MARCO ANTONIO INSAURRIAGA GONZALEZ

Matriz Square: Mecanismo Integrador dos Modelos da Metodologia OMT

Dissertação apresentada como requisito
parcial à obtenção de grau de mestre.
Curso de Mestrado em Informática,
Instituto de Informática,
Pontifícia Universidade Católica do
Rio Grande do Sul.
Orientador: Prof. Dr. Afonso Inácio Orth.

PORTO ALEGRE

1996/AGOSTO

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Gonzalez, Marco Antonio Insaurriaga

SQUARE: Mecanismo Integrador dos Modelos da Metodologia OMT / Marco Antonio Insaurriaga Gonzalez. — Porto Alegre: Instituto de Informática da PUC, 1996.

134 p.: il.

Dissertação (mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, Instituto de Informática, Porto Alegre, 1996. Orientador: Orth, Afonso Inácio.

Dissertação: Engenharia de Software, Análise de Software, Object Modeling Technique, Consistência de modelos, Integração de modelos

AGRADECIMENTOS

Durante este período em que foram revisadas idéias, pesquisadas as fontes de conhecimento disponíveis, organizadas e dirigidas as informações para dar a forma atual a esta dissertação, também foi necessário criar para que na alteração do já estabelecido fosse encontrada uma nova solução. Sem o apoio do meu orientador, freqüentemente acreditando mais em mim do que eu mesmo, não seria possível ter chegado até aqui. Por isto agradeço ao prof. Afonso Inácio Orth.

Se em alguns casos o vôo alto leva longe, também por vezes faz perder de vista as referências. É bom ter por perto alguém que nos faça descer um pouco, ou até bastante, e ver melhor a realidade. Por isto agradeço à profª. Karin Becker.

Uma idéia, muitas vezes, não é como uma luz que se acende de repente do nada. Em alguns momentos parece que nem vai se acender. Sem o incentivo na hora certa pode-se pensar até em trocar de lâmpada. Por isto agradeço ao prof. Duncan D.A. Ruiz.

Uma idéia também é composta de pequenas idéias. E elas podem ser trocadas. Por isto agradeço aos colegas de mestrado.

Em toda caminhada é indispensável o apoio que se recebe das instituições dirigidas por pessoas que investem, acreditando, em nós. Por isto agradeço ao Instituto de Informática da PUC, ao SENAC e à FAPERGS.

Mas também em toda caminhada é preciso que as pessoas que vão conosco olhem na mesma direção. Por isto agradeço à Aira e à Tatiana e a todos os meus familiares.

SUMÁRIO

LISTA DE ABREVIATURAS

AOO	Análise Orientada a Objetos
ATM	Máquina de Caixa Automático (Automatic Teller Machine)
CASE	Computer-Aided Software Engineering
CPGCC	Curso de Pós-Graduação em Ciência da Computação
DE	Diagrama de Estados
DFD	Diagrama de Fluxo de Dados
DFE	Diagrama de Fluxo de Eventos
DO	Diagrama de Objetos
ED	Dependência de Existência
IFIP	International Federation of Information Processing
JSD	Diagrama de Estrutura
M.E.R.O.DE.	Model-Driven Entity-Relationship Object-Oriented Development Method
MD	Modelo Dinâmico
MF	Modelo Funcional
MO	Modelo de Objetos
OBA	Object Behavior Analysis
OET	Tabela Evento-Objeto
OMT	Object Modeling Technique
OOA	Object-Oriented Analysis
OOAD	Object-Oriented Analysis and Design
OOD[1]	Object Oriented Development
OOD[1]	Object-Oriented Design
OOO	OOO Metodology
OORSM	Object-Oriented Requirements Specification Method
OOSA	Object-Oriented Systems Analysis
UFRGS	Universidade Federal do Rio Grande do Sul

LISTA DE FIGURAS

[1] Serão mantidas as abreviaturas originais dadas pelos autores. A ambiguidade é resolvida localmente no texto.

LISTA DE TABELAS

RESUMO

A Matriz Square é proposta como mecanismo integrador dos modelos construídos, a partir da prática diagramática na fase de análise, pela metodologia “Object Modeling Technique” (OMT) de Rumbaugh et al [RUM91]. Os modelos gerados são independentes mas possuem conexões explícitas que permitem a integração. A Matriz Square é uma ferramenta gráfica que produz um novo modelo (o modelo matricial), que complementa aqueles gerados pela metodologia OMT.

Classes de conexões explícitas, constituídas por vínculos entre os elementos modelados, são estabelecidas e representadas na Matriz Square para viabilizar a verificação de consistência integrando os diferentes modelos da metodologia OMT. O modelador pode buscar esta integração pela verificação visual ou através da aplicação de regras de consistência sobre as conexões explícitas detectadas no modelo matricial. A verificação visual é possível pela utilização de filtros sobre a matriz, obtendo-se fragmentos da própria matriz, onde podem ser analisados padrões, e paralelamente, fragmentos correspondentes dos diagramas, permitindo observar a coerência entre eles. As regras de consistência podem ser aplicadas localmente sobre um elemento modelado ou globalmente sobre toda a matriz, gerando relatório de consistência.

PALAVRAS-CHAVE: Engenharia de Software, Análise de Software, Object Modeling Technique, Consistência de modelos, Integração de modelos.

ABSTRACT

Matriz Square: Integrator Device for OMT Methodology Models

Matriz Square is proposed as a integrator device for the origined models from the diagramatic pratic in the analysis phase by the “Object Modeling Technique” (OMT) of the Rumbaugh et al [RUM91]. The generated models are independent but they have explicit connections to make possible the integration. Matriz Square is a graphic tool that produce a new model (the matrix model) and it complements the OMT models.

Explicit connections classes are settled and they consist of joins between modeled elements. They are represented in the Matriz Square to enabling the consistency verification between OMT models. User may get the integration using visual verification or consistency rules on the explicit connections in the matrix model. The visual verification is possible by filters on the matrix. Filtered matrix fragments allow the analysis of patterns, and they provide a parallel observation of correspondent diagram fragments. This visual verification detects the coherency between models. The consistency rules may be aplicated on a modeled element (locally) or verify all matrix (totally), generating a consistency report.

KEYWORDS: Software Engineering, Software Analysis, Object Modeling Technique, Models Consistency, Models Integration.

1 INTRODUÇÃO

1.1 Motivação

O paradigma de orientação a objetos têm ganho cada vez mais importância na área de análise de sistemas. A análise orientada a objetos (AOO), abordada por diferentes metodologias, pode resultar em múltiplos modelos e duas dimensões principais têm sido consideradas: a estrutural e a dinâmica. Uma terceira dimensão, a funcional, é tratada em algumas metodologias, entre elas a "Object Modeling Technique" (OMT) de Rumbaugh et al. [RUM91].

Ao usar múltiplos modelos, uma metodologia faz com que seja necessária a integração destes modelos com coerência na busca de uma descrição precisa do domínio do problema. A verificação de consistência dos elementos modelados em perspectivas diferentes, entretanto, é ainda muito pouco considerada [DED94] [HAY91]. Ela é, porém, uma questão inevitável em favor da qualidade da especificação. A integração completa das informações geradas, além de qualidade, dá produtividade à fase de análise e faz com que a descrição torne-se mais clara, consistente e completa.

1.2 Objetivo

O objetivo desta dissertação é propor um mecanismo, chamado Matriz Square, para integração dos modelos da metodologia OMT com consistência. Para isso, os elementos modelados através da OMT são traduzidos das técnicas diagramáticas para um modelo único no formato matricial. A matriz formada serve como complemento aos diagramas, auxiliando a unificação dos elementos modelados e a integração dos modelos gerados.

A metodologia OMT utiliza 3 modelos (de Objetos, Dinâmico e Funcional) que dão visões diferentes do domínio do problema. Estes modelos são gerados a partir dos diagramas de Objetos, de Fluxo de Eventos, de Estado e de Fluxo de Dados e os elementos modelados necessitam estar integrados para que produzam uma descrição coerente. O mecanismo de integração proposto necessita do apoio de uma ferramenta que automatize a construção da Matriz Square, gerando um modelo matricial, a medida que a atividade diagramática se desenvolve. Deste modo, os elementos modelados nos diagramas são, também, representados na matriz e o modelador pode:

- visualizá-los sob um novo formato em fragmentos de matriz, obtidos através de filtros, que destacam também fragmentos correspondentes de diagramas, ou
- ter em seu auxílio regras de consistência para orientá-lo na prática de diagramação.

Destas duas maneiras a Matriz Square complementa a metodologia OMT com o objetivo de cumprir o papel de mecanismo integrador dos seus modelos.

Mais concretamente, o objetivo desta dissertação pode ser subdividido em três partes, conforme segue:

- propor uma nova ferramenta gráfica, a Matriz Square, geradora do modelo matricial, abordando a representação unificada dos elementos modelados, com o principal objetivo de consistir os modelos da metodologia OMT, integrando-os numa descrição coerente do domínio do problema;
- Caracterizar uma ferramenta que automatize a construção do modelo matricial, suportando a metodologia OMT e
- apresentar um protótipo desta ferramenta, cumprindo algumas de suas funções.

1.3 Plano de Apresentação

A proposta é apresentada e discutida e é analisada a contribuição que ela dá à integração dos modelos da OMT. Com este objetivo, a dissertação está organizada em oito diferentes tópicos de discussão:

- O capítulo 2 refere-se aos modelos utilizados na análise orientada a objetos, abordando uma visão da modelagem de software, e ao problema da integração dos modelos de uma metodologia com o objetivo de melhorar a qualidade da especificação.

- O capítulo 3 apresenta a metodologia OMT, discutindo suas características, seus modelos e os elementos neles modelados.

- O capítulo 4 trata do estado da arte em relação ao problema de integração dos modelos de uma metodologia.

- O capítulo 5 apresenta os fundamentos para a proposta de integração dos modelos da OMT.

- Os capítulos 6 a 9 apresentam a proposta de integração constituída pela Matriz Square. O capítulo 6 apresenta definição, origem, interpretação e notação da Matriz Square e discute as vantagens da representação unificada. O capítulo 7 estabelece as características fundamentais de uma ferramenta que apoie a construção da Matriz Square, mostrando componentes e funções necessárias. O capítulo 8 detalha a integração dos modelos da OMT através do uso de filtros sobre a Matriz Square, permitindo a consistência de forma visual, e aborda os padrões de modelagem que podem ser detectados. O capítulo 9 detalha a integração através da aplicação de regras de consistência sobre a Matriz Square.

- O capítulo 10 apresenta dois estudos de caso: preparação de conferência de trabalho da IFIP e a máquina de caixa automático (ATM).

- O capítulo 11 apresenta um protótipo da ferramenta Square mostrando suas funções, o uso de filtros e padrões e a aplicação das regras de consistência.

- O capítulo 12 conclui sobre as contribuições da ferramenta Square para a integração dos modelos da metodologia OMT e sobre trabalhos futuros abordando o uso do modelo matricial.

2 MODELAGEM DE SOFTWARE E INTEGRAÇÃO DE MODELOS

> "*A realidade é um tecido sem costura. Tudo o que se disser sobre ela, qualquer descrição dela é apenas um resumo.*" Rumbaugh et al [RUM91]

2.1 Introdução

Neste capítulo são discutidos aspectos sobre o desenvolvimento de software, principalmente quanto à fase de análise e ao uso de modelos na descrição de um sistema. A seção 2.2 introduz o problema da crescente complexidade do software, identifica a fase de análise, discute a influência da especificação na qualidade do sistema produzido e relaciona o paradigma de orientação a objetos com a fase de análise, listando algumas metodologias conhecidas. A sessão 2.3 verifica o uso de modelos na análise orientada a objetos e discute o problema da integração dos modelos de uma metodologia.

