializada de forma exclusiva nas classes "Projeto", "Educação_Ambiental",

"Lazer_Contemplativo", "Evento" e "Preservação". Como na abordagem orientada a objetos, as classes podem ser construídas utilizando os mecanismos de generalização/especialização e agregação, aumentando o poder de abstração da aplicação.

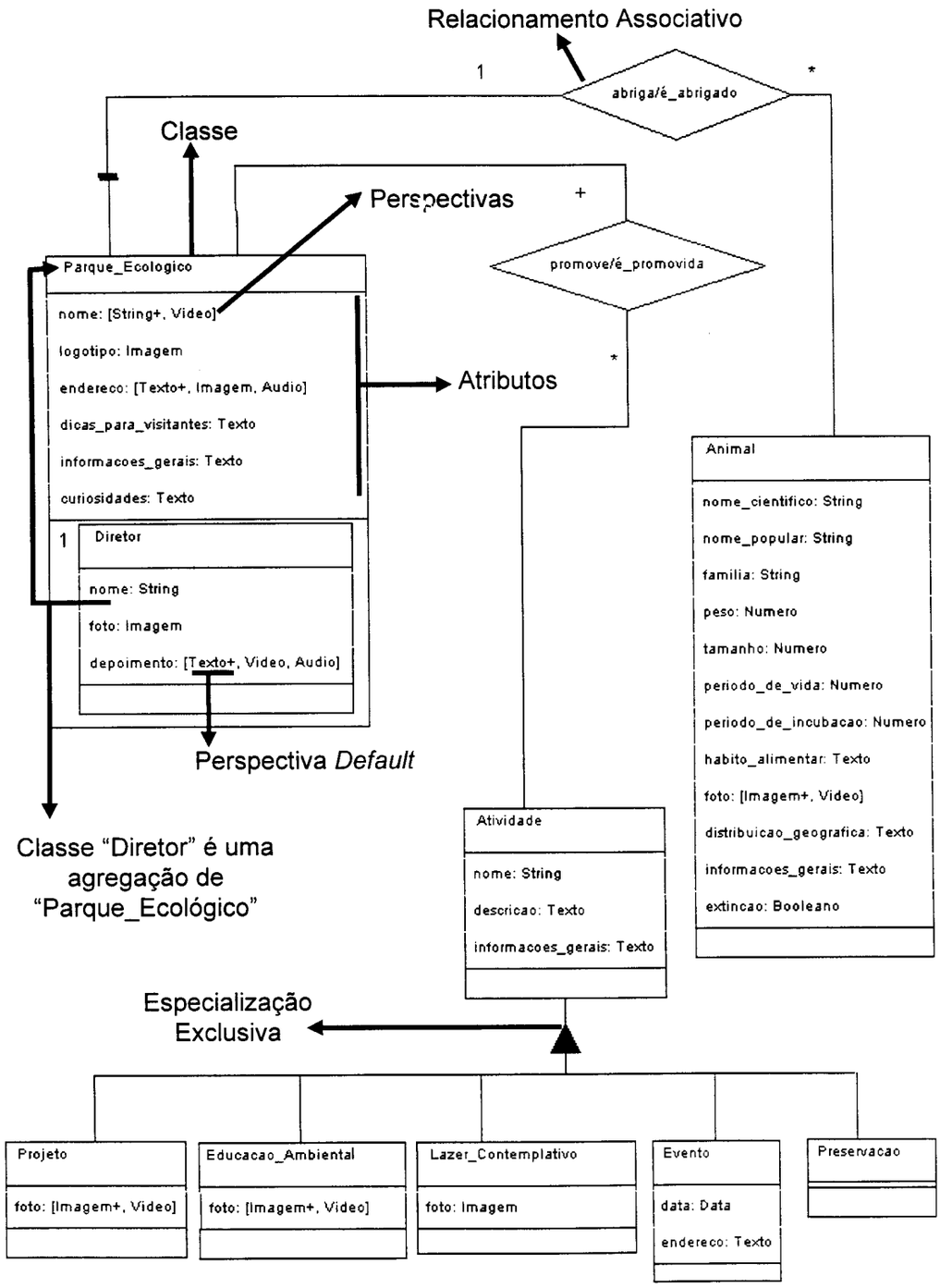


Figura 7.2: Modelo de objetos do PESC.

O atributo "endereço" da classe "Parque_Ecológico" é representado por "endereço: [Texto+, Imagem, Áudio]", indicando que a informação associada ao atributo pode ser apresentada de três formas: a perspectiva "Texto" é a *default*, e contém uma descrição textual da localização do parque; a perspectiva "Imagem" contém um mapa de sua localização; e a perspectiva "Áudio" contém uma versão sonora da descrição textual.

Modelo de Fatias

O modelo de fatias é o modelo de objetos enriquecido com a primitiva de modelagem "fatia", que contém as unidades de informação do hiperdocumento a serem apresentadas durante a navegação. As fatias são representadas graficamente por um retângulo no interior da classe, e a fatia *default* possui no canto superior direito uma marcação específica (quadrado preenchido). A Figura 7.3 ilustra o modelo de fatias para o PESC originado a partir do modelo de objetos da Figura 7.2. O termo "fatia" foi introduzido no método RMM e indica uma coleção de atributos agrupados segundo critérios como: atributos que podem ser visitados e/ou apresentados simultaneamente, atributos muito extensos e atributos apresentados por um mesmo tipo de mídia.

O agrupamento dos atributos das classes em fatias permite determinar, em tempo de autoria, as unidades de informação do hiperdocumento que serão exibidas sem fragmentação e simultaneamente. Na Figura 7.2, a classe "Animal" tem um grande número de atributos de diferentes naturezas, como informações textuais ("nome_científico", "família", "hábito_alimentar") e gráficas, ("foto"). Pode ser impraticável ou indesejável apresentar para o leitor todas as informações relativas a esses atributos em uma mesma janela lógica. Na Figura 7.3 nota-se que os atributos da classe "Animal" foram agrupados em três fatias ("Descrição", "Foto_Imagem", "Foto_Vídeo"), considerando a fatia "Descrição" como *default*.

Os atributos da classe "Parque_Ecológico", por exemplo, foram agrupados em seis fatias denominadas "Identificação", "Principal", "Geral", "Endereço", "Dicas" e "Curiosidades", sendo que a fatia "Principal" é a *default*. Desse modo, cada classe no modelo de fatias terá uma ou mais fatias, e para toda classe deve existir uma fatia *default*. Nesta fase o autor não se preocupa em definir os relacionamentos semânticos entre as fatias da classe.

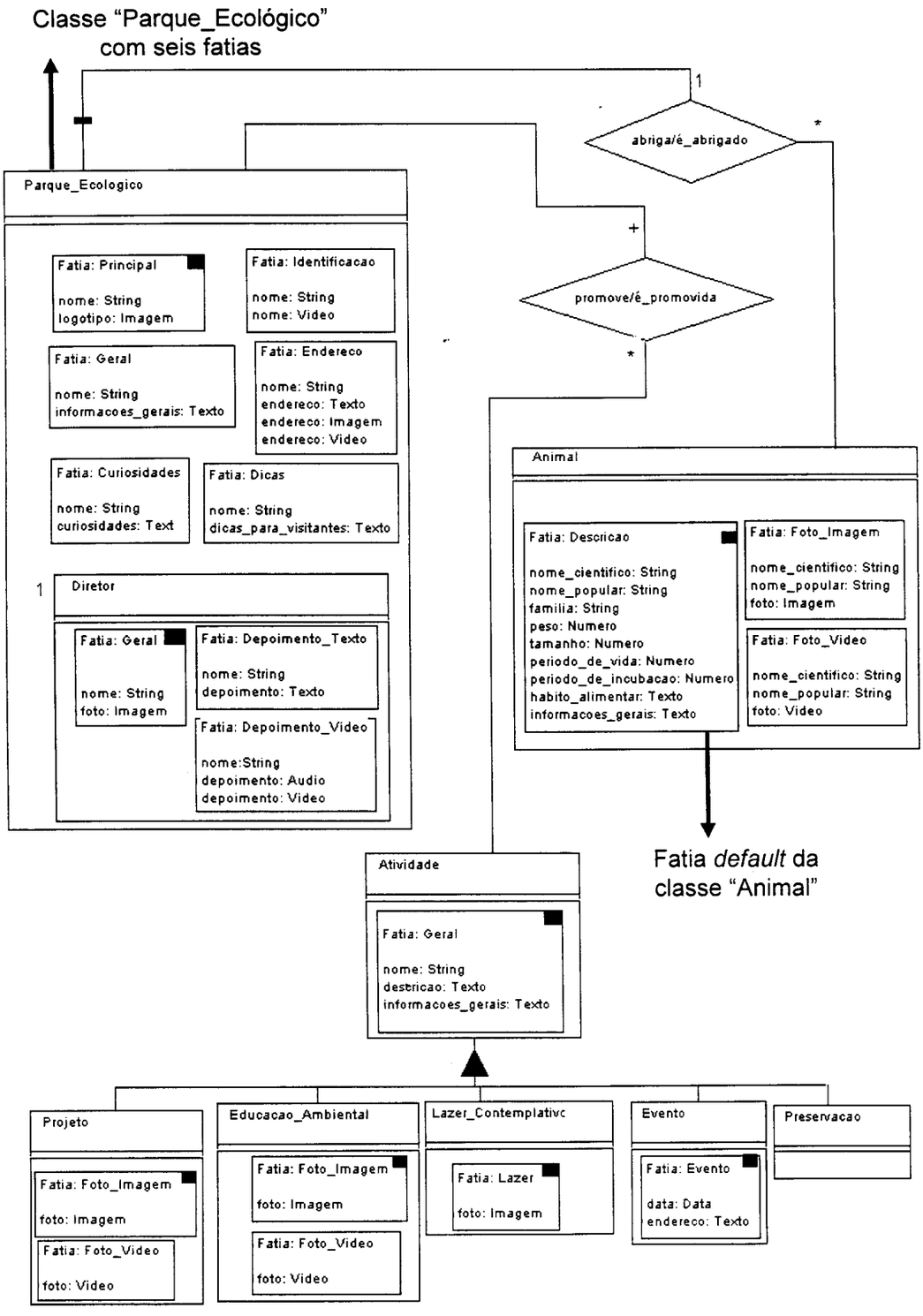


Figura 7.3: Modelo de fatias do PESC.

7.3.2. Projeto Navegacional

Para produzir um hiperdocumento navegável, as informações representadas no modelo de fatias devem ser reorganizadas. A fase de projeto navegacional é crucial para o processo de desenvolvimento e tem como objetivo especificar a estrutura navegacional do hiperdocumento. A partir dessa fase, protótipos podem ser desenvolvidos para avaliar se o hiperdocumento corresponde às necessidades do leitor e aos objetivos do autor.

De acordo com Rossi [1996], para que o autor possa especificar a estrutura navegacional e comportamental de um hiperdocumento é necessário definir vários aspectos:

a) Quais objetos serão navegados e quais atributos desses objetos são relevantes?
b) Quais relacionamentos existentes entre os objetos serão mapeados em caminhos de navegação? Quais estruturas de acesso serão utilizadas nos caminhos de navegação?
c) Qual é a estrutura navegacional subjacente ao hiperdocumento? Em que contextos navegacionais o leitor poderá navegar?
d) Como proceder a navegação quando o leitor "pula" de um assunto a outro, isto é, qual o efeito da navegação nos objetos de origem e alvo, e em outros objetos relacionados?

O item a) já está especificado no modelo de fatias. Na fase de projeto navegacional os demais itens serão definidos e dois novos modelos são gerados, sendo denominados modelo navegacional de tipos e modelo navegacional de instâncias. O modelo navegacional de tipos retrata a estrutura navegacional estática do hiperdocumento. Os padrões dinâmicos e comportamentais são especificados no modelo navegacional de instâncias, que adota o modelo HMBS para especificar precisamente o comportamento navegacional do hiperdocumento.

Modelo Navegacional de Tipos

O modelo navegacional de tipos é uma visão orientada à navegação do modelo de fatias e inclui a especificação dos contextos de navegação e das estruturas de acesso. Deve-se verificar quais relacionamentos associativos do modelo de fatias serão acessíveis durante a navegação. O relacionamento, dependendo da sua cardinalidade e de sua importância para a navegação, é

substituído por uma ou mais estruturas de acesso. Tais estruturas especificam os diferentes caminhos de navegação disponíveis para acessar as informações do hiperdocumento. As estruturas de acesso utilizadas são as mesmas definidas pelo RMM [Isakowitz et al. 1995]: ligações uni e bidirecionais, índice, roteiro guiado, agrupamento e roteiro guiado indexado.

As ligações estruturais uni e bidirecionais especificam o relacionamento entre as fatias de uma mesma classe, além dos relacionamentos entre as estruturas de acesso. Um índice atua como uma tabela de conteúdo para uma lista de objetos da classe, fornecendo acesso direto a cada item. A estrutura de acesso roteiro guiado implementa um caminho linear como uma coleção de objetos, permitindo ao leitor mover-se ao longo do caminho para frente ou para trás. Roteiros guiados fornecem caminhos rígidos entre unidades de informação do hiperdocumento. A estrutura agrupamento define os pontos de entrada no hiperdocumento, isto é, um mecanismo do tipo menu que permite o acesso a estruturas índice e roteiro guiado, ou mesmo a unidades de informação. Um roteiro guiado indexado é similar ao roteiro guiado com a adição de um acesso direto a cada objeto que forma o roteiro, fornecendo uma estrutura de acesso mais rica.

Segundo Schwabe e Rossi [1995] e Schwabe et al. [1996], a especificação de contextos de navegação é fundamental no projeto de hiperdocumentos, pois permitem implementar estratégias de navegação bem definidas e incluir informações contextuais que auxiliem o leitor a percorrer as páginas de seu interesse. Por definição, os contextos permitem reunir informações do domínio da aplicação que estão relacionadas segundo alguns critérios, como navegar por todos os animais da família "Réptil", ou todas as obras artísticas de um mesmo pintor. A noção de contexto de navegação ajuda o leitor a criar domínios específicos dentro da aplicação, reduzindo a sobrecarga cognitiva imposta pela navegação, além de minimizar o problema da desorientação do leitor. Rossi [1996] apresenta vários tipos de contextos navegacionais que variam de acordo com o modo como são originados: arbitrário, derivado de classe, derivado de elo e derivado de um composto. No modelo navegacional de tipos, o autor utiliza as estruturas de acesso para especificar os contextos navegacionais.

A Figura 7.5 ilustra o modelo navegacional de tipos para o PESC. A legenda das primitivas utilizadas no modelo navegacional de tipos é apresentada na Figura 7.4. O ponto de entrada no hiperdocumento é definido pela primitiva agrupamento

"Menu_PESC". As informações sobre o parque (classe "Parque_Ecológico") são disponíveis a partir de uma ligação estrutural bidirecional. Um roteiro guiado indexado denominado "Roteiro_Atividade" fornece acesso a cada um dos cinco objetos da classe "Atividade"[19] ("Projeto", "Evento", "Lazer_Contemplativo", "Educação_Ambiental" e "Preservação"). Existem quatro contextos para obter as informações sobre os animais: duas estruturas índice que apresentam os animais em ordem alfabética de nome ("Índice_Nome") e de família ("Índice_Família"), e dois roteiros guiados, o "Roteiro1" especifica um caminho linear por todos os animais em uma ordem pré-estabelecida pelo autor e o "Roteiro2" especifica um caminho por todos os animais em extinção do PESC.

Os relacionamentos associativos de cardinalidade um para muitos (1:*) do modelo de fatias foram transformados em duas ligações de aplicação. Por exemplo, o relacionamento "abriga/é_abrigado" entre "Parque_Ecológico" e "Animal" foi mapeado na ligação de aplicação "é_abrigado" entre "Animal" e "Parque_Ecológico", e na ligação "abriga", via estrutura de acesso "Menu_Animal". Todas as fatias de uma mesma classe foram inter-relacionadas por ligações estruturais uni ou bidirecionais. Dependendo da necessidade dos leitores, ligações arbitrárias podem ser derivadas (conceito do HDM), como a ligação entre as classes "Animal" e "Atividade", especificada pela estrutura "Roteiro_Atividade".

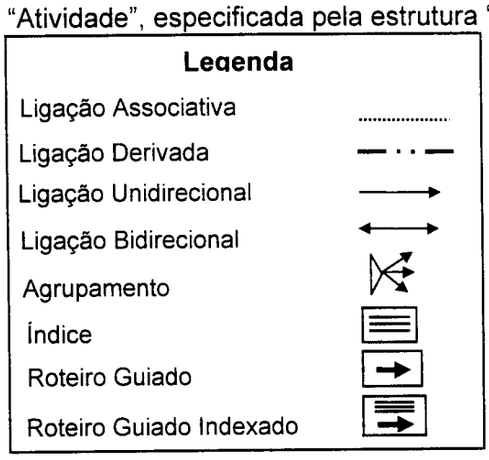


Figura 7.4: Legenda para as primitivas do modelo navegacional de tipos.

[19] Por simplificação, esses objetos da classe "Atividade" não foram representados no modelo de tipos.

