MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO DE PESQUISAS ESPACIAIS

AUTORIZAÇÃO PARA PUBLICAÇÃO
AUTHORIZATION FOR PUBLICATION

PALAVRAS CHAVES/KEY WORDS
AUTORES/AUTHORS
ANÁLISE ESTRUTURADA
SOFTWARE TEMPO REAL
ESPECIFICAÇÃO DE SISTEMAS

AUTORIZADA POR/AUTHORIZED BY
Ralf Giełow
Pres.Cons.Pos-Grad.

AUTOR RESPONSAVEL
RESPONSIBLE AUTHOR
André L.R. Mayoral

DISTRIBUIÇÃO/DISTRIBUTION
[ ] INTERNA / INTERNAL
[X] EXTERNA / EXTERNAL
[ ] RESTRITA / RESTRICTED

REVISADA POR / REVISED BY
Luiz Alberto V. Dias

CDU/UDC
681.3.014

DATA / DATE
Maio 1990

PUBLICAÇÃO Nº
PUBLICATION NO
INPE-5078-TDL/412

ORIGEM
ORIGIN
PG/LAC

PROJETO
PROJECT
CAP

Nº DE PAG.
NO OF PAGES
182

ULTIMA PAG.
LAST PAGE
A.16

VERSÃO
VERSION

Nº DE MAPAS
NO OF MAPS

TÍTULO/TITLE
COMBINAÇÃO DOS MÉTODOS HATLEY E WARD/MELLOR PARA SISTEMAS TEMPO-REAL

AUTORES/AUTHORSHIP
André Luiz Rosa Mayoral

RESUMO - NOTAS / ABSTRACT - NOTES

O uso crescente das técnicas de análise estruturada de sistemas em aplicações comerciais levou a tentativas de se usar essas mesmas técnicas em aplicações de engenharia. Por sua vez, sua utilização no desenvolvimento de sistemas tempo-real mostrou a necessidade de se estender o método para representar aspectos peculiares a tais sistemas. Duas extensões foram feitas: a de Hatley e a de Ward e Mellor. Este trabalho analisa os dois métodos e faz sua combinação. O método híbrido resultante usa a abordagem dirigida por eventos (de Ward e Mellor), na estrutura do modelo de requisitos de Hatley, à qual se juntou os diagramas de estrutura do método JSD para modelar entidades externas.

OBSERVAÇÕES / REMARKS

Dissertação de Mestrado em Computação Aplicada, aprovada em março de 1990.

Aprovada pela Banca Examinadora
em cumprimento a requisito exigido
para a obtenção do Título de Mestre
em Computação Aplicada

Dr. Flávio Roberto Dias Velasco

Presidente

Dr. Luiz Alberto Vieira Dias

Orientador

Dr. Nizam Omar

Membro da Banca
-convidado-

Dr. Tatuó Nakanishi

Membro da Banca

Candidato: André Luiz Rosa Mayoral

São José dos Campos, 05 de março de 1990

ABSTRACT

The increasing use of structured analysis techniques for business applications yelded to its use for engineering applications. In so doing, the use of structured analysis for real-time systems development showed the need to extend the method to represent aspects specific to those systems. Two such extensions where made, one by Hatley and the other by Ward and Mellor. This work describes a combination of these two real-time structured analysis methods. The hybrid method uses the event-driven approach (from Ward and Mellor) in the Hatley's requirements model structure. JSD structure diagrams where added to the model structure to describe external entities.

AGRADECIMENTOS

Ao meu orientador, Dr. Vieira Dias, ao Dr. Tatuó Nakanishi, ao Dr. Velasco (pela idéia de usar o JSD); ao prof. Francisco Moral, pela cuidadosa revisão; ao apoio do amigo e colega Darlan C. Marliere; ao vívido incentivo dos amigos Mário Kataoka Fº e Francisco A. F. Gomes Fº; e sobretudo a Geni, pela paciência e apoio extremos.

Este trabalho foi parcialmente financiado pela EMBRAER, à qual agradeço nas pessoas dos engenheiros Mário Lehmert Renaud, Fernando J. Franchi e Carlos Augusto R. de Oliveira.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS

BNF	- Backus-Naur Form
CCD	- Diagrama de contexto de controle
CFD	- Diagrama de fluxo de controle
CSPEC	- Especificação de controle
DD	- Dicionário de dados
DCD	- Diagrama de contexto de dados
DER	- Diagrama de entidades e relacionamentos
DFD	- Diagrama de fluxo de dados
DTE	- Diagrama de transição de estados
JSD	- "Jackson System Design"
PDL	- Linguagem de projeto de programas
PSPEC	- Especificação de processo
RD	- Dicionário de requisitos
RDI	- Item do dicionário de requisitos
SA	- Análise Estruturada

CAPÍTULO 1

INTRODUÇÃO

Este trabalho trata de dois métodos de análise estruturada de sistemas do tipo tempo-real, ambos desenvolvidos como extensão da análise estruturada de sistemas de DeMarco (1978). Originou-se da constatação de que, embora sendo semelhantes, e tendo o mesmo objetivo, havia entre eles diferenças e também recursos de modelagem não comuns. Logo, talvez pudessem ser combinados com vantagem. Nosso objetivo é combiná-los para modelar requisitos de sistemas tempo-real de grande porte, segundo os critérios do Capítulo 5. Configurou-se daí o seguinte trabalho: descrever os métodos, compará-los quanto a seus recursos de modelagem, verificar que recursos são mais adequados à abordagem adotada e, finalmente, combinar os dois métodos ou mesmo decidir por um deles.

É importante ressaltar que as descrições dos métodos feitas neste trabalho não são adequadas para se aprender a usá-los. Elas têm o seguinte objetivo: servir de base para uma comparação detalhada dos métodos com o propósito de combiná-los aproveitando as vantagens de cada um. As descrições dos métodos contidas nas duas referências principais (Hatley e Pirbhai, 1987; Ward e Mellor, 1985) têm naturalmente o objetivo de apresentar e ensinar o uso dos métodos. Assim, o assunto é organizado de maneira didática. Para atender ao objetivo deste trabalho, foi necessária uma reorganização do material da descrição dos métodos separando-o segundo aspectos distintos. Isso facilitará uma comparação sistemática, comparando aspecto por aspecto.

Como introdução, este capítulo apresenta uma caracterização de sistemas tempo-real --para situar o assunto--, um breve histórico da evolução da análise estruturada que levou até essas extensões, e por fim, estabelece a terminologia usada para organizar as descrições feitas.

1.1 - CARACTERIZAÇÃO DE SISTEMAS TEMPO-REAL

Com a finalidade de estabelecer o contexto no qual se insere este trabalho, esta seção apresenta algumas definições, exemplos e características de sistemas chamados tempo-real.

Em primeiro lugar, examinemos o conceito de "sistema", para podermos fazer algumas distinções úteis. No dia a dia vemos quase tudo sendo chamado de sistema e, a princípio, a palavra pode parecer quase tão vaga quanto a palavra "coisa". No entanto, o conceito de sistema, embora muito geral e abrangente, pode ser muito útil em todas as ciências. Von Bertalanffy (1969) define um sistema como sendo um conjunto de elementos que interagem entre si (note que "conjunto" é outra noção muito geral, abrangente, e útil).

Weinberg (1975), ao citar a definição de Hall e Fagen (1968) de que "um sistema é um conjunto de objetos junto com relacionamentos entre os objetos e entre seus atributos", ressalta que embora a idéia de relacionamentos seja parte essencial do conceito de sistema, a definição falha ao não dar a mais leve indicação de que um sistema é relativo ao ponto de vista do observador. Esta noção de ponto de vista é enfatizada na definição dada por Floyd (1988):

> "Um sistema é uma parte do mundo, a qual uma pessoa (ou grupo de pessoas), durante algum intervalo de tempo, e por alguma razão --escolhe considerar como um todo consistindo de componentes, cada componente caracterizado por propriedades que são selecionadas como sendo relevantes, e por ações que se relacionam a estas propriedades e às propriedades de outros componentes."

Floyd ressalta ainda que, no âmbito do desenvolvimento de software, estamos preocupados com "sistemas" de duas maneiras diferentes:

- A parte do mundo real que levamos em conta ao desenvolver programas é considerada um sistema; o "sistema de referência".

- O conjunto de programas que produzimos, e suas interfaces, são considerados um sistema; o "sistema de software".

Repassando, vimos então:

- que um sistema é um conjunto,

- que seus componentes se relacionam e interagem,

- e que tanto os componentes, quanto os relacionamentos e interações entre eles, são selecionados (dentre os inúmeros possíveis) para caracterizá-lo como "sistema", dentro de uma certa ótica, e "por alguma razão", isto é, com algum objetivo em mente.

A distinção entre o que Floyd chama de "sistema de referência" e o sistema sendo desenvolvido é particularmente útil para o desenvolvimento de sistemas "tempo-real", pois estes são freqüentemente considerados como "sistemas embutidos" num sistema maior, do qual fazem (ou farão) parte. Nesta situação, o sistema em desenvolvimento não é necessariamente um sistema apenas de software:

- geralmente é considerado consistir de software e de hardware, às vezes incluindo pessoas,

- o sistema em desenvolvimento fará parte --estará embutido-- no sistema de referência. O restante do sistema de referência será considerado como seu "ambiente",

- e terá características de sistema tempo-real.

Vejamos então o conceito de "tempo real". Este conceito é particularmente difícil de definir de maneira a incluir todos os sistemas com as características (citadas adiante) de tempo real e ao mesmo tempo excluir os chamados "sistemas interativos 'on-line'". Estes últimos, embora muitas vezes sejam chamados de sistemas tempo-real

quando requerem tempo rápido de resposta, possuem poucas --se algumas-- das características de sistemas tempo-real (Ward e Mellor, 1985).

Uma boa definição, que parece excluir tais sistemas, é dada por Wirth (1983), só que ele define em termos de "programa". Ele parte da idéia de programa seqüencial, e chega à de programa tempo-real acrescentando poucos conceitos simples de cada vez; passando pela definição de multiprograma.

Wirth começa definindo programa seqüencial como progama em que cada ação começa quando sua predecessora única foi completada. Ressalta que esta restrição faz com que os resultados computados sejam independentes da velocidade (tempo) de execução de seu processador. Assim, o tempo de execução não é uma propriedade característica do programa, e sim do processador empregado.

Em seguida, Wirth define multiprograma (Figura 1.1) como sendo um programa consistindo de vários processos com as seguintes características:

- cada processo é puramente seqüencial,

- são executados (possivelmente) concorrentemente por vários processadores,

- comunicam-se via variáveis compartilhadas (com exclusão mútua de acesso) e sinais de sincronização;

havendo duas razões para usá-los. A principal é menor tempo de execução. Outra é que há processadores com capacidades especiais que outros não têm (exemplo: dispositivos de entrada e saída). Wirth ressalta que multiprogramas podem e devem ser projetados de maneira que sejam independentes do tempo de execução absoluto ou relativo dos processadores, de modo que sua validade só dependa das instruções do programa.

Por fim, Wirth define programa tempo-real como "um multiprograma cuja validade depende do tempo de execução dos processadores utilizados". A principal razão para usá-los, bem diferente da de multiprogramas, é que "há certos processos que NÃO ESPERAM por sinais de sincronização emitidos pelo programa tempo-real, devido ao fato de não serem programáveis e/ou estarem no 'ambiente' do programa", como mostra a Figura 1.2. Assim, "cooperar com tais processos [e aqui entra validade] necessariamente dependerá do tempo de execução dos processadores utilizados".

Esta definição é dada para um programa tempo-real, pois o assunto do artigo citado é sobre linguagens de programação. Este trabalho trata de sistemas constituídos de hardware e de software. Assim, a expressão "sistema tempo-real" refere-se aqui a um sistema de vários componentes de hardware, cada um dos quais --e ao menos um deles-- podendo abrigar um programa tempo-real como definido por Wirth. A Figura 1.3 ilustra um sistema tempo-real típico.

Neste trabalho, o conceito associado ao termo "tempo-real" é esse da definição de Wirth. Como o processo com o qual o programa (ou o sistema incluindo o programa) interage não espera sinais de sincronização, o sistema tempo-real tem que dar conta de acompanhá-lo por seus próprios meios, e isso depende da velocidade dos processadores empregados. É diferente de um multiprograma; nele, todos os processos são programáveis. Pode-se fazer com que todos se sincronizem. Se não se tomar cuidado, a validade do multiprograma acabará dependendo da capacidade dos processadores; no entanto, sua validade sempre pode (e deve, como ressaltou Wirth) independer dos processadores.

Um exemplo de sistema tempo-real é o de controle de tráfego numa área central de uma cidade. Há vários sensores que sinalizam a passagem de carros por pontos determinados, de modo que o sistema possa controlar todos os semáforos da área, para otimizar o fluxo de tráfego. É a passagem dos carros que envia sinais para o sistema através dos sensores. O "processo de referência" é o trânsito

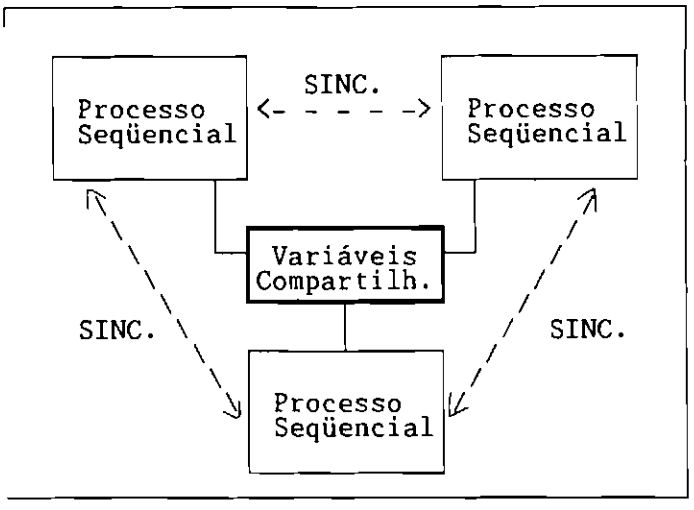


Fig. 1.1 - Multiprograma.

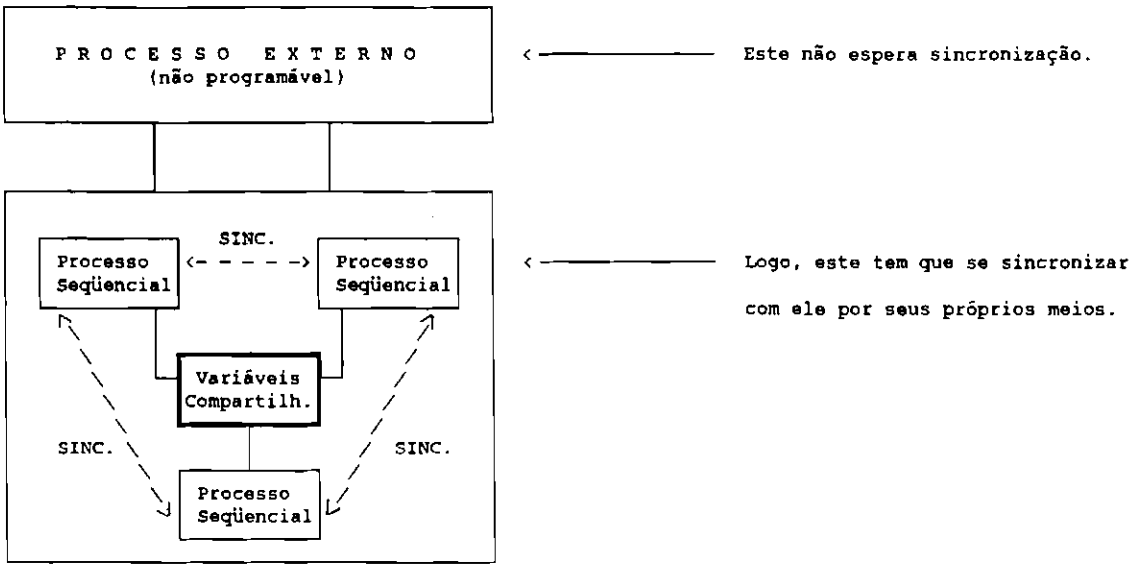


Fig. 1.2 - Programa tempo-real.

na área. Ele é idealizado pela passagem de veículos por dois tipos de pontos: uns com sensores e outros com semáforos, que são controlados. Esta idealização é a visão, de um certo ponto de vista, que se tem do sistema de referência. O trânsito não espera que o sistema de controle sinalize que armazenou o sinal da passagem de um carro, para depois enviar outro. A validade do programa depende de que a velocidade de execução seja suficiente para acompanhar um intervalo de tempo mínimo entre sinais do sensor. Outros exemplos são:

- Sistemas de controle e/ou monitoração de processos industriais, tais como fábricas de produtos químicos, linhas de montagem, linhas de laminação de aço, refinarias.

- Sistemas de controle de veículos, tais como controle de velocidade de cruzeiro de automóveis, condução de míssil, controle de vôo de aviões.

Estão listadas abaixo algumas propriedades que caracterizam sistemas tempo-real. Duas características importantes são apontadas por Hatley (1986).

- Sistemas tempo-real contêm dois tipos distintos de informação. Um tipo é o familiar "fluxo de dados", que é usado dentro de processos de dados. O outro tipo tem o objetivo principal de modificar a resposta do sistema aos dados que entram, em vez de ser processado por ele: é o "fluxo de controle".

- É requerido de sistemas tempo-real que reconheçam eventos passados, estado atual e eventos futuros esperados; e que modifiquem a resposta do sistema conforme seu estado.

São essas duas características acima as mais relevantes para as extensões feitas na análise estruturada descritas neste trabalho.

Ward e Mellor (1985) também apontam algumas características que são comuns em sistemas tempo-real:

- seu ambiente freqüentemente contém dispositivos que agem como sensores,

- o sistema freqüentemente processa entradas múltiplas concorrentemente,

- o vocabulário da formulação do problema para sistemas tempo-real é geralmente tirado de disciplinas científicas ou de engenharia (pode não parecer mas este tópico merece menção, pois tem a ver com a definição da fronteira entre o sistema de referência, ou "o ambiente", e o sistema em desenvolvimento),

- muitas vezes as escalas de tempo usadas são muito pequenas para os padrões humanos,

- a precisão requerida no tempo de resposta geralmente é maior que a requerida para outros sistemas.

É interessante ressaltar agora que, em qualquer aplicação de sistemas tempo-real, a escolha do sistema de referência, ou seja de seu "ambiente", e a do "sistema tempo-real", geralmente atende a um de dois objetivos gerais:

- monitoração de processo(s) no sistema de referência, ou

- controle de processo(s) no sistema de referência. Esse controle sempre requererá monitoração.

Conforme esses objetivos, Murata (1985) classifica sistemas tempo-real em sistemas de aquisição de dados e sistemas de controle de processos, respectivamente. A Figura 1.3 é derivada de Murata (1985) e apresenta um sistema tempo-real genérico (no caso de sistemas de aquisição de dados, não existe o sub-sistema atuador).

Com estas definições, exemplos e características de sistemas tempo-real, procurou-se estabelecer um contexto de referência para a apresentação dos dois métodos a seguir, de modo a caracterizar o tipo de problema que eles procuram resolver. Antes de passarmos às descrições, apresentamos na Seção 1.2 a origem desses dois métodos.

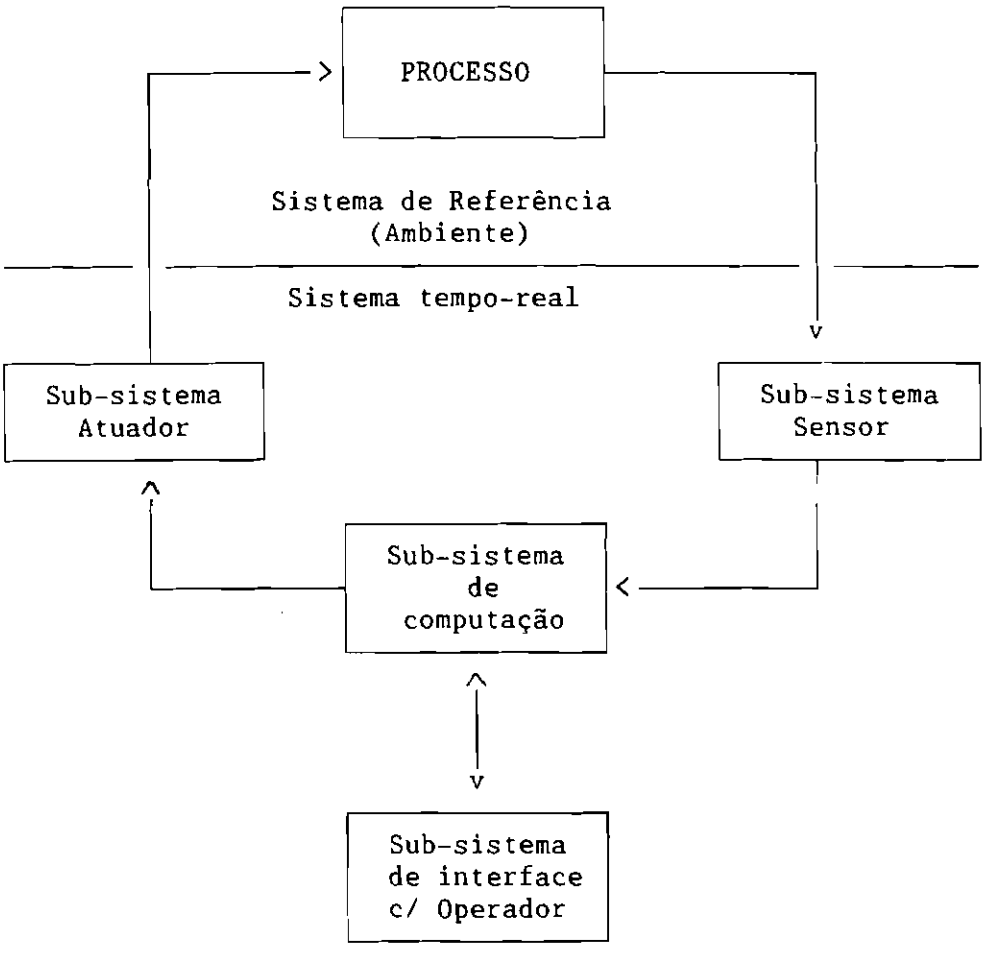


Fig. 1.3 - Sistema tempo-real genérico e seus componentes.
FONTE: Murata (1985), p. 6.

1.2 - A EVOLUÇÃO DA ANÁLISE ESTRUTURADA CONVENCIONAL

A Análise Estruturada de Sistemas (DeMarco, 1978; Gane e Sarson, 1979) é um método baseado no conceito de decomposição funcional, voltado para análise de requisitos de processamento de dados. Foi desenvolvido para sistemas de processamento de dados, sendo portanto bem adequado a aplicações comerciais. Esse método é bastante difundido e amadurecido pela prática. O amadurecimento levou a algumas alterações no processo de modelagem descrito por DeMarco (1978). De um trabalho de "análise do processamento de dados feito por um sistema existente a ser automatizado", passou-se a fazer com o método um trabalho de "análise dos requisitos formulados pelo cliente para um sistema a ser desenvolvido". Essa alteração reflete a informatização ocorrida nas empresas desde aquela época.

Um avanço significativo na análise estruturada foi o trabalho desenvolvido por McMenamin e Palmer, no começo dos anos 80, culminando com seu livro "Essential System Analysis" (McMenamin e Palmer, 1984).

Na falta de métodos mais adequados, a Análise Estruturada vinha sendo usada também para análise e especificação de requisitos de sistemas tempo-real para aplicações industriais e militares. Com a crescente complexidade dos sistemas tempo-real e o uso crescente de software embutido em tais sistemas, surgiu uma necessidade cada vez maior de métodos mais adequados para seu desenvolvimento.

Surgiram recentemente duas extensões da Análise Estruturada para sistemas tempo-real: o método Ward/Mellor (Ward, 1985) e o método Hatley (Hatley, 1985). Esses métodos acrescentaram às técnicas de representação de requisitos de processamento de dados, técnicas adicionais para representar requisitos de controle e requisitos de tempo.

1.3 - MÉTODOS: CONCEITOS E TERMINOLOGIA

Antes de iniciar a descrição dos métodos, convém ter uma visão geral que será útil para criar um arcabouço conceitual no qual os métodos se inserem.

Em primeiro lugar, ambos os métodos têm o mesmo propósito: produzir uma completa especificação de requisitos para o sistema, que evite ao máximo restringir sua implementação. Essa especificação é chamada de "modelo", pois é vista como um modelo, essencialmente gráfico, dos requisitos de dados, de controle e de tempo do sistema.

Esse modelo de requisitos especifica os requisitos essenciais (McMenamin e Palmer, 1984) para o sistema, i.e. "o que" o sistema deverá fazer, "que" problema o sistema deverá resolver, independentemente de qualquer que seja a forma de implementação. Ele não especifica a estrutura do projeto do sistema, i.e. "como" será organizada a implementação do sistema. É interessante notar que ambos os métodos baseiam-se no conceito de requisitos essenciais de McMenamin e Palmer para separar análise de projeto.

Em segundo lugar, é útil deixar claro desde já o conceito de "método" utilizado neste trabalho. Um "método", em termos simples, é uma maneira de fazer as coisas. Mais especificamente é "uma definição clara do que fazer e uma maneira bem definida de o fazer" (Freeman, 1983, p. 13). Para fixar terminologia, podemos dizer:

MÉTODO	=	MODELO dos requisitos	+	PROCESSO DE CONSTRUÇÃO do modelo
		(o que fazer)		(maneira de o fazer)

Cada autor usa um nome para a especificação de requisitos desenvolvida com seu método. Ward e Mellor a chamam de "Modelo Essencial", mantendo o termo de McMenamin e Palmer; Hatley a chama de "Modelo de Requisitos".

Fixados esses conceitos e terminologia, pode-se passar à descrição dos métodos. O Capítulo 2 descreve o método Hatley, e o Capítulo 3 descreve o método Ward/Mellor.

CAPÍTULO 2

MÉTODO HATLEY

Neste capítulo é descrito o método de Hatley de modelagem de requisitos. Vejamos inicialmente a abordagem adotada para a modelagem de requisitos, bem como as extensões feitas ao modelo convencional da análise estruturada.

A abordagem do método de Hatley usada na modelagem de requisitos é a de "modelagem dirigida por funções", usando decomposição funcional. Ou seja, os requisitos formulados pelo cliente são expressos como grandes funções (processos) do sistema, que são sucessivamente decompostas, ou refinadas, em sub-funções cada vez mais simples; sempre procurando modelar os "requisitos essenciais" (McMenamin e Palmer, 1984), aqueles que devem ser satisfeitos qualquer que seja a implementação do sistema.

Quanto às extensões feitas, elas visam basicamente a representação de aspectos da dinâmica do sistema. O método convencional de análise estruturada de sistemas (SA) foi desenvolvido para sistemas que não são do tipo tempo-real, como automação de bancos e outras aplicações comerciais. Devido à natureza de tais sistemas, a filosofia é adiar todas as considerações de tempo e de controle para a fase de projeto. Mesmo porque nestes sistemas a idéia de controle se refere a condições de testes feitos sobre dados, que vão afetar o fluxo de controle do programa, i.e., em qual seqüenciamento as instruções do programa tomarão controle da CPU do processador usado. Tais detalhes não são importantes na fase de análise de requisitos. Contudo, em sistemas tempo-real, requisitos de tempo e controle são tão complexos e importantes quanto requisitos de processamento de dados. No caso de sistemas tempo-real, "controle" tem outro significado: se refere a mudanças de estado do sistema para se adequar a eventos ocorridos no seu ambiente. Isto implica que o sistema tem comportamentos diferentes (estados), dependendo das condições externas detectadas (eventos). Conforme o estado há funções que, por exemplo, são proibidas de aceitar

entradas que surjam, pois não podem funcionar naquele modo de comportamento.

Das características específicas que distinguem sistemas tempo-real de sistemas não tempo-real duas são particularmente importantes no método Hatley; são aquelas já listadas no final da Seção 1.1.

A título de introdução, vamos listar na Tabela 2.1 a terminologia usada pelo autor. Ela é derivada, como era de se esperar, da terminologia da SA clássica (DeMarco, 1978). Esta lista também serve para dar uma rápida visão das extensões feitas pelo novo método.

TABELA 2.1

EXTENSÕES DO MÉTODO HATLEY

SA CLÁSSICA	MÉTODO HATLEY	
Diagrama de contexto	Diagrama de contexto de dados	(DCD)
_____	Diagrama de contexto de controle	(CCD)
Diagrama de fluxo de dados	Diagrama de fluxo de dados	(DFD)
_____	Diagrama de fluxo de controle	(CFD)
mini-especificação	Especificação de processo	(PSPEC)
_____	Especificação de controle	(CSPEC)
_____	Especificação de tempos	
Dicionário de dados	Dicionário de requisitos	(RD)

O método de Hatley é descrito em duas partes. A Seção 2.1 contém uma descrição do Modelo de Requisitos já construído. A Seção 2.2 apresenta o "processo de construção" do modelo.

2.1 - O MODELO DE REQUISITOS

Para permitir uma comparação sistemática dos dois métodos descritos neste trabalho, a descrição que se segue do Modelo de Requisitos é dividida em três partes. Cada parte refere-se a um aspecto distinto do modelo:

1) sua Estrutura, ou seja, a maneira como seus componentes são organizados e se relacionam,

2) sua Notação, que é a "sintaxe" do método, ou seja, como representar seus vários componentes graficamente, e

3) suas Convenções, que compõem a "semântica" do método, isto é, o significado (ou a interpretação) dos componentes. Sem elas, não se pode compreender o que se pretendeu representar pelo modelo.

O Apêndice A contém algumas ilustrações do modelo de requisitos do método Hatley.

2.1.1 - ESTRUTURA

O Modelo de Requisitos é organizado numa estrutura que está ilustrada na Figura 2.1. Esta seção descreve seus componentes e sua organização baseado nesta figura.

A coluna de DFDs à esquerda corresponde ao modelo de SA clássica: uma hierarquia de diagramas onde cada nível é um detalhamento do nível anterior (para fins de ilustração é mostrado um só diagrama em cada nível), terminando em PSPECs.

A coluna da direita representa também uma hierarquia, mas de CFDs. É uma hierarquia "paralela" à de DFDs: cada CFD corresponde a um DFD. Tal como os DFDs mostram os caminhos seguidos pelos fluxos de dados, os CFDs mostram os caminhos seguidos pelos fluxos de controle.

A coluna central mostra as CSPECs, que representam os requisitos de controle do sistema. O propósito da CSPEC é especificar duas coisas: processamento de controle e controle de processos. Isto será detalhado adiante, na Seção 2.1.1.2. Cada CSPEC é associada a um CFD e seu DFD correspondente. Este é um aspecto importante da estrutura do Modelo de Requisitos.

Cada nível do modelo equivale ao nível anterior, só que com mais detalhes. Todos os processos de um nível são mutuamente exclusivos; ou seja, nenhum processo tem sub-processo em comum com outro.

A Especificação de Tempos de Resposta registra os requisitos de tempo de resposta entre entradas e saídas do sistema.

O Dicionário de Requisitos contém definições de todos os fluxos do sistema, tanto de dados como de controle.

As várias setas que ligam os componentes da estrutura representam, nesta figura, o balanceamento de fluxos de dados e de controle entre eles (ex.: fluxos de controle que entram e saem de um CFD devem entrar e sair do processo pai no CFD do nível acima). Pode-se dizer que é o balanceamento que mantém "ligada" a estrutura do modelo.

É comum chamar-se os DFDs e PSPECs de "Modelo de Processos" e os CFDs e CSPECs de "Modelo de Controle". Os dois, junto com o Dicionário de Requisitos e a Especificação de Tempo formam o Modelo de Requisitos. Nas próximas duas seções, descrevemos com um pouco mais de detalhe o Modelo de Processos e o Modelo de Controle.

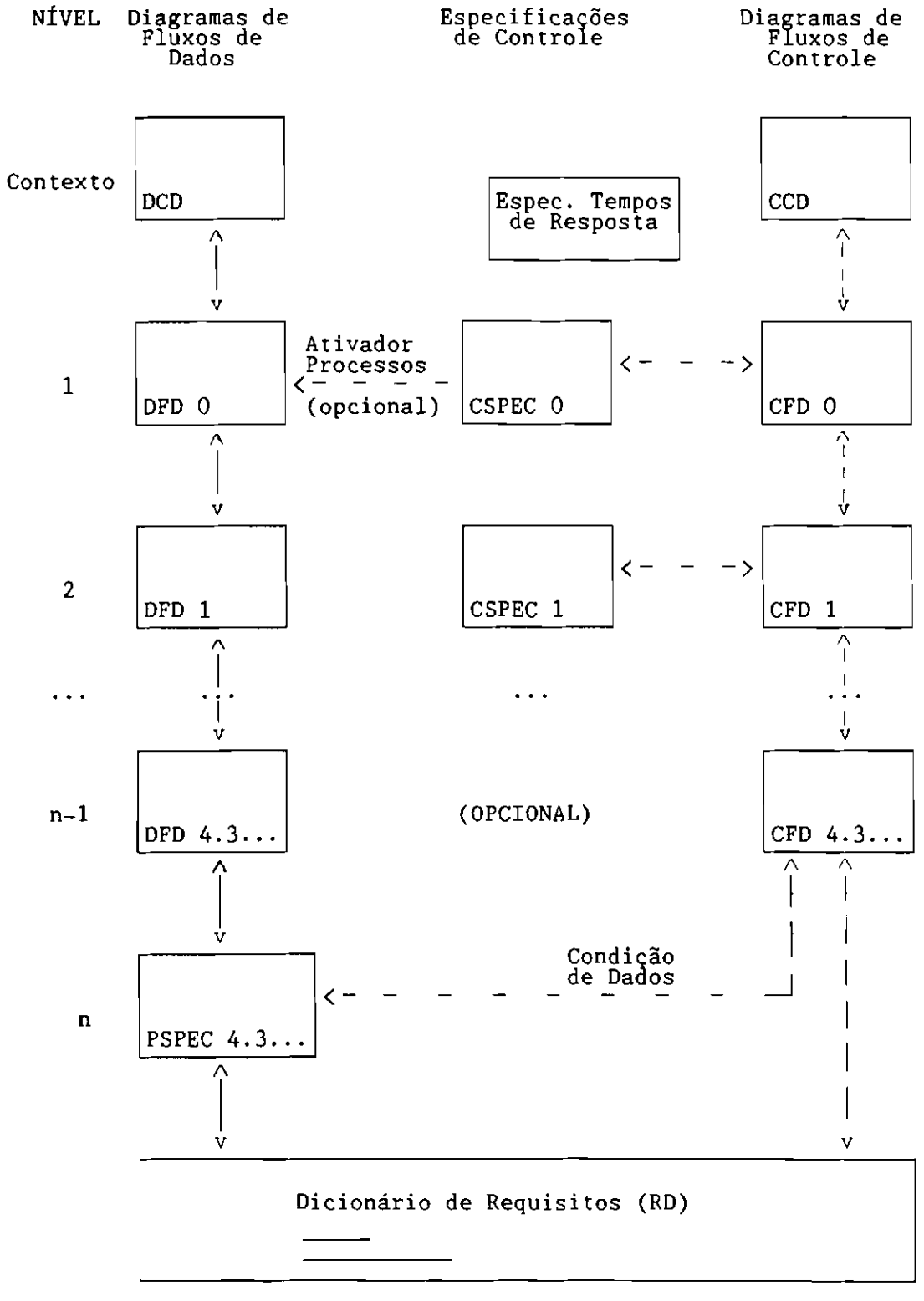


Fig. 2.1 - Estrutura do Modelo de Requisitos.

2.1.1.1 - O MODELO DE PROCESSOS

O objetivo do Modelo de Processos é especificar os requisitos de processamento de dados do sistema.

No nível mais alto, o de contexto, temos o DCD. O Diagrama de Contexto de Dados é o mesmo diagrama de contexto da SA clássica: mostra as entidades externas com as quais se requer que o sistema se comunique, os dados trocados com elas, e estabelece a função global do sistema na forma de um único processo com o nome dessa função.

Cada um dos outros níveis compõe-se de DFDs que detalham os processos do nível anterior. Por fim, os processos suficientemente simples, ou "primitivos", se decompõem em PSPECs. O último nível é composto inteiramente de PSPECs. Nos outros níveis também pode haver PSPECs; e normalmente há, embora isto não esteja evidenciado na Figura 2.1.

2.1.1.2 - O MODELO DE CONTROLE

O Modelo de Controle compõe-se de CFDs e CSPECs. Seu propósito geral é determinar o que o Modelo de Processos deve fazer sob dadas condições externas ou internas ou sob dados modos de operação do sistema. Sua estrutura é dependente da estrutura do modelo de processos.

Dentro do Modelo de Processos só existem decisões nos níveis mais baixos: as decisões descritas dentro das PSPECs. Já no Modelo de Controle existem, a princípio, decisões em qualquer nível, mas tipicamente elas existem nos níveis mais altos do modelo: elas determinam grandes mudanças no modo de operação do sistema e envolvem ativação e desativação de grandes grupos de processos. A estrutura de controle também é responsável por receber informação sobre os estados de sistemas externos para tomar suas decisões, e também transmitir para sistemas externos informação de estado do próprio sistema. Os

componentes do Modelo de Controle são o Diagrama de Contexto de Controle (CCD), os CFDs e as CSPECs.

2.1.1.2.1 - DIAGRAMAS DE CONTROLE

O CCD, tal como o DCD, mostra a interação entre a função global do sistema e suas entidades externas; só que se trata de interação de controle, ou seja, comandos de controle e sinalização de estado.