2.2 Desenvolvimento de Software

2.2.1 Introdução

O termo "Software Engineering" foi introduzido pela primeira vez nos anos 60 com o objetivo de discutir o que era então chamado de "software crisis". A introdução da terceira geração de computadores viabilizou tal ordem de recursos que tornou possível a criação de aplicações que requerem a construção de grandes sistemas de software, até então impossíveis de maneira confiável [SOM92].

As primeiras experiências com a construção de grandes sistemas de software mostrou que os métodos existentes de desenvolvimento não eram suficientemente bons. Técnicas aplicáveis à pequenos sistemas não podiam ser utilizadas. Novas técnicas e métodos eram necessários para controlar a complexidade inerente dos grandes sistemas de software [SOM92].

Embora tenham ocorrido reais melhoramentos nos métodos e nas técnicas da engenharia de software e nas ferramentas para desenvolvimento de sistemas, necessitamos ainda de melhores ferramentas, técnicas e métodos [SOM92]. Rentsch [REN82] predisse que a programação orientada a objetos seria nos anos 80 o que a programação estruturada foi nos anos 70. Entretanto, um grau de complexidade foi incluído, já que a adoção do paradigma orientado a objetos causou uma dependência muito mais forte do uso de ferramentas e de ambientes de suporte do que as técnicas de projeto procedural [KOR90].

2.2.2 A Fase de Análise

Ghesi et al [GHE91] defendem que o termo “especificação”, ainda que usado em diferentes estágios do desenvolvimento de software, é mais apropriado quando define aquilo que a implementação necessita prover. E isto ocorre durante a fase de análise.

Rumbaugh et al [RUM91] definem a fase de análise como aquela que se ocupa com o delineamento de um preciso, conciso, compreensível e correto modelo do mundo real. Deve estabelecer o que deve ser feito, sem restrições à maneira como deve ser feito e evita decisões de implementação.

2.2.3 Qualidade na Especificação de Sistemas

Podem ser construídas especificações formais ou informais, operacionais ou descritivas, mas sempre estas especificações devem ter qualidades que as tornem claras, consistentes e completas [GHE91].

O objetivo da fase de análise é formular e comunicar descrições do domínio do problema. As notações usadas necessitam ser intuitivas, expressivas e precisas. As atuais técnicas de análise orientada a objetos conseguem produzir notações intuitivas e expressivas [HAY91]. A precisão, também necessária, depende da perícia do modelador, do seu conhecimento sobre o problema e da técnica e/ou ferramenta que utiliza para construir a descrição. A verificação da consistência é fundamental para alcançar a precisão: para produzir modelos coerentes e para dar qualidade à descrição.

A construção de um modelo rigoroso do domínio do problema força o desenvolvedor a encontrar

os erros no processo do desenvolvimento o mais cedo possível, enquanto eles são mais fáceis de corrigir [RUM91]. O esforço de correção é menor já que os erros ainda estão confinados localmente.

A consistência da especificação de sistemas tem sido identificada como um dos fatores que afetam a qualidade do software produzido. A consistência possibilita um projeto uniforme e tem sido identificada como responsável direta por itens que determinam a qualidade do software: precisão, flexibilidade, manutenibilidade e confiabilidade [JAN94].

2.2.4 Análise Orientada a Objetos

O paradigma de orientação a objetos inclui quatro conceitos principais, segundo Rumbaugh et al [RUM91]:

- Identidade. Os dados são subdivididos em entidades discretas e distintas, denominadas objetos.
- Classificação. Os objetos com a mesma estrutura de dados (atributos) e o mesmo comportamento (operações) são agrupados em uma classe.
- Polimorfismo. Uma operação pode atuar de modos diferentes em classes diferentes.
- Herança. Atributos e operações são compartilhados entre classes com base em um relacionamento hierárquico.

Coad e Yourdon [COA92a] entendem que a equação “classes e objetos + herança + comunicação por mensagens” define o que se entende como orientado a objetos. Wirfs-Brock et al [WIR90] definem a orientação a objetos como uma abordagem que tenta gerenciar a complexidade inerente dos problemas do mundo real, através da abstração do conhecimento, encapsulando-a em objetos.

O paradigma de orientação a objetos, nascido na área da programação, tem cada vez mais ganho importância na análise de sistemas. Diversas metodologias têm sido propostas, entre elas:

- Object Oriented Development (OOD) [BOO86], de Grady Booch;
- Object-Oriented Systems Analysis (OOSA) [SHL88], de Sally Shlaer e Stephen J. Mellor;
- Object-Oriented Requirements Specification Method (OORSM) [BAI89], de Sidney C.

Bailin;

- Object-Oriented Design (OOD) [WIR90], de Rebecca Wirfs-Brock, Brian Wilkerson e Lauren Wiener;
- Object Modeling Technique (OMT) [RUM91], de James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy e Willian Lorensen;
- O*Model [BRU91], de Joël Brunet;
- Object-Oriented Analysis (OOA) [COA92a], de Peter Coad e Edward Yourdon;
- Object-Oriented Systems Analysis (OOSA) [EMB92], de D. W. Embley, B. D. Kurtz e S. N. Woodfield;
- Object-Oriented Analysis and Design (OOAD) [NER92], de Jean-Marc Nerson;
- Object Behavior Analysis (OBA) [RUB92], de Kenneth S. Rubin e Adele Goldberg;
- OOO Metodology (OOO) [HEN93], de B. Henderson-Sellers e J. M. Edwards e
- Model-Driven Entity-Relationship Object-Oriented Development Method (MERODE) [DED94], de G. Dedene e M. Snoeck.

Reinoso e Heuser [REI94a] definem a análise orientada a objetos como "*o processo de construção de modelos do domínio do problema, identificando e especificando um conjunto de objetos semânticos que interagem e comportam-se conforme os requisitos estabelecidos para o sistema*".

Korson e McGregor [KOR90] sustentam que as fases de análise e de projeto orientados a objetos possuem uma linha divisória muito difusa porque tanto uma como outra modelam objetos. A diferença, segundo esses autores, é que na fase de análise são identificados objetos do domínio do problema enquanto que na fase de projeto são incluídos objetos necessários para especificar uma solução orientada ao computador. Entre as metodologias listadas anteriormente, algumas não fazem uma clara distinção entre as fases de análise e de projeto.

2.3 Modelagem de Software

2.3.1 Introdução

Modelos são utilizados em diversas áreas onde o homem precisa da abstração para conceber algo novo. Com este propósito são construídos modelos de aviões e carros, modelos arquitetônicos e outros tantos. Segundo Rumbaugh et al [RUM91], os modelos servem a diversos objetivos, quais sejam:

- para testar uma entidade física antes de lhe dar forma,
- para comunicar como está sendo concebido algo a ser construído.
- para os autores conseguirem visualizar suas idéias e
- para reduzir a complexidade. E aqui ficam englobados todos os itens anteriores.

Durante a fase de análise, no desenvolvimento de software, modelos ganham importância, pois descrevem o domínio do problema a ser especificado. Nesta fase, eles são abstrações construídas para que um problema seja compreendido antes da implementação de uma solução [RUM91].

Um bom modelo deve incorporar os aspectos fundamentais de um problema e ignorar os demais; não deve procurar a verdade absoluta mas sim a adequação a algum propósito [RUM91].

2.3.2 Modelos na Análise Orientada a Objetos

A análise orientada a objetos é essencialmente baseada em modelagem. Podem ser levadas em conta duas dimensões principais [REI94a]: (a) a dimensão estrutural dos objetos, que inclui a identidade de cada objeto, sua classificação, seu encapsulamento e seus relacionamentos, e (b) a dimensão dinâmica do comportamento dos objetos, que inclui a definição dos estados válidos dos objetos e a especificação dos eventos que causam a transição desses estados. Uma terceira dimensão também é reconhecida em alguns casos: a dimensão funcional dos requisitos [REI94a], que estabelece as funções de transformação do sistema.

A tabela 2.1 mostra como algumas metodologias orientadas a objetos trabalham modelos. A identificação de cada metodologia refere-se à lista apresentada em 2.2.4, vista na seção anterior.

Metodologia	dimensões		
	estrutural	**dinâmica**	**funcional**
OOD [BOO86]	Diagrama de Objetos	Modelo Dinâmico	Diagrama de Fluxo de Dados de objetos
OOSA [SHL88]	Modelos de Informação	Modelos de Estados e Modelos de Comunicação de Objetos	Modelos de Processos
OORSM [BAI89]	Diagrama de Entidade-Relacionamento		Diagrama de Fluxo de Dados de Entidades
OOD [WIR90]	Diagramas de Hierarquia e Especificações de Classes e Subsistemas	Diagramas de Colabo-rações e Especificações de Contratos	
OMT [RUM91]	Modelo de Objetos	Modelo Dinâmico	Modelo Funcional
O*Model [BRU91]	Descrição Textual e Especificação de Interrelações Estáticas	Especificação de Interrelações Dinâmicas	
OOA [COA92a]	Níveis de classes&Objetos, de estruturas, de assuntos e de atributos	Nível de serviços	
OOSA [EMB92]		Redes de Estado e Modelo de Comunicação de Objetos	
OOAD [NER92]	Modelo Estático	Modelo Dinâmico	
OBA [RUB92]	Modelos de Objetos	Modelos da Dinâmica do Sistema	
OOO [HEN93]	Diagramas de análise, de Projeto, de Herança e de Interface de Classe	Diagramas de Contrato	
MERODE [DED94]	Modelo de Serviços		Modelos de Projeto e de Tecnologia

Tabela 2.1. Algumas metodologias orientadas a objetos e as dimensões modeladas

2.3.3 O Problema da Integração de Modelos de uma Metodologia

A verificação de consistência na fase de análise envolve aspectos mais amplos que o problema da integração de modelos, isto é, da consistência entre modelos. A verificação de consistência, de uma maneira ou de outra, é uma tarefa que nenhuma metodologia descarta. Mas quando a intenção é modelar um sistema sob dimensões diferentes, quanto mais independência se deseja para cada uma, menos conexões explícitas existirão entre elas. Mais difícil, nesses casos, será a verificação de consistência entre as dimensões trabalhadas.

Além disso, a dualidade "*mais modelos menos complexos x menos modelos mais complexos*" tem desafiado a eficiência da modelagem de sistemas. É o caso, também, das metodologias propostas para a análise orientada a objetos. Na tentativa de obter uma descrição precisa, quanto menor a verificação de consistência entre os modelos, mais importante torna-se o conhecimento do

modelador sobre o domínio do problema para estabelecer semântica e consistência [HAY91].

A necessidade de ser intuitivos e possuir expressão e precisão exige que os modelos da análise tenham: ausência de ambigüidade, abstração e consistência [HAY91]. A construção de modelos consistentes no sentido mais amplo tem sido buscada através do formalismo e sua necessidade tem sido discutida pela comunidade voltada ao paradigma de orientação a objetos [CHA91].