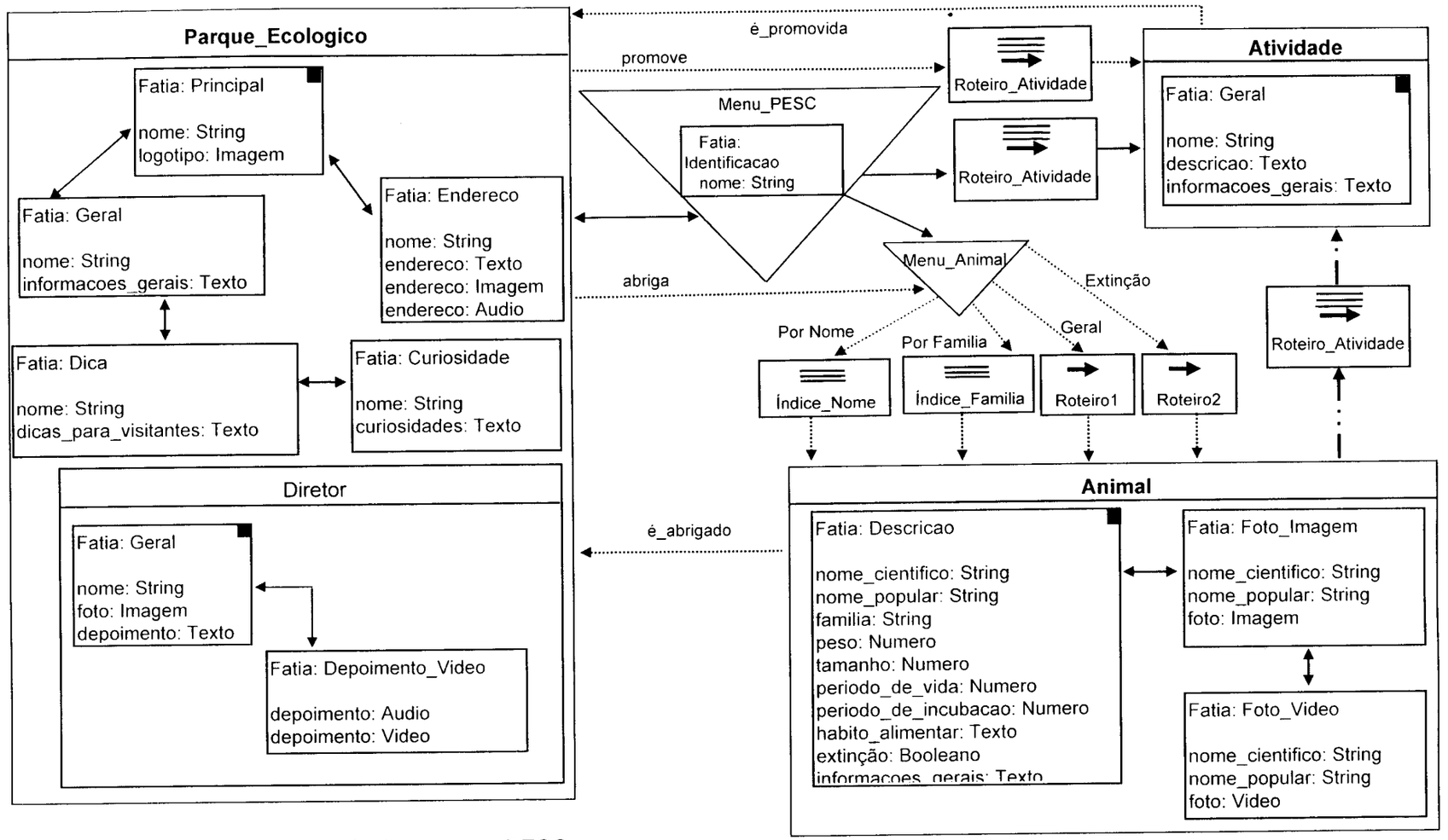


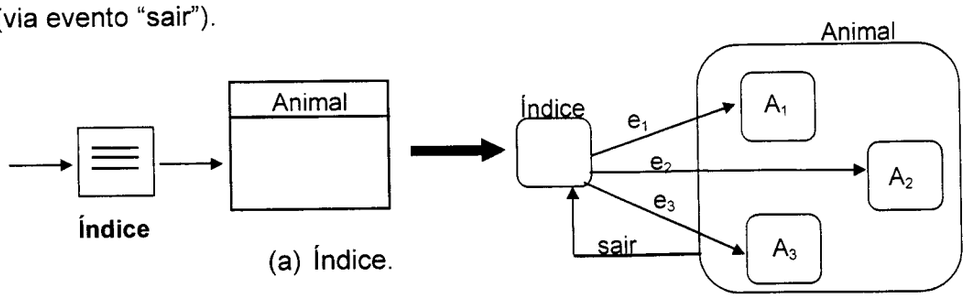
Figura 7.5: Modelo navegacional de tipos para o PESC.

Modelo Navegacional de Instância

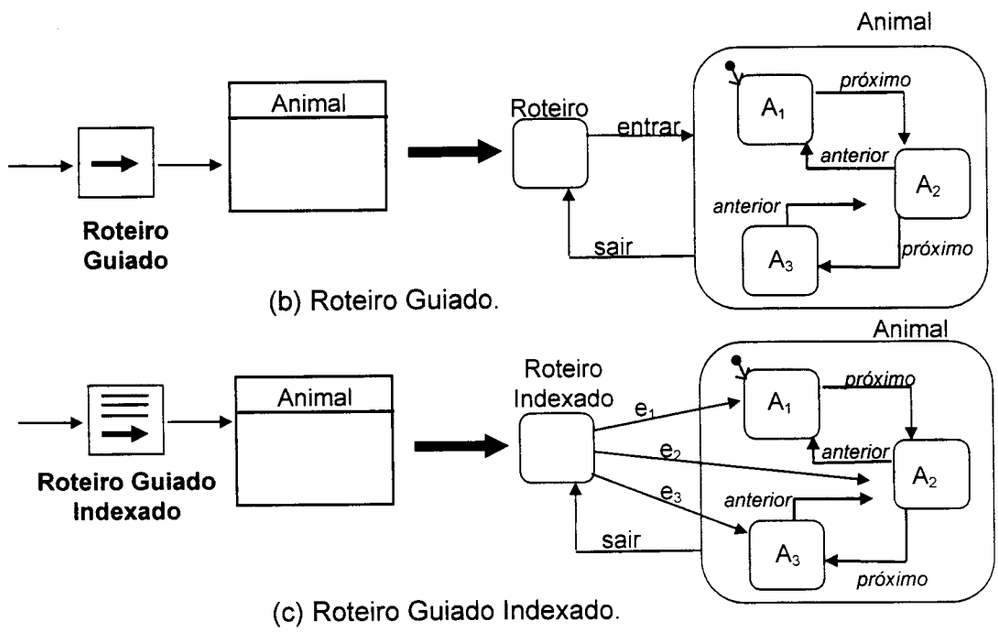
Após obter o modelo navegacional de tipos, o autor deve especificar a estrutura navegacional em HMBS, produzindo o modelo navegacional de instância do hiperdocumento. Os objetos estruturais do modelo HMBS (estados, transições e eventos) permitem instanciar o modelo navegacional de tipos para uma estrutura de um hiperdocumento específico que, posteriormente, pode ser "traduzida" para a notação de uma plataforma ou sistema destino.

As unidades de informação do hiperdocumento são definidas com base nos atributos das classes, e as ligações entre as páginas são geradas pelos relacionamentos especificados entre as classes do modelo navegacional de tipos e entre as fatias de cada classe.

Na Figura 7.6 são ilustradas algumas abordagens de projeto para mapear as estruturas de acesso (índice, roteiro guiado e roteiro guiado indexado) em objetos estruturais do HMBS que especificam um comportamento navegacional. Supondo que a classe "Animal" do PESC contivesse somente três animais (A_1, A_2, A_3), pode-se mapeá-la em um estado OR de nome "Animal" com três subestados, que definem as instâncias da classe. A Figura 7.6 (a) ilustra o acesso à classe "Animal" por uma estrutura índice, mapeada em um estado "Índice". Cada um dos objetos animal "A_1", "A_2" e "A_3" pode ser acessado por uma transição com rótulo "e_1", "e_2" e "e_3", respectivamente. Para retornar ao índice, uma transição rotulada pelo evento "sair" é especificada. A Figura 7.6 (b) ilustra a transformação de um roteiro guiado e a Figura 7.6 (c) a transformação de um roteiro guiado indexado. No roteiro guiado, transições rotuladas pelos eventos "próximo" e "anterior" entre os subestados "A_1", "A_2" e "A_3" são definidas e permitem a movimentação para a frente e para trás no caminho navegacional. Nas duas estruturas roteiro guiado, o leitor pode abandonar qualquer configuração de contexto e voltar ao estado origem que inicia o roteiro (via evento "sair").



(a) Índice.



(b) Roteiro Guiado.

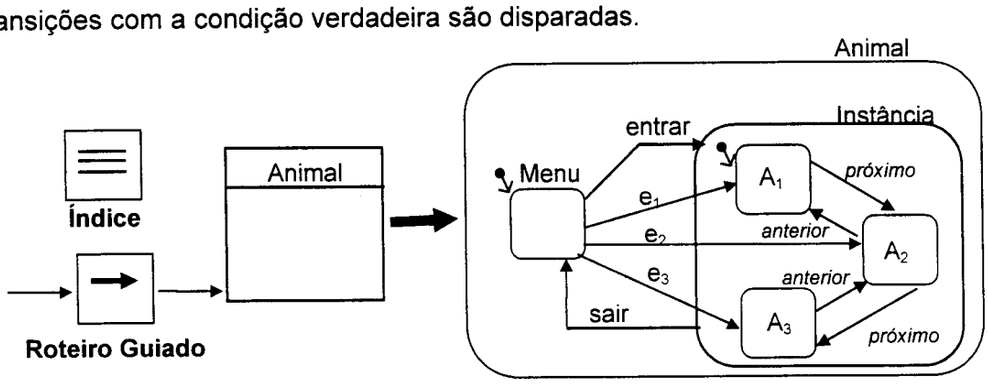
(c) Roteiro Guiado Indexado.

Figura 7.6: Mapeamento de estruturas de acesso em objetos estruturais HMBS.

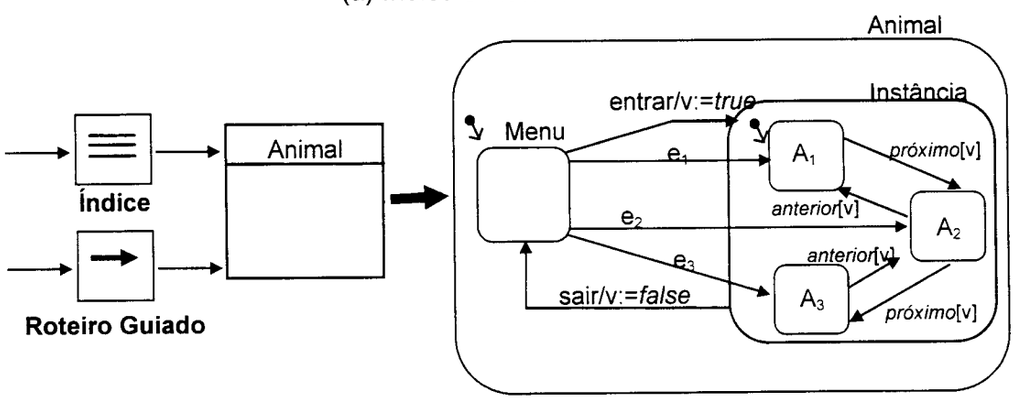
As estratégias de mapeamento apresentadas acima não são únicas, e o autor pode, dependendo da aplicação, explorar diferentes alternativas. Por exemplo, se existem várias estruturas de acesso à classe "Animal", pode-se mapeá-las conforme a Figura 7.7 (a). Nesse exemplo, duas estruturas de acesso (Índice e Roteiro Guiado Indexado) fornecem pontos de entrada à classe "Animal". Como estratégia, a decisão de qual estrutura seguir ficou implícita no subestado "Menu", que é *default* do superestado "Animal" e fornece acesso às diferentes instâncias da classe "Animal" pelas transições correspondentes. A transição rotulada pelo evento "entrar" inicia o roteiro guiado e as outras transições com origem no estado "Menu" fornecem acesso direto às instâncias. Esse poder de abstração (fatoração de transições como, por exemplo, o evento "sair") permite compactar transições, gerando um modelo de instância mais simples.

A especificação da estratégia ilustrada na Figura 7.7 (a) apresenta um problema de controle na navegação. Se o leitor escolhesse o contexto especificado pela estrutura "Índice", não se poderia permitir que ele caminhasse para a frente e para trás nas instâncias (via as âncoras "próximo" e "anterior"). Por exemplo, estando em "Menu" e disparando a transição com rótulo "e1", a página associada ao estado "A1" é atingida. Nessa configuração de contexto, o leitor não deve ser capaz de ativar o evento "próximo", mas somente de voltar ao estado "Menu"

disparando "sair". Para controlar o acesso nos diferentes contextos de navegação pode-se definir variáveis de controle de acesso (como discutido na Seção 5.3.3). Por exemplo, na Figura 7.7 (b) é definida uma variável lógica "v" que controla o acesso ao contexto "Roteiro Guiado". Inicialmente, o autor inicia tal variável como *false*, mas quando a transição rotulada pelo evento "entrar" é disparada, uma ação é gerada e o valor da variável é modificada para *true*. Dessa maneira, somente as transições com a condição verdadeira são disparadas.



(a) Índice e Roteiro Guiado.



(b) Índice e Roteiro Guiado com variável de controle de acesso.

Figura 7.7: Estratégias alternativas para o mapeamento de estruturas de acesso.

A Figura 7.8 ilustra o modelo navegacional de instância do PESC originado a partir do modelo navegacional de tipos da Figura 7.5. As estruturas navegacionais das classes "Animal" e "Atividade" são especificadas em outro nível de abstração. A Figura 7.9 ilustra o refinamento do modelo para o subestado "Animal" e o refinamento do subestado "Atividade" é apresentado na Figura 7.10.

Durante o processo de mapeamento do modelo navegacional de tipos para o modelo navegacional de instância, o autor deve tomar algumas decisões apoiadas em um conjunto de regras heurísticas que devem considerar os interesses

navegacionais do conjunto de leitores alvo:

DECISÃO 1:

As classes do modelo navegacional de tipos são mapeadas para estados compostos do tipo OR ou AND. Em geral, as composições OR são usadas para agrupar fatias de uma mesma classe e/ou instâncias de uma classe. As composições AND são usadas para agrupar fatias cujas informações são apresentadas simultaneamente ou, ainda, atributos de diferentes perspectivas pertencentes às fatias.

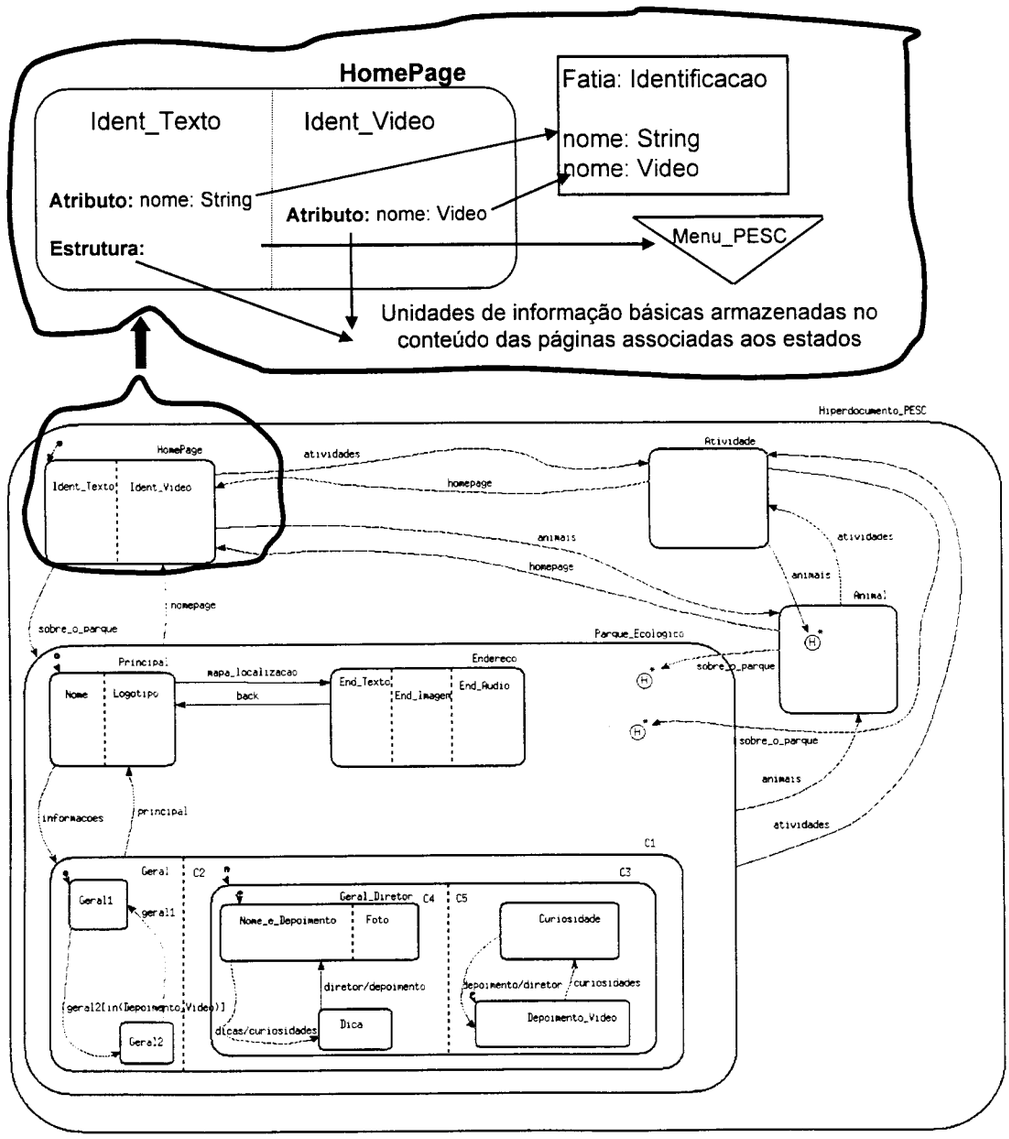


Figura 7.8: Modelo navegacional de instância do PESC (alto nível).

Por exemplo, na Figura 7.9 a classe "Animal" da Figura 7.5 foi instanciada em um estado OR com três subestados "Menu_Animal" (estado *default*), "Índice_Animal" e "Instância_Animal". A partir do estado "Menu_Animal" pode-se explorar quatro contextos navegacionais distintos (índice dos animais por nome e por família, um roteiro guiado sobre os animais em extinção e um outro aleatório), que são obtidos pelos eventos "índice", "roteiro1" e "roteiro2". O subestado "Indice_Animal" agrupa as estruturas índice ("Índice_Nome" e "Índice_Família"), que fornecem acesso direto às informações sobre os animais encontrados no subestado "Instância_Animal". Esse estado, por sua vez, foi decomposto em sete subestados, um para cada animal do PESC: "Arara_Vermelha", "Ema", "Harpia", "Lobo_Guará", "Mico_Leão_Dourado", "Onça_Pintada" e "Tucano" (Figura 7.9). A variável "cn2", definida como *true* a partir da ativação do evento "roteiro2", determina um contexto navegacional formado por todos os animais em extinção. Por outro lado, a variável "cn1", definida como *true* a partir da ativação do evento "roteiro1", permite explorar um caminho aleatório sobre os animais.