Os CFDs, tal como os DFDs, mostram por onde passam os fluxos; no caso, fluxos de controle. Os processos do CFD são os mesmos que os do DFD correspondente.

Há nos CFDs um componente que não existe nos DFDs: as "barras de CSPEC". Note-as no CFD da Figura 2.2. Estas barras representam a interface entre o CFD e sua CSPEC: fluxos que entram nas barras são entradas da CSPEC; fluxos que saem das barras são saídas da CSPEC. Todas as barras de um mesmo CFD se referem à sua única CSPEC associada.

2.1.1.2.2 - ESPECIFICAÇÃO DE CONTROLE (CSPEC)

As CSPECs especificam os requisitos de controle do sistema. Seu propósito é análogo ao das PSPECs -- mostrar precisamente como suas saídas são geradas a partir das entradas -- mas elas fazem isto usando tabelas de decisão e/ou diagramas de transição de estado, em vez de Portugês Estruturado, equações, etc. Mas há uma diferença importante entre o "papel" de PSPECs e CSPECs na estrutura do modelo. Toda PSPEC é decomposição de um processo primitivo, processo este que aparece no seu DFD pai e no CFD correspondente. Já as CSPECs não são decomposição de processo algum. Uma CSPEC é uma entidade independente, que "liga" um CFD a um DFD; especificando os requisitos de controle para todo o DFD associado a ela (ou seja, para o processo pai do DFD associado). Suas entradas são fluxos de controle do CFD que entram nas barras. Já suas saídas são fluxos de controle de dois tipos:

1) Ativadores de Processos, que por convenção não são mostrados no CFD, e

2) outros fluxos de controle que aparecem no CFD saindo das barras da CSPEC.

2.1.1.2.3 - CONDIÇÕES DE DADOS

Há ainda um último aspecto relativo a controle que se refere às PSPECs. Elas podem gerar fluxos de controle baseado em testes feitos sobre os dados que elas processam. Esses fluxos de controle são chamados de "Condições de Dados" (Data Conditions) e devem balancear com o CFD correspondente ao DFD pai da PSPEC. Este balanceamento é mostrado na Figura 2.1 pela seta que vai da PSPEC ao CFD.

2.1.1.3 - A ESPECIFICAÇÃO DE TEMPOS DE RESPOSTA

Esta especificação mostra os requisitos de tempo decorrido entre cada entrada do sistema e sua saída correspondente. Ela lista eventos que são detetados por meio das entradas para o sistema (seus "estímulos"), os eventos correspondentes que se requer que ocorram nas saídas do sistema (as "respostas" aos estímulos) e as restrições de tempo em que o sistema deverá gerar estas respostas.

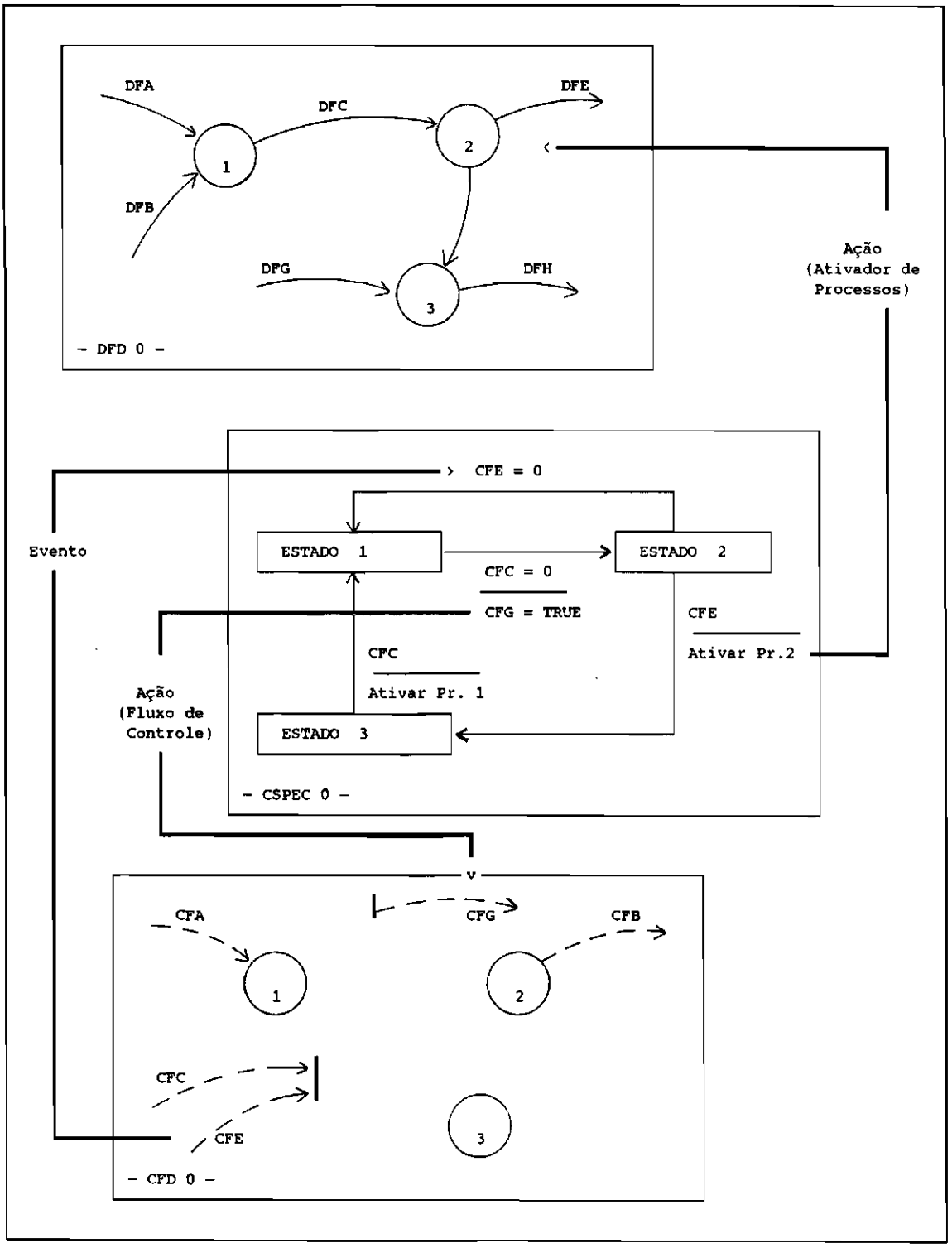


Fig. 2.2 - DFD, CSPEC, CFD e seus relacionamentos.

2.1.1.4 - O DICIONÁRIO DE REQUISITOS (RD)

O RD tem um papel vital no modelo. Ele contém definições precisas de todos os fluxos de controle e de dados em todos os níveis de decomposição destes fluxos (os fluxos também se decompõem, paralelamente aos processos ao longo dos níveis).

O RD, como mostrado pelas setas na Figura 2.1, deve balancear-se com as PSPECs, DFDs e CFDs.

A composição e o significado exatos dos fluxos nos diagramas nunca podem ser inferidos somente a partir dos diagramas; isto está no RD.

2.1.1.4.1 - ESTRUTURA DO RD

O RD é uma lista de todos os fluxos, depósitos e seus componentes. Neste trabalho vamos chamar qualquer um destes de "item do dicionário de requisitos" (RDI - Requirements Dictionary Item). Isso é feito aqui para tratar o dicionário de maneira uniforme, não faz parte da nomenclatura descrita em (Hatley e Pirbhai, 1987). Assim, um RDI pode ser:

1) Fluxo,

2) Depósito, ou

3) Estrutura contida em outro RDI.

A forma geral do RD é uma seqüência de RDIs em ordem alfabética. Para usar a notação do próprio RD, poderíamos representar assim:

RD = { RDI }

Cada RDI é da forma:

RDI: <Nome> = <Atributos>

onde os Atributos são definidos nas convenções sobre o RD. Esses atributos são especificados usando a simbologia definida na notação sobre o RD.

2.1.2 - NOTAÇÃO

2.1.2.1 - NOTAÇÃO DO DFD E CFD

A notação para DFDs e CFDs é mostrada nas Figuras de 2.3 a 2.8. Os significados de cada símbolo são:

a) Entidade Externa

É qualquer entidade (outros sistemas, usuários, operadores, etc.) que não faz parte do sistema sendo modelado e que interage com ele. Não interessa como processa, só como interage: a especificação de sua interface com o sistema e os eventos relacionados a ela.

b) Processo

Um processo deve representar uma única função naquele nível de abstração em que é usado. Seu número é dado pela convenção de numeração.

c) Fluxo de Dados

São dados transmitidos entre dois processos, entre processos e depósitos , ou entre o sistema e seu ambiente. Deve ser pensado como um tubo pelo qual fluem dados de composição conhecida, que pode consistir de um único elemento ou um grupo de elementos.

d) Fluxo de Controle

São informações de controle (comandos ou sinais) sendo transmitidas. Deve ser pensado como um tubo pelo qual fluem informações de controle de composição conhecida, que pode consistir de um único elemento ou um grupo de elementos.

e) Barra de CSPEC

Interface com a CSPEC, onde é processado controle. Todas as barras de um CFD referem-se à única CSPEC correspondente.

f) Depósito (de dados ou de controle)

Representa armazenamento do conteúdo de fluxos para uso posterior.

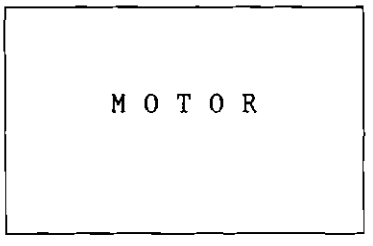


Fig. 2.3 - Entidade Externa: retângulo com um nome dentro.

Fig. 2.4 - Processo: Um círculo com um nome e número.

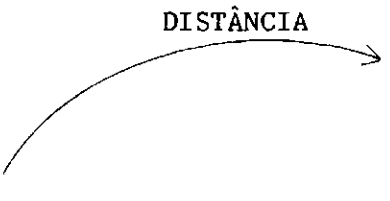


Fig. 2.5 - Fluxo de Dados: um arco em linha cheia com um nome.

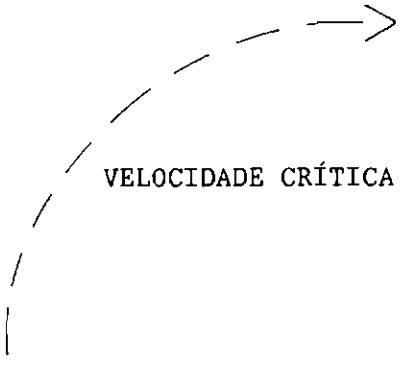


Fig. 2.6 - Fluxo de Controle: Um arco em linha tracejada com um nome.

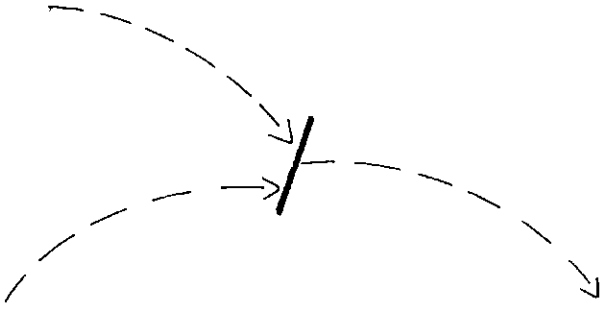


Fig. 2.7 - Barra de CSPEC: Uma barra curta e sem nome.

PARÂMETROS DE CALIBRAÇÃO

Fig. 2.8 - Depósito: Duas linhas paralelas contendo um nome.

2.1.2.2 - NOTAÇÃO DO DTE

O Diagrama de Transição de Estados (DTE) é uma das representações de máquinas seqüenciais usadas no método Hatley. Outras representações usadas são: Tabela de Transição de Estados e Matriz de Transição de Estados. Embora as três representações sejam bem conhecidas da teoria de autômatos, é necessário colocar aqui a notação usada para o DTE, que difere da notação comumente usada nos textos sobre teoria de autômatos. Os símbolos usados no DTE são mostrados nas Figuras de 2.9 a 2.11. Seus significados são:

a) Estado

É um modo de comportamento externamente observável. Aplica-se a todo o DFD associado ao DTE. É representado por um retângulo, geralmente alongado na horizontal, contendo o nome do estado que ele representa.

b) Transição

É uma mudança de estado, ou seja, uma mudança de um modo de comportamento para outro. Ocorre sempre em tempo nulo. Representa-se com uma seta orientada indo de um estado para outro qualquer do DTE. A seta pode ser em linha reta ou em trechos retos perpendiculares entre si (ver Figura 2.11).

c) Condição (ou evento)

É de sinais de entrada para a máquina de estados representada pelo DTE. Um evento pode fazer a máquina mudar de estado, produzir saídas (ações), ou ambas as coisas. É representado por seu nome como um rótulo para o arco da transição que ele causa (ver Figura 2.11).

d) Ação

É uma saída da máquina de estados, causada por um evento. É mostrada pelo seu nome ao lado do evento que a causa. São as ações causadas pelo evento que vão efetivar a mudança de comportamento representada pela transição. Representa-se pelo seu nome, adjacente ao evento que a causa, separando-se os dois por uma linha.

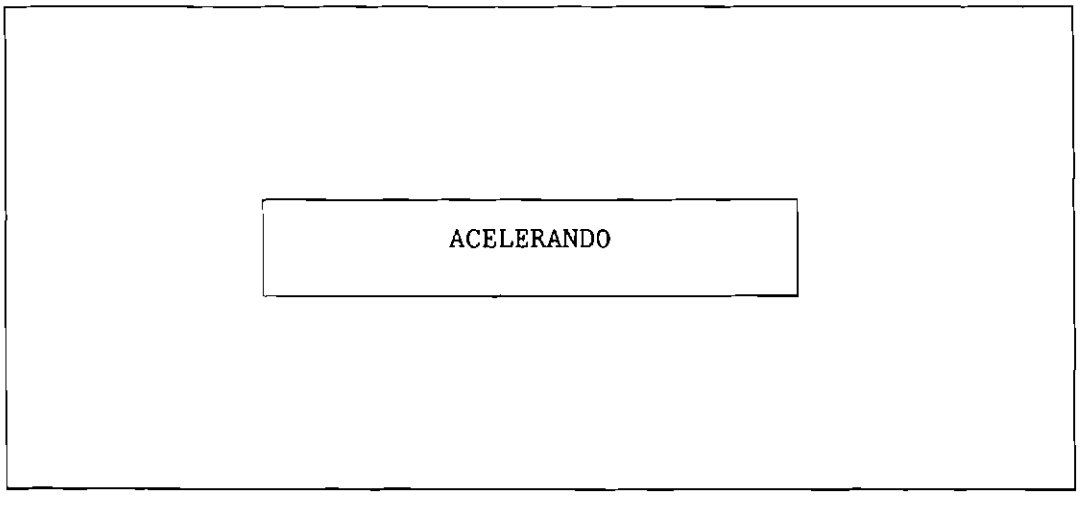


Fig. 2.9 - Estado: Um retângulo com um nome dentro.

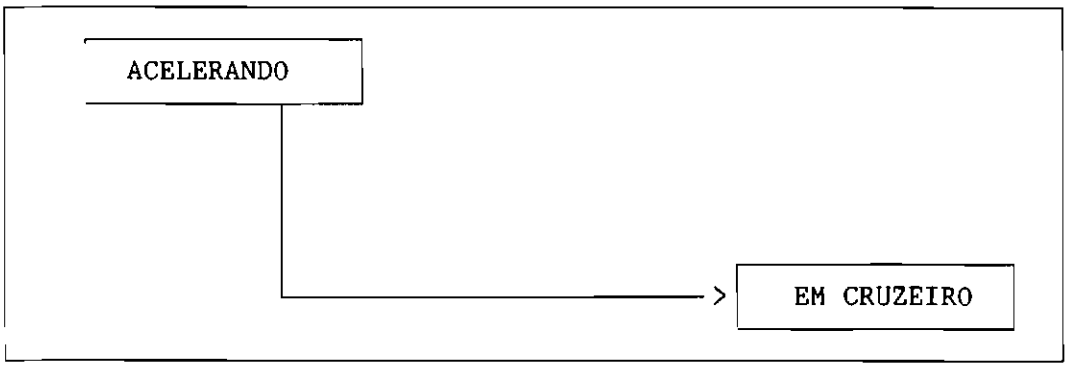


Fig. 2.10 - Transição: Uma seta orientada.

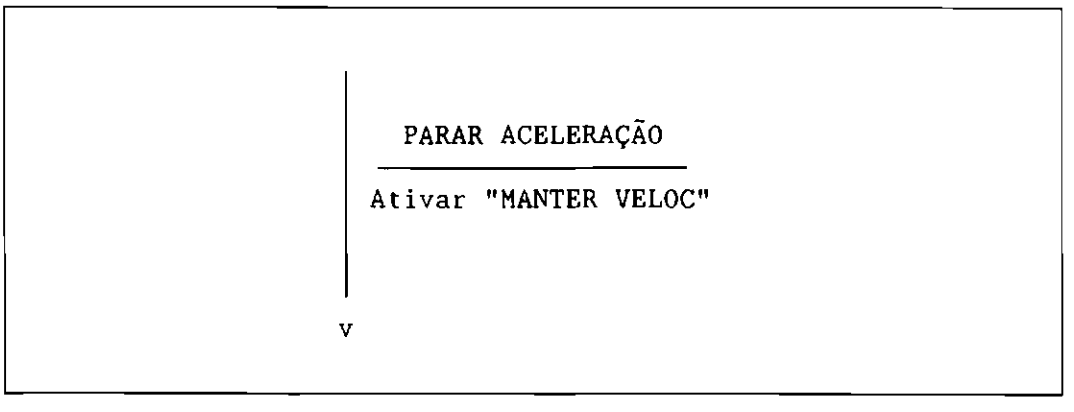


Fig. 2.11 - Transição, condição e ação.

2.1.2.3 - NOTAÇÃO DO RD

A notação para especificar os atributos (composição, significado, etc.) dos ítens do dicionário de requisitos é a da simbologia da Tabela 2.2.

TABELA 2.2

NOTAÇÃO DO DICIONÁRIO DE REQUISITOS

SÍMBOLO	SIGNIFICADO	DESCRIÇÃO
=	é especificado por	Indica que o RDI cujo nome está à sua esquerda é especificado pelos atributos à sua direita.
+	junto com	Indica que ambos os ítens à esquerda e à direita vão juntos na Composição do RDI. NÃO IMPLICA ORDEM (é como a união de conjuntos).
{...}	iterações de	Um item dentro de chaves pode ocorrer qualquer número de vezes (zero ou mais) em cada ocorrência do RDI. Para indicar faixas para o número de vezes, usar índices: m{ }n : de m a n iterações { }n : de 0 a n iterações m{ } : m ou mais iterações
[.....]	selecione um de	(Dois ou mais ítens entre colchetes, separados por barras). Indica que um e somente um item aparece em cada ocorrência do RDI (é a função booleana "OU exclusivo").
(...)	opcional	É o mesmo que { }1. Por ser freqüente, é preferido para indicar que o item pode ocorrer ou não no RDI.
"..."	literal	Símbolos entre aspas constituem literalmente o RDI, em oposição ao nome de outro RDI. É freqüentemente usado para mensagens a serem exibidas. Pode ser usado na Composição ou na lista de Nomes dos valores.
\...\	significado	Delimita texto que especifica o Significado atribuído ao Nome do RDI. DEVE SER PRECISO.
...	comentário	Delimita texto que não é parte formal da definição do RDI.

2.1.3 - CONVENÇÕES

2.1.3.1 - NUMERAÇÃO E NOMES DOS DIAGRAMAS

A indexação na estrutura do modelo é feita segundo as seguintes convenções para numeração e nomes dos diagramas:

a) Um diagrama de fluxo de dados deve ter nome e número igual ao processo pai que ele descreve. Os números dos processos desse DFD devem começar com o número do processo pai, acrescentando-se .1, .2, .3, etc. (ver Figura 2.1).

b) O processo único de um diagrama de contexto recebe o número 0, e assim o DFD do nível 1 também é numerado com 0.

c) O número e nome da PSPEC devem ser iguais aos do processo que ela descreve.

d) Os componentes (processos e depósitos) em um CFD, e o próprio CFD, devem receber nome e número idênticos aos do seu DFD correspondente.

e) Cada CSPEC tem o mesmo nome e número que o DFD e o CFD com os quais é associada.

f) Uma CSPEC pode ter mais de uma página no modelo. Cada folha leva o nome e número da CSPEC como um todo e, em adição, segue a conhecida convenção "folha M de N".

2.1.3.2 - CONVENÇÕES SOBRE DFDs

As convenções abaixo permitem interpretar os requisitos expressos nos DFDs.

a) O Modelo de Processamento é altamente idealizado, no sentido de que ele é considerado ser disparado por dados e infinitamente

rápido: sempre que dados, suficientes para um certo processo desempenhar sua tarefa, aparecem em suas entradas ele os processa e produz as saídas instantaneamente.

b) Quando um processo é ativado, ele exibe seu comportamento implícito: disparado por dados e infinitamente rápido. Quando um processo é desativado, o modelo comporta-se diferentemente, o processo não faz nada, e suas saídas são nulas (exatamente como elas são quando há entradas insuficientes para que ele opere).

c) O nome de um processo é da forma:

<verbo> + <objeto (opcional)>

2.1.3.3 - CONVENÇÕES SOBRE ATIVADORES DE PROCESSO

a) Ativadores de processos são saídas (fluxos) de controle de CSPECs que têm o propósito principal de ativar e desativar processos. Eles têm portanto somente dois valores: "Ativar" e "Desativar" (ou 0 e 1). Ativadores de processos são definidos dentro das CSPECs, onde eles recebem os nomes dos processos que eles controlam. As CSPECs determinam se seus ativadores de processos são "Ativar" ou "Desativar", e os processos correspondentes são ativados ou desativados como conseqüência. Os ativadores de processos NÃO são mostrados nem nos DFDs nem nos CFDs, pois é fácil achar esses processos já que estão todos no DFD adjacente.

b) Um processo está ativo somente quando ele e todos os seus ancestrais estão ativos. Em outras palavras, quando um processo é desativado, todos seus descendentes são também desativados.

c) Um processo SEM um ativador é considerado estar ativo sempre que todos seus ancestrais estejam ativos.

d) TABELAS DE ATIVAÇÃO DE PROCESSOS -- São tabelas de decisão que são usadas para ativar processos. Em vez de usar apenas os valores binários usuais de saída 0 e 1, algumas linhas na tabela contêm 0, 1, 2, 3, etc. O zero tem o significado usual: Falso ou Desativado. A seqüência 1, 2, 3 indica que quando o critério de entrada para aquela linha se torna "Verdadeiro", os processos são ativados naquela seqüência. Esse recurso é usado quando é importante que certas saídas de processos sejam disponíveis antes que outros comecem suas tarefas, mas quando isto não está claramente implicado pelos fluxos de dados.

2.1.3.4 - CONVENÇÕES SOBRE FLUXOS

As convenções listadas abaixo permitem interpretar os requisitos, que são de interfaceamento, expressos por fluxos de dados e de controle.

a) Um fluxo pode ser de dados, de controle ou de dados e controle ao mesmo tempo. No caso de ser de dados e controle, ele aparece tanto no CFD quanto no DFD. O que determina o tipo é o uso do fluxo.

b) Fluxos nos DFDs ou CFDs podem ser agrupados ou decompostos, usando-se ramificações, segundo as convenções da tabela 2.3. Note que os ramos devem balancear com o tronco, segundo as definições no dicionário de requisitos. Conforme a necessidade, pode-se remificar sucessivamente um fluxo, criando "árvores".

c) O nome de um fluxo de dados é da forma:

<substantivo> <compl. adjetivo (opcional)>

d) Fluxos de dados podem ter ou valores discretos ou valores contínuos.

e) Salvo se especificado numa PSPEC, todo fluxo de dados é continuamente disponível no tempo; exceto, naturalmente, os fluxos de dados de/para depósitos (ver Seção 2.1.3.5.c).

f) O nome de um fluxo de controle é da forma:

- Se for de comando:

<verbo no infinitivo> <objeto (opcional)>

- Se for sinal (de evento ou estado):

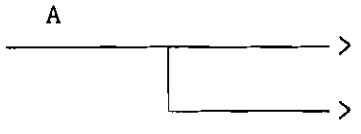
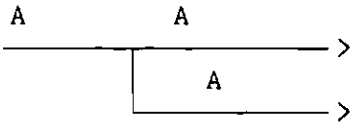
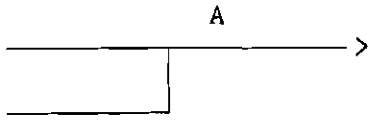
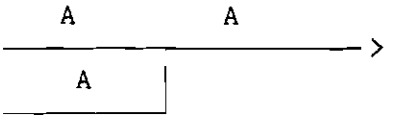
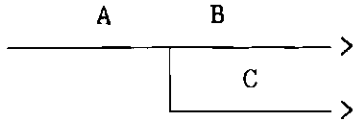
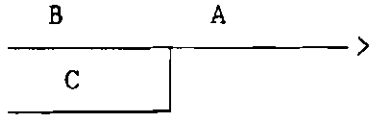
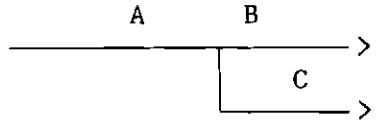
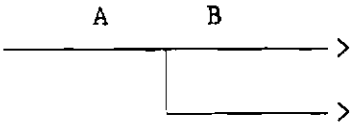
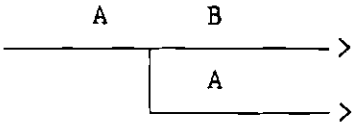
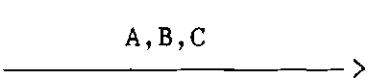
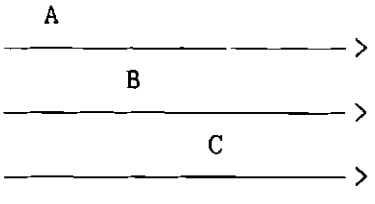
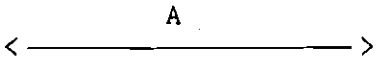
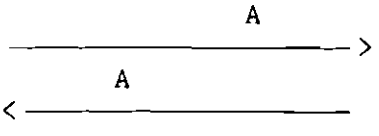
<substantivo> <compl. adjetivo ou
verbo no particípio passado>

g) Todo fluxo de controle é discreto no tempo, isto é, sua duração é instantânea.

h) Fluxos de controle primitivos podem ser binários ou, no caso usual, ter mais de dois valores.

TABELA 2.3

CONVENÇÕES PARA AGRUPAR E DECOMPOR FLUXOS

CONVENÇÃO	SIGNIFICADO
	
	
	B e C são sub-conjuntos de A. A intersecção entre B e C pode ser vazia ou não.
	Idem acima
	Pode ser: A = [B | C] , ou A = (B) | (C) , ou A = B | C
	
	
	

2.1.3.5 - CONVENÇÕES SOBRE TEMPO

As convenções listadas abaixo referem-se ao uso e interpretação da noção de tempo, decorrido ou absoluto, no modelo de requisitos.

a) A informação de tempo é universalmente disponível a todas as PSPECs e CSPECs sem aparecer sob a forma de fluxos. Assim, é sempre aceitável referenciar tempo absoluto ou tempo relativo (isto é, tempo decorrido desde que um evento percebido pelo processo tenha ocorrido).

   Já que o tempo não aparece como um fluxo, e portanto não é definido no dicionário, deve-se ser cuidadoso para especificar completa e exatamente que referência de tempo e que unidades está usando (na PSPEC). Se há necessidade de informação de tempo relativo a algum evento NÃO conhecido do processo, então esta medida específica de tempo deve ser representada como um fluxo, saindo do processo que tem a necessária referência de tempo.

b) RISCO DE CAIR EM PROJETO: O tempo nunca deve ser usado para definir tempos internos da eventual implementação, p. ex., coisas tais como freqüência de chamada.

c) Geralmente nos requisitos, fluxos são considerados estarem presentes continuamente no tempo. Em outras palavras, quando um processo gera um valor específico de um fluxo, considera-se que ele permanece até ser mudado ou desativado. Assim, um processo que recebe esse fluxo pode lê-lo qualquer número de vezes antes que ele mude. Em alguns casos, contudo, necessitamos de um fluxo que seja transiente no tempo; isto é, um fluxo que esteja normalmente em um estado nulo mas que assume seu valor desejado momentaneamente e depois retorna ao estado nulo. Quando esta característica é requerida, isto é indicado na PSPEC pela palavra 'emitir'. O uso de qualquer outra palavra, como

'calcular' ou 'produzir' estará indicando o caso mais usual contínuo no tempo.

2.1.3.6 - CONVENÇÕES SOBRE DEPÓSITOS

As convenções listadas abaixo permitem interpretar os requisitos expressos por depósitos de dados e depósitos de controle.

a) Depósitos de dados podem ser só para leitura (constantes), para leitura e escrita (dados produzidos por um processo para uso posterior), ou ambos. Depósitos só para escrita não têm utilidade, portanto não são usados.

b) Um dado depósito aparece somente em um DFD. Se suas saídas ou entradas são usadas em outros diagramas, elas fluem para lá como fluxos de dados da maneira usual. Contudo, naquele DFD único, o depósito pode aparecer mais de uma vez, para simplificar o diagrama.

c) Se por alguma razão há necessidade de leitura destrutiva, o processo que recebe o fluxo vindo de um depósito pode escrever '0' ou 'nulo' de volta no depósito.

d) Em todos os outros aspectos, depósitos são tratados exatamente como fluxos:

- Como fluxos compostos, os nomes dos depósitos são os nomes de seus conteúdos, e devem ser definidos no dicionário.

- Fluxos saindo de depósitos: Um fluxo sem nome herda o nome do depósito, e todo o conteúdo do depósito flui por ele. Um fluxo rotulado tem que ser um subconjunto da informação armazenada.

- Fluxos entrando em depósitos seguem as mesmas regras para os que saem.

Exceção: quando uma parte do depósito é somente para leitura, fluxos entrando têm que ser rotulados com o nome do subconjunto do conteúdo do depósito que eles carregam.

e) Depósitos podem ser usados bastante liberalmente no modelo de requisitos, sempre que se sentir que eles esclarecerão o significado. Segundo Hatley e Pirbhai (1987, p. 150), "em sistemas computadorizados eles nunca implicam em requisitos artificiais, pois todos os fluxos de dados e de controle são armazenados em algum meio ou dispositivo de memória, de algum modo".

f) Depósitos de controle têm exatamente o mesmo papel nos CFDs que os de dados nos DFDs: eles armazenam fluxos de controle depois que suas fontes cessaram de produzi-los. Um dado depósito pode ser único de um DFD, único de um CFD, ou usado tanto em um DFD quanto em seu CFD associado.

g) As convenções para depósitos de dados se aplicam também a depósitos de controle:

- Eles retêm seu conteúdo até ser sobreescrito.

- Fluxos não rotulados de e para o depósito levam toda a informação do depósito.

- Fluxos levando subgrupos da informação armazenada são rotulados com os nomes dos subgrupos.

- Um dado depósito somente aparece em um CFD, mas pode aparecer mais de uma vez nele.

2.1.3.7 - CONVENÇÕES SOBRE CFDs

As convenções listadas abaixo permitem interpretar os requisitos expressos nos CFDs.

a) Os processos em um CFD "não" representam processamento dos fluxos de controle entrando neles, isto é feito pelas CSPECs; nem representam estados do sistema, estes estão contidos dentro das CSPECs; e finalmente, os fluxos de controle entrando nos processos não ativam ou desativam os processos, isto é feito por Ativadores de Processos, também dentro das CSPECs. Os processos dm um CFD são os mesmos que os do DFD correspondente. Apenas se desenha os fluxos de controle num diagrama separado.

b) Se não for ficar emaranhado demais, pode-se sobrepor o CFD ao DFD correspondente. Contudo, sempre se considera que são dois diagramas distintos.

c) Pode haver qualquer quantidade de barras de CSPEC em um CFD, mas todas elas referenciam toda a CSPEC associada (isto é, nenhuma delas é associada com qualquer objeto ou folha específica da CSPEC).

2.1.3.8 - CONVENÇÕES SOBRE DICIONÁRIO DE REQUISITOS

Dentro da estrutura do RD, definida na Seção 2.1.1.4, os atributos de um item do dicionário (RDI) são especificados conforme as convenções a seguir. Cada item do dicionário de requisitos possui quatro características:

1) Pode ser composto ou primitivo,

2) Pode ser interno ou externo ao sistema,

3) Pode ser de dados ou de controle,

4) Pode ter valores discretos ou contínuos.

Essas características são definidas como segue:

a) Primitivo ou composto

Um RDI primitivo é um RDI de uma unidade de informação indivisível.

Um RDI composto é uma coleção de outros RDIs compostos e/ou primitivos. No caso de fluxo, pensar neles como fluxos separados fluindo juntos pelo mesmo tubo. Seus membros não são presos entre si, e podem ser separados em qualquer ponto do trajeto.

b) Interno ou externo

Um RDI interno existe somente dentro do sistema. Não são, nem fazem parte de fluxos dos diagramas de contexto.

RDIs externos são fluxos dos diagramas de contexto ou componentes deles. São a forma de comunicação do sistema com seu ambiente; por isso se aplicam os atributos de Faixa, Resolução e Freqüência.

c) Dados ou controle

RDIs de Dados são aqueles usados em cálculos e algoritmos dentro de processos.

RDIs de Controle são aqueles usados para modificar o modo de operação do sistema (ou partes dele); têm sempre valores discretos.

d) Contínuo ou discreto (valores, não no tempo)

RDIs Contínuos são os que podem adotar um número ilimitado de valores, com variação contínua de um valor para outro.

Assim, pequenas mudanças de valor têm geralmente efeito insignificante sobre o sistema.

RDIs Discretos são os que têm um número finito de valores; e são todos igualmente significativos. Assim, qualquer mudança de valor geralmente tem efeito significativo sobre o sistema.

A Tabela 2.4 resume todos os tipos possíveis de RDIs e os atributos aplicáveis a cada tipo. Ela deve ser lida como uma tabela de decisão, em que combinações de tipos de RDI determinam os atributos aplicáveis. As várias combinações de tipos de RDI possíveis pela Tabela 2.4 e os correspondentes atributos obrigatórios são:

a) PARA FLUXO:

1) Primitivo, interno, de dados, contínuo:
Nome, Significado, Unidades.

2) Primitivo, externo, de dados, contínuo:
Nome, Significado, Unidades, Faixa, Resolução, Freqüência.

3) Primitivo, externo, discreto:
Nome, Significado, Nomes dos Valores, Freqüência.

4) Primitivo, interno, discreto:
Nome, Significado, Nomes dos Valores.

5) Composto:
Nome, Composição.

b) PARA DEPÓSITO:

É importante ressaltar que um depósito é sempre da forma: m{estrutura}n; ou seja, uma repetição de estruturas (de dados, ou de informação de controle). Às vezes ocorre ter tamanho

unitário, isto é, 0{estrutura}1. Assim, um depósito é sempre especificado como

<Nome do depósito> = {estrutura}n

e, para a estrutura que se repete, tem-se os seguintes casos possíveis:

1) Primitivo, interno, de dados, contínuo:
   Nome, Significado, Unidades.

2) Primitivo, interno, discreto:
   Nome, Significado, Nomes dos Valores.

3) Composto:
   Nome, Composição.

Definidos os casos possíveis de RDIs, a seguir passamos a definir cada atributo.

a) Nome

É o identificador principal do RDI; e sua única identificação nos diagramas. Todos os Nomes devem ser únicos e ilustrativos.

b) Significado

Descreve o que representa o Nome dado à RDI. O significado deve definir precisamente o papel que o item referido pelo Nome desempenha no sistema. Por exemplo, compare estas duas definições para o Nome POSIÇÃO DO AVIÃO:

- Medida da posição de um avião, relativa ao centro da Terra.

- É a latitude, longitude e altura do avião.

Em princípio, pode-se pensar que são equivalentes, e que qualquer das duas serve para o RD. Mas elas são equivalentes apenas para determinar a posição do avião; para uso no modelo elas não são. Veja porque: a segunda é dada em termos de composição, confunde a composição do dado com sua interpretação, e é desnecessariamente restritiva. Se mais tarde, na modelagem, for alterada a composição do dado, seu significado fica sem sentido, enquanto que a primeira definição independe da composição escolhida.

c) Composição (Backus-Naur Form, BNF)

Especifica os componentes do RDI e sua estrutura; i.e., como esses componentes podem se agrupar, usando os símbolos de composição da notação para o RD. Importante, deve ser válida para todas as ocorrências do RDI.

d) Unidade

Unidade de medida física (kg, joules, etc.).

e) Faixa

Os limites dentro dos quais um RDI contínuo e primitivo pode existir. (OBS.: Só se aplica a fluxos externos, para os internos é variável de projeto).

f) Resolução

É o menor incremento requerido na magnitude de um valor, para RDI contínuo, de dados, primitivo. (OBS.: Idem à de Faixa).

g) Freqüência

É a freqüência em que é requerido que o fluxo externo ocorra. (OBS.: Idem à de Faixa).

h) Nomes dos valores

Lista dos valores literais de um RDI primitivo de valores discretos.

i) Número de valores

Quantidade de nomes de valores. (Não é estritamente necessário, pois é derivável da lista de valores, mas é muito útil para verificações durante o desenvolvimento).

j) Classe

Atributo que indica no RD se o RDI é de Dados (D),de Controle (C) ou de ambos (D/C). (OBS.: Também não é necessário à modelagem mas pode ser útil à consulta ao RD.)

k) Comentários

Texto que não é parte formal da definição do RDI. Usar para qualquer informação adicional (ou lembretes) que possa ser útil. Evitar usar para apenas completar o significado. Pode-se, no entanto usar para esclarecê-lo.

TABELA 2.4

REFERÊNCIA PARA ESPECIFICAÇÃO DE RDIs

<table>
<tr><th colspan="4">T I P O D E R D I</th><th colspan="3" rowspan="2">A T R I B U T O S</th></tr>
<tr><th>Composto/
Primitivo</th><th>Interno/
Externo</th><th>Dados /
Controle</th><th>Discreto/
Contínuo</th></tr>
<tr><td rowspan="5">Primitivo</td><td>Interno/
Externo</td><td>Controle</td><td rowspan="4">Contínuo</td><td colspan="3">Inválido - fluxos de controle não podem ter valores contínuos</td></tr>
<tr><td>Interno</td><td rowspan="3">Dados</td><td>Unidades</td><td>—</td><td rowspan="4">Nome
Significado
(Classe(D/C))
(Comentários)</td></tr>
<tr><td rowspan="2">Externo
*</td><td>Faixa
Resolução</td><td rowspan="2">Freq.
*</td></tr>
<tr><td rowspan="2">Dados /
Controle</td><td rowspan="2">Discreto</td><td>Nomes dos
Valores</td></tr>
<tr><td>Interno</td><td>(No. de
Valores)</td><td>—</td></tr>
<tr><td>Composto</td><td>Interno/
Externo</td><td>Dados /
Controle</td><td>Discreto/
Contínuo</td><td colspan="2">Composição (BNF)</td><td>Nome
(Significado)
(Classe(D/C))
(Comentários)</td></tr>
</table>

LEGENDA: * : O campo não se aplica a depósitos.

( ) : Atributo não obrigatório.

2.1.3.9 - CONVENÇÕES SOBRE CSPEC

As convenções listadas abaixo permitem interpretar os requisitos de processamento de informações de controle, expressos na especificação de controle (CSPEC).

a) Suas entradas são fluxos de controle, e suas saídas são ativadores de processo e fluxos de controle. Entradas para uma CSPEC vêm do CFD associado via os símbolos de barra, suas saídas ou vão de volta para o mesmo CFD, também via as barras, ou elas controlam processos no DFD associado.

b) Quaisquer processos que a CSPEC controla estão em seu DFD associado.

c) Já que uma CSPEC de várias folhas pode ficar muito complexa internamente, as seguintes convenções são usadas para organizá-la:

   - Cada folha leva: CSPEC n, folha p/q "Nome da CSPEC" "Nome da folha"

   - Se a CSPEC é composta, os diagramas são arranjados na seqüência: lógica de eventos, máquina seqüencial, e lógica de ações.

   - Se a CSPEC tem mais de três ou quatro folhas (e elas são às vezes muito maiores), então é incluído um guia de usuário descrevendo o arranjo geral da CSPEC, e listando as folhas onde cada entrada, cada saída e cada sinal interno aparece (exemplo na Figura 2.14).

d) Há dois modelos bem conhecidos de máquina seqüencial: o de Mealy e o de Moore. No modelo Mealy, as ações são associadas com as transições. No modelo Moore, as ações são funções somente do estado atual, o que é um arranjo mais simples, mas

menos flexível. No modelo de requisitos usa-se uma abordagem híbrida: a convenção usual é dirigir as ações de saída a partir das transições, mas quando for conveniente elas podem ser conduzidas também pelos estados.

e) Para máquina de estados, uma convenção importante usada neste modelo é que as ações, embora sejam associadas com as transições, que são instantâneas por natureza, são consideradas em efeito até que ocorra a próxima transição. A ativação de processos é explícita, por uma ação; mas a desativação é feita implicitamente, por se "sair" do estado em que estavam ativos.

f) A Figura 2.12 mostra um esquema genérico para uma CSPEC que só use máquina de estados combinatorial.

g) A Figura 2.13 mostra um esquema genérico para uma CSPEC que use máquina de estados seqüencial. Note que no caso de só haver a máquina seqüencial, os eventos são as entradas de fluxos de controle da CSPEC, e as ações da máquina são os ativadores de processo e as saídas de fluxos de controle.

h) Deve-se escolher estados, eventos e ações que sejam familiares aos usuário, e usar nomes que os usuários normalmente usem. Para serem familiares aos usuários, devem estar no ambiente do sistema, que é o que o usuário conhece.

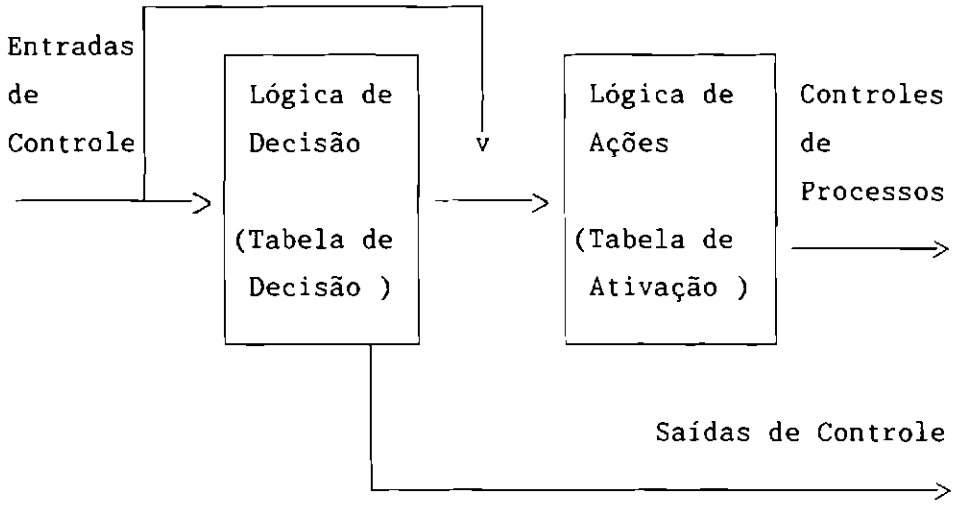


Fig. 2.12 - CSPEC genérica combinatorial.

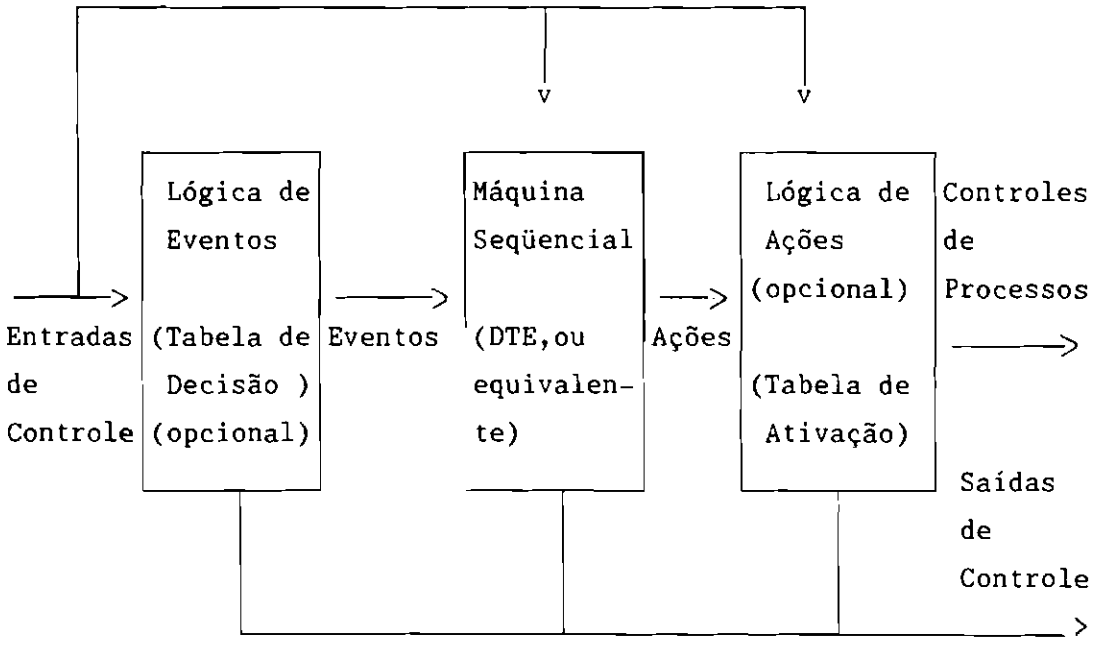


Fig. 2.13 - CSPEC genérica seqüencial.

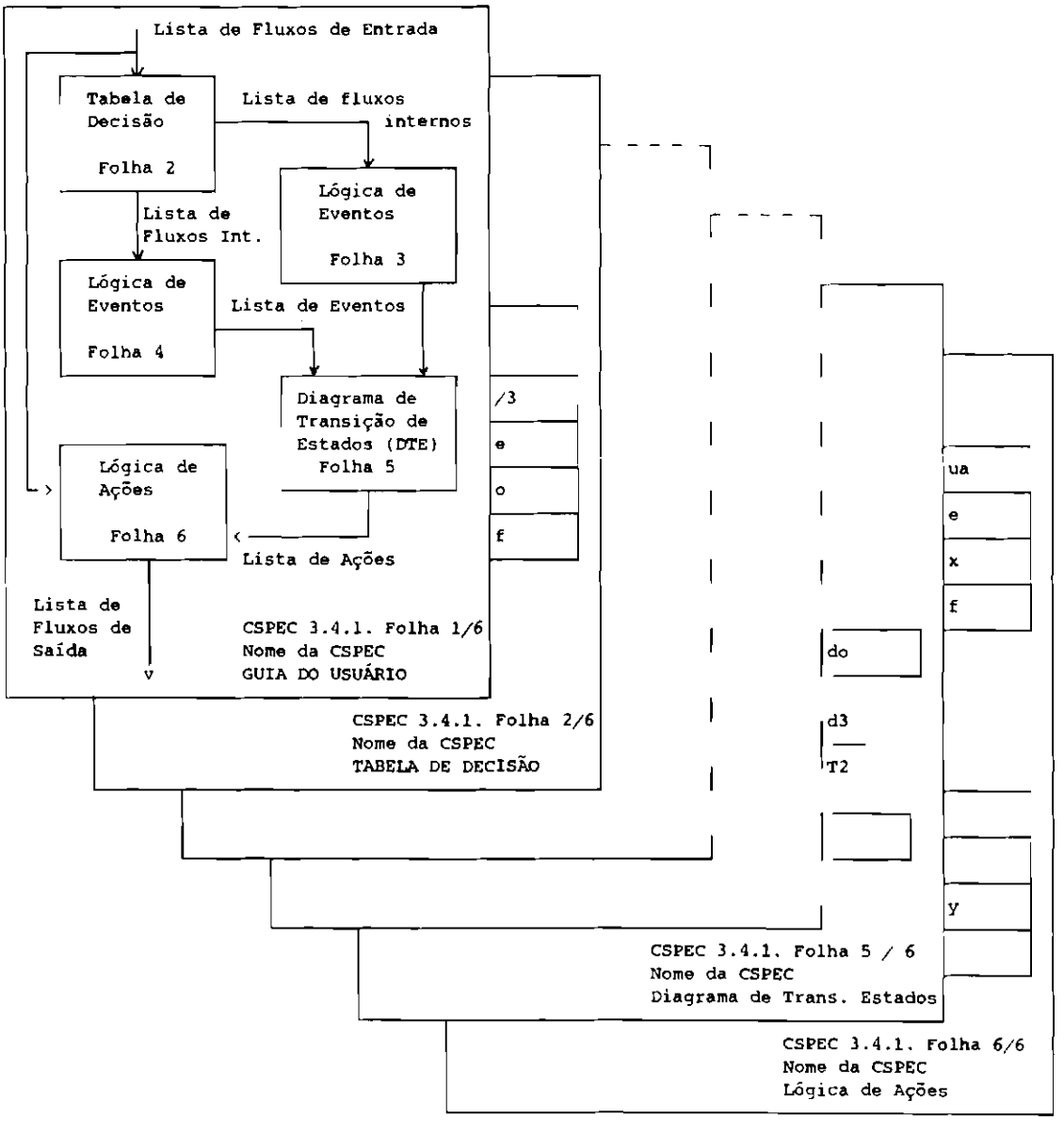


Fig. 2.14 - Organização de CSPEC complexa.

2.1.3.10 - CONVENÇÕES SOBRE PSPECs

As convenções listadas abaixo permitem interpretar os requisitos de processamento de dados expressos nas PSPECs.

a) Já que o tempo não aparece como um fluxo, e portanto não é definido no dicionário, deve-se ser cuidadoso para especificar completa e exatamente que referência de tempo e que unidades está usando.

b) Um processo primitivo só tem como entradas fluxos de dados. Já suas saídas podem ser fluxos de dados ou condições de dados (ver Seção 2.1.1.2.3). Isto impõe restrições ao tamanho de algumas PSPECs: Como fluxo de controle não entra em processo primitivo (só em não primitivo ou em barra de CSPEC), um processo pequeno que recebe fluxo de controle deve ser decomposto em um DFD, para poder ter seu CFD e CSPEC.

c) Sobre tabelas:

- Usar sempre que possível; são muito expressivas.

- Pode-se usar qualquer tipo, mesmo tabela de decisão ou matriz de transição de estados (esta, apenas para controle local).

- Única proibida: Tabela de Ativação de Processos (só usada em CSPEC).

d) Comentários na PSPEC devem ser claramente identificados (com asteriscos ou qualquer outra identificação), pois não fazem parte formal da PSPEC (não são verificados).

e) Inglês Estruturado: certos tipos de palavras são preferidas, e estruturas gramaticais são limitadas a algumas muito simples. Esta simplicidade, junto com sua concisão, assegura que as especificações sejam não ambíguas. Os tipos de palavras usadas

são:

1) verbos de comando aplicados a um objeto (verbos transitivos e imperativos),

2) nomes de fluxos de dados (em maiúsculas),

3) preposições e conjunções necessárias para mostrar relacionamentos lógicos,

4) termos técnicos, físicos e matemáticos de uso comum,

5) outras palavras conforme necessário para atingir os objetivos da PSPEC; usadas o menos possível,

6) equações matemáticas,

7) ilustrações como tabelas, diagramas e gráficos.

As estruturas gramaticais são construções simples, de uma saída e uma entrada:

1) concorrência: mais de uma atividade ocorre simultaneamente,

2) seqüência: atividades ocorrem em uma seqüência temporal especificada (se não for especificada a seqüência, elas são consideradas concorrentes),

3) decisão: uma ramificação no fluxo de atividades é feita baseado nos resultados de um teste sobre uma entrada,

4) repetição: a mesma atividade é repetida até que algum limite ou resultado especificado seja alcançado.

f) No inglês estruturado, usar MAIÚSCULAS para nomes de fluxos e depósitos, e deslocamento da margem para mostrar subordinação.

2.2 - PROCESSO DE CONSTRUÇÃO DO MODELO

Nesta seção está descrito o processo de construção do Modelo de Requisitos. Para compreender este processo, é necessário conhecer como é um modelo pronto. Isto está descrito na Seção 2.1.

Esta seção apresenta uma visão geral do processo e suas heurísticas básicas.

2.2.1 - DESCRIÇÃO DO PROCESSO

Os passos para construir o modelo estão listados abaixo. Segundo Hatley e Pirbhai (1987, p. 121), "a ordem em que eles são dados não deve ser interpretada como a ordem exata para serem seguidos--apenas como guia para o fluxo geral de trabalho. Tipicamente, o modelador trabalha em vários passos em paralelo e faz várias iterações entre passos, para frente e para trás." Os passos básicos são listados abaixo. A Figura 2.15 ilustra o processo.

1) Organize os requisitos do cliente/usuário em seus principais grupos funcionais.

2) Identifique as entidades externas(tais como outros sistemas, operadores, painéis de controle) com que é requerido que o sistema se comunique.

3) Identifique os grupos principais de informação que devem fluir entre o sistema e as entidades externas.

4) Comece a construir um diagrama de fluxos de alto nível, representando os grupos funcionais principais como processos, as entidades externas como fontes e destinos dos fluxos do sistema, e os grupos principais de informação como fluxos entre eles.

5) Estude o diagrama resultante e faça a si próprio perguntas como as seguintes:

   - Está certa a abrangência do sistema? Será que alguma entidade externa deveria fazer parte do sistema, ou algum processo ser entidade externa?

   - Será que os processos se relacionam bem com a maneira que os usuários vêem os requisitos do sistema? Será que um reparticionamento não tornaria os requisitos mais claros ao usuário?

   - Será que os fluxos movem-se de e para os lugares certos? Será razoável que os processos produzam as saídas mostradas com as entradas dadas?

   - A figura ficaria mais clara se os fluxos fossem reagrupados?

   Faça as mudanças resultantes de suas respostas a essas questões.

6) Desenhe um diagrama de contexto reunindo todos os processos em um único processo e, se apropriado, comprimindo os fluxos em menos grupos.

7) Remova as entidades externas do diagrama inicial de alto nível, formando um diagrama do nível 1.

8) Examine os requisitos principais, e decida se há quaisquer modos ou condições do sistema sob as quais se requeira que algum processo de alto nível seja desativado. Se hover, identifique os sinais representando esses modos ou condições, designe-os como sinais de controle, e construa uma CSPEC de nível 1. (IMPORTANTE: antes de fazer esse passo, assegure-se de que o mesmo efeito não seria obtido pela simples presença ou ausência dos dados apropriados a esses modos e condições.)

9) Tome cada processo do DFD atual e decomponha-o em um DFD filho (ou PSPEC).

10) Tome decisões sobre controle, iguais às tomadas para o nível 1, para cada processo do novo diagrama; construa as CSPECs e os CFDs necessários.

11) Cada vez que adicionar um novo fluxo de dados ou de controle, adicione-o ao dicionário de requisitos. Sempre que possível, adicione também sua definição.

12) Em paralelo com seu processo de decomposição, estude os requisitos do usuário, divida-os em requisitos elementares (usualmente uma divisão frase por frase), e identifique qual processo ou CSPEC acomoda cada um.

13) A cada dois ou três níveis feitos, verifique-os e corrija-os para torná-los mais compreensíveis e mais corretos. Verifique e corrija erros de balanceamento. Resolva ambiguidades e omissões nos requisitos, se possível; se não puder, anote-as para futura resolução. Faça iterações dessa maneira freqüentemente: quanto mais se atrasa as modificações, mais trabalhosas elas se tornam!

14) Quando puder expressar sem ambigüidades a função de um processo em algumas linhas de texto ou equações, ou com um diagrama simples, escreva uma PSPEC.

Note que os CFDs não são feitos por decomposição do nível de cima, mas a partir dos DFDs correspondentes (passo 10).

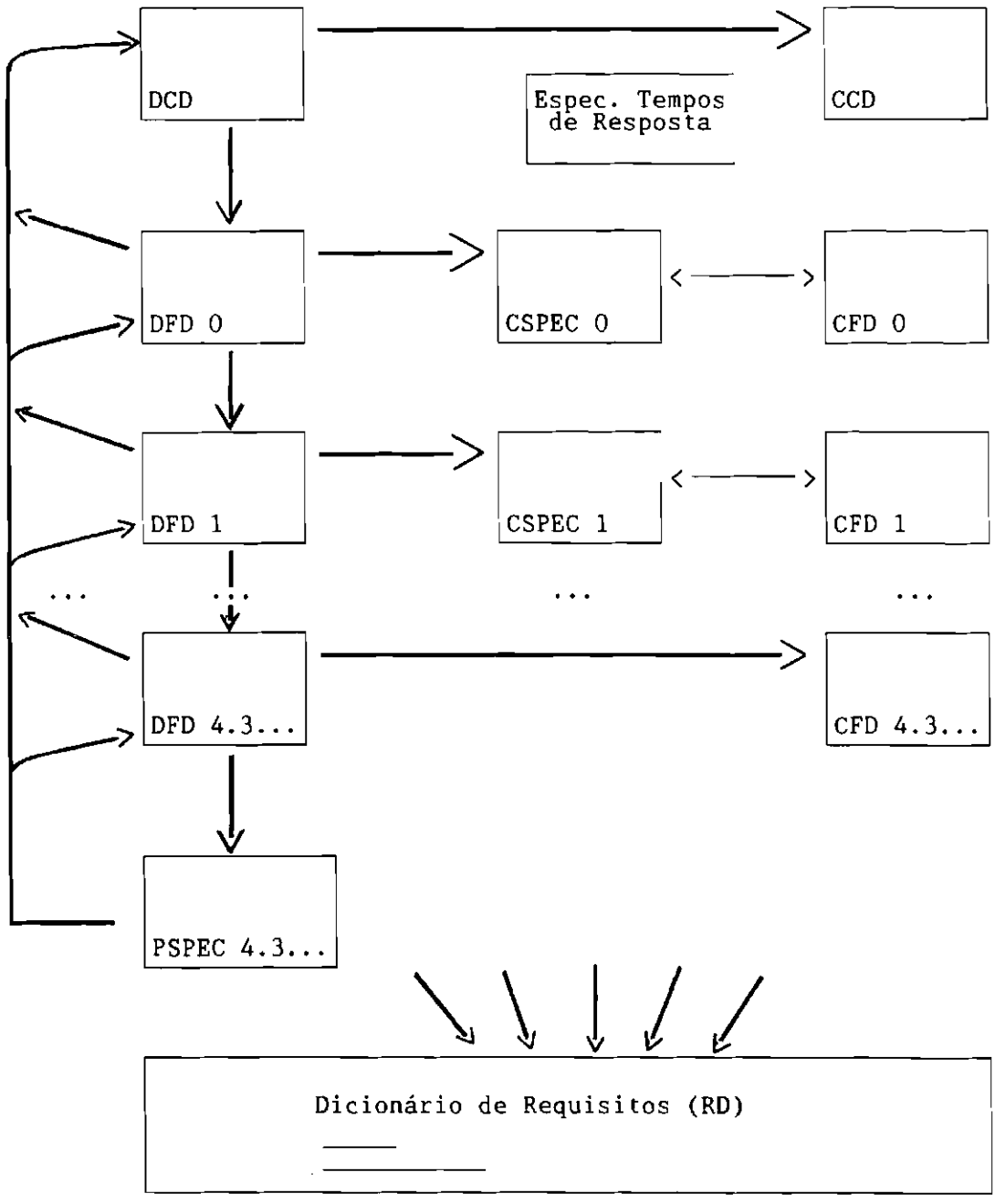


Fig. 2.15 - Processo de modelagem.

Legenda - Fluxo do trabalho: Principal..... ——

Realimentação... ———

2.2.2 - HEURÍSTICAS E DIRETRIZES

Esta seção apresenta as principais heurísticas do método. Elas orientam o processo de modelagem. Segundo Hatley e Pirbhai (1987, p. 123), "é importante frisar que construir um Modelo de Requisitos não é uma ciência exata. De certo modo, ensinar os passos da construção do modelo é como ensinar os passos para escrever um romance: pode-se ensinar algumas técnicas eficazes, mas não se pode ensinar as pessoas a serem grandes escritores. Analogamente, não há um único modelo correto para um dado conjunto de requisitos; pessoas diferentes farão modelos diferentes, todos podendo ser igualmente efetivos. O método é uma ferramenta, e como ele é usado é parcialmente uma função do sistema que ele está modelando."

As heurísticas do método são regidas pelos seus objetivos. Os objetivos de construir um conjunto hierarquizado de diagramas são: fazer uma definição clara e concisa dos requisitos em cada nível de detalhe, e particionar os requisitos em subconjuntos mutuamente exclusivos com relacionamentos precisamente definidos. Segue-se uma lista dessas heurísticas.

a) Faça a estrutura do modelo de controle vinculada à do modelo de processos. Ela deve ser um reflexo da estrutura de processos. É necessário conhecer o que vai ser controlado para definir como controlá-lo.

b) Decomponha fluxos em paralelo com processos. A decomposição pode ser feita: ou entre diagramas pai e filho, ou através de ramificação e junção de fluxos no diagrama filho.

c) Procure manter os diagramas simples, com uma distribuição uniforme de atividades. Para isso, faça iterações para cima e para baixo na estrutura do modelo.

d) Normalmente, inclua 7 ± 2 processos em um diagrama, mas reduza esse número quando os trajetos dos fluxos forem muito

complexos, e aumente-o quando forem muito simples.

e) Escolha nomes claros e concisos para processos e fluxos a fim de criar diagramas compreensíveis. Os nomes devem descrever a abrangência exata do fluxo, processo ou depósito; nem mais nem menos.

f) Use depósitos de dados e de controle em qualquer lugar em que eles ajudem a esclarecer o que o diagrama expressa.

g) Use a forma mais simples de modelo que puder: somente DFDs sempre que possível. Se for necessário usar controle, tente máquina combinacional primeiro; somente como último recurso use máquina seqüencial.

h) Escolha um arranjo para os diagramas que os torne mais claros e compreensíveis, e que ilustre a estrutura do sistema.

Pode-se resumir todas essas diretrizes em sua idéia essencial como: agrupe requisitos relacionados o mais próximo possível e por nível de detalhe, e expresse-os da maneira mais simples possível.

CAPÍTULO 3

MÉTODO WARD/MELLOR

Neste capítulo é descrito o método Ward/Mellor, que consiste em construir o assim chamado "Modelo Essencial". Ele expressa os requisitos essenciais do sistema, conforme citado na Seção 1.2.

Embora tenha o mesmo arcabouço geral do método Hatley, como mostrado na Seção 1.2, a abordagem de Ward/Mellor para a modelagem de requisitos é radicalmente diferente. Ao invés de usar decomposição funcional, a abordagem usada é a de modelagem dirigida por eventos externos, ou ainda, modelagem baseada no ambiente. Resumidamente, o ambiente no qual o sistema vai operar é representado por uma lista dos eventos que ocorrem neste ambiente, aos quais o sistema deve responder. Toda a modelagem do comportamento do sistema é baseada nesta lista de eventos.

A abordagem de Ward pode ser enquadrada tanto no paradigma tradicional de desenvolvimento por fases --de análise de requisitos, projeto e implementação--, como no paradigma "operacional", i.e., aquele que se baseia em dois conceitos: especificações executáveis e implementação por transformação, como definido por Bergland e Zave (1986). No entanto, o método é apresentado por Ward (1986) sem enfatizar nenhum dos dois paradigmas.

Um outro aspecto do método que merece menção é a idéia das "três dimensões" de complexidade de sistemas (Yourdon, 1988): sistemas computacionais têm em geral três componentes ortogonais de complexidade, como ilustrado na Figura 3.1. Próximo a cada eixo na figura, está uma região do espaço de complexidade correspondente a um sistema típico daquela região. Em alguns sistemas, somente uma dimensão tem complexidade significativa. Mas cada vez mais há sistemas complexos em duas ou três dimensões. Para sistemas tempo-real, embora a componente principal de complexidade seja referente a tempo, o uso crescente de software em sistemas embutidos cada vez maiores, faz

surgir sistemas tempo-real com grande carga de processamento de dados e grande complexidade de dados armazenados. Isso equivale a estender a região de sistemas tempo-real da Figura 3.1 para cima e para a direita.

A análise estruturada convencional é uma ferramenta para ajudar a lidar com a complexidade funcional. Ela não modela nada de dinâmica do sistema e ajuda muito pouco com os dados. O modelo de entidades e relacionamentos, emprestado pelos autores e incorporado ao método é uma ferramenta para lidar com a complexidade de dados armazenados. As extensões ao DFD relativas a estados e controle formam a ferramenta para lidar com a complexidade de dinâmica.

O método Ward/Mellor é descrito em duas partes, de acordo com o conceito de método firmado em 1.2. A Seção 3.1 contém uma descrição do chamado "Modelo Essencial" já construído. A Seção 3.2 apresenta o processo de construção do modelo.

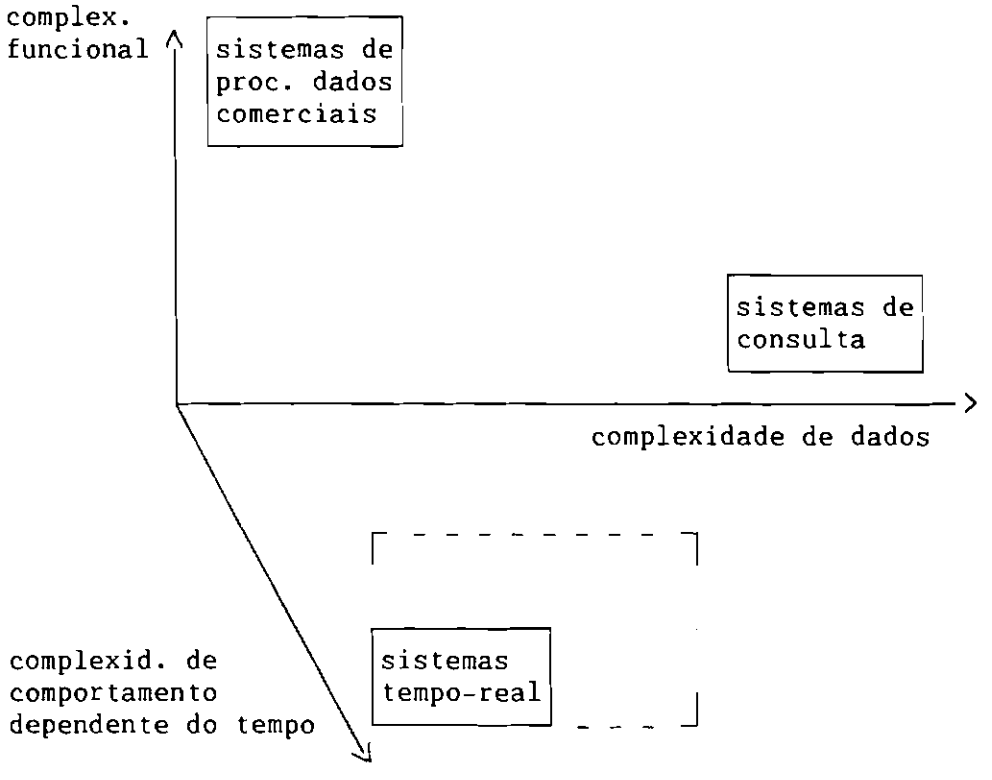


Fig. 3.1 - As três dimensões de complexidade de sistemas.

3.1 - O MODELO ESSENCIAL

Para permitir uma comparação sistemática dos dois métodos descritos neste trabalho, a descrição que se segue do Modelo Essencial é dividida em três partes. Cada parte refere-se a uma aspecto distinto do modelo:

1) sua Estrutura, ou seja, a maneira como seus componentes são organizados e se relacionam,

2) sua Notação, que é a "sintaxe" do método, ou seja, como representar seus vários componentes graficamente, e

3) suas Convenções, que compõem a "semântica" do método, isto é, o significado (ou a interpretação) dos componentes. Sem elas, não se pode compreender o que se pretendeu representar pelo modelo.

O Apêndice A contém algumas ilustrações do Modelo Essencial.

3.1.1 - ESTRUTURA

O Modelo Essencial é organizado numa estrutura que está ilustrada na Figura 3.2. Esta seção descreve seus componentes e sua organização baseado nesta figura.

A parte principal do Modelo Essencial é repersentada à esquerda. É uma hierarquia --uma árvore-- de "Diagramas de Transformações", que são extensões dos DFDs da análise estruturada clássica, obtidas com o acréscimo de fluxos e transformações de controle.

Nesta hierarquia de diagramas se integra também o diagrama de contexto, como o topo da hierarquia. A árvore é formada da seguinte maneira: cada transformação de um diagrama é detalhada no nível abaixo como um novo diagrama de transformações terminando em uma

especificação de transformação. (A Figura 3.2 representa, por simplicidade, todas as especificações do modelo no nível mais baixo da hierarquia. Na verdade elas são as folhas da árvore).

À direita, paralelamente à árvore de diagramas de transformações, há os diagramas de dados armazenados que são diagramas de entidades e relacionamentos (DERs) (Chen, 1976), para os depósitos de dados usados nos diagramas de transformações. Estes DERs não constituem uma hierarquia. Eles são pedaços do DER geral do sistema, correspondentes ao uso de depósitos de dados feito por diagramas de transformações. Isso visa facilitar a localização da entidade no DER que corresponde a um dado depósito de dados.

Além desses dois conjuntos de diagramas, há ainda dois componentes, a lista de eventos e o dicionário de dados. A primeira é uma lista dos eventos que ocorrem no ambiente, aos quais o sistema deve reagir.

O dicionário de "dados", apesar de seu nome, contém todas as especificações de fluxos de dados e de controle e depósitos de dados e de controle do modelo. É uma extensão do dicionário de dados da Análise Estruturada convencional. Sua estrutura interna é em forma de lista alfabética. Cada item do "dicionário de dados" é da forma:

<Nome> = <Significado> + <Composição> , se composto, ou

<Nome> = <Significado> + <Tipo> , se primitivo.

A definição segundo Ward e Mellor desses atributos (sinificado, composição e tipo), é dada na parte das convenções sobre o dicionário. Eles têm ainda uma simbologia própria, que é mostrada na parte sobre notação do dicionário.

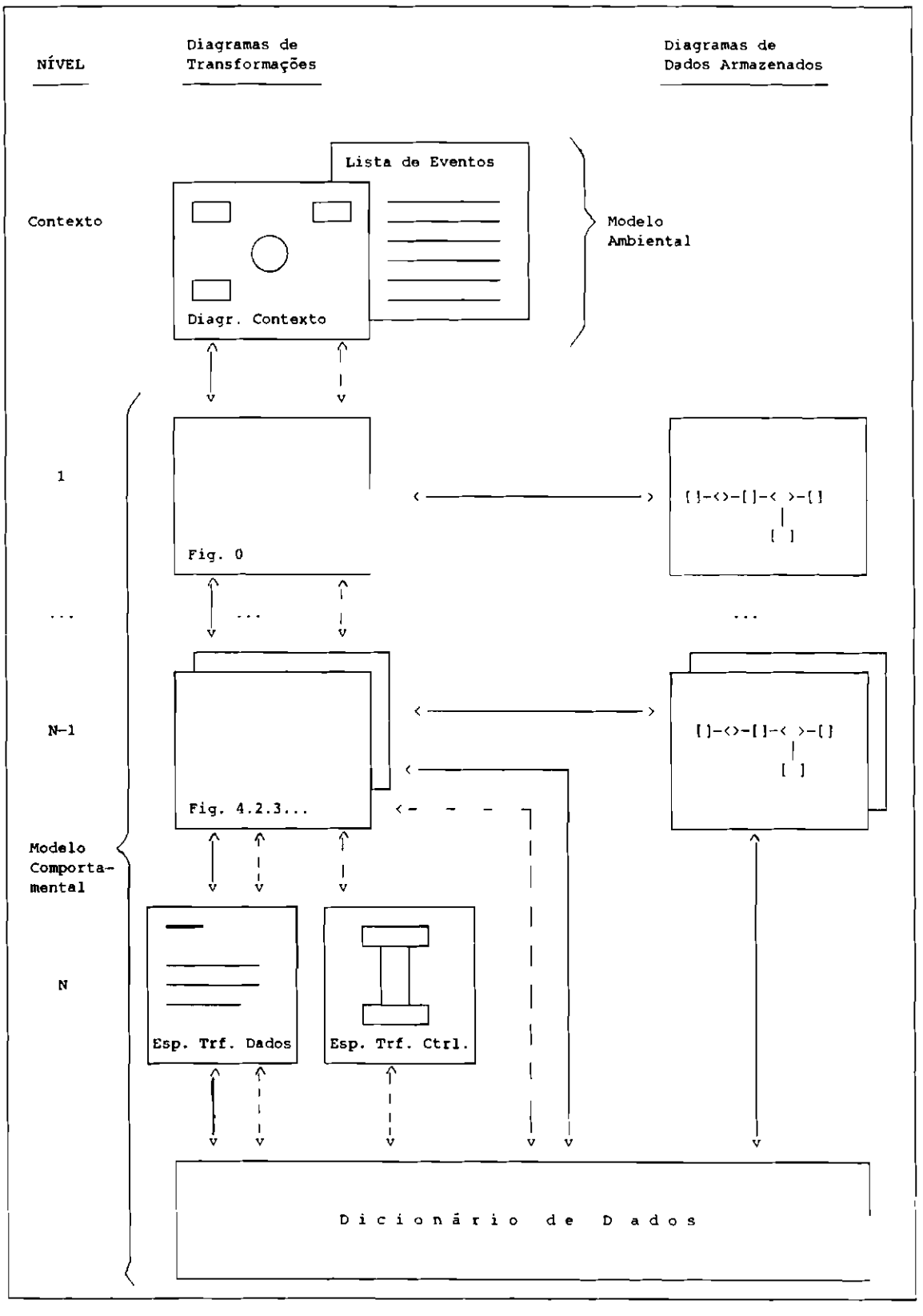


Fig. 3.2 - Estrutura do Modelo Essencial.

O que mantém a coesão da estrutura do modelo são duas coisas: uma regra de numeração e nomes para os diagramas e o balanceamento de fluxos entre diagramas e entre eles e o dicionário; tal como na Análise Estruturada convencional. A regra de numeração e nomes é descrita na Seção 3.1.3 sobre as convenções do modelo. O balanceamento é representado na Figura 3.2 pelas setas. Setas tracejadas representam balanceamento de fluxos de controle. Setas cheias representam balanceamento de fluxos de dados.

Como mostra a Figura 3.2, o Modelo Essencial é dividido conceitualmente em duas partes principais, chamadas de Modelo Ambiental e Modelo Comportamental. Cada um destes tem seus componentes específicos. Essa subdivisão está mostrada na Tabela 3.1. O Modelo Ambiental compõe-se do Diagrama de Contexto e da Lista de Eventos. O Modelo Comportamental se divide em Diagramas de Transformações, Especificações de Transformações, Diagramas de Dados Armazenados e Especificações de Dados, estas contidas no Dicionário de Dados.

TABELA 3.1

COMPONENTES DO MODELO ESSENCIAL

MODELO AMBIENTAL	Descrição do ambiente em que o sistema opera	DIAGRAMA DE CONTEXTO	Descrição da fronteira separando o sistema do ambiente
		LISTA DE EVENTOS	Descr. dos eventos externos aos quais o sistema deve responder.
MODELO COMPORTAMENTAL	Descrição do comportamento do sistema em resposta a eventos no ambiente	DIAGRAMA TRANSFORM. E ESPECs	Descr. das Transf. que o sistema faz em resposta a eventos.
		DIAGR. DE DADOS (DER) E DICIONÁRIO DADOS	Descr. da informação que o sistema deve armazenar para responder aos eventos.

3.1.2 - NOTAÇÃO

3.1.2.1 - NOTAÇÃO DOS DIAGRAMAS DE TRANSFORMAÇÕES

Os símbolos usados nos diagramas de transformações são mostrados nas Figuras 3.3 a 3.9. Os significados de cada símbolo são:

a) Entidade Externa

Tem o mesmo significado e representação que no método Hatley (ver Seção 2.1.2.1.a).

b) Transformação de Dados

É a mesma transformação de dados descrita por DeMarco (1978) e Gane e Sarson (1979), por eles chamada de função ou processo. Ela transforma entradas de dados ou dados armazenados em saídas de dados e/ou armazena essas saídas. Também tem o papel de informar a ocorrência de eventos.

c) Transformação de Controle

É uma abstração de uma porção da lógica de controle do sistema. Servem para controlar o comportamento de outras transformações ativando-as ou desativando-as. Também servem para o sistema trocar informações de controle com o ambiente.

d) Fluxo de Dados

Há dois tipos de fluxos de dados: fluxo de dados discreto (no tempo) e fluxo de dados contínuo (no tempo); cada qual com seu significado:

- Um "fluxo de dados discreto" representa um conjunto de dados de valores variáveis que é transmitido, ou tem valores, somente em pontos discretos no tempo e considera-se ter valor indefinido ou

nulo fora desses pontos no tempo (ex.: um relatório, um nome de cliente, etc.). É uma abstração para uma transação ou outros agregados de dados transmitidos como uma unidade pelo sistema.

- Um "fluxo de dados contínuo" representa um conjunto de valores variáveis que é transmitido continuamente ao longo de um intervalo de tempo (ex.: temperatura, pressão).

Não confundir fluxo contínuo com a idéia de faixa contínua de valores. Uma variável pode ter uma faixa contínua de valores e mesmo assim fazer parte de um fluxo discreto no tempo.

e) Fluxo de Evento (ou seja, "de controle")

Representa uma sinalização de um acontecimento ou a transmissão de um comando. Fluxos de evento não têm conteúdo de dados a serem transformados.

f) Depósito de Dados

Um conjunto de dados armazenados para uso posterior.

g) Depósito de Eventos

Um conjunto de informações sobre eventos ocorridos, armazenado para uso posterior. Funciona como um semáforo (Dijkstra, 1968).

h) Transformações múltiplas idênticas

Existem várias cópias idênticas de uma transformação no sistema. Ver convenção na Seção 3.1.3.6.f.

Fig. 3.3 - Transf. de Dados: Um círculo cheio com um nome e um número.

Fig. 3.4 - Transf. de Controle: Um círculo tracejado com um nome e um número.

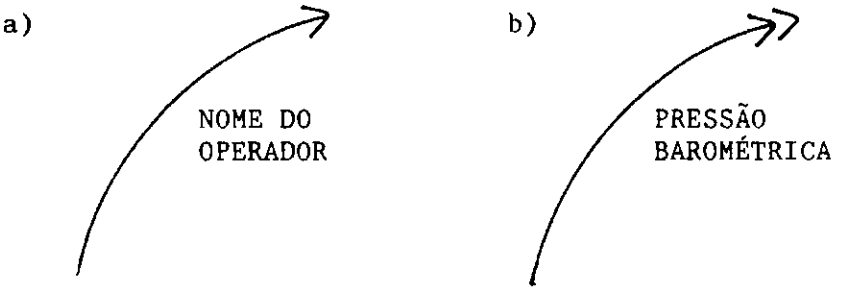


Fig. 3.5 - a) Fluxo de Dados Discreto: Arco em linha cheia, orientado, rotulado.

b) Fluxo de Dados Contínuo: Arco em linha cheia, com dupla seta, rotulado.

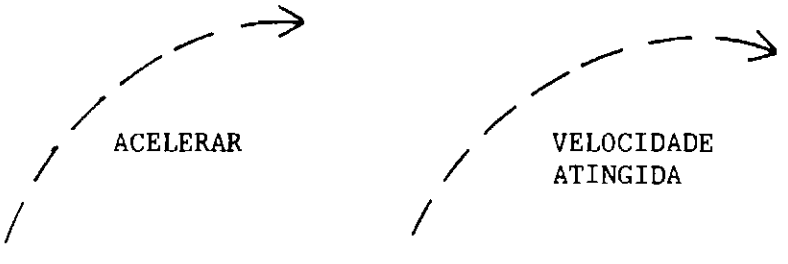


Fig. 3.6 - Fluxo de Evento: Arco tracejado com seta e nome.

AERONAVE

Fig. 3.7 - Depósito de Dados: Duas linhas paralelas contendo um nome.

COMBUSTÃO ANORMAL

Fig. 3.8 - Depósito de Eventos: Duas linhas paralelas tracejadas contendo um nome.

Fig. 3.9 - Transformações Múltiplas: Círculo duplo.

3.1.2.2 - NOTAÇÃO DO DIAGRAMA DE TRANSIÇÃO DE ESTADOS (DTE)

É idêntica à notação do método Hatley (ver Seção 2.1.2.2).

3.1.2.3 - NOTAÇÃO DO DIAGRAMA DE DADOS ARMAZENADOS (DER)

Essa notação é basicamente a mesma definida por Chen (1976) para o diagrama de entidades e relacionamentos (DER), com algumas extensões propostas por Flavin (1981), como substituir o uso de atributos de relacionamento pela noção de "objeto associativo". Também dá outros nomes aos elementos do diagrama. Como o DER pode ser usado de maneira mais geral para modelagem de informação sobre qualquer assunto, esta mudança de nomes, principalmente chamar o DER de diagrama de dados armazenados, visa deixar claro o uso particularizado do DER para modelagem de dados armazenados no sistema. Entre parênteses é mostrado o nome dado por Chen a cada componente do diagrama.

Os símbolos usados nos DERs são mostrados nas Figuras 3.10 a 3.15. Os significados de cada símbolo são:

a) Tipo de Objeto (conjunto de entidades)

Conjunto de dados armazenados, todos da mesma categoria (corresponde a um depósito de algum diagrama de transformações). Seu nome é indicativo da categoria.

b) Relacionamento (conjunto de relacionamentos)

Associações que existem entre os tipos de objeto. Seu nome indica o significado dessa associação.

c) Arco de Ligação

Ligação entre um relacionamento e os tipos de objeto associados por ele.

d) Tipo de Objeto Associativo

Como o nome já diz, é um elemento do diagrama que tem um papel duplo: como um relacionamento, associa tipos de objeto (contém informação estrutural do diagrama, identificando as ocorrências de cada tipo de objeto ligadas por ele). Como um tipo de objeto, é descrito por elementos de dados atribuídos a ele, e pode ser ligado a outros tipos de objeto por outros relacionamentos.

e) Arco de Ligação de Subordinação

Usado com um caso particular de relacionamento, o relacionamento de tipo, que associa um tipo de objeto com outros que são seus subtipos. Um traço cortando o arco de ligação denota o supertipo.

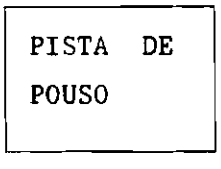


Fig. 3.10 - Tipo de Objeto:
Um retângulo contendo um nome.

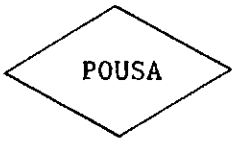


Fig. 3.11 - Relacionamento:
Um losango contendo um nome.

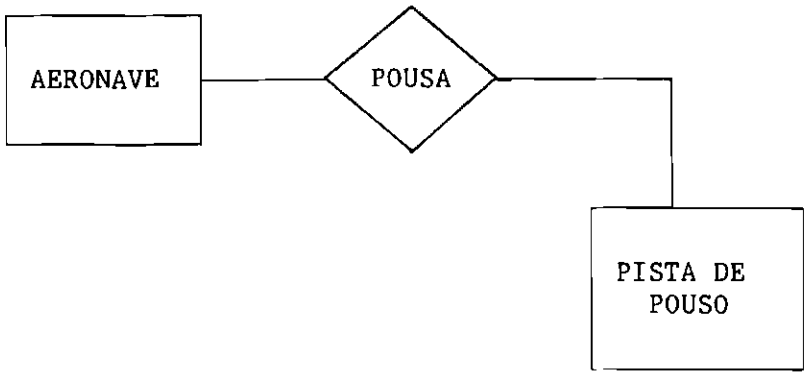


Fig. 3.12 - Arco de Ligação: Linha de segmentos retos.

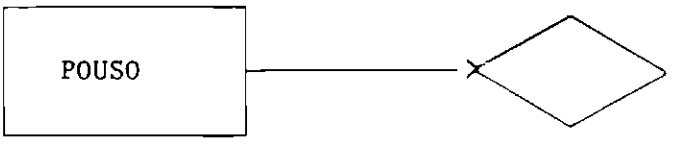


Fig. 3.13 - Tipo de Objeto Associativo: Losango vazio ligado por seta a um retângulo com o nome do objeto associativo.

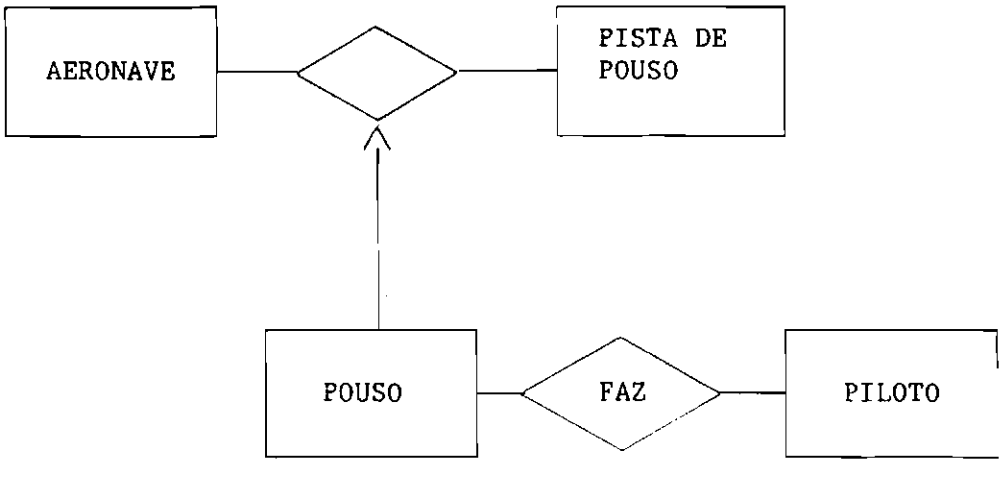


Fig. 3.14 - Exemplo de uso de objeto associativo.

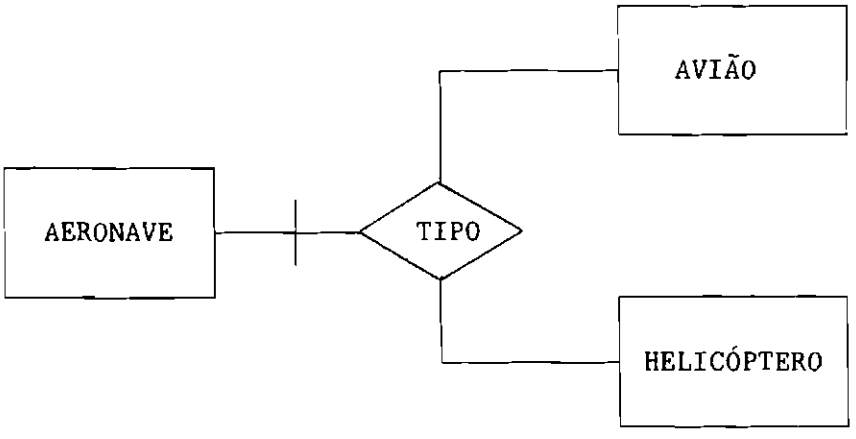


Fig. 3.15 - Arco de Subordinação: Linha cortada por um traço curto.

3.1.2.4 - NOTAÇÃO DO DICIONÁRIO DE DADOS

A notação para especificar itens do dicionário de dados é dada pela Tabela 3.2.

Além da simbologia da Tabela 3.2, para especificar Faixa de Valores, Precisão e Unidades de itens de dados primitivos, não há uma simbologia especial, exceto que são especificados entre asteriscos, por exemplo:

Faixa: real, limites 0-100; Precisão 0.01; Unidades: km/h
Faixa: inteiro; Unidades: ciclos

TABELA 3.2

NOTAÇÃO DO DICIONÁRIO DE DADOS

SÍMBOLO	SIGNIFICADO	DESCRIÇÃO
=	é especificado por	Indica que o item cujo nome está à sua esquerda é especificado pelos atributos à sua direita.
+	junto com	Indica que ambos os ítens à esquerda e à direita vão juntos na Composição do item. NÃO IMPLICA ORDEM (é como a união de conjuntos).
[... ... ...]	selecione um de	(Dois ou mais ítens entre colchetes, separados por barras). Indica que um e somente um item aparece em cada ocorrência do item (é a função booleana "OU exclusivo").
{ ... }	iterações de	Um item dentro de chaves pode ocorrer qualquer número de vezes (zero ou mais) em cada ocorrência do item. Para indicar faixas para o número de vezes, usar índices: m{ }n : de m a n iterações { }n : de 0 a n iterações m{ } : m ou mais iterações
@	identificador para dados armazenados	Usado na especificação de Composição de tipos de objeto e de relacionamentos, para marcar os ítens de dados que compõem a chave, ou seja, o identificador de unívoco de cada instância de objeto ou de relacionamento.
* ... *	significado	Texto entre asteriscos especifica o Significado do Nome do item.
[... ... ...]	enumeração	Os valores de um item de dados primitivo discreto são enumerados separados por barras dentro de (*[) e (]*).

3.1.3 - CONVENÇÕES

A maioria das convenções desta seção se aplicam à segunda etapa do processo de modelagem (derivar o modelo comportamental) -- descrito na Seção 3.2.1. Nesta etapa, faz-se um diagrama de transformações único para todo o sistema, e, se for o caso, um diagrama de dados armazenados também único. Para se compreender adequadamente as convenções é importante ter em mente que neste método não se trabalha na hierarquia de diagramas.

As convenções que se referem à hierarquia de diagramas, que é produzida na etapa final do processo de modelagem, estão identificadas explicitamente no texto.

3.1.3.1 - CONVENÇÕES SOBRE HIERARQUIA

As convenções abaixo referem-se a aspectos ligados à hierarquização dos diagramas do modelo.

a) Fluxos e depósitos podem ser hierarquizados assim como transformações. Os de nível mais alto servem como "empacotamentos" para os detalhes do nível mais baixo. Assim, o que cada elemento de um diagrama pode agrupar do nível de baixo, é:

   - Uma transformação de dados, pode agrupar um conjunto de transformações de dados, e de controle.

   - Uma transformação de controle, pode agrupar um conjunto só de transformações de controle.

   - Um fluxo dados discreto, contínuo, ou um fluxo de evento, pode agrupar um conjunto de fluxos do mesmo tipo.

   - Um depósito de dados ou de eventos, pode agrupar um conjunto de depósitos do mesmo tipo.

b) Convenção de numeração e nomes dos diagramas: O diagrama de contexto não é numerado. O diagrama do próximo nível abaixo é numerado como figura zero, e suas transformações são numeradas como 1, 2, 3, ... A numeração não implica seqüenciamento das transformações mas apenas as diferencia. Da figura zero para baixo cada diagrama leva não somente o nome mas também o número da transformação que resume seus detalhes. As transformações nesses diagramas levam o número do diagrama mais um ponto decimal e outro número. Por exemplo, as transformações na Figura 2.2 são numeradas como 2.2.1, 2.2.2, etc.

c) Embora o conjunto de diagramas de transformações seja uma hierarquia, as conexões entre eles representam inerentemente apenas resumos -- em outras palavras, ela é uma hierarquia de representação. Contudo, é possível usar fluxos de eventos em conjunção com uma hierarquia de diagramas de maneira que os níveis da hierarquia sejam coincidentes com níveis de controle, e assim, tem-se um hierarquia de controle.

d) O diagrama de dados armazenados (DER para o sistema todo) precisa ser partido pelas mesmas razões que os diagrama de transformações (limitar a complexidade). Parte-se o diagrama de dados quebrando-o em pedaços correspondentes ao uso de dados armazenados por diagramas individuais da hierarquia de diagramas de transformações. Esse esquema fará com que tipos de objetos apareçam redundantemente em porções do diagrama de dados, mas ajudará a compreender os sub-diagramas individuais.

3.1.3.2 - CONVENÇÕES SOBRE FLUXOS DE DADOS

As convenções listadas abaixo permitem interpretar os requisitos, que são de interfaceamento, expressos por fluxos de dados.

a) Uma transformação de dados pode ser disparada por dados ou por fluxos de controle, (ordens de execução "Ativar/Desativar" e "Disparar"). Há uma importante distinção entre um par de

transformações conectadas por um fluxo e um par delas ligado por um depósito, mostrada na Figura 3.16. Essa distinção visa satisfazer a abordagem do método de obter uma especificação executável. Ward e Mellor usam um conceito chamado "transformação síncrona", que impõe certas restrições à interface de uma transformação, de modo a executar o diagrama sem ter especificações das transformações de dados (ver convenção na Seção 3.1.3.6.e).

b) O uso de convergência e divergência de fluxos, como mostrado na Tabela 3.3, serve para indicar nos diagramas, múltiplas origens ou múltiplos destinos para os fluxos de dados ou de controle. No caso de fluxos de dados, pode indicar também ocorrer combinação ou separação de conteúdo.

c) Fluxos contínuos no tempo são úteis para representar caractarísticas do ambiente físico (temperaturas, voltagens, etc) que podem ser detectadas por algum sensor do sistema, e para representar sinais de saída (por exemplo, uma desejada posição de uma válvula pelos quais o sistema controla aspectos continuamente variáveis da tecnologia externa).

d) Transformações de dados podem exercer controle externo criando fluxos de dados contínuos como saída para o ambiente.

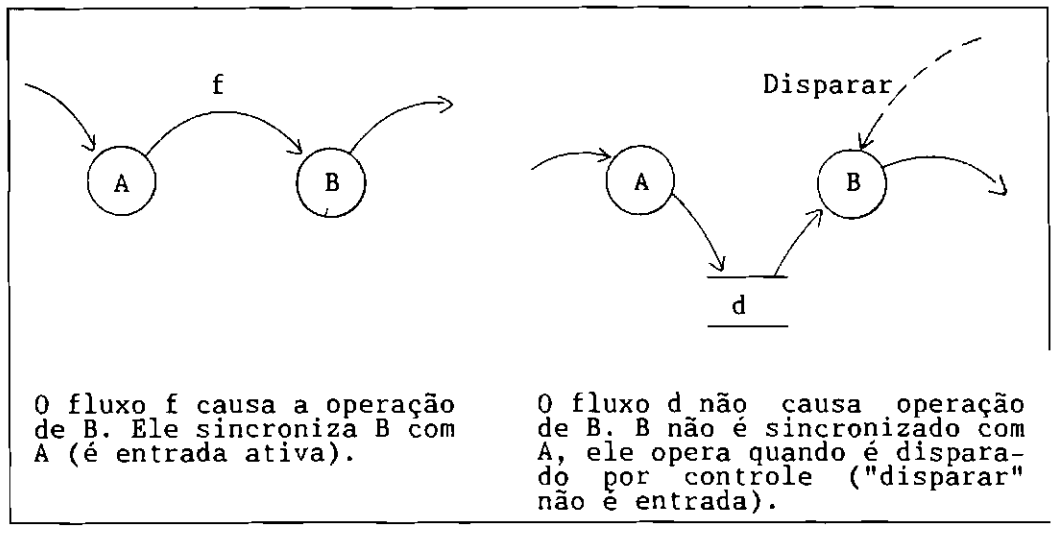


Fig. 3.16 - Fluxos e depósitos ligando transformações.

TABELA 3.3

CONVERGÊNCIA E DIVERGÊNCIA DE FLUXOS

CONVENÇÃO	INTERPRETAÇÃO
Z X Y	Dois subconjuntos de Z são usados por duas transformações sucessoras diferentes
Z	Todo o fluxo Z é usado por duas transformações sucessoras
X Z Y	Z é composto de dois subconjuntos fornecidos por duas transformações predecessoras
Z	Todo o fluxo Z pode ser fornecido por uma de duas transformações predecessoras

3.1.3.3 - CONVENÇÕES SOBRE FLUXOS DE EVENTO (CONTROLE)

As convenções listadas abaixo permitem interpretar os requisitos, que são de interfaceamento, expressos por fluxos de evento.

a) Fluxos de evento são fluxos que não têm nenhum conteúdo (a ser processado como dado), eles são simplesmente sinais de que algo aconteceu ou dão um comando.

b) Pode-se separar os aspectos de um sistema que são contínuos no tempo dos discretos no tempo com as seguintes convenções:

   - para uma transformação de controle, somente fluxos de evento são permitidos como entrada e saída.

   - para uma transformação de dados, somente fluxos de dados são permitidos como entrada, mas tanto fluxos de dados como de evento podem ser produzidos como saída.

   - tanto para transformações de dados quanto para de controle, fluxos rotulados como "Ativar", "Desativar", ou "Disparar" são interpretados como uma "ordem de execução", não como entradas (simplesmente as ligam e desligam), e somente podem ser emitidos por transformações de controle.

c) O uso de fluxos de evento como ordens de execução, combinado com o uso de fluxos contínuos e discretos, permite interpretação rigorosa (não ambígua) do comportamento de transformações de dados ao longo do tempo. O comportamento depende do tipo de entradas e saídas; como mostrado na Tabela 3.4.

Note que não há necessidade de que uma transformação de dados sempre ordem de execução. Uma transformação sem ordem de execução se comporta da mesma maneira todo o tempo.

TABELA 3.4

COMPORTAMENTO DINÂMICO DE TRANSFORMAÇÕES

Entradas	Saídas	Comportamento
Contínuas	Contínuas	Produz saídas após um comando "Ativar" e antes de um "Desativar", mas não produz saída durante outros intervalos de tempo.
Discretas	Quaisquer	Aceitará entradas após um "Ativar" e antes de um "Desativar", e não aceitará entradas em outros intervalos.
Contínuas	Discretas	(com "Disparar" como ordem de execução) Produzirá sua saída discreta baseada no valor instantâneo da entrada contínua a cada vez que recebe o comando.

3.1.3.4 - CONVENÇÕES SOBRE DEPÓSITOS DE DADOS

As convenções listadas abaixo permitem interpretar os requisitos expressos pelos depósitos de dados.

a) As convenções para conectar transformações a depósitos se relacionam ao fluxo total entre eles, das seguintes maneiras. O

fluxo ligando o depósito e a transformação nunca é rotulado -- ele representa disponibilidade do depósito para a transformação. O fluxo apontando para a transformação significa que alguma saída da transformação usa algo do depósito. O fluxo apontando para o depósito significa que ele é modificado em valor(es) ou na quantidade de elementos pela transformação.

b) Os itens num depósito permanecem invariantes durantes períodos de tempo entre instantes em que transformações os modificam. Durante esses períodos, o conteúdo do depósito é disponível para ser usado por outras transformações. Assim, um depósito representa um relacionamento entre processos, defasado no tempo. Modificações ou usos de depósitos são representados como ocorrendo em pontos discretos no tempo, ou seja, por um fluxo de dados discreto.

c) Um fluxo de dados pode entrar ou sair de um depósito somente via uma transformação.

d) (Na hierarquia:) Já que fluxos entre depósitos e transformações não têm rótulo, deve-se mostrar um depósito no nível mais alto em que ele se conecta com duas ou mais transformações e em todos os níveis abaixo (para se saber o nome dos fluxos).

3.1.3.5 - CONVENÇÕES SOBRE DEPÓSITOS DE EVENTOS

As convenções listadas abaixo permitem interpretar os requisitos expressos pelos depósitos de eventos.

a) É usado para registrar ocorrências de fluxos de eventos que ocorreram mas não devem ser usados ainda. Funciona no diagrama como um semáforo, conforme Dijkstra (1968).

b) Um fluxo de evento vindo do exterior pode ser ligado diretamente a um depósito de eventos sem nenhuma transformação

no meio. Esta convenção é diferente da usada para depósitos de dados.

3.1.3.6 - CONVENÇÕES SOBRE O DIAGRAMA DE TRANSFORMAÇÕES

As convenções listadas abaixo permitem interpretar os requisitos de interfaces de dados e controle expressos pelos diagramas de transformações.

a) O Diagrama de Contexto é freqüentemente muito grande para uma única folha de papel (note, não o da hierarquia, mas o da etapa de modelagem do ambiente). A solução é simples: dividir o diagrama em várias folhas. Cada uma mostra um trecho do contorno de um grande círculo (a fronteira do sistema) e algumas das entidades externas. As várias folhas juntas formam um diagrama de contexto completo.

b) Transformações de dados podem exercer controle externo ao sistema gerando fluxos de saída contínuos. Contudo, somente transformações de controle podem ativar e desativar transformações dentro do sistema (além de exercer controle no exterior do sistema).

c) Uma transformação de controle só aceita fluxos de evento e só produz fluxos de evento.

d) Além de receber fluxos de eventos de entrada, transformações de controle podem ser ativadas ou desativadas por outras transformações de controle. Ao ser ativada, a transformação de controle começa em seu estado inicial. Ao ser desativada, as convenções são as seguintes: Quaisquer transformações do sistema ou atividades externas ao sistema que tenham sido ativadas pela transformação de controle são desativadas. O estado da transformação de controle se torna indefinido.

e) Transformação de dados "síncrona": A transformação de dados síncrona obedece ao seguinte conjunto de convenções a respeito de seus fluxos de dados discretos (ordens de execução e ligações a depósitos de dados não são considerados ao aplicar-se as convenções):

- Pode existir no máximo uma entrada discreta.

- Se a entrada discreta é composta, todos os seus elementos devem estar presentes para a transformação operar.

- Podem existir zero ou mais saídas discretas.

- Se há duas ou mais saídas discretas, elas têm que ser alternativas, e no máximo uma pode ser produzida para cada operação da transformação."

> NOTA: Visa execução esquemática do diagrama preliminar, ou seja, sem as especificações de dados e de controle. "Síncrona" equivale a "sem ambigüidade no diagrama", isto é, com no máximo uma "entrada ativa" e saídas discretas alternativas.

f) Múltiplos subsistemas equivalentes. -- Quando se tem várias réplicas de um subsistema, com alguma coordenação geral entre elas, usa-se o círculo duplo para as transformações que devem existir em réplicas, com as seguintes convenções:

- os fluxos de/para elas são desenhados como se só existisse uma transformação, mas entende-se que há um fluxo para cada réplica,

- não se faz uma especificação para cada réplica da transformação, pois seriam todas iguais. Faz-se uma só,

- os depósitos usados por elas não são replicados, são únicos, mas contêm os dados referentes a todas as réplicas.

g) Idealizações do modelo. -- As transformações de dados ou de controle, ao serem disparadas ou enquanto ativas, operam em tempo zero. Os depósitos têm capacidade infinita e acesso a eles leva tempo zero.

3.1.3.7 - CONVENÇÕES SOBRE ESPECIFICAÇÃO DE TRANSFORMAÇÃO DE DADOS

Abaixo são listadas as convenções a respeito dos requisitos de processamento de dados, que vão nas especificação de transformação de dados.

a) As maneiras possíveis de descrever uma transformação de dados são (segundo Ward, 1985):

- usando linguagem de programação (Ada ou Pascal),

- uma Linguagem de Projeto de Programas (PDL), que tem sintaxe formal,

- Pseudocódigo (sintaxe informal),

- Inglês Estruturado ,

- Árvore ou tabela de decisão,

- Pré e pós condições.

b) Um diagrama ou tabela de transição de estados é outra ferramenta de especificação gráfica que pode ser usada quando os fluxos de saída têm valores discretos. Embora seja útil para transformação de controle, ele também pode ser usado para certas transformação de dados.

c) Um método de especificação particularmente poderoso envolve relacionar condições sobre valores de entrada a condições correspondentes sobre valores de saída. Chamado de pré-condições e pós-condições. Em geral é necessário usar mais de uma combinação de pre-condição e pós-condição. No entanto é muito difícil para transformações não triviais.

3.1.3.8 - CONVENÇÕES SOBRE ESPECIFICAÇÃO DE TRANSFORMAÇÕES DE CONTROLE

As convenções listadas abaixo permitem interpretar os requisitos de processamento de informações de controle, expressos numa especificação de transformação de controle.

a) A memória interna (os estados) de uma transformação de controle é caracterizada em termos do comportamento externamente observável do sistema que está sendo controlado.

b) Um "estado" representa um modo de comportamento externamente observável. O nome do estado é o nome do comportamento exibido pelo sistema. A transformação de controle não faz nada enquanto dura um estado. Podemos considerar o estado como um coordenador que está aguardando algo acontecer para tomar uma atitude. O tempo de permanência num estado depende de quando ocorrem eventos que provocam as transições ligadas a ele, sendo geralmente maior que zero e vinculado ao ambiente.

c) "Transições" representam movimento de um estado para outro. Esse movimento leva tempo zero. A uma transição estão associadas condições e ações.

d) "Condições" fazem o sistema fazer uma transição. Cada condição é identificada com um fluxo de evento de entrada. O DTE NÃO descreve a lógica envolvida na computação da condição.

e) "Ações" são tomadas quando uma transição ocorre. Uma ação é uma atividade única e indivisível, e é identificada com um fluxo de

evento de saída. Várias ações independentes podem ser tomadas em uma única transição. Considera-se que elas ocorrem instantaneamente e também simultaneamente a menos que seja indicada uma ordem explícita. Pode haver uma transição sem nenhuma ação associada, se o sistema precisa apenas lembrar-se de uma mudança de comportamento mas não precisa influenciá-la.

f) Usa-se estados para representar intervalos de tempo durante os quais algum comportamento persiste e transições para representar pontos no tempo em que o comportamento muda. Portanto usa-se o modelo Mealy de máquina de estado, já que os símbolos de saída, os fluxos de evento, são também localizados em pontos no tempo.

g) Estado Transitório. -- É comum haver requisitos em que a ação de resposta a um evento que causa uma transição de estado é condicional; o sistema deve checar alguma condição para decidir para qual estado irá nesta transição. Este cheque é uma transformação (função) interna ao sistema, que enviará fluxos de eventos sinalizando a condição. Para tais casos, a convenção é:

- usa-se um estado durante o qual a condição é checada. É um estado transitório, pois sua duração é zero, diferentemente da noção comum de estado, que dura $t > 0$, enquanto as condições externas não mudam;

- para diferenciá-lo, não se dá nome ao estado transitório, mesmo porque o nome não poderia ser dado em termos do ambiente, mas da função interna sendo executada.

3.1.3.9 - CONVENÇÕES SOBRE DIAGRAMA DE ENTIDADES E RELACIONAMENTOS

O modelo de entidades e relacionamentos pode ser usado, de uma maneira bem geral, para modelar a semântica de um assunto em estudo. Usado desta maneira, ele corresponde melhor à porção

esquemática de um Modelo de Informação. Ele ajuda a esclarecer o pensamento sobre um assunto permitindo expor as categorias relevantes de informação e suas associações -- um uso que pode não ter nada a ver com um desenvolvimento de um sistema.

Ao escolher o nome "diagrama de dados armazenados" para o modelo, Ward e Mellor (1985) restringem o uso do DER para pensar nas entidades como espécies de depósitos de dados. Assim, uma entidade é uma abstração para os vários mecanismos para armazenamento de dados -- arquivos, bases de dados, buffers, pilhas, filas, áreas comuns -- usados em sistemas automatizados ou manuais. Este uso não conflita com a interpretação mais geral do modelo de entidades e relacionamentos. As convenções a esse respeito são listadas abaixo.

a) Regras de coerência: As regras para coerência do diagrama de dados são derivadas do papel do diagrama de especificar um esquema de classificação de dados. Esse papel requer que dados sejam capazes de ser "colocados dentro" e "tirados" do modelo de maneira não ambígua. Para um diagrama de dados ser considerado internamente coerente, as seguintes regras se aplicam:

   - cada entidade deve ter um nome único,

   - cada relacionamento deve ter um nome único,

   - cada ocorrência de cada entidade ou de cada relacionamento deve ser unicamente identificável (ter uma chave).

b) Somente entidades podem ter elementos de dados como atributos. Quando se tem elementos de dados atribuíveis a um relacionamento, a convenção é usar uma "entidade associativa". Ela tem dois papéis no modelo: como relacionamento, liga entre si outras entidades do modelo; como entidade, é descrita por seus atributos e pode ser ligada a outras entidades por outros relacionamentos.

3.1.3.10 - CONVENÇÕES SOBRE DICIONÁRIO DE DADOS

No dicionário de dados, tanto fluxos e depósitos quanto entidades e relacionamentos devem ser especificados precisamente.

Dados têm três aspectos -- significado, composição e tipo -- que comumente são confundidos. Visando uma clara especificação, Ward e Mellor definem as seguintes convenções para o dicionário.

a) Especifica-se o "significado" de um dado descrevendo o papel que a coisa referida pelo dado desempenha no sistema. Essa especificação não deve restringir a composição dos dados usados para representá-lo.

b) Uma especificação de "composição" declara os nomes e relacionamentos dos dados que, quando tomados juntos, desempenham o papel definido pelo significado.

c) Usa-se a palavra "tipo" para denotar o conjunto de valores que um elemento de dados pode assumir. Pode-se especificar o tipo através de qualquer combinação de: enumeração, faixa de valores, precisão e unidades.

d) Especificação de componentes do Diagrama de Transformações: Todos os fluxos e depósitos necessitam de especificações de dados. Os aspectos a incluir são identificados na Tabela 3.5.

e) Especificação de componentes do DER. Os aspectos a especificar para cada componente do DER são identificados na Tabela 3.6.

f) Uma especificação de dados pode servir para especificar tanto um tipo de objeto quanto o depósito de dados correspondente. Já que categorias de dados armazenados tipicamente representam coleções de entidades similares do mundo real, é freqüentemente necessário dar nome tanto para uma única ocorrência de uma entidade quanto para o conjunto de entidades. A convenção é:

. o tipo de objeto do diagrama de dados leva o nome da única ocorrência,

. o depósito leva o nome do conjunto de ocorrências,

. a especificação de significado é dada sob o nome da única ocorrência,

. a especificação de composição para o conjunto é em termos de uma iteração da única ocorrência.

Seja, por exemplo, a representação em dados armazenados de um conjunto de posições de uma aeronave. O nome do depósito seria "Posições da Aeronave", e o nome da entidade seria "Posição da Aeronave". A especificação de dados seria:

Posições da Aeronave = {Posição da Aeronave}

Posição da Aeronave = *medida de distância de uma aeronave a partir do centro da Terra*

= Latitude + Longitude + Altura

Assim como tipos de objeto, relacionamentos correspondem a depósitos de dados, aplicando-se as mesmas convenções de nomes.

g) Especificações de composição para entidades associativas que dependem de ocorrências únicas de outras entidades não incluem seus atributos identificadores. Em vez disso, adiciona-se o sufixo 'ref' ao nome da entidade para indicar a conexão a uma ocorrência da entidade, não interessando os verdadeiros elementos de dados usados para identificá-la (esta convenção permite mudar o identificador de uma entidade sem ter que alterar outras).

TABELA 3.5

CONVENÇÕES DO DICIONÁRIO PARA FLUXOS E DEPÓSITOS

	Fluxo (dados/contr)		Depósito	
	elementar	composto	elementar	composto
Significado	X	X	X	X
Composição		X		X
Tipo	X		X	

TABELA 3.6

CONVENÇÕES DO DICIONÁRIO PARA FLUXOS E DEPÓSITOS

	Tipos Obj. (Entidades)	Relacionamentos	Atributo Elementar
Significado	X	X	X
Composição	X	X	
Tipo			X

3.2 - PROCESSO DE CONSTRUÇÃO DO MODELO

Esta seção descreve o processo de construção do Modelo Essencial. A Seção 3.2.1 descreve a parte mecânica do trabalho. Embora apresentadas em separado, na Seção 3.2.2, as heurísticas fazem parte integrante do processo de modelagem.

3.2.1 - DESCRIÇÃO DO PROCESSO

O fluxo geral do trabalho de modelagem é mostrado na Figura 3.17. Como se pode ver, há iterações entre os passos 1 e 2 e dentro do passo 4.

O passo 1 do processo é fazer o Modelo Ambiental, composto de um Diagrama de Contexto e uma Lista de Eventos. Não faz diferença por qual começar; podem ser feitos iterativamente.

Ao fazer o diagrama de contexto, deve-se observar o seguinte:

- cada fluxo de entrada deve ser necessário para o sistema:
  - . ou reconhecer que um evento ocorreu,
  - . ou produzir uma resposta a um evento,
  - . ou ambos;
- cada fluxo de saída deve ser uma resposta a um evento.

Ao fazer a lista de eventos, deve-se observar o seguinte:

- cada evento não temporal deve ter um fluxo de entrada do qual o sistema possa detectar que o evento ocorreu (evento temporal é aquele que suscita uma ação do sistema pelo decorrer do tempo. Não há fluxo de entrada associado a ele);

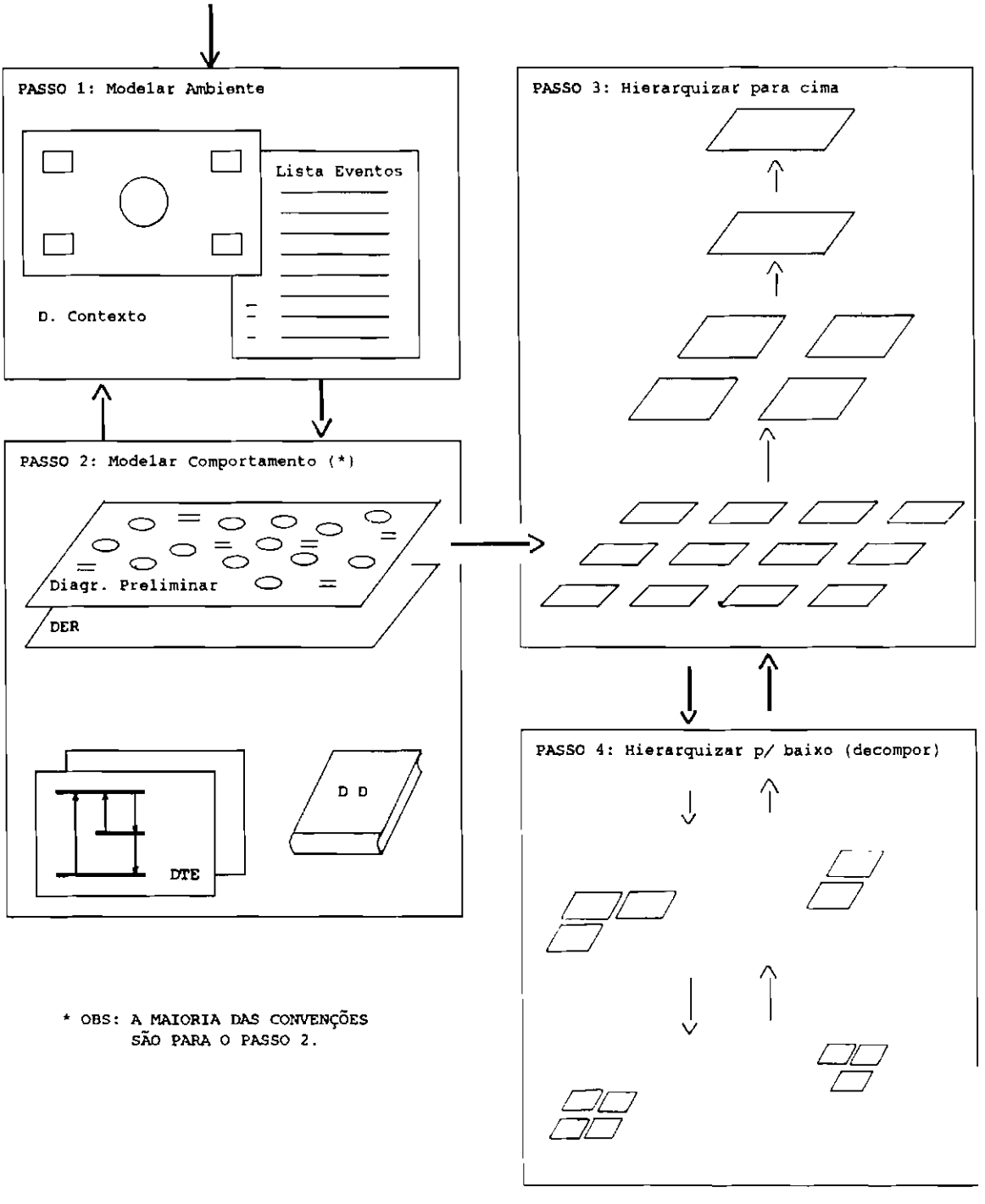


Fig. 3.17 - Método Ward/Mellor - Processo de modelagem.

- para cada evento, o sistema deve, como resposta:

  . ou produzir uma saída,

  . ou armazenar dados a serem exteriorizados mais tarde como uma resposta a algum outro evento,

  . ou mudar seu estado.

O passo 2 da modelagem de Ward/Mellor consiste em derivar o Modelo Comportamental, construindo:

- Um único Diagrama de Transformações Preliminar para todo o sistema,

- Um conjunto de DTEs, um para cada transformação de controle,

- Um único Diagrama de Dados Armazenados (DER) para todo o sistema.

Cabe ressaltar que o Diagrama de Transformações Preliminar é um diagrama num só nível de detalhamento do sistema, o nível de resposta a eventos. É importante destacar que grande parte das convenções se aplicam a esse diagrama, ou melhor, a todo o passo 2. Ward o chama de preliminar porque ele será repartido em vários diagramas no passo seguinte.

Ao construir o Diagrama de Transformações Preliminar, deve-se observar que:

- Cada evento da Lista de Eventos tem ao menos uma transformação (de dados ou de controle) no diagrama cujo trabalho é produzir a resposta requerida ao evento.

- Uma resposta a um evento pode ser armazenada num depósito de dados para servir à resposta a algum evento posterior.

Deve-se fazer repetidas revisões no Modelo Comportamental até que ele seja coerente internamente e coerente com os requisitos do cliente/usuário. Revisa-se o Diagrama de Transformações Preliminar, os DTEs e o Diagrama de Dados Armazenados. Para tais revisões, pode-se:

- fazer entrevistas com o cliente/usuário (só usando o diagrama de contexto),

- "executar" o Diagrama de Transformações Preliminar, de maneira parecida com a execução de uma rede Petri, para verificar o comportamento ou para prototipar o sistema (prototipar requer especificações para cada transformação de dados).

Estando o Modelo Comportamental todo coerente, está terminado o passo 2.

O passo 3 da modelagem consiste em um trabalho de hierarquizar o Modelo Comportamental, agrupando transformações de modo a criar níveis acima do nível de detalhe do Diagrama de Transformações Preliminar. É um trabalho quase mecânico, que pode ser descrito pelo seguinte algoritmo:

1) Partir do nível do Diagrama de Transformações Preliminar.

2) Agrupar transformações do nível atual, criando um diagrama um nível acima. Fluxos e depósitos são também agrupados. O diagrama do novo nível determina uma divisão do diagrama inicial, em vários diagramas, um para cada transformação.

3) Repetir (2) para o novo nível até obter um só diagrama de transformações que seja um primeiro detalhamento do Diagrama de Contexto.

4) Refazer o Diagrama de Contexto com os fluxos agrupados obtidos em (3).

5) Particionar o Diagrama de Dados conforme o uso de depósitos pelos diagramas de transformações da hierarquia.

O passo 4 trata de completar o Modelo Comportamental detalhando, quando necessário, transformações de dados em mais diagramas.

3.2.2 - HEURÍSTICAS E DIRETRIZES

As heurísticas para os passos 1 e 2 da modelagem (ver Figura 3.16) são as seguintes:

a) Modelar baseado no ambiente

O Modelo Comportamental é derivado não por decomposição, mas por um mapeamento dos eventos para transformações que ou o reconhecem, ou respondem a ele, ou uma de cada tipo. Esse mapeamento é feito pensando-se no sistema como um mecanismo de estímulo e resposta.

b) Modelar baseado na natureza do problema

A heurística é que somente as porções do ambiente do sistema embutido que sejam mais vinculadas à natureza do problema devem influenciar a estrutura do Modelo Comportamental e os nomes usados nele.

O método é voltado para sistemas "embutidos" ("embedded systems"); que são aqueles que fazem parte de, ou contribuem para sistemas maiores, cujo propósito primário não é computação e que empregam outro tipo de tecnologia além da de computadores. Ex: sistema de controle de tráfego aéreo. Estes sistemas maiores, dos quais os sistemas embutidos fazem parte, podem ser caracterizados por uma classificação de seus componentes, como segue:

- Classe a: Objetos do espaço de percepção/atuação do sistema (embutido). São entidades quaisquer percebidas pelo sistema

embutido ou sobre as quais ele atua.

- Classe b: Sensores e atuadores do sistema embutido.

- Classe c: Ligações (ou meios) de comunicação.

- Classe d: Processadores (computadores e pessoas).

Esta heurística visa complementar a primeira (a de se basear no ambiente), pois nem todos os elementos do sistema maior são de igual relevância para o sistema embutido nele. As classes acima estão ordenadas pelo seu grau de vinculação --ou de especificidade-- com a natureza do problema; da maior vinculação para a menor, a dos processadores. Por exemplo, a escolha das entidades externas é sempre feita entre elementos nas classes (a) e (b). A identificação de eventos externos é guiada pelo que acontece com tais entidades, e todo o Modelo Comportamental deriva da lista de eventos.

c) Considerar a perspectiva dominante

Há duas perspectivas de descrição do sistema no Modelo Comportamental:

1) Perspectiva ativa, dada pelo Diagrama de Trasformações Preliminar, que mostra o sistema como um mecanismo para transformar entradas em saídas.

2) Perspectiva passiva, dada pelo Diagrama de Dados Armazenados. Segundo os autores, essa perspectiva mostra o sistema em termos dos dados que ele deve armazenar sobre as entidades do seu ambiente, e das associações existentes entre elas (relacionamentos).

Conforme o sistema, predomina uma, outra ou ambas têm igual relevância. Isso poderá ser visto na lista de eventos, que terá a maioria dos eventos de um tipo.

A heurística é que focar inicialmente na perspectiva dominante facilita a compreensão do sistema.

d) Como modelar respostas a eventos

Geralmente convém representar respostas a eventos como transformações de dados quando:

- o evento é percebido pela chegada de um fluxo de dados discreto no tempo,

- a expressão da resposta se aplica a muitos valores de dados possíveis,

- entidades foram escolhidas dentre objetos do espaço de percepção/atuação do sistema,

- há muitas ocorrências da entidade no ambiente.

Ao contrário, geralmente convém representar respostas a eventos como uma transição de estado quando:

- o evento é associado com fluxos de dados contínuos que requerem um mecanismo de reconhecimento de eventos, ou o evento é sinalizado pela chegada de fluxos de eventos,

- a expressão da resposta é significativamente diferente para diferentes valores de dados,

- entidades externas consistem de tecnologia de sensores e/ou atuadores no ambiente do sistema,

- há poucas ou uma única ocorrência da entidade no ambiente (representada direta ou indiretamente pela tecnologia de sensores/atuadores).

As heurísticas acima se aplicam aos passos 1 e 2. Para o passo 3, que é um trabalho de hierarquizar o modelo, o método tem as seguintes diretrizes:

a) Refletir a estrutura do ambiente no modelo

Assim como acontecia no diagrama preliminar, os agrupamentos dos níveis mais altos devem continuar a refletir a estrutura do ambiente no qual o sistema opera. Para tanto, pode-se usar:

- grupos relacionados pelas respostas (ex.: transformação que reconhece junto com a que responde a um evento).

- grupos relacionados à mesma entidade externa.

Deve-se também suspeitar de grupos que não possam ter um nome relacionado à natureza do problema.

b) Facilitar a compreensão do modelo

O modelo hierarquizado deve ser o mais fácil possível de compreender e verificar (pelo cliente). Para tanto, há duas diretrizes:

- Particionar para minimizar interfaces.

- Identifique hierarquias de controle e incorpore-as à hierarquia do modelo (uma hierarquia de controle é uma transformação de controle controlando uma ou mais transformações de controle).

As heurísticas para o passo 4, de detalhamento de transformações, são as seguintes:

- Basear-se nas estruturas dos dados a serem transformados, conforme Jackson (1975).

- Basear-se na "dispersão" dos dados a serem transformados (ex.: alguns podem estar armazenados e outros serem fornecidos por um operador).

- Evitar decomposição funcional, devido ao risco de polarizá-la com idéias de implementação.

CAPÍTULO 4

COMPARAÇÃO

Este capítulo apresenta uma comparação dos dois métodos descritos. Tirando proveito da organização das descrições feitas nos Capítulos 2 e 3, os recursos de modelagem de cada método são comparados: estrutura, notação, convenções, processo de modelagem, e heurísticas do método. Também são comparadas as abordagens.

Esta comparação é intrínseca, isto é, não se está comparando segundo um critério alheio à capacidade de se representar requisitos com os métodos em questão. Por exemplo, não se está comparando ainda qual é o mais adequado para modelar requisitos de sistemas aviônicos ou de defesa. Os critérios usados são:

- quais recursos de modelagem são equivalentes, i.e., expressam o mesmo conceito,

- quais recursos são semelhantes mas não equivalentes (exemplo: mesmo símbolo para conceitos diferentes),

- quais recursos estão presentes em apenas um dos métodos.

O objetivo de comparar segundo estes critérios é facilitar a combinação dos métodos a ser feita; esta sim seguindo critérios de adequabilidade à modelagem de requisitos para os sistemas visados no Capítulo 5. Porque, ao combinar os métodos, as decisões dependerão dos três critérios acima:

- para recursos equivalentes nos dois métodos, deve-se decidir apenas se devem ser incluídos ou não no método combinado.

- Para recursos não equivalentes mas semelhantes deve-se decidir, nesta ordem:

1º) se os dois conceitos diferentes devem ser incorporados, ou só um, ou nenhum,

2º) no caso de incorporar os dois conceitos, como diferenciá-los.

- Para os recursos presentes em um dos métodos apenas, como não há nada comparável no outro método, deve-se decidir se o recurso deve ser incluído no método combinado ou não --e isso envolve entender porque num dos métodos ele não foi considerado necessário.

A seguinte simbologia é usada na comparação:

- equivalente: <=>

- não equivalente: =#=

- parcialmente equivalente: ±

- não tem correspondente: _____> Ø

4.1 - ABORDAGENS

A comparação das abordagens está na Figura 4.1, que mostra as diferenças e semelhanças entre elas. Como é natural, as diferenças na abordagem determinam todas as principais diferenças no processo de modelagem, nas heurísticas e nas convenções. Notação e estrutura do modelo são bem menos afetadas pela abordagem, exceto num item importante da estrutura -- a tabela de tempos de resposta, que Hatley usa e Ward/Mellor não.

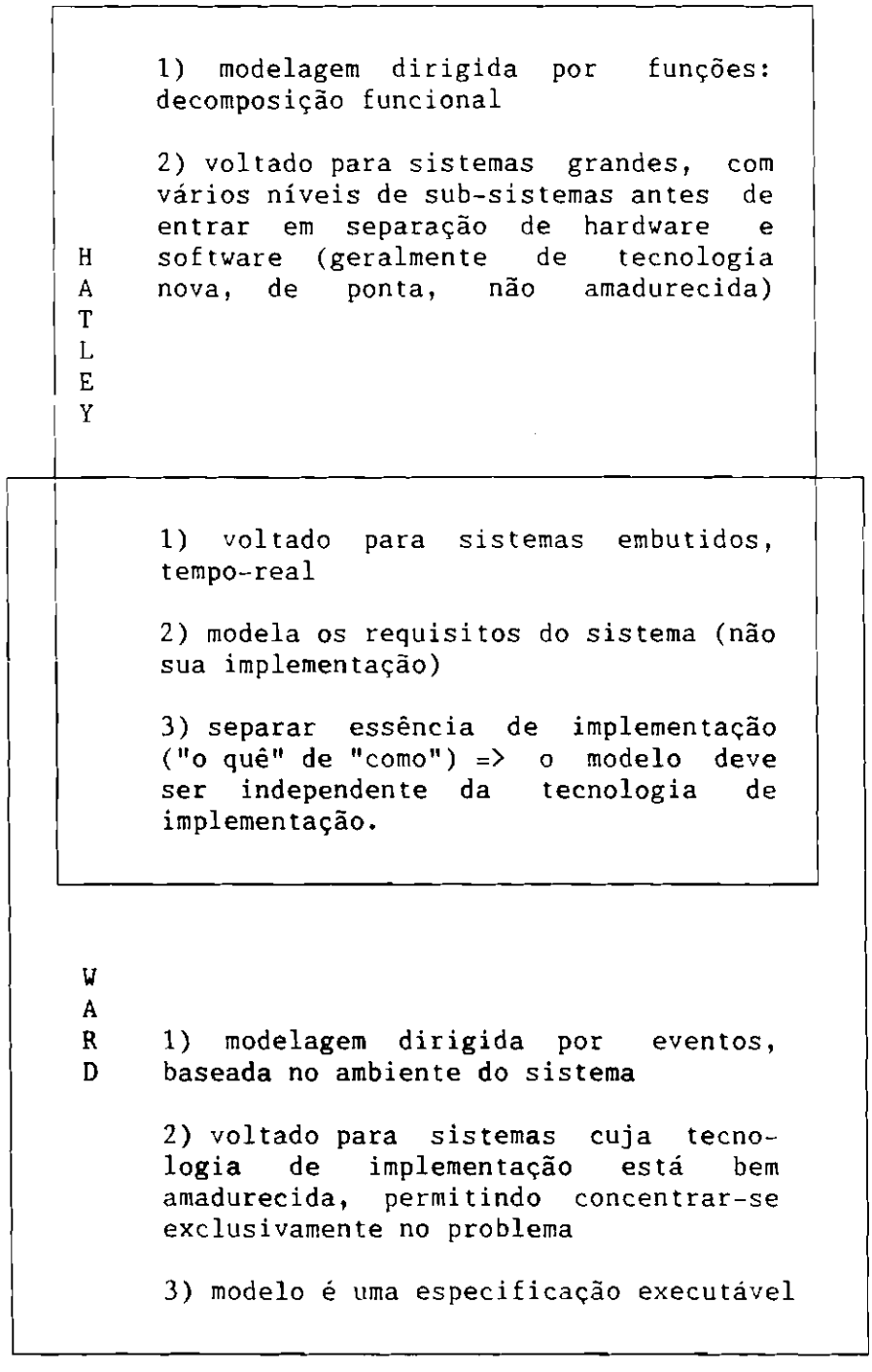


Fig. 4.1 - Abordagem Hatley versus Ward/Mellor.

4.2 - HEURÍSTICAS

Nas heurísticas dos dois métodos, há várias semelhanças e diferenças. As semelhanças são:

- abstrair-se de considerações de projeto,

- ênfase na escolha criteriosa dos nomes para bolhas, fluxos, estados, etc.,

- particionar/agrupar par minimizar interfaces,

- as hierarquias de controle são incorporadas à hierarquia do modelo. No método Hatley esta heurística aplica-se ao longo de todo o processo de modelagem; no método Ward/Mellor, aplica-se à etapa específica para hierarquizar o modelo.

Quanto às diferenças nas heurísticas, estas são mostradas na Tabela 4.1.

TABELA 4.1

HEURÍSTICAS DIFERENTES

HATLEY	WARD
- Desenfatizar controle. Analogia com malha de controle por realimentação: 1º) modelar processamento de dados (DFDs/PSPECs), a parte controlada. 2º) modelar a parte controladora (CFDs e CSPECs).	- Modelar primeiro a perspectiva que for dominante dentre: . processamento de dados (funções) . controle (máq. estados) . dados armazenados Dominância pode ser derivada da: . natureza do problema, ou do sistema, . quantidade de eventos de cada tipo na lista de eventos.
- Procurar manter os diagramas simples (princípio dos 7 ± 2 de Miller (1956)).	- Ao modelar (no passo 2): modelar baseado nos eventos externos e no ambiente; não visa limitar TAMANHO do diagrama preliminar. (Resta questionar se isso não é outra maneira de se obter simplicidade.) Passos 3 e 4: o diagrama de nível mais alto deve continuar a refletir a estrutura do ambiente.

4.3 - PROCESSOS DE MODELAGEM

Quanto à mecânica dos processos, são claramente diferentes, basta ver as Figuras 2.13 e 3.16. É mais interessante comparar aqui os processos de modelagem quanto ao trabalho mental feito pelo modelador por seguir este ou aquele processo aliado à sua abordagem.

Devido às abordagens (uma, por funções, outra, por eventos), os processos são diferentes e quase totalmente incompatíveis. O de Hatley envolve constantes vaivéns para cima e para baixo na estrutura, fazendo abstrações de detalhes ao subir, e detalhamentos ao descer e ao especificar dados no dicionário. O de Ward/Mellor envolve identificação de eventos externos e dos processamentos necessários para responder a eles. A hierarquização de Ward não faz parte da verdadeira modelagem, feita no passo 2; ela envolve abstração de detalhes ao se fazer o trabalho de agrupamento de bolhas. Também requer detalhamentos, no passo 4. Ward na verdade não dá heurísticas fortes para se fazer detalhamento. Ao abandonar a decomposição funcional pela modelagem por eventos ele criou os passos 1, 2 e 3; mas ele não supriu o passo 4 com heurísticas tão fortes quanto às dos demais passos.

A diferença realmente importante entre os dois processos de modelagem refere-se a abstrações de detalhes, feitas ao se decompor ou agrupar funções: como mostra a Figura 4.2, no método Hatley, por se trabalhar na hierarquia de diagramas, tem-se que identificar funções ao mesmo tempo que se faz abstração de detalhes; enquanto que no método Ward/Mellor, quando se faz identificação de funções (passo 2) não se fica mudando de nível de abstração de detalhes, isto é feito nos passos 3 e 4. Ora, tanto a identificação de funções quanto abstrair-se de detalhes, em sistemas da área de engenharia, podem ser árduos trabalhos mentais.

Quanto ao trabalho mental ser maior ou não no método Hatley, resta saber se identificar funções é mais ou menos difícil que identificar eventos e trabalhar num diagrama com centenas de bolhas.

Yourdon (1988) diz que trabalhar no diagrama de Ward/Mellor não é muito difícil, porque os analistas envolvidos podem trabalhar por regiões do diagrama. Eles dividem entre si as entidades externas, e a região de cada analista será aquela que contém bolhas que interagem com as "suas" entidades. No entanto, ele se referia à modelagem de sistemas tipicamente comerciais, de processamento de dados, usando o método Ward/Mellor. Este é o ramo mais conhecido e trabalhado no desenvolvimento de sistemas de software, onde os sistemas típicos são desenvolvidos em prazos tipicamente menores que dois anos, e os requisitos são modelados por dois a quatro analistas. No caso de sistemas tempo-real de defesa, por exemplo, os projetos levam mais de cinco anos, sendo o desenvolvimento normalmente dividido entre algumas empresas; é outra escala de grandeza.

Embora haja diferenças entre os métodos sobre quando são feitas abstrações de detalhes, há uma semelhança entre eles ligada à heurística de abstrair-se de considerações de projeto. Em ambos os métodos, esta heurística se aplica ao longo de todo o processo de modelagem. Examinemos então, na Figura 4.2, sua aplicação ao longo de cada um dos processos. A abstração de considerações de projeto é um trabalho mental que está constantemente entrelaçado com os trabalhos mentais envolvidos ao modelar os requisitos essenciais; e esse entrelaçamento é diferente em cada etapa da modelagem. No método Ward/Mellor ela fica desvinculada dos processos mentais do passo 3.

Resumindo, a mecânica dos dois processos de modelagem é claramente diferente. Esta mecânica, em cada um dis métodos, aliada às suas heurísticas, leva a diferentes atividades mentais pelo(s) modelador(es). Estas por sua vez, são complexas e sua análise mais elaborada está fora da abrangência deste trabalho.

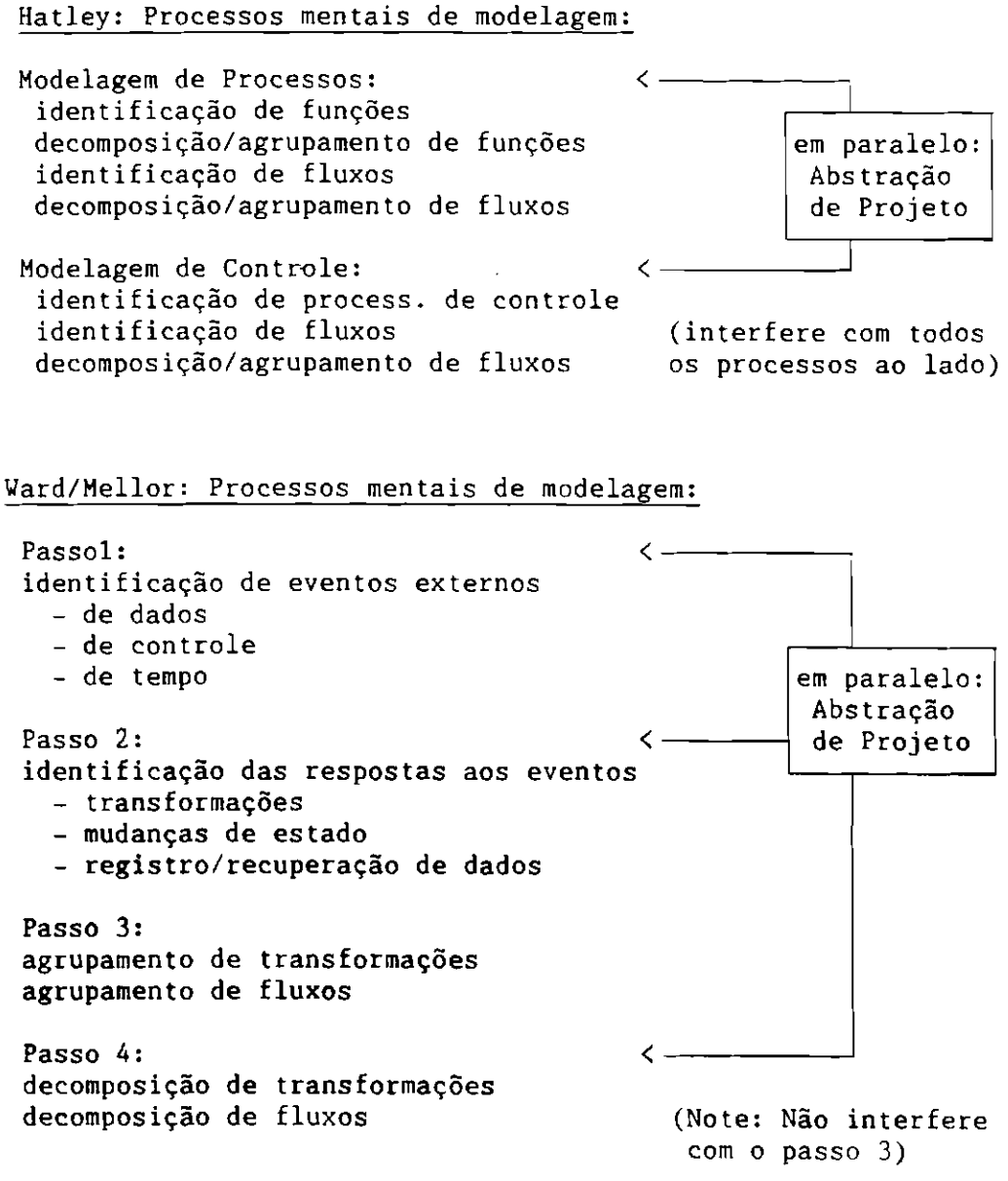


Fig. 4.2 - Trabalhos de abstração nos dois métodos.

4.4 - ESTRUTURAS

Os vários componentes das estruturas de cada método são confrontados na Tabela 4.2.

TABELA 4.2

COMPARAÇÃO DAS ESTRUTURAS

HATLEY	<=> ou =#=	WARD
- Diagramas de Contexto (de dados e de controle)	<=>	- Diagrama de Contexto
- Especificação de Tempos de Resposta (mais as especificações de freqüências de fluxos de entrada e saída no RD)	.	————> Ø
Ø <————	.	- Lista de Eventos
- DFDs + CFDs	<=>	- Diagr. de Transformações
- CSPECs	<=>	- Especs. Transf. Controle
- PSPECs	<=>	- Especs. de Transf. Dados
- Dicionário de Requisitos	<=>	- Dicionário de Dados
Ø <————	.	- Diagramas de Dados Armazenados (DER)

4.5 - NOTAÇÃO

As notações usadas em cada método são comparadas na Tabela 4.3. Note que duas notações são consideradas iguais quando tanto o símbolo quanto o significado são os mesmos.

TABELA 4.3

COMPARAÇÃO DAS NOTAÇÕES

Item	Hatley X Ward	Observação
de DFD/CFD:		
.Entidade Externa	Iguais	
.Processo/ Transf. Dados	Iguais	
.Fluxo de Dados Discreto	Iguais	Significado: dados sendo transmitidos.
.Fluxo de Dados Contínuo	Ø <———	
.Fluxo de Controle	<=>	
.Barra de CSPEC/ Transf. Controle	<=>	A bolha permite dar nº e nome, e decompor
.Depósito de Dados	Iguais	
.Depósito Controle	<=>	
Diagr. Transição de Estados (DTE)	Iguais	
Espec. Controle Combinatorial	———> Ø	
Diagrama de Dados Armazenados	Ø <———	
Dicionário de Dados	<=>	

4.6 - CONVENÇÕES

As convenções são comparadas, em sua maioria, na Tabela 4.4. Além das convenções comparadas na Tabela 4.4, as seguintes comparações são feitas aqui, fora da tabela, por ser mais conveniente.

a) Hierarquização de elementos

Na Figura 4.3 são comparados como se agrupam elementos de um nível no nível acima.

b) Processo disparado por dados

Quanto a processos disparado por dados, o método Hatley é equivalente ao método Ward/Mellor, quando neste se usa as convenções para execução "como protótipo", isto é, usando as P-specs sem restringir as transformações a serem "síncronas". No entanto, usando convenções para execução "esquemática" (isto é, diagrama com todas as transformações "síncronas", e sem usar nenhuma P-spec), o método Ward não é equivalente.

c) Ativação e Desativação de processos.

Este aspecto tem convenções equivalentes nos dois métodos. No método Hatley, os fluxos ativar/desativar (que não são desenhados nos CFDs) equivalem aos fluxos "enable/disable" do método Ward/Mellor.

d) Fluxos de Dados

As convenções para interpretação de fluxos de dados estão comparadas na Figura 4.4. É interessante notar que, enquanto para fluxos de dados há tanta variedade de interpretação, para fluxos de controle a interpretação é única, e a mesma nos dois métodos: o caso (a) da Figura 4.4, fluxos com valores de variação discreta, emitidos em pontos discretos no tempo.

TABELA 4.4

COMPARAÇÃO DAS CONVENÇÕES

Assunto	<=> ou =#=	Hatley nº conv.	Ward nº conv.	Observações
Numeração e nomes dos diagramas	<=>	2.1.3.1	3.1.3.1.b	
Disparo de transf/processo	±	2.1.3.2.a	3.1.3.2.a	
Dinâmica de ativação de processo	±	2.1.3.2.b	3.1.3.3.c	
Regras de ativação de processo	±	2.1.3.3	3.1.3.3.b & '' .c	O fato de Hatley trabalhar na hierarquia e Ward num só plano afeta as diferenças.
Ativação de processo com a chegada de entradas de dados	=#=	2.1.3.2.a	3.1.3.6.e	Na verdade, para o Ward, 3.1.3.6.e visa execução do modelo usando somente o diagrama, sem "pspecs".
Dualidade dado x controle	=#=	2.1.3.4.a	3.1.3.2.d	
Convergência/divergência de fluxo	±	'' .b	3.1.3.2.b	Aplicações diferentes. Hatley faz uso mais diversificado.
Fluxo de dados contínuos	=#=	'' .e	3.1.3.2.c	Ward usa notação específica.
Fluxo de controle é discreto	<=>	'' .g	definição	Mesmo conceito.
Fluxo de controle primitivo: quantidade de valores	=#=	2.1.3.4.h &2.1.3.3.d	definição &3.1.3.3. .b.3	Hatley: N-ário, ou (usual/te) binário. Condensa, simplifica o modelo. Ward: 1 só valor (evento ocorreu).
Fluxos de controle compostos: (hierarquizar)	<=>	2.1.3.4.b	3.1.3.1.a	Hatley tem algumas convenções a mais.
Acesso à informação de tempo.	<=>	2.1.3.5.a	definição	(de evento temporal)
Em que nível mostrar depósito	=#=	2.1.3.6.b	3.1.3.4.d	
Fluxos ligados a depósitos	=#=	2.1.3.6.c & '' .d	3.1.3.4.a	
Uso de depósitos de controle	=#=	2.1.3.6.f & '' .g	3.1.3.5	HATLEY: uso mal definido pela CSPEC. Pode necessitar intercalar uma transf. de DADOS. WARD: Bem definido seu uso: semáforo.
Uso de depósitos de dados no modelo. Implicações.	=#=	2.1.3.6.e	3.1.3.2.a &3.1.3.4.c	
Papel de transf. de dados e de controle	<=>	2.1.3.7 &2.1.3.9.a & '' .b	3.1.3.6.b & '' .c & '' .d	Isto é: DFD/CFD <=> Diagr. Transform.
Uso do RD / DD	<=>	2.1.3.8	3.1.3.10	Ward define mais claramente a especificação de depósitos, devido a usar o DER

(continua)

Tabela 4.4 - Conclusão.

Assunto	<=> ou =#=	Hatley nº conv.	Ward nº conv.	Observações
Abrangência de uma especificação de controle	?	2.1.3.9.b	— › Ø	Ward: não necessariamente se obtém o mesmo efeito que no Hatley ao se montar a hierarquia. Ao seguir as diretrizes para o passo 3, no entanto, pode ser que se obtenha o mesmo efeito.
Organização de CSPEC grande	-	2.1.3.9.c	— › Ø	
Uso da teoria de máq. estados finitos.	=#=	2.1.3.9.f & '' .g	3.1.3.8	Ward: Só máq. sequencial. Hatley: Sequencial e combinatorial.
Modelo usado de máquina de estados finitos sequencial (DTE)	±	2.1.3.9.d	3.1.3.8.f	HATLEY: Principalmente Mealy. Às vezes híbrido com Moore. WARD: Só Mealy.
Desativação ao mudar de estado no DTE.	=#=	2.1.3.9.e	3.1.3.8.e	Ward: explícita. Hatley: implícita.
E/S de processo primitivo	<=>	2.1.3.10.b +(ver obs)	3.1.3.3. .b.2	Hatley: Mais a def. de condição de dados
Inglês Estruturado (Pspec)	=#=	2.1.3.10.e '' .f	3.1.3.7.a	Hatley usa bem; até p/ concorrência. Ward não valoriza, prefere pre/pós-condições.
Pspec com pre/pós-condições	—	Ø ‹ ——	3.1.3.7.c	Inviável para transf. não triviais.
Tabelas, em Pspecs.	<=>	2.1.3.10.c	3.1.3.7.a '' .b	
Uso do DER	—	Ø ‹ ——	todas	
Controle externo contínuo	=#=	Ø ‹ ——	3.1.3.2.d	Hatley: controle externo só por CSPEC.
Alteração instantânea do conteúdo de depósitos	<=>	2.1.3.4.e qto a dep.	3.1.3.4.b	
Múltiplos subsistemas equivalentes	—	Ø ‹ ——	3.1.3.6.f	
Idealizações do modelo	<=>	2.1.3.2.a	3.1.3.6.g &3.1.3.8.c	
Como caracterizar estado	<=>	2.1.3.9.h	3.1.3.8.a '' .b	Ward define melhor
Estado transitório no DTE	—	Ø ‹ ——	3.1.3.8.g	

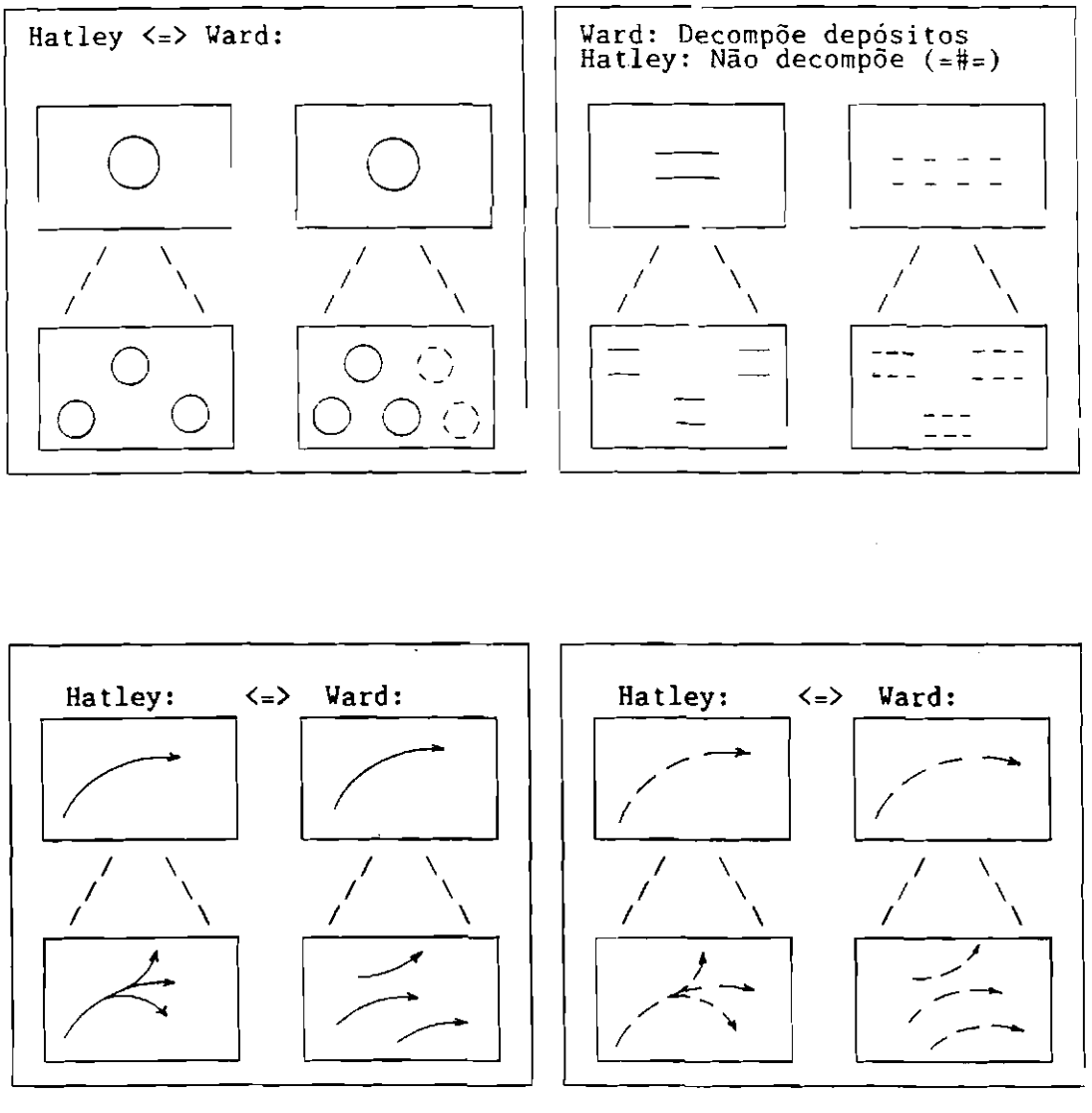


Fig. 4.3 - Comparação das convenções de hierarquização.

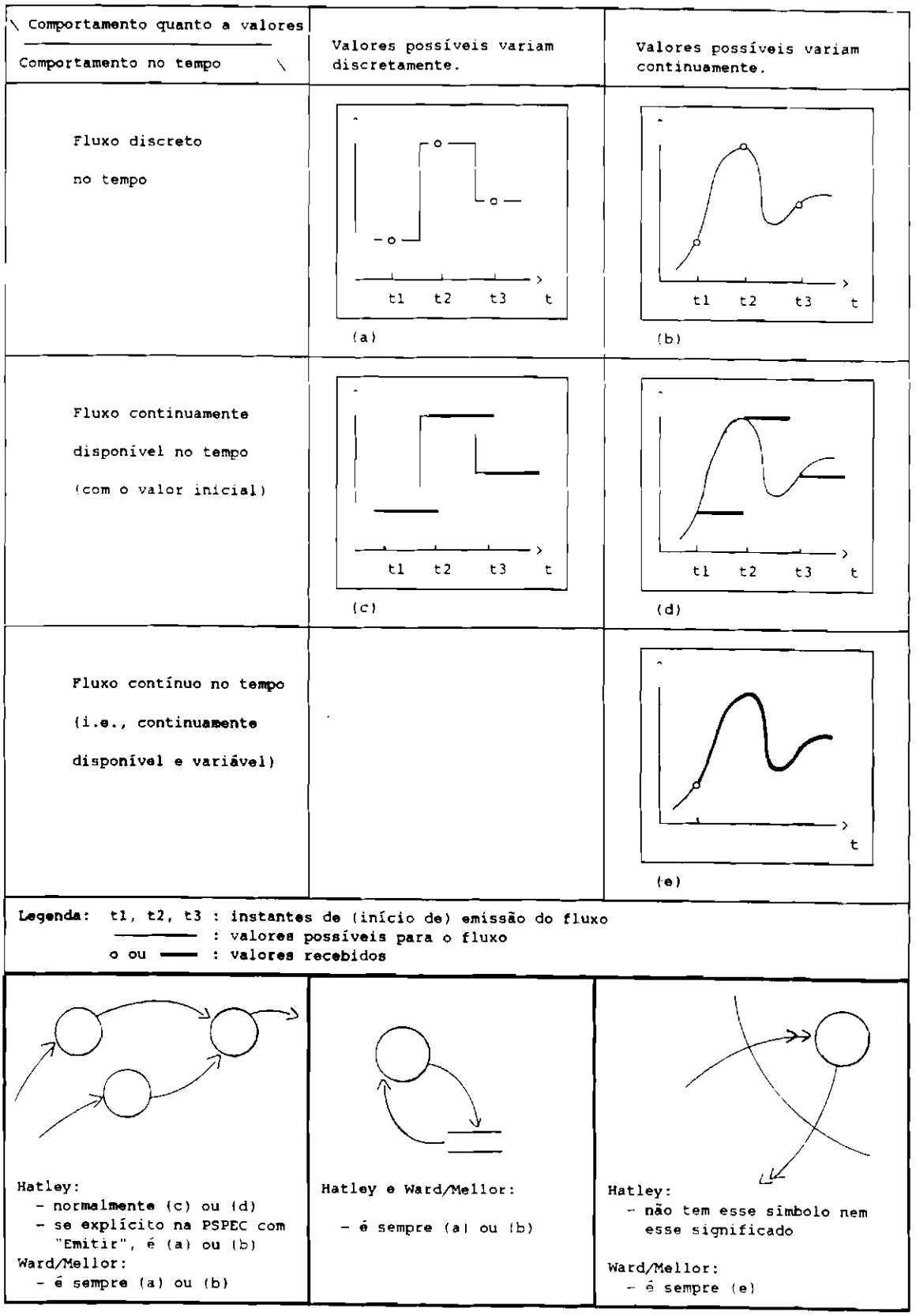


Fig. 4.4 - Comparação de convenções para fluxos de dados.

CAPÍTULO 5

SISTEMAS TEMPO-REAL DE GANDE PORTE: FATORES E CRITÉRIOS PARA COMBINAÇÃO DOS MÉTODOS

Este capítulo estabelece os critérios principais para a combinação dos métodos deste trabalho, procurando mostrar a ligação dos métodos à realidade das aplicações visadas. O objetivo principal desta combinação é aproveitar as técnicas de cada método que melhor sirvam para o desenvolvimento de sistemas tempo-real de grande porte. Nesta classe de sistemas incluem-se geralmente aplicações de engenharia, tais como: sistemas de defesa, sistemas de aviônica civil e militar (sistemas de gerenciamento de vôo, sistemas de piloto automático, sistemas de navegação, sistemas de controle de disparo de armas etc.), sistemas de comunicação, e sistemas de testes dos anteriores.

Sistemas desse tipo, principalmente os de aviônica, têm algumas características peculiares:

a) Intenso processamento de dados - Boa parte do processamento geralmente é cíclico, pois tipicamente as entradas variam continuamente no tempo e as saídas devem ser atualizadas a frequências tipicamente de 20, 50, 100 Hz, para simular continuidade no tempo. Além dessas entradas contínuas, entradas que ocorrem discretamente são processadas em tempos de resposta tipicamente curtos, da ordem de 1, 10, 100 milissegundos. O atrazo na resposta muitas vezes é crítico.

b) Múltiplos modos de operação (estados) - caracterizados por diferenças no processamento de dados feito em cada modo. Cada modo ativa ou combina certos tipos de processamento. (Exemplo: certos sistemas de piloto automático têm algo como 6 modos de vôo na horizontal e 8 na vertical, podendo haver inúmeras combinações). Esta característica resulta em que a resposta do sistema a um dado estímulo depende muito das condições atuais do ambiente, dos eventos passados e eventos futuros previstos.

c) São desenvolvidos geralmente em paralelo com sistemas com os quais irão interagir (formando um sistema maior) - Isto implica em que as interfaces com esses sistemas estão em processo de definição, ou seja, o "ambiente" do sistema não é predefinido, o que resulta em constantes mudanças nos requisitos (tanto requisitos de sistema quanto de software).

d) O desenvolvimento de hardware muitas vezes é feito em paralelo com o desenvolvimento de software - Mudanças nos requisitos de hardware afetam os de software, principalmente na porção do software que lida com interfaceamento, justamente o aspecto mais crítico de qualquer sistema. Este problema tem sido resolvido em grande parte com o crescente uso de padrões de interfaceamento e comunicação.

e) Utilizam tecnologia geralmente recente e a mais sofisticada possível - Nas aplicações civis, a concorrência manda que a tecnologia que gera mais benefícios a custos suportáveis (ex.: maior segurança num vôo de passageiros, ou maior número de ligações por minuto num sistema telefônico), venda mais. Nas aplicações militares, é uma questão de estratégia usar tecnologia recente, desconhecida por nações potencialmente hostis.

f) Tais sistemas são cada vez maiores e mais complexos - Visando facilitar a utilização da tecnologia empregada sem sobrecarregar o usuário (operador, controlador ou piloto), o software em tais sistemas incorpora cada vez maior carga de processamento e de complexidade decisória, de modo a continuar deixando para o homem somente as decisões críticas.

Consideremos agora as características dos dois métodos básicos para a combinação:

- O método Hatley visa sistemas de grande porte (foi desenvolvido para especificar um sistemas de controle de vôo para o Boeing

737-300, que se tornou equipamento básico também nos modelos 400 e 500. O método foi adotado na Boeing como padrão interno e passou a ser imposto como padrão para todos os seus fornecedores de equipamentos aviônicos). Mas tem características meio antigas, como decomposição funcional, que tornam a modelagem muito dependente da experiência do modelador.

- O método de Ward/Mellor tem bons conceitos úteis, mas é mais voltado para aplicações do tipo de controle de processos. Eis o que escrevem sobre sua abordagem: "a tecnologia atual de computação tem ampla capacidade de resolver um grande conjunto de problemas tempo-real tipicos. [...] Uma abordagem efetiva para o desenvolvimento de sistemas requer uma avaliação das capacidades da tecnologia e da natureza do problema e a escolha de uma abordagem dominada ou pelo problema ou pela tecnologia. As capacidades da tecnologia existente sugerem que uma abordagem dominada pelo problema é apropriada para muitas tarefas do desenvolvimento de sistemas tempo-real" (Ward e Mellor, 1985, Vol. 1, p. 8).

Foi uma escolha de abordagem assim que fizemos para esta combinação. Considerando as características das aplicações típicas visadas, e as dos métodos, adotamos os seguintes critérios básicos para dirigir a combinação dos métodos:

- Condensar informação nos diagramas sempre que possível, sem prejudicar a clareza, buscando simplificá-los; devido a (a), (b), (f).

- Usar conceitos do método Ward/Mellor (e do método de Jackson (JSD)) na estrutura e notação do método Hatley, mais apropriada para sistemas de grande porte.

- Usar as tabelas de tempo de resposta e especificação de frequências no dicionário; devido a (a), (b), (e).

Quanto ao aspecto de quantidade de alterações nos requisitos, as características mais relevantes são (c), (d) e (e). No entanto, a quantidade de alterações nos requisitos é um problema extra-método. Os métodos (sendo diagramáticos, estruturados e com indexação própria) já visam facilidade de alteração, mas por outro motivo: redesenhar os diagramas faz parte do processo de compreender os requisitos; mesmo que eles não mudem. Se mudam muito, tem-se que redesenhar mais vezes ainda. De modo que este aspecto, por grave que seja, não afeta a combinação dos métodos. Afeta mais à escolha de ferramentas de software (CASE) para apoiar o uso do método e à gerência do processo de desenvolvimento.

É ineressante observar que, até recentemente, predominava a especificação textual nas áreas de aplicação de engenharia citadas. O uso das técnicas estruturadas para especificação de requisitos se difundiu muito nas aplicações comerciais. Mas na área de software tempo-real, de aplicações industriais e militares, só na década de 80 vem se difundindo (note, este é o estado da prática; no estado da arte, visto nas conferências sobre software tempo-real, o uso de análise estruturada é uma premissa). As extensões da análise estruturada para tempo-real de Hatley e de Ward/Mellor só chegaram recentemente. De modo que já é um avanço nesta área a generalização que vem acontecendo do uso da análise estruturada de sistemas tempo-real. A tendência futura parece ser o uso de métodos orientados por objetos. Embora o Departamento de Defesa americano, que encomenda a maior parte do software produzido nos EUA, esteja promovendo o uso da linguagem Ada para sistemas embutidos e de defesa em geral, a transição para o uso de Ada é lenta, e não pode ser imposta às empresas (Ada é considerada "semi" orientada por objetos por alguns autores). Historicamente, é a abordagem de uma linguagem que dita a abordagem para o projeto e para a análise de requisitos. Já existem técnicas para projeto arquitetural orientado por objetos apropriadas para Ada (Buhr, 1984; Carter, 1988), mas não ainda para análise de requisitos.

CAPÍTULO 6

O MÉTODO HÍBRIDO HATLEY/WARD/JSD

Este capítulo descreve o método híbrido proposto. Em termos simples, o método Hatley é uma combinação da análise estruturada convencional com o modelo de máquinas de estado mais o uso de CFDs, que seriam DFDs para informações de controle. O método de Ward/Mellor faz uso combinado de três modelos: análise estruturada, máquinas de estado seqüenciais e diagramas de entidades e relacionamentos. O trabalho de combinar métodos para aspectos diferentes de um sistema já foi feito pelos autores dos dois métodos. Nosso trabalho é combinar esses dois métodos já mais parecidos, juntando uma parte apropriada do método JSD (Jackson, 1983) para descrever eventos.

A abordagem adotada para o método híbrido é a de modelagem dirigida por eventos (Ward/Mellor), voltada para sistemas grandes (Hatley). A Figura 6.1 mostra a abordagem adotada, ressaltando a origem de cada aspecto. Comparando a Figura 6.1 com a Figura 4.1, pode-se notar quais aspectos foram removidos para se obter a abordagem do método híbrido.

O Modelo de Requisitos a ser construído é parecido com o do método Hatley, mas adota a idéia de divisão em Modelo Ambiental e Modelo Comportamental. Além dos DFDs, CFDs, PSPECs e CSPECs do método Hatley, o modelo inclui (Figura 6.3):

- especificação de tempos de resposta,

- especificação de requisitos de freqüências no RD, para dados de interface externa,

- diagramas JSD para cada entidade externa, para reforçar o Modelo Ambiental,

- DER,

- uso de CSPEC combinatorial.

O modelo tem duas grandes partes, como o de Ward/Mellor. A primeira parte é um modelo do ambiente, chamado Modelo Ambiental, e a segunda é um modelo do comportamento interno requerido para o sistema, o Modelo Comportamental.

Ao Modelo Ambiental foi acrescentado um diagrama JSD para cada entidade externa, no lugar da lista de eventos. Com isto, tem-se um melhor modelo do ambiente. Os diagramas JSD substituem a lista de eventos com as seguintes vantagens:

- os eventos estão mais organizados, separados por entidade externa e, para cada uma, organizados segundo o diagrama, que mostra a seqüência temporal dos eventos daquela entidade externa,

- a estrutura, ou seja a organização, do diagrama JSD facilita o trabalho de identificação de eventos. Em vez de trabalhar-se com eventos "soltos", como com a lista, vai-se montando a seqüência de eventos no diagrama enquanto que sua própria estrutura ajuda o usuário a lembrar-se, por assim dizer, do evento que vem em seguida,

- facilita verificar se não estão faltando eventos. Ao verificar se uma lista está completa, não há nada que auxilie a memória. Usando o diagrama, sua própria estrutura ajuda a nos perguntarmos se aqui ou ali não estaria faltando algo,

- facilita alterar os eventos e checar se não ficaram incoerências. Importante para qualquer sistema, alterar e checar é mais importante ainda para os sistemas visados pelo método (ver Capítulo 5), para os quais muitas vezes as entidades externas são outros sistemas que estão sendo desenvolvidos em paralelo e, portanto, têm seus requisitos indefinidos ou mudados com muita freqüência.

O Modelo Comportamental é composto de DFDs, CFDs, PSPECs, CSPECs e dicionário, na mesma estrutura do método Hatley, adicionado de um DER.

Dada esta visão geral do Modelo de Requisitos, as próximas seções descrevem em maiores detalhes a combinação feita para se obter o método híbrido.

Antes método como um todo, a Seção 6.1 descreve o uso que se faz, neste método híbrido, dos diagramas de estrutura do método JSD.

É importante ressaltar que as Seções 6.2 e 6.3 não apresentam uma descrição do método híbrido apropriada para se usá-lo. Elas descrevem a combinação feita para se obter o método híbrido. Do mesmo modo, como já ressaltado no início do Capítulo 1, as descrições feitas nos Capítulos 2 e 3 não se prestam à utilização dos métodos. Os Capítulos 2 e 3 registram o resultado de uma "dissecação", de um trbalho de análise dos dois métodos. Este Capítulo 6 registra o resultado de um trabalho de síntese dos métodos referenciando os elementos utilizados, para evitar reescrever boa parte do material dos Capítulos 2 e 3. Partindo da descrição neste capítulo, não fica difícil obter uma descrição didática do método híbrido; mas isto já seria um outro trabalho, além do escopo dessa dissertação.

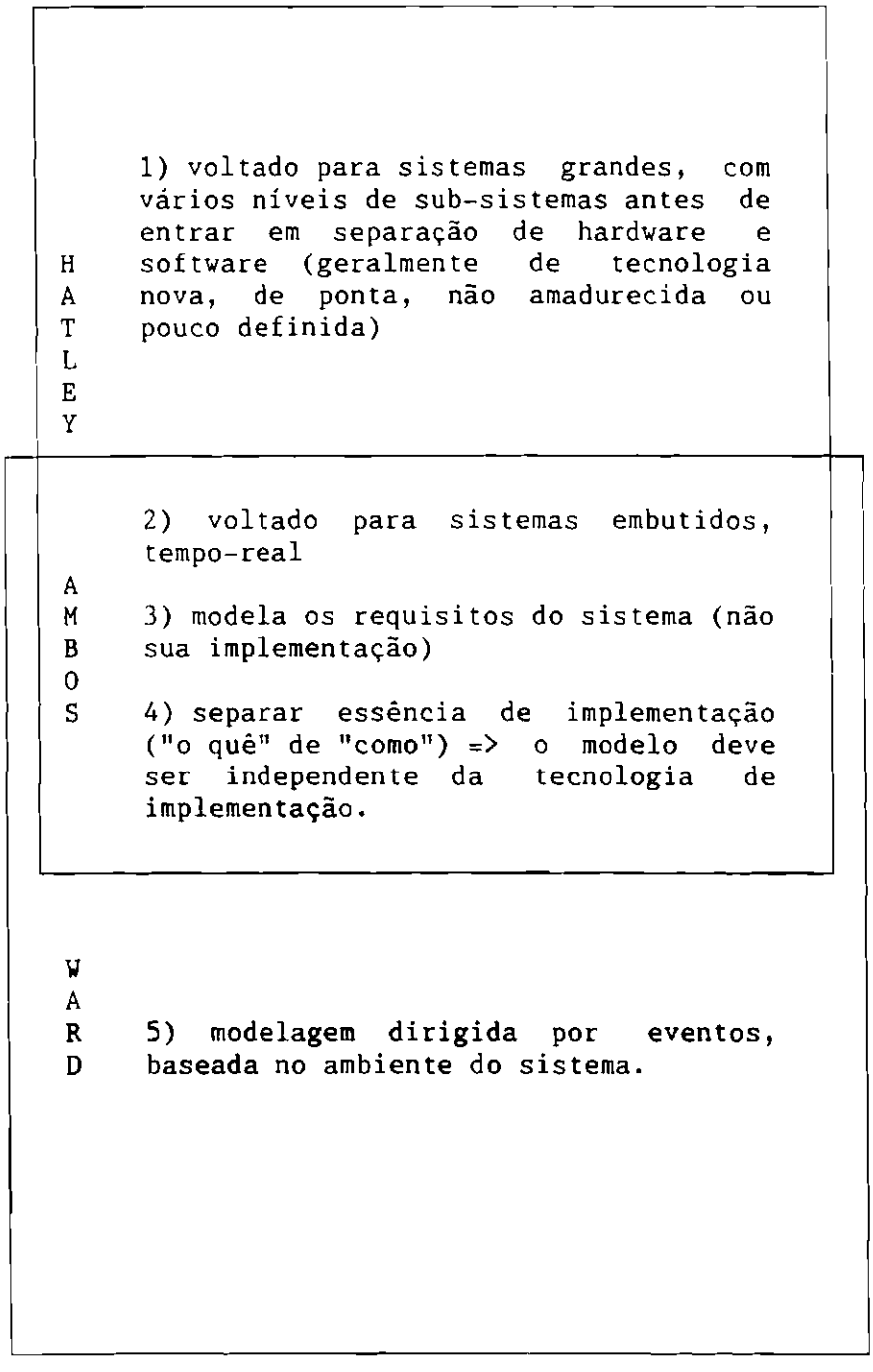


Fig. 6.1 - Abordagem do método híbrido.

6.1 - O USO DOS DIAGRAMAS JSD

Esta seção descreve a modelagem das entidades externas no método híbrido usando-se um recurso do método JSD (Jackson, 1983).

O método JSD tembém é dirigido por eventos, tal como o método Ward/Mellor. Ele se divide em seis etapas bem definidas, cobrindo desde a especificação até a implementação.

Sua abordagem básica de modelagem de requisitos é que nos interessa mais. Ela consiste em fazer primeiramente um modelo, não dos requisitos, mas das entidades externas. As entidades externas realizam ações, que têm uma seqüência ao longo do tempo. Essas ações são percebidas pelo sistema como eventos através das entradas de dados. Jackson modela cada entidade externa com um diagrama como o da Figura 6.2. O diagrama da Figura 6.2 é idêntico ao usado por Jackson para estruturas de dados no método JSP (Jackson, 1975), só que a notação é usada com outro significado. Os retângulos são ações que a entidade executa. Deste modo, tem-se um estrutura dando a seqüência dos eventos de uma entidade externa ao longo do tempo.

Uma estrutura como esta é feita para cada entidade externa. No método JSD, o conjunto dessas estruturas constitui o modelo do ambiente do sistema. Em seguida tais estruturas são convertidas para um formato textual. Este formato textual é visto como um processo que descreve a seqüência de ações das entidades externas. O conjunto destes processos, chamado modelo inicial, passa a constituir um núcleo da especificação do sistema. A este núcleo serão acrescentados requisitos do cliente. Isso pode ser feito ou implantando funções requeridas nos processos do modelo inicial, ou acrescentando novos processos para as funções. O que se obtém é um modelo de requisitos que incorpora dentro dele um modelo do ambiente, e cujo comportamento é casado com o do ambiente.

O que nos interessa mais é o modelo das entidades externas, para juntarmos ao método combinado. As estruturas de eventos

para modelar as entidades externas são muito mais organizadas e "trabalháveis" que uma simples lista de eventos para todas as entidades externas. Ao se trabalhar com uma estrutura gráfica, fica mais fácil descobrir novos eventos, erros e falhas. É realmente um modelo, não uma lista.

Outro aspecto importante é que, como a estrutura é feita em termos de seqüência, seleção e iteração, ela corresponde a um autômato finito, que também poderia ser representado por um diagrama de transição de estados (DTE). Para uma entidade externa, o diagrama JSD parece preferível a um DTE, porque ele enfatiza os eventos. Para o sistema, o que interessa do exterior são os eventos e não os estados.

É interessante notar como esta abordagem para as entidades externas se coaduna bem com a concepção de sistema tempo-real descrita na Seção 1.1. Lá, tinha-se um sistema que interagia com um "processo externo" um tanto nebuloso, do qual, pelo método Ward/Mellor, tem-se uma lista dos eventos que nele ocorrem.

Esta lista não define bem o processo externo; tem-se somente os eventos soltos. Com os diagramas JSD, vê-se que este grande processo externo é composto de vários processos, um para cada entidade externa, cada um dos quais descrito claramente por um diagrama separado, em termos de ações (correspondentes a eventos) seqüenciadas ao longo do tempo. Apesar de o "processo externo" ser uma complexa composição de processos, não se precisa combinar os diagramas JSD. Pode-se usar cada um em separado, mas deve-se sempre ter em mente que todos estes processos se desenrolam simultaneamente ao longo do tempo, o que implica que a qualquer instante o sistema poderá receber um evento de qualquer uma das entidades externas. O efeito desejado, de um evento vindo de uma entidade externa sobre outra, já é requisito funcional para o sistema. Não deve aparecer nos diagramas das entidades. Tais efeitos serão mostrados nos DTEs e no processamento interno ao sistema (seu Modelo Comportamental).

Com estes modelos para as entidades externas, uma sequência previsível é dada aos eventos externos, o que é um avanço em relação à lista de eventos. Essa sequência serve para determinar as sequências dos de estados internos ao sistema nos DTEs, coisa que tinha que ser feita mentalmente ao especificá-los. Ao se determinar os DTEs internos com base nos diagramas JSD para as entidades externas, na verdade estar-se-á fazendo um trabalho equivalente à internalização dos modelos como núcleo do sistema do JSD. Os DTEs têm o mesmo papel, casar a dinâmica do sistema com a do ambiente (lembrar que um estado é um modo de comportamento do sistema externamente observável). Esta é a maneira de conciliar a abordagem do JSD com a da análise estruturada para tempo-real sem transtornar demais os modelos existentes. Usando JSD para representar as entidades externas, tem-se um modelo melhorado do ambiente em termos de autômatos finitos. Sempre que necessário, ao modelar os requisitos, deriva-se os DTEs (internos) dos autômatos JSD para as entidades externas mais os requisitos de interação aplicáveis.

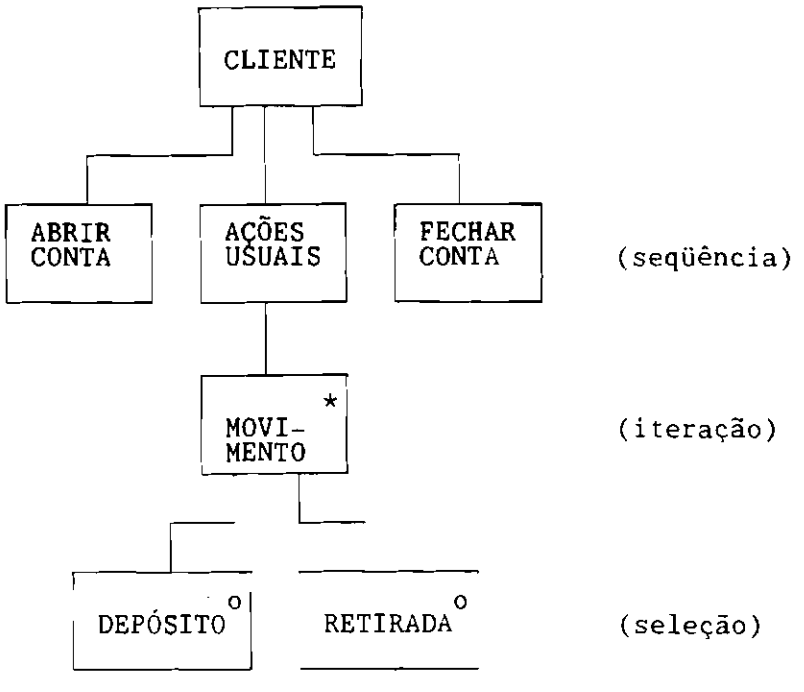


Fig. 6.2 - Modelo de uma entidade externa em termos de suas ações (eventos).

6.2 - O MODELO DE REQUISITOS

Esta seção descreve o Modelo de Requisitos do método híbrido em termos de sua estrutura, sua notação e suas convenções.

6.2.1 - ESTRUTURA

A estrutura do Modelo de Requisitos é ilustrada na Figura 6.3. Os vínculos entre os elementos da estrutura são o balanceamento mais a convenção 6.2.3.a de numeração e nomes dos diagramas. As setas na figura representam o balanceamento entre os componentes. Setas tracejadas representam balanceamento de fluxos de controle. Setas cheias representam balanceamento de fluxos de dados.

Como no método Ward/Mellor, o modelo se divide em duas partes: o Modelo Ambiental e o Modelo Comportamental. O Modelo Ambiental descreve o ambiente no qual o sistema a ser desenvolvido se inserirá; ou nos termos do Capítulo 1, descreve seu "sistema de referência". Ele é composto de:

- DFD + CFD de Contexto: Como qualquer par DFD/CFD, podem ser feitos num só diagrama. Eles mostram quais são as entidades externas, e as interfaces do sistema com ela. Sobretudo, estabelecem os limites do que vai ser desenvolvido; por meio da fronteira entre o que fará parte do sistema e o que será externo.

- Um Diagrama JSD para cada entidade externa: Para descrever a dinâmica de cada entidade externa.

- Especificação de Tempos de Resposta: Ela tem dois papéis. Primeiro, ela correlaciona eventos de estímulo (nas entradas), com seus correspondentes eventos de resposta (nas saídas). Ou seja, quando o sistema detecta um evento externo, ela mostra que evento ele deve produzir como resposta. Segundo, especifica o intervalo de tempo requerido entre um estímulo e sua resposta. A

especificação de tempos define o comportamento externo do sistema sendo modelado dentro do sistema maior no qual se insere.

O Modelo Comportamental descreve o comportamento interno requerido para o sistema de modo que ele satisfaça aos requisitos de comportamento externo contidos no Modelo Ambiental. O Modelo Comportamental é composto de DFDs, CFDs, PSPECs, CSPECs e Dicionário de Requisitos, na mesma estrutura hierárquica do método Hatley. Em termos de estrutura, esta porção do modelo é equivalente à do método Ward/Mellor, como mostrado na comparação das estruturas, Seção 4.4. Sendo assim, acrescentamos à estrutura do Modelo Comportamental um DER, a ser usado se e quando necessário. Se o DER ficar muito grande, ele pode ser repartido correspondentemente aos DFDs, como no método Ward/Mellor.

6.2.2 - NOTAÇÃO

Quanto aos diagramas (DFDs, CFDs, DTEs) a notação deste método é a mesma notação usada no método Hatley, com os seguintes simbolos do método Ward/Mellor adicionados:

- fluxo de dados contínuo (Figura 3.4),

- subsistemas múltiplos equivalentes (Figura 3.8).

Quanto ao RD (Dicionário de Requisitos), não faz muita diferença qual das duas notações usar; a de Hatley e a de Ward/Mellor são totalmente equivalentes. Para ficar coerente com as convenções para o RD, adotaremos a notação do método Hatley (Tabela 2.1), apenas acrescentando o símbolo "@" da Tabela 3.2, do método Ward/Mellor, para uso com elementos do DER.

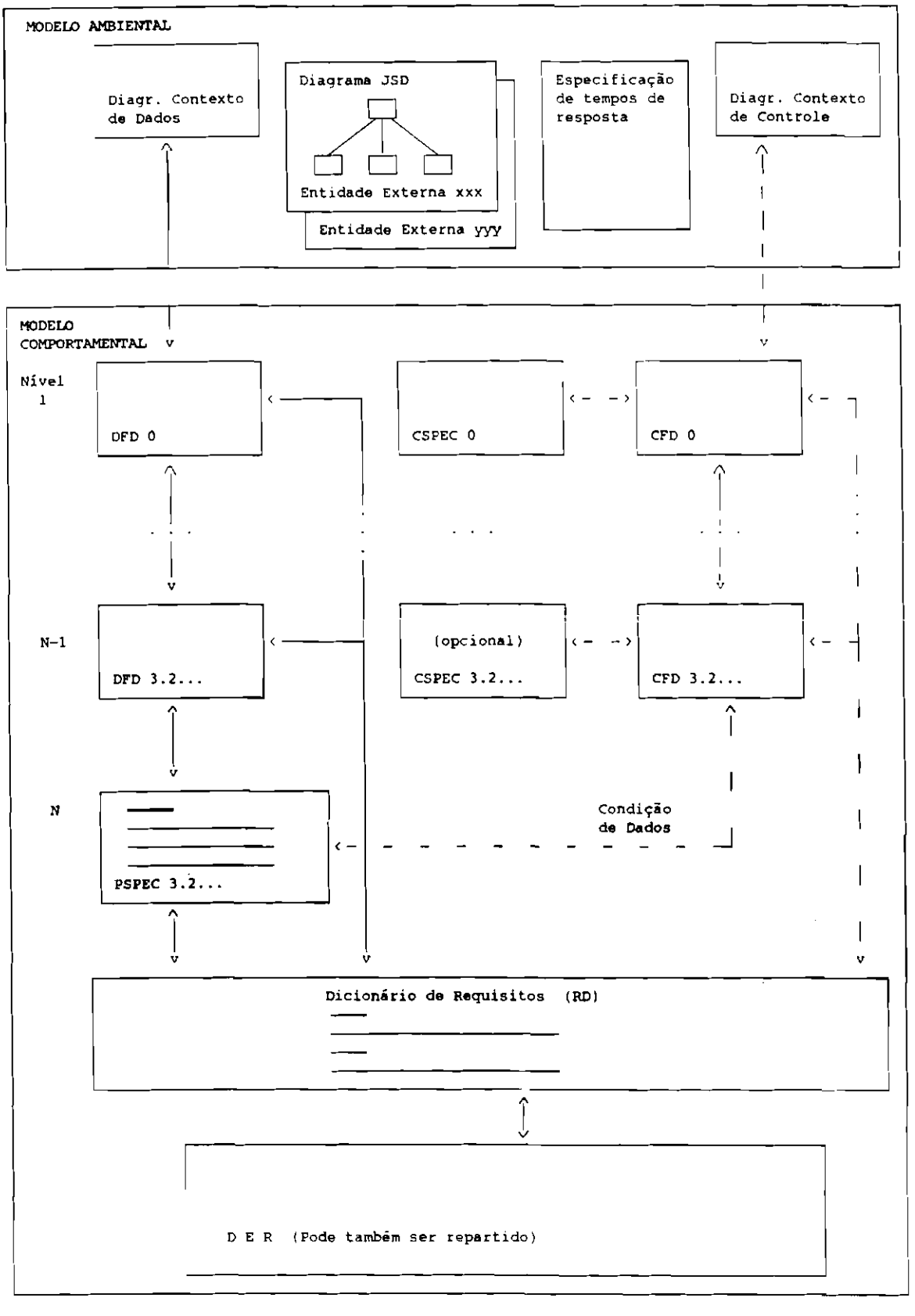


Fig. 6.3 - Estrutura do modelo híbrido.

6.2.3 - CONVENÇÕES

As principais convenções adotadas nesta combinação de métodos, por afetarem e condicionarem o modelo como um todo, são listadas abaixo:

a) Numeração e nomes: a mesma do método Hatley (2.1.3.1).

b) CSPEC: as mesmas do método Hatley (2.1.3.9.a e 2.1.3.9.c).

c) Todos os processos que uma CSPEC controla estão no DFD associado (2.1.3.9.b). Esta convenção é necessária para que se adote algumas convenções de simplificação do modelo (6.2.3.2.a, 6.2.3.3.a, 6.2.3.3.d).

d) Um processo sem fluxo de controle como ordem de execução é disparado por dados suficientes em suas entradas, conforme a(s) PSPEC(s) que a(s) detalha(m) (Hatley, 2.1.3.2.a). (Ou seja, processos não precisam ser "síncronos", como na convenção 3.1.3.6.e do método Ward/Mellor.)

A adoção das convenções foi feita baseado na Tabela 4.4 de comparação das convenções. Assim, são apresentadas aqui em termos de convenções equivalentes, não equivalentes, e únicas de um dos métodos. O critério básico, conforme o Capítulo 5, é simplificar o modelo, condensando informação. As convenções que objetivamente atendem ao critério de simplificação são: 6.2.3.2.a, 6.2.3.2.c, 6.2.3.3.a, 6.2.3.3.c, 6.2.3.3.d.

6.2.3.1 - CONVENÇÕES EQUIVALENTES NOS DOIS MÉTODOS

São adotadas todas as convenções equivalentes da Tabela 4.4, na Seção 4.6. Quando a observação na tabela fala melhor de um dos métodos, recomendamos usar a convenção ao modo deste.

6.2.3.2 - CONVENÇÕES DE UM SÓ DOS MÉTODOS

As seguintes convenções que existem em apenas um dos métodos foram adotadas:

a) Uso de ativadores de processos

Usar os três ativadores de processos de Ward/Mellor (ativar, desativar, disparar), mas só dentro das CSPECs, sem desenhar, conforme a Figura 6.4 (2.1.3.3.a). Não desenhando tais fluxos de controle, desobstrui-se consideravelmente os diagramas, permitindo sobrepor-se a maioria dos pares DFD/CFD, sem prejudicar a legibilidade.

b) Dinâmica no DFD

Usar as regras de dinâmica de Ward/Mellor (3.1.3.3.c), que usam fluxos de dados contínuos, para modelar comportamento dinâmico dos processos. A Figura 6.4 mostra os casos possíveis de tais regras.

c) Múltiplos subsistemas

Usar a convenção (3.1.3.6.f) para múltiplos subsistemas equivalentes. Isto reduz o número de processos num diagrama.

d) Convenções para o "DER"

As convenções para o DER são as do método Ward/Mellor (3.1.3.9).

e) Estado transitório

Usar nos DTEs, quando necessário, a convenção (3.1.3.8.g) para estado transitório.

6.2.3.3 - CONVENÇÕES NÃO EQUIVALENTES

Abaixo são listadas as convenções adotadas no método híbrido que existem nos dois métodos mas não são equivalentes.

a) Desativação implícita no DTE

No DTE, uma transição desativa implicitamente os processos ativos ao ser executada (2.1.3.9.e). Não é necessário escrever ações nas transições para desativar processos.

b) Fluxos de dados contínuos

Seguem a mesma convenção do método Ward/Mellor; sendo usados somente para interfacear com o ambiente (3.1.3.2.c).

c) Fluxos de dados discretos

São usados tanto na interface com o ambiente quanto internamente. (Em outras palavras, na interface externa, pode-se ter fluxos de dados contínuos e discretos. No interior do sistema, pode-se ter apenas fluxos de dados discretos.) Os fluxos de dados discretos são interpretados como nos casos (a) e (b) da figura 4.4, à maneira do método Ward/Mellor. Não são continuamente disponíveis, como para Hatley. Isto implica em ter-se que usar depósito para passar informação entre dois processos não sincronizados, deixando este fato mais explícito.

d) Valores de um fluxo de controle primitivo

O número valores pode ser qualquer, como no método Hatley. Embora Ward associe apenas um fluxo de controle com um valor a cada evento de controle, nada impede de (como com os dados) vários valores para um fluxo de controle primitivo. Assim fazendo, a cada valor de um fluxo de controle primitivo corresponderá um único evento dos diagramas JSD. Isso reduz grandemente a quantidade de fluxos num

diagrama.

e) Depósitos de dados

Podem ser decompostos entre DFDs, como no método Ward/Mellor. Desenhar um depósito em todos os DFDs onde ele for usado.

f) Fluxos ligados a depósitos

Usar a convenção do método Hatley (2.1.3.6.d).

g) Depósito de controle

São normalmente pequenas estruturas de informação de controle, mas se algum deles ficar grande, pode-se decompô-lo também.

h) Uso de depósito de controle por uma CSPEC

Usado como semáforo, como no método Ward/Mellor (3.1.3.5).

i) CSPECs grandes

Organizar como em 2.1.3.9.c, 2.1.3.9.f, 2.1.3.9.g.

j) Convergência/divergência de fluxos

Como no método Hatley (2.1.3.4.b).

k) Dicionário de Requisitos

Embora apresentadas diferentemente, as convenções de Hatley e de Ward/Mellor para o dicionário são totalmente equivalentes. Vamos adotar as de Hatley (2.1.3.8), devido à facilidade de consulta à Tabela 2.1. Também porque o dicionário de Hatley incorpora informação de freqüência de dados externos, um requisito de tempo importante.

1) Requisitos de tempo

Existem os seguintes casos:

- Especificação de tempos de resposta: usar como no método Hatley. Geralmente a forma de tabela basta para especificar claramente. Quando o relacionamento entre eventos de entrada e os de saída forem mais complexos, pode-se usar gráficos ou qualquer recurso que especifique claramente o vínculo de tempo (isto também faz parte do método Hatley).

- Freqüências requeridas para ocorrência de fluxos externos: são especificadas no RD (ver Tabela 2.1). Note que só se aplicam a fluxos externos; especificá-las para fluxos internos já é trabalho de projeto.

- Uso de tempo em PSPECs e CSPECs. Fazer como no método Hatley: somente em casos especiais especificar requisitos de tempo em PSPECs e CSPECs. Em geral, recomenda-se não fazê-lo. É uma área muito sujeita a entrar em definições de projeto. Tais casos seriam quando a freqüência de ativação de um processo é parte essencial deste processo, como no caso de filtros digitais, ou quando a freqüência de um fluxo externo primitivo que está sendo gerado é vinculada com a de outro fluxo.

A Figura 6.5, extraída de Hatley e Pirbhai (1987), ilustra as várias partes em que os requisitos de tempo afetam o Modelo de Requisitos. Ela mostra fluxos primitivos externos constantes do diagrama de contexto e definidos, com suas freqüências de repetição, no dicionário. A figura também mostra um linha da especificação de tempos de resposta ligada ao resto do modelo pelas definições no dicionário de seus fluxos de entrada e saída.

m) Convenções para PSPECs

As convenções para PSPECs foram adotadas do método Hatley (convenções 2.1.3.10).

O uso de estrutura gramatical para concorrência no Inglês (ou Português) estruturado (2.1.3.10.e) é muito útil. Permite evitar uma prática muito comum quando não é requerida nenhuma seqüência específica entre algumas ações relacionadas: força-se um algorítmo usando ciclos ("loops"), apenas porque não se tem outra maneira de expressar o requisito. Colocando-se como concorrentes, fica a cargo do projetista elaborar um algorítmo eficiente que satisfaça às restrições de tempo e implemente a concorrência. Um exemplo, extraído de Hatley e Pirbhai (1987), em que três ações podem ser feitas em qualquer ordem, inclusive em paralelo, é:

Calcular DISTÂNCIA AUXNAV como grande círculo desde
   POSIÇÃO AERONAVE até POSIÇÃO AUXNAV.
Calcular DIFERENÇA ALTITUDE como
   ALT. AERONAVE - ALT. AUXNAV.
Pôr FREQÜÊNCIA RADIO em FREQÜÊNCIA AUXNAV.

Apenas no caso em que alguma ordenação das ações é realmente requerida (seqüência, iteração ou seleção) deve ser explicitada. Isto é feito respectivamente com "DEPOIS", "PARA CADA... FAZER", ou "SE... ENTÃO... SENÃO". Por exemplo, as ações acima, em seqüência, seriam:

Calcular DISTÂNCIA AUXNAV como grande círculo desde
   POSIÇÃO AERONAVE até POSIÇÃO AUXNAV.
DEPOIS, Calcular DIFERENÇA ALTITUDE como
   ALT. AERONAVE - ALT. AUXNAV.
DEPOIS, Pôr FREQÜÊNCIA RADIO em FREQÜÊNCIA AUXNAV.

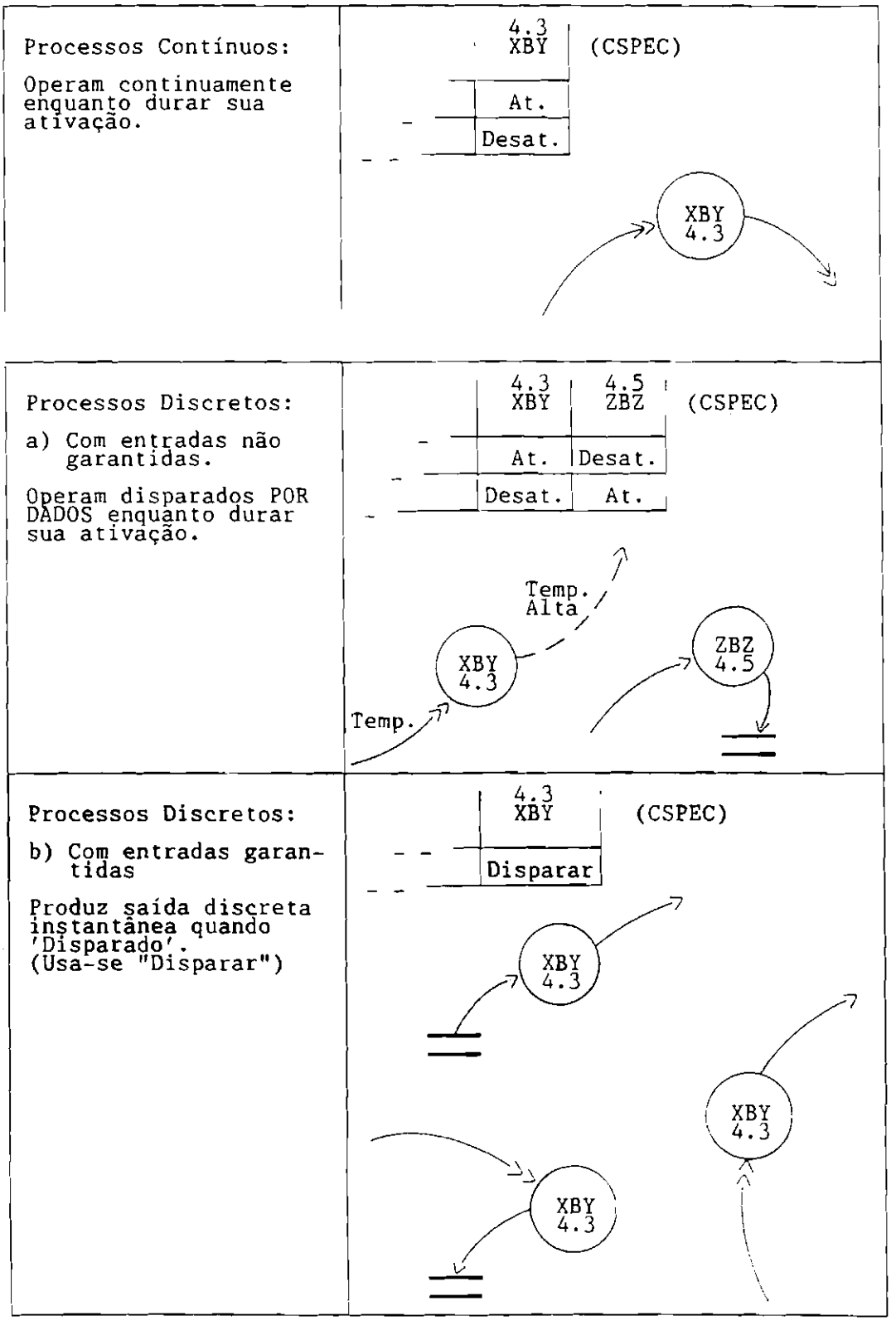


Fig. 6.4 - Regras de comportamento dinâmico de processos, e o uso de ativadores de processo.

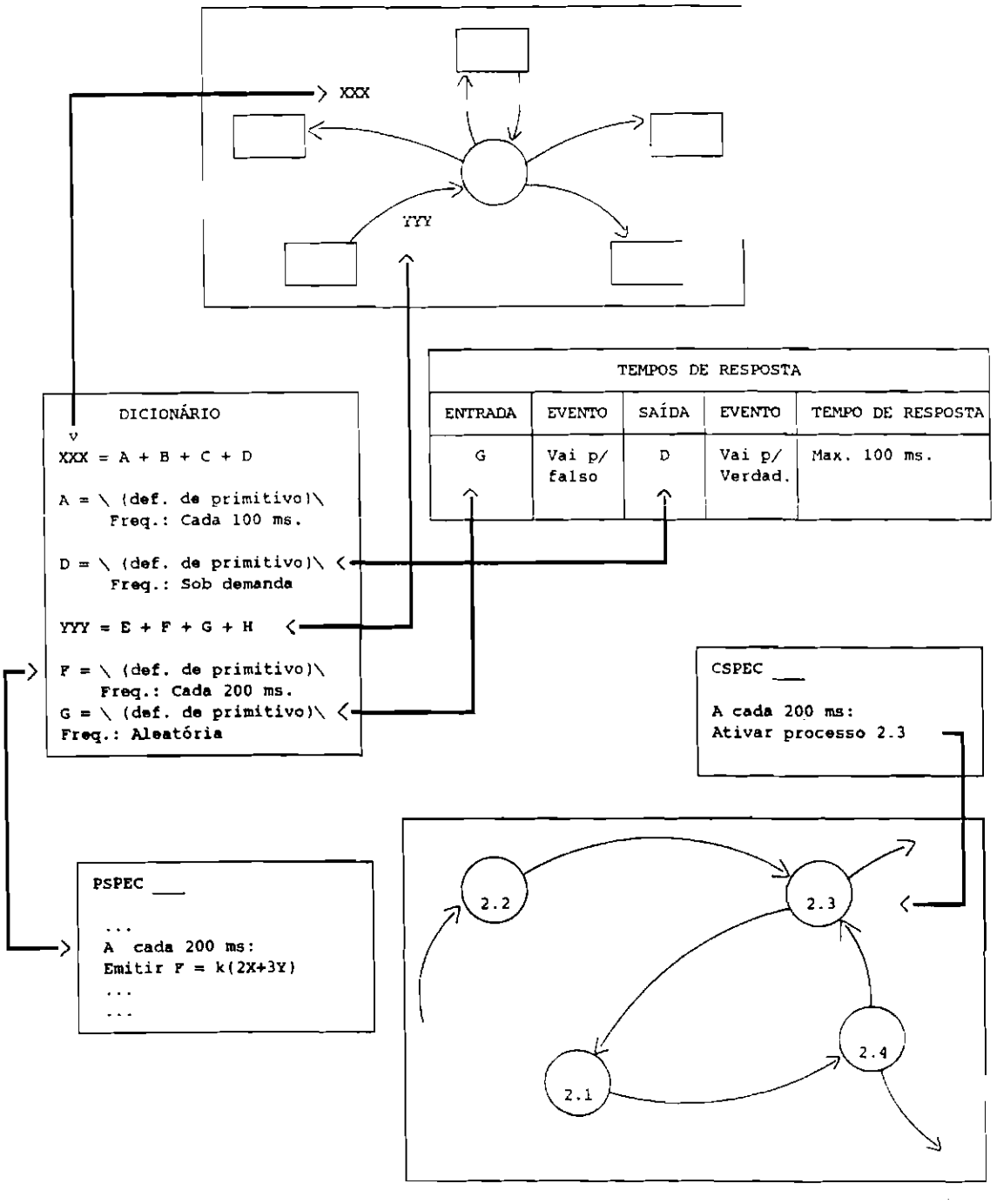


Fig. 6.5 - Requisitos de tempo ao longo do modelo.

FONTE: Hatley e Pirbhai (1987), p. 191.

6.3 - O PROCESSO DE MODELAGEM

Esta seção descreve o processo de modelagem do método híbrido. A Seção 6.3.1 descreve os passos do processo os quais são seguidos aplicando-se as heurísticas na Seção 6.3.2.

6.3.1 - OS PASSOS DA MODELAGEM

O processo do método híbrido terá duas etapas distintas, à maneira do método Ward/Mellor.

A primeira etapa consiste em modelar o ambiente. Faz-se os diagamas de contexto e um diagrama JSD para cada entidade externa, conforme descrito em 6.1.1. Com os eventos definidos, constrói-se a especificação de tempos de resposta.

A especificação de tempos de resposta descreve para cada evento do ambiente (detectado por uma entrada), qual evento deverá ocorrer na saída do sistema (sua resposta) e o intervalo de tempo necessário. Ou seja, descreve a dinâmica externa requerida do sistema para acompanhar o sistema de referência.

A cada evento da especificação de tempos, Hatley associa um fluxo primitivo (veja no Apêndice A). Isso força a que só no fim da modelagem se tenha a tabela completada, embora ele enfatize a necessidade de se começá-la desde o início. Com o uso dos diagramas JSD, não temos necessidade de vincular os eventos a fluxos primitivos nesta etapa. Assim, a especificação de tempos de resposta pode ser preenchida, nesta etapa, somente nas colunas dos eventos e do intervalo de tempo.

A segunda etapa do processo de modelagem consiste em modelar o comportamento requerido internamente ao sistema. O modelo do comportamento é derivado do Modelo Ambiental, como no método Ward/Mellor. A idéia básica é modelar dirigido por eventos, trabalhando na estrutura do modelo, em vez de trabalhar-se num grande diagrama de

um só nível (para sistemas aviônicos, por exemplo, é comum ter-se da ordem de vinte a trinta entidades externas, resultando em centenas de eventos. Isso levaria a um diagrama único impraticavelmente grande).

Sendo dirigido por eventos, começa-se tal como no método Ward/Mellor, derivando para cada evento a reação interna necessária (uma transformação de dados, uma transição de estados, um armazenamento de dados, ou a produção de uma saída de dados ou de controle) e/ou uma transformação reconhecedora do evento. À medida em que mais processos vão sendo adicionados, vai-se montando a estrutura do modelo. Faz-se iterações entre os níveis do modelo, identificando mais transformações e agrupando-as de modo que o modelo reflita o ambiente e que se deixe sempre todo o controle que houver para um DFD em sua CSPEC associada. Deve-se iterar também entre DFDs e DER, conforme a heurística nº1 na Seção 6.3.2.

Deve-se seguir todas as heurísticas da Seção 6.3.2; elas fazem parte do processo. Deixar sempre que possível o detalhamento das transformações respondedoras para depois.

Por último, acrescenta-se na tabela de tempos de resposta os fluxos primitivos externos associados a cada evento. Isto permitirá registrar no modelo as associações entre cada parte do modelo e os tempos de resposta, como mostrado na Figura 6.5.

Como resultado final, tem-se as transformações reconhecedoras de e respondedoras a eventos em níveis diferentes da estrutura do modelo, tal como se as tem ao terminar o passo três do método Ward/Mellor.

Este processo aqui descrito é uma solução de compromisso entre dificuldade de se fazer abstrações e dificuldade de se trabalhar com diagramas excessivamente grandes. No método Ward/Mellor, separa-se abstração de detalhes (feitas no passo 3) da identificação das respostas aos eventos, às custas de se fazer um diagrama único (o preliminar). No método Hatley, trabalha-se usando a estrutura do

modelo, de modo a se ter diagramas pequenos (sempre ligados pela indexação da estrutura), às custas de ter-se que fazer juntamente abstração de detalhes e identificação de funções e interfaces necessárias. A solução adotada evita o diagrama único porque a abordagem visa sistemas tipicamente muito maiores que os sistemas típicos visados por Ward/Mellor. Além disso, o agrupamento de processos surge bastante naturalmente à medida que vai-se adicionando processos no DFD.

6.3.2 - HEURÍSTICAS DE MODELAGEM

Todas as heurísticas semelhantes listadas na Seção 4.2 valem igualmente neste método híbrido.

Quanto às heurísticas diferentes (as da Tabela 4.1):

1) Primeiro DFDs, depois DTE, ou primeiro a perpectiva dominante? Tanto faz. Determine a perpectiva dominante: DFD, DTE ou DER. O importante é alternar entre as perspectivas, sabendo que a dominante sempre será mais informativa sobre o sistema. Lembre-se que há sistemas com duas perspectivas igualmente importantes.

2) Seguir o princípio dos 7 ± 2 ou não? Sim. Seguir o enunciado de Hatley (2.2.2.d).

3) Quanto a detalhar bolhas abaixo do nível de resposta a eventos: usar decomposição funcional mesmo. As heurísticas do método Ward/Mellor não bastam. Tentar não usar decomposição funcional sem ter outra diretriz é contraproducente.

4) Manter todo o controle de um DFD em sua CSPEC associada.

CAPÍTULO 7

PROJETO: MODELO DO SISTEMA A IMPLEMENTAR

Com o objetivo de dar uma visualização de como a especificação estruturada de requisitos se liga com a real implementação do sistema (em código e hardware), este capítulo trata do projeto de sistemas tempo-real.

O projeto é visto como a construção de um modelo da implementação pretendida para o sistema. A atividade de projeto pode ser descrita basicamente como um trabalho de, iterativamente:

- escolher meios de implementação (processadores, memórias, sensores, atuadores, interfaces, tarefas de software, módulos, estruturas de dados, etc.),

- identificar restrições de implementação, (capacidades de processamento e de memória, características de sensores e atuadores, restrições da linguagem de programação, número máximo de tarefas no sistema operacional etc.) e

- alocar requisitos à tecnologia de implementação escolhida.

No Projeto Estruturado convencional, de Yourdon e Constantine (1975), a premissa básica é de que todo o programa que implementa os requisitos (DFDs), rodaria como uma única tarefa (ou "job") num grande computador central ("mainframe"). Os métodos de Hatley e de Ward/Mellor fizeram extensões às técnicas de Projeto Estruturado para o projeto de sistemas tempo-real. Levaram em conta a utilização de múltiplos processadores, podendo ter sistemas operacionais multi-tarefas. Segue-se uma descrição breve de como são essas extensões.

7.1 - O "MODELO DA IMPLEMENTAÇÃO" DE WARD/MELLOR

Ward e Mellor (1985) chamam o trabalho de projeto arquitetural de "modelagem da implementação". Dividem o trabalho em três etapas (Figura 7.1):

- Etapa de processadores (concorrência verdadeira): Requisitos são alocados a processadores.

- Etapa de tarefas (concorrência simulada): Requisitos são alocados a tarefas em cada processador.

- Etapa de módulos (concorrência proibida): Requisitos são alocados a módulos dentro de cada tarefa.

Nas duas primeiras etapas, usa-se os mesmos diagramas do modelo essencial, só que não são diagramas "de transformações", pois as bolhas não são transformações de dados: na primeira etapa as bolhas são processadores, e na segunda etapa são tarefas. Na etapa dos módulos usa-se as técnicas tradicionais de projeto estruturado para organizar o processo interno de uma tarefa seqüencial em módulos de software.

Esta divisão em etapas é baseada em características de concorrência. Visa lidar com a concorrência por meio de uma mudança progressiva da concorrência do Modelo Essencial, levando em conta as características pertinentes da tecnologia de implementação. São usadas as seguintes heurísticas:

- alocação de cima para baixo à tecnologia de implementação, o que leva à divisão em três etapas,

- minimizar a distorção do modelo essencial ao fazer o projeto,

- aproximação satisfatória do comportamento ideal do modelo essencial, ou seja, satisfazendo às restrições de tempo de resposta.

Às heurísticas e diretrizes da etapa de tarefas, poderíamos acrescentar provavelmente com benefícios os critérios do método DARTS de Gomaa (1984). Gomaa fornece seis critérios para agrupar processos dos DFDs em tarefas a implementar. É o que Ward/Mellor chamam de "alocar processos do Modelo Essencial a tarefas do Modelo da Implementação".

7.2 - O "MODELO DA ARQUITETURA" DE HATLEY

O método de Hatley é voltado para sistemas com vários níveis de subsistemas antes de ser feita a separação entre o que vai ser implementado em hardware e em software, ou seja, seus subsistemas são equipamentos aviônicos ou conjuntos deles. Usa dois tipos de diagrama especialmente criados para mostrar a estrutura desses vários níveis de arquitetura. São os "diagramas de fluxos da arquitetura" (AFDs) e os "diagramas de ligações da arquitetura" (AIDs). Os AFDs são muito parecidos com DFDs. Só mudam os formatos dos elementos: retângulos arredondados para equipamentos (tanto processadores quanto dispositivos de memória), e setas retas fazendo ângulos retos para fluxos. Essa notação modificada é apenas para distinguir visualmente que se trata de projeto. São totalmente equivalentes aos DFDs de processadores de Ward/Mellor.

Quanto à arquitetura de software, que é o que nos interessa mais, Hatley e Pirbhai (1987, p. 282) falam pouco: "está além da abrangência deste livro entrar em detalhes de como criar módulos de software. Há muitas maneiras diferentes." Sugere combinar AFDs e diagramas de estrutura (do Projeto Estruturado), usando AFDs para tarefas enquanto forem concorrentes entre si, e diagramas de estrutura para decompor em módulos uma tarefa que contenha um processo seqüencial.

7.3 - COMPARAÇÃO E COMENTÁRIOS

As diferenças básicas entre as extensões do projeto estruturado de Ward/Mellor e Hatley são duas.

A primeira é que Ward considera a tecnologia de implementação bem dominada (seu método é voltado para controle de processos), portanto, não se preocupa com desenvolvimento de processadores em paralelo com o de software. O objetivo dos DFDs de processadores é só mostrar a alocação, não seu desenvolvimento. Ele considera que se usa processadores disponíveis no mercado, muitas vezes com sistema operacional multi-tarefas já instalado. Assim, ele dá algumas diretrizes para alocação a processadores e dedica-se bem mais ao projeto do software.

A segunda diferença é que Hatley considera a tecnologia de implementação pouco dominada e/ou pouco definida durante o desenvolvimento (seu método é voltado para sistemas aviônicos). Preocupa-se com o desenvolvimento de processadores e de hardware em geral feito em paralelo com o desenvolvimento de software. Enfatiza bastante a arquitetura "de sistema" e não visa a de software. Mas suas linhas básicas para projeto de software, se bem que sucintas, são as mesmas que as de Ward e Mellor.

Uma razão para essas diferenças está no grau de complexidade dos sistemas visados. Para aplicações de controle de processos (a área de Ward), com alguns processadores e um microcomputador para interface com o usuário, pode-se controlar toda uma fábrica com várias linhas de produção. Para aplicações aviônicas (a área de Hatley), tem-se geralmente dezenas de equipamentos, vários com processadores, compondo o sistema aviônico de um único avião; sempre com redundância e modos de operação combinados para aumentar a segurança.

Ambos os métodos se preocupam em expressar os requisitos de maneira a não restringir a implementação nem as decisões de projeto. Não obstante, tanto a modelagem de requisitos como um todo, quanto a modelagem do sistema (o projeto), em ambos os métodos foram direcionadas para tipos diferentes de aplicação (ou, talvez mais propriamente, foram derivadas da experiência em tipos diferentes de aplicação).

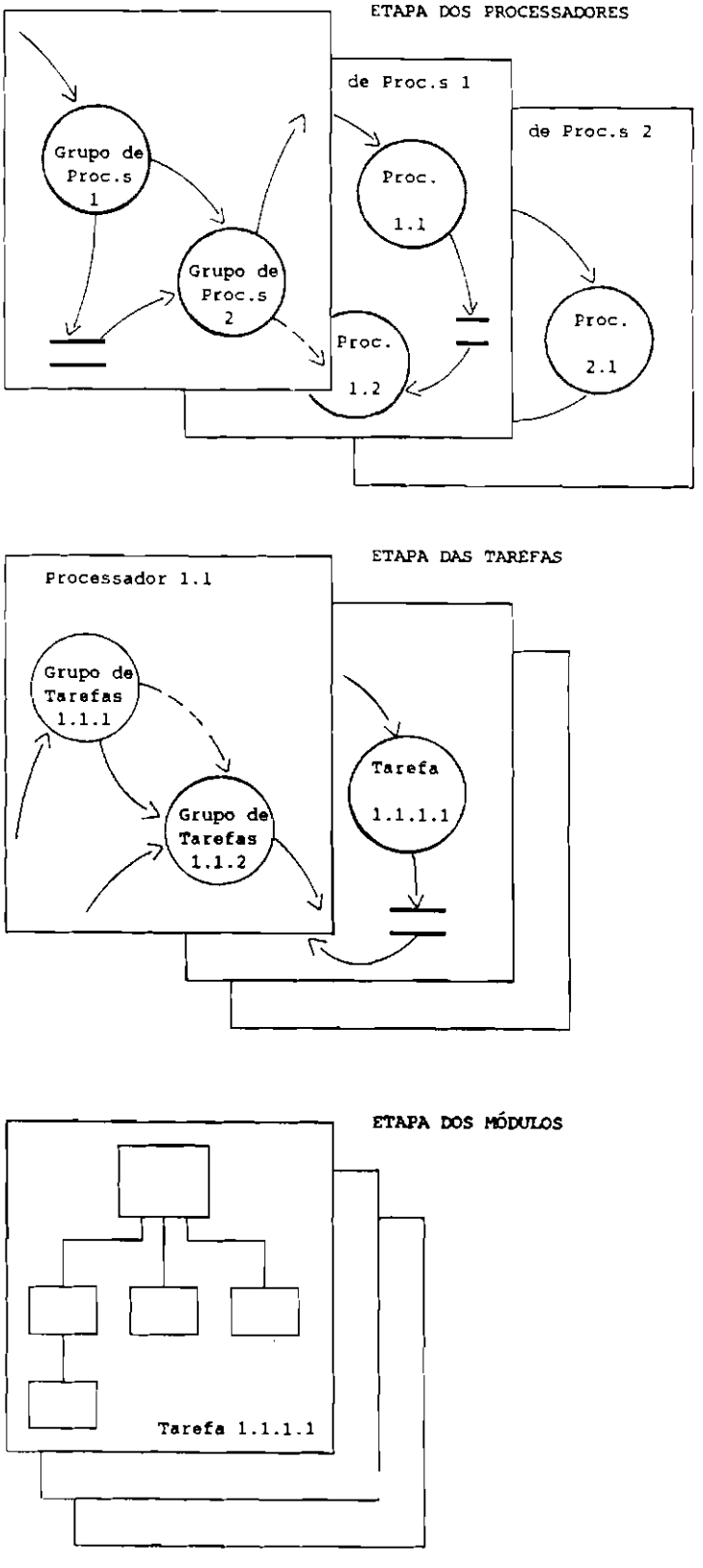


Fig. 7.1 - Etapas do projeto estruturado de Ward/Mellor.

REFERÊNCIAS BIBLIOGRÁFICAS

BERGLAND, G.D.; ZAVE, P. Guest editors' prologue. IEEE Transactions on Software Engineering, SE-12(2):185-191, Feb. 1986.

BUHR, R. System design with Ada. Englewood Cliffs, NJ, Prentice-Hall, 1984.

CARTER, J.R. MMAIM: A software development method for Ada. Ada Letters, 8(3):107-114, May/June 1988.

CHEN, P.P. The entity-relationship model -- toward a unified view of data. ACM Transactions on Database Systems, 1(1):9-36, Mar. 1976.

DEMARCO, T. Structured analysis and system specification. Englewood Cliffs, NJ, Prentice-Hall, 1978.

DIJKSTRA, E.W. Cooperating sequential processes. In: GENUYS, F., ed. Programming languages. New York, NY, Academic, 1968. cap. 3, p. 43-112.

FLAVIN, M. Fundamental concepts of information modeling. Englewood Cliffs, NJ, Prentice-Hall, 1981.

FLOYD, C. Outline of a paradigm change in software engineering. Software Engineering Notes, 13(2):25-38, 1988.

FREEMAN, P. Fundamentals of design. In: FREEMAN, P.; WASSERMAN, A.I. Tutorial on software design techniques. 4. ed. New York, IEEE, 1983. p. 2-22.

GANE, C.; SARSON, T. Structured systems analysis. Englewood Cliffs, NJ, Prentice-Hall, 1979.

GOMAA, H. A software design method for real-time systems. Communications of the ACM, 27(9):938-949, Sept. 1984.

HALL, A.D.; FAGEN, R.E. Definition of system. In: BUCKLEY, W., ed. Modern system research for the behavioral scientist. Chicago, Aldine, 1968. p. 18-22.

HATLEY, D.J. The use of structured methods in the development of large software-based avionics systems. In: AIAA/IEEE DIGITAL AVIONICS SYSTEMS CONFERENCE, 6., Baltimore, MD, Dec. 3-6, 1984. Proceedings. New York, AIAA, 1985, p. 6-15.

HATLEY, D.J. A structured method for real-time and general systems development. In: THE IEEE COMPUTER SOCIETY'S ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 10., Chicago, IL, Oct. 8-10, 1986. Proceedings. Washington, DC, IEEE Comput. Soc. Press, 1986, p. 273.

HATLEY, D.J.; PIRBHAI, I.A. Strategies for real-time system specification. New York, NY, Dorset House, 1987.

JACKSON, M.A. Principles of program design. London, Academic, 1975.

JACKSON, M.A. System development. Englewood Cliffs, NJ, Prentice-Hall, 1983.

MCMENAMIN, S.M.; PALMER, F. Essential systems analysis. Englewood Cliffs, NJ, Prentice-Hall, 1984.

MILLER, G.A. The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review, 63(2):81-97, Mar. 1956.

MURATA, M. Aspectos computacionais de sistemas de tempo real. Dissertação de Mestrado em Computação Aplicada. São José dos Campos, INPE, 1985. (INPE-4157-TDL/269).

VON BERTALANFFY, L. General systems theory. New York, NY, Baziller, 1969.

WARD, P.T. The transformation schema: an extension of the data flow diagram to represent control and timing. IEEE Transactions on Software Engineering, SE-12(2):198-210, Feb. 1986.

WARD, P.T.; MELLOR, S.J. Structured development for real-time systems. Englewood Cliffs, NJ, Prentice-Hall, 1985. 3 v.

WEINBERG, G.M. An introduction to general systems thinking. New York, John Willey & Sons, 1975.

WIRTH, N. Toward a discipline of real-time programming. In: GLASS, R.L., ed. Real-time software. Englewood Cliffs, NJ, Prentice-Hall, 1983. cap. 3, p. 128-142.

YOURDON, E. Novos desenvolvimentos da análise estruturada de sistemas, Seminário do IBPI. Rio de Janeiro, 3-5/Out/1988.

YOURDON, E.; CONSTANTINE L.L. Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. Englewood Cliffs, NJ, Prentice-Hall, 1975.

APÊNDICE A

ILUSTRAÇÕES

Este apêndice contém algumas figuras, com o propósito de servir de ilustração para os diagramas citados neste trabalho. As Figuras de A.1 até A.8 são todas extraídas de Hatley e Pirbhai (1987); de um exemplo de Modelo de Requisitos para um sistema de controle de cruzeiro para automóveis. As Figuras de A.9 até A.15 são todas extraídas de Ward e Mellor (1985); de um exemplo do Modelo Essencial para um sistema de controle de cruzeiro para automóveis. Embora muito semelhantes, os requisitos não são os mesmos para os dois exemplos.

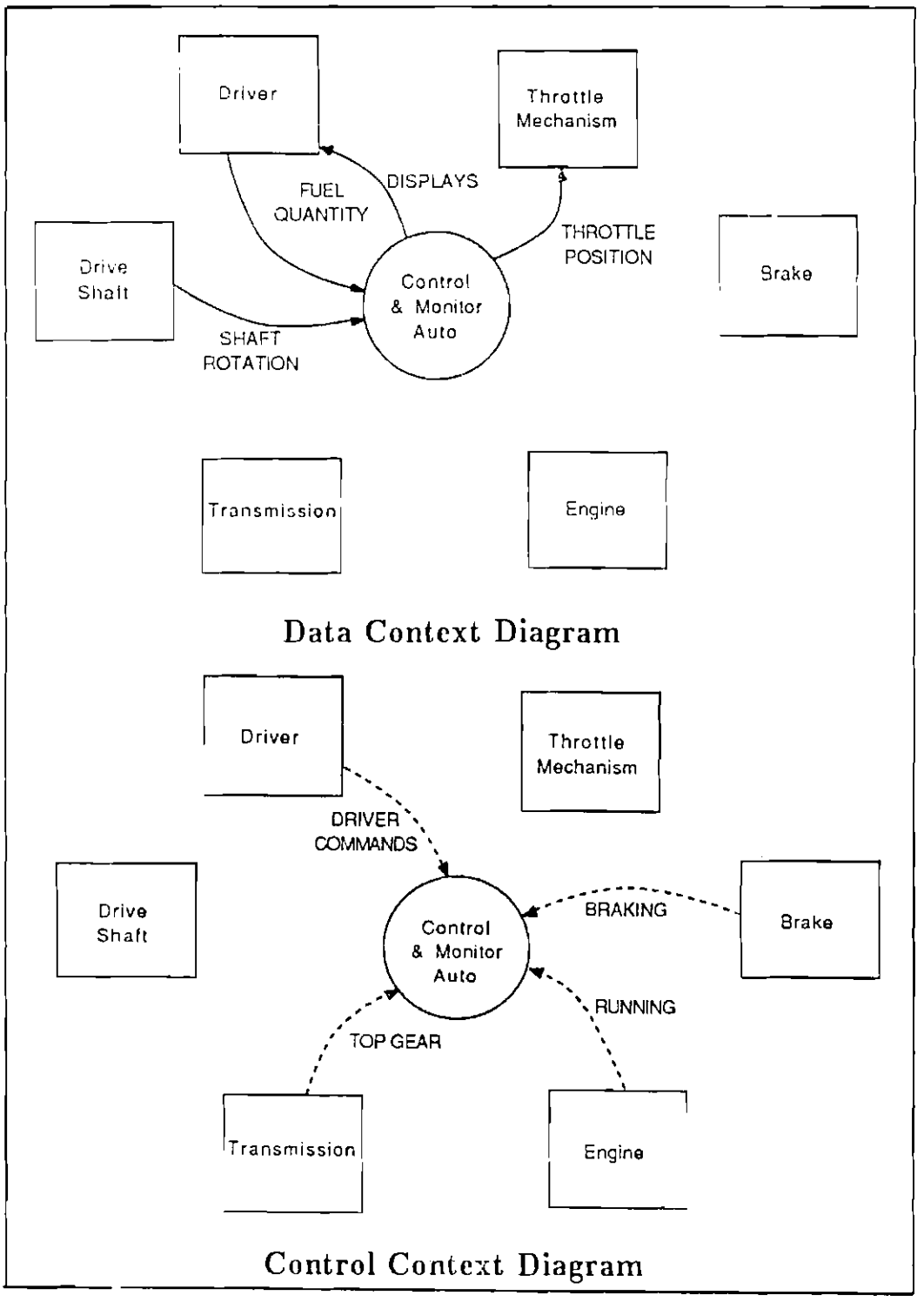


Fig. A.1 - Diagramas de contexto: de dados e de controle.
FONTE: Hatley e Pirbhai (1987), p. 301 (reprodução).

TIMING SPECIFICATION

INPUT	EVENT	OUTPUT	EVENT	RESPONSE TIME
ACTIVATE	Turns On	THROTTLE POSITION	Goes to selected cruise value	0.5 sec. max.
RESUME	Turns On			
DEACTIVATE	Turns On	THROTTLE POSITION	Goes to null condition	0.5 sec. max.
TOP GEAR	Turns Off			
BRAKING	Turns On			
START ACCEL	Turns On	THROTTLE POSITION	Goes to acceleration value	0.5 sec. max.
STOP ACCEL	Turns On while THROTTLE POSITION at acceleration value	THROTTLE POSITION	Goes to cruise value	0.5 sec. max.
RUNNING	Turns On	THROTTLE POSITION	Goes to null value	0.5 sec. max.
SHAFT ROTATION	Rotation rate changes	THROTTLE POSITION	Corresponding change in value	1 sec. max.
	Distance since last maintenance exceeds maintenance warning distance	MAINT NEEDED	Displayed	10 sec. max.
FUEL QUANTITY	Entered	FUEL CONSUMPTION	Displayed	1 sec. max.
AV SPEED RQST	Turns On	AV SPEED	Displayed	1 sec. max.
START TRIP				No time-critical outputs
START MEASURED MILE				
STOP MEASURED MILE				

Fig. A.2 - Especificação de tempos de resposta.
FONTE: Hatley e Pirbhai (1987), p. 304 (reprodução).

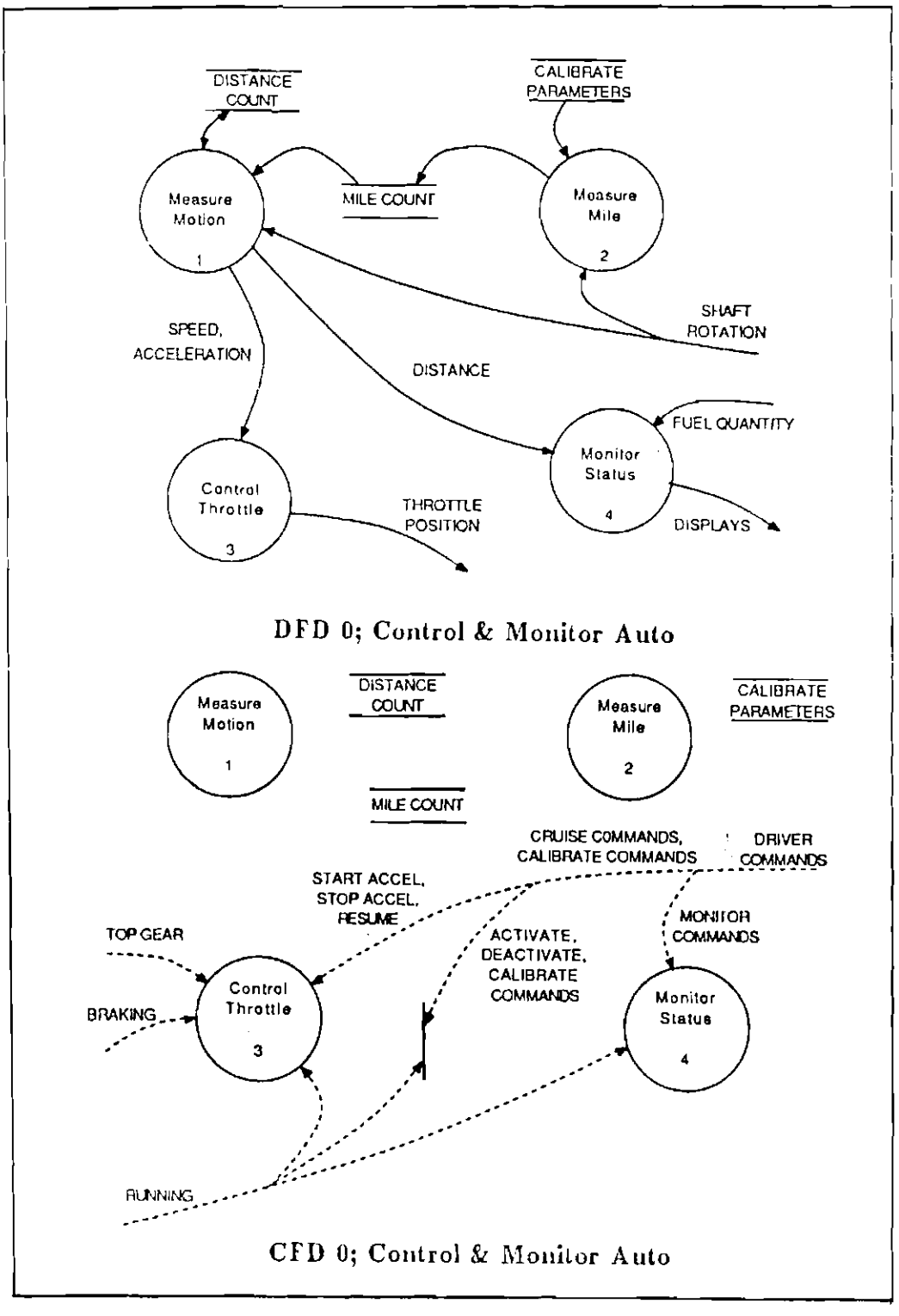


Fig. A.3 - Diagramas de fluxos: DFD 0 e CFD 0.
FONTE: Hatley e Pirbhai (1987), p. 302 (reprodução).

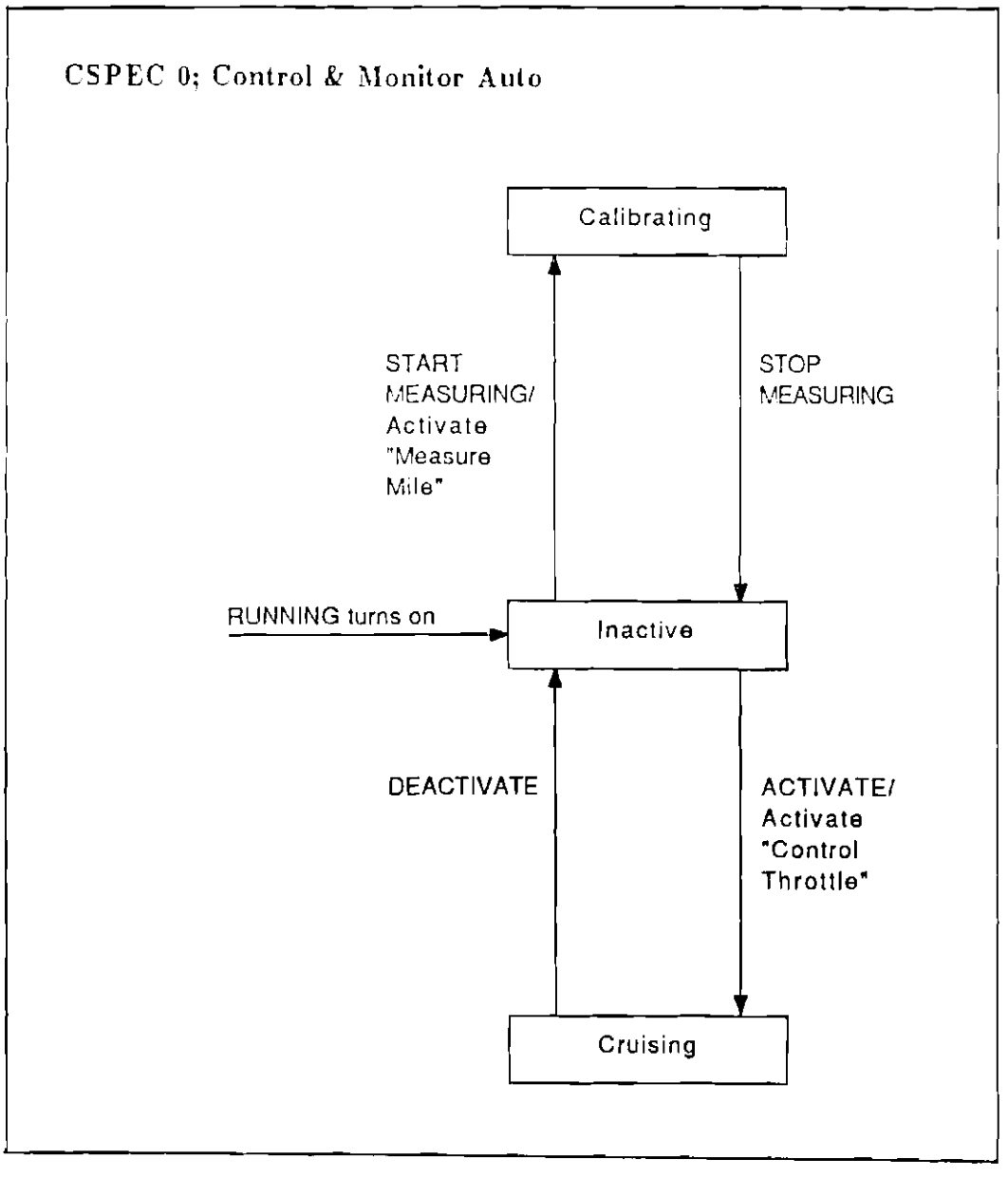
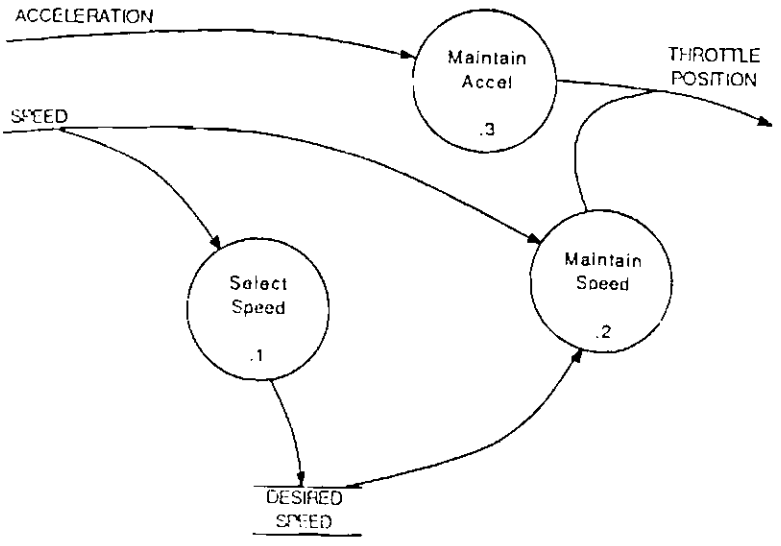
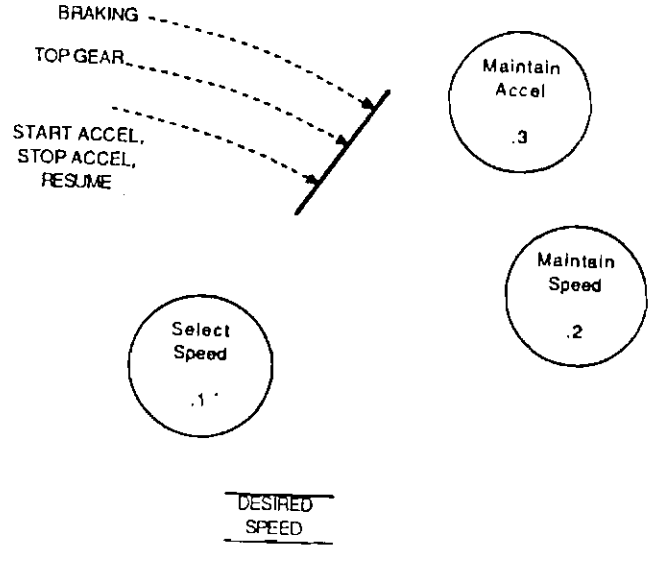


Fig. A.4 - Especificação de Controle: CSPEC 0.
FONTE: Hatley e Pirbhai (1987), p. 303 (reprodução).



DFD 3; Control Throttle



CFD 3; Control Throttle

Fig. A.5 - Diagramas de fluxos: DFD 3 e CFD 3.
FONTE: Hatley e Pirbhai (1987), p. 307 (reprodução).

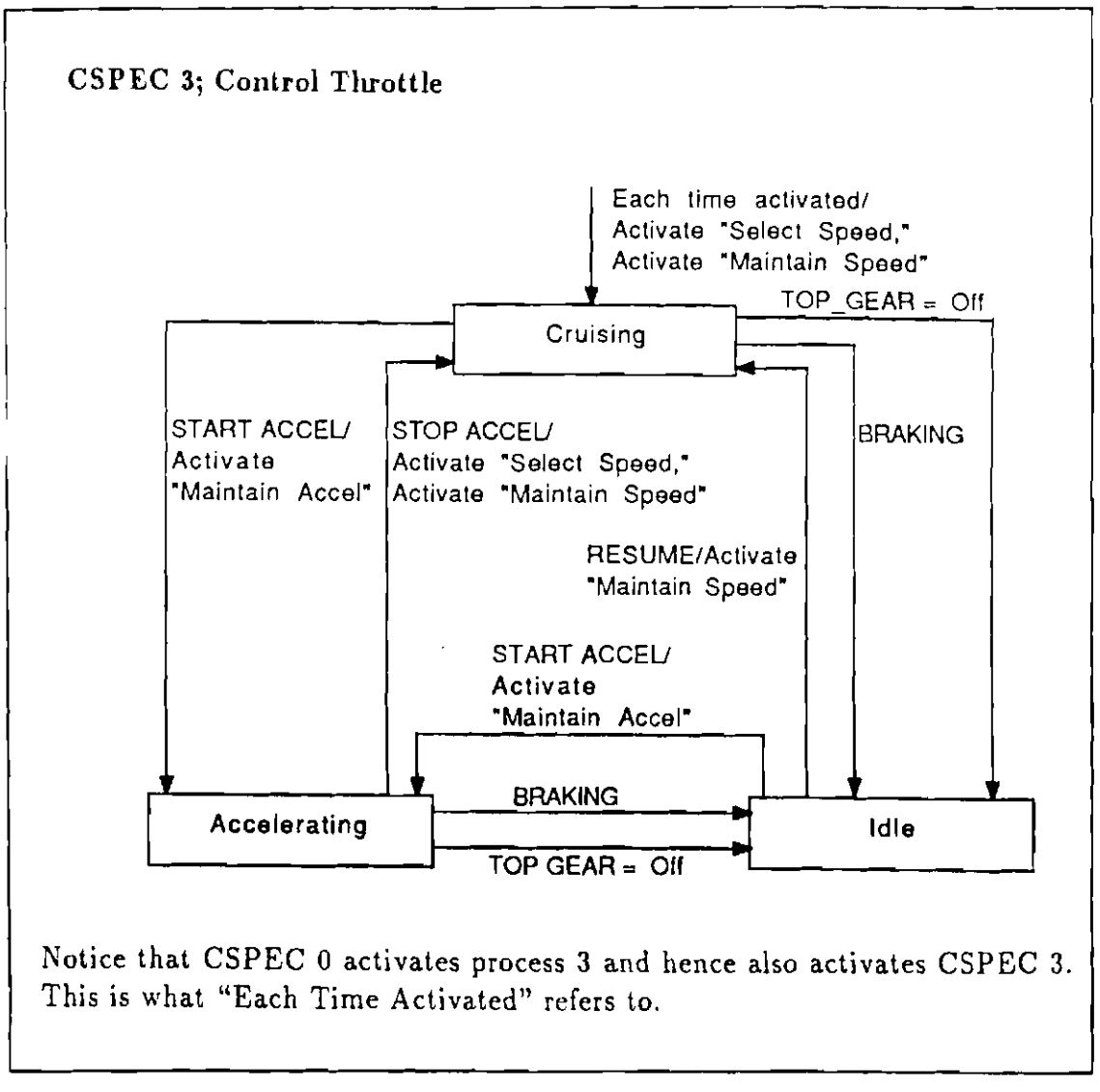


Fig. A.6 - Especificação de Controle: CSPEC 3.
FONTE: Hatley e Pirbhai (1987), p. 308 (reprodução).

PSPEC 3.2; Maintain Speed

Set:

$$V_{\mathrm{Th}} = \begin{cases} 0; & & (S_D - S_A) & > & 2 \\ 2(S_D - S_A + 2); & -2 & \leq & (S_D - S_A) & \leq & 2 \\ 8; & & -2 & > & (S_D - S_A) \end{cases}$$

subject to:

$$\frac{dV_{\mathrm{Th}}}{dt} \leq .8V/\mathrm{sec.}$$

where:

V_{Th} = THROTTLE POSITION
S_D = DESIRED SPEED
S_A = SPEED

Varies throttle opening from closed to fully open as speed varies from 2 mph above desired speed, to 2 mph below it. Restricts rate of opening to .8V/sec.

PSPEC 3.3; Maintain Accel

Set:

$$V_{\mathrm{Th}} = \begin{cases} 0; & & A & > & 1.2 \\ 20(1.2 - A); & 0.8 & \leq & A & \leq & 1.2 \\ 8; & & 0.8 & > & A \end{cases}$$

subject to:

$$\frac{dV_{\mathrm{Th}}}{dt} \leq .8V/\mathrm{sec.}$$

where:

V_{Th} = THROTTLE POSITION
A = ACCELERATION

Varies throttle opening from closed to fully open as acceleration varies from 1.2 mph/sec to 0.8 mph/sec. Restricts rate of opening to .8V/sec.

Fig. A.7 - Especificações de processos: PSPECs.
FONTE: Hatley e Pirbhai (1987), p. 309 (reprodução).

REQUIREMENTS DICTIONARY

		Data (D) or Control (C)
ACCELERATION =	\Measured vehicle acceleration\ Units: Miles per hour per second	D
ACTIVATE =	\Driver's cruise control activate command\ 2 Values: On, Off	C
AV SPEED =	\Calculated average trip speed\ Units: Miles per hour	D
AV SPEED RQST =	\Driver's request to display average trip speed\ 2 Values: On, Off	C
BRAKING =	\Input signal indicating brakes applied\ 2 Values: On, Off	C
CALIBRATE CMDS =	([START MEASURED MILE] STOP MEASURED MILE])	C
DEFAULT COUNT =	\Constant = *TBD;* Default value of calibrated mile count\ Units: Dimensionless *Need to get this value from auto manufacturer*	D
DESIRED SPEED =	\Speed cruise control is desired to maintain\ Units: Miles per hour	D
DISPLAYS =	(FUEL CONSUMPTION) + (MAINT NEEDED) + (AV SPEED)	D
DISTANCE =	\Calculated distance traveled by vehicle\ Units: Miles	D

Fig. A.8 - Trecho do Dicionário de Requisitos (RD).
FONTE: Hatley e Pirbhai (1987), p. 316 (reprodução).

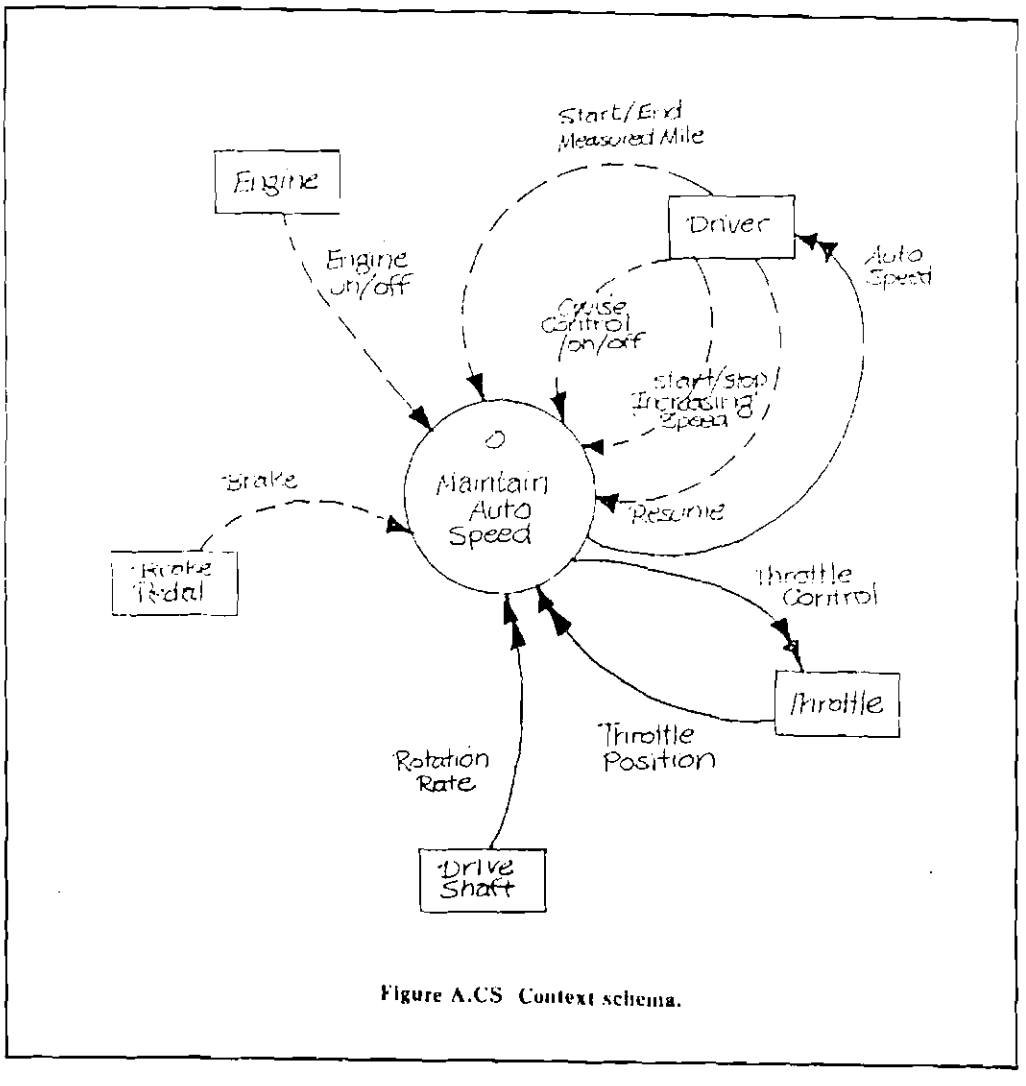


Figure A.C5 Context schema.

Fig. A.9 - Diagrama de contexto.

FONTE: Ward e Mellor (1985), p. 90 (reprodução).

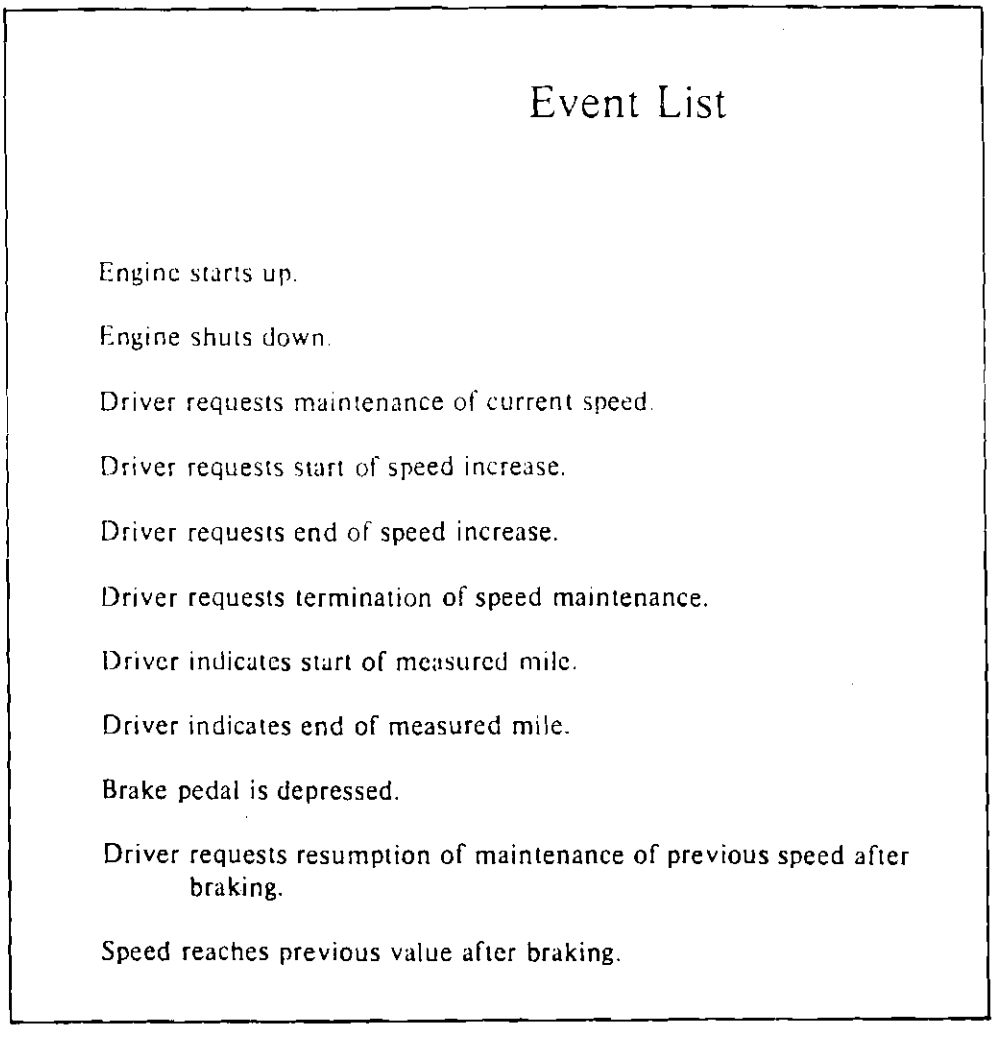


Fig. A.10 - Lista de Eventos.
FONTE: Ward e Mellor (1985), p. 91 (reprodução).

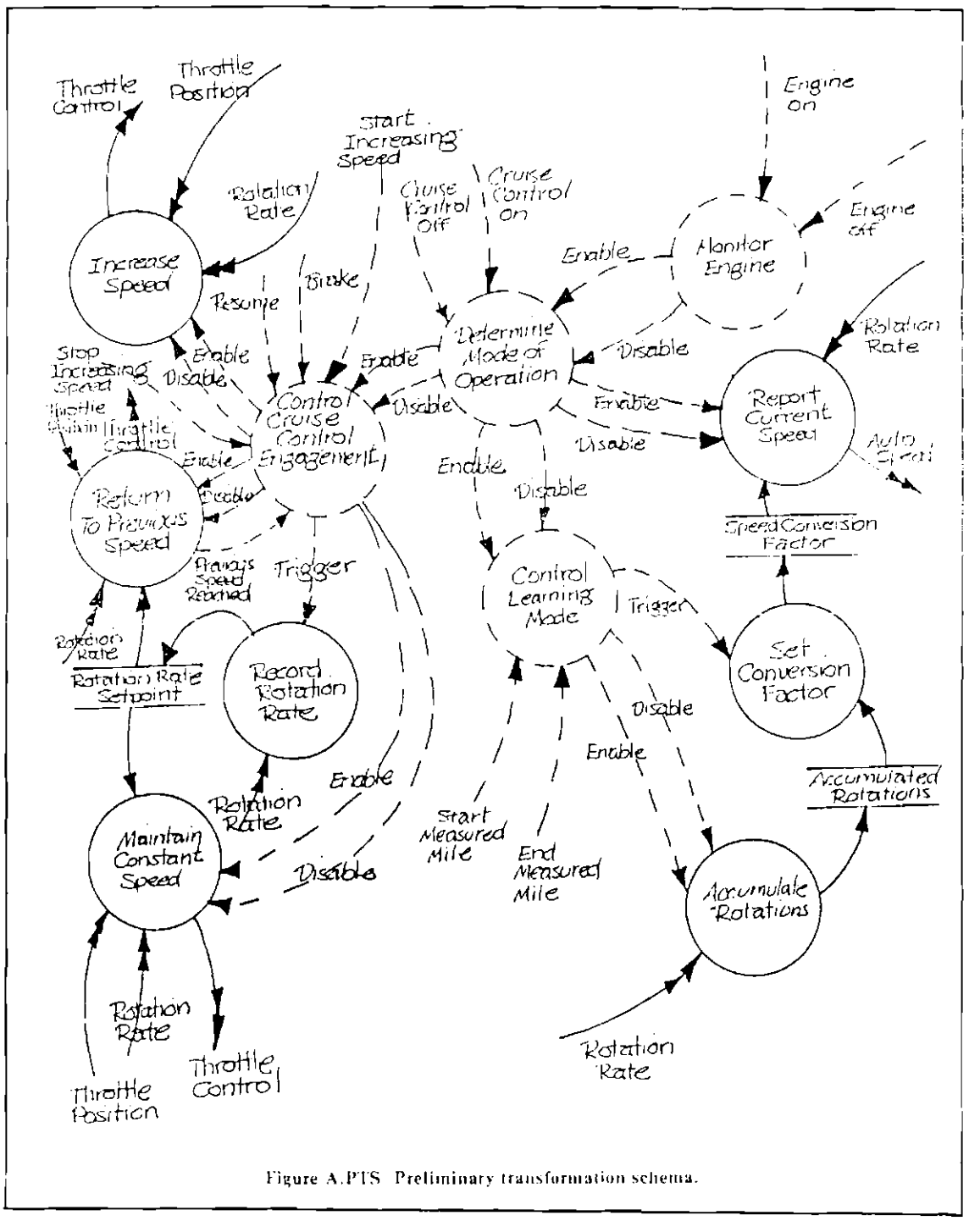


Figure A.PTS Preliminary transformation schema.

Fig. A.11 - Diagrama de transformações preliminar.

FONTE: Ward e Mellor (1985), p. 92 (reprodução).

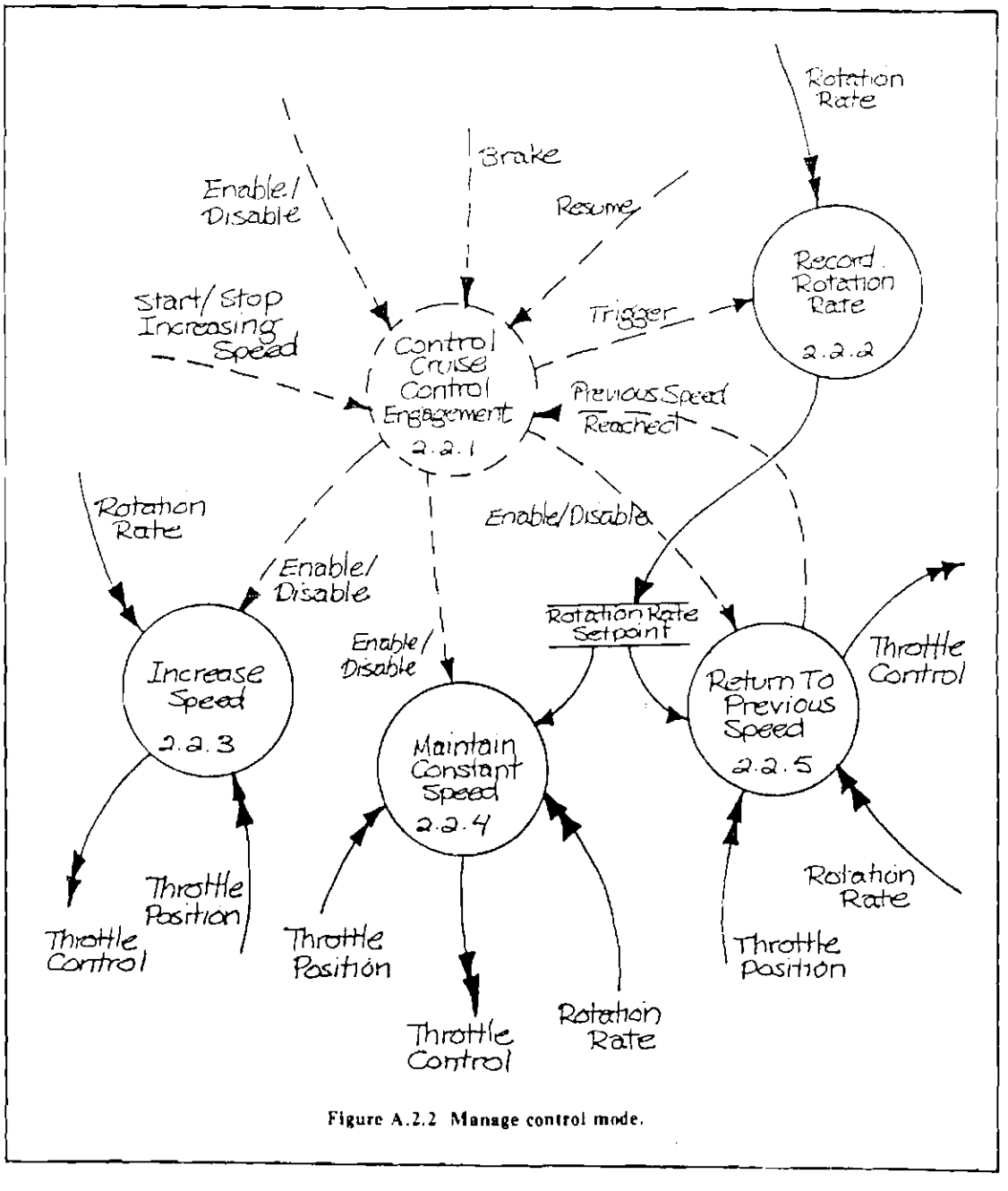


Figure A.2.2 Manage control mode.

Fig. A.12 - Diagrama de transformações 2.2.
FONTE: Ward e Mellor (1985), p. 97 (reprodução).

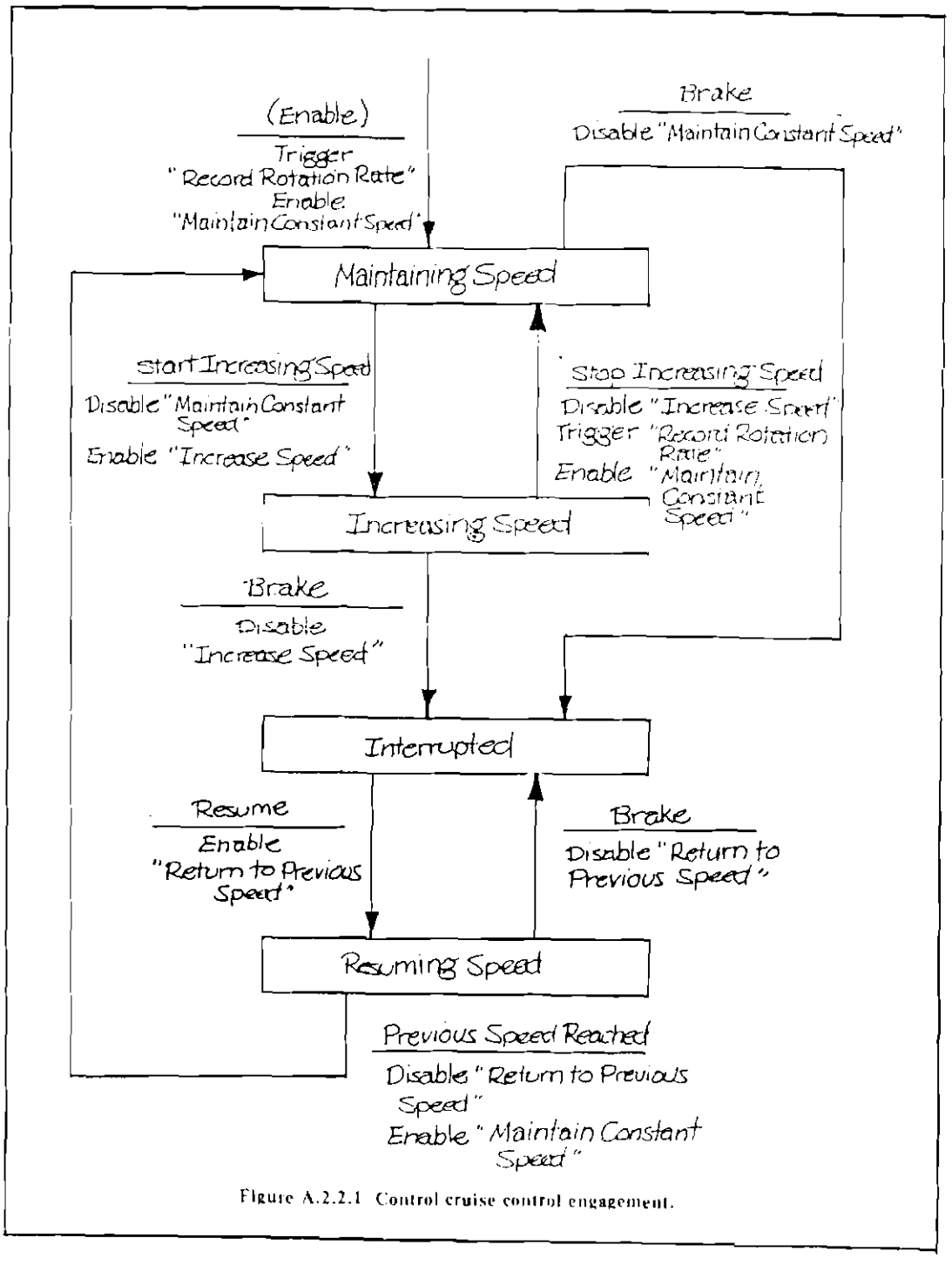


Figure A.2.2.1 Control cruise control engagement.

Fig. A.13 - Especificação da transformação de controle 2.2.1 (DTE).
FONTE: Ward e Mellor (1985), p. 98 (reprodução).

Transformation Specifications

2.2.3 Increase Speed

Precondition 1

None

Postcondition 1

THROTTLE POSITION at $<$ 80% of max. — Instantaneous rate of increase per second of ROTATION RATE maintained at 2% $\pm$ 25% of current value of ROTATION RATE

For THROTTLE POSITION at $\geq$ 80% of max. — Instantaneous rate of increase per second of ROTATION RATE maintained at 0.

2.2.4 Maintain Constant Speed

Precondition 1

ROTATION RATE within 1% of ROTATION RATE SETPOINT

Postcondition 1

For time $\leq$ 0.5 seconds after enable, match THROTTLE CONTROL to THROTTLE POSITION

For time $>$ 0.5 seconds after enable, maintain ROTATION RATE within 1% of ROTATION RATE SETPOINT

Fig. A.14 - Especificações de transformações de dados.
FONTE: Ward e Mellor (1985), p. 102 (reprodução).

Data Dictionary

accumulated rotations	=	*instantaneous value of number of rotations since start of measured mile*
		units : rotations
auto speed	=	*displayed value of current speed*
		units : miles per hour
brake	=	*signal that driver has depressed brake pedal*
cruise control off	=	**
cruise control on	=	**
end measured mile	=	**
engine off	=	**
engine on	=	**
previous speed reached	=	**
resume	=	*signal that driver wishes to resume previously set speed*
rotation rate	=	*instantaneous rate of rotation of wheel*
		units : revolutions per second
rotation rate setpoint	=	*desired rate of rotation of wheels*
		units : revolutions per second
speed conversion factor	=	*conversion between rotation rate of wheels and speed of car*
		units : rotations per mile
start increasing speed	=	**

Fig. A.15 - Trecho do Dicionário de Dados.
FONTE: Ward e Mellor (1985), p. 100 (reprodução).