UNIVERSIDADE FEDERAL DE JUIZ DE FORA - UFJF
INSTITUTO DE CIÊNCIAS EXATAS - ICE
DEPARTAMENTO DE MATEMÁTICA

Um Método para Desenvolvimento de Sistemas Interativos

Aluno: Carlos Henrique Cabral Duarte
Orientador: Michel Heluey Fortuna

Apresentação do Trabalho de Conclusão de Curso da Modalidade Bacharelado em Informática, segundo normas contidas na Resolução nº 01/88, do Colegiado do Curso de Matemática da Universidade Federal de Juiz de Fora.

março/1992

"... quando conjecturava eu sobre a relação existente entre a natureza de estruturas de controle e as funções computáveis recursivas. Olhei para aquele cara, que me encarava fixamente e perguntei:

- Já ouviu falar nisto?

'Só um pouquinho', me respondeu ele.

E continuei a falar, acreditando que minha crença no assunto me desse autoridade suficiente para ignorar o conhecimento de outras pessoas daquela forma."

Ao amigo Ascendino

AGRADECIMENTOS

- A milha família, pela compreensão de que estes momentos de ausência foram por uma boa causa, pela sua dedicação e carinho.

- A meu orientador, Michel Heluey Fortuna, pela dedicação, seriedade e paciência durante estes 15 meses de trabalho, a ainda pelo incentivo ao estudo deste e de outros assuntos.

- Aos amigos de trabalho, Alexandre Faria Guimarães, Anderson de Oliveira e Wagner Antônio Arbex, pelos momentos de incentivo e discussão.

- A Maria Angélica de Andrade Leite, pelo incentivo e apoio no início destes estudos.

- Aos professores Custódio Gouveia Lopes da Motta, Hélio José Correia Barbosa e Raul Fonseca Neto, pelo incentivo para a realização deste estudo e em outros futuros.

- Ao amigo Aloísio Cabalzar Filho, pelo interesse, vontade e paciência de realizar um estudo conjunto.

- Ao amigo José Marcos Lanna de Oliveira, pela hospitalidade e apoio.

- A todos os companheiros de turma, pelo excelente curso que tivemos juntos, pelas boas, e más, experiências.

ÍNDICE

LISTA DE FIGURAS 6

CAPÍTULO 1. Introdução 7
1.1. Identificação do problema 8
1.2. Objetivos 10
1.3. Escopo 11
1.4. Organização do trabalho 12
1.5. Terminologia..... 14
1.5.1. Fundamentos formais..... 15
1.5.2. Outras definições..... 22

CAPÍTULO 2. Revisão da literatura 30
2.1. Diálogo e interfaces com o usuário 31
2.1.1. Estilos de interação 32
2.1.2. Feedback semântico e independência do diálogo 34
2.1.3. Representação do diálogo 36
2.1.4. Toolkits, UIDS e geração automática 39
2.1.5. Arquitetura e controle de componentes 40
2.1.6. Conclusões 42
2.2. Dinâmica 44
2.2.1. Eventos 44
2.2.2. Critérios para uma representação bem definida 45
2.2.3. Representação da dinâmica 47
2.2.4. Conclusões 50

CAPíTULO 3. Os sistemas BIBLOS e DIAGRAMA: visões da modelagem do diálogo 53
3.1. Descrição do sistema BIBLOS 54
3.2. Modelagem do diálogo 59
3.3. O sistema DIAGRAMA 63
3.4. Derivando uma representação menos complexa 67

CAPíTULO 4. O método proposto 77
4.1. Alguns conceitos do método essencial 78
4.2. Um modelo estrutural para o diálogo 82
4.3. Modelagem da dinâmica 88
4.4. A estrutura do método proposto 96
4.5. Estratégias de transformação 100

CAPíTULO 5. Automatizando o método proposto 105
5.1. Avaliação das especificações do diálogo ... 106
5.2. Aplicação de algoritmos de M.E.F.s às especificações do diálogo 110
5.3. Geração automática de código 113

CAPíTULO 6. Conclusões e indicações para trabalhos futuros 128

BIBLIOGRAFIA 135

APÊNDICE. Notação sugerida para elaboração de modelos conceituais 138

LISTA DE FIGURAS

FIGURA	PAGINA
1	17
2	17
3	17
4	26
5	41
6	55
7	55
8	56
9	57
10	62
11	65
12	68
13	68
14	68
15	69
16	69
17	69
18	70
19	71
20	74
21	78
22	83
23	92
24	93
25	97
26	104

Capítulo 1

Introdução

1.1 Identificação do problema

Observa-se uma grande distância entre a especificação de um sistema e a sua implementação através de programas. Tais programas devem ser corretos em relação às especificações e válidos em relação às necessidades do usuário. Daí decorre a necessidade de métodos para desenvolvimento de sistemas capazes de guiar o projetista.

A distância entre especificação e implementação aparece principalmente devido à preocupação existente nos métodos em modelar características extremamente genéricas, como processos [DeMar89], ou atividades e informações [Ward86], não levando em consideração aspectos particulares, presentes em grandes classes de aplicações, fundamentais para levar-nos à uma boa modelagem do sistema.

Tendo em mente estas considerações, alguns métodos enfocam aspectos essenciais peculiares à uma classe de aplicações específica [Quent88], e este é o seu grande mérito: privilegiar os aspectos relevantes para especificação, que estão sempre presentes no domínio das aplicações.

Contudo, não basta apenas privilegiarmos a modelagem de um aspecto, mas estabelecer uma relação entre vários destes, explorando ao máximo as potencialidades de cada um, balanceando visões do domínio e buscando maior riqueza nas informações especificadas.

Neste trabalho nos preocupamos com um destes aspectos em particular, que é o Diálogo entre o usuário e a aplicação. Sem dúvida esta é uma característica comum à quase totalidade dos sistemas desenvolvidos atualmente, e por isto merecendo um estudo mais detalhado.

Como aparece destacado na literatura sobre o assunto, o diálogo é uma característica fundamental para a especificação e implementação de interfaces com o usuário, e em nossa opinião tem um papel ainda com maior destaque, mas na maioria dos casos não aparece especificado em uma fase inicial do desenvolvimento da aplicação, devido, mais uma vez, à omissão dos métodos utilizados.

Acredita-se que a especificação das mensagens do diálogo entre o usuário e a aplicação possa resolver problemas comuns no desenvolvimento de sistemas. Um destes seria a pequena participação do usuário na modelagem. Este problema poderia ser resolvido com a participação dos usuários na escolha das mensagens trocadas, levando a uma

validação previa da implementação, e na escolha da forma de comunicação, evitando dificuldades no uso.

1.2. Objetivos

Ao longo deste trabalho nos preocupamos em responder às seguintes questões:

. Qual a importância do diálogo para a modelagem de um Sistema?

. Quais são os componentes básicos de uma especificação do diálogo?

. Em que nível de abstração deve ser especificado o diálogo?

. Qual a relação existente entre uma especificação do diálogo e de outros aspectos, tais como funções? Em particular, existe alguma diferença entre diálogo e dinâmica no contexto de estudo proposto?

. É possível chegar a alguma forma de código executável a partir destas especificações?

Além destas questões, tentamos desenvolver um estudo metodológico indicando quando usar o método proposto ou não,

e como se dá a implementação de facilidades automáticas de auxílio ao desenvolvimento.

1.3. Escopo

Como descrito anteriormente, desejamos estabelecer um método para desenvolvimento de sistemas que leve em conta a especificação do diálogo, e ainda que possa ser usado em uma classe de aplicações que seja usualmente encontrada em nosso ambiente de trabalho particular.

Esta classe de aplicações deve estar inclusa na dos sistemas interativos, ou seja, que possuem alguma forma de comunicação distinta da inicialização da execução da aplicação ou da explicitação dos resultados finais, visto que esta é uma característica dos sistemas comumente desenvolvidos atualmente. Excluem-se assim as aplicações em "batch".

Outra preocupação cabível é que este método se aplique ao desenvolvimento de grande parte das aplicações que são corriqueiramente desenvolvidas em nosso ambiente de trabalho. Estas são tipicamente sistemas de informação onde não aparecem características como concorrência ou emersão, e

não há restrições de temporização que sejam fundamentais para a execução.

Ao nos preocuparmos com a forma como se dá o diálogo nestas aplicações, constatamos que este é feito de forma seqüencial, e usualmente textual, onde mensagens são trocadas de forma altamente restringida, ou seja, tem definidas previamente sua seqüênciação e conteúdo.

Considerando as restrições postas acima para o escopo das aplicações às quais o método proposto pode ser aplicado, podemos ainda afirmar que o diálogo entre o usuário e a aplicação se dará sempre de forma síncrona, visto que assincronismos estão amarrados à existência de concorrência. Entretanto ao final do trabalho são propostas extensões que contemplem estes casos não abordados.

1.4. Organização do trabalho

Após identificar o problema, estabelecer os objetivos e o escopo deste trabalho, traçaremos um perfil geral do restante dos tópicos abordados, visando dar ao leitor uma visão global do desenvolvimento.

No restante deste capítulo 1 são tratadas definições gerais que serão referenciadas em vários pontos do texto (outras definições locais são dadas no momento em que os termos aparecem citados), separadas em formais, referindo-se aos objetos formais de estudo, e outros termos, relacionando conceitos comumente encontrados na literatura sobre o assunto, permitindo que se tenha uma visão correta sobre estes.

No capítulo 2 são abordados assuntos relacionados ao diálogo e a dinâmica, na visão de outros autores, procurando caracterizar estes dois aspectos dentro do contexto de desenvolvimento de sistemas, explorando as características mais importantes de cada um e traçando uma comparação entre eles.

Em seguida, no capítulo 3, são explorados dois exemplos de especificação, procurando fazer um estudo comparativo entre a modelagem do diálogo nos sistemas BIBLOS e DIAGRAMA, mostrando ainda que uma representação hierarquizada do diálogo criada a partir das funções do sistema não é a mais adequada.

No capítulo 4 é explicitado o método proposto, iniciando-se pela descrição de um modelo estrutural de diálogo, falando também das características relevantes do

método essencial e as relacionadas com a modelagem do diálogo. Propõe-se que, no escopo de estudo, o diálogo seja considerado essencial. Em seguida são traçadas algumas estratégias de transformação entre modelos.

Já no Capítulo 5 são introduzidos conceitos relacionados à automatização do método, incluindo critérios de como avaliar as representações criadas e a automatização do apoio a construção de representações seguindo estes critérios. É comentada a possibilidade da implementação de algoritmos que possam contribuir de alguma forma com a melhoria das condições de desenvolvimento de um sistema seguindo o método proposto, e é mostrado como pode ser feita a geração automática de código, a partir da especificação desenvolvida com o método proposto no capítulo 4.

Finalmente, no Capítulo 6 são apresentadas conclusões e várias indicações para trabalhos futuros detectadas ao longo dos estudos para o desenvolvimento deste texto.

1.5. Terminologia

São colocadas nesta seção as definições dos termos utilizados ao longo do trabalho, agrupados de acordo com seu caráter formal ou informal. Estas definições nos serão úteis

para discutir de maneira precisa os conceitos relacionados ao diálogo comumente citados em Engenharia de Software e áreas afins.

1.5.1. Fundamentos formais

Definição 1: Grafo

"G = (V,U) é um grafo quando V é um conjunto de vértices ou pontos, e U é um conjunto de pares ordenados de elementos pertencentes a V."

A inclusão de um par ordenado (v_i, v_j) em V indica que existe uma relação entre v_i e v_j, a interpretação desta é mesma para todos os elementos de V.

Algumas extensões deste conceito são possíveis. Por exemplo, denotamos por (v_i, v_j) o relacionamento entre v_i e v_j onde não importa a ordem, ou seja, que é simétrico. Por outro lado, denotamos por [v_i, v_j] o relacionamento entre v_i e v_j onde a ordem (ou orientação) é importante, ou seja, [v_i, v_j] $\neq$ [v_j, v_i]. A primeira forma de relacionamento entre os vértices v_i e v_j chamamos aresta e a segunda arco.

Pode-se também imaginar uma estrutura onde V seja um conjunto de produtos cartesianos de elementos de U sem dimensão fixa e a isto chamamos HiperGrafo. Outra forma

encontrada usualmente permite a repetição de elementos em V, ou seja, V deixa de ser um conjunto e passa a ser um "bag" [Peter81], e a isto chamamos p-Grafo, onde p é o número máximo de repetições de cada elemento. Quando não é conhecido a priori o valor de p, chamamos esta construção de multigrafo.

Outra extensão de interesse particular neste trabalho são os grafos valorados G = (U, V, r), com r:V --> C, onde C é um conjunto qualquer. Exemplos de grafos, hipergrafos e grafos valorados são mostrados na figura 1.

Definição 2: Máquina de Estados Finitos [Barre79]

"Uma máquina de estados finitos é um multigrafo orientado valorado M, M=(Q,S,s,qo,F), onde

Q é um conjunto finito de estados;

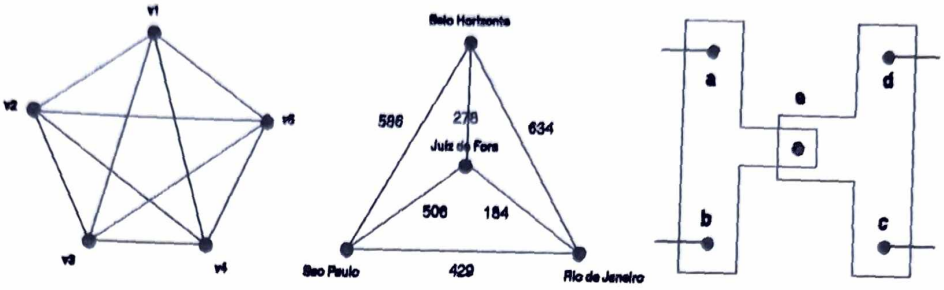
S é um conjunto de símbolos de entrada;

s é uma relação que mapeia um estado e um símbolo de entrada em um outro estado, s:Q x (S U {*E*}) --> Q;

qo é um estado inicial;

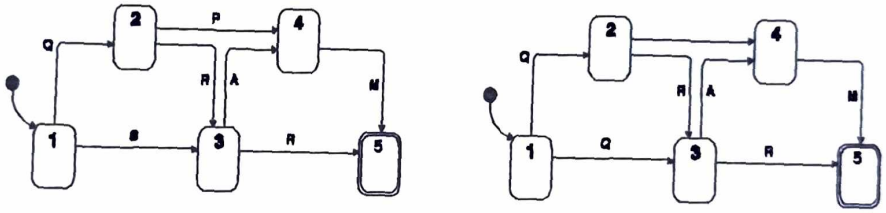
F é um conjunto de estados finais."

Na estrutura acima, cada par (q, s(q)), q pertencente a Q, é chamado transição. Por esta razão, quando s é uma



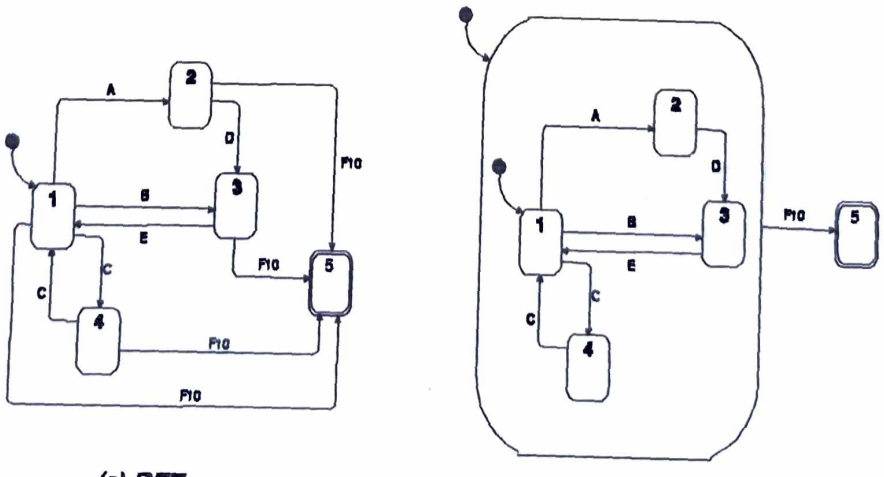
(a) Grafo (b) Grafo Valorado (c) HiperGrafo

FIGURA 1



(a) DET determinístico (b) DET nao-determinístico

FIGURA 2



(a) DET (b) StateChart equivalente

FIGURA 3

função, é chamada função de transição. Observe que uma máquina de estados finitos é sempre um grafo orientado valorado.

O princípio de operação de uma máquina de estados finitos está fundamentado na leitura dos símbolos de S a partir de uma fita, através de uma cabeça de leitura, e a conseqüente ocorrência de uma transição. *E* é chamado símbolo nulo e indica que a máquina realiza uma transição sem que seja lida a fita, e portanto sem deslocamento da cabeça de leitura.

Máquinas de estados finitos (M.E.F.s de forma resumida) são também chamados autômatos finitos, e são criados para o reconhecimento de linguagens de entrada. Tentam criar um modelo a ser seguido por vários dispositivos automáticos de computação, como os computadores atuais.

Dizemos que uma M.E.F. é não-determinística quando alguma transição ocorre devido a *E*, ou quando #s(q,s) <> 1, para algum p pertencente a Q e um s pertencente a S. Caso contrário ela é determinística.

Chamamos de diagrama de estados e transições (resumidamente D.E.T.) à representação gráfica de uma Máquina de Estados Finitos. Na figura 2 são mostrados dois D.E.T.s

representando máquinas determinística e não-determinística. Ao longo do texto convencionamos representar D.E.T.s seguindo a representação posta no Apêndice.

Boa parte da literatura sobre o assunto não considera a distinção entre máquinas de estados finitos, ou extensões destas, de sua representação gráfica, o que pode levar a confusões.

Outra extensão de grafos agrega a estas estruturas os conceitos da Teoria dos Conjuntos, considerando que cada vértice possa conter outros elementos da construção. Neste caso cada vértice é chamado bolha.

Definição 3: HiGrafo [Harel88]

"Um higrafo é uma quádrupla H =(B, z, p, E) onde

B é um conjunto de bolhas;

E é um conjunto de pares ordenados de elementos de B;

z é uma função z:B --> 2^B, que associa à uma bolha o conjunto de sub-bolhas que esta contém (se x pertence a B, então x não pertence a z(x));

p é uma função de particionamento p:B --> $2^{B \times B}$, associando a cada bolha uma relação de equivalência que induz um particionamento no conjunto de sub-bolhas desta bolha."

Obviamente todo grafo é um higrafo, onde a função de inclusão é a função vazia, ou seja, z(x) = {}, para todo x pertencente a B, o mesmo acontecendo para a função de particionamento.

Trazendo M.E.F.s para o contexto de higrafos, podemos definir uma extensão destas:

Definição 4: Máquina de Estados Finitos Estendida

"Máquina de estados finitos estendida é uma setúpla M = (Q, z, p, S, s, Qo, F), onde

Q é um conjunto de estados;

z é uma função que associa um estado a seus sub-estados;

p é uma função de particionamento de estados;

S é um conjunto de símbolos de entrada;

s é uma função de transição;

Qo é um conjunto de estados iniciais;

F é um conjunto de estados finais."

A função p divide z(u), u pertencente a Q, em subconjuntos u_1, u_2, ..., u_n, com a união destes subconjuntos igual a z(u) e a interseção entre eles igual ao conjunto vazio. A cada u_i chamamos componente ortogonal de u.

A semântica de M.E.F.E.s não permite que cada componente ortogonal de um estado possua mais de um estado inicial, e, da mesma forma, não admite que um estado seja sub-estado de si mesmo. Contudo permitem construções bem mais gerais que máquinas normais. Uma destas construções aparece na figura 3.b. Lá aparece uma transição partindo de um estado composto, sendo ativada sempre que F10 for detectado, em qualquer dos sub-estados.

Representações gráficas de M.E.F.E.s são chamadas StateCharts, e neste caso a notação gráfica será a mesma das máquinas de estados finitas, com as ressalvas feitas no Apêndice.

Algumas máquinas de estados finitos podem ser transformadas em M.E.F.E.s, acarretando uma diminuição considerável no número de estados e transições. Na figura 3 é apresentado um exemplo de transformação.

Nos capítulos seguintes serão mostradas formas de representação do diálogo e da dinâmica que utilizam como

ferramentas D.E.T.s e statecharts. Embora estas sejam formas diagramáticas de representação, tem embutidos todos os conceitos formais apresentados aqui [Harel88].

1.5.2. Outras definições

Definição 5: Sistema [Quent88]

"É um conjunto identificável e coerente de elementos que interagem coesivamente, onde cada elemento pode ser um sistema."

Decorre desta definição que deve haver uma fronteira facilmente demarcável que separa os elementos identificados do resto do universo de discurso, possuindo um padrão organizacional estável.

Por exemplo, um grupo de indivíduos desempenhando atividades corriqueiras de contabilidade em seu ambiente de trabalho, na mesma seção, caracteriza um sistema, pois possui um padrão organizacional constante, interação coesiva e uma fronteira física demarcável.

Existem características marcantes num sistema, entre elas a capacidade de tratá-lo como um único elemento, em um

contexto mais abrangente, ou de um elemento ser outro sistema. Isto decorre da própria definição de sistema.

O nosso exemplo da seção de contabilidade seria considerado um único elemento num sistema que o englobasse. Por exemplo, numa análise de desempenho global da empresa, onde os resultados fossem apresentados discriminados por seção, a seção de contabilidade seria um elemento.

Quando desenvolvemos a tarefa de análise, nos preocupamos em especificar aquelas informações de interesse presentes num sistema, de maneira precisa, com vistas a uma possível automatização de tarefas. Um modelo é o instrumento para isto.

Definição 5: Modelo [Quent88]

> "É uma representação abstrata que privilegia um número reduzido de aspectos característicos de um sistema do mundo real, visando apresentar características descritivas e prescritivas da realidade".

Modelos devem possuir propriedades de totalidade, ou seja, permitir que o modelador represente todos os aspectos do sistema, e de conceitualização, ou seja, de permitir que o

modelador represente apenas aqueles que tenha interesse em modelar [Heusse90].