A Figura 7.9 ilustra uma situação em que o estado "Mico_Leão_Dourado" é compartilhado por roteiros guiados distintos, situação apresentada em detalhes na Figura 7.11. Nesse caso, a página associada ao estado "Mico_Leão_Dourado" terá uma âncora "Próxima Página" associada ao evento "next" que, dependendo do contexto corrente disparará transições diferentes. Observe que o autor pode restringir ainda mais as condições de acesso para evitar inconsistências estruturais, por exemplo, o evento "next[cn2]" poderia ser definido como "next[cn2 and not cn1]".

De forma análoga, a classe "Atividade" (Figura 7.10) foi mapeada em um estado OR composto por dois subestados: um atômico denominado "Roteiro_Atividade" que fornece o acesso direto aos cinco subestados do estado "Instância_Atividade": "Projetos", "Evento", "Lazer_Contemplativo", "Educação_Ambiental" e "Preservação".

DECISÃO 2:

Os atributos das fatias constituirão as unidades de informação do hiperdocumento e podem ser mapeados de forma isolada ou agrupada, isto é, podem ser apresentados em separado ou inter-relacionados. Por exemplo, na Figura 7.8 são ilustradas as unidades de informação básicas armazenadas no

conteúdo da página associada ao estado *default* ("HomePage") formado por uma decomposição AND ("Ident_Texto" e "Ident_Vídeo"), cujos estados estão associados à fatia "Identificação" do "Parque_Ecológico".

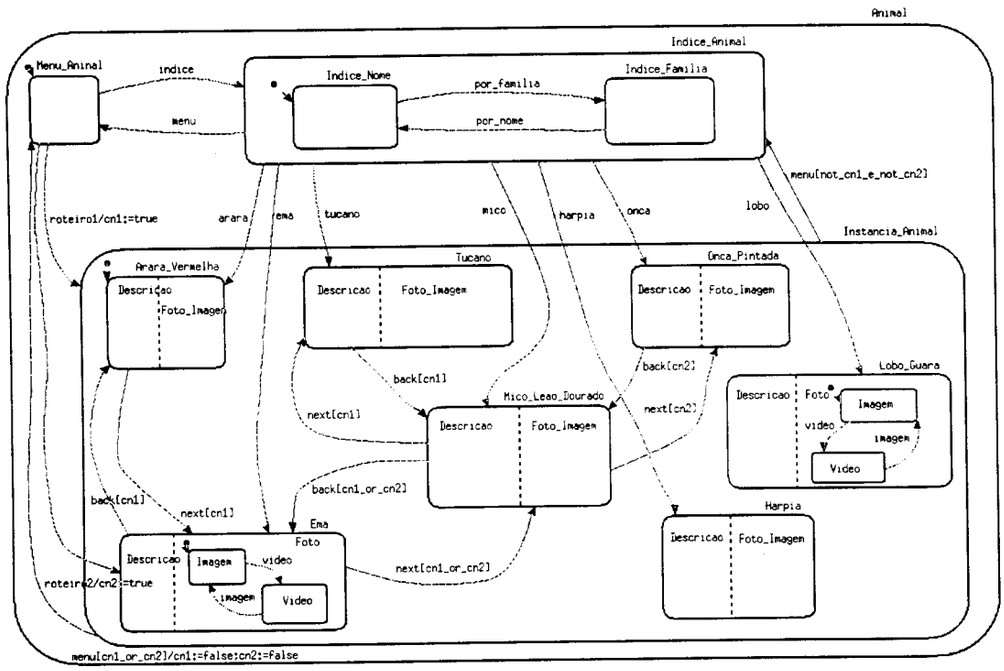


Figura 7.9: Modelo navegacional de instância do subestado "Animal".

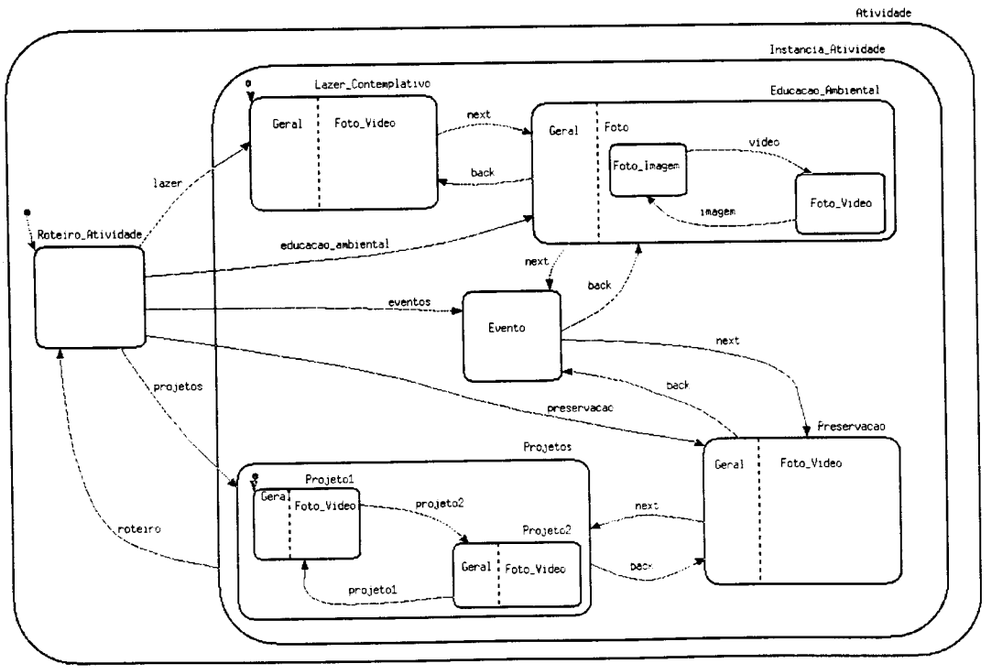


Figura 7.10: Modelo navegacional de instância do subestado "Atividade".

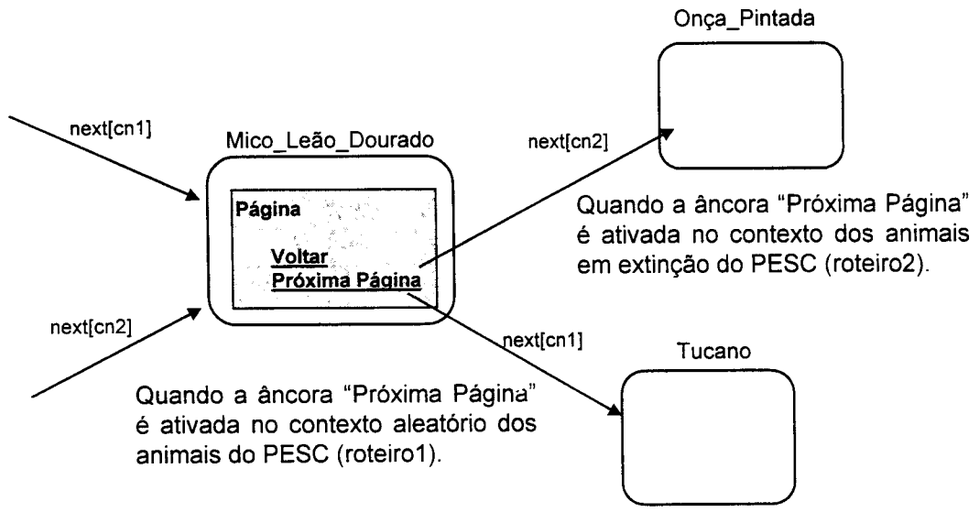


Figura 7.11: Especificação de um estado compartilhado por contextos navegacionais distintos.

Os atributos da fatia "Endereço" da classe "Parque_Ecológico" (Figura 7.5) foram representados por três subestados paralelos do estado "Endereco" (Figura 7.8), no qual foi mapeada a fatia "Endereço": "End_Texto", "End_Imagem" e "End_Áudio". A página associada ao estado "End_Texto" deve conter informações relativas aos atributos "nome:String" e "endereço:Texto", "End_Imagem" refere-se ao atributo "endereço:Imagem" e "End_Áudio" ao atributo "endereço:Audio"

DECISÃO 3:

As estruturas de acesso devem ser mapeadas para estados isolados ou acopladas a estados já especificados. Por exemplo, o roteiro guiado indexado "Roteiro_Atividade" da Figura 7.5 foi mapeado como subestado do estado "Atividade" (Figura 7.10). O disparo do evento "preservação" ativa o estado "Preservacao" e, conseqüentemente, habilita os eventos "next" e "back". Cada evento especificado no modelo navegacional de instância pode ser associado a um objeto âncora do HMBS pertencente à página. A tarefa de especificar os objetos navegacionais do HMBS será realizada na fase seguinte do processo de desenvolvimento, denominada projeto de interface.

7.3.3. Projeto de Interface

Uma vez especificada a estrutura organizacional e navegacional do

hiperdocumento em Statechart, o autor inicia as tarefas em nível de autoria em ponto pequeno, especificando os objetos navegacionais (página, âncora, ligação) e os objetos de apresentação (canais) que definem a aparência dos mesmos.

Durante esta fase, as páginas de informação são projetadas a partir dos atributos das fatias que originaram os estados. O projeto da interface inclui o projeto do leiaute das janelas lógicas para toda página do hiperdocumento, incluindo a aparência dos botões, âncoras, conteúdo e a localização de ajudas navegacionais. O nível de visibilidade do hiperdocumento, que define a profundidade da hierarquia quando as páginas são exibidas durante a navegação também deve ser especificado pelo autor. No fim dessa fase a estrutura navegacional do hiperdocumento estará completamente especificada pelo HMBS, independentemente da plataforma de implementação. Estudos poderão ser realizados em trabalhos futuros para analisar quais tarefas desta fase podem ser parcialmente automatizadas por uma ferramenta computacional.

Para exemplificar, a Figura 7.12 ilustra o resultado do projeto de interface para o estado *default* "HomePage". Duas páginas devem ser especificadas, uma associada ao estado "Ident_Texto" e outra ao estado "Ident_Vídeo". A Figura 7.12(a) apresenta o leiaute esquemático da janela lógica associada ao estado "Ident_Texto", que contém as informações: título "Home"; o conteúdo da página é do tipo texto e tem como informação básica o nome do parque e um menu para outras páginas; os pontos de origem das ligações são especificados por três âncoras ("Sobre_o_Parque", "Atividades" e "Animais") que correspondem às transições cuja origem é o estado "HomePage". A Figura 7.12 (b) ilustra o leiaute para a janela lógica associada ao estado "Ident_Vídeo" do tipo vídeo. Assim, para todos os estados do tipo XOR e atômicos do modelo navegacional de instância considerados relevantes à navegação deve-se projetar sua interface.

7.3.4. Implementação e Avaliação

Para produzir um hiperdocumento navegável, o autor deve mapear o modelo navegacional de instância e os objetos especificados no projeto da interface em objetos e/ou dispositivos de interface do ambiente de implementação escolhido. Em particular, duas abordagens distintas para realizar tal tarefa foram discutidas no Capítulo 6: suporte interpretado e traduzido. A abordagem de suporte interpretado foi apoiada no protótipo do sistema HySCharts, que permite criar, interpretar e

executar uma estrutura navegacional em HMBS.

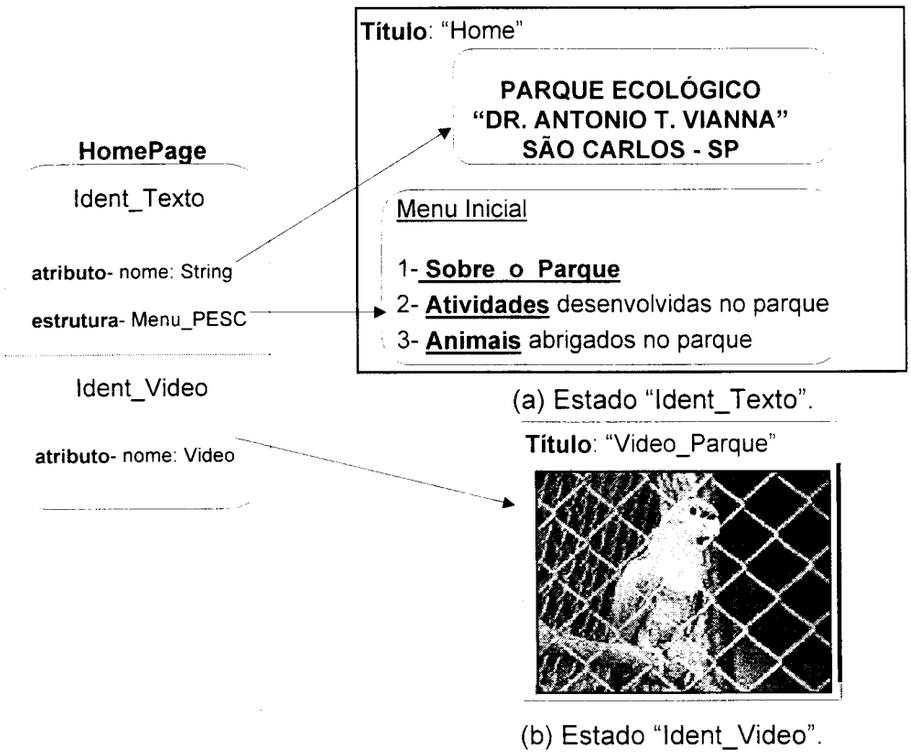


Figura 7.12: Projeto de Interface do estado "HomePage".

Para verificar a potencialidade do suporte traduzido, implementou-se, a título de ilustração, uma versão do PESC em HTML disponível para a WWW. A partir da estrutura navegacional em HMBS e da versão executada usando o sistema HySCharts, avaliações e manutenções corretivas na especificação foram realizadas e uma versão em HTML foi gerada. Estudos para produzir regras de conversão da estrutura navegacional HMBS em páginas HTML podem ser investigados em trabalhos futuros.

A estrutura navegacional completa do PESC foi apresentada no decorrer deste capítulo (Figuras 7.8, 7.9 e 7.10). Definida essa estrutura, após a especificação dos objetos navegacionais e de apresentação do HMBS, o hiperdocumento está disponível para navegação. A seguir serão apresentadas algumas situações durante o processo de navegação no sistema HySCharts.

Implementando as especificações resultantes do projeto de interface, como ilustrado na Figura 7.12, no sistema HySCharts pode-se obter a página inicial mostrada na Figura 7.13(a). Uma versão dessa mesma apresentação traduzida em páginas HTML na WWW pode ser ilustrada na Figura 7.13(b). Dois *frames* foram

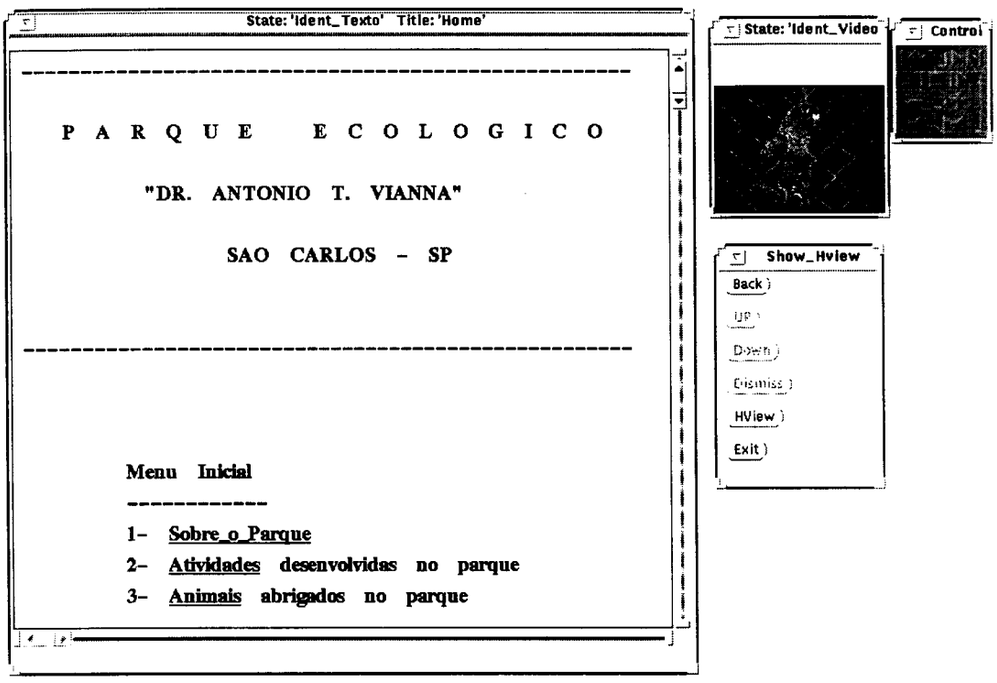
especificados: um para a página associada ao estado "Ind_Texto" (informação textual) e outro para o estado "Ind_Vídeo" (um vídeo).

Se o leitor selecionar a âncora "Sobre_o_Parque" na janela ilustrada na Figura 7.13 (a), o evento "sobre_o_parque" associado à âncora é ativado e a transição correspondente é disparada (veja situação na Figura 7.8). A próxima janela de apresentação é ilustrada na Figura 7.14. A Figura 7.14(a) ilustra a versão do PESC no HySCharts e a Figura 7.14(b) na WWW. Selecionando a âncora "Mapa" na Figura 7.14(a) a janela subseqüente é ilustrada na Figura 7.15. A versão no HySCharts ilustrada na Figura 7.15(a) apresenta três janelas lógicas e a versão na WWW possui três *frames* associados às janelas lógicas. Um outro *frame* contém as âncoras a serem ativadas. Finalmente, selecionando a âncora "Animais" no sistema HySCharts obtém-se a janela de apresentação ilustrada na Figura 7.16.