Abordando a integração de modelos mas sem perder de vista a verificação de consistência no sentido mais geral (correção, adequação à especificação), alguns critérios são formulados por Dedene e Snoeck [DED94] para auxiliar a análise de metodologias:

- existência de definição formal própria da metodologia para a sintaxe e para a semântica das técnicas usadas para modelar aspectos estáticos, dinâmicos e de interação entre objetos;
- existência de definição do comportamento global do sistema em termos dos objetos individuais e da interação entre eles e
- existência de procedimento de verificação da consistência entre os modelos;

Usando estes critérios, Dedene e Snoeck [DED94] analisam metodologias da AOO (OMT de Rumbaugh et al, FUSION de Hayes e Coleman, OSA de Embley et al, OOSA de Shlaer e Mellor, OOA de Coad e Yourdon, O/B de Kappel e Schrefl, OOD de Booch e OOD de Wirfs-Brock et al) e chegam a seguinte conclusão:

> "E*sta abordagem deixou claro que é feito somente um uso limitado de técnicas formais. Principalmente que são muito pouco consideradas a definição de sintaxe e de semântica dos aspectos dinâmicos (incluindo o comportamento global do sistema) e a verificação de consistência entre modelos ... A ausência de publicações sobre o assunto de verificação de consistência entre modelos ou verificação de precisão, para modelos de comportamento e de interação, indicam que verificações formais são um tópico muito complexo. A escolha de técnicas para modelagem de aspectos dinâmicos é crucial se desejarmos obter procedimentos de verificação aceitáveis."* [DED94].

A verificação de consistência entre os modelos é um modo de assegurar a qualidade do

desenvolvimento de software. As notações e o processo utilizados por uma metodologia devem auxiliar a produção de um conjunto coerente de modelos que constituem uma descrição fundamentada e consistente do domínio do problema [HAY91].

2.4 Conclusões

Metodologias de desenvolvimento de software têm sido concebidas com o objetivo de controlar a complexidade cada vez maior dos grandes sistemas. Na fase de análise, principalmente no paradigma de orientação a objetos, o uso de modelos é fundamental e as notações usadas necessitam ser intuitivas, expressivas e precisas. A verificação de consistência deve ser uma preocupação sempre presente, pois a necessidade de representar a realidade exige que os modelos da análise não apresentem ambigüidades e possuam abstração e consistência. Quando vários modelos são utilizados, o que não é incomum, deve ser buscada a integração entre eles para que a consistência seja conseguida no todo.

O próximo capítulo apresenta a metodologia OMT [RUM91] que é o alvo da proposta desta dissertação no sentido de integração de seus modelos.

3 A METODOLOGIA OMT

> "*Consideramos útil modelar um sistema a partir de três pontos de vista diferentes porém relacionados, cada um capturando importantes aspectos do sistema, mas todos necessários para uma descrição completa.*"
> Rumbaugh et al [RUM91]

3.1 Introdução

Neste capítulo é discutida a metodologia OMT. A seção 3.2 identifica suas fases e caracteriza a metodologia dentro do paradigma de orientação a objetos. A seção 3.3 apresenta os modelos construídos pela OMT, os diagramas utilizados e os elementos neles modelados. A seção 3.4 apresenta algumas considerações sobre a metodologia OMT.

3.2 A Metodologia

A "Técnica de Modelagem de Objetos" ou OMT (Object Modeling Technique) de Rumbaugh et al [RUM91] é representativa de um grupo definido como técnicas integracionistas [REI94a], isto é, técnicas da análise orientada a objetos que integram modelos separados das dimensões estrutural, dinâmica e funcional. A escolha desta metodologia neste trabalho deve-se justamente ao fato dela utilizar múltiplos modelos e, também, por ser fartamente documentada e bem conhecida.

Segundo os autores, os modelos gerados separam um sistema em visões ortogonais que podem ser trabalhadas e representadas com uma notação uniforme. Cada um dos modelos evolui durante o ciclo de desenvolvimento. A metodologia OMT, conforme Rumbaugh et al [RUM91] compõe-se de três fases: (1) a análise, que se preocupa com a compreensão e a modelagem da aplicação e do domínio em que atua; (2) o projeto do sistema, que o organiza em subsistemas, sendo adicionadas construções do domínio da solução, e (3) o projeto de objetos, onde são elaborados, refinados e otimizados os modelos da análise para a produção de um projeto prático.

A metodologia é apresentada como uma abordagem de desenvolvimento de software orientado a objeto, com as seguintes características [RUM91]: (1) desloca o esforço de desenvolvimento para a análise; (2) enfatiza a estrutura de dados e não as funções; (3) compõe um processo de desenvolvimento contínuo e (4) não é seqüencial, mas interativa.

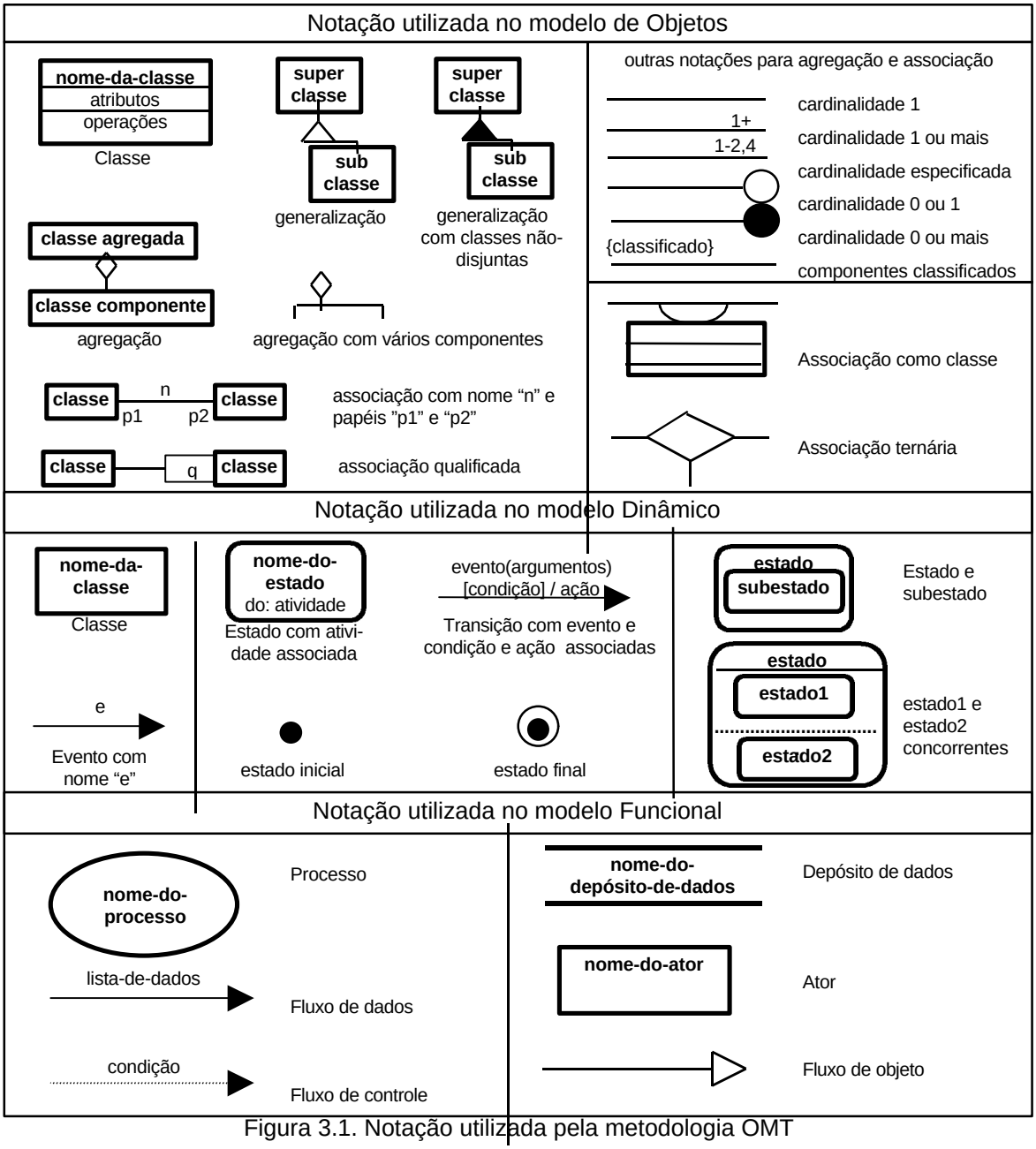


Figura 3.1. Notação utilizada pela metodologia OMT

A figura 3.1 mostra a notação utilizada na metodologia OMT, para o modelo de Objetos (diagrama de Objetos), o modelo Dinâmico (diagramas de Fluxo de Eventos, de Estados) e o

modelo Funcional (diagrama de Fluxo de Dados). A figura 3.2 mostra exemplos dos diagramas utilizados para construir os modelos de Objetos, Dinâmico e Funcional, na metodologia OMT.

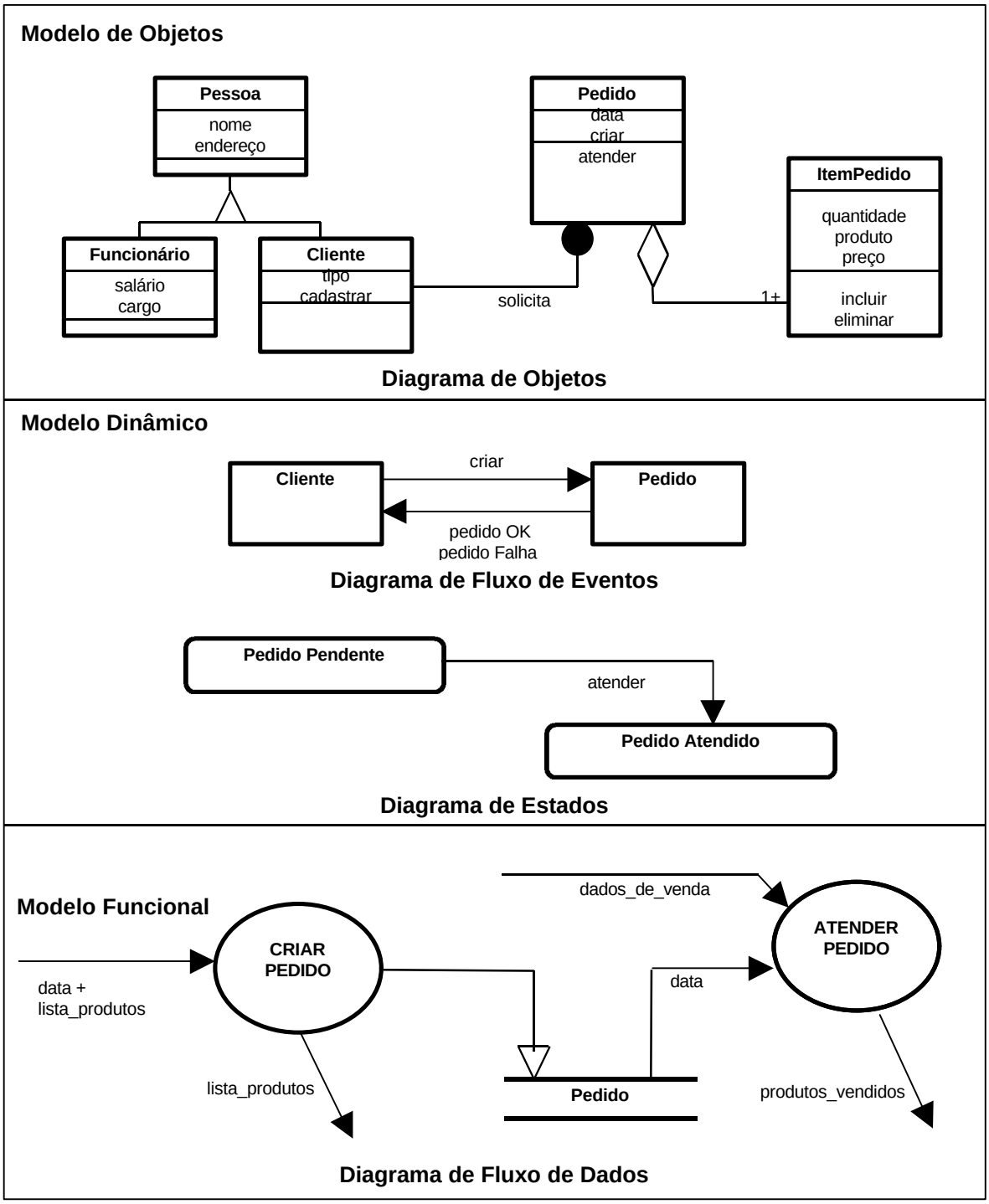


Figura 3.2. Exemplos com os diagramas da metodologia OMT

3.3 As Três Dimensões

Na fase de análise, a OMT utiliza três modelos, cada um deles capturando uma dimensão diferente do sistema. O modelo Funcional estabelece “o quê” acontece, o modelo Dinâmico define “como” e “quando” e o modelo de Objetos descreve os objetos que participam destes acontecimentos e como se relacionam. Representam aspectos diferentes do sistema:

- **Modelo de Objetos**: representa os aspectos estáticos e estruturais dos dados de um sistema.
- **Modelo Dinâmico**: representa os aspectos comportamentais de controle de um sistema.
- **Modelo Funcional**: representa os aspectos relativos às transformações das funções de um sistema.