No contexto de Modelagem aparecem dois termos, um usado normalmente para caracterizar o processo de especificação e desenvolvimento de uma aplicação computacional, denominado modelagem conceitual [Heusse90][Kung89], e outro indicando a intenção de representar uma visão geral de objetos, suas relações e processos envolvidos, chamado modelagem estrutural [Harts89a][Almei90].

Destaca-se aí a diferença de objetivos: o primeiro termo denota a preocupação em representar aspectos sintáticos e semânticos da realidade, e o segundo mostra uma preocupação central com o aspecto pragmático.

No quadro abaixo tentamos clarear a diferença entre os dois termos:

Enfoque	Objetos modelados são	Visão da realidade
conceitual	elementos	semântica
estrutural	conjuntos	pragmática

Normalmente os objetos representados num modelo estrutural são conjuntos, enquanto que na visão conceitual,

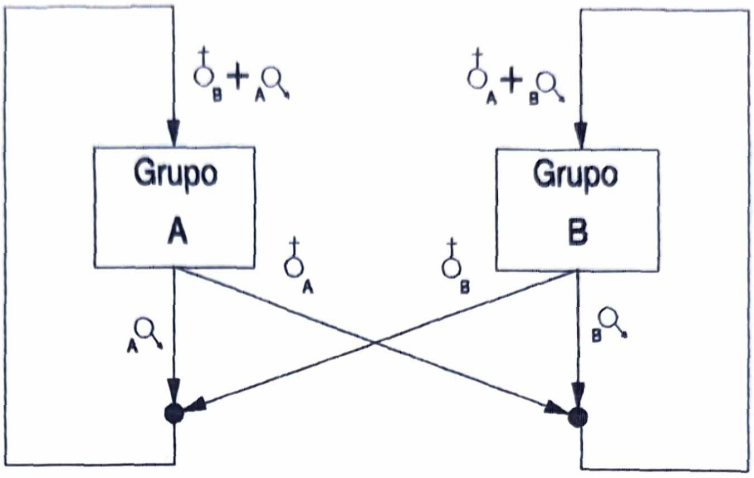
estes seriam indivíduos. Modelos estruturais são modelos de modelos conceituais, ou seja, são meta-modelos.

Na figura 4 é apresentado um modelo estrutural para as trocas de mulheres entre grupos indígenas e um modelo conceitual para a estrutura de parentesco de um grupo específico é modelado. O primeiro é uma representação das características relevantes em todos os modelos que podem ser criados falando sobre estruturas de parentesco de grupos indígenas que trocam mulheres.

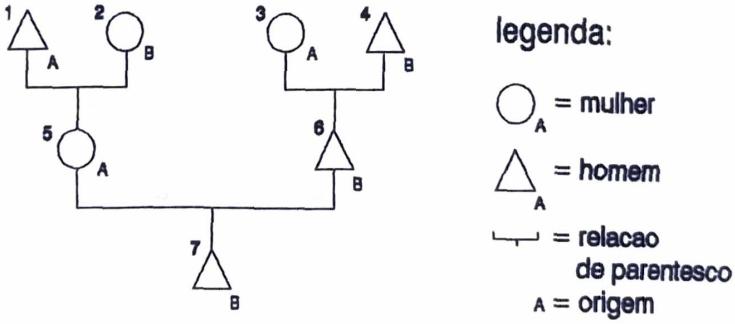
Num dos capítulos seguintes mostraremos que o estabelecimento de um modelo estrutural do diálogo é fundamental para estabelecermos como realizar a modelagem conceitual de aplicações.

Modelos são usados para representar características relevantes de um sistema, usando um esquema notacional apropriado. Quando nos referimos a representações estaremos falando deste esquema notacional.

Na tarefa de especificação com vistas a automatização, são elaborados vários modelos de um mesmo sistema, com o objetivo específico de representar aspectos distintos, enfatizando a idéia de visões múltiplas



(a) Modelo Estrutural



(b) Modelagem Conceitual para uma estrutura de parentesco

FIGURA 4

complementares do sistema modelado. Daí a preocupação de estabelecer diretrizes para o trabalho do analista.

Definição 6: Método [Dasgu89]

"É um procedimento explicitamente prescrito ou um conjunto de regras as quais podem ser seguidas pelo projetista com objetivo de produzir o projeto".

Comumente encontramos confusões entre os termos método e metodologia [Dasgu89]. O segundo trata-se de um estudo realizado no sentido de indicar métodos para utilização. Sem dúvida, este trabalho é um estudo metodológico.

Dando continuidade a esta seção de definições, colocamos agora os termos diálogo e dinâmica, objetos principais de discussão a partir de agora.

Definição 7: Diálogo

"É a troca de idéias, de opiniões, de conceitos ou de mensagens, com vista à solução de problemas, ao entendimento e a harmonia"

Definição 8: Dinâmica

"São as alterações de estado pelas quais passa um objeto e o relacionamento destas alterações às causas que as produziram".

Procuramos nestas definições caracterizar tais termos de forma a não nos atermos simplesmente aos conceitos encontrados em Ciência da Computação, enfatizando o caráter multidisciplinar que os acompanha.

No contexto de desenvolvimento de sistemas, diálogo se refere ao processo de comunicação entre dois ou mais elementos de um sistema, e dinâmica se refere ao comportamento destes dependente do tempo. Está comumente representada na implementação pelos conceitos de controle: repetição, seqüenciamento e seleção.

Daqui para frente, sempre que falarmos em diálogo estaremos nos referindo especificamente ao diálogo entre usuário e uma aplicação computacional, por estarmos interessados em sistemas interativos, e não em sistemas de comunicação, por exemplo.

Diálogo se torna uma das causas para a ocorrência de um comportamento dinâmico de um sistema, principalmente nesta classe de aplicações que nos propusemos a estudar. Seguindo

esta linha de raciocínio, podemos afirmar que, mesmo que não houvesse diálogo em um sistema computacional (excluímos aí a ativação deste), poderia haver dinâmica, mas sem dinâmica não haveria como realizar qualquer espécie de diálogo.

Se encararmos um sistema do ponto de vista das funções que ele desempenha, o diálogo seria incluído entre estas. Entretanto, devido a sua importância para a modelagem do sistema, consideraremos aqui sua representação fatorada, ou seja, separada das outras funções.

Capítulo 2

Revisão da literatura

2.1 Diálogo e interfaces com o usuário

Nesta seção estamos interessados em estudar características do diálogo, tentando determinar quais destas são importantes para a modelagem e implementação de uma aplicação.

Sabe-se, a princípio, que o diálogo está sempre associado à idéia de interface com o usuário. Isto se deve ao fato da interface servir como mediador no processo de comunicação, dando forma às mensagens trocadas.

Normalmente é colocada uma série de orientações que devem ser consideradas ao se criar código para interface e controle do diálogo, entre elas:

1. Interface deve ser fácil de usar e, de preferência, padronizada, evitando que o usuário tenha que aprender uma maneira de se comunicar diferente a cada aplicação que deseje usar;

2. Interface deve ser projetada e implementada de forma a prever independência dos dispositivos disponíveis, evitando reescrita a cada troca de ambiente de execução;

3. O código de controle da interface e do diálogo (geralmente emerso na aplicação [Wu89]) deve ser separado o máximo possível do processamento das outras funções, facilitando a manutenção;

4. O código de controle de interface e do diálogo deve ser reusável, já que tem criação difícil e cara;

5. Diálogo e interface devem ser bem projetados, pois em muitos casos o sucesso da aplicação depende do usuário achar fácil a sua utilização , ou seja, deste não precisar trocar um número muito grande de mensagens para atingir seus objetivos.

Portanto um estudo detalhado das características do diálogo e de interfaces se faz necessário, para que sejam definidas formas de resolver tais problemas.

2.1.1. Estilos de interação

Uma das características do diálogo entre usuário e aplicação é o estilo. Este se refere a qualidades

características da interação, o que o diferencia da aparência, que é apenas uma manifestação externa. Esta última não será objetivo de estudo neste trabalho.

Podemos colocar os estilos de interação classificados em conversacional e baseado na manipulação de objetos, sendo esta a classificação mais ortogonal encontrada [Gaine86][Harts89a].

O estilo conversacional se caracteriza pelo estabelecimento de uma linguagem de interação, que pode ser a própria linguagem natural, ou algum tipo de linguagem formal. A diferenciação entre estas linguagens se faz ao avaliarmos a quantidade de informações semânticas trocadas.

Ressalta-se aqui que a interação em linguagem natural deve suportar apenas um contexto limitado, para que seja garantida a compreensão mútua entre os participantes do diálogo [Gaine86], já que sempre existe ambiguidade neste tipo de linguagem.

O estilo baseado na manipulação de objetos [Jacob86] é, em geral, feito de forma gráfica, e é o mais agradável ao usuário, já que a quantidade de informação contida em uma representação gráfica manipulada é bem maior que um trecho de

linguagem textual. Entretanto requer controle assíncrono [Harts89b].

A integração entre estes estilos vem se fazendo de forma muito rápida, e aparece um termo específico na literatura para denotar isto, que é diálogo de múltiplas formas. Este é encontrado normalmente no ambiente Windows ou no sistema operacional do Machintoch.

2.1.2. Feedback semântico e independência do diálogo

Independência do diálogo é uma analogia aos conceitos de independência de dados, de dispositivos e de linguagem.

Constata-se que a medida que se desenvolve o projeto de uma aplicação computacional, o diálogo se torna mais ligado a aplicação, e assim tornando-se cada vez mais difícil de alterar sem que sejam envolvidos outros aspectos do projeto.

Independência do diálogo não é separação do código que processa o diálogo do código da aplicação. Esta separação aliás não é possível, já que o diálogo é feito normalmente

com o objetivo de realizar uma função da aplicação [Harts89a].

Um fator que influencia fortemente esta amarração entre diálogo e aplicação é a necessidade de fornecer ao usuário informações sobre os objetos da aplicação.

Chama-se feedback semântico à mensagem no sentido computador-usuário, rica em semântica da aplicação, dada em resposta a uma mensagem no sentido usuário-computador. É sempre uma informação sobre os objetos do sistema [Harts89a].

Em diálogo conversacional, a ocorrência de feedback semântico é fundamental para indicar o que foi entendido pela aplicação, a partir da mensagem enviada pelo usuário.

Uma solução que permitiria independência seria considerar a modelagem do diálogo nas fases iniciais no desenvolvimento de sistemas, e também a especificação de feedback semântico nesta fase, como parte integrante do diálogo.

Concluindo, independência do diálogo é a tentativa de separar ao máximo o diálogo da implementação, tanto em tempo de projeto quanto de implementação. Entretanto, por não ser um conceito que pode ser usado de forma efetiva, pois não

garante nenhuma das características acima de separação, não será considerado neste trabalho.

2.1.3. Representação do diálogo

Modelagem do diálogo é uma forma de privilegiar este aspecto, quando feita em tempo de projeto, e tem influência marcante na implementação do código correspondente.

a) Representação baseada em D.E.T.s:

É a mais comumente encontrada e é a forma de representação que foi usada em primeiro lugar. Como as outras formas de representação discutidas aqui, a exceção da baseada em linguagens orientadas a eventos (item C), permite apenas a modelagem de diálogo síncrono .

Pode dar origem a um bom modelo conceitual, dependendo de quais são as informações representadas e a maneira como isto é feito, pois mostram explicitamente o seqüenciamento de interação [Jacob83].

Existem autores que encaram com algum ceticismo o uso desta representação, pois apresenta problemas, como ser muito complexa por apresentar grande número de estados e

transições, mas este é um problema que pode ser contornado, utilizando extensões como StateCharts.

Outro problema é a mistura usual entre a funcionalidade do sistema e o diálogo, muitas vezes expressada em representações hierárquicas [Denert77]. Isto também pode ser resolvido, quando direcionamos o trabalho de modelagem através de um método que considere esta separação.

b) Representação baseada em EBNF:

EBNFs são construções que nos permitem definir uma linguagem em função de termos terminais e construções criadas através de agrupamento destes [Kasik82].

Não dão origem a modelos conceituais, pois o processo recursivo de substituição de símbolos não-terminais para a verificação da corretude de uma mensagem trocada não é um conhecimento natural do usuário. Além disso requerem o conhecimento da linguagem de especificação e não tem a capacidade de exibir sensibilidade ao contexto.

c) Representações baseadas em linguagens orientadas a eventos:

São construções que permitem representar como maior riqueza o processo de interação, inclusive assincronismos.

Porém são difíceis de programar e manter, por não terem mecanismos de construção estruturados. Além disso não representa de forma explicita o seqüenciamento do diálogo. Por estas razões não serve de Modelo Conceitual.

d) Representações baseadas em orientação a objetos:

Em muitos casos servem de Modelo Conceitual para o Diálogo, pois tratam o problema de forma inteligível pelo usuário.

Sua principais desvantagens em relação a outras formas de representação são a não representação explícita do seqüenciamento do diálogo, já que o conhecimento sobre o estado em que se encontra a interação fica oculto dentro da representação de cada objeto [Couta87], e a necessidade, não obrigatória mas altamente recomendável, de se criar uma implementação em uma linguagem orientada a objetos.

2.1.4. Toolkits, UIDSs e geração automática de interfaces

Na tentativa de seguir as orientações descritas no início da seção, algumas soluções propostas tentam facilitar as tarefas de desenvolvimento.

Toolkits são bibliotecas de rotinas pré-definidas que podem ser ligadas a qualquer aplicação em criação. Permitem reuso e padronização, mas o controle da interface e do diálogo continua emerso na aplicação. Aparecem em geral como procedimentos ou objetos e provém um número limitado de estilos e aparências para a interação.

UIDSs são ambientes de desenvolvimento de interfaces com o usuário. Permitem o desenvolvimento de interfaces e do diálogo no próprio ambiente, mas em geral não consideram outras características da aplicação.

Geração automática de interfaces com o usuário é difícil de ser implementada e, em geral, se baseia na semântica da aplicação definida em alguma especificação [Myers89]. Provém número limitado de estilos e aparências, e o código gerado é, em geral, difícil de se manter.

2.1.5. Arquitetura e controle de componentes

Nas formas descritas acima para resolver o problema de se implementar interfaces e diálogo procura-se separar ao máximo o código do processamento do diálogo do código da aplicação, prover Feedback Semântico ao usuário e criar código reusável.

Como parece evidente que nenhuma das propostas na seção anterior consegue resolver completamente o problema, propões-se a criação de uma arquitetura que suporte o uso de um gerenciador de interface comum às aplicações desenvolvidas, reusável, separado da aplicação, que forneça recursos para realizar interação da forma desejada.

Esta arquitetura considera a criação de um programa que dará suporte à execução das aplicações construídas, com código executável separado do da aplicação. São arquiteturas como GKS, MOTIF e X-WINDOWS, que contém a estrutura mostrada na figura 5.

Além disso, sugere-se que os módulos da própria aplicação sejam criados de forma que o processamento do

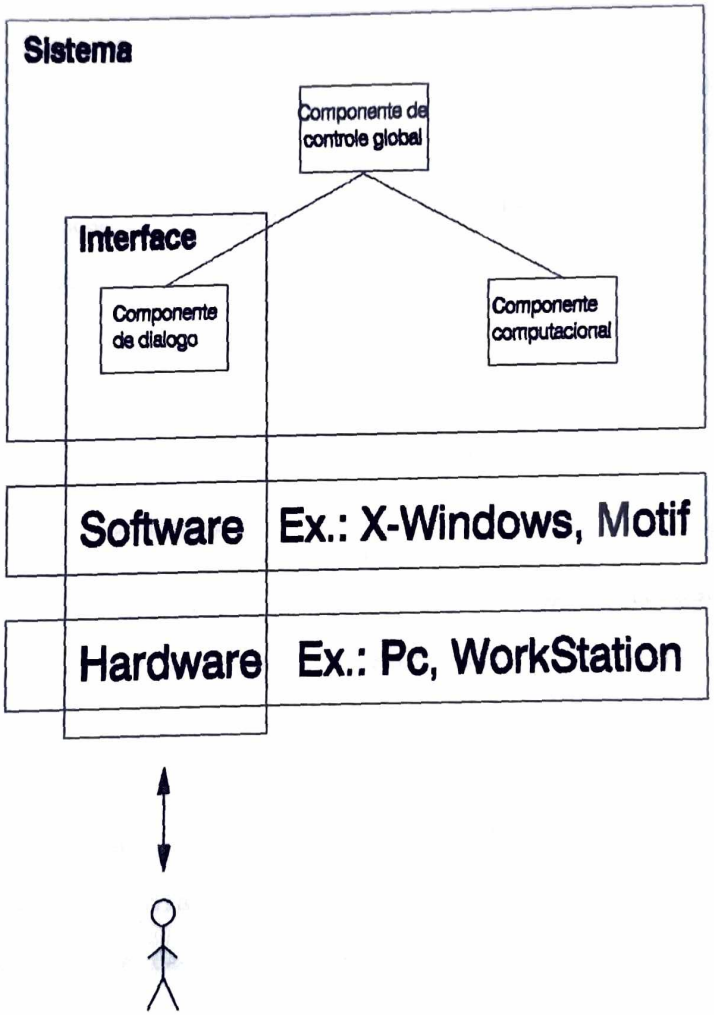


FIGURA 5 - Visao Arquitetural de uma aplicacao [Harts89]

diálogo seja colocado em separado, como aparece na figura 5, dentro da caixa "sistema".

Desta forma, apareceria um outro diálogo, entre a aplicação e o ambiente, em tempo de execução. Este diálogo teria como participantes a aplicação e o gerenciador de interfaces.

2.1.6. Conclusões

Fica aparente que feedback semântico não é especificado explicitamente nas representações estudadas. Geralmente é esquecida a existência das mensagens trocadas no sentido computador-usuário.

Outra constatação é que não se pode separar totalmente diálogo da aplicação, exatamente por causa da necessidade de se obter feedback semântico na aplicação.

Com vistas a modelagem do diálogo, nos parece fundamental estabelecer previamente um Modelo Estrutural de diálogo para definir quais as informações poderão ser representadas no modelo conceitual para representação do diálogo no método a ser proposto. Observe que isto não é sempre explicitado nas representações usuais, ou seja, o

papel de criar mentalmente um modelo estrutural fica a cargo do modelador.

Como dito anteriormente, independência do diálogo não se consegue na criação de aplicações, pois sempre é necessário feedback semântico, o que torna a implementação do diálogo amarrada a implementação.

Não foi encontrado nenhum método que contempla-se a modelagem do diálogo de forma conjunta com a especificação de outros aspectos. Para realizar tal tarefa no método, tanto o modelo conceitual a ser proposto quanto a redundância de informação entre este e outros devem ser bem estudados.

Um modelo do diálogo não é por si só uma especificação completa da aplicação. Embora as atividades desta sejam tratadas no conteúdo das mensagens trocadas, não é especificado como estas são computadas. Da mesma forma, informações são trocadas, mas em momento nenhum é especificado como devem ser armazenadas ou calculadas.

A implementação de programas que facilitem a tarefa de projeto e implementação é de importância vital para a viabilidade de um método baseado na modelagem do diálogo, devido à complexidade das especificações criadas.

2.2. Dinâmica

Após traçarmos as características importantes no diálogo que podem influenciar na modelagem e implementação, procuraremos fazer o mesmo para dinâmica nesta seção.

A princípio nos interessa estabelecer se é possível criar modelos conceituais ortogonais para o diálogo e a dinâmica, e se estes apresentam as características de totalidade e conceitualização.

2.2.1. Eventos

Nos modelos conceituais estudados para representação da dinâmica, o conceito de evento se faz presente em todos os momentos: para que um sistema tenha características comportamentais dependentes do tempo especificadas é necessária a explicitação dos estímulos que ocorrem relacionados ao sistema.

Eventos são classificados comumente pela sua origem, ocorrência ou dependência mútua.

Dizemos que um evento tem origem externa quando é produzido fora do sistema, mas tem relevância na modelagem

deste, e interna, quando são produzidos no interior do próprio sistema modelado [Keuff91a].

Eventos podem ocorrer de forma contínua ou discreta [Ward86][Quent88], sendo que a diferenciação se dá quando verificamos se existe um intervalo de tempo onde o evento não ocorre, e aí constatamos seu caráter discreto. Caso contrário este será contínuo.

Outra forma de classificá-los, desta vez de maneira relativa, é verificar se a ocorrência de um evento está condicionada à ocorrência de outro [Flory78]. Neste caso chamamos a dependência entre estes de lógica. Se além disso sabemos que um evento ocorre depois do outro, a dependência é cronológica.

2.2.2. Critérios para uma representação bem-definida

Para garantirmos os requisitos de totalidade e conceitualização em um modelo conceitual, é necessário verificar quando a representação da dinâmica de um sistema está bem definida.

Num contexto onde modelamos sistemas baseando-nos em redes de transformações de dados [Ward86], onde cada uma das transformações recebe dados de entrada e envia dados de saída, que podem ser armazenados em repositórios de armazenamento, uma representação estaria bem definida quando :

1. A composição de entradas, saídas e repositórios de armazenamento estão bem definidos;
2. Uma função de entrada e saída para cada transformação está definida;
3. As transformações operam infinitamente rápido;
4. Repositórios de armazenamento tem capacidade infinita;
5. Transições chegando simultaneamente são aceitas seqüencialmente em ordem indeterminada.

No escopo ao qual este trabalho deseja atender, simultaniedade nunca ocorre, e a especificação funcional não é presa a apenas uma forma de representação. Portanto poderiam ser colocados novos critérios:

1. A modelagem de propriedades funcionais estáticas está bem definida;
2. A modelagem da dinâmica explicita o controle sobre as funções processadas pelo sistema;

3. Toda a modelagem é feita desprezando aspectos ligados à tecnologia de automação.

2.2.3. Representação da dinâmica

Normalmente são usados duas formas para representação nos modelos conceituais da dinâmica: D.E.T.s e diagramas de redes de petri.

Na primeira forma, eventos são associados às transições que são disparadas no momento da ocorrência destes. Obviamente não podem haver ciclos de eventos contínuos, para que não seja modelado uma situação de espera indefinida.

Cada D.E.T. está associado a uma estrutura formal, e é uma forma de representação visual, características que recomendam fortemente a sua utilização [Harel88].