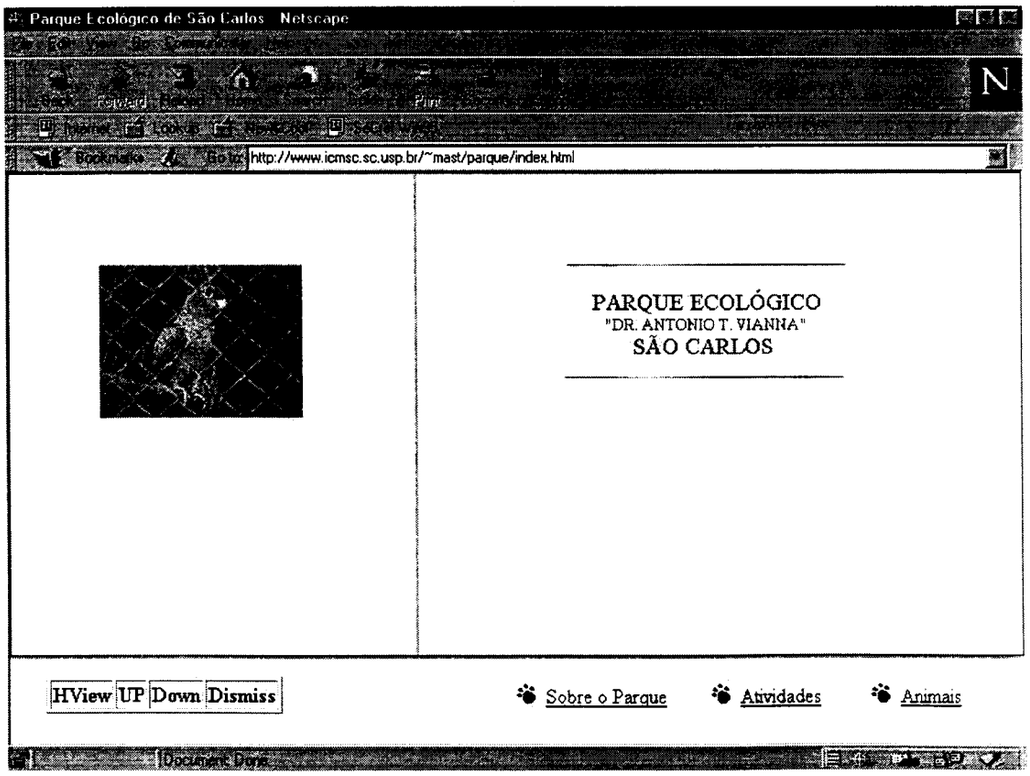
7.4. CONSIDERAÇÕES FINAIS

Neste capítulo foi delineada uma abordagem de projeto sistemática que utiliza o modelo HMBS e o sistema HysCHarts para obter a especificação da estrutura organizacional e navegacional do hiperdocumento. A abordagem de projeto apresentada utiliza conceitos e modelos definidos por outros métodos, como o RMM e o OOHDM, além de integrar as características do modelo HMBS. Tal abordagem é discutida no contexto da aplicação hipermídia desenvolvida para o Parque Ecológico de São Carlos.

No próximo capítulo é apresentada uma análise resumida comparando outros modelos e métodos de projeto para a especificação de hiperdocumentos com o HMBS. Tal análise é essencial para mostrar a relevância do modelo HMBS, suas vantagens e desvantagens.

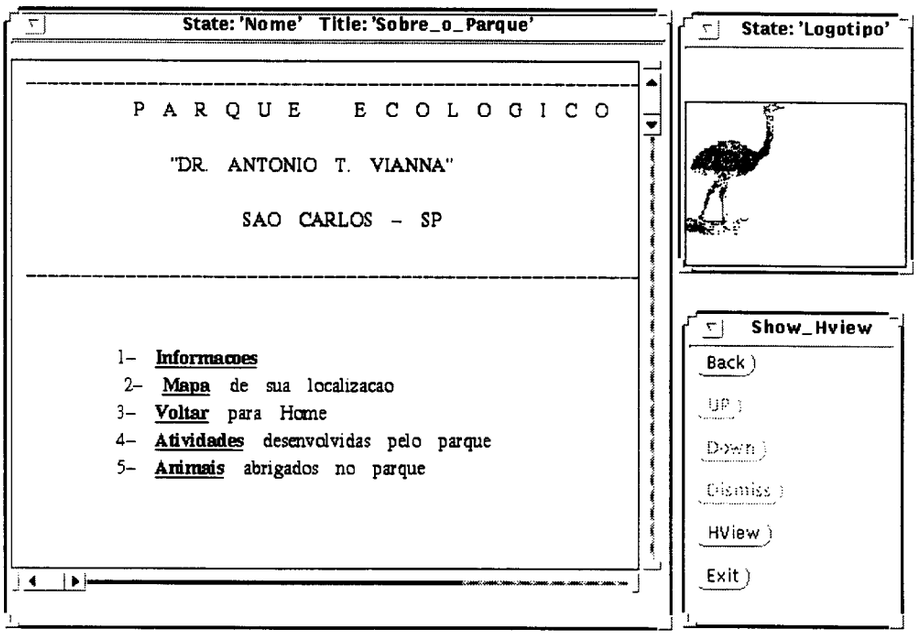


(a) Plataforma: Sistema HySCharts.

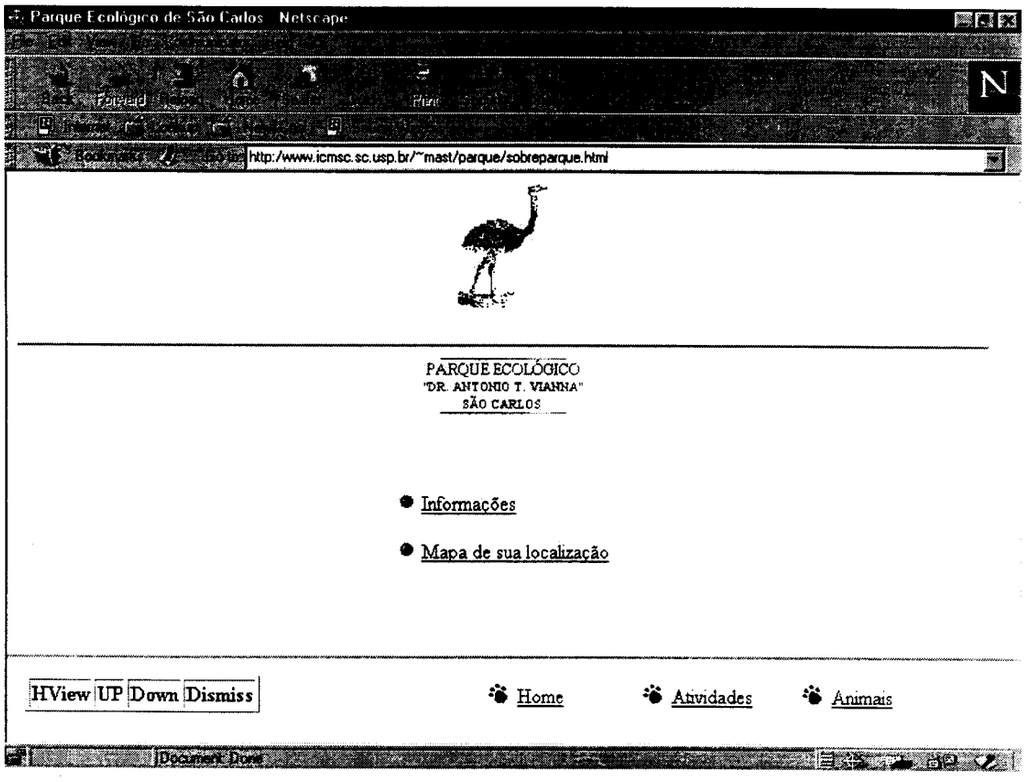


(b) Plataforma: WWW.

Figura 7.13: Implementação da interface do estado "HomePage" da Figura 7.12.

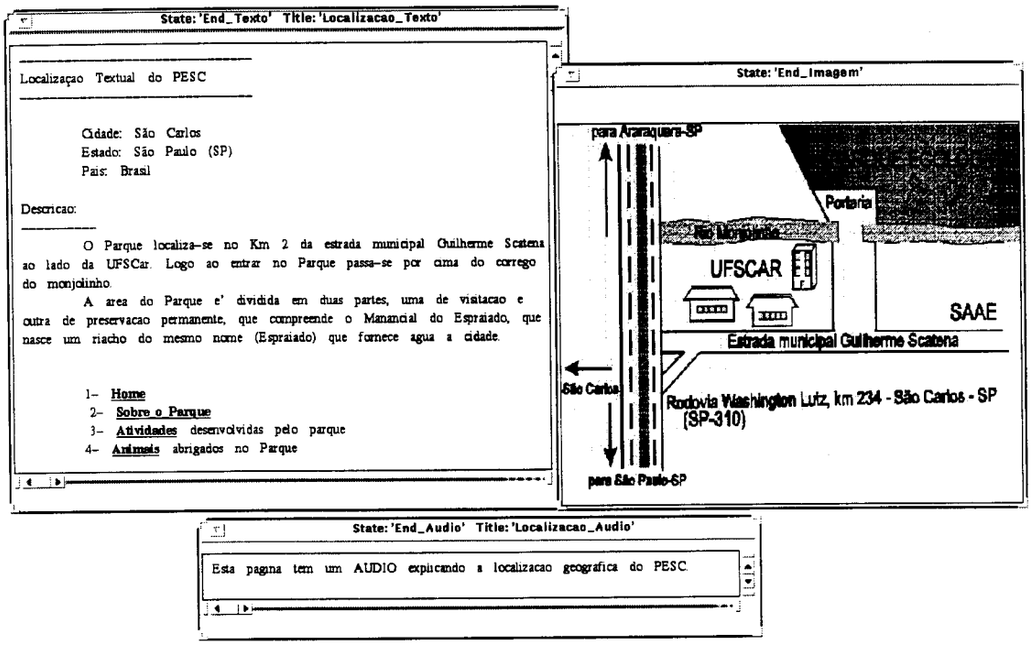


(a) Plataforma: Sistema HySCharts.

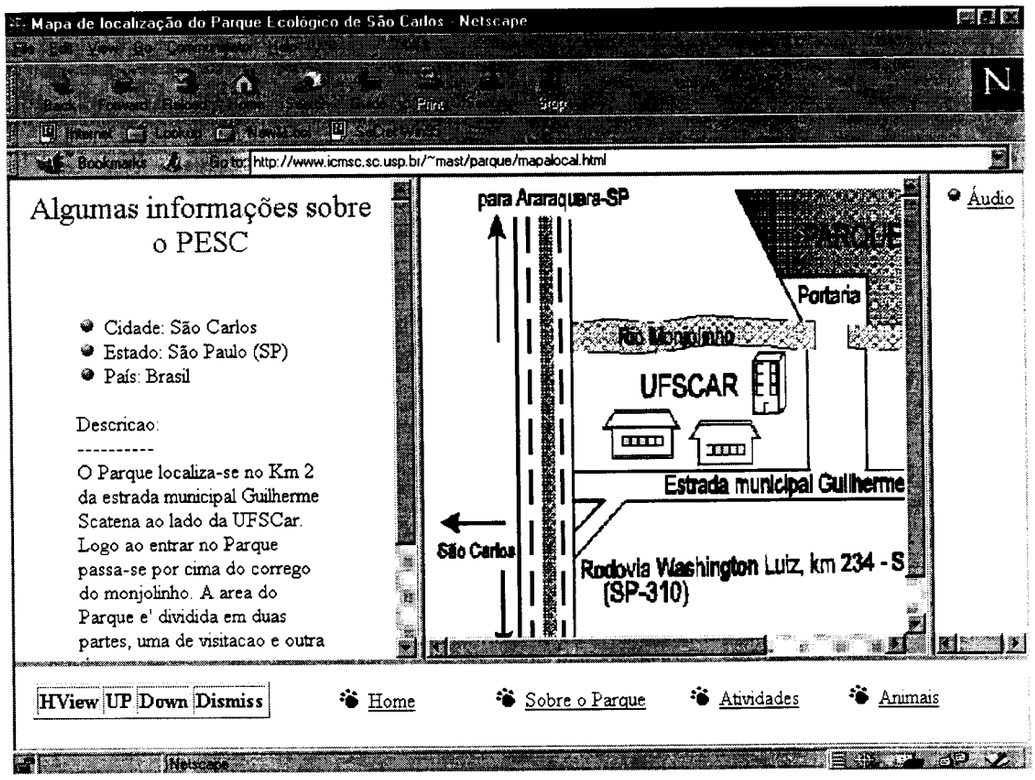


(b) Plataforma: WWW.

Figura 7.14: Janela de apresentação resultante da ativação da âncora "Sobre_o_Parque" da Figura 7.13.

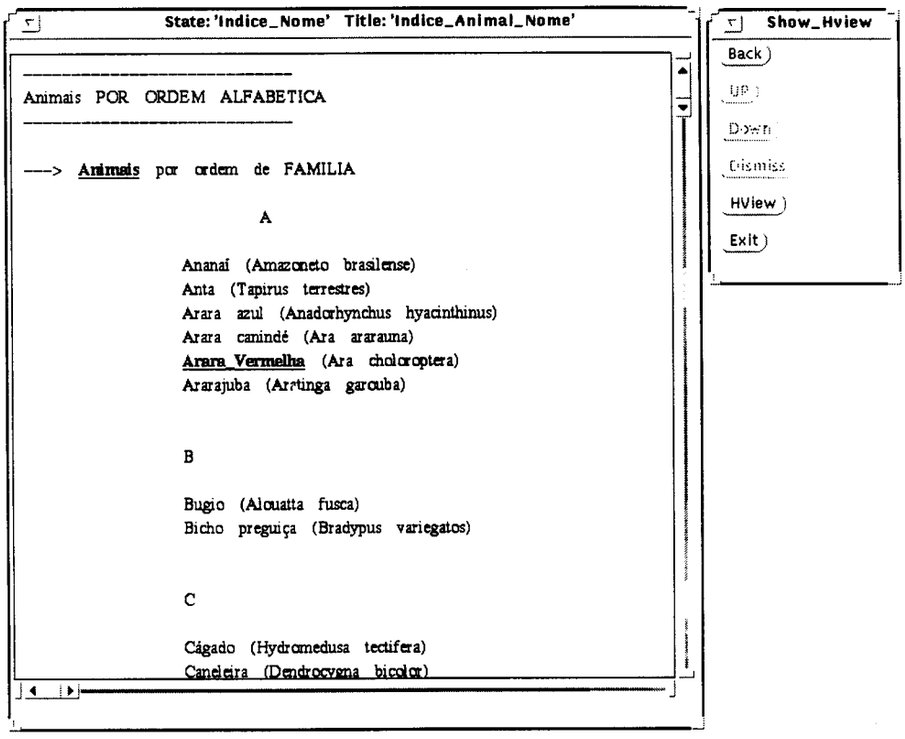


(a) Plataforma: Sistema HySCharts.

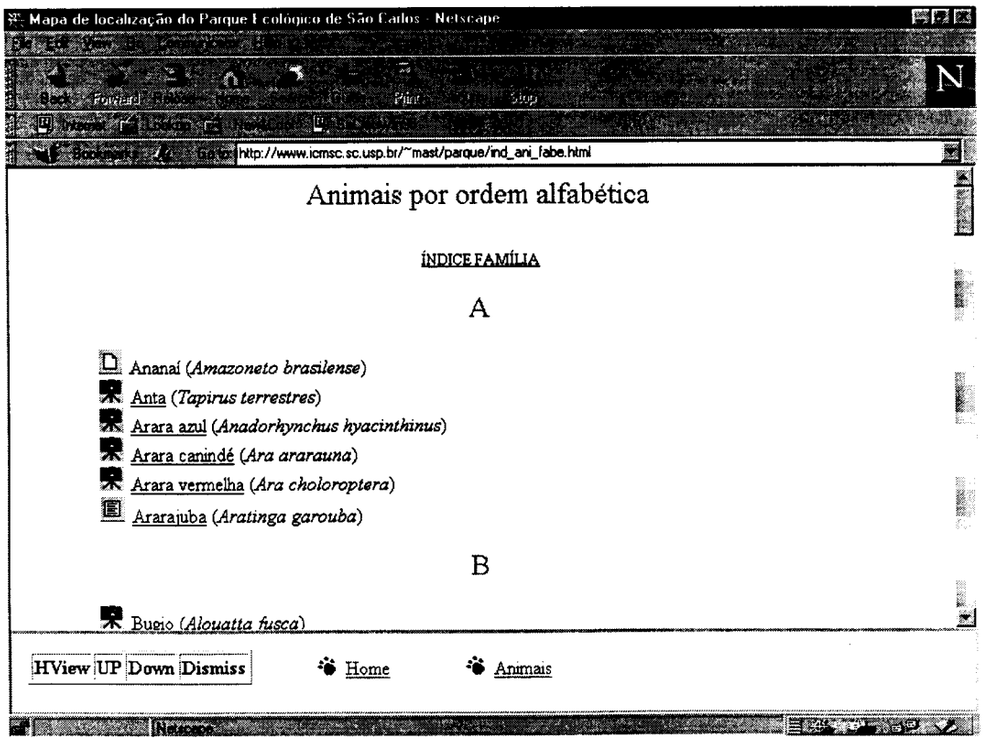


(b) Plataforma: WWW.

Figura 7.15: Janela de apresentação resultante da ativação da âncora "Mapa" da Figura 7.14.



(a) Plataforma: Sistema HySCharts.



(b) Plataforma: WWW.

Figura 7.16: Janela resultante da ativação da âncora "Animais" da Figura 7.15.

Capítulo 8

TRABALHOS RELACIONADOS: UMA ANÁLISE COMPARATIVA

8.1. CONSIDERAÇÕES INICIAIS

A revisão bibliográfica realizada nesta tese deixa claro que as diferentes características e objetivos dos modelos destinados à especificação e ao projeto de hiperdocumentos dificultam uma análise comparativa. Este capítulo tem como objetivo situar o modelo HMBS em relação a outros modelos de referência para hiperdocumentos (apresentados no Capítulo 2), enfatizando os benefícios do modelo proposto, bem como suas desvantagens. Na Seção 8.2 é apresentada uma discussão sobre os modelos e métodos de projeto, comparando-os, quando pertinente, ao HMBS. Na Seção 8.3 algumas características são identificadas e usadas para estabelecer uma análise comparativa entre os modelos estruturais (dígrafos e MCA), os comportamentais (Trellis, HMBS) e o modelo RMDM do método RMM. Na Seção 8.4 são apresentadas as vantagens e limitações do HMBS para a modelagem de hiperdocumentos. Na Seção 8.5 é apresentada uma extensão do HMBS que trata da sincronização temporal de dados multimídia. As conclusões deste capítulo são apresentadas na Seção 8.6.

8.2. UMA DISCUSSÃO

Tompa [1989] e Parunak [1989] declaram que os modelos baseados em dígrafos, embora extremamente simples, não são adequados para a modelagem de hiperdocumentos porque permitem apenas a representação de relações binárias entre os nós de informação. Além disso, tais modelos não fornecem mecanismos para separar a estrutura organizacional (rede de nós e ligações) de seu conteúdo, ignoram padrões de compartilhamento entre nós (composição de nós) e ligações, não permitem especificar estruturas hierárquicas, mecanismos de concorrência e sincronismo de informações.

Parunak [1991] mostrou que a relação de hierarquia é fundamental para a representação de nós compostos, podendo ser modelada utilizando conceitos da teoria dos conjuntos como os diagramas de *Venn/Euler*, que permitem representar coleções de componentes estabelecendo uma hierarquia entre eles.