Os autores da OMT consideram útil a descrição de um sistema a partir de três dimensões relacionadas mas diferentes entre si, cada uma envolvendo importantes aspectos, todos necessários para uma descrição completa [RUM91].

3.3.1 Modelo de Objetos

O modelo de Objetos especifica a estrutura dos objetos: identidade, relacionamentos, atributos e operações. No diagrama de objetos é especificada a estrutura dos objetos, é construída uma hierarquia de classes e são definidas as associações existentes entre elas. Proporciona a estrutura necessária na qual podem ser colocados os modelos Dinâmico e Funcional [RUM91].

O diagrama de objetos oferece uma notação gráfica para a modelagem de objetos e seus inter-relacionamentos. No diagrama de objetos são consideradas as classes de objetos, seus atributos e operações, e os relacionamentos envolvidos: associações, agregações e generalizações. A seguir são descritos os elementos modelados:

- **Classe de Objetos** é definida como um grupo de objetos com propriedades semelhantes (atributos), o mesmo comportamento (operações), os mesmos relacionamentos com outros objetos e a mesma semântica. Objeto é um conceito, uma

abstração, algo com limites nítidos e significado em relação ao problema em causa.

- **Atributo** é um valor de dado guardado pelos objetos de uma classe.
- **Operação** é uma função ou transformação que pode ser aplicada a ou por objetos de uma classe. Todos os objetos de uma classe compartilham as mesmas operações. Objeto-alvo é o argumento implícito que toda operação tem, é o objeto que sofre os efeitos da operação. Método é a implementação de uma operação para uma classe. Assinatura de um método é o conjunto formado pelo seu nome, a quantidade e tipos de argumentos e o tipo do valor resultante. As operações podem ser triviais (as que não têm necessidade de serem listadas ou especificadas) ou não-triviais. Operações de acesso são operações triviais. Elas lêem ou atualizam os atributos ou as ligações de um objeto. As operações não-triviais podem ser: consultas, ações ou atividades. Uma consulta é uma operação sem efeitos colaterais no estado externamente visível de um objeto. É uma função pura. Os conceitos de ação e de atividade aparecem no modelo Dinâmico.
- **Associação** descreve um grupo de ligações com estrutura e semântica comuns. São intrinsecamente bidirecionais. Ligação é uma conexão física ou conceitual (entre objetos) que mostra um relacionamento entre dois (ou mais) objetos. Papel é uma extremidade de uma associação. Identifica inequivocamente um objeto ou um conjunto de objetos associado a outro objeto na extremidade oposta. É um atributo derivado da classe de objetos da extremidade oposta.
- **Associação como classe** é uma associação modelada também como classe. Os autores da OMT aconselham esta opção quando as ligações podem participar de associações com outros objetos ou quando as ligações estão sujeitas a operações.
- **Agregação** é um relacionamento “parte-todo” ou “uma-parte-de” no qual os objetos que representam os componentes de algo são associados a um objeto que representa a estrutura inteira. A agregação é transitiva e anti-simétrica. Agregação é um modo de associação forte na qual um objeto agregado é feito de componentes.
- **Generalização** é o relacionamento entre uma classe e uma ou mais versões especializadas dela. No relacionamento, a classe mais genérica é chamada de superclasse e cada versão especializada a partir dela é denominada subclasse. A generalização é transitiva através de um número arbitrário de níveis. Classes disjuntas são aquelas subclasses cujas instâncias não podem ser de mais de uma delas. Nas classes não-disjuntas isso pode ocorrer.

No modelo de Objetos são estabelecidos os seguintes relacionamentos (figura 3.3) entre os elementos modelados:

- Uma classe, numa hierarquia de classes, representa o papel de superclasse ou de subclasse. Uma generalização liga uma superclasse, mais genérica, a uma subclasse, mais especializada.
- Uma classe é composta de objetos, todos do mesmo tipo, possuindo as mesmas propriedades (atributos), o mesmo comportamento (operações) e os mesmos relacionamentos (ligações) com outros objetos.
- Uma agregação entre uma classe, denominada agregada, e outra, denominada componente, é composta por uma ou mais ligações entre objetos daquelas classes.
- Uma associação, que vincula duas classses, é composta por uma ou mais ligações entre objetos daquelas classes.

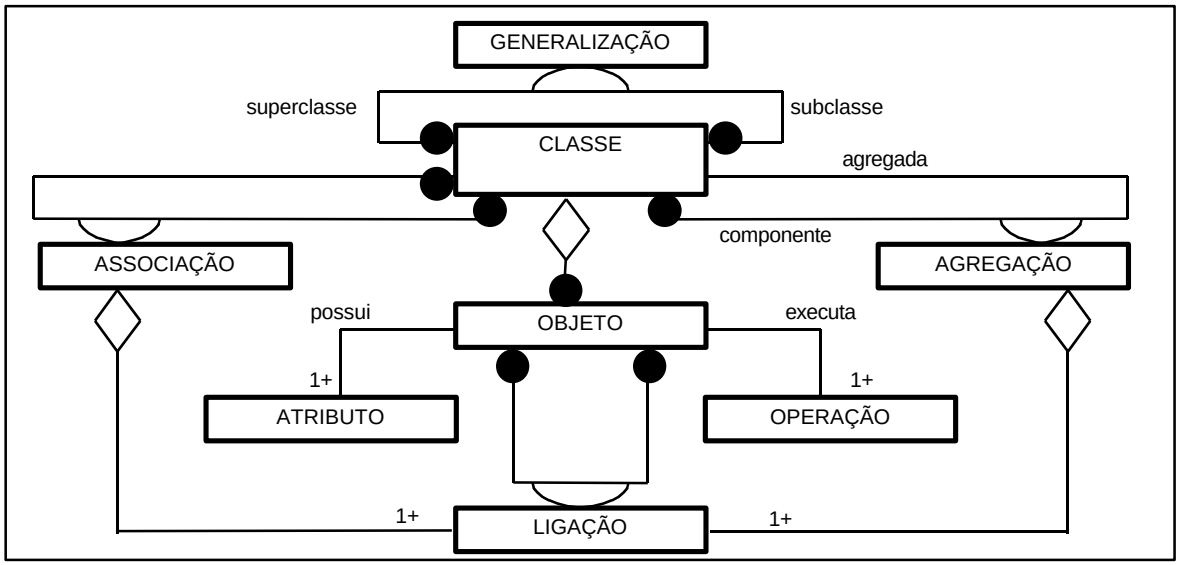


Figura 3.3. Metamodelo referente ao modelo de Objetos OMT

3.3.2 Modelo Dinâmico

O modelo Dinâmico preocupa-se com os aspectos relacionados ao tempo e às seqüências de operações. São mostrados eventos, a seqüência em que ocorrem e as transformações de estados decorrentes. No diagrama de fluxo de eventos é determinado como ocorre a comunicação entre os objetos e no diagrama de Estados é descrito como os eventos afetam o ciclo de vida dos objetos. A seguir são descritos os elementos modelados:

- **Estado** é uma abstração dos valores de atributos e ligações de um objeto. Um estado pode estar associado a uma atividade ou a uma condição. Subestado é um refinamento (num diagrama de nível inferior) de um estado generalizado (superestado) de nível mais elevado. Os subestados (de um mesmo superestado) estão em "relacionamento-ou" entre si. Um objeto no superestado estará no subestado-1 ou no subestado-2 (e assim por diante). Estados concorrentes são os estados dos objetos-componentes de um objeto-agregado. Estado agregado corresponde ao estado do objeto-agregado. É a combinação dos estados dos objetos-componentes. O estado agregado é um estado do objeto-componente-1 e um estado do objeto-componente-2 (e assim por diante). Estados concorrentes num único objeto são os estados em que um único objeto pode estar se for particionado em subconjuntos de atributos ou ligações, tendo cada um seu próprio diagrama. Neste caso o estado do objeto compreende um estado de cada diagrama .
- **Transição** é a transformação que um objeto sofre ao passar de um estado para outro. Constituem uma transição: um evento, uma condição e uma ação. Uma transição pode ocorrer sem que ocorra um evento, bastando apenas que uma condição seja verdadeira ou podendo ocorrer (transição automática) quando o estado atual finaliza a execução da atividade associada a ele.
- **Evento** é um estímulo individual de um objeto para outro. Um sistema de objetos interage pelo intercâmbio de eventos. Um evento não tem duração considerável em termos de escala de tempo, é algo que ocorre num dado momento. Cada evento tem origem num objeto e destino em outro. Tem uma causa (algo ocorrido no ambiente externo ao sistema ou a execução de uma operação do objeto origem do evento) e um efeito (alguma ocorrência no ambiente externo ao sistema ou a execução de uma operação do objeto destino do evento). Alguns eventos podem ser simples sinais de que

alguma coisa aconteceu, enquanto outros transportam valores de dados. Os valores de dados conduzidos por um evento são seus atributos. Neste caso um evento conduz informações de um objeto para outro.

- **Condição** é uma função booleana de valores de objetos. É válida dentro de um intervalo de tempo. Uma condição em uma transição de estado serve de guarda desta transição: se o evento associado ocorre, dispara a transição que só se efetiva se a condição vinculada for verdadeira.
- **Ação** é uma transformação que tem efeitos colaterais no objeto-alvo ou em outros objetos do sistema alcançáveis a partir do objeto-alvo. Não tem duração no tempo, é logicamente instantânea. Todas as ações devem ser definíveis em termos de atualizações dos atributos e das ligações. Uma ação, associada a uma transição, é uma resposta adicional ao evento ocorrido, além da própria transição de estado. Pode representar operações internas de controle, como o estabelecimento de valores de atributos ou a ativação de outros eventos.
- **Atividade** é uma operação com duração no tempo. Sempre tem efeitos colaterais. Aparece no diagrama de Estados como uma operação associada a um estado, sendo iniciada na entrada do estado e finalizada na saída.