A segunda forma de representação também está associada a uma definição formal, que tem estrutura chamada Rede de Petri [Peter81], R = (P, T, I, O), onde

P é um conjunto de lugares;

T é um conjunto de transições;

I é uma função de entrada que leva uma transição no conjunto de lugares que estão conectados de forma incidente nesta transição, I:T --> 2^P;

O é uma função de saída que leva uma transição no conjunto de lugares que estão conectados de forma emergente a esta transição, O:T --> 2^P.

Novamente esta é uma estrutura derivada do conceito de grafo (uma extensão deste). É uma linguagem para modelagem mais expressiva que máquinas de estados finitos, pois podemos mostrar que toda modelagem usando M.E.F.s pode ser feita usando Redes de Petri. Permitem que concorrência e sincronização de ocorrência de eventos sejam modeladas facilmente.

Embora tenham as características citadas acima, diagramas de redes de petri geram representações mais complexas que D.E.T.s e, a princípio, definem modelos conceituais menos inteligíveis ao usuário.

Na literatura sobre modelagem de dinâmica, ambas representações são usadas, como mostrado abaixo:

Modelo	Representação	Nível de abstração	Enfoque
[Flory78]	D.E.T.	baixo	informações
[Kung89]	D.R.P.	baixo	informações
[Quent88]	D.E.T.	alto	atividades
[Ward86]	D.R.P.	alto	atividades

No primeiro modelo, a representação da dinâmica é construída após a explicitação das tabelas do modelo relacional e das relações funcionais entre os atributos destas. O D.E.T. aparece da explicitação das dependências lógicas e cronológicas entre eventos.

A ocorrência de um evento é considerada o aparecimento de uma tupla de tabela. Os estados são considerados as configurações que o banco de dados do sistema pode assumir, sendo alterados pela ocorrência de eventos associados a transições, que tem associadas também a ativação de funções.

No segundo modelo, a representação da dinâmica é criada depois do modelo de entidades e relacionamentos do sistema. Lugares são criados na rede de conexões entre entidades e relacionamentos, e transições são disparadas por causa da ocorrência de eventos, correspondendo à disponibilidade de instâncias de entidades e ao cumprimento de pré e pós-condições.

No terceiro modelo, a representação da dinâmica é feita após a especificação das atividades do sistema, seja esta usando D.F.D.s ou uma lista estruturada de atividades. Cada estado está associado ao processamento de funções ou à momentos de espera por eventos externos ou internos.

O quarto modelo tem a representação da dinâmica feita agregando informações de controle ao D.F.D. de um sistema. Cada fluxo é considerado um lugar e transformações de dados (também chamados processos) são consideradas transições, que são disparadas pelo cumprimento de restrições de temporização e/ou de pré e pós-condições.

2.2.4 Conclusões

Na tabela mostrada na seção anterior, vê-se que comportamento dinâmico modelado a partir de informações é sempre feito em um nível de abstração baixo, e, a partir de atividades, em um nível de abstração alto.

Portanto modelagem da dinâmica baseada em informações não pode ser usada como conceitual, exatamente por pecar ao representar uma pequena quantidade de informações relevantes ao usuário, por causa do baixo nível de abstração destas.

As representações baseadas em atividades estudadas, mostradas na tabela da seção anterior, não se adequam a modelagem de todas as classes de aplicações porque, nos casos extremos, de pouca ou muita funcionalidade no sistema, temos representações muito pobres ou extremamente complexas.

Verificou-se com este estudo de modelagem da dinâmica que esta é elaborada sempre considerando um método para desenvolvimento de sistemas, o que não acontece normalmente com a modelagem do diálogo.

Na literatura sobre dinâmica foram encontradas apenas duas formas de representação, enquanto que para diálogo foram encontradas várias, o que mostra que as formas de representação da dinâmica são mais satisfatórias e concensuais.

Constatou-se também que sempre existe uma causa para a ocorrência de comportamento dinâmico, sendo estas temporização, controle ou diálogo, e isto nos leva a acreditar que seria mais interessante considerar a modelagem destas características de forma fatorada, evitando confusão com outras características representadas no diagrama.

No próximo capítulo mostraremos como a modelagem do diálogo pode ser feita, e no capítulo 4 será mostrada como a dinâmica pode ser representada, baseada nesta construção anterior.

Capítulo 3

Os sistemas BIBLOS e DIAGRAMA: visões da modelagem do diálogo

Neste capítulo estudaremos os sistemas BIBLOS e DIAGRAMA do ponto de vista da modelagem do diálogo. Embora a intensão do autor deste último sistema seja especificar a dinâmica, podemos encará-lo da forma colocada acima, já que o diálogo é a causa para o comportamento dinâmico daquele sistema.

Nosso objetivo é alcançar um método coerente e útil para modelagem de sistemas interativos, e para isto serão apontados, na modelagem dos dois sistemas, pontos que contribuam para estabelecer um método mais completo.

3.1. O sistema BIBLOS

O sistema BIBLOS [Fortu90] foi projetado para realizar o controle sobre informações relacionadas a referências bibliográficas, no intuito de recuperar estas informações quando desejado.

Como mostrado na figura 6, foi considerada a existência de três tipos de referência biblografica: artigos, periódicos e edições destes últimos. Cada artigo seria registrado fazendo referência à edição do periódico na qual está incluído. Ao manipularmos periódicos estaríamos

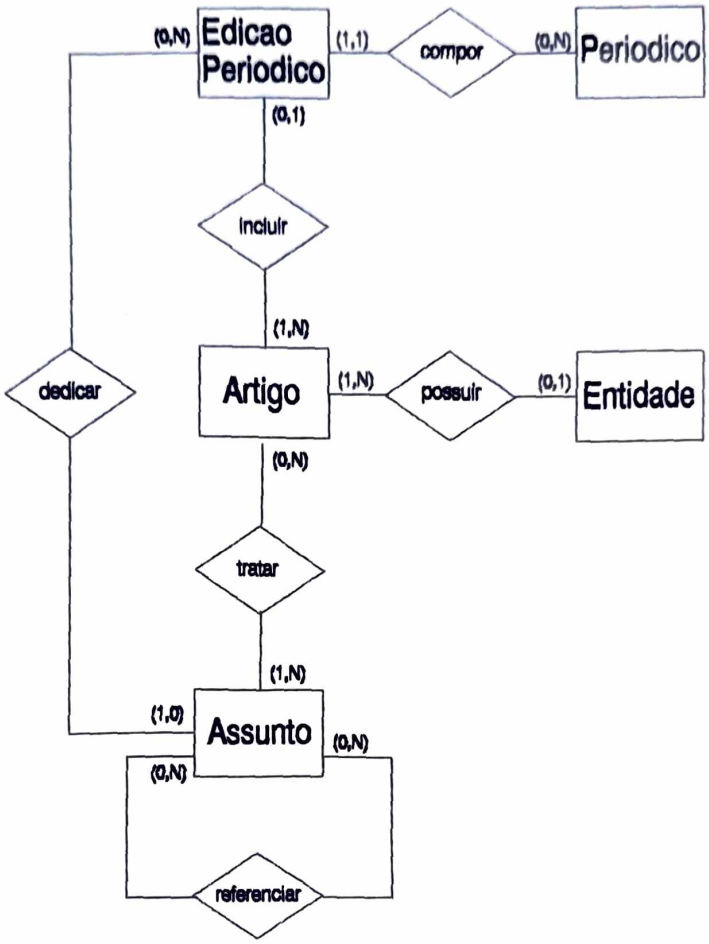


FIGURA 6-MER do Sistema BIBLOS

Periodico(per#, titulo, titulo_abrev)
Artigo(titulo, autor, per#, vol, num, epoca, ano, ent_id, pg_inic, pg_final,
rg, ass_cod1, ass_cod2, ass_cod3, ass_cod4)
Entidade(ent_id, nome, end, tel)
Assunto(ass_cod, descr)
Referenciar(ass_refte, ass_refado)
Dedicar(per#, id, num, epoca, ano, ass_cod)

FIGURA 7-Tabelas do Modelo Relacional do Sistema BIBLOS

F1. Processa consulta
- F1.1. Consulta artigo
  - F1.1.1. Define Escopo
  - F1.1.2. Colsulta artigos na tela
  - F1.1.3. Consulta artigos na impressora
- F1.2. Consulta periodicos
  - F1.2.1. Consulta periodicos na tela
  - F1.2.2. Consulta perioriodicos na impressora
- F1.3. Consulta assunto
  - F1.3.1. Consulta assunto na tela
  - F1.3.2. Consulta assunto na impressora
- F1.4. Consulta entidade
  - F1.4.1. Consulta entidade na tela
  - F1.4.2. Consulta entidade na impressora
- F1.5. Consulta edicao de periodico
  - F1.5.1. Consulta edicao de periodico na tela
  - F1.5.2. Consulta edicao de periodico na impressora

F2. Processa atualizacao
- F2.1. Processa inclusao
  - F2.1.1. Inclue artigo
  - F2.1.2. Inclue periodico
  - F2.1.3. Inclue entidade
  - F2.1.4. Inclue assunto
  - F2.1.5. Inclue referencia
  - F2.1.6. Inclue edicao de periodico
- F2.2. Processa exclusao
  - F2.2.1. Exclue artigo
  - F2.2.2. Exclue periodico

  ⋮
- F2.3. Processa alteracao
  - F2.3.1. Altera artigo
  - F2.3.2. Altera periodico

  ⋮

F3. Processa utilitario

FIGURA 8- Lista Estruturada de Atividades do Sistema BIBLOS

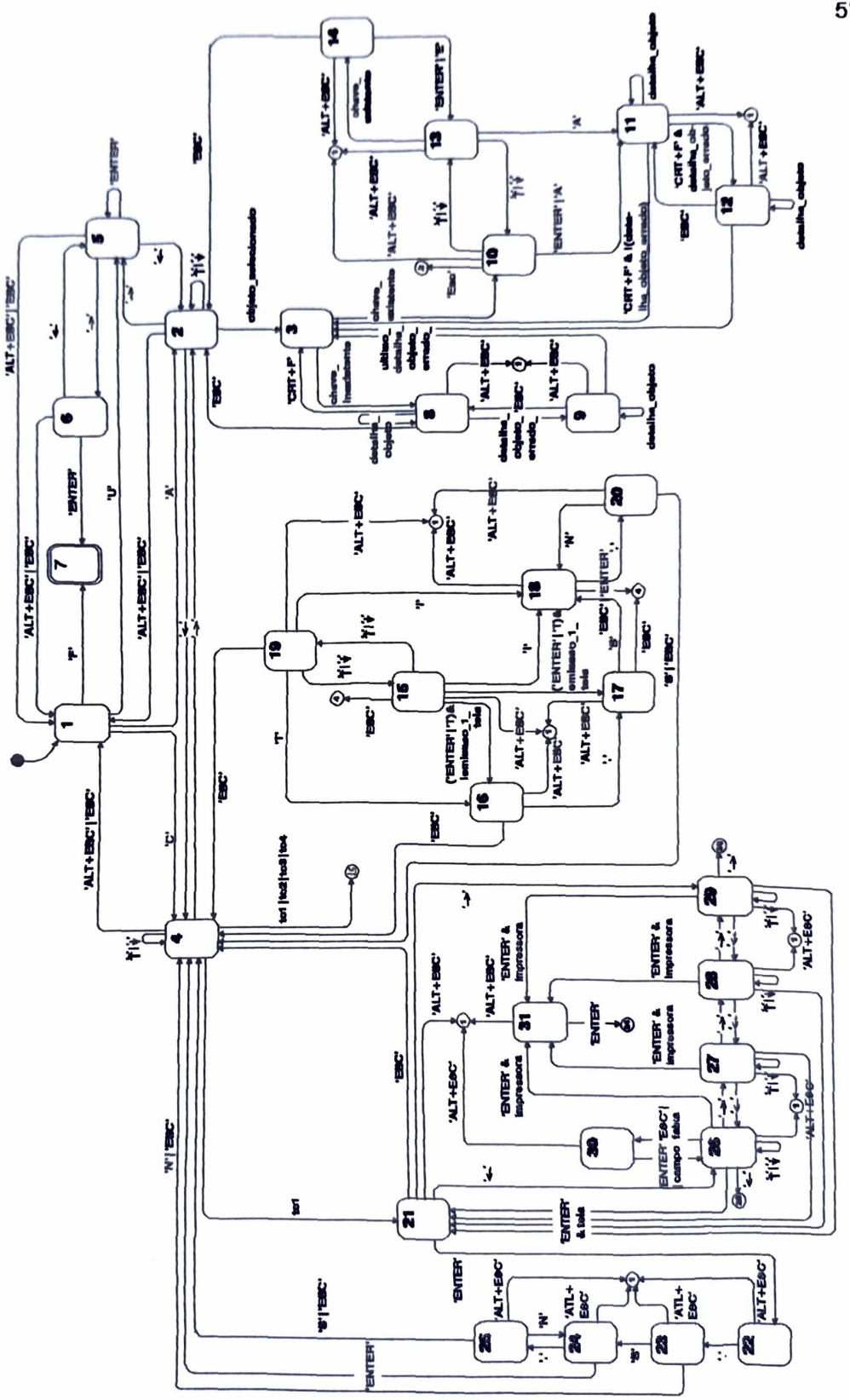


FIGURA 9-Modelagem do Dialogo do Sistema BIBLOS

trabalhando com suas características, presentes em todas as edições existentes (por exemplo o nome do periódico).

Poderia ser registrado que uma edição do periódico é dedicada a um assunto específico e que artigos tratam de um ou mais assuntos. Cada assunto poderia referenciar outro assunto, como explicitado pela relação "referenciar".

Ainda na figura 6 aparece modelado, através do relacionamento "possuir", a necessidade de armazenar o local onde pode ser encontrado cada artigo, visando facilitar a tarefa de obter uma referência para consulta.

Como aparece bem registrado em [Fortu90], a princípio tentou-se aplicar técnicas convencionais para especificação das características funcionais do sistema, como as baseadas na determinação dos processos [Demar89] ou atividades e informações [Ward84]. Entretanto foi constatado que usando estas formas de especificação, o sistema estaria sendo modelado de forma muito pobre, deixando de lado a função de comunicação do sistema com o usuário.

Desta forma foram representadas as atividades de processamento de informação usando uma lista estruturada de atividades, mostrada na figura 8 e feita a representação do diálogo usando diagramas de estados e transições, de maneira

semelhante ao mostrado na figura 9, representando de forma fatorada esta função.

Comprovou-se naquela especificação do BIBLOS que a modelagem do diálogo era fundamental para uma especificação mais completa do sistema e que existia muita semelhança entre a organização da representação do diálogo e o diagrama hierárquico de módulos produzido.

3.2. Modelagem do diálogo

Estaremos descrevendo aqui a modelagem do diálogo apresentada em [Fortu90] para o sistema BIBLOS, procurando consolidar este modelo para utilização em outros sistemas.

No diagrama mostrado na figura 9, estados são representados por bolhas, como apresentado na notação colocada no Apêndice, indicando cada um a situação em que se encontra o diálogo entre o usuário e a aplicação em um determinado instante. As transições, representadas por setas, estão sempre associadas à respostas do usuário, ou seja, à emissão pelo usuário de uma mensagem, podendo ser o pressionar de uma tecla ou uma cadeia de caracteres, ou mesmo a seleção de um objeto.

Na especificação do sistema BIBLOS, sempre era representada a resposta do usuário associada a transição, seja pela especificação do nome da tecla que deveria ser pressionada (representada entre aspas) ou pelo nome da mensagem trocada.

No diagrama da figura 9, entre os estados 1 e 2 é mostrada a resposta associada à transição que liga estes dois estados, ou seja, a tecla 'A'. Quando uma mensagem era trocada, seu nome era associado à transição, como na transição entre os estados 2 e 3 na figura 9.

Entretanto, este tipo de representação do nome da mensagem levava a ocorrência de não-determinismos, necessitando-se de anotações ao diagrama para explicitar a seqüência do diálogo. Na figura 9.a as anotações seriam as inscrições existente e inexistente, se referindo a chave digitada pelo usuário no estado 3.

As respostas podem ser agrupadas usando construções da lógica sentencial, ou seja, os conectores lógicos AND, OR e NOT, representados no diagrama por &, ¦ ou !.

Obviamente, quanto mais informações semânticas representadas, mais o diagrama estará amarrado à especificação de um sistema específico, diminuindo o reuso,

tanto de especificação quanto de implementação do seqüenciamento do diálogo. Entretanto a utilização destas é quase sempre inevitável, para estabelecermos não determinismos.

No modelo usado para a especificação do BIBLOS podem ser representadas todas as seqüências de diálogo possíveis, em sistemas interativos comuns, garantindo que estes possuem a característica de totalidade.

Outra característica de interesse neste modelo é a possibilidade de eliminar detalhes de implementação através do uso da especificação do nome da mensagem. Por exemplo, observe na figura 10 que o estado 4 poderia ser substituído por 5 estados, tornando o diagrama mais complexo. Esta característica garante a propriedade de conceitualização do modelo original e do alterado.

No modelo descrito em [Fortu90], a única forma de redundância de informação passível de representação são as conexões, simbolizadas por um círculo com o número do estado de destino. Estas podem aparecer em vários pontos do diagrama, com destino idêntico, como por exemplo a ligação de todos os estados ao inicial, pela transição disparada por ´ALT+ESC´. Este tipo de construção nos permite particionar o diagrama, seguindo critérios do modelador.

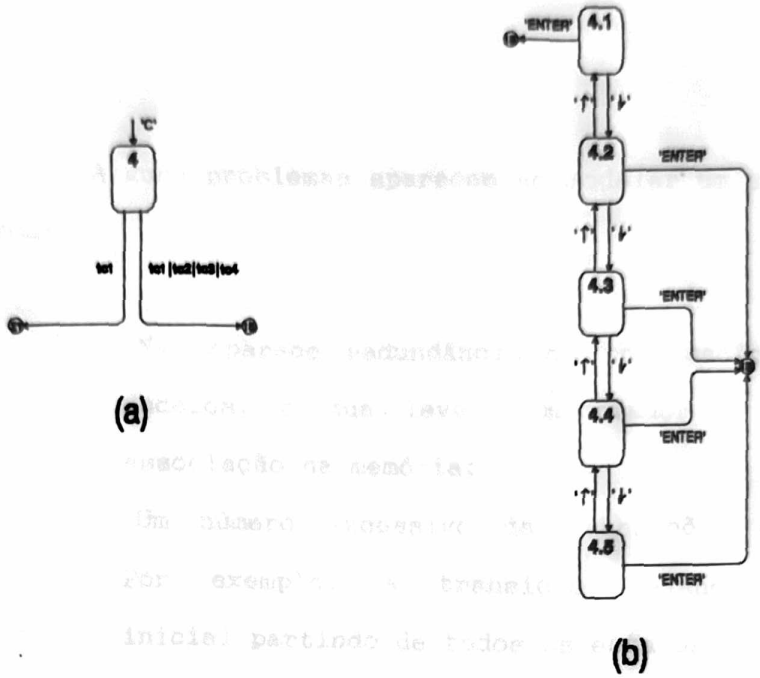


FIGURA 10-Escolha do nivel de abstracao desejado no Modelo

Alguns problemas aparecem ao modelar um sistema desta forma:

1. Não aparece redundância de informação com outros modelos, o que leva o modelador a manter esta associação na memória;
2. Um número excessivo de transições é modelado. Por exemplo, a transição levando ao estado inicial partindo de todos os estados.

Tais problemas serão resolvidos, com o método proposto no próximo capítulo.

3.3. O sistema DIAGRAMA

O sistema DIAGRAMA foi desenvolvido para automatizar a tarefa de construção de diagramas e foi usado para experimentação do método proposto em [Quent88].

Os elementos básicos de cada diagrama que podem ser criados pelo sistema são setas, legendas, linhas, ligações ou objetos, sendo estes últimos compostos pelos outros quatro. As tarefas de criação, recuperação e edição de diagramas são suportadas pelo ambiente que possui uma interface gráfica com

o usuário, e estilo de interação conversacional, baseado nas respostas do usuário através do teclado.

A modelagem deste sistema aparece mostrada em [Quent88], e foi baseada no método proposto em [Ward84], especificando as atividades e informações presentes no sistema. Devido a importância do comportamento dinâmico neste sistema, foi usado um recurso adicional para a especificação da dinâmica, baseada nos diagramas de estados e transições, como mostrado na figura 11.

Esta especificação da dinâmica está fundamentada na associação de estados e transições a cada função do sistema. Estas funções devem ser organizadas de maneira hierárquica, impondo também este tipo de organização ao conjunto de D.E.T.s.

Considera-se que cada máquina de estados finitos especificada faz o papel de atividade de controle daquele nível da decomposição funcional.

Para isto, são associados às transições eventos externos, na forma de respostas do usuário (representadas entre chaves na figura 11), e internos, podendo também ser associada uma informação de controle sobre as funções do sistema, controle este que ocorreria quando a transição fosse

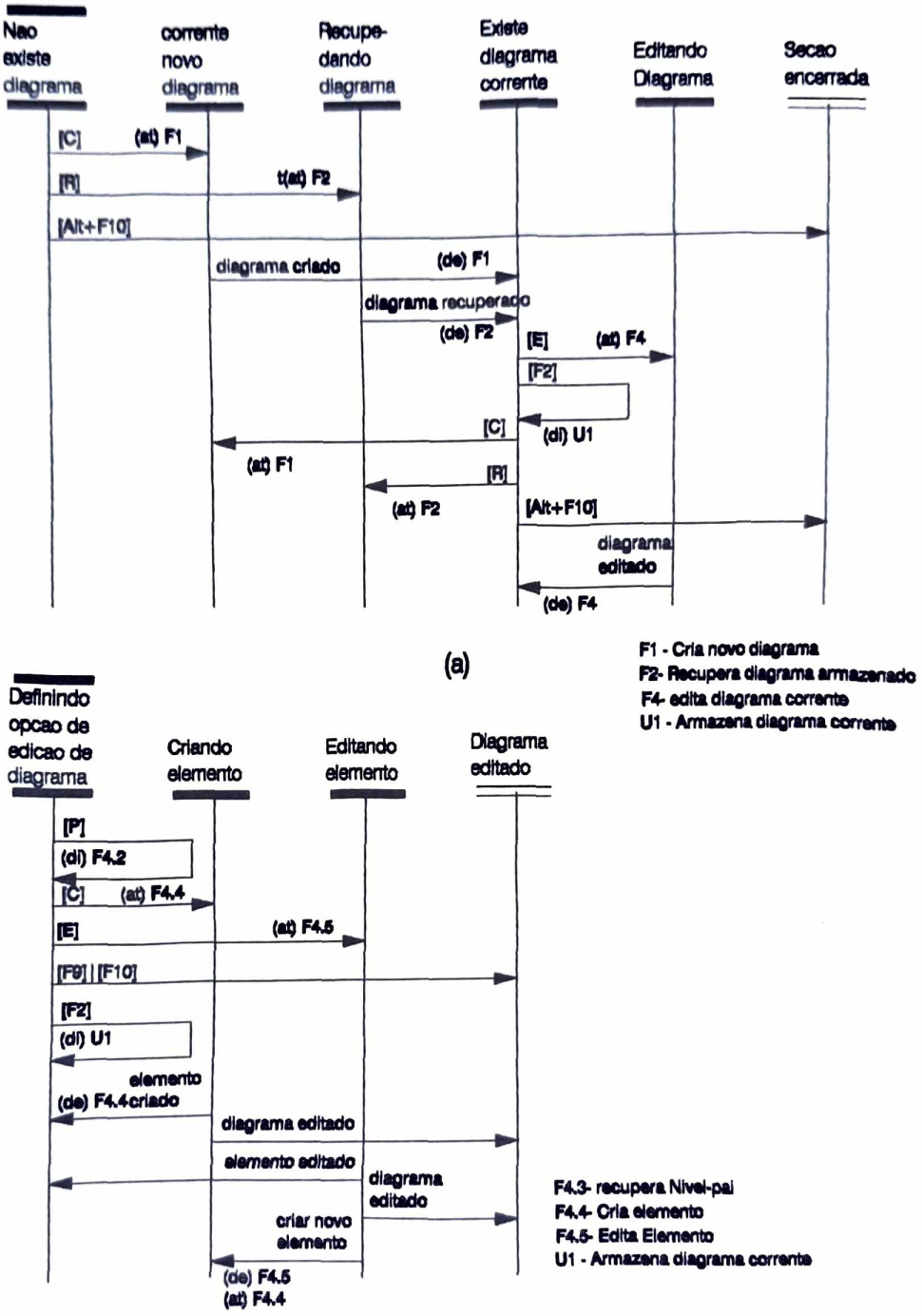


FIGURA 11-Exemplos de Modelagem da Dinamica no sistema DIAGRAMA [Quent89]

disparada. Estas informações de controle seriam a ativação, desativação, disparo, suspensão e reativação de uma função.

Cada estado em um dos diagramas estaria associado à um momento de espera por uma resposta do usuário ou a decisão sobre qual o processamento realizado logo em seguida a resposta, sendo estes chamados inativos, ou associado ao processamento de uma função, sendo estes estados chamados ativos. Estados ativos poderiam ou não ter o controle da função associada detalhado por um D.E.T..

O controle seria passado a um diagrama no nível abaixo do atual quando atingíssemos um estado ativo. Neste ponto seria iniciada a atividade de controle da função associada a este estado ativo, e a respectiva M.E.F. iniciaria sua operação, a partir do seu estado inicial.

O controle seria devolvido ao nível acima do atual quando a M.E.F. atingisse um de seus estados finais. Neste momento o diagrama no nível acima se tornaria o corrente e ocorreria uma transição ativada por uma condição de nome igual ao estado final atingido no nível abaixo.

Se observarmos esta representação com uma visão crítica, constataremos que a redundância de informações representadas é excessiva, levando a um conjunto de diagramas

bastante complexo, ou seja, com excesso de estados e transições.

Se observarmos esta representação do ponto de vista de alguém que deseja modelar o diálogo, veremos que este assume uma representação hierárquica, que não é típica de sua natureza (diálogo é particionável e não hierarquizável), e aparece representado de forma unidirecional, apenas no sentido usuário-aplicação.

3.4. Derivando uma represe-tação menos complexa

Como dito anteriormente, procuraremos agora encarar a modelagem feita para o sistema DIAGRAMA do ponto de vista do especificador do diálogo, mostrando que os diagramas de estados e transições daquele sistema podem ser reduzidos à forma utilizada no sistema BIBLOS, se omitirmos temporariamente as informações de controle das funções do sistema.

Para tal, mostramos nas figuras 12, 13, 14, 15, 16 e 17 os D.E.T.s do sistema DIAGRAMA, segundo a modelagem representada em [Quent88], porém usando a notação convencionada neste texto oriunda do sistema BIBLOS, omitindo

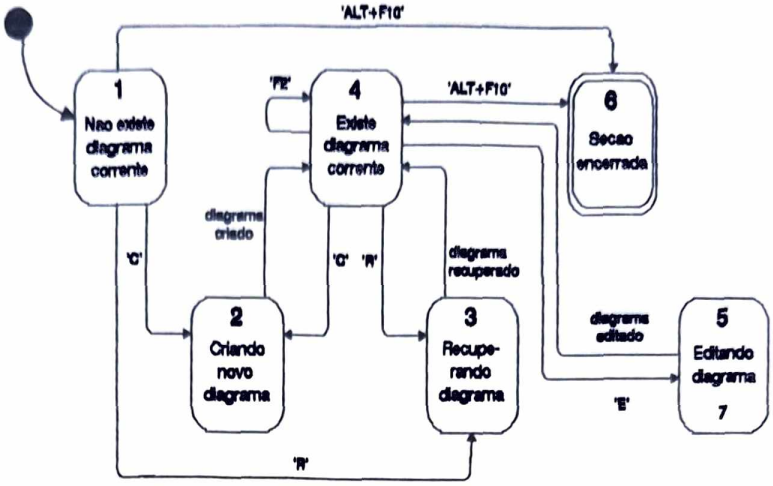


FIGURA 12. Funcao 'Cria diagramas' do sistema DIAGRAMA

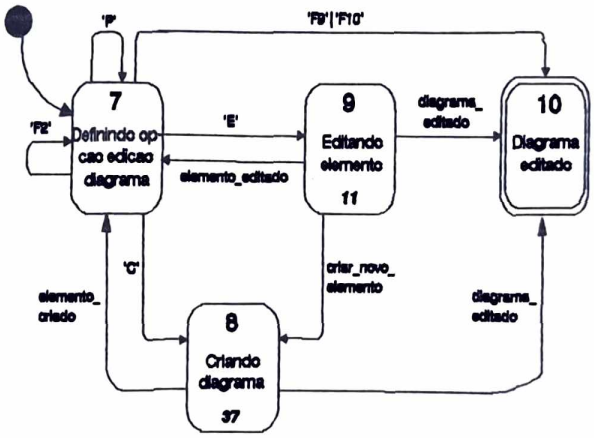


FIGURA 13. Funcao 'Controla Edicao de Diagrama' do Sistema Diagrama

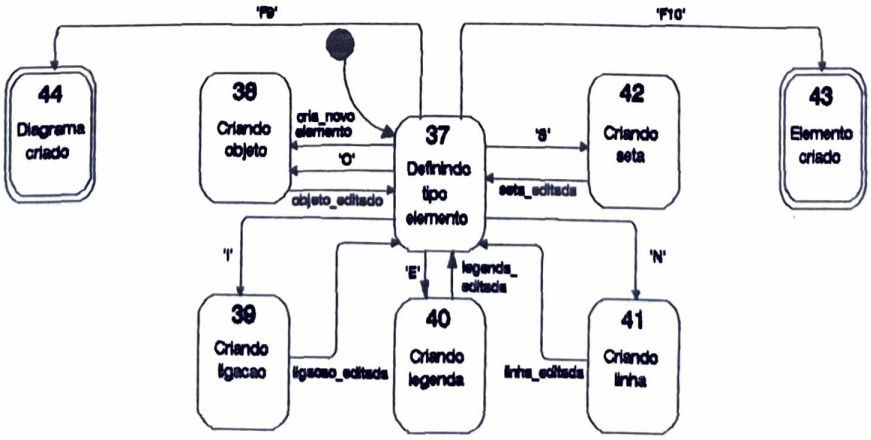


FIGURA 14. Funcao 'Cria elemento' do Sistema DIAGRAMA

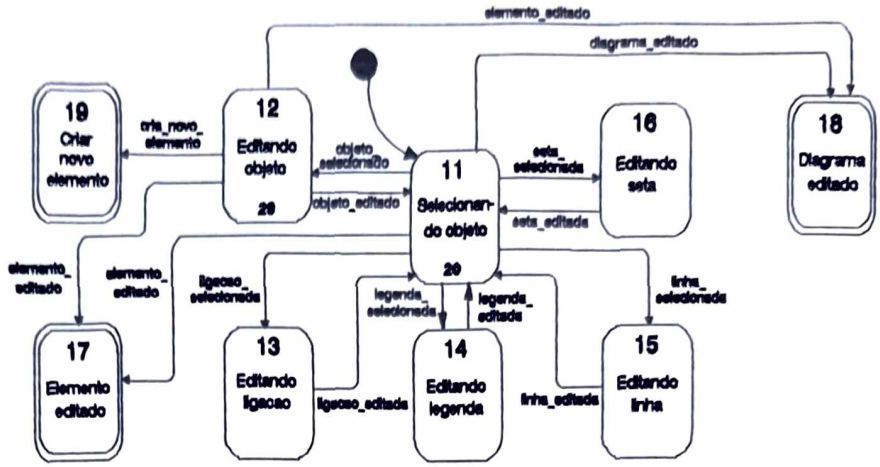


FIGURA 15. Funcao 'Edita elemento' do Sistema DIAGRAMA

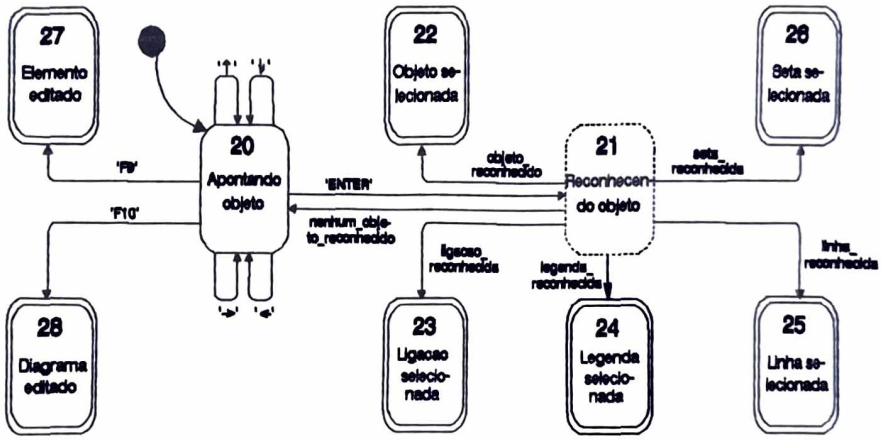


FIGURA 16. Funcao 'Seleciona elemento' do Sistema DIAGRAMA

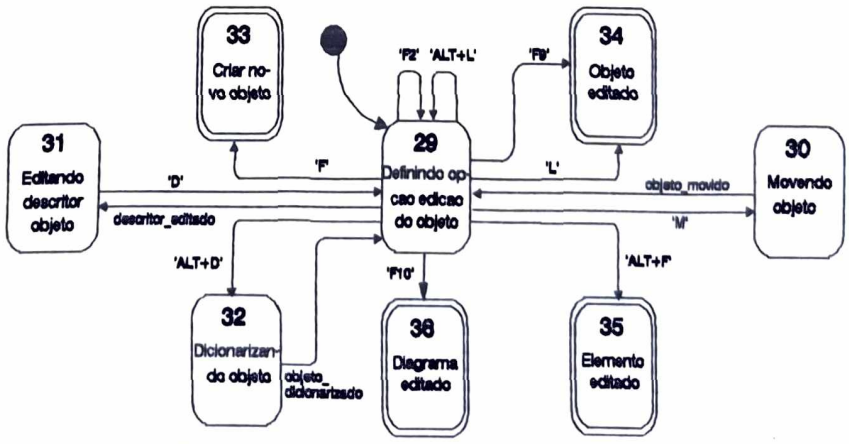


FIGURA 17. Funcao 'Edita objeto' do Sistema DIAGRAMA

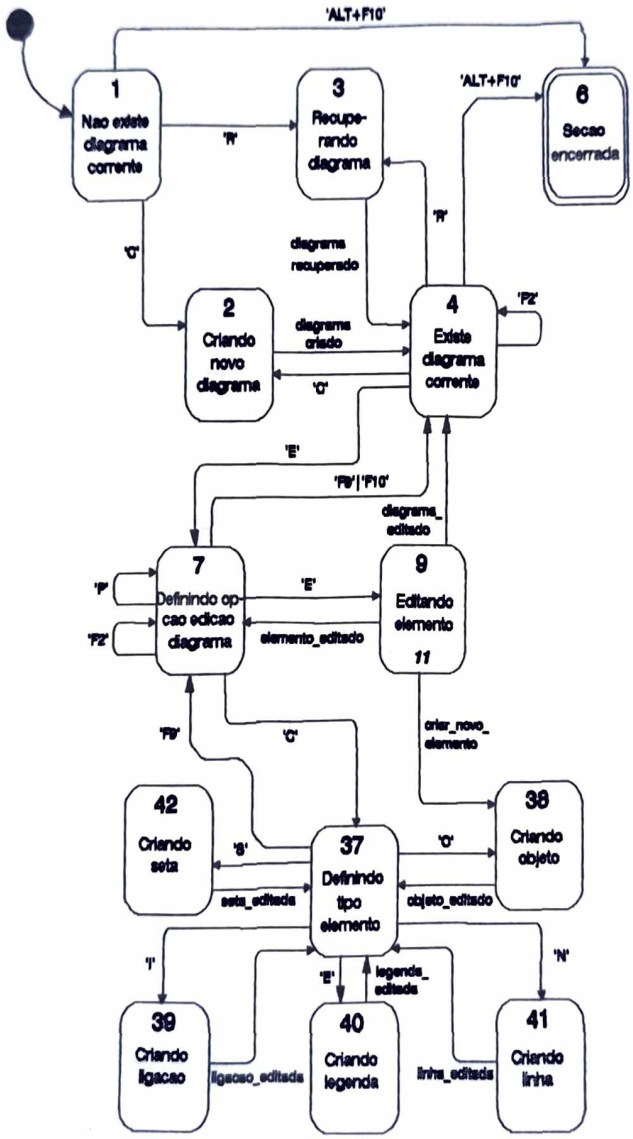


FIGURA 18. Derivacao parcial No. 1 da Modelagem do Dialogo do Sistema Diagrama

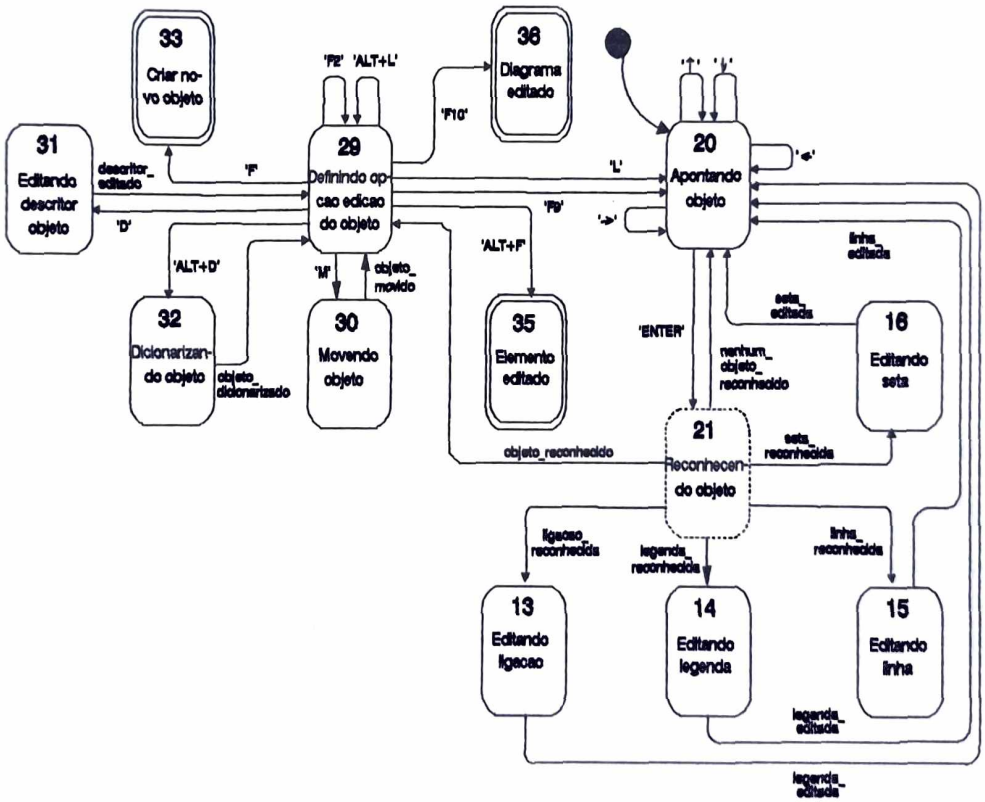


FIGURA 19. Derivacao parcial No. 2 da Modelagem do Dialogo do Sistema Diagrama

ativações, desativações, disparos, suspensões e reativações de funções.

Já que as informações que associam um diagrama a outros são exatamente o controle sobre as funções, convencionamos colocar o número do estado que detalha um estado ativo abaixo do nome deste estado, contornando assim a ausência explicada acima.

Este processo de derivação consiste em substituir estados ativos que tiveram o controle detalhado através de um D.E.T. pelo diagrama inteiro que o detalha. Assim, conectando as transições incidentes neste estado ativo ao estado inicial do D.E.T. associado e as transições que levam a estados finais aos estados de destino das transições com condição idêntica ao nome destes estados finais, estaremos eliminando vários estados e transições.

Por exemplo, ao substituirmos o estado 5 na figura 12 por todo o diagrama da figura 13, a transição chegando a 5 passa a chegar ao estado 7, e a transição saindo do estado 9 passa a estar incidindo no estado 4. Com isto desaparecem os estados 5 e 10 e a transição do estado 5 ao estado 4.

Todo este processo de substituição continua, até que nenhum estado ativo que foi detalhado na especificação

(aqueles com o número em itálico abaixo do nome do estado) reste na representação. Estas substituições podem ser acompanhadas na figuras 18, que é a junção dos diagramas mostrados nas figuras 12, 13 e 14, e na figura 19, que é a junção das figuras 15, 16 e 17. Finalizando o processo, o estado 9 na figura 16 é substituído pelo diagrama que aparece na figura 17, e o resultado final é mostrado na figura 19.

Neste processo de derivação de um diagrama único onde apareça representado todo o diálogo no sistema, foram eliminados 22 estados e 20 transições, o que nos mostra que foi realizada uma considerável simplificação. Observe que todas as transições eliminadas foram aquelas cuja condição era um evento interno, o que mostra que estes na realidade eram requerimentos para a corretude da construção hierárquica, e não eventos de fato.

No sistema DIAGRAMA, estados estavam associados ao processamento de controle. Entretanto no seqüenciamento do diálogo, cada estado é uma situação durante o processo de comunicação. Portanto poderemos suprimir também todos aqueles estados ativos que aparecem modelados na figura 20, ou seja, aqueles que existiam só para indicar que em determinado instante uma função estaria sendo processada.

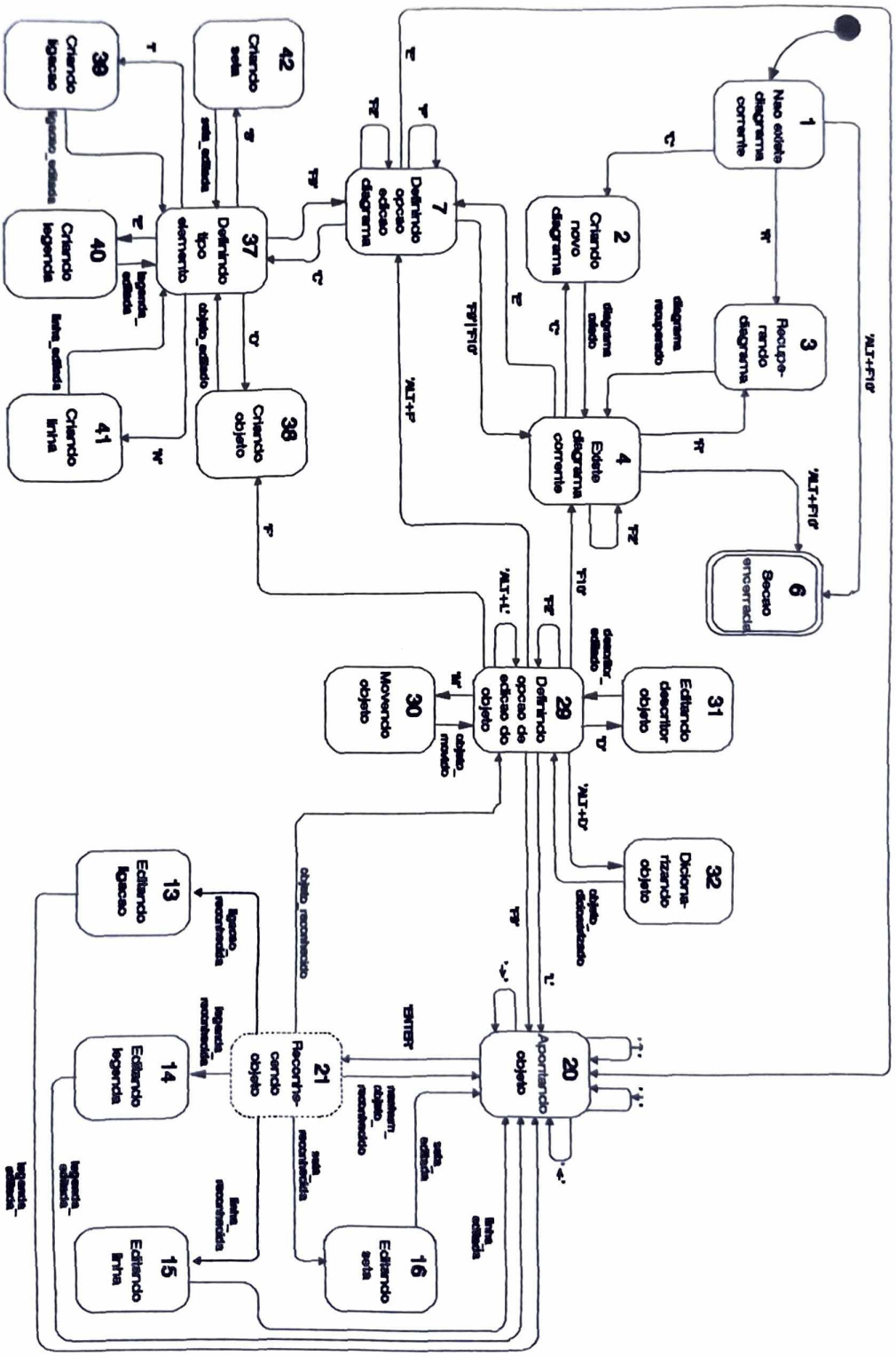


FIGURA 20. Derivacao parcial da Modelagem do Dialogo do Sistema DIAGRAMA

Esta supressão é feita retirando o estado e a transição que partem deste, e conectando as transições que incidem neste estado ao destino da transição suprimida. Podem aparecer casos em que haja necessidade de alterar as condições associadas às transições, explicitando o nome da mensagem trocada como resposta pelo usuário.

A construção derivada a partir desta hipótese é mostrada no diagrama da figura 21. Para ilustrar, tomemos por exemplo o estado 42 da figura 20. Este é um estado ativo, e portanto podemos suprimi-lo, retirando o estado e a transição com origem neste, e ligando a transição que parte do estado 37 ao estado de destino da transição que parte do estado 42, ou seja, ao próprio estado 37. O resultado é mostrado na figura 21, como uma transição que tem origem e destino no estado 37.

Concluímos assim que existe uma conversão mecânica de todas as construções criadas usando o modelo proposto em [Quent88] para o modelo proposto em [Fortu90], do ponto de vista do diálogo, sendo este último modelo o mais aconselhável para utilização, por ser mais simples.

No próximo capítulo mostraremos como representar outras características usando uma extensão do modelo

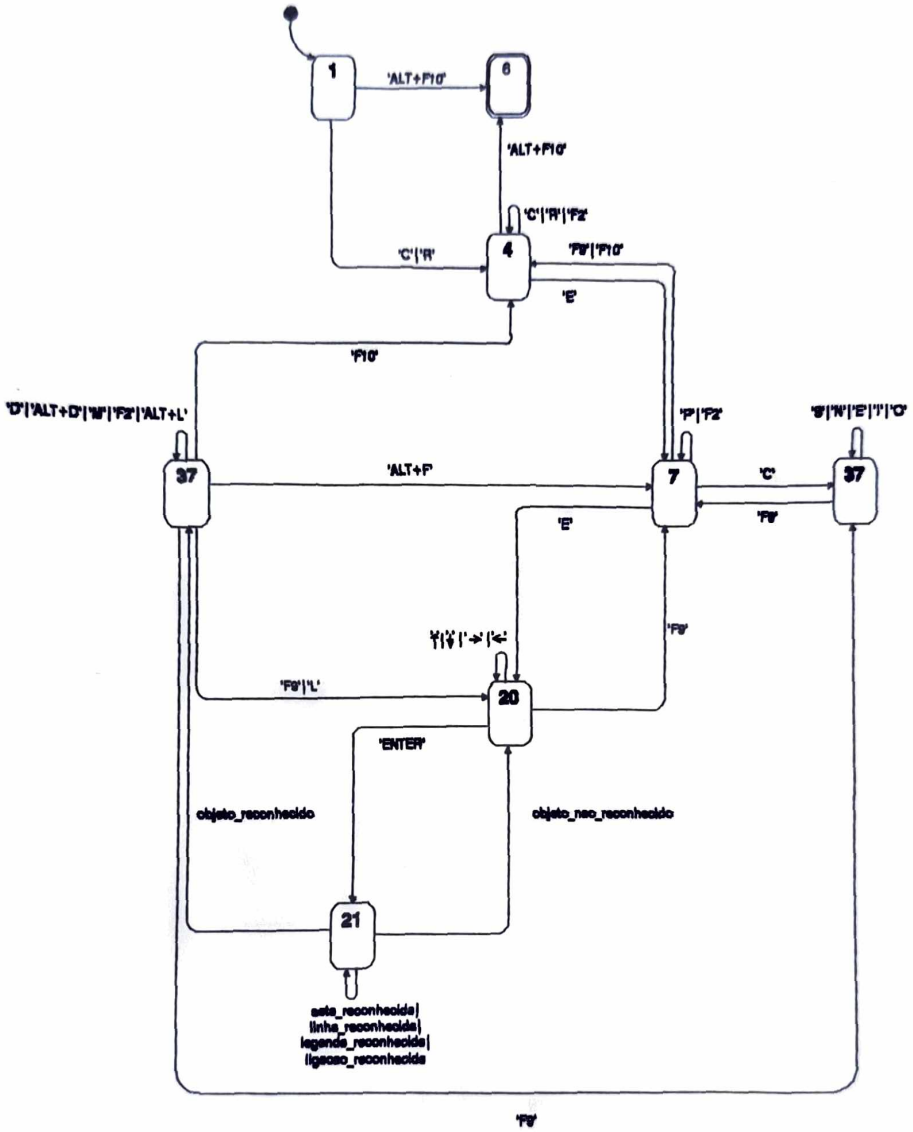


FIGURA 21. Modelagem do Dialogo do Sistema DIAGRAMA

apresentado aqui. Entre estas características está, por exemplo, o diálogo no sentido aplicação-usuário.

Capítulo 4

O método proposto

Como visto no capítulo anterior, mesmo os métodos estudados para especificação do diálogo são falhos em alguns aspectos, como por exemplo, a especificação do diálogo no sentido aplicação-usuário.

Portanto, neste capítulo estaremos preocupados em definir uma forma de especificação destas características, e estabelecer um método para desenvolvimento de sistemas que nos permita modelar sistemas e ter uma implementação válida e correta.

4.1. Alguns conceitos do método essencial

Durante todo o trabalho foram citados alguns termos que fazem parte do jargão enunciado em [Ward84], definindo o método essencial de análise de sistemas.

Naquele texto são consideradas duas características em sistemas que devem necessariamente que ser considerados durante o processo de modelagem de um sistema, que são essência e encarnação.

Essência se refere à descrição das características próprias do sistema, em seu estado natural, sem componentes

tecnológicos influenciando no seu comportamento, e independentes dos detalhes de implementação de uma aplicação computacional.

Encarnação se refere à uma implementação particular de uma aplicação que automatize, em parte ou totalmente, as tarefas presentes no sistema.

Dentro destes conceitos, enuncia-se alguns princípios que devem nortear a modelagem de sistemas [Keuff91a][Azeve90], que são:

. Lei do orçamento da complexidade: um diagrama deve ser compreensível, ou seja, permitir que a capacidade cognitiva do leitor das especificações as compreenda diretamente, ou seja, que estas tenham entre 5 e 9 objetos;
. Neutralidade tecnológica: não são considerados aspectos tecnológicos;
. Tecnologia interna perfeita: não há limitações impostas pela tecnologia;
. Modelo essencial minimal: se existem vários modelos para a especificação de um sistema, deve se escolher o menos complexo.

Obviamente toda a modelagem do diálogo deve estar pautada nestes princípios.

A medida que identificamos as interfaces já existentes entre os elementos do sistema e especificamos as mensagens trocadas nestas, estaremos modelando o diálogo essencial aí presente. Como vimos anteriormente, toda a dinâmica em um sistema interativo, caracterizado no escopo deste trabalho, é guiada pelo diálogo, e portanto propõe-se considerar o diálogo uma característica essencial na modelagem de sistemas.

Um fator que nos permite realizar a afirmação acima com caráter mais enfático é a colocação do motivos para se considerar a dinâmica essencial, colocadas em [Quent88]. Totas as justificativas naquele texto se referem a interface e o diálogo, e por estas razões, podemos afirmar com mais certeza a essencialidade do diálogo na modelagem de sistemas interativos, e ainda mais ao especificar posteriormente a dinâmica do sistema, usando como base o modelo do diálogo, como será mostrado na seção 4.3.

Entretanto, convém lembrar que, ao considerarmos o diálogo essencial, não estamos negando que possam haver

classes de aplicações em que as duas características sejam essenciais.

No método essencial propõe-se que sejam criados três modelos principais do sistema: um modelo de essência, representado as características essenciais do sistema, um modelo de implementação, considerando aspectos tecnológicos na modelagem, e um modelo de automação, que seria a aplicação que automatizaria tarefas do sistema.

O modelo de essência estaria subdividido em um modelo de contexto e um modelo de comportamento, e a construção do primeiro estaria fundamentada da especificação do contexto onde o sistema modelado estaria inserido e dos eventos externos aos quais o sistema deveria produzir respostas. O modelo de comportamento especificaria as características comportamentais e o padrão organizacional do sistema.

Para tal, uma estrutura para criação dos modelos foi estabelecida, baseada na especificação das atividades de processamento de dados existentes no sistema, os chamados processos, e as informações relevantes para a especificação do padrão organizacional.

4.2. Um modelo estrutural para o diálogo

Como colocado no início do texto, o estabelecimento de um modelo estrutural de um processo geral que terá ocorrências particulares especificadas em um modelo conceitual nos ajuda a definir quais as características fundamentais que necessitam ser modeladas conceitualmente.

No caso do diálogo, poderiam ser propostos vários modelos estruturais, e todos estes considerariam a existência de prompts, echos e mensagens [Harts89a]. Na modelagem do sistema BIBLOS [Fortu90] foram propostos, mas não usados, os seguintes elementos: escopo, prompt, resposta e feedback.

No nosso caso, optou-se por um modelo que contemplasse várias fazes do processo de comunicação: a definição sobre o assunto que seria tratado, a indicação de que a aplicação estaria a espera de uma mensagem do usuário, a própria mensagem do usuário e o estabelecimento de informações a respeito dos objetos presentes no sistema. Estas fases receberam o nome de prompt, escopo, resposta e feedback, como mostrado na figura 22.

Enfatizando o caráter estrutural dado a este modelo, cada seta representada na figura 22 denota um conjunto de

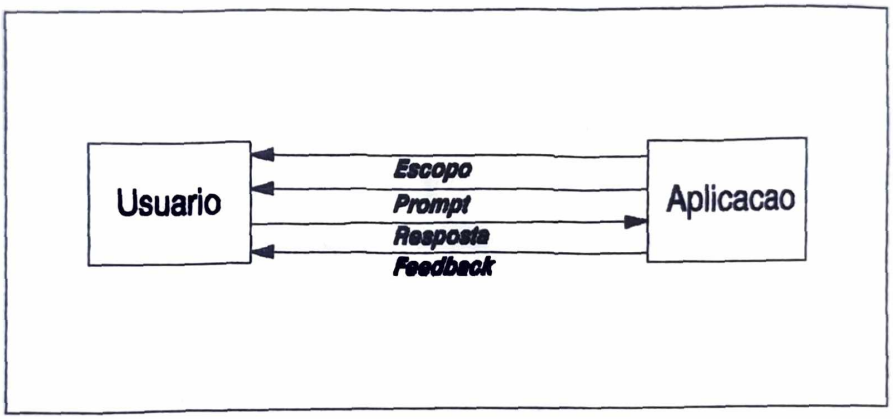


FIGURA 22-Um Modelo Estrutural para o Dialogo

informações trocadas, cada qual composta de mensagens entre os participantes.

Pode-se observar que nem todas estas características estão presentes na modelagem do sistema DIAGRAMA, e é exatamente aí que constata-se a importância do estabelecimento de um modelo estrutural: explicitando as mensagens trocadas em um sistema específico estaremos completando o modelo deste sistema com todas as informações envolvidas no processo de comunicação com o usuário. A esta especificação chamaremos seção detalhada da especificação do diálogo.

Podemos então associar a cada estado uma 5-tupla, (número, escopo, prompt, resposta, feedback), informando quais as informações trocadas no sentido aplicação-usuário, ou resumidamente (N, E, P, R, F). Esta afirmação pode causar estranheza a primeira vista, já que resposta é uma mensagem trocada exatamente no sentido oposto, como falado no capítulo anterior, entretanto sua especificação aqui se dá em um nível e com objetivo diferente do presente na definição da M.E.F. que modela o Diálogo usuário-aplicação.

Esta diferenciação se torna clara ao estabelecermos um exemplo para o caso do sistema BIBLOS:

```
-------------------------------------------------
Estado 1
E: mensagem('Sistema BIBLOS');
P: mensagem('Escolha a função desejada:');
R: opção;
F: se (não_existem_dados_cadastrados e opção = 'C')
  então
      mensagem('Não é possível realizar consultas,
              pois não existem dados cadastrados');
   fim-se;
-------------------------------------------------
```

A resposta do usuário é vista como uma variável, que contém a mensagem emitida pelo usuário a aplicação. É usada para a determinação das mensagens emitidas feedback.

Embora o número do estado não esteja presente no modelo estrutural, faz o papel de redundância controlada mínima para que haja uma associação entre o D.E.T. e esta especificação detalhada.

Escopo e prompt poderiam ser unidos em uma mensagem única, mas devido ao papel semântico distinto desempenhado pelos dois, optou-se por mantê-los em separado. Um define o contexto de uma comunicação e o outro apenas informa que a resposta está sendo aguardada.

Feedback é a construção que permite a troca de informações ricas em semântica da aplicação, antecipando o projeto desta característica, e facilitando o estabelecimento de um código do diálogo mais separado da aplicação do que sem a sua especificação.

Podemos colocar uma definição formal para a sintaxe desta especificação detalhada do diálogo, usando EBNF:

```
<DiálogoAplicaçãoUsuário> = ["(" <número> ","
                            [<escopo>] [[","] <prompt>
                            ["," <resposta>
                             "," <feedback>]]] ")"]
<prompt> = <mensagem> ";"
<escopo> = <mensagem> ";"
<resposta> = variável
<feedback> = <seleção> ";"
<seleção> = <condição> | <escolha>
<condição> = se <expressão> então <SeqüênciaComandos>
           [senão <SeqüênciaComandos>] fim-se
<escolha>  = caso <expressão> for <caso> {";" <caso>}
           [senão <SeqüênciaComandos>] fim-caso
<caso> = <ListaRótulos> ":" <SeqüênciaComandos>
<SeqüênciaComandos> = <comando> {";" <comando>}
<comando> = [<mensagem> | <atribuição> | <condição> |
           <escolha>]
<atribuição> = <variável> ":=" <expressão>
<expressão> = <fator> {[<relação> <fator>]}
<fator> = <ExpressãoSimples> | "(" <expressão> ")
<ExpressãoSimples> = [não] <valor>
<valor> = <constante> | <variável>
<constante> = verdadeiro | falso
```

<relação> = "=" | "<>" | e | ou
<variável> = <letra> | <letra> <numero> | <letra> <variavel>
<mensagem> = mensagem "(" <SeqüênciaSaida> ")"
<SeqüênciaSaida> = SímboloSaída {SímboloSaída}
<ListaRótulos> = <rótulo> {"," <rótulo>}
<rótulo> = SímboloEntrada

Se observarmos esta definição, a linguagem fica amarrada às variáveis, que tem o papel de realizar a comunicação entre a aplicação e o processamento do diálogo, e aos símbolos de entrada e saída, que podem ser escolhidos de maneira deixar livre a escolha da aparência do processo de comunicação.

Fica evidente que esta especificação tem nível de abstração mais baixo que a representada no D.E.T., como a descrita no capítulo anterior, principalmente pelo fato de normalmente considerarmos o estabelecimento das mensagens emitidas por uma aplicação como um detalhe de implementação. Contudo, já que consideramos neste trabalho que o estabelecimento destas mensagens facilita a validação da implementação, e como a especificação detalhada do diálogo nos fornece um modelo mais completo da aplicação, considera-se estas especificações parte do modelo conceitual do diálogo.

A especificação detalhada do diálogo pode não ser feita, naqueles caso em que o modelador não a julgue necessária. Foi o que aconteceu na modelagem do BIBLOS.

4.3. Modelagem da dinâmica

Embora tenhamos eliminado as informações sobre o comportamento das funções na modelagem do diálogo, verificamos que é necessário o estabelecimento das características comportamentais de um sistema em algum momento.

Portanto, a associação das informações sobre o controle das funções ao diálogo deve ser feito, e este é um momento bem oportuno para colocar esta associação, já que definimos completamente o modelo conceitual do diálogo.

Sugere-se que este processo de funcionalização das representações do diálogo faça-se inicialmente através do estabelecimento de fronteiras que englobem vários estados na representação do diálogo, criando estados compostos no diagrama, como as construções sugeridas em [Harel88].

Observe que existe uma diferença fundamental entre o processo de criação de estados compostos proposto aqui e em outros textos, já que:

a. Não é intensão do modelador, ao realizar esta tarefa, em criar uma representação hierárquica da aplicação, já que esta é feita na modelagem das atividades do sistema;
b. A associação da característica funcional do sistema ao diálogo é feita após este último ter sido especificado, em parte ou completamente.

Estas fronteiras seriam traçadas segundo um processo bottom-up, de agrupamento de vários estados em um estado composto, seguindo as seguintes diretrizes:

a. Identificar o conjunto de estados que processam o diálogo de uma função;
b. Identificar conjuntos de estados que tenham transições incidindo apenas em um destes estados, com um número de transições entre estes maior que com o resto dos estados do diagrama.

A primeira diretriz deve ser considerada a partir das funções identificadas na análise funcional, em seu nível mais baixo, já que fica difícil mapear funções de nível mais alto inicialmente. A segunda é uma analogia aos conceitos de coesão e acoplamento, agora no contexto do diálogo.

Como exemplo, observe os estados 8 e 9 na figura 9, no modelo do diálogo do sistema BIBLOS. Neste conjunto de dois estados, existe apenas uma transição incidente, que é aquela ativada a partir do estado 3, pela ocorrência da condição CHAVE (no caso desta ser inexistente), levando ao estado 8. Observamos que existem 4 transições conectadas a estes estados e incidentes nos mesmos, e 4 transições que levam a estados fora deste conjunto.

Por outro lado, existe a função F2.2, colocada na figura 6, que trata da inclusão de objetos no banco de dados, e os estados mencionados acima estão exatamente na parte da especificação do diálogo relacionada a atualização, pois o usuário necessita escolher 'A' a partir do estado inicial para atingir o estado 4. Além disso, a chave do objeto selecionado não existe, já que para atingirmos o estado 8 esta condição deve ser cumprida, o que caracteriza a intenção de incluir um novo objeto com esta chave.

Visto que estes estados estão relacionados ao processamento do diálogo da função F2.2, podemos traçar a fronteira de um estado composto, que irá incluí-los. A segunda diretriz para a criação de estados compostos nos mostrará sempre, além da fronteira, o estado inicial do estado composto, que será sempre o estado onde incide a transição no conjunto. Este processo foi realizado para o

sistema BIBLOS e aparece na figura 23, e para o sistema DIAGRAMA na figura 24.

Algumas dificuldades podem aparecer nesta criação de estados compostos. A primeira ocorre quando o diálogo de uma função não está especificado (ou mesmo não existe). Neste caso não é criado nenhum estado composto.

A segunda problema aparece quando tentamos identificar o diálogo em uma função e este já foi associado a outra, fato constatado pela existência do respectivo estado composto. Isto indica que está havendo reuso de diálogo, ou seja, duas ou mais funções possuem o mesmo diálogo, e neste caso, nenhum outro estado composto será criado. Foi o que aconteceu no momento em que foram identificadas no diagrama as funções de consulta, F1.2 a F1.5, mostradas na figura 23 associadas a uma transição saindo do estado 4.

Como proposto em [Harel88], não existe a necessidade de criar estados finais para cada estado composto, evitando o excesso de complexidade encontrado nos diagramas de [Quent88].

Este tipo de formalismo visual usado, que na verdade são statecharts, permite que transições sejam representadas partindo de estados compostos, que são disparadas quando o

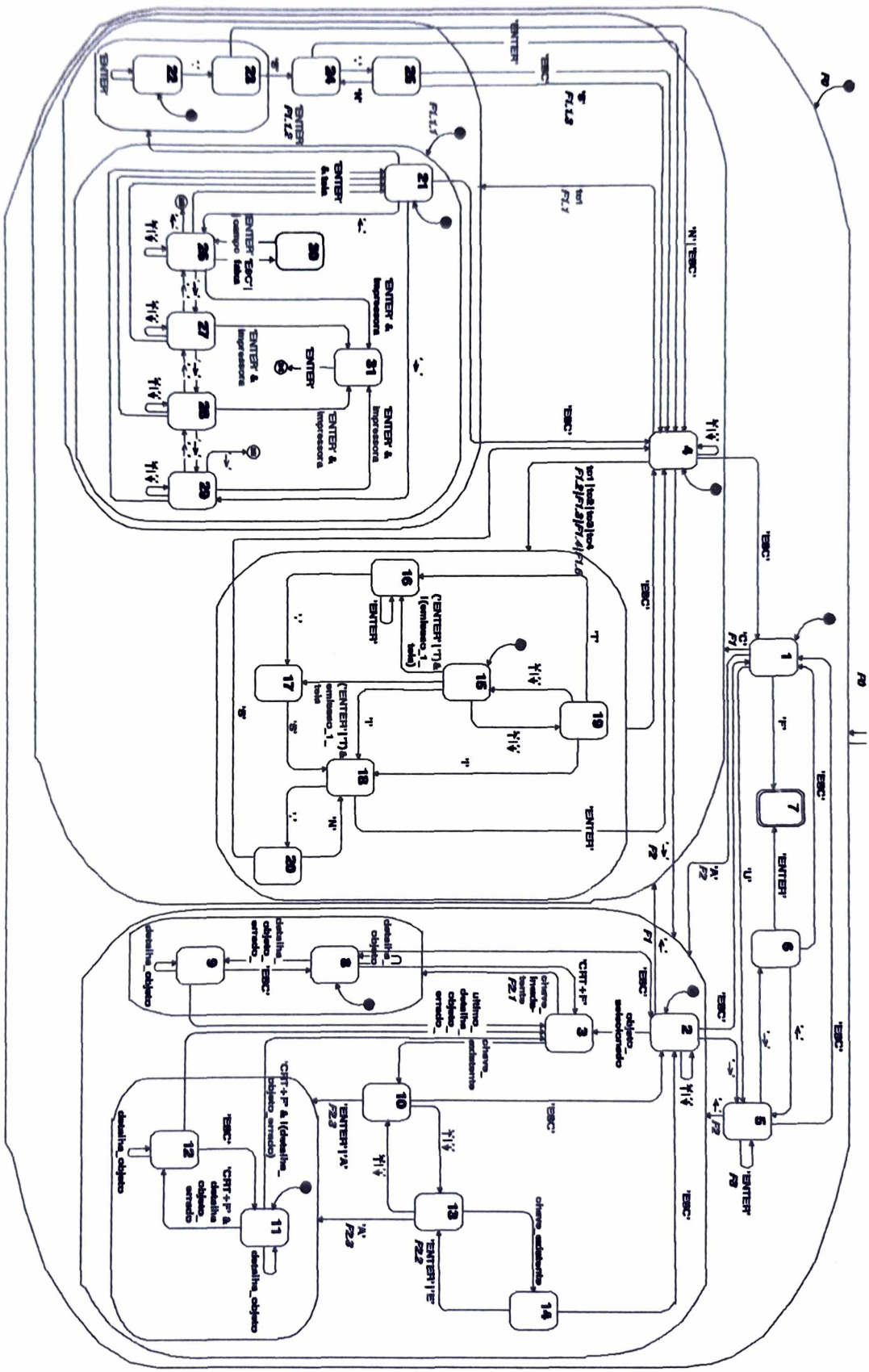


FIGURA 23-Modelagem da Dinamica do Sistema BIBLOS

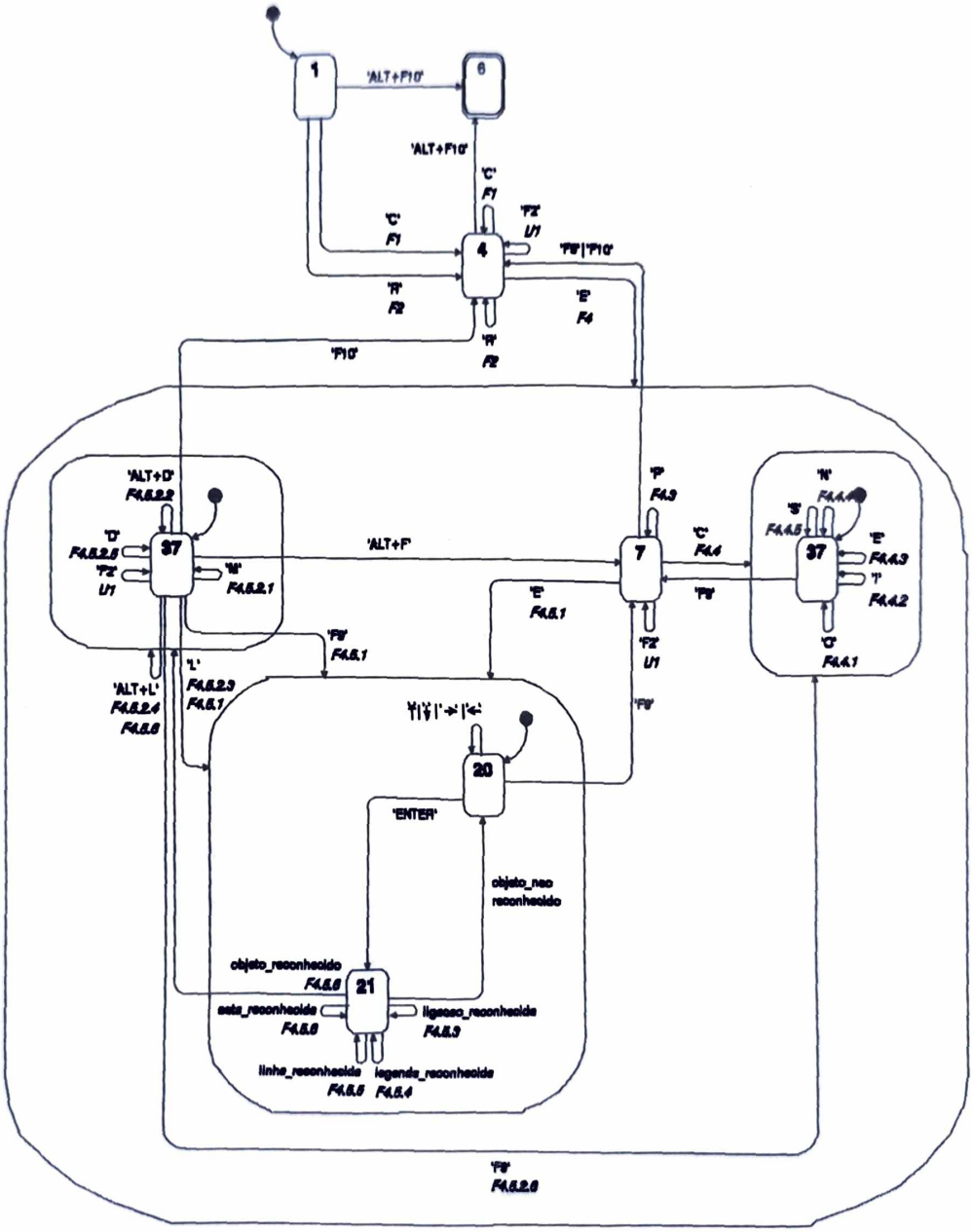


FIGURA 24. Modelagem da Dinamica do Sistema DIAGRAMA

símbolo de entrada associado é detectado em qualquer um dos estados inclusos no interior do estado composto. Este aspecto aparece modelado no sistema BIBLOS, através da supressão de todas as transições que levavam ao estado inicial, partindo de cada estado do diagrama, ao ser pressionado 'ALT+ESC', e criando-se um estado composto com todos os estados do diagrama, com uma transição partindo deste ao ser pressionado 'ALT+ESC'.

A característica acima, associada à diminuição da distância entre os objetos [Prots91], devido a criação de estados compostos, leva a uma simplificação considerável no diagrama.

Como o aparecimento de estados compostos no diagrama está associado apenas à identificação do diálogo que é processado nas funções do sistema, não faz sentido especificar uma descrição detalhada do diálogo para um estado composto.

O próximo passo nesta associação das funções ao modelo do diálogo é a explicitação da ativação das funções. Até o momento identificamos apenas o diálogo processado por cada função, mas não podemos estabelecer o comportamento temporal destas.

Para tal, associamos funções a transições, indicando a ativação destas no momento de ocorrência da transição. A desativação destas funções pode ocorrer quando se encerra a transição, no caso em que o estado de destino não detalha o diálogo desta função, ou no momento em que o controle é levado para fora do estado composto, quando este estado composto detalha o diálogo desta função.

Podem ser associados uma ou mais funções a cada transição. No segundo caso, pode-se desejar representar duas situações: uma na qual as várias funções são executadas em seqüência, e outra, quando somente uma será processada, mas não se deseja especificar qual.

Convencionou-se representar a associação de funções em seqüência, para aquelas que são disparadas de forma seqüencial a partir da ocorrência de uma transição, ou separadas por ¦, quando não se deseja especificar qual será processada, entre um conjunto de funções com execução mutuamente exclusiva.

A partir daí pode-se pensar na especificação do processamento de diálogo assíncrono ou de funções concorrentes, mas isto foge ao escopo do trabalho. Entretanto são dadas indicações para tal no último capítulo do trabalho.

Exemplos de associação de funções a transições podem ser encontrados nas figuras 23 e 24.

Dando este enfoque à ativação de funções, podemos dizer que a dinâmica de um sistema estará bem definida, pois a modelagem das características funcionais estáticas está bem definida, na especificação das atividades do sistema, não há restrições quanto a tecnologia interna, já que não mencionamos a duração de uma transição ou do processamento de uma função, e o comportamento dinâmico está totalmente modelado pela representação da ativação e desativação de funções no diagrama de estados e transições.

Podemos justificar a necessidade da construção descrita nesta seção não só por representar a dinâmica de forma bem definida ou por simplificar o D.E.T., mas também por criar um modelo intermediário e rico em informações para geração automática de código, levando portanto a uma implementação de maneira mais fácil.

4.4. A estrutura do método proposto

Na figura 25 aparece a estrutura proposta pelo método para especificação de sistemas. Ali destaca-se a divisão das

<table>
<tr><th colspan="3">Modelo</th><th colspan="2">Representacao</th></tr>
<tr><td rowspan="5">Modelo
de Essencia</td><td colspan="2">Modelo
de Contexto</td><td colspan="2">Definicao do Sistema: Informal
Esquema de Contexto: DFD, DER
Lista de Eventos Externos: Texto</td></tr>
<tr><td rowspan="4">Modelo
de Comportamento</td><td></td><td>Secao
Esquematica</td><td>Secao
Detalhada</td></tr>
<tr><td>Modelo
de Atividades</td><td>D.F.D.</td><td>Portuguee
Estruturado
ou equivalente</td></tr>
<tr><td>Modelo
de Informacoes</td><td>D.E.R.</td><td>Dicionario
de Dados</td></tr>
<tr><td>Modelo
do Dialogo</td><td>D.E.T.</td><td>(escopo, prompt,
resposta, feedback)</td></tr>
<tr><td colspan="2">Modelo de Implementacao</td><td>Modelo
da Dinamica</td><td>StateChart
com funcoes</td><td>(escopo, prompt,
resposta, feedback)</td></tr>
<tr><td colspan="3">Modelo de Implementacao</td><td colspan="2">Diagrama
Hierarquico
de Modulos</td></tr>
<tr><td colspan="3">Modelo de Automacao</td><td colspan="2">Linguagem
escolhida
p/ implementacao</td></tr>
</table>

FIGURA 25-A estrutura do metodo proposto

especificações do modelo de comportamento em seção esquemática e seção detalhada.

Como podemos ver, o modelo proposto para especificação do diálogo possui estas duas seções, como mostrado neste capítulo e no anterior. Esta divisão serve para isolar os detalhes de menor nível de abstração daqueles que contém informações mais relevantes no sistema modelado.

Constata-se também que o método de desenvolvimento proposto aqui auxilia a construção de sistemas automatizados usando um processo gradativo e usando linguagens próprias em cada fase. Isto se evidencia pela inclusão do modelo do diálogo e do modelo da dinâmica neste processo de desenvolvimento.

A princípio não existe uma ordem pré-definida para a elaboração dos modelos da especificação do comportamento do sistema. Pode-se iniciar ou por atividades, ou por informações ou pelo diálogo indistintamente, sendo sugerido que o crie-se primeiro a modelagem estática do sistema, ou seja, se comece o trabalho por informações.

Uma característica constatada é que, se forem providas facilidades automáticas para o desenvolvimento e prototipagem rápida, o ciclo de vida estrela proposto em

[Hart89a] será suportado por este método. Este ciclo de vida enfatiza a alternação constante entre as várias visões do sistema modelado, ora por suas características funcionais, ora pelo diálogo, centrando o projeto na característica de avaliação constante.

Como será mostrado adiante, existem critérios de avaliação das especificações do diálogo produzidas aqui, passíveis de automatização, mostrando que estas especificações se adequam perfeitamente nas caraterísticas ideais postas naquele texto.

Podem ser listadas várias vantagens no método proposto, entre as quais:

1. Diminuir a distância entre a especificação e a implementação, tornando este processo mais gradativo, já que são elaborados mais modelos intermediários;
2. As especificações criadas são mais completas, por abordarem mais aspectos que o comumente feito na especificação de sistemas;
3. Proporciona múltiplas visões do sistema modelado, enfatizando a validação e a corretude.

Este método possui uma forte tendência a facilitar a automação do processo de desenvolvimento, pois: (i) existem estratégias de transformação naturais entre modelos (que serão mostradas a seguir), (ii) existem mais modelos que em outros métodos, (iii) pode ser realizada avaliação automática, e (iv) pode ser gerado código inicial para a aplicação de forma automática. As duas últimas características serão tratadas no próximo capítulo.

4.5. Estratégias de transformação

Visto que no processo de elaboração de modelos colocado no método proposto, muitas informações se preservam de uma para outra representação, mudando a forma ou o nível de detalhe em que aparecem, podem ser definidas estratégias de transformação entre modelos.

Estratégias de transformação nos ajudam a criar um novo modelo a partir das informações existentes em algum outro, elaborado anteriormente da tarefa de especificação do sistema. Estas são chamadas estratégias pois criam apenas uma representação inicial, que deve ser melhorada, por acréscimo de detalhes. Por esta razão não são chamadas algoritmos.

Em geral, estratégias baseia-se nas informações presentes na especificação das funções do sistema para a derivação de um diagrama hierárquico de módulos inicial. São a análise de transformação e a análise de transação [Page88].

Análise de transformação baseia-se na identificação da atividade, ou processo, de transformação de dados chamado centro de transformação, que realize pouco processamento e muito processamento de atividades no diagrama de nível mais alto do conjunto de D.F.D.s do sistema. Pendura-se todo o diagrama por esta bolha, conservando fluxos e criando uma estrutura hierárquica, onde cada processo será transformado em um módulo.

Análise de transação considera que o D.F.D. já está nivelado e balanceado, dividido de acordo com o processamento de transações do sistema, que são os conjuntos de dados processados por um conjunto único de atividades. Cada atividade que identifica um conjunto de dados e o remete a outra será considerado uma transação. A estratégia coloca que todas as atividade que processam transações sejam transformadas em módulos, conectadas seguindo a hierarquia de níveis do D.F.D..

Estes enfoques necessitam que sejam criados um conjunto de D.F.D.s não essenciais para que a transformação

seja feita com algum proveito [Keuff91b]. Entretanto, consideramos que o número de modelos elaborados se torna muito grande desta forma, sugerindo sempre que possível o uso de transformação centrada no diálogo.

Esta estratégia de transformação consiste em aproveitar a associação das funções à representação do diálogo e a ativação destas para criar um diagrama hierárquico de módulos.

Segue-se os seguintes passos:

1. Para cada estado composto da representação do diálogo cria-se um módulo. Se não existe um estado englobando todos os estados do diagrama, deve-se criar também um módulo como se este existisse;
2. Cada módulo criado recebe um nome, associado à função, ou ao conjunto de funções, que tem aqueles estados especificando o diálogo associado a esta(s);
3. Módulos de nível mais baixo são associados a seu módulo-pai, derivado a partir de um estado composto que contém o estado que deu origem ao módulo em questão;

4. Para cada módulo identificado é criado um módulo-filho, que se encarregará de conter a implementação do processamento do diálogo da função que gerou o módulo. Este módulo será implementado fisicamente dentro do módulo pai, para que variáveis que ligam o controle do diálogo à implementação sejam compartilhadas (aquelas que aparecem no feedback). Enviará uma ligação de controle chamada "função" ao módulo pai, contendo a informação sobre qual função será ativada em seguida (se existir).

As funções que não possuem diálogo ou não tem diálogo especificado não terão módulos contendo a implementação derivados por esta estratégia de transformação. Isto se deve ao fato de toda estratégia ter um ponto arbitrário de escolha sobre quais informações aproveitar do modelo anterior, e este foi o escolhido em nossa estratégia. Na figura 26 é mostrada a derivação do D.H.M. do sistemas BIBLOS.

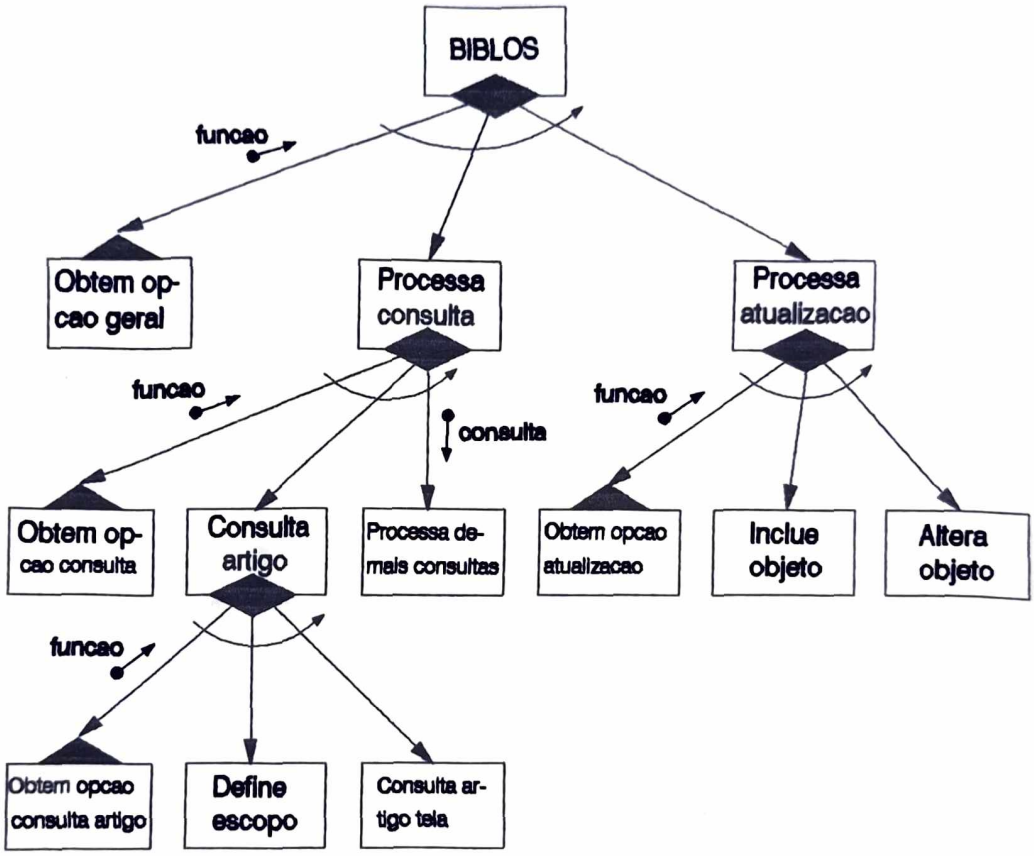


FIGURA 26-Diagrama Hierarquico de Modulos do Sistema BIBLOS

Capítulo 5

Automatizando o método

Neste capítulo será dada atenção a técnicas que possam ser aplicadas às representações criadas usando o método proposto, no intuito de melhorá-las, reorganizá-las ou transformá-las.

Tem-se interesse particular nestas técnicas, já que são, por natureza, fáceis de interpretar através de programas, escritos em uma linguagem qualquer, além de poderem ser realizadas manualmente.

5.1 Avaliação das especifi-cações do diálogo

Como colocado no capítulo anterior, o processo de especificação do seqüenciamento do diálogo e de interfaces com o usuário deve estar centrado em avaliação constante [Harts89a].

Naquele capítulo falamos ainda que a especificação do diálogo deve estar fundamentada nas mensagens trocadas no sistema a ser automatizado, chamado diálogo essencial. Ao realizarmos um processo de avaliação, verificamos quando deve ser incluído diálogo não-essencial na especificação. Esta avaliação é feita durante a especificação dos D.E.T.s, onde estão representadas as respostas do usuário, objetivando a

melhoria da qualidade na implementação do processo de diálogo.

Um primeiro critério está baseado na capacidade cognitiva de interpretação e decisão do usuário. Como citado comumente na literatura, pessoas ao se depararem com alguma atividade de decisão ou interpretação agrupam opções de escolha ou objetos em conjunto de 7, mais ou menos 2, dependendo da pessoa, visando diminuir a complexidade do processo, e a partir daí realizam a atividade desejada para cada conjunto criado mentalmente. Posteriormente os resultados encontrados para cada conjunto são agrupados e é formulada uma conclusão.

Portanto devemos evitar ao máximo a realização de agrupamento, induzindo o usuário a fazer interpretações e tomar decisões de forma imediata. Mapeando estes conceitos em nosso modelo da interação usuário-aplicação, temos:

Definição: 1º Critério de avaliação

"Em um estado, deve-se ter entre 5 e 9 transições válidas partindo deste estado".

No Modelo do diálogo de um sistema, ao olharmos para o D.E.T., devemos contar diretamente as transições partindo

de cada estado. Já no modelo da dinâmica, além de verificar as transições partindo de cada estado, devemos considerar também aquelas que partem dos estados compostos que contém o estado que está sendo analisado.

Podemos estabelecer ainda um segundo critério de avaliação, se considerarmos que o usuário está sempre orientado para a satisfação de seus objetivos [Gaine86].

Estes objetivos normalmente são o processamento de alguma atividade de transformação de dados, a obtenção de ajuda ao utilizar o sistema ou o término do processamento.

Definição: 2º Critério de avaliação

"O número de transições que devem ser disparadas a partir de um estado para se atingir um estado onde o usuário realiza os objetivos não deve ser excessivo. Tais objetivos podem ser:

1. Realizar alguma atividade de transformação de dados;
2. Realizar outra atividade de transformação de dados, após uma ter sido concluída;
3. Obter ajuda na utilização da aplicação;
4. Encerrar o processamento."

Nos diagramas de estados e transições podemos visualizar a avaliação por este critério como a contagem do número de transições (que são arestas do grafo) que ligam um estado a outro.

Observe que esta definição deixa em aberto o número de transições consideradas excessivas. Ao realizar a especificação do sistema, este parâmetro fica a cargo do modelador, mas para a implementação de procedimentos de avaliação automática, este critério pode ser analisado de forma relativa, ou seja, comparando o número de interações realizadas objetivando a ativação de uma e outra função, por exemplo.

Fazendo uma análise destes dois critérios, constatamos que um está ligado a estados e o outro a transições, um é absoluto e o outro é relativo, um avalia de forma qualitativa e outro de forma quantitativa a especificação do diálogo com o usuário. Isto sugere que estes nos dêem uma forma de avaliação completa do seqüenciamento do diálogo.

Poderiam ser colocados critérios de avaliação para a aparência e estilo da interação, e também para a avaliação do diálogo em tempo de execução, mas estes não são os objetivos neste texto.

5.2. Aplicação de algoritmos de M.E.F.s às representações do diálogo

Algoritmos para manipulação de máquinas de estados finitos são amplamente difundidos no contexto de Construção de Compiladores e Reconhecimento de Linguagens [Barre79].

Como utilizamos este tipo de formalismo aqui, podemos conjecturar se sua utilização é de alguma forma valiosa para a manipulação de especificações do diálogo.

Um primeiro algoritmo é capaz de encontrar para cada M.E.F. uma máquina minimal, ou seja, que possuí o menor número de estados entre as máquinas que reconhecem a mesma linguagem. Entretanto não poderá ser aplicado às especificações do diálogo.

Isto se justifica pelo fato de algoritmos aplicados à M.E.F.s convencionais não levarem em conta a dependência de cada estado ao contexto. No reconhecimento de linguagens, se existem duas seqüências onde os estados de cada uma são ligados por transições cujas condições são idênticas, a linguagem reconhecida pelas duas seqüências é a mesma, não havendo a necessidade das duas estarem presentes.

Na especificação do diálogo, cada estado está associado a uma situação no seqüenciamento do diálogo, com escopo, prompt, resposta e feedback próprios, amarrados a uma situação contextual específica, e provavelmente diferentes de qualquer outro. Portanto estados não podem ser suprimidos desta especificação.

Um segundo algoritmo trata da supressão de estados onde acontecem não-determinismos. Este trabalha realizando a junção de todos os estados onde incidem aquelas transições partindo do mesmo estado, disparadas por condições de ocorrência idênticas. As transições que incidiriam em estados diferentes serão substituídas por uma única, incidente no estado criado pela junção.

Como dito anteriormente a representação de não determinismos não é desejável em nossos modelos, por não especificarem completamente e não representarem de maneira clara a seqüenciação do diálogo. Portanto aconselha-se que estas construções sejam feitas em tempo de especificação, acrescentando anotações sobre condições dependentes da semântica da aplicação. Mesmo assim, se ocorrerem, não poderá ser aplicado o algoritmo para a remoção destes, porque o algoritmo também retira estados da especificação.

Numa máquina de estado finitos podem estar especificados estados que não podem ser alcançados a partir de outros simplesmente pela ocorrência de transições. Estes estados são chamados inatingíveis.

Existe um algoritmo de detecção e remoção destes estados. O passo de detecção pode ser aplicado, já que um estado inatingível especifica um ponto da interação que não pode ser alcançado, e portanto inútil como especificação. Entretanto, como dito anteriormente o passo de remoção pode acabar por suprimir um trecho do diálogo necessário, mas que por um descuido do modelador, foi especificado incorretamente.

Pode-se argumentar que critérios de avaliação e algoritmos aplicáveis a máquinas de estados finitos tenham o *mesmo objetivo*, que é a reestruturação das especificações, se distinguindo apenas pelo nível de formalização das definições.

Entretanto, critérios se preocupam em garantir a validação da aplicação em desenvolvimento, porque medem, sob o aspecto do diálogo, o quanto uma aplicação é fácil de usar.

Os algoritmos apresentados aqui se preocupam em preservar a corretude das especificações, pois tentam transformá-las sem que haja perda de informações relevantes.

5.3. Geração automática de código

Como falado anteriormente, com as especificações da dinâmica do sistema feitas da forma como sugerido, é possível realizar a geração automática do código de boa parte do controle da aplicação e de todo o diálogo com o usuário.

Para tal, levamos em consideração os módulos criados a partir da estratégia de transformação colocada no capítulo anterior, gerando um código modular. Neste trabalho realizaremos a geração de código para a linguagem TURBO PASCAL 5.0, por ser modular e estruturada, mas poderíamos mudar a linguagem alvo para qualquer uma que possuísse estas características.

Esta geração de código considera que sempre haverá no corpo do programa principal a declaração de uma variável global chamada ESTADO. Esta será a única informação compartilhada necessária para o processamento de todas as funções do sistema.

Colocamos a seguir um exemplo de geração de código, para o módulo específico de inclusão de objetos, mostrado na figura 26. Este módulo possui dinâmica detalhada pelos estados 8 e 9 que aparecem na figura 22.

```
unit InclueObjeto;

interface

procedure ProcessaInclusão;

implementation

function Dialogo:byte;
const EstadosPertencentes: set of byte = [8,9];
var FUNCAO:byte;
    RESPOSTA:char;
    FIM:boolean;
begin
    FUNCAO := 0;
    FIM := false;

    while (not(FIM) and (FUNCAO = 0)) do begin

        RESPOSTA := readkey;
        case ESTADO of
           8:if (detalhe_objeto) then
                ESTADO := 8
             else
                if (detalhe_objeto_errado) then
                    ESTADO := 9
                else
                    case RESPOSTA of
                        'AltEsc':ESTADO := 1;
                        'Esc'   :ESTADO := 2;
                        'CrtF'  :ESTADO := 3;
                    end;
```

```
            9:if (ultimo_detalha_objeto_errado) then
                  ESTADO := 3
              else
                 if (detalha_objeto) then
                     ESTADO := 9
                 else
                    case RESPOSTA of
                       'AltEsc':ESTADO := 1;
                       'Esc'   :ESTADO := 8;
                    end;
        end;
    end;

    if ((FIM) or not(ESTADO in EstadosPertencentes)) then
        FUNCAO := -1;
    Dialogo := FUNCAO;
end;

procedure ProcessaInclusão;
var F:byte;
begin

    F := 19;
    while (not(ESTADO in EstadosFinais) and (F > 0)) do
  begin
        F := Dialogo(ESTADO);

        (* Ativação ou definição das funções processadas *)

    end
end;

end.
```

Inicialmente deve ser criado para cada módulo do D.H.M. uma unit, contendo a implementação de todo o diálogo e do controle das funções processadas pelo módulo, com nome idêntico ao especificado no D.H.M. As definições das interfaces de todos os procedimentos incluídos neste módulo deverão ser copiadas logo após a declaração interface.

Para cada módulo gerado é criada uma função para realizar o processamento do diálogo de todas as funções

implementadas neste módulo. Estas funções tiveram o diálogo em comum identificado através de uma fronteira traçada no modelo da dinâmica do sistema, gerando um estado composto.

Em todos os módulos esta função de processamento do diálogo terá o mesmo nome, chamando-se DIÁLOGO, podendo ser vista no código acima implementada na seção implementation, por ser local ao módulo.

DIÁLOGO corresponderá a implementação dos módulos que obtém as opções de processamento do usuário, mostrados no D.H.M., e especificamente no BIBLOS mostrados na figura 26. Como explicado no capítulo anterior, no caso do módulo InclueObjeto ester módulo não aparece no diagrama hierárquico de módulos por ter sido o único módulo determinado no modelo da dinâmica que estaria pendurado em InclueObjeto.

No exemplo acima, a única função processada pelo módulo é exatamente a que processa a inclusão de objetos. O nome da função é copiado da especificação no modelo de atividades do sistema, associada a esta função.

A estrutura interna da função DIÁLOGO conterá um laço de repetição, onde é testada a condição FIM, que assumirá valor true quando o processamento de controle neste módulo, e conseqüentemente nos mais abaixo, se encerrar.

Na condição deste laço será verificado também se nenhuma função que tem implementação num nível abaixo deste foi ativada. Isto estará sempre associado a atribuição de um valor maior que zero a variável FUNÇÃO. Conseqüentemente, a cada função estará associado um número, podendo ser a posição na ordenação hierárquica das funções feita na modelagem das atividades do sistema.

Na geração de código colocada acima, este teste sobre as variáveis FIM e FUNCAO pode ser visto associado ao while mostrado abaixo da inicialização das variáveis locais no módulo. No conteúdo deste deverão ser processadas as respostas do usuário e as transições entre estados.

Dentro da estrutura de repetição deverão ser lidas as respostas do usuário, e realizado o disparo das transições válidas a partir de cada estado. Isto é feito pela inclusão no código executável das declarações RESPOSTA := readkey; e case ESTADO of ... end;, como mostrado no código gerado para o módulo InclueObjeto.

Cada rótulo do case deverá corresponder a um estado pertencente ao estado composto que deu origem a este módulo. No exemplo do módulo InclueObjeto, os únicos estados que processam o diálogo do módulo são o 8 e o 9, como mostrado no

diagrama da figura 22, e por isto só existem estes rótulos na estrutura de seleção.

Para que seja realizado o processamento das transições, as condições de disparo de todas as transições de um mesmo estado devem ser agrupadas, e divididas em dois conjuntos: S, com as condições com anotações dependentes da semântica da aplicação, e R com as respostas normais. Para o estado 8 do exemplo do módulo InclueObjeto, estes conjuntos seriam S = {detalhe_objeto,detalhe_objeto_errado} e R = {´AltEsc´, ´Esc´, ´CrtF´}.

Associado a cada rótulo do case explicado acima, deverá ser colocada uma estrutura encadeada de if then else, onde em cada if deverá ser testada uma condição dependente da semântica da aplicação, que foram agrupadas no conjunto S, como explicado no parágrafo anterior.

No último else, quando não houverem mais condições dependentes da semântica a serem associadas a ifs, deverá ser criada uma estrutura de escolha, testando o valor da RESPOSTA do usuário. Cada rótulo deste case estará associado a uma resposta do usuário contida no conjunto R criado anteriormente.

No módulo listado acima, para o estado 8 é criada uma estrutura de if s_1 then P_{s1} else {if s_1 then P_{s1} else}, seguida por case RESPOSTA of r_1: P_{R1}; {r_1: P_{R1};} end;, contendo partes P_{s1} ou P_{r1}, onde é realizado o processamento das transições.

Neste processamento das transições deve ser alterado o valor da variável ESTADO, substituindo seu conteúdo atual pelo número do estado de destino da transição disparada por uma condição. No caso do estado 8, quando é pressionado ´Esc´ o diálogo é levado para o estado 2, o que está representado no código fonte listado pela declaração ESTADO := 2 inserida dentro da estrutura de escolha, sendo disparada pela ocorrência do rótulo ´Esc´ como valor da variável RESPOSTA.

A criação de código descrita acima deve ser feita para cada estado presente no módulo em questão, sendo encerrada pela declaração do end que encerra o laço de repetição while declarado no início da função.

Após o término do processamento do diálogo, a variável FUNÇÃO deverá conter o número associado a uma função que será disparada, com o respectivo código gerado junto com a alteração da variável ESTADO. No módulo InclueObjeto este caso não ocorreu.

Também associada a mudança do valor da variável ESTADO deve estar a alteração do conteúdo da variável FIM, quando o seqüenciamento do diálogo for levado a um estado final. Por exemplo, se o estado 8 fosse um estado final, associado ao rótulo do case onde é realizado o teste sobre o valor da variável ESTADO, deveria ser declarado FIM := true;. Desta forma, o trecho de código descrito acima ficaria assim:

```
RESPOSTA := readkey;
case ESTADO of
   8:FIM := true;
   9:if (ultimo_detalha_objeto_errado) then
         ESTADO := 3
     else
        if (detalha_objeto) then
            ESTADO := 9
        else
           case RESPOSTA of
              'AltEsc':ESTADO := 1;
              'Esc'   :ESTADO := 8;
           end;
end;
```

Obviamente não poderiam haver transições partindo de 8, já que este seria neste caso um estado final.

Para cada módulo deveria ser criado um conjunto constante, de nome EstadosPertencentes, contendo o número de todos os estados condidos no estado composto que deu origem a este módulo. Se algum estado que não estivesse incluso em EstadosPermanetes fosse atingido, deveria ser encerrado o processamento do diálogo neste estado.

Para que o controle voltasse à função chamadora da que está tendo o diálogo processado em determinado instante, um valor de retorno deveria ser passando pela função DIÁLOGO à função em processamento, sinalizando o fim do processamento. Isto deveria ser feito atribuindo à variável FUNÇÃO o valor -1, e retornando no nome da função DIÁLOGO o valor da variável FUNÇÃO.

Este conhecimento aparece representado no trecho de código no final da função DIALOGO do módulo InclueObjeto. Caso fosse atingido um estado final, a variável FIM conteria um valor verdadeiro, e caso o controle fosse levando para um estado fora da função, o conteúdo de ESTADO não estaria em EstadosPermanetes.

O código gerado para cada função corresponderia apenas à ativação e desativação das sub-funções. Este controle seria realizado apenas se o estado atual não fosse um estado final, e estaria refletido numa estrutura de repetição **while**, onde seria testado se o valor da variável F é maior que 0. F seria inicializada com o valor associado a função na Modelagem de Atividades.

Observe o exemplo criado para a função ProcessaInclusão no módulo InclueObjeto. Como não é ativada a partir desta nenhuma função identificada na modelagem das

atividades do BIBLOS, o código gerado se resumiria ao controle sobre quando encerrar o diálogo. F receberia o valor 19 (F2.1 é a décima noma função listada na figura 6) no início da função, para que o controle não fosse desviado de volta à função chamadora.

O exemplo do módulo InclueObjeto é um tanto trivial. Existem casos de geração de código que não puderam ser observados na listagem colocada.

Um deles é a ocorrência de chamada a outras funções a partir do módulo. Neste caso, duas providências deveriam ser tomadas. Uma seria declarar explicitamente

uses <NomeModulo1>, ..., <NomeModulon>;

após a declaração interface, para cada módulo que possui funções chamadas a partir deste. Outra seria estabelecer uma estrutura case, dentro do corpo da função chamadora, verificando o valor de F, e de acordo com este, desviando o controle para outra função.

Suponha que existisse e tivesse sido modelado o diálogo nas funções F2.1.1, F2.1.2, ..., F1.2.6, declaradas na figura 6. Neste caso, o código gerado para a função ProcessaInclusão deveria ser

```
procedure ProcessaInclusão;
var F:byte;
begin

    F := 19;
    while (not(ESTADO in EstadosFinais) and (F > 0)) do
  begin
        F := Dialogo(ESTADO);

        case F of
            20:InclueArtigo;
            21:IncluePeriodico;
            22:InclueEntidade;
            23:InclueAssunto;
            24:InclueReferência;
            25:InclueEdicaoDePeriodico;
        end;
    end
end;
```

Cada uma destas estaria associada ao rótulo contendo um número, que seria a posição de cada uma destas funções na ordenação hierárquica das funções encontradas na modelagem das atividades do sistema.

Outro caso que não foi modelado no exemplo citado acima é a seção detalhada do diálogo especificada através de prompt, escopo, resposta e feedback. Note que este, em teoria deveriam ser modelados, pois apareceram condições dependentes da semântica da aplicação (detalhe_objeto, detalhe_objeto_errado e ultimo_detalha_objeto_errado), devido às estas anotações ao diagrama, que deveriam ter a seqüência de valores ao longo do processamento da função explicitado no feedback dos estados 8 e 9.

Se fosse feita a opção por especificar esta estrutura, inicialmente deveriam ter substituídas todas as construções usando pseudocódigo para PASCAL, e criadas variáveis na seção de implementação do módulo para cada condição dependente da semântica da aplicação.

Além disso, deveriam ser declarados os seguintes comandos, em substituição a RESPOSTA := readkey;:

```
case ESTADO of

    (* Rótulos dos estados que tem prompt ou escopo
       especificados, e emissão das respectivas
       mensagens *)

end;
case ESTADO of

    (* Rótulos dos estados que tem resposta e feedback
       especificados, leitura da resposta e inserção da
       estrutura equivalente à declarada no feedback *)

else
    RESPOSTA := readkey;
end;
```

Apareceriam como rótulos no primeiro case os números dos estados que possuíssem prompt ou escopo especificados. Se apenas um destes fosse especificado, simplesmente a mensagem associada seria emitida. Caso os dois aparecessem, haveria a necessidade de declarar a impressão das mensagens entre begin e end, como mostrado abaixo:

```
case ESTADO of

    i:(* esta mensagem foi especificada no escopo *)
```

```
        writeln('CAPíTULO 4');

      j:begin
          (* estas mensagens foram especificadas no
             escopo e no prompt *)
          writeln('CAPíTULO 5');
          writeln('Aguardando término capítulo');
        end;
  end;
```

A leitura da resposta do usuário e a determinação do feedback estariam incluídas dentro do segundo case especificado. Para resposta deveria ser criada uma variável, de acordo com o tipo da mensagem lida, e incluídas a leitura no rótulo correspondente dentro case. Observe que, se não houvesse resposta especificada, seria obtida do teclado a resposta do usuário, sendo colocada na variável RESPOSTA.

O feedback apareceria logo em seguida ao código da leitura da resposta obtida naquele estado, alterando o valor das variáveis dependentes da semântica da aplicação.

A estrutura mais geral que poderia ser criada é mostrada abaixo, nos casos em que todos as situações listadas acima ocorressem, para um módulo do sistema em especificação.

```
unit <NomeDoMódulo>;

interface

uses <NomeModulo1>, ..., <NomeMódulo1>;

procedure <NomeFuncao1>;
procedure <NomeFuncaoJ>;

implementation
```

```
var variável_dependente_da_semanticai,
    variável_dependente_da_semanticaj:boolean

function Diálogo:byte;
const Estados_Pertencentes:set of byte = [...];
var RESPOSTA:char;
    FIM:boolean;
    FUNÇÃO:byte;
    respostai:tipoi;
begin

    FUNÇÃO := 0;
    FIM := false;
    while (not(fim) and (FUNÇÃO = 0)) do begin

        case ESTADO of
            (*  Rótulos  dos  estados  que  tiveram  escopo
                ou prompt especificados *)
        end;

        case ESTADO of
            (*  Rótulos  dos  estados  que  tiveram  resposta
              e feedback especificados *)
        else
            RESPOSTA := readkey;
        end;

        case ESTADO of
            (* Estrutura definida no texto *)
        end;
    end;

    if ((fim) or not(ESTADO in Estados_ Pertencentes) then
        FUNÇÃO := -1;
    Diálogo := FUNÇÃO;
end; (* Função Diálogo *)
```

```
(* Implementação das funcoes do Módulo *)
procedure <NomeFuncaoi>;
var F:byte;
begin

    F := (* Número da funcão *);
    while (not(ESTADO in EstadosFinais) and (F > 0)) do
  begin
        F := Dialogo(ESTADO);

        case F of

         (* Ativação das funções chamadas a partir daqui *)

        end;
    end
end;

end;

end. (* Módulo *)
```

Como dito inicialmente, o programa principal necessitaria conter uma variável global chamada ESTADO. No corpo deste seria declarado num uses os nomes dos módulos que conteriam as funções ativadas diretamente a partir do programa principal. No exemplo do sistema BIBLOS, o programa principal deveria conter

```
uses ProcessaConsulta, ProcessaAtualizacao;
```

O restante do programa principal seria idêntico ao de um módulo qualquer, a exceção de que a função diálogo deveria receber outro nome, DIÁLOGO_PRINCIPAL por exemplo, para evitar confusão de escopo com a função DIÁLOGO implementada em cada módulo.

Capítulo 6

Conclusões e indicações para trabalhos futuros

6.1. Conclusões

Neste trabalho foi apresentado um método para desenvolvimento de sistemas, que enfatiza a especificação do diálogo e da dinâmica em sistemas interativos não concorrentes com diálogo seqüencial.

Constatou-se que, em vários métodos, o diálogo é especificado em uma fase mais próxima da implementação. Estes métodos geralmente não levavam em consideração outras formas de especificação do sistema, como a modelagem de informações ou de atividades.

Paralelamente, verificou-se que a especificação da dinâmica era feita de forma inadequada para os sistemas interativos aos quais nos propusemos estudar, por apresentarem alta complexidade ou baixo nível de abstração.

Foi feito um estudo detalhado do modelo do diálogo usado no sistema BIBLOS e traçado um paralelo com a especificação do sistema DIAGRAMA, do ponto de vista da modelagem do diálogo. Este estudo nos serviu de base para o estabelecimento e a conclusão de várias características desejáveis em um modelo conceitual do diálogo. Foi

evidenciado também que o modelo usando no sistema DIAGRAMA não era adequado, por ser muito complexo.

Estudou-se a forma de especificação detalhada do diálogo, contemplando a modelagem das mensagens trocadas no sentido aplicação-usuário, proposta na modelagem do BIBLOS.

Foi verificado ainda que a dinâmica do sistema estaria bem especificada ao identificarmos estados compostos e ao associarmos a ativação de funções às transições representadas no modelo do diálogo do sistema.

Um método para especificação de sistemas interativos foi colocado, baseado na análise essencial de sistemas, e nos resultados descritos neste texto sobre a especificação do diálogo e da dinâmica.

Foram propostas estratégias de transformação, critérios de avaliação e a geração automática de código, todos baseados na especificação da dinâmica e do diálogo em um sistema interativo.

Todos estes resultados serviram para a conclusão de que o diálogo exerce um papel fundamental na especificação de

sistemas interativos, não concorrentes e com diálogo seqüencial.

Conclue-se que o diálogo poderia ser identificado na essência do sistema, e por isto sua especificação deveria ser antecipada para as fases mais recentes do processo de desenvolvimento de uma aplicação, antecedendo a etapa de especificação da dinâmica.

Destaca-se ainda que a especificação do diálogo proporciona a aproximação das etapas do desenvolvimento e fornece uma visão diferente do sistema, comumente encarado a partir de suas caraterísticas estáticas. Estas visões são complementares e levam a uma especificação mais completa da aplicação.

O modelo conceitual do diálogo proposto aqui se subdivide naturalmente em uma seção esquemática e uma seção detalhada, uma modelando o diálogo usuário-aplicação e outra o diálogo no sentido inverso. A seção esquemática se torna muito simplificada, pelo acréscimo de estados compostos. Além disso, o modelo conceitual possuí os requisitos de totalidade e conceitualização tão desejados.

Destaca-se que o método proposto para desenvolvimento de sistemas interativos apresenta características

importantes, entre elas, possuir baixa redundância de informação entre modelos, integrar visões do sistema que comumente aparecem isoladas e facilitar a automatização do desenvolvimento, por possuir várias tarefas mecânicas que podem ser realizadas.

6.2. Indicações para trabalhos futuros

Devido às várias comparações realizadas entre modelos conceituais realizadas, anteviu-se a possibilidade de especificar formalmente as informações passíveis de representação em um modelo.

Esta especificação teria como objetivo verificar a redundância de informações entre modelos, automatizando ainda mais a verificação da sua completude e correção, e a implementação (geração de código).

Num nível mais baixo, poderiam ser estudadas extensões deste trabalho para outras classes de sistemas, verificando-se a possibilidade de modelar diálogos assíncronos e especificar a dinâmica de funções concorrentes.

Para que isto fosse realizado, seria necessário o uso de particionamento de estados compostos em componentes

ortogonais, como colocado no início deste trabalho, e a extensão da associação de funções às transições, de forma a especificar o paralelismo na execução destas.

No nível de implementação, poderia ser definida uma ferramenta automática para reconhecimento de estados compostos, a partir das duas diretrizes colocadas aqui, ou seja, da análise de conectividade entre estados e a associação de conjuntos de estados a funções.

Esta ferramenta deveria ser capaz de gerar automaticamente os estados compostos e a associação de funções às transições, usando conceitos de Teoria dos Grafos e Inteligência Artificial, gerando automaticamente o modelo conceitual da dinâmica.

Ferramentas utilizando os critérios de avaliação propostos teriam um princípio de funcionamento baseado na remoção e inclusão de estados e transições. Neste tipo de trabalho poderia ser estudada a forma de manipulação destes objetos que levasse à melhor especificação.

Da mesma forma, os algoritmos para manipulação de máquinas de estados finitos poderiam ser estendidos, de forma a representarem sensibilidade ao contexto que estiverem manipulando.

BIBLIOGRAFIA

[Almei90] - Almeida, Mauro W. Barbosa, "Symetry and Entropy, Mathematical Metaphors in the work of Levi-Strauss", Current Antropology, Vol 31, No. 4, 367-385, Aug./Oct. 1990.

[Azeve90] - Azevedo, João Bisco Gomes de Oliveira, Modelagem dos aspectgos sócios-técnicos de um sistema de transporte, Tese Mestrado, Dep. Informática/Puc-RJ, Dez. 1990

[Barre79] - Barret,Willian A. & John D. Couch,John D., Compiler Construction: Teory and Pactice, Barret,Willian A. & John D. Couch,John D. Ed. Science Research Associetes Inc, 1979

[Couta87] - Coutaz, Joëlle, "The contruction of User Interfaces and the object paradigm", Lecture Notes in Computer Science, editores: Goods, G. & Hartmanis, J, No. 276, 121-130, 1987

[Dasgu89] - Dasgupta, Subrata, "The Structure of Design Processes", Advances on Computers, Vol. 28, 1-67, 1989

[DeMar89] - DeMarco, Tom, Análise Estruturada e Especificacão de Sistemas, Ed. Campus, 1989, 333 páginas

[Dener77] - Denert, Ernest, Specification and Design of Dialogue Systemas with State Diagrams, International Computing Symposium, 1977, Ed North Holland Publishing Company

[Fairl85] - Fairley, Richard E. Software Engineering Concepts, Ed. McGraw Hill, 1985, 364 páginas

[Flory78] - Flory, A & Kouloumdjam, J. "A model for the description of Information Systems Dynamics", Lecture Notes in Computer Science, No. 65, 1978, 307-318

[Fortu90] - Fortuna, Michel Heluey, Documentação do Sistema BIBLOS, manuscrito, 1990

[Gaine86] - Gaines, Brian R. & Shaw, Mildred L. G. "Foundations os dialog engineering: the development of human-computer interaction - Part II", International Journal of Man-Machine Studies, 1986, Vol. 24, 101-123.

[Harel88] - Harel, David, "On Visual Formalisms", Communications of ACM, Vol. 31, No. 5, May 1988, 512-530

[Harts89a] - Hartson, Rex & Hix, Debora "Humam-Computer Interface Development: Concepts and Systems for its Management", ACM Computing Surveys, Vol. 21, No. 1, Mar. 1989, 5-92

[Harts89b] - Hartson, Rex, "User Interface Management Control and Comunication", IEEE Software, No. 1, 62-70, Jan. 1989

[Heuse91] - Heuser, Carlos A. Modelagem Conceitual de Sistemas = Redes de Petri, V Escola Brasileiro-Argentina de Informática, 1991, 150 páginas

[Jabob83] - Jacob, Robert J. K., "Using formal specifications in the design os a Human Computer Interfaces", Communications of ACM, Vol. 26, No. 4, 259-264, Apr. 1983

[Jacob86] - Jacob, Robert J. K. "A Specification Language for Direct-Manipulation User Interfaces", ACM Transactions on Graphics, Vol. 5, No. 4, 283-317, Oct. 1986

[Kasik82] - Kasik, David J., "A User Interface Managemant System", Computer Graphics, Vol. 16, No. 3, 99-106, Jul. 1982

[Keuff91a] - Keuffel, Warren. "Event lists and Essencial Models", Computer Language, Vol. 8, No. 5, Mai 1991.

[Keuff91b] - Keuffel, Warren. "Transformation Strategies", Computer Language, Vol. 8, No. 10, Oct. 1991.

[Kung89] - Kung, C. H. "Conceptual Modeling in the Context of Software Development", IEEE Transactions on Software Engineering, vol 15, No. 10, Oct. 1989, 1176-1187

[Laend89] - Kaender, Alberto Henrique Frade & Fernandes, Antônio Carlos, "MER+: Uma extensão do modelo de entidades e relacionamentos para projeto conceitual", Revista Brasileira de Computação, No. 1, 5-14, ul/Set. 1989

[Lucen87] - Lucena, Carlos J. P. Inteligência Artificial e Engenharia de Software, Publicações Acadêmico/Científicas PUC/RJ e IBM Brasil, 1987, 305 páginas

[Meira91] - Meira, Carlos A. Alves e Masiero,Paulo Cesar, Um Gerador de Aplicações para Sistemas Reativos, Anais do V Simpósio Brasileiro de Engenharia de Software, Out. 1991, 45-60

[Myers89] - Myers, Brad A."User Interface Tools: Introduction and Survey", IEEE Software, Jan. 1989, Vol. , No. 1, 15-23

[Page88] - Meilir Page-Jones, Projeto Estruturado de Sistemas, Ed. Mc-Graw Hill, 1988

[Peter81] - Peterson, James Lyle, Petry Mets Theory and the modeling os systems, Ed. Prentice-Hall Inc., 1981, 368 páginas

[Prots91] - Protsko, L. Beth, Sorenson, Paul G., Tremblay J. Paul & Schaefer, Douglas A. "Towards the Automatic Generation of Software Diagrams", IEEE Transactions on Software Enginnering, Vol. 17, No. 4, Jan. 1991, 10-21.

[Quent88] - Quental,Francisco de San Tiago Dantas Barbosa Modelagem da Dinâmica de Sistemas Interativos, Tese-Mestrado, Dep. Informática/PUC-RJ Jan. 1988.

[Silva85] - Silva, Marcelo Vieira, Processador de Diálogo: Uma ferramenta para geração de sistemas interativos, Tese Mestrado, FEE/UNICAMP, Mai. 1985

[Ward84] - Ward, P. T. Systems Development Without Pain, Yourdon Press, 1984

[Ward86] - Ward, P. T.,"The Transformation Schema: An Extension of the DFD to represent Control and Timing, IEEE Transactions on Software Engineering, vol SE-12, No. 12, Feb. 1986, 198-210

[Wu89] - Wu, Peter Y. F. "User Interface Management Systems: Survey and Assesment", IBM Research Report, 18 páginas, Mai 1989.

Apêndice

Notação sugerida para elaboração de Modelos Conceituais

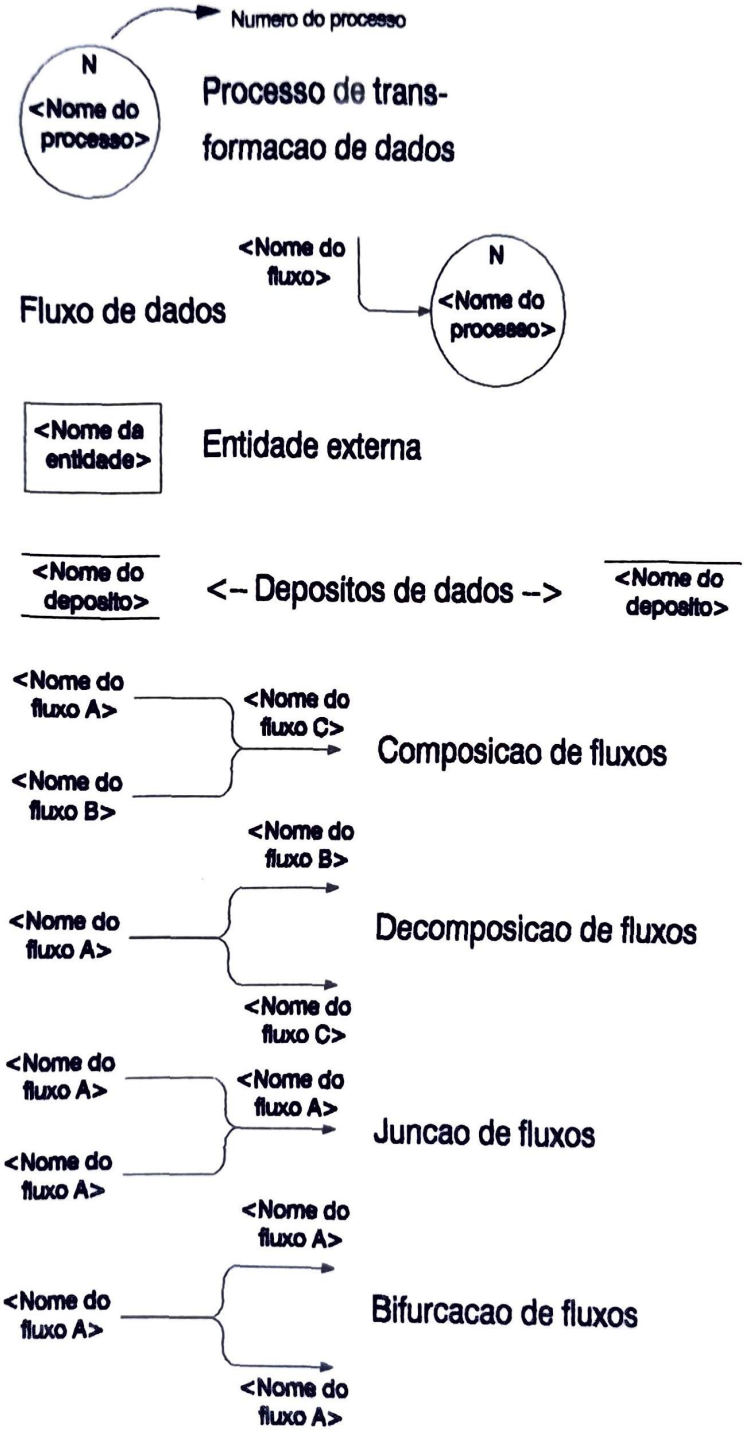


FIGURA A-1. Notacao para Diagramas de Fluxos de Dados

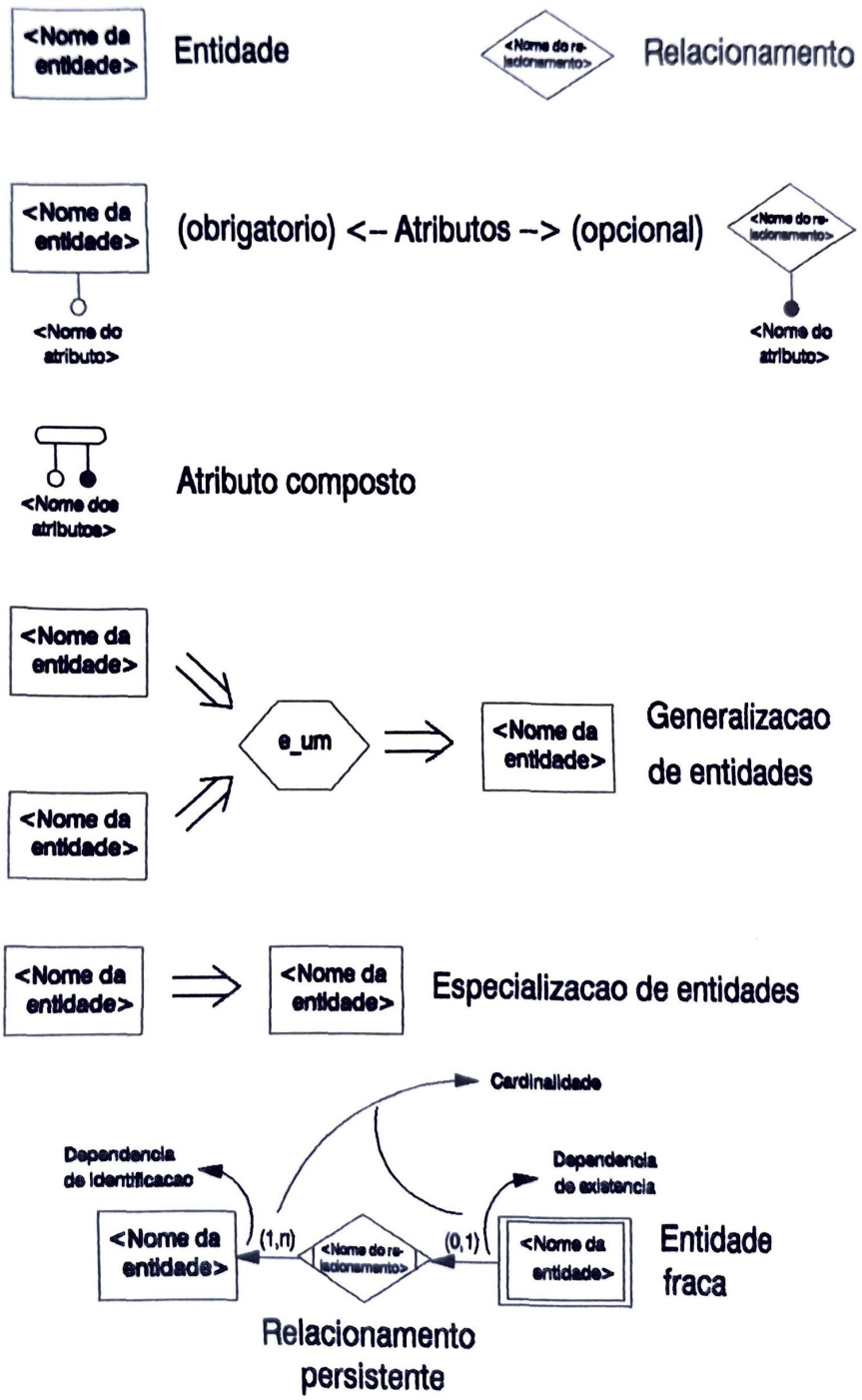


FIGURA A-2. Notacao para Diagramas de Entidades e Relacionamentos

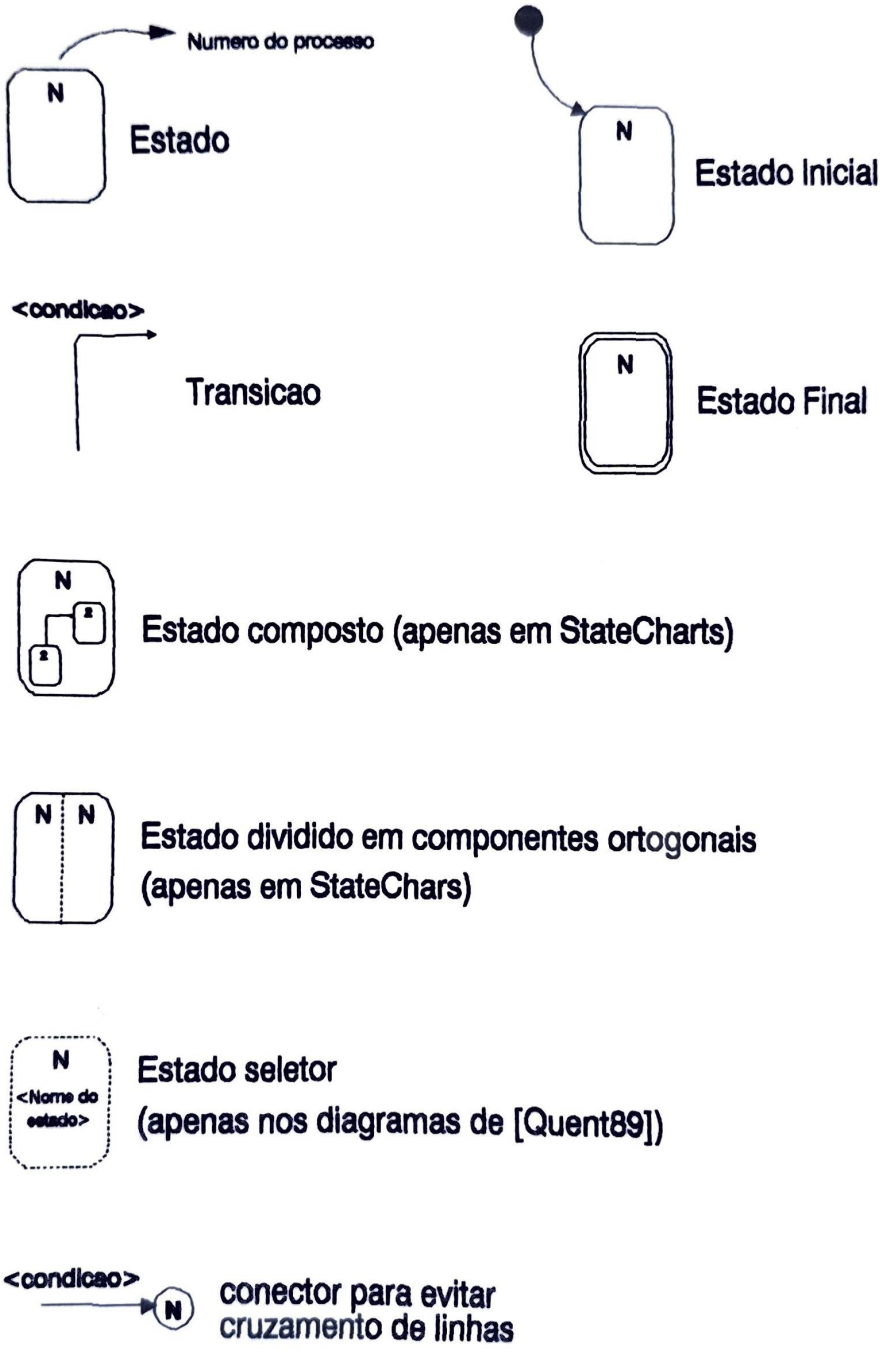


FIGURA A-3. Notacao para Diagramas de Estados e Transicoes

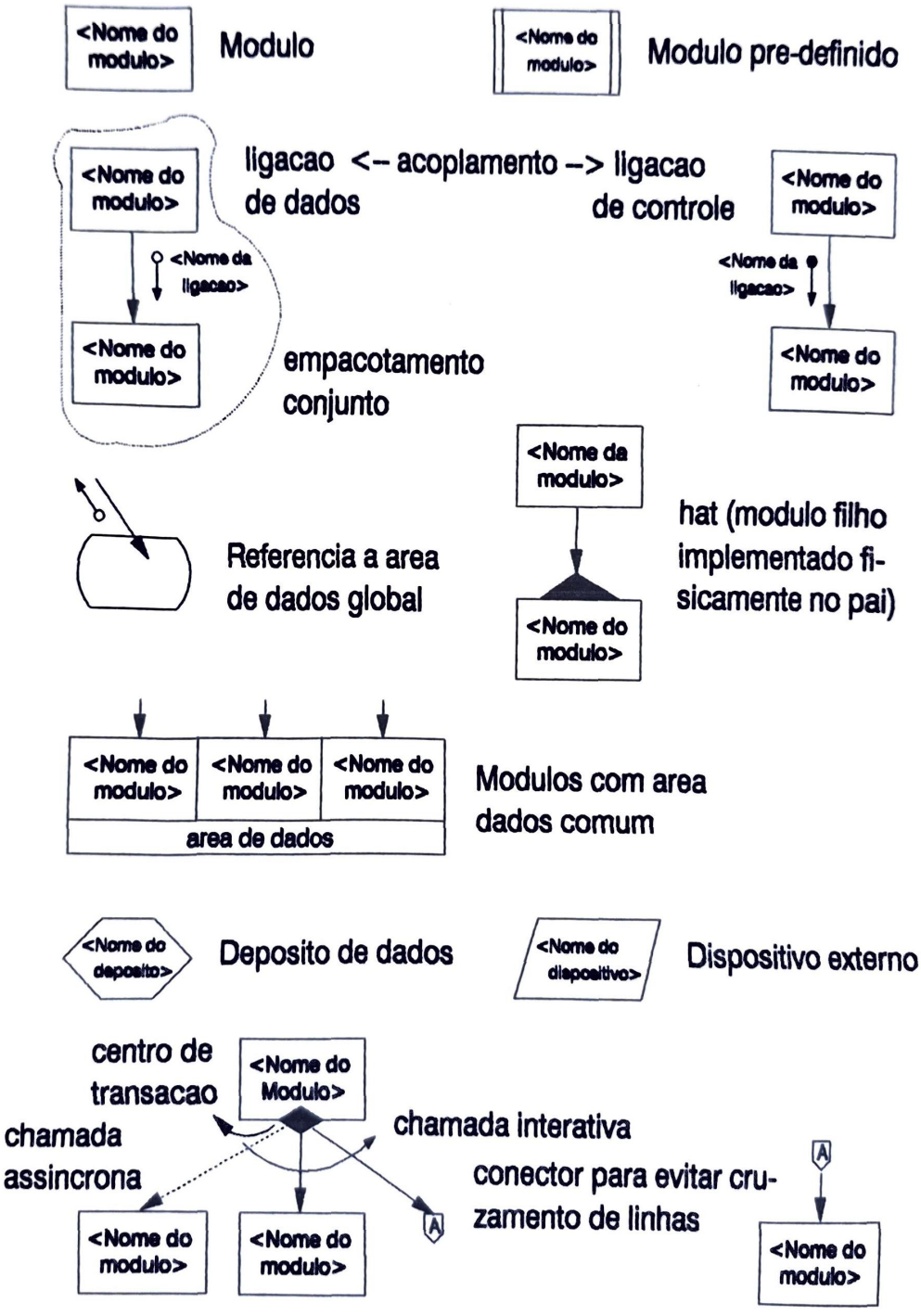


FIGURA A-4. Notacao para Diagramas Hierarquicos de Modulos