Para minimizar as deficiências de dígrafos, Tompa [1989] propôs um formalismo baseado em hipergrafos para modelar estruturas genéricas de hiperdocumentos. Os hipergrafos permitem identificar propriedades comuns aos componentes (nós, ligações e rótulos), bem como referenciar diretamente um "grupo de nós" (hipernó) por uma ligação semântica comum (hiperarco). Assim como o HMBS, o modelo permite a separação entre a estrutura e o conteúdo do hiperdocumento. Isso é alcançado pela especificação de novos objetos conceituais e dos seus respectivos mapeamentos, permitindo definir uma semântica de navegação orientada a conjunto.

Diferentemente do HMBS, o comportamento dinâmico de dígrafos e hipergrafos é especificado pelo posicionamento e movimentação de marcadores nos nós. Essa não é uma solução satisfatória, pois a abordagem de marcação é muito geral e não permite especificar restrições no processo de marcação.

Finalmente, é importante observar que, embora os modelos baseados em hipergrafos sejam relevantes para a expressão de relações n-árias entre os componentes, eles não oferecem recursos para especificar relações de hierarquia entre tais componentes. Ademais, a representação gráfica associada aos hipergrafos não torna claro o comportamento dinâmico do hiperdocumento.

Outro modelo estrutural relevante à comparação com o HMBS é o Modelo de Contextos Aninhados (MCA) [Casanova et al. 1991]. Ambos os modelos, HMBS e MCA, permitem especificar a estrutura organizacional hierárquica de hiperdocumentos abordando o processo de autoria em vários níveis de abstração e granularidade. Para isso, o MCA utiliza contextos aninhados (composição de nós de

contexto), enquanto que o HMBS herda da técnica Statecharts a decomposição de estados. A representação gráfica do hiperdocumento em HMBS torna explícito o tipo de composição (paralela ou exclusiva), enquanto o MCA permite definir tal situação implicitamente nas ligações entre os nós. O HMBS possui uma semântica de navegação bem definida e formal. Por outro lado, o MCA parece ser de assimilação mais simples do que o HMBS, que exige dos seus usuários familiaridade com a notação e a gramática do formalismo Statecharts. Atualmente, o MCA é também mais abrangente, pois permite especificar sincronização de dados multimídia em conformidade com o padrão MHEG, apóia o controle e o gerenciamento de versões e trabalho cooperativo, características que podem ser incorporadas ao HMBS em trabalhos futuros, como sugerido no Capítulo 9.

Com relação aos modelos comportamentais, o Trellis [Stotts; Furuta 1989] permite a especificação formal de hiperdocumentos utilizando Redes de Petri, que são precisas, têm uma representação gráfica bem definida e são fortemente dirigidas a evento. Entretanto, sua principal desvantagem é a ausência de facilidades para a decomposição hierárquica, que dificulta a tarefa de especificação de sincronização em diferentes níveis da hierarquia do hiperdocumento. A hierarquia em Trellis pode ser especificada considerando o conteúdo (C_L) de um lugar como uma outra Rede de Petri. Porém, o modelo não tem associada uma semântica de navegação quando a estrutura é hierárquica e, além disso, tal solução não representa explicitamente os níveis de decomposição hierárquica. Por outro lado, o formalismo de Statecharts subjacente ao HMBS permite a decomposição de estados em subestados que são também Statecharts e se comportam como tal.

Como ilustração, na Figura 8.2 é apresentado o mesmo exemplo de modelagem descrito na Figura 8.1 utilizando uma Rede de Petri hierárquica. Por exemplo, os lugares "p_7", "p_8" e "p_9" da Figura 8.1 foram substituídos pelo lugar "p_{03}" e representados em um outro nível da Rede de Petri (Figura 8.2). A Rede de Petri hierárquica pode ser modelada por um Statechart como ilustrado na Figura 8.3. Comparando ambas as representações, a Rede de Petri da Figura 8.2 e o Statechart da Figura 8.3, observa-se que a especificação em Statechart é mais intuitiva e clara, pois fornece facilidades de estruturação hierárquica inerentes ao modelo. Na Figura 8.2, uma decomposição do lugar "p_3" não é facilmente descrita com uma Rede de Petri e não há mecanismos para especificar sincronização, transições e concorrência dentro dessa decomposição. Entretanto, tal decomposição em Statecharts (Figura 8.3) pode ser diretamente mapeada em

subestados do estado "p_3", sendo que qualquer nível adicional de decomposição estrutural é facilmente representado e tem uma semântica comportamental associada.

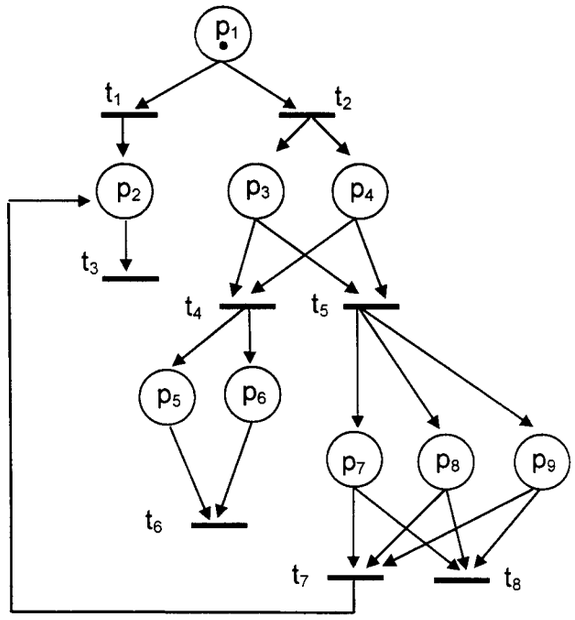


Figura 8.1: Rede de Petri (sem hierarquia) que especifica concorrência [Stotts; Furuta 1989, p.18].

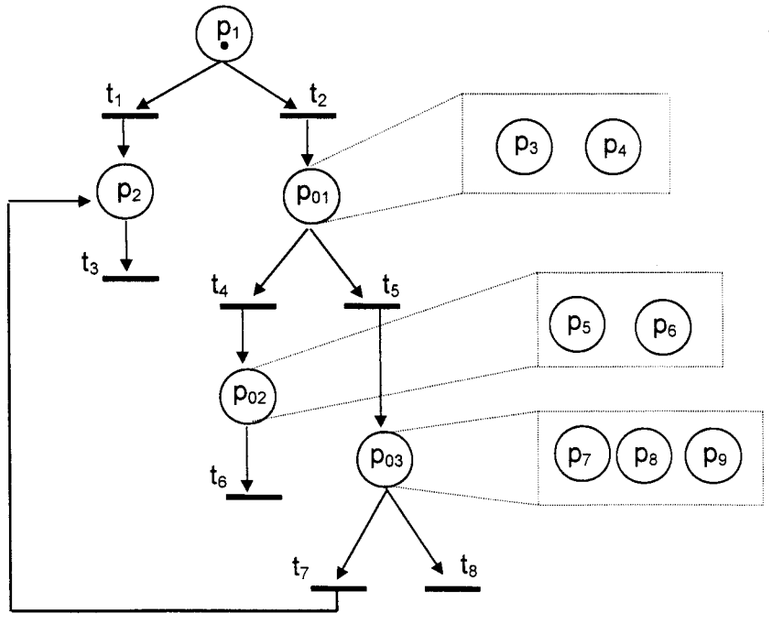


Figura 8.2: Rede de Petri (com hierarquia) da Figura 8.1 que especifica concorrência [Stotts; Furuta 1989, p.18].

De acordo com Zheng e Pong [1992], o comportamento definido pelo uso de marcadores em lugares da Rede de Petri não é facilmente visível a partir da representação gráfica estática da rede e, portanto, não é intuitivo especificar e compreender a semântica de navegação de hiperdocumentos utilizando Redes de Petri. Por outro lado, a semântica de execução de Statecharts não usa marcadores e pode ser facilmente compreendida a partir da representação gráfica. Assim, acredita-se que o HMBS é um modelo mais intuitivo e mais completo do que o Trellis para especificar a semântica de navegação de hiperdocumentos.

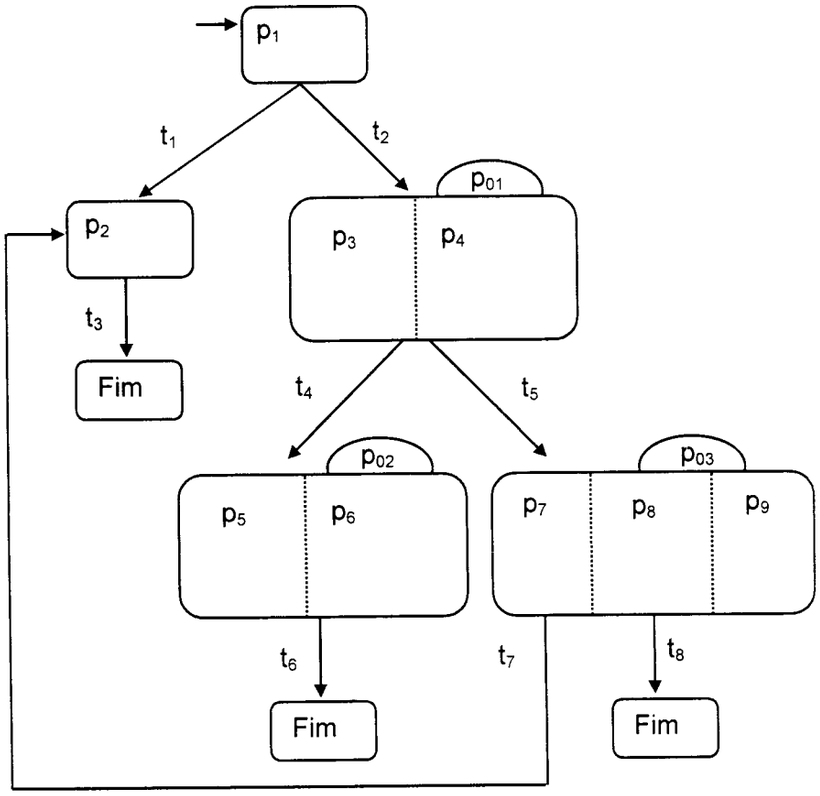


Figura 8.3: Statechart com estados concorrentes, equivalente a Rede de Petri ilustrada na Figura 8.2.

Como o HMBS, Zheng e Pong [1992] também propõem a utilização de Statecharts na modelagem de hiperdocumentos, mas se preocupam, principalmente, com a especificação dos aspectos comportamentais pertinentes à interface de sistemas hiperdocumento. Essa abordagem é interessante, mas não é comparável com a abordagem desta tese, na qual os Statecharts foram empregados no contexto de modelagem da estrutura (organizacional e navegacional) e da semântica de navegação de hiperdocumentos.

Ao contrário do HDM, do RMM, do OOHDM e do EORM, apresentados no Capítulo 2, o HMBS não é um método de projeto, ainda que, como apresentado no Capítulo 7, ele possa ser integrado a uma abordagem de projeto de mais alto nível para apoiar o processo de desenvolvimento de hiperdocumentos. Desse modo, não é razoável estabelecer uma comparação direta entre o HMBS e esses métodos, mas algumas observações são pertinentes quanto ao uso do modelo HMBS no contexto de métodos de projeto.

O HDM [Garzotto et al. 1993a] concentra-se, principalmente, na modelagem semântica e nos aspectos representacionais do modelo, sem muita ênfase em aspectos comportamentais do hiperdocumento. O modelo não fixa, a *priori*, o domínio da aplicação e, desse modo, suas primitivas de representação são genéricas, permitindo o projeto de aplicações em diversos domínios. O HMBS também não restringe o domínio da aplicação, porém é particularmente adequado para modelar aplicações que apresentam explicitamente uma estrutura hierárquica, tais como livros, artigos científicos e manuais on-line. Uma diferença fundamental entre os modelos é que o HDM se baseia na definição de um esquema que pode ser instanciado para classes de documentos do mesmo tipo, enquanto que o HMBS modela a estrutura organizacional de cada hiperdocumento em particular. Apesar do HMBS não utilizar os conceitos de esquema e instância de esquema, nada impede que a especificação seja compartilhada por diferentes hiperdocumentos, ou por diferentes visões do mesmo hiperdocumento.

Tanto o modelo de dados do método RMM [Isakowitz et al. 1995], denominado RMDM, quanto o HMBS, permitem especificar a estrutura e as características navegacionais do hiperdocumento. Porém, o RMDM não especifica o comportamento dinâmico, isto é, as transformações navegacionais dinâmicas do hiperdocumento.

Do ponto de vista da notação e terminologia, o modelo de dados do HDM e o RMDM são similares ao MER, enquanto que o HMBS é baseado em Statecharts. Porém, Harel [1988] mostrou que tanto Statecharts como os MERs são baseados em um conceito básico comum denominado *higrafo*[20] (*high graphs* e *hierarchical graphs*). Assim, do ponto de vista de notação e terminologia os modelos compartilham conceitos comuns.

---

[20] O *higrafo* é um formalismo topológico-visual que combina noções de conjuntos (representadas pelos diagramas de *Venn/Euler*) e hipergrafos (conjuntos de elementos e relação entre eles).

Nos métodos de projeto, modelos são propostos e/ou enriquecidos nas diferentes fases de desenvolvimento. A modelagem do domínio da aplicação, vista como a capacidade de apoiar a modelagem semântica ou lógica da aplicação, não é satisfeita pelo HMBS. O RMDM utiliza o MER para a modelagem conceitual, e o OOHDM [Schwabe et al. 1996] e o EORM [Lange 1994] utilizam o modelo de dados orientado a objetos OMT, que oferece mecanismos de abstração mais ricos. A abordagem de projeto baseada no HMBS sugerida no Capítulo 7 propõe, na fase de modelagem conceitual, a utilização do modelo de objetos do método orientado a objetos *Fusion*, porém um outro modelo poderia ser utilizado.

Por ser um método de projeto, a abordagem proposta pelo OOHDM é mais ampla que a do Trellis, de Zheng e Pong e do HMBS. O OOHDM modela não apenas o comportamento navegacional, mas também o comportamento da interface. O projeto navegacional não é visto somente como um processo de enriquecimento do esquema conceitual com estruturas de acesso, como no RMM, mas também de definição de visões sobre o esquema. No OOHDM são especificadas as classes navegacionais e os contextos navegacionais nos quais o leitor poderá navegar. O comportamento dinâmico, ou seja, as transformações navegacionais no espaço navegacional são especificadas utilizando Diagramas de Navegação, que são facilmente mapeados para Statecharts.

O projeto de interface abstrata em OOHDM [Rossi et al. 1995] utiliza o formalismo visual ADVcharts para especificar transformações de interface (aspectos dinâmicos, como atributos que podem aparecer ou desaparecer de acordo com uma decisão de controle, por exemplo, um menu *pop-up*). O modelo HMBS não tem por objetivo especificar o comportamento dinâmico da interface do sistema hiperdocumento. Porém, como proposto por Zheng e Pong, o formalismo de Statecharts pode também ser utilizado no projeto de interface. Tal abordagem mostra que o poder de especificação de Statecharts pode ser útil tanto na modelagem estrutural e navegacional de hiperdocumentos quanto na modelagem de interface de sistemas hiperdocumento.

8.3. AS CARACTERÍSTICAS DE COMPARAÇÃO

Na Tabela 8.1 é apresentada uma comparação do HMBS com outros modelos afins na literatura, segundo as características definidas a seguir:

a) Classe de modelagem: Considera a classe de modelagem na qual se enquadra o modelo. Conforme apresentado no Capítulo 2, pode-se ter

modelos fechados, orientados ao sistema, estruturais e comportamentais, além dos métodos de projeto;

b) Formalismo de apoio: Considera o formalismo Matemático utilizado para representar a estrutura do hiperdocumento (por exemplo, grafos, Redes de Petri, Statecharts e MEFs);

c) Autoria em ponto grande: Considera o tipo de suporte oferecido pelo modelo para especificar a estrutura organizacional do hiperdocumento na autoria em ponto grande;

d) Estruturação da informação: Considera quais tipos básicos de estruturas (linear, rede e hierárquica [Ginige et al. 1995]) são utilizados pelo modelo para organizar os hiperdocumentos;

e) Decomposição hierárquica: Considera se o modelo permite especificar o hiperdocumento em diferentes níveis de abstração, facilitando a tarefa de decomposição do problema;

f) Separação entre conteúdo e estrutura: Considera se o modelo permite separar a estrutura do conteúdo, de forma que uma modificação no conteúdo não demande alterações sobre a estrutura definida anteriormente;

g) Ligações temporais: Considera se o modelo permite especificar ligações a serem ativadas automaticamente, caso o leitor não ative uma ligação específica dentro de um intervalo de tempo determinado;

h) Prototipação: Considera se, após a especificação da estrutura do hiperdocumento, o modelo pode se tornar disponível rapidamente em forma de protótipo para os autores e os leitores verificarem sua adequação e facilidades de uso;

i) Semântica de navegação: Considera a capacidade do modelo em definir de maneira rigorosa as propriedades dinâmicas e de visualização do hiperdocumento em tempo de navegação;

j) Sistema de apoio: Considera se existe um sistema que apóia a utilização do modelo, ou seja, que permita criar, interpretar e executar especificações do hiperdocumento segundo as primitivas subjacentes ao modelo; e

k) Análise e validação: Considera se o modelo possui algoritmos definidos sobre uma base formal que permitem analisar e validar o hiperdocumento para detectar inconsistências e anomalias tanto na sua estrutura quanto no seu comportamento dinâmico.

Tabela 8.1: Comparação entre os modelos.