No modelo Dinâmico são estabelecidos os seguintes relacionamentos (figura 3.4) entre os elementos modelados:

- Um sistema de objetos interage pelo intercâmbio de eventos.
- Cada objeto possui um ou mais estados válidos.
- Cada um dos estados de um objeto pode estar associado a uma atividade, executada por ele enquanto permanecer neste estado.
- Uma transição pode ser composta por um evento, que a dispara, por uma condição, que (se satisfeita) efetiva a transição, e por uma ação, que está associada à ocorrência da transição. Todos esses elementos são opcionais numa transição de estado: ela pode ser composta só por um evento ou só por uma condição.
- Alguns eventos, no caso de comporem transições junto a condições, dependem destas condições para disparar as transições.
- Alguns eventos, no caso de comporem transições de estado junto a ações, ativam estas ações ao dispararem as transições.

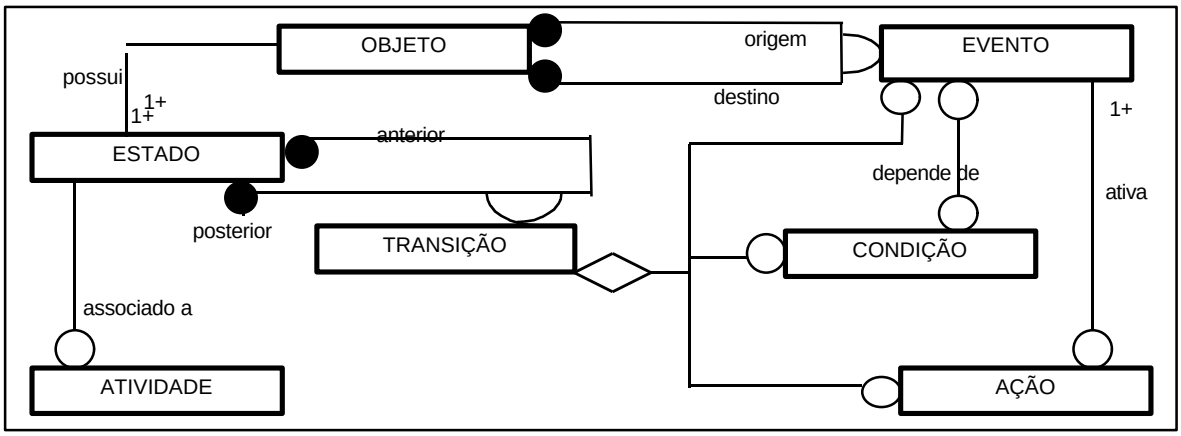


Figura 3.4. Metamodelo correspondente ao modelo Dinâmico da OMT

3.3.3 Modelo Funcional

O modelo Funcional identifica transformações de valores de dados através de funções, mapeamentos, restrições e dependências funcionais. São mostrados fluxos de dados e de controle, processos, atores e depósitos de dados. No diagrama de fluxo de dados são identificadas as transformações dos dados através dos processos e como esses dados fluem entre processos, atores e depósitos de dados. A seguir são descritos os elementos modelados:

- **Processo** é uma transformação de valores de dados. Os processos são implementados como métodos de operações sobre objetos. Cada processo do nível mais baixo, ou atômico, é uma função pura.
- **Fluxo de dados** é a interligação da saída de um objeto ou processo à entrada de outro objeto ou processo. Representa um valor de dado em algum ponto do processamento. Mostra todos os caminhos possíveis de computação de valores. Fluxo de dados como objeto, ou fluxo de objeto, é um fluxo de dados que gera um objeto e tem como destino um depósito de dados.
- **Ator** é um objeto ativo que dirige o diagrama de Fluxo de Dados produzindo ou consumindo valores. É vinculado às entradas e às saídas de um diagrama de Fluxo de Dados. As vezes são chamados de terminadores.
- **Depósito de dados** é um objeto passivo, em um diagrama de Fluxo de Dados, que armazena dados para uso futuro. Apenas atende solicitações de armazenamento ou de acesso aos dados.

- **Fluxo de Controle** é um valor booleano que afeta a maneira como um processo é avaliado. Não é um valor de entrada para o processo por si mesmo. Embora decisões e seqüências sejam problemas de controle que fazem parte do modelo Dinâmico, os fluxos de controle são úteis no modelo Funcional para que não sejam esquecidos e para que as dependências de seus dados possam ser mostradas. Apesar disso devem ser usados parcimoniosamente.

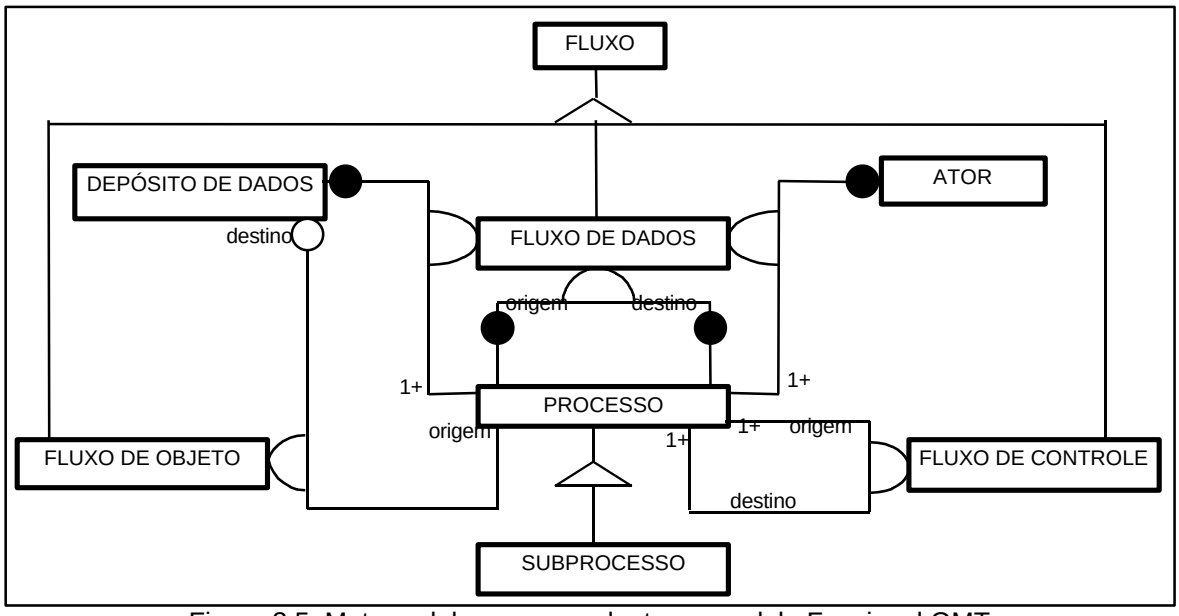


Figura 3.5. Metamodelo correspondente ao modelo Funcional OMT

No modelo Funcional são estabelecidos os seguinte relacionamentos (figura 3.5) entre os elementos modelados:

- Um fluxo pode ser um fluxo de objeto, quando gera um objeto, um fluxo de controle quando afeta a maneira como um processo é avaliado, ou um fluxo de dados quando simplesmente transporta dados.
- Um fluxo de objeto, obrigatoriamente, liga um processo a um depósito de dados.
- Um fluxo de controle parte de um processo, produzindo um valor booleano, e chega a outro processo que está sendo controlado.
- Um fluxo de dados pode interligar processos ou processos e depósitos de dados ou processos e atores.
- Um processo é composto por subprocessos, produzindo a decomposição funcional.

3.4 Algumas Considerações sobre a Metodologia OMT

- **Estratégia de modelagem**. Bruegge et al [BRU92] sugerem que a modelagem OMT acarreta freqüentes interações entre os modelos. A formulação do modelo de Objetos, isto é, a descrição das classes, atributos e associações é identificada na declaração do problema. Os modelos Funcional e Dinâmico são especialmente úteis como "fontes de métodos", isto é, eles auxiliam a identificação de "métodos" para classes do modelo de Objetos. Reinoso [REI94] define a estratégia OMT como "middle-out", isto é, não claramente "top-down" nem "bottom-up", devendo ser especificados os objetos, paralelamente, em termos estruturais, dinâmicos e funcionais.
- **Interdependência entre os modelos**. Segundo Reinoso [REI94] a potencialidade das ferramentas da modelagem dinâmica permite a inclusão de aspectos funcionais. Assim, alguns elementos podem ser modelados em mais de um local: no modelo Dinâmico e no modelo Funcional. Por exemplo: parâmetros de eventos correspondem aos dados dos fluxos de dados. Considera, ainda, este autor que existe uma multiplicidade de referências entre os diagramas das diferentes modelagens. Isto implica numa grande interdependência entre os modelos que acarreta modificações em diversos diagramas quando é alterado algum elemento num deles.
- **Notação**. Hayes e Coleman [HAY91] definem como formal a semântica da notação usada no modelo de Objetos, implicando em não-ambigüidade com um nível de abstração adequado. Por outro lado consideram as notações dos modelos Dinâmico e Funcional menos formais e propensas a ambigüidades já que necessitam de linguagem natural para definir conceitos básicos, como ações no modelo dinâmico e processos no modelo funcional.
- **Interação entre objetos**. Cada diagrama de Estados define o comportamento de um objeto individual e, segundo Hayes e Coleman [HAY91], nenhuma comunicação entre esses diagramas é definida. Por isso, o comportamento de uma comunidade de objetos interagindo não pode ser compreensível. Isto faz com que seja impossível, segundo aqueles autores, verificar se o modelo Dinâmico é consistente com o modelo Funcional.
- **O uso de DFDs**. Segundo Reinoso [REI94] a identificação de processos do DFD como implementadores de operações definidas para classes de objetos é difícil. Algumas funções podem acessar vários objetos e os autores da OMT não indicam um critério

útil para deter a explosão dos DFDs. É sugerido que a decomposição poderia ser feita até a obtenção de funções aplicadas a apenas um objeto. Já Hayes e Coleman [HAY91] consideram os DFDs como uma escolha inadequada para modelo funcional. Por um lado seu propósito é mostrar como valores de saída são derivados de valores de entrada, sem considerar a ordem na qual os valores são computados. Por outro há a correspondência entre processos dos níveis inferiores do DFD com operações sobre os objetos. Claramente, segundo os últimos autores, qualquer DFD que mostre o comportamento de objetos está prescrevendo, no mínimo em parte, uma computação particular mais do que especificando o comportamento do sistema.

- **Aplicação**. Bruegge et al [BRU92] sugerem que a metodologia OMT é adequada para aplicações de qualquer porte e que permite um equilíbrio entre a formalidade e a informalidade durante a modelagem.
- **A coerência entre modelos**. Segundo Hayes e Coleman [HAY91] os modelos da fase de análise da OMT não formam um conjunto coerente de descrições. Eles podem ser ambíguos e não é possível alcançar consistência entre os modelos adequadamente.
- **A integração dos modelos**. Segundo Monarchi e Puhr [MON92] metodologias que utilizam três modelos, como a OMT, os constróem geralmente independentes entre si e capturam diferentes aspectos do sistema. Contudo, segundo eles, esta abordagem combinativa deteriora-se quando ocorre a tentativa de integração dos diferentes pontos de vista na fase de projeto. Consideram que o problema maior acontece na correspondência entre processos do DFD e eventos e atividades do modelo Dinâmico e os objetos do modelo de Objetos.

3.5 Conclusões

A metodologia OMT [RUM91], adotando o desenvolvimento de software orientado a objeto, modela as dimensões (1) estrutural, que representa os aspectos estáticos e estruturais dos dados de um sistema, (2) dinâmica, que representa os aspectos comportamentais de controle de um sistema, e (3) funcional, que representa os aspectos relativos às transformações das funções de um sistema.

Segundo usuários da metodologia, há freqüentes interações necessárias entre os modelos e é exatamente com o objetivo de integração destes modelos que a proposta desta dissertação é desenvolvida, abordando soluções para representar e verificar aquelas interações.