Modelo / Requisito	Dígrafo	MCA	Trellis	HMBS	RMDM
Classe de modelagem	Estrutural	Estrutural	Comportamental	Estrutural / Comporta-mental	Estrutural
Formalismo	Grafos	Hipergrafo	Redes de Petri	Statecharts	MER
Autoria em ponto grande	Nós e ligações	Hierarquia de classes	Lugares, transições e arcos	Estados, transições e eventos	Entidades, atributos, relaciona-mentos; fatias; primitivas de acesso
Estruturação da informação	Linear e Rede	Linear, Hierárquica e Rede	Linear e Rede	Linear, Hierárquica e Rede	Linear, Rede
Decomposição hierárquica	Não	Nós do tipo composição	Não	Decomposição OR e AND	Não
Separação entre estrutura e conteúdo	Não	Hierarquia das classes; conteúdo; lista de âncoras de cada nó	Rede de Petri; conjunto de conteúdo; botões; mapeamentos	Statechart; páginas; âncoras; mapeamentos	Não
Ligações temporais	Não	Sim	Não	Não	Não
Prototipação	Não	Não	Simulação da Rede de Petri subjacente	Simulação do Statechart subjacente	Não
Semântica de navegação	Não	Nó *default* e trilha	Semântica operacional de Redes de Petri	Semântica operacional de Statecharts	Agrupa-mento inicial
Sistema de apoio	Não	HyperProp	∝Trellis	HySCharts	RMCase
Análise e Validação	Não	Não	Algoritmos fundamentados em Redes de Petri	Algoritmos fundamentados em Statecharts	Não

8.4. VANTAGENS E DESVANTAGENS

As vantagens da utilização de abordagens tradicionais de engenharia de

software no projeto e desenvolvimento de hiperdocumentos já foram demonstradas [Garzotto et al. 1991; Garzotto et al. 1993a; Balasubramanian et al. 1997]. A concepção do modelo HMBS vai ao encontro dessa idéia, permitindo especificar de maneira sistemática e formal a estrutura organizacional e a semântica de navegação de hiperdocumentos. Os benefícios proporcionados pelo modelo podem ser agrupados em duas classes: aqueles inerentes ao formalismo subjacente ao HMBS e aqueles cujas características são introduzidas no próprio modelo.

Os benefícios decorrentes da utilização do formalismo Statecharts na especificação de hiperdocumentos são:

a) Tem-se um modelo intuitivo para profissionais da área de computação, pois é baseado em Statecharts, uma extensão de MEFs e a modelagem baseada em estados e eventos é bem aceita e difundida na comunidade de Ciência da Computação;
b) Existem recursos que permitem especificar facilmente aspectos de concorrência, pois os Statecharts foram projetados para modelar sistemas reativos concorrentes e permitir decomposições paralelas e seqüenciais com semântica associada;
c) Existem ferramentas e algoritmos que permitem identificar anomalias na estrutura navegacional, bem como detectar caminhos de navegação redundantes e faltosos;
d) Existe uma representação visual da estrutura do hiperdocumento. Segundo Harel [1992], formalismos visuais podem ter um grande impacto positivo em aplicações interativas;
e) Encoraja a prototipação, pois é possível "executar" a especificação HMBS;
f) Encoraja uma maior portabilidade de hiperdocumentos entre os vários sistemas e ambientes existentes; e
g) Tem-se um modelo com sintaxe e semântica formalmente definidas, possibilitando ao autor analisar a especificação do hiperdocumento, bem como verificar e provar a consistência interna e a corretitude da sintaxe da especificação.

Os benefícios do modelo HMBS são:

a) Permite separar a estrutura organizacional, representada em Statecharts, do conteúdo do hiperdocumento, representado pelos objetos navegacionais;
b) Fornece facilidades, como a operação "Mostra_VisãoH", para navegar pela estrutura hierárquica do hiperdocumento. O modelo é adequado para

especificar hiperdocumentos organizados hierarquicamente e que apóiam a "navegação estruturada". Por exemplo, aplicações orientadas a auxiliar o processo de ensino/aprendizagem, de forma que o leitor navega ou explora o hiperdocumento com a meta de ganhar conhecimento sobre um certo tópico ou assunto ("leitura para compreensão" Thüring et al. [1995]), ao invés de simplesmente localizar uma informação;

c) Fornece uma interpretação coerente da decomposição estrutural e da semântica de navegação;
d) Facilita a determinação de erros e ambigüidades na fase de tradução da especificação formal em uma implementação em alguma plataforma destino, como por exemplo, a WWW. Dada uma especificação formal do hiperdocumento e a definição de uma linguagem de *markup*, por exemplo, é possível verificar se a implementação está de acordo com a especificação [Fraser et al. 1994];
e) Possibilita a geração automática de código em outras linguagens e plataformas a partir de especificações HMBS;
f) Possibilita a utilização da especificação na fase de teste como um guia para a geração e identificação de casos de teste apropriados; e
g) Permite coordenação espacial das informações do hiperdocumento, embora simples, utilizando os canais de apresentação.

Alguns problemas inerentes ao modelo HMBS são mencionados a seguir. A principal desvantagem do modelo é a sobrecarga associada à abordagem formal, pois a notação e a gramática de Statecharts requerem familiaridade com Matemática discreta e lógica simbólica, que muitos projetistas, autores e programadores não têm. Porém, outros modelos e métodos, como o Trellis e o OOHDM, também apresentam uma curva de aprendizagem alta: o formalismo de Redes de Petri é básico no Trellis; no OOHDM o autor necessita compreender o método OMT, bem como os Diagramas de Navegação e os ADVcharts, que são dialetos de Statecharts. Desse modo, esse não é um problema exclusivo do HMBS. Outras limitações do modelo são:

a) Não fornece mecanismos para especificar requisitos temporais e de sincronização relativos à apresentação de informações multimídia. Satisfazer a esses requisitos é essencial para a descrição de uma variedade maior de aplicações, visto que, em geral, o domínio de uma aplicação abrange apresentações de dados multimídia, além de dados

puramente estáticos. Na Seção 8.5 é apresentada, resumidamente, uma extensão do HMBS para tratar desses requisitos;

b) Não possui primitivas próprias de modelagem para o gerenciamento de versões[21] de hiperdocumento. O conceito de versões em sistemas hiperdocumento pode estar associado à manutenção de um histórico de cada documento, possibilitando preservar e manipular versões cronológicas dos documentos;

c) O modelo não permite a associação de duas ou mais páginas a um mesmo estado (mapeamento "m"). Tal definição não limita a especificação de hiperdocumentos. Por exemplo, duas visões de informação (uma página em Inglês e Português) podem ser modeladas considerando cada visão como um estado independente, porém a especificação do hiperdocumento em Statecharts se torna volumosa;

d) Evidências empíricas da ciência cognitiva sugerem que nas fases iniciais de resolução de problemas, quando o domínio é relativamente mal estruturado, o uso de linguagens e representações formais inibe a exploração de alternativas e é prejudicial à qualidade do produto [Fraser et al. 1994]. Desse modo, a linguagem de especificação formal, cujos requisitos são definidos formalmente por um vocabulário, sintaxe e semântica, não é uma ferramenta ideal para explorar e descobrir a estrutura do problema durante o processo de análise. Assim, pode ser prejudicial a utilização do HMBS quando se deseja especificar aplicações não estruturadas; e

e) A modelagem semântica, ou conceitual, do domínio de informação não é apoiada pelo HMBS de forma que, para projetar um hiperdocumento, não é suficiente a utilização do HMBS por si só.

8.5. XHMBS: UMA EXTENSÃO DO HMBS

Em sistemas hiperdocumento, a sincronização de informações multimídia é uma característica importante e essencial na apresentação de documentos. O modelo HMBS não inclui, em sua definição original, mecanismos para especificar e gerenciar esse tipo de informação. Entretanto, como parte de um trabalho nessa

---

[21] Segundo Dittrich e Lorie [1988], uma versão ou revisão de um objeto é uma descrição sob uma certa abstração, isto é, um contexto do objeto que o usuário ou o sistema deve conservar.

linha de pesquisa, o HMBS foi estendido para incluir mecanismos relativos à especificação de requisitos temporais e de sincronização de informações multimídia em hiperdocumentos [Paulo et al. 1997, Paulo 1997, Paulo et al. 1998]. Tal modelo foi denominado XHMBS (*e**X**tended **H**yperdocument **M**odel **B**ased on **S**tatecharts*). Entre as inovações introduzidas estão a possibilidade de especificar informações distribuídas utilizando os canais de apresentação e comunicação, a separação total entre o conteúdo e a estrutura do hiperdocumento pelo uso melhorado de âncoras, além da especificação pelo formalismo Hypercharts, que substituem os Statecharts convencionais como técnica de especificação subjacente ao modelo. Qualquer especificação em Hyperchart pode ser transformada em um Statechart que exibe o comportamento equivalente ao do Hyperchart.

Os Hypercharts oferecem um conjunto de notações que permite especificar concisamente, com base na mesma semântica operacional de Statecharts, o comportamento dinâmico de hiperdocumentos complexos, principalmente aqueles envolvendo requisitos de sincronização de dados multimídia em ambientes distribuídos. Nos Hypercharts, além das notações permitidas em Statecharts, são introduzidas notações baseadas no modelo TSPN [Díaz; Sénac 1993, Sénac et al. 1994], indispensáveis para a especificação do comportamento temporal de aplicações hipermídia. Em termos gerais, as extensões são: relógio local associado a estado (armazena o número de passos durante os quais o estado permanece ativo), registrador de história temporal associado a estado (mantém o último instante de ativação de um estado), história temporal associada a transições, transições temporais, mecanismos de sincronização, parametrização de estados e abstração de transições. Assim, o XHMBS permite que especificações de hiperdocumentos complexos sejam realizadas de forma concisa e sem perda de legibilidade.

8.6. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada uma comparação resumida entre o modelo HMBS e os modelos MCA e Trellis, a proposta de Zheng e Pong e os métodos HDM, OOHDM, RMM e EORM para o projeto de hiperdocumentos. A independência da estrutura e do conteúdo do hiperdocumento e a possibilidade de navegação hierárquica e estruturada são algumas das vantagens propiciadas pelo HMBS. Os problemas e limitações do modelo também foram discutidos neste capítulo. No próximo capítulo são apresentadas as conclusões gerais desta tese, apontando as principais contribuições e as sugestões de trabalhos futuros.

UFSC — SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

Capítulo 9

CONCLUSÕES

9.1. CONSIDERAÇÕES INICIAIS

Nesta tese foi apresentado um novo modelo que permite especificar formalmente a estrutura organizacional, o comportamento dinâmico e a semântica de navegação de hiperdocumentos. O modelo proposto é inovador ao sugerir a técnica Statecharts como base para um modelo formal de especificação de hiperdocumentos. Esta tese oferece uma contribuição para o avanço da tecnologia hipermídia ao sugerir um modelo que pode ser integrado a um processo de projeto e desenvolvimento de hiperdocumentos. O formalismo subjacente ao HMBS permite ao autor especificar a estrutura (organizacional e navegacional) e o comportamento de hiperdocumentos de modo preciso e semanticamente rigoroso, encorajando o desenvolvimento de hiperdocumentos estruturados. O modelo possui características e mecanismos que apóiam a autoria de forma estruturada, com facilidades para a análise e a avaliação sistemática dos hiperdocumentos. Em analogia com o processo de desenvolvimento de software, o uso de modelos de especificação é de grande interesse tanto para os projetistas quanto para os usuários de hiperdocumentos, pois permitem reduzir o tempo de desenvolvimento dos produtos e melhorar a sua qualidade, o que se reflete positivamente nas características de usabilidade, confiabilidade e manutenibilidade do produto final.

Neste capítulo são apresentadas as conclusões gerais deste trabalho, enfatizando suas principais contribuições (Seção 9.2) e as sugestões para trabalhos futuros (Seção 9.3).

9.2. CONTRIBUIÇÕES DESTA TESE

Esta tese constitui uma proposta original no que se refere à especificação de hiperdocumentos baseada em um formalismo rigoroso. O HMBS (***H**yperdocument **M**odel **B**ased on **S**tatecharts*) é um modelo formal baseado em Statecharts que permite especificar a estrutura e o comportamento dos hiperdocumentos. Na tese foram expostos os conceitos e a definição formal do modelo (Capítulo 4), estabelecendo como a estrutura e a semântica de execução de Statecharts são utilizadas para especificar a estrutura organizacional e a semântica de navegação de hiperdocumentos. O modelo explora a estruturação hierárquica do hiperdocumento, e pode ser utilizado, particularmente, na modelagem de hiperdocumentos que apresentem estrutura hierárquica, tais como livros, artigos científicos, manuais e relatórios on-line. A hierarquia de estados estabelecida pelo Statecharts subjacente fornece um mecanismo apropriado para modelar organizações hierárquicas.

Adicionalmente, como os Statecharts foram projetados para modelar sistemas reativos, o HMBS possui mecanismos eficientes para a especificação de concorrência na apresentação de múltiplas unidades de informação, e é apoiado por algoritmos para a análise de propriedades de hiperdocumentos (Capítulo 5). As características do modelo, bem como as vantagens e as limitações que o tornam relevante para o desenvolvimento de hiperdocumentos foram discutidas no Capítulo 8. Acredita-se que o modelo HMBS é um progresso em relação ao Trellis (o outro modelo inerentemente comportamental proposto na literatura e discutido no Capítulo 2), sendo mais completo e consistente no apoio oferecido tanto para a especificação da estrutura quanto da semântica de navegação. bem como nos algoritmos de análise de propriedades.

Para provar e validar a possibilidade prática de uso do modelo num contexto real, permitindo ao autor criar, interpretar e executar especificações HMBS de um hiperdocumento e ver rapidamente e visualmente os resultados, foi fundamental o desenvolvimento do ambiente HySCharts. O ambiente possui dois módulos principais: um módulo de autoria que permite a especificação de hiperdocumentos usando os objetos estruturais, navegacionais e de apresentação. e um módulo de navegação que permite interpretar e "executar" as especificações HMBS (Capítulo 6). O HySCharts fornece facilidades de prototipação rápida e simulação interativa da estrutura navegacional subjacente ao hiperdocumento. Permite ainda que propriedades do modelo sejam analisadas, possibilitando que modificações na

estrutura e no conteúdo do hiperdocumento possam ser feitas rapidamente. Adicionalmente, o HySCharts oferece, além da navegação tradicional, um novo mecanismo de navegação hierárquica ("*Show_Hview*") que possibilita ao leitor navegar pelas páginas associadas a estados acima ou abaixo do nível de hierarquia definido pelo autor, contribuindo para minimizar a sua eventual desorientação.

Para ilustrar a possibilidade de integração do modelo HMBS a um processo de projeto abrangente e sistemático, com fases e objetivos bem definidos, a exemplo dos principais métodos de projeto discutidos na literatura, foi delineada no Capítulo 7 uma abordagem composta por quatro fases: projeto conceitual, projeto navegacional, projeto de interface e implementação/avaliação. O modelo proposto foi utilizado durante as fases de projeto navegacional e de interface.

9.3. SUGESTÕES DE NOVAS PESQUISAS

A partir das propostas apresentadas nos capítulos anteriores pode-se vislumbrar uma variedade de possibilidades no que se refere a extensões do modelo e novas pesquisas nessa área.

Como discutido no Capítulo 6, um hiperdocumento especificado em HMBS pode ficar disponível para navegação por intermédio dos suportes traduzido ou interpretado. O sistema HySCharts implementa o suporte interpretado, tendo sido adotado para validar o HMBS e ilustrar um ambiente automatizado de apoio ao processo de especificação de hiperdocumentos segundo o modelo.

Por outro lado, pode-se também desenvolver trabalhos que adotem a abordagem de suporte traduzido, que objetiva implementar especificações HMBS em plataformas específicas para a composição de hiperdocumentos. Esse é o caso do trabalho que vem sendo realizado por Shibuya [1998], que descreve regras de conversão de especificações HMBS para a linguagem HTML, permitindo "executar" o hiperdocumento na WWW. Outras possibilidades de pesquisa seriam investigar e propor regras para converter especificações HMBS para a linguagem Java ou para os sistemas Director® e Toolbook. Enfim, esses trabalhos envolvem a geração de código a partir de especificações em HMBS, as quais podem ter sido produzidas com o apoio de um sistema (por exemplo, o HySCharts) ou manualmente. Essas especificações definem os objetos navegacionais e de apresentação do hiperdocumento, bem como os mapeamentos para os objetos estruturais, especificados pelo Statechart subjacente. Durante a geração pode-se decidir por

preservar a semântica de navegação do hiperdocumento segundo o modelo HMBS ou, simplesmente, por permitir somente navegação hierárquica/estrutural.

O crescimento do número de ambientes de trabalho cooperativo, permitindo várias pessoas trabalharem no mesmo hiperdocumento simultaneamente, exige que os novos modelos sejam concebidos considerando a necessidade de preservar mais de um estado de seus objetos, ou seja, as diferentes versões de suas informações. Desse modo, o HMBS poderia ser estendido para apoiar a autoria cooperativa e mecanismos de gerenciamento de versões, ampliando o seu campo de abrangência para incluir aplicações que necessitam manter, de forma consistente, múltiplas versões de dados. Um estudo detalhado poderia ser iniciado pelo modelo MCA, uma vez que, segundo Soares et al. [1995b], os mecanismos para tratamento de versões utilizados no MCA podem ser adaptados a qualquer modelo que contenha nós de composição que possam ser aninhados.

Um método abrangente para o projeto e o desenvolvimento de hiperdocumentos pode se beneficiar do HMBS. Carvalho [1997] está definindo um método que sistematiza a abordagem de projeto esboçada no Capítulo 7. O método deve estabelecer formas de modelagem do domínio da informação, provendo os elementos de especificação consumíveis pelo HMBS, como páginas, âncoras e ligações, por exemplo. Em uma fase posterior à especificação dos hiperdocumentos, pode-se utilizar o modelo para fornecer diretrizes de implementação para a plataforma destino, bem como para promover testes da especificação gerada.

Mais especificamente, para a validação e testes de uma especificação realizada em HMBS, abordagens aplicadas tradicionalmente a Statecharts poderiam ser exploradas como, além dos algoritmos de análise discutidos no Capítulo 5, a simulação programada [Cangussu 1993] e a aplicação do critério análise de mutantes [Fabbri 1996]. Para implementar tais algoritmos deve-se, inicialmente, integrar ao módulo de autoria do sistema HySCHarts ferramentas de geração e análise da árvore de alcançabilidade. Acredita-se que esses recursos de validação podem auxiliar os autores na tarefa de identificar problemas na especificação do hiperdocumento, subsidiando a melhoria da qualidade do produto final. Além disso, pode-se estudar como integrar ao sistema HySCharts a avaliação da estrutura de hiperdocumentos sugerida por Fortes [1996]. Tal abordagem fundamenta-se no exame de hiperdocumentos segundo uma análise relacional para implementação do critério de reuso de componentes de ligações [Fortes et al. 1996].

Para permitir a especificação de requisitos temporais e de sincronização relativos à definição e à apresentação de informações multimídia em um ambiente distribuído ou não, é necessário estender o sistema HySCharts para implementar as extensões propostas no modelo XHMBS. Para isso, o sistema deve oferecer apoio para os Hypercharts, a técnica formal subjacente ao XHMBS. Os Hypercharts podem ser considerados uma extensão sintática dos Statecharts, permitindo ocultar especificações completas, utilizando parametrização de estados e abstração de transições [Paulo 1997]. Além dos Hypercharts, o sistema teria que tratar os novos conceitos introduzidos no XHMBS, como os canais de comunicação e canais de apresentação para as âncoras. Com isso, o HySCharts se tornaria um sistema hiperdocumento capaz de apoiar a criação e interpretação de hiperdocumentos contendo mídias dinâmicas. Essa extensão do HySCharts seria importante para validar e testar especificações XHMBS.

Finalizando, é importante destacar que mais experimentos envolvendo o uso do modelo em diferentes domínios de aplicação são exigidos. A tecnologia hipermídia fornece uma nova abordagem para o desenvolvimento de sistemas de gerenciamento de informação, tais como enciclopédias, dicionários, manuais, *handbooks*, documentos on-line, sistemas de aprendizagem, quiosques interativos, museus (*Museum Exhibits*), trabalho cooperativo, sistemas de suporte à decisão (*Decision Support Systems*), engenharia de software e simulação de interface com usuário. A adequação do modelo para especificar tais aplicações poderia ser investigada, possivelmente, com a identificação de extensões necessárias. Dessa forma, espera-se que o HMBS possa evoluir com a maturação do campo, pois a discussão e os conceitos apresentados nesta tese são questões fundamentais para o avanço da pesquisa na busca por modelos e métodos de projeto e desenvolvimento de hiperdocumentos.

REFERÊNCIAS BIBLIOGRÁFICAS

AKSCYN, R.M.; McCRACKEN, D.L.; YODER, E.A. KMS: a distributed hypermedia system for managing knowledge in organizations. *Communications of the Association of Computer Machinery*, v.31(7), p.820-35, July 1988.

ALI, M. A better way to state it. *IEEE Computer*, v.28(12), p.72-4, December 1995.

AMANN, B.; SCHOLL, M. Gram: a graph data model and query language. In: *Proceedings of the European Conference on Hypertext Technology, ACM ECHT Conference*, Milano, November 30 - December 4 1992, p.201-11.

ANDREWS, K.; KAPPE, F.; MAURER, H. Serving information to the web with Hyper-G. *Computer Networks and ISDN Systems*, 27, p.919-26, 1995.

BALASUBRAMANIAN, P.; ISAKOWITZ, T.; STOHR, E. Designing hypermedia applications. In*: Proceedings of the 27th International Conference on System Sciences*, Maui, Hawaii, New York, IEEE Press, January 4-7 1994, p.354-65.

BALASUBRAMANIAN, V.; MA, B.M.; YOO, J. A systematic approach to designing a WWW application. *Communications of the Association of Computer Machinery*, v.38(8), p.47-8, August 1995.

BALASUBRAMANIAN, P.; BIEBER, M.; ISAKOWITZ, T. Systematic hypermedia design. *Technical Report, CRIS Working Paper Series*, New York, 1996. (http://eldorado.stern.nyu.edu/~tisakowi/rmm)

BALASUBRAMANIAN, V.; BASHIAN, A.; PORCHER, D. A large-scale hypermedia application using document management and web technologies. In: *Proceedings of Hypertext'97, VIII International ACM Hypertext Conference,* Southampton, UK, April 6-11 1997, p.134-45.

BATISTA, J.E.S. *Um editor gráfico para Statecharts*. Dissertação de Mestrado, ICMC-USP, São Carlos, 1991.

BATISTA, T.V. *Controle de versões no modelo hipermídia de contextos aninhados.* Dissertação de Mestrado, Departamento de Informática, PUC, Rio de Janeiro, Março 1994.

BEECK, M.V.D. A comparison of statecharts variants. In: *Proceedings of the Joint Symposium on Formal Technique in Real-Time and Fault-Tolerant Systems and Working Group Provably Correct Systems ProCoS, Lecture Notes in Computer Science,* 863, Springer-Verlag, 1994, p.128-48.

BEGORAY, J.A. An introduction to hypermedia issues, systems and application areas. *International Journal of Man-Machine Studies*, v.33, p.121-47, 1990.

BERNERS-LEE, T.; CAILLIAU, R.; LUOTONEN, A, NIELSEN, H.F.; SECRET, A. The world wide web. *Communications of the Association of Computer Machinery*, v.37(8), p.77-82, August 1994.

BIEBER, M.; KACMAR, C. Designing hypertext support for computational applications. *Communications of the Association of Computer Machinery*, v.38(8), p.99-107, August 1995.

BIEBER, M.; VITALI, F. Toward support for hypermedia on the world wide web. *IEEE Computer*, p.62-70, January 1997.

BIGELOW, J. Hypertext and case. *IEEE Software*, p.23-7, March 1988.

BOAVENTURA, I.A.G. *Propriedades dinâmicas de Statecharts.* Dissertação de Mestrado, ICMC-USP, São Carlos, 1992.

BOTAFOGO, R.; MOSSÉ, D. The MORENA model for hypermedia authoring and browsing. *In: Proceedings of the International Conference on Multimedia Computing and Systems*, IEEE Press, Los Alamitos, CA, USA, May 15-18 1995, p.42-9.

BROWN, P.J. Interactive documentation. *Software, Practice and Experience,* v.16(3), p.291-99, March 1986.

BROWN, P.J. Turning ideas into products: the Guide system. In: *Proceedings of Hypertext'87, International ACM Hypertext Conference,* Chapel Hill, New York, November 1987, p.33-40.

BROWN, P.J. Assessing the quality of hypertext documents. In: *Proceedings of Hypertext'90, I International ACM Hypertext Conference,* France, November 1990, p.1-12.

BROWN, K. *O ABD do Toolbook for windows.* São Paulo, Makron Books, 1991.

BUCHMANN, G; SUNSHINE, C.A. Formal methods in communication protocol design. *IEEE Transactions Communications*, v.28(4), p.624-31, 1980.

BULTERMAN, D.C.A. Specifying and support of adptable networked multimedia. *ACM Multimedia Systems*, v.1(2), p.68-76, 1993.

CALOINI, A. Matching hypertext models to hypertext systems: a compilative approach. In: *Proceedings of the European Conference on Hypertext Technology, ACM ECHT Conference*, Milano, November 30 - December 4 1992, p.91-101.

CAMPBELL, B.; GOODMAN, J.M. HAM: a general purpose hypertext abstract machine. *Communications of the Association of Computer Machinery,* v.31(7), p.856-61, July 1988.

CANGUSSU, J.W.L. *Execução programada de Statecharts.* Dissertação de Mestrado, ICMC-USP, São Carlos, 1993.

CARVALHO, M.R. *Um método orientado a objetos para o projeto e o desenvolvimento de aplicações hipermídia.* Qualificação de Mestrado, ICMC-USP, São Carlos, Abril 1997.

CARVALHO, M.R. HMBS/M: u*m método orientado a objetos para o projeto e o desenvolvimento de aplicações hipermídia.* Dissertação de Mestrado, ICMC-USP, São Carlos, Abril 1998.

CASANOVA, M.A.; LIMA, M.J.D.; RANGEL, J.L.; RODRIGUEZ, N.; SOARES, L.F.G.; TUCHERMAN, L. The nested context model for hyperdocuments. In: *Proceedings of Hypertext'91, International ACM Hypertext Conference*, San Antonio, Texas, December 15-18 1991, p.193-201.

CASANOVA, M.A.; SOUZA FILHO G.L.; SOARES, L.F. Anchors and links for composite nodes. *Technical Report MCC 32/95*, PUC, Rio de Janeiro, October 1995.

CAVALLARO, U.; GARZOTTO, F.; PAOLINI, P.; TOTARO, D. HIFI: hypertext interface for information systems. *IEEE Software*, v.10(6), p.48-51, November 1993.

CHEN, P. The entity-relationship approach - toward a unified view of data. *ACM Transactions on Database Systems*, v.1(1), p.9-36, 1976.

COLCHER, S.; SOARES, L.F.G.; CASANOVA, M.A. Arquitetura do sistema HyperProp. In: *Anais do I Workshop em Sistemas Hipermídia Distribuídos, WOSH'95,* São Carlos, SP, Julho 1995, p.13-25.

COLEMAN, D.; HAYES, F.; BEAR, S. Introducing Objectcharts, or how to use statecharts in object-oriented design. *IEEE Transactions on Software Engineering*, v.18(1), p.9-18, January 1992.

COLEMAN, D.; ARNOLD, P.; BODOFF, S.; DOLLIN, C; GILCHRIST, H.; HAYES, F.; JEREMAES, P. *Object-oriented development: the Fusion method.* New Jersey, USA, Prentice Hall, Englewood Cliffs, 1994.

CONKLIN, J. Hypertext: an introduction and survey. *IEEE Computer,* v.20(9), p.17-41, September 1987.

CONKLIN, J.; BEGEMAN, M.L. gIBIS: a hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems,* v.6(4), p.303-31, October 1988.

COOPER, K. *$TSPN_{UI}$: A Petri Net model for specifying user interactions in multimedia presentations.* MSc Dissertation, The University of British Columbia, Canadá, 1995.

COSTA, F.R.; SOARES, L.F.G.; SOUZA, G.L. Editor e browser gráfico para sincronização temporal e espacial de objetos multimídia/hipermídia. In: *Anais do II Workshop em Sistemas Hipermídia, WOSH'96,* Fortaleza, CE, 18 e 19 de Maio 1996, p.21-30.

COURTIAT, J.P.; OLIVEIRA, R.C.D. Proving temporal consistency in a new multimedia synchronization model. In: *Proceedings ACM Multimedia*, Boston, MA, USA, 1996.

COWAN, D.D.; LUCENA, C.J.P. Abstract data views: an interface specification concept to enhance design for reuse. *IEEE Transactions on Software Engineering*, v.21(3), March 1995.

DELISLE, N.; SCHWARTZ, M. Neptune: A hypertext system for CAD applications. In: *Proceedings of the ACM International Conference on Management of Data, SIGMOD'86*, Washington, D.C., NewYork, May 28-30 1986, p.132-42.

DELISLE, N.M.; SCHWARTZ, M.D. Contexts: a partitioning concept for hypertext. *ACM Transactions on Information Systems*, v.5(2), p.168-86, April 1987.

DÍAZ, M.; SÉNAC, P. Time stream Petri Nets: a model for multimedia streams synchronization. In: *Proceedings of the International Conference on Multimedia Modeling*, Singapura, November 1993.

DÍAZ, 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.

DÍAZ, A.; ISAKOWITZ. T.; MAIORANA, V.; GILABERT, G. RMCase: a tool to design WWW applications. *World Wide Web Journal*, v.1(1), p.559-66, 1995.

DITTRICH, K.; LORIE, R. Version support for engineering database systems. *IEEE Transactions on Software Engineering*, v.14(4), April 1988.

ELMASRI, R.; NAVATE, S. *Fundamentals of database systems.* The Benjamin/Cummings Publishing Company, Second Edition, 1990.

FABBRI, S.C.P.F. *A análise de mutantes no contexto de sistemas reativos: uma contribuição para o estabelecimento de estratégias de teste e validação.* Tese de Doutorado, IFSC-USP, São Carlos, 1996.

FORTES, R.P.M. *Uma ferramenta de apoio à utilização de statecharts para especificação do comportamento de sistemas de tempo real complexos.* Dissertação de Mestrado, ICMC-USP, São Carlos, Março 1991.

FORTES, R.P.M.; MASIERO, P.C. Uma linguagem para especificação de Statecharts e seu analisador. *Relatório Técnico*, ME-91/01, ICMC-USP, São Carlos, 1991.

FORTES, R.P.M. *Análise e avaliação de Hiperdocumentos: uma abordagem baseada na representação estrutural.* Tese de Doutorado, IFSC-USP, São Carlos, Brasil, 1996.

FORTES, R.P.M., GARCIA NETO, A.; NICOLETTI, M.C. A formal approach to consistency and reuse of links in hypermedia applications. In: *Proceedings of the Workshop on Formal Methods in Human Computer Interaction: Comparison, Benefits, Open Questions*, *ACM CHI'96,* Vancouver, Canada, April 14-15 1996, p.33-36.

FRAÏSSÉ, S. A task driven method and its associated tool for automatically generating hypertexts. In: *Proceedings of Hypertext'97, VIII International ACM Hypertext Conference,* Southampton, UK, April 6-11 1997, p.234-35.

FRASER, M.D.; KUMAR, K.; VAISHNAVI, V.K. Informal and formal requirements specification languages: bridging the gap. *IEEE Transactions on Softwrare Engineering*, v.17(5), p.454-66, May 1991.

FRASER, M.D.; KUMAR, K.; VAISHNAVI, V.K. Strategies for incorporating formal specifications in software development. *Communications of the Association of Computer Machinery*, v.37(10), p.74-86, October 1994.

GARG, P.K. Abstraction mechanisms in hypertext. *Communications of the Association of Computer Machinery*, v.31(7), p.862-70, July 1988.

GARZOTTO, F.; PAOLINI, P.; SCHWABE, D. HDM - a model for the design of hypertext applications. In: *Proceedings of Hypertex'91, III ACM Hypertext Conference,* San Antonio, TX, December 15-18 1991, p.313-28.

GARZOTTO, F.; PAOLINI, P.; SCHWABE, D. HDM - a model-based approach to hypertext application design. *ACM Transactions on Information Systems*, v.11(1), p.1-26, January 1993 (a).

GARZOTTO, F.; MAINETTI, L; PAOLINI, P. Navigation patterns in hypermedia data bases. In: *Proceedings of the 26th Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, January, v.III, 1993 (b), p.370-79.

GARZOTTO, F.; MAINETTI, L; PAOLINI, P. Adding multimedia collections to the Dexter model. In: *Proceedings of the European Conference on Hypermedia Technologies, ACM ECHT'94*, Edinburgh, Scotland, September 1994, p.70-80.

GARZOTTO, F.; MAINETTI, L; PAOLINI, P. Hypermedia design, analysis, and evaluation issues. *Communications of the Association of Computer Machinery*, v.38(8), p.74-86, August 1995.

GARZOTTO, F.; MAINETTI, L; PAOLINI, P. Information reuse in hypermedia applications. In: *Proceedings of Hypertext'96, VII International ACM Hypertext Conference,* Washington DC, USA, New York, March 1996, p.93-104.

GARZOTTO, F.; MAINETTI, L; PAOLINI, P. Designing modal hypermedia applications. In: *Proceedings of Hypertext'97, VIII International ACM Hypertext Conference,* Southampton, UK, April 6-11 1997, p.38-47.

GINIGE, A.; LOWE, D.B.; ROBERTSON, J. Hypermedia authoring. *IEEE Multimedia*, v.2(4), p.24-35, Winter 1995.

GRØNBAEK, K. Composites in a Dexter-based hypermedia framework. In: *Proceedings of the European Conference on Hypermedia Technologies, ACM ECHT'94*, Edinburgh, Scotland, Proceedings, September 1994, p.59-69.

GRØNBAEK, K.; TRIGG, R.H. Hypermedia system design applying the Dexter model. *Communications of the Association of Computer Machinery,* v.37(2), p.26-39, February 1994 (a).

GRØNBAEK, K.; TRIGG, R.H. Design issues for a Dexter-based hypermedia system. *Communications of the Association of Computer Machinery*, v.37(2), p.41-9, February 1994 (b).

GRØNBAEK, K.; HEM, J.A.; MADSEN, O.L.; SLOTH, L. Cooperative systems: a Dexter-based architecture. *Communications of the ACM,* v.37(2), p.65-74, February 1994.

GRØNBAEK, K.; BOUVIN, N.O.; SLOTH, L. Designing Dexter-based hypermedia services for the World Wide Web. In: *Proceedings of Hypertext'97, VIII International ACM Hypertext Conference,* Southampton, UK, April 6-11 1997, p.146-56.

HALASZ, F.G. Reflections on Notecards: seven issues for the next generation of hypermedia systems. *Communications of the Association of Computer Machinery,* v.31(7), p.836-52, July 1988.

HALASZ, F.G.; SCHWARTZ, M. The Dexter hypertext reference model. *Communications of the Association of Computer Machinery*. v.37(2), p.30-9, February 1994.

HARDMAN, L.; BULTERMAN, D.C.A.; VAN ROSSUM, G. Links in hypermedia: the requirements for context. In: *Proceedings oh Hypertext'93, International ACM Hypertext Conference*, Seattle, Washington, USA, November 1993, p.183-91.

HARDMAN, L.; BULTERMAN, D.C.A.; VAN ROSSUM, G. The Amsterdam hypermedia model: adding time and context to the Dexter model. *Communications of the Association of Computer Machinery*, v.37(2), p.50-62, February 1994.

HAREL, D. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, v.8(3), p.231-74, July 1987.

HAREL, D.; PNUELI, A.; SCHMIDT, J.P.; SHERMAN, R. On the formal semantics of statecharts. In: *Proceedings of the 2nd IEEE Symposium on Logic in Computer* Science, IEEE Press, Ithaca, New York, 1987, p.54-64.

HAREL, D. On visual formalisms. *Communications of the Association of Computer Machinery,* v.31(5), p.514-30, May 1988.

HAREL, D.; LACHOVER, H.; NAAMAD, A. PNUELI, A.; POLITI, M.; SHERMAN, R. SHTULL-TRAURING, A. TRAKHTENBROT, M. STATEMATE: a working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering,* v.16(4), p.403-14, April 1990. (http://www.ilogix.com/)

HAREL, D. Biting the silver bullet. *IEEE Computer*, p.8-21, January 1992.

HAREL, D.; KAHANA, C.A. On Statecharts with overlapping. *ACM Transactions on Software Engineering and Methodology*, v.1(4), p.399-421, October 1992.

HAREL, D.; NAAMAD, A. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, v.5(4), p.293-333, October 1996.

HAREL, D.; GERY, E. Executable object modeling with Statecharts. *IEEE Computer*, v.30(7), p.31-42, July 1997.

HOOMAN, J.J.M.; RAMESH, S.; ROEVER, W.P. A compositional axiomatization of Statecharts. *Theoretical Computer Science*, 101, p.289-335, 1992.

HUIZING, C.; GERTH, R.; ROEVER, W.P. Modelling statecharts behaviour in a fully abstract way. *Lecture Notes in Computer Science*, 299, Springer-Verlag, Berlin, p.271-94, 1988.

HUIZING, C.; ROEVER, W.P. Introduction to design choices in the semantics of statecharts. *Information Processing Letters*, 37, p.205-13, 1991.

HULL, R.; KING, R. Semantic database modeling: survey, applications, and research issues. *ACM Computing Surveys*, v.19(3), p.201-60, September 1987.

ISAKOWITZ, T.; STOHR, E.A.; BALASUBRAMANIAN, P. RMM: a methodology for structured hypermedia design. *Communications of the Association of Computer Machinery*, v.38(8), p.34-44, August 1995.

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 (a). (http://eldorado.stern.nyu.edu/~tisakowi/rmm/)

ISAKOWITZ, T; DÍAZ, A.; MAIORANA, V.; GILABERT, G. Computer support for designing structured hypermedia applications. *Tecnhical Report*, 1997 (b). (http://eldorado.stern.nyu.edu/~tisakowi/rmm/)

JAHANIAN, F.; MOK, A.K. Modechart: a specification language for real-time systems. *IEEE Transactions on Software Engineering*, v.20(12), p.933-47, December 1994.

KENSTEN, Y.; PNUELLI, A. Timed and hybrid statecharts and their textual representation. *Lecture Notes in Computer Science*, 571, Springer-Verlag, p.591-620, January 1992.

LADD, B.C.; CAPPS, M.V.; STOTTS, P.D. The World Wide Web: what cost simplicity? In: *Proceedings of Hypertext'97, VIII International ACM Hypertext Conference,* Southampton, UK, April 6-11 1997, p.210-11.

LANGE, D.B. A formal model of hypertext. In: *Proceedings of Hypertext'90, I International ACM Hypertext Conference,* France, November 1990, p.145-66.

LANGE, D.B. An object-oriented design method for hypermedia information systems. In: *Proceedings of the 27th International Conference on System Sciences*, Maui, Hawaii, New York, IEEE Press 1994, p.366-75.

LEGGETT, J.J.; SCHNASE, J.L. Viewing Dexter with open eyes. *Communications of the Association of Computer Machinery*, v.37(2), p.76-86, February 1994.

LEVESON, N.G.; STOLZY, J.L. Safety analysis using Petri Nets. *IEEE Transactions on Software Engineering*, v.13(3), p.386-97, 1987.

LITTLE, T.D.C.; GHAFOOR, A. Synchronization and storage for multimedia objects. *IEEE Journal on Selected Areas in Communications*, v.8(3), p.413-27, April 1990.

LUCENA, F.N. *Xchart: um modelo de especificação e implementação de gerenciadores de diálogo.* Tese de Doutorado, Departamento de Informática, Unicamp, SP, Dezembro 1997.

MARANINCHI, F. Argonaute: graphical descritpion, semantics and verification of reactive systems by using a process algebra. *Lecture Notes in Computer Science*, 407, p.38-53, 1989.

MARCHIONINI, G.; SHNEIRDERMAN, B. Finding facts vs. browsing knowledge in hypertext systems. *IEEE Computer,* v.21(1), p.70-80, January 1988.

MARMANN, M.; SCHLAGETER, G. Towards a better support for hypermedia structuring: the HyDesign Model. In: *Proceedings of the European Conference on Hypertext Technology, ACM ECHT Conference*, Milano, November 30 - December 4 1992. p.232-41.

MARSHALL, C.C.; IRISH, P.M. Guided-tours and on-line presentations: how authors make existing hypertext intelligible for readers. In: *Proceedings of Hypertext'89, International ACM Hypertext Conference*, November 1989, p.15-26.

MARSHALL, V.A. WWW: The world wide web. *Lecture Notes in Computer Science*, 1012, p.350-62, 1995.

MASIERO, P.C., FORTES, R.P.M.; BATISTA NETO, J.E.S. Editing and simulating the behavioral aspects of real-time systems. In: *Anais XVIII Seminário Integrado de Hardware e Software,* SEMISH, Santos, Agosto 5-9 1991, p.45-61.

MASIERO, P.C.; MALDONADO, J.C.; BOAVENTURA, I.G. A reachability tree for statecharts and analysis of some properties. *Information and Software Technology,* v.36(10), p.615-24, 1994.

MUCHALUAT, D.C.; SOARES, L.F.G. Browsers e trilhas no sistema HyperProp. In: *Anais do I Workshop em Sistemas Hipermídia Distribuídos, WOSH'95,* São Carlos, SP, Julho 1995, p.58-66.

NANARD, J.; NANARD, M. Hypertext design environments and the hypertext design process. *Communications of the Association of Computer Machinery*, v.38(8), p.49-56, August 1995.

NEMETZ, F.; LIMA, J.V.; BORGES, A.C. Uma técnica para modelagem de aplicações hipermídia e um exemplo de uso. In: *Anais XXII Seminário Integrado de Hardware e Software, SEMISH, XXI Conferência Latina-Americana de Informática, CLEI*, Canela, Agosto 1995, p.261-74.

NICHOLAS, C.K.; ROSENBERG, L.H. Canto: a hypertext data model. *Electronic Publishing,* v.6(2), p.93-113, June 1993.

NIELSEN, J. *Hypertext and hypermedia.* Academic Press, CA-USA, Boston, 1990(a).

NIELSEN, J. The art of navigating through hypertext. *Communications of the Association of Computer Machinery,* v.33(3), p.296-310, March 1990 (b).

OLIVEIRA, M.C.F.; TURINE, M.A.S.; MASIERO, P.C.M. A Statechart: based model for hypertext. *Notas do ICMC-USP,* 19, São Carlos, 1995.

OLIVEIRA, M.C.F.; TURINE, M.A.S.; MASIERO, P.C. An overview of HMBS: a statechart-based model for hypertext. In: *Anais do II Workshop em Sistemas Hipermídia, WOSH'96,* Fortaleza, CE, 18 e 19 de Maio 1996, p.11-20.

OLIVEIRA, M.C.F.; TURINE, M.A.S.; MASIERO, P.C. A Statechart-based model for hypertext. Submetido à *ACM Transactions on Information Systems*, 1998.

PARK, B.K.; LOH, W.K.; LEE, J.J.; PARK, C.M.; WHANG, K.K. An object-oriented hypermedia system based on the Dexter reference model and the MHEG standard. *IEICE Transactions Information & System*, v.E79-D (6), p.687-93, June 1996.

PARUNAK, H.V.D. Hypermedia topologies and user navigation. In: *Proceedings of Hypertext'89, International ACM Hypertext Conference*, November 1989, p.43-50.

PARUNAK, H.V.D. Don't link me in set based hypermedia for taxonomic reasoning. In: *Proceedings of Hypertext'91, International ACM Hypertext Conference*, San Antonio, Texas, December 15-18 1991, p.233-41.

PAULO, F.B.; MASIERO, P. C.; OLIVEIRA, M. C. F. Extensões ao HMBS para especificação de apresentações hipermídia. In: *Anais do X Simpósio Brasileiro de Engenharia de Software, SBES'96*, UFSCar, São Carlos, São Paulo, 1996, p.241-56.

PAULO, F.B. *Especificação de aplicaçõses hipermídia baseada em Statecharts.* Dissertação de Mestrado, ICMC-USP, São Carlos, 1997.

PAULO, F.B.; MASIERO, P. C.; OLIVEIRA, M. C. F. Hypercharts: extended statecharts to support hypermedia specification. In: *Proceedings ICECCS, Third IEEE International Conference on Engineering of Complex Computer Systems*, Como, Italy, September 8-12 1997, p.152-61.

PAULO, F.B.; TURINE, M.A.S.; OLIVEIRA, M.C.F.; MASIERO, P. C. XHMBS: a formal model to support hypermedia specification. A ser apresentado na *IX International ACM Conference on Hypertext and Hypermedia, Hypertext'98,* Marriott City Center, Pittsburgh, PA, USA, 20-24 June 1998.

PENTEADO, R.A.D.; MASIERO, P.C.; GERMANO, F.S.R. Manual de uso do sistema STATSIM. *Relatório Técnico,* ICMC-USP, São Carlos, Março 1995.

PENTEADO, R.A.D. *Um método para engenharia reversa orientada a objetos.* Tese de Doutorado, IFSC-USP, São Carlos, 1996.

PETERSON, J.L. Petri nets. *Computing Surveys,* v.9(3), p.223-52, September 1977.

PRABHAKARAN, B.; RAGHAVAN, S. V. Synchronization models for multimedia presentation with user interaction. In: *Proceedigns of the ACM Multimedia*, Califórnia, EUA, June 1993, p.157-66.

RAYMOND, D.R.; TOMPA, F.W. Hypertext and the Oxford English dictionary. *Communications of the Association of Computer Machinery,* v.31(7), p.871-9, July 1988.

REIN, G.L.; ELLIS, C.A. rIBIS: a real-time group hypertext system. *International Journal of Man-Machine Studies,* v.34, p.349-67, 1991.

ROSSI, G.; SCHWABE, D.; LUCENA, C.J.P.; COWAN, D.D. An object-oriented model for designing the human-computer interface of hypermedia applications. In: *Proceedings of the International Workshop on Hypermedia Design, IWHD'95*, Springer Verlag, November 1995.

ROSSI, G. *Um método orientado a objetos para o projeto de aplicação hipermídia.* Tese de Doutorado, Departamento de Informática, PUC, Rio de Janeiro, 1996.

ROSSI, G.; SCHWABE, D.; GARRIDO, A. Design reuse in hypermedia applications development. In: *Proceedings of Hypertext'97, VIII International ACM Hypertext Conference,* Southampton, UK, April 6-11 1997, p.57-66.

RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. *Object-oriented modeling and design.* New Jersey, Englewood Cliffs, 1991.

SCHÜTT, H.A.; STREITZ, N.A. Hyperbase: a hypermedia engine based on a relational database management system. In: *Proceedings of Hypertext'90, I International ACM Hypertext Conference,* France, November 1990, p.95-108.

SCHWABE, D.; CALOINI, A. GARZOTTO, F.; PAOLINI, P. Hypertext development using a model-based approach. *Software, Practice and Experience,* v.22(11), p.937-62, November 1992.

SCHWABE, D.; ROSSI, G. From domain models to hypermedia applications: an object-oriented approach. In: *Proceedings of European Conference Hypermedia Technology, International Workshop on Methodologies for Designing and developing Hypermedia Applications*, Edinburgh, September 1994.

SCHWABE, D.; BARBOSA, S.D.J. Navigation modeling of hypermedia applications. *Technical Report MCC 42/94*, Departamento de Informática, PUC, Rio de Janeiro, 1994.

SCHWABE, D.; ROSSI, G. Building hypermedia applications as navigational views of information models. In: *Proceedings of the 28th Hawaii International Conference on System Sciences*, Hawaii, January 1995, p.231-40.

SCHWABE, D.; ROSSI, G.; BARBOSA, S.D.J. Systematic hypermedia application design with OOHDM. In: *Proceedings of Hypertext'96, VII International ACM Hypertext Conference,* Washington DC, USA, New York, March 1996, p.116-28.

SÉNAC, P., DIAZ, M.; SAQUI-SANNES, P. Toward a formal specification of multimedia synchronization scenarios. *Annals of Telecommunications*, v.49(5-6), p.297-314, May 1994.

SHIBUYA, R. *Geração automática de especificações HMBS em HTML.* Dissertação de Mestrado, ICMC-USP, São Carlos, 1998 (em preparação).

SIGNORE, O. Issues on hypertext design. In: *Database and Expert Systems Applications, 6th International Conference, DEXA' 95*, Springer-Verlag, Berlin, Germany, September 4-8 1995, p.283-92.

SMITH, J.B.; WEISS, S.F. An overview of hypertext (introduction to special issue on hypertext). *Communications of the Association of Computer Machinery,* v.31(7), p.816-9, July 1988.

SOARES, L.F.G.; CASANOVA, M.A.; RODRIGUEZ, N.L.R. Um modelo conceitual hipermídia com nós de composição e controle de versões. In*: Anais do VII Simpósio Brasileiro de Engenharia de Software, SBES'93*, Rio de Janeiro, Brasil, Outubro 1993, p.365-80.

SOARES, L.F.G.; RODRIGUES, N.; CASANOVA, M.A. Modelo de contextos aninhados - um modelo conceitual de hipermídia. *Revista Brasileira de Computação,* v.7(2), p.35-48, Janeiro/Junho 1994.

SOARES, L.F.G. et al. HyperProp: uma visão geral. In: *Anais do I Workshop em Sistemas Hipermídia Distribuídos, WOSH'95,* São Carlos, SP, Julho 1995 (a), p.1-12.

SOARES, L.F.G.; CASANOVA, M.A.; RODRIGUEZ, N.L.R. Nested composite nodes and version control in an open hypermedia system. *International Journal on Information Systems, Special Issue on Multimedia Information Systems.* September 1995 (b).

SOARES, L.F.G.; RODRIGUES, R.F. Autoria e formatação estruturada de documentos hipermídia com restrições temporais. In: *Anais do III Workshop em Sistemas Multimídia e Hipermídia, WOMH'97*, São Carlos, SP, 23 a 25 de Maio 1997, p.183-97.

SOUZA FILHO, G.L.; SOARES, L.F.G.; CASANOVA, M.A. Synchronization aspects of an hypermedia presentation model with composite nodes. In: *Proceedings of*

ACM Workshop on Effective Abstractions in Multimedia , in connection with the ACM *Multimedia'95*, San Francisco, EUA, November 1995.

SOUZA FILHO, G.L.; SOARES, L.F.G. O modelo de apresentação do Hyperprop. In: *Anais do I Workshop em Sistemas Hipermídia Distribuídos, WOSH'95,* São Carlos, SP, Julho 1995, p.47-57.

SOUZA FILHO, G.L.; SOARES, L.F.G. Edição e execução de apresentações de documentos hipermídia no HyperProp. In: *Anais do II Workshop em Sistemas Hipermídia, WOSH'96,* Fortaleza, CE, 18 e 19 de Maio 1996, p.61-70.

STOTTS, P.D.; FURUTA, R. Petri Net based hypertext: document structure with browsing semantics. *ACM Transactions on Information System*, v.7(1), p.3-29, January 1989.

STOTTS, P.D.; FURUTA, R. Hierarchy, composition, scripting languages, and translators for structured hypertext. In: *Proceedings of Hypertext'90, I International ACM Hypertext Conference,* France, November 1990, p.180-93.

STOTTS, P.D.; FURUTA, R. Dynamic adaptation og hypertext structure. In: *Proceedings of Hypertext'91, International ACM Hypertext Conference*, San Antonio, Texas, December 15-18 1991, p.219-231.

STOTTS, P.D.; FURUTA, R.; RUIZ, J.C. Hyperdocuments as automata: trace-based browsing property verification. In: *Proceedings of the European Conference on Hypertext Technology, ACM ECHT Conference*, Milano, November 30 - December 4 1992, p.272-81.

THÜRING, M.; HANNEMANN, J.; HAAKE, J.M. Hypermedia and cognition: designing for comprehension. *Communications of the Association of Computer Machinery,* v.38(8), p.57-66, August 1995.

THUERING, M.; HAAKE, J.M.; HANNEMANN, J. What's Eliza doing in the chinese room? Incoherent hyperdocuments - and how to avoid them. In: *Proceedings of Hypertext'91, International ACM Hypertext Conference*, San Antonio, Texas, December 15-18 1991, p.161-77.

TOMPA, F.WM. A data model for flexible hypertext database system. *ACM Transactions on Information Systems,* v.7(1), p.85-100, January 1989.

TRIGG, R.H.; IRISH, P.M. Hypertext habitats: experiences of writers in NoteCards. In: *Proceedings of Hypertext'87, International ACM Hypertext Conference,* Chapel Hill, New York, November 1987, p.89-108.

TURINE, M.A.S.; MASIERO, P.C.; OLIVEIRA, M.C.F. Modelagem de hiperdocumentos baseada em Statecharts. In: *Anais do I Workshop de Teses em Engenharia de Software, X SBES*, São Carlos, SP, 14 de Outubro 1996, p.13-15.

TURINE, M.A.S.; OLIVEIRA, M.C.F.; MASIERO, P.C. Designing structured hypertext with HMBS. In: *Proceedings of Hypertext'97, VIII International ACM Hypertext Conference,* Southampton, UK, April 6-11 1997 (a), p.241-56.

TURINE, M.A.S.; MASIERO, P.C.; JUNIOR, V.A.C. Modelagem do sistema HySCharts utilizando o método Fusion. *Relatório Interno*, ICMC-USP, São Carlos, Agosto 1997 (b).

TURINE, M.A.S.; OLIVEIRA, M.C.F.; MASIERO, P.C. HySCharts: um ambiente de autoria e navegação baseado no modelo HMBS. In: *Anais do IV Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, SBMIDIA'98*, Rio de Janeiro, Brasil, 29 a 31 de Maio de 1998, p-27-38.

VAN ROSSUM, G.; JANSEN, J.; MULLENDER, K.S.; BULTERMAN, D.C.A. CMIFed: A presentation environment for portable hypermedia documents. In: *Proceedings of First International Conference on Multimedia*, Anaheim, CA, NewYork, August 1993, p.183-8.

VILAIN, P.; HEUSER, C.A. Modelagem de hipertextos: uma experiência com HDM. In: *Anais do VIII Simpósio Brasileiro de Engenharia de Software, SBES'94*, Curitiba, PR, 25 a 28 de Outubro 1994, p.195-209.

VUONG, S.T.; PEREIRA FILHO, J.G. HAS: an object oriented model for hypermedia authoring systems. In: *Proceedings of the 1996 Pacific Workshop on Distributed Multimedia Systems, DMS'96*, Hongkong, June 25-28 1996, p.71-80.

WAN, J; BIEBER, M. GHMI: a general hypertext data model supporting integration of hypertext and information systems. In: *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, HICSS'96, Maui, Hawaii, New York, IEEE Press, January 1996, p.47-56.

WANG, H.K.; WU, J.L.C. Interactive hypermedia applications: a model and its implementation. *Software, Practice and Experience,* v.25(9), p.1045-63, September 1995.

WILLIAMS, G. Hypercard. *Byte*, v.12(14), p.109-17, December 1987.

WING, J.M. A specifier's introduction to formal methods. *IEEE Computer*, v.23(9), p.8-24, September 1990.

WOO, M.; QAZI, N.; GHAFOOR, A. A synchronization framework for communication of pre-orchestrated multimedia information. *IEEE Network,* January/February 1994.

YANKELOVICH, N.; HAAN, B.J.; MEYROWITZ, N.K.; DRUCKER, S.M. Intermedia: the concept and the construction of a seamless information environment. *IEEE Computer,* v.21(1), p.81-96, January 1988.

ZELLWEGER, P.T. Scripted documents: a hypermedia path mechanism. In: *Proceedings of Hypertext'89, International ACM Hypertext Conference*, November 1989, p.1-14.

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 1992, p.242-50.