O próximo capítulo apresenta um panorama do estado da arte, com alguns trabalhos importantes voltados para a integração de modelos de uma metodologia.

4 ESTADO DA ARTE

> "*Na prática, provas são totalmente impraticáveis. Assim, não se espere que o analista prove a consistência entre modelos a partir de casos gerais.*"
> Hayes e Coleman [HAY91]
> "*Certamente este não é o modo correto de garantir qualidade ao desenvolvimento de software.*"
> Dedene e Snoeck [DED94]

4.1 Introdução

Neste capítulo são apresentados os trabalhos mais representativos que introduzem soluções para integração dos modelos de uma metodologia. A seção 4.2 apresenta o trabalho de Hayes e Coleman, discutindo a coerência entre os modelos da OMT. A seção 4.3 apresenta o trabalho de Ramackers e Verrijn-Stuart, abordando a integração de perspectivas de modelagem. A seção 4.4 apresenta o trabalho de Dedene e Snoeck: M.E.R.O.DE.

4.2 Modelos Coerentes para Análise Orientada a Objetos (Fiona Hayes e Derek Coleman) [HAY91]

Hayes e Coleman [HAY91] trabalham com a metodologia OMT e sugerem uma solução de integração onde o objetivo é construir modelos coerentes com formalismo. São modificados os modelos informais da OMT e introduzido um outro, resultando em quatro modelos que buscam maior precisão: modelo de Objetos, modelo de Estrutura de Objetos, modelo Funcional e modelo Dinâmico.

O modelo de Estrutura de Objetos é introduzido como um refinamento do modelo de Objetos. Forma uma ligação entre o modelo de Objetos e os modelos Dinâmico e Funcional. Os relacionamentos são instanciados como atributos do tipo ponteiro explícito entre os objetos envolvidos. Cada classe torna-se um tipo registro (record type) contendo: um identificador de instâncias (self); definição de tipo para cada atributo da classe e um identificador para cada relacionamento (set of).

A figura 4.1 mostra um exemplo de modelo de Estrutura de Objetos para um editor de gráfico. A classe “Caixa”, com identificador de instâncias “Caixa_id”, possui localização (posn, dada por coordenadas), largura, altura e ligações com instâncias identificadas por “Linha_id”. Este é o identificador das instâncias da classe “Linha” que possui localização, comprimento e, por sua vez, ligações com instâncias da classe “Caixa”. Também é criada a união disjunta chamada “Objeto” através da união dos valores de “Caixa” rotulados por “caixa” e dos valores de “Linha” rotulados por “linha”. De forma similar é criada a união “Obj_id” e posteriormente é feito um mapeamento finito do domínio de “Obj_id” em “Objeto” definindo, assim, “Sistema”.

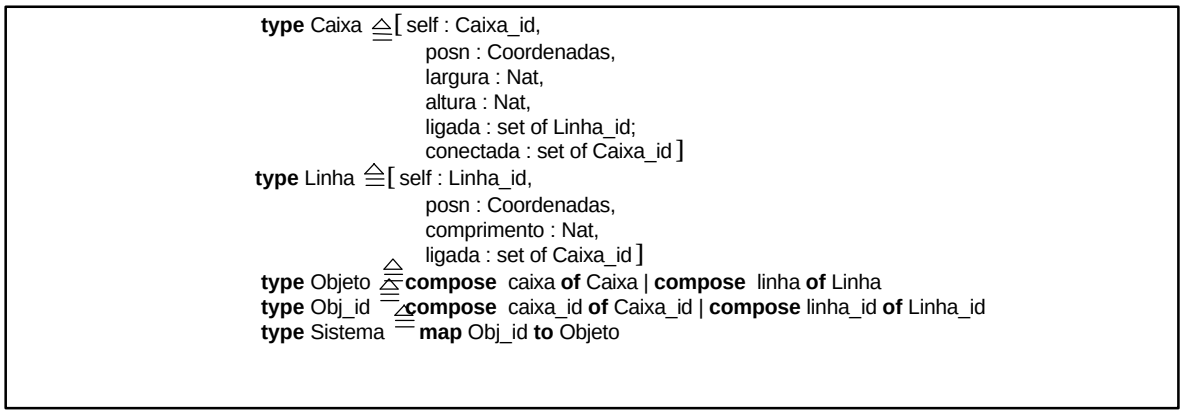


Figura 4.1. Exemplo de modelo de Estrutura de Objetos [HAY91]

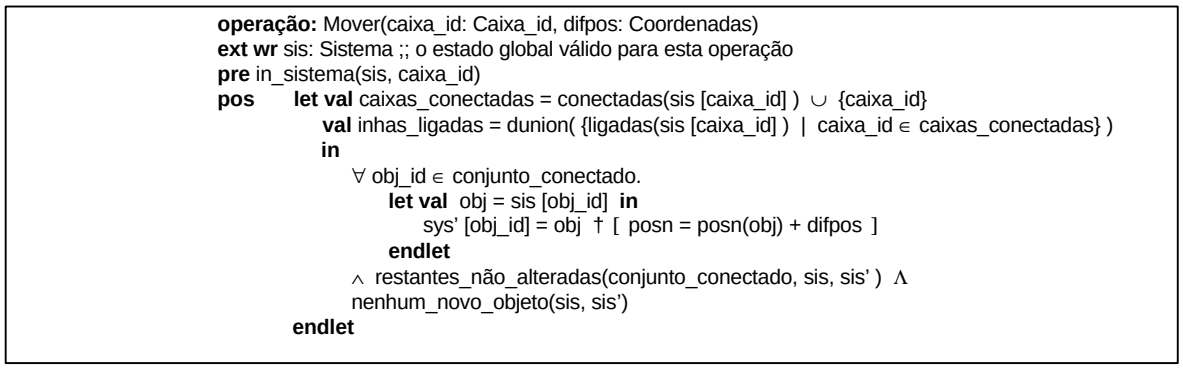


Figura 4.2. Exemplo de modelo Funcional [HAY91]

O modelo Funcional formal proposto, segundo os autores, dispensa o uso de DFDs e descreve as operações a nível de sistema através de especificações de pré e pós-condições sobre o modelo de Estrutura de Objetos. As operações recebem especificações formais como o exemplo mostrado na figura 4.2, com a operação “mover”, que desloca um retângulo e as linhas nele conectadas, num editor de gráfico.

O modelo Dinâmico especifica como os objetos interagem dinamicamente para viabilizar o comportamento descrito pelo modelo Funcional. Sendo, segundo os autores, as pré e pós-condições equivalentes nos modelos Funcional e Dinâmico, deve-se deduzir daí a sua consistência. Os efeitos dos eventos sobre atributos locais são definidos através da especificação de pré e pós-condições, conforme mostra a figura 4.3. Os efeitos globais são definidos usando-se comunicações de eventos síncronos entre objetos.

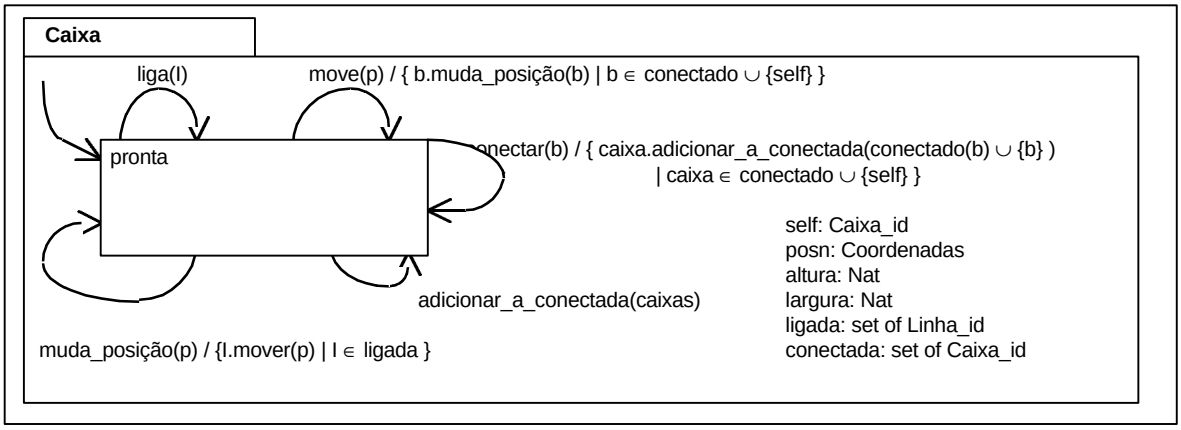


Figura 4.3. Exemplo de modelo Dinâmico [HAY91]

São sugeridas as seguintes etapas para a construção dos modelos:

- Descrição do problema em linguagem natural.
- Construção do modelo de Objetos junto com um dicionário de dados de classes, relacionamentos e atributos. Verificação da completeza através do estabelecimento de condições de consistência.
- Construção do modelo de Estrutura de Objetos, do modelo Funcional e do modelo Dinâmico.
- Teste de fluxos de eventos, verificando a consistência do modelo Dinâmico e do modelo Funcional.

4.3 Integrando Perspectivas de Sistemas de Informação com Objetos (Guus J. Ramackers e Alexander A. Verrijn-Stuart) [RAM91]

Segundo Ramackers e Verrijn-Stuart [RAM91], a modelagem integrada de um sistema provê uma base adequada que sustentará as múltiplas perspectivas que o descrevem: dos processos, dos dados e do comportamento.

Os autores esboçam o modo como os construtores fundamentais da modelagem proposta podem ser levados paralelamente numa descrição formal a ser usada para gerar as três perspectivas. A descrição feita é apenas a nível de sintaxe, somente indicando como redes de Petri coloríveis podem ser usadas para definir a semântica operacional.

O sistema consiste de um número de componentes (objetos) interrelacionados que agem entre si. Individualmente os componentes são caracterizados por uma única identidade, por um tipo (definição por intenção) e por uma classe (definição por extensão).

Um sistema S é definido pelos autores da seguinte forma:

S = <I, T, C> é a estrutura do sistema onde
I é um conjunto de identidades,
T é um conjunto de tipos de componentes (a ser definido) e
C: I $\rightarrow$ T é a função componente (que determina as instâncias componentes)

T é um conjunto de tuplas <X, R, A, L> onde
X são atributos, R são relacionamentos, A são ações e L é o ciclo de vida.

Assim, cada tipo componente T tem um conjunto de atributos e os relacionamentos estruturais (classificação, associação, especialização e agregação) da perspectiva dos dados são definidos como relações entre dois ou mais componentes. Uma ação de um componente pode ter a ela associadas entradas e saídas. Uma ação transforma suas entradas em suas saídas dependendo do estado em que se encontra o objeto. O ciclo de vida de um componente descreve a ordem na qual as ações dos objetos devem ser ativadas. A execução de uma ação por um componente individual constitui um evento. Os processos de um sistema descrevem a ativação dos eventos entre os objetos. O relacionamento de ativação entre os eventos é definido por uma rede de ações.

Levando-se em conta a definição estrutural de um sistema S como um número de tipos componentes T com atributos, relacionamentos, ações e um ciclo de vida, os conceitos de eventos e processos são definidos, segundo os autores, através dos aspectos dinâmicos de redes de Petri coloríveis.

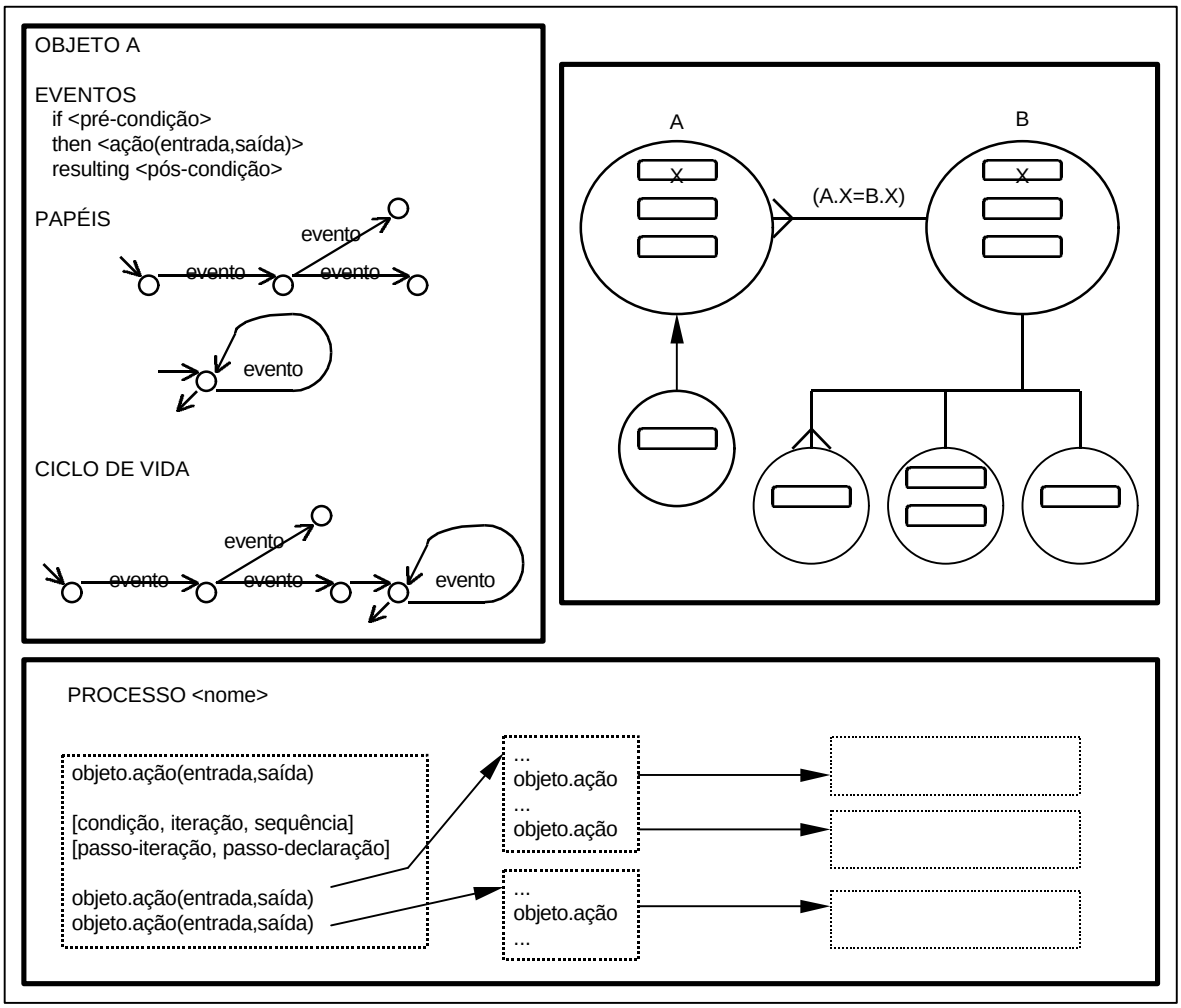


Figura 4.4. Exemplo de perspectivas de modelagem como visões separadas [RAM91]

Há dois modos fundamentais para se usar as perspectivas como visões num contexto integrado baseado em objetos:

- Primeiro: uma visão pode corresponder a uma das três perspectivas. Elas se complementam e podem ser vistas conjugadas. Esta abordagem é mostrada na figura 4.4.
- Segundo: os aspectos de dados, processos e comportamento dos objetos podem ser combinados em uma visão única. O objeto possibilita uma integração básica e um princípio de modularidade. Esta abordagem é mostrada na figura 4.5.

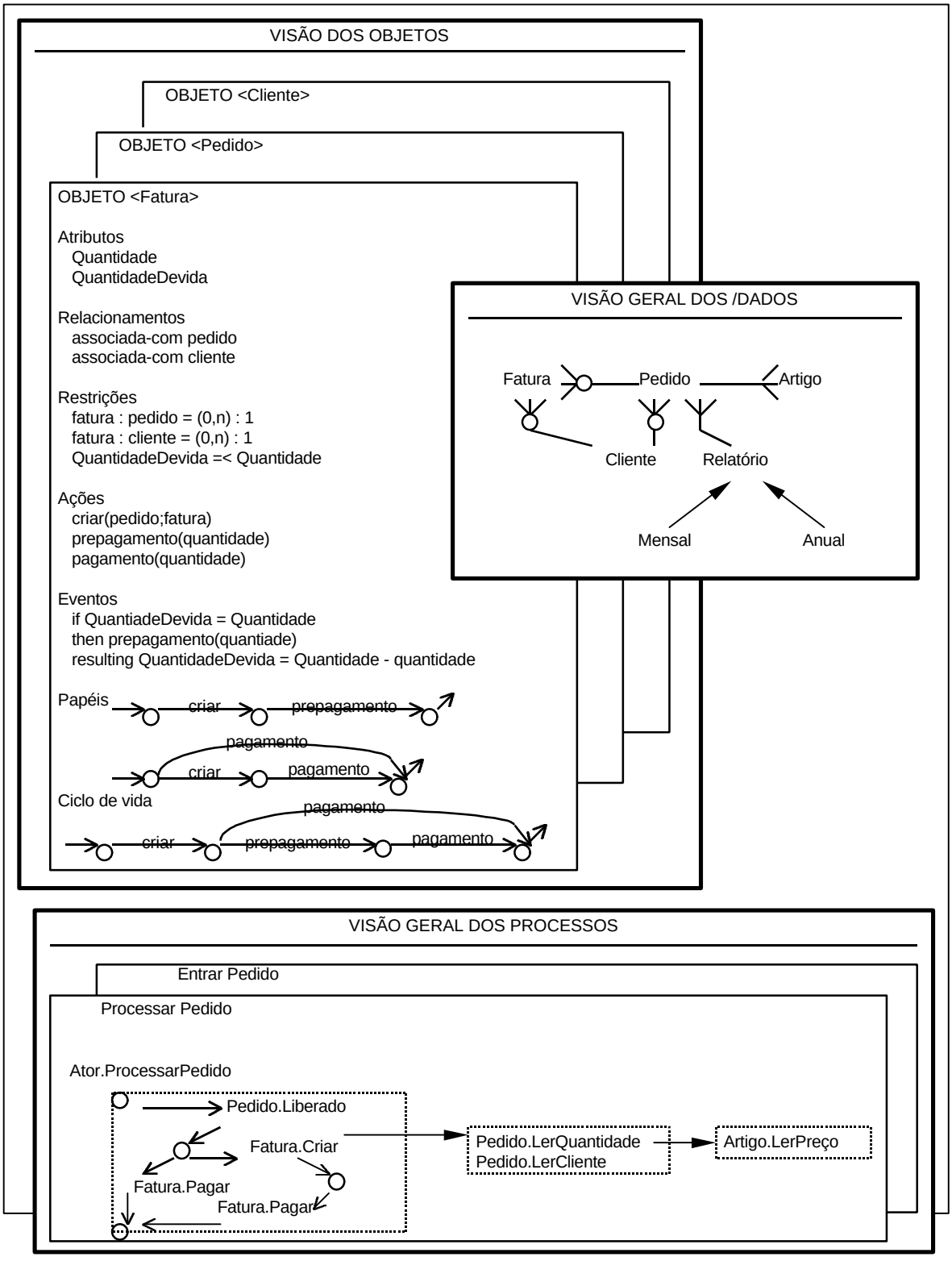


Figura 4.5. Exemplo de uma “visão de objeto” combinada com “visões gerais” de dados e processos [RAM91]

4.4 M.E.R.O.DE.: Um Método de Desenvolvimento Orientado a objetos Dirigido ao Modelo Entidade-Relacionamento

(G. Dedene e M. Snoeck) [DED94]

M.E.R.O.DE. (Model-driven Entity Relationship Object Oriented Development) [DED94] é um método que usa uma abordagem combinada para desenvolvimento de software. Diferentes técnicas são usadas para modelar diferentes aspectos do sistema. Os modelos trabalhados são:

- **Modelo de Escopo.** Modela o contexto e a estratégia adotada para um sistema;
- **Modelo de Serviço.** Modela o funcionamento exato dos serviços, mostrando entidades, restrições e regras;
- **Modelo de Projeto.** Modela as funções de informação para entradas e saídas de informações e
- **Modelo de Tecnologia.** Modela o sistema como é implementado numa tecnologia específica.

Um exemplo de modelo de Escopo seria: “Uma organização de ajuda ao Peru (Help Peru) deseja um sistema de informação para administração de doadores e comprovantes de doações (certificados). A contabilidade e a administração dos projetos em desenvolvimento serão feitas manualmente”.

O modelo de Serviços usa diagrama Entidade-Relacionamento e diagrama de Dependência de Existência (ED) para descrever os aspectos estáticos dos objetos. Os aspectos dinâmicos são descritos pela tabela Evento-Objeto (OET), identificando eventos relevantes classificados em criação, modificação e eliminação, e pelo diagrama de estrutura (JSD), seguindo os princípios de Jackson, com iterações (j) e alternativas (#). O modelo de Serviços também inclui uma definição de tipos de dados abstratos para cada tipo de objeto, contendo o vetor-estado e um método para cada evento em que esse tipo de objeto participa. A figura 4.6 mostra um exemplo do modelo de Serviços.

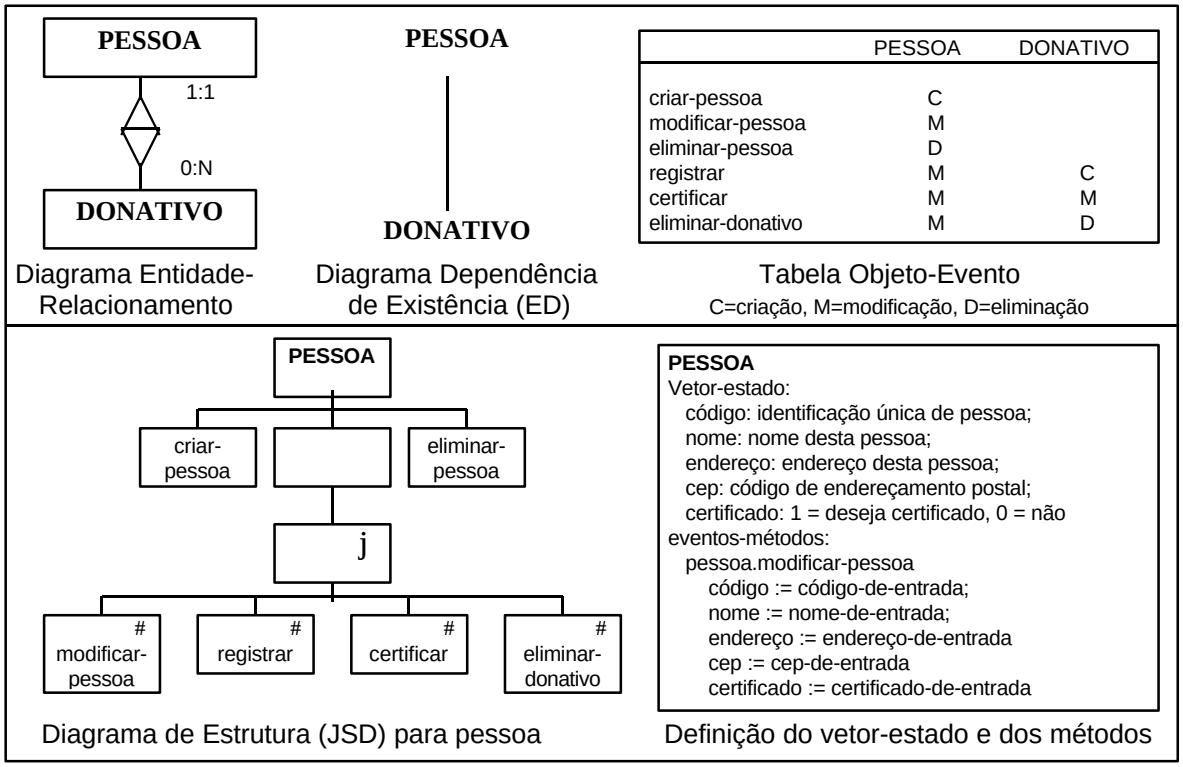


Figura 4.6. Exemplo de modelo de Serviços [DED94]

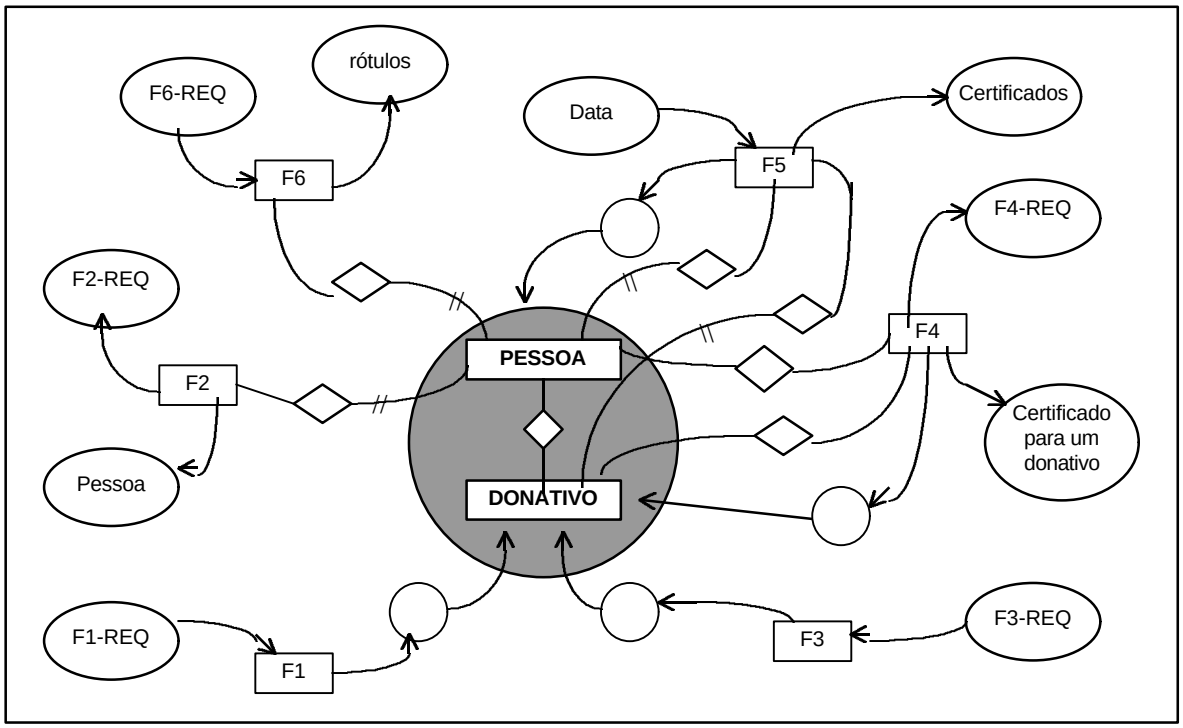


Figura 4.7. Exemplo de modelo de Projeto [DED94], com as seguintes funções:
F1 = Registrar nova pessoa; F2 = Buscar uma pessoa, dados nome ou endereço; F3 = Registrar um donativo; F4 = Criar um certificado para um donativo; F5 = Gerar certificados anualmente e F6 = Imprimir rótulos com endereço para cada pessoa.

O modelo de Projeto, também chamado modelo Funcional, define dois tipos de funções: (a) funções de informações com saídas para o usuário final e (b) funções de interação, que geram eventos transmitidos ao modelo de Serviços. Estas funções têm acesso ao vetor-estado dos

objetos e são identificadas, no modelo, por um losango. O acréscimo de um duplo-traço indica que várias instâncias são verificadas. Os eventos são marcados por círculos. Esta notação é mostrada na figura 4.7, com um exemplo do modelo de Projeto.

O modelo de Tecnologia pode utilizar, por exemplo, diagrama de Fluxo de Dados. A figura 4.8 mostra um exemplo do modelo de Tecnologia.

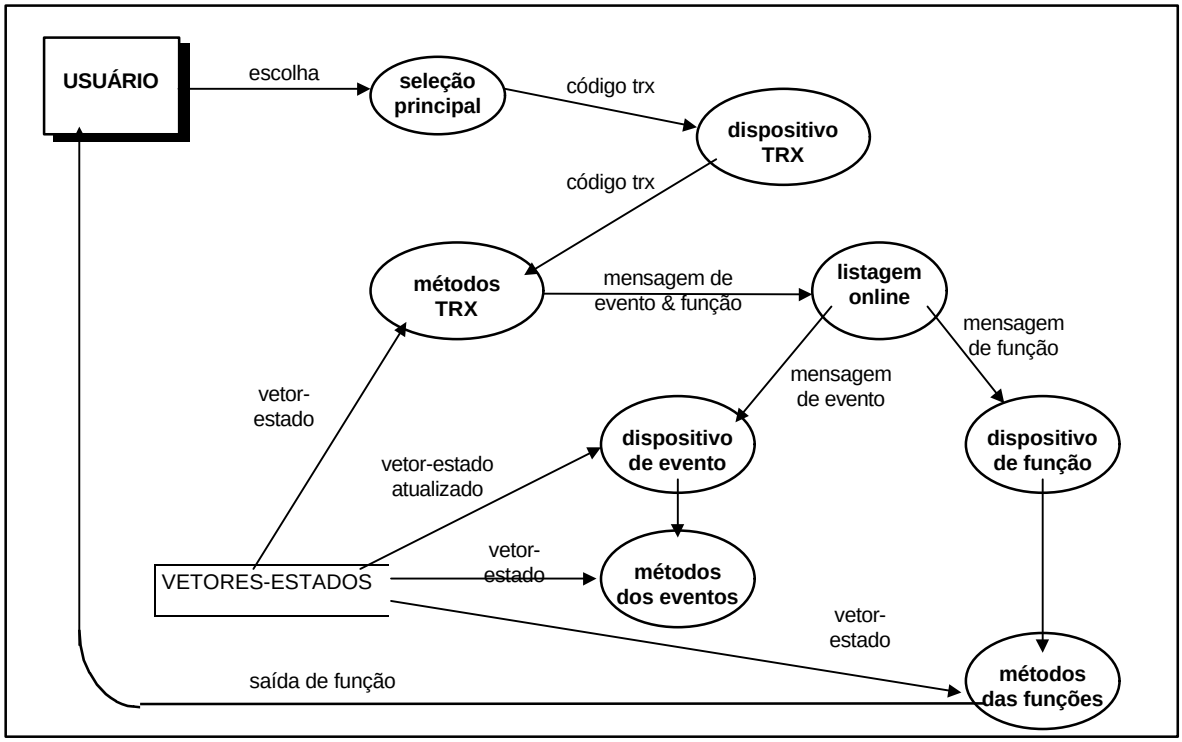


Figura 4.8. Exemplo de modelo de Tecnologia [DED94]

O esquema E-R define tipos de entidades e tipos de relacionamentos. Os autores consideram ambos como objetos tipo. A semântica do diagrama de dependência de existência (ED) é um subconjunto da semântica do modelo E-R. A relação de dependência de existência (ED) define a ordem parcial entre os objetos tipo. Esta ordem pode ser derivada do esquema E-R através de regras como: “Cada tipo de relacionamento R é dependente em existência dos tipos de entidades participantes”.

A semântica de cada diagrama ED também é um subconjunto dos esquemas dinâmicos. Isto significa que para cada objeto do diagrama ED há uma coluna da tabela OET e um diagrama JSD (e vice-versa).

O universo dos tipos de eventos em um modelo é denotado por A. Em função da tabela OET, um

subconjunto de A é associado a cada tipo de objeto. Este subconjunto é chamado de “alphabet” de um tipo de objeto P e é denotado por S_AP, que é particionado em três conjuntos mutuamente disjuntos:

$c(P) = \{ a \in A \mid$ a cria uma ocorrência do tipo P $\} \subseteq S_AP$

$m(P) = \{ a \in A \mid$ a modifica uma ocorrência do tipo P $\} \subseteq S_AP$

$d(P) = \{ a \in A \mid$ a elimina uma ocorrência do tipo P $\} \subseteq S_AP$

O diagrama de estrutura pertencente a P deve conter todos os tipos de eventos contidos em S_AP. O diagrama de estrutura deve também respeitar restrições como: “objetos não podem ser modificados antes de terem sido criados nem após terem sido eliminados” e “objetos não podem ser destruídos antes de serem criados”.

Uma relação ED entre objetos tipo adiciona duas restrições ao comportamento de um objeto tipo: (1) se o objeto P é dependente em existência de Q, então P não pode fazer nada sem que Q participe e (2) cada seqüência de eventos que é aceita para P também pode ser aceita para Q.

Segundo os autores, as definições dos diferentes esquemas ficam, através da formalização, fortemente interconectadas, facilitando as verificações de consistência.

4.5 Conclusões

4.5.1 Quanto à Formalização

Uma metodologia é essencialmente um conjunto de notações associadas a uma estratégia e a heurísticas para aplicá-las. As melhores entre as novas metodologias têm estratégias efetivas e contém heurísticas úteis, mas são caracterizadas por enfatizar a naturalidade de expressão e a intuição [CHA91].

Entretanto alguns autores [RAM91] [DED94] [CHA91] defendem que a precisão da análise depende do grau de formalismo utilizado na modelagem. Uma série de trabalhos tratam do formalismo utilizando ontologia, lógica temporal, álgebra, redes de transição e outras abordagens [RAM91]. Um pequeno número, porém, aborda a verificação de consistência entre os modelos construídos [HAY91] [DED94].

Há vários modos das técnicas formais auxiliarem o desenvolvimento orientado a objetos: (a) descrevendo os conceitos básicos numa linguagem matemática formal; (b) suportando os métodos de desenvolvimento de software através de pré e pós-condições e invariantes; (c) suportando a reutilização de componentes de software descrevendo o que fazem sem se referir ao código e (d) possibilitando a verificação formal (aqui se inclui, principalmente, a validação de código). Acredita-se que passará longo tempo antes da verificação formal tornar-se rotineira [CHA91].

Quando notações formais forem incorporadas às metodologias orientadas a objetos, espera-se, entre outros benefícios, que serão produzidos modelos coerentes capazes de serem validados em relação aos requisitos [CHA91].

4.5.2 Quanto às Propostas Apresentadas

Hayes e Coleman [HAY91], no trabalho apresentado na seção 4.2, alteram os modelos da metodologia OMT e incluem novos modelos com a intenção de formalização, enfatizando pré e pós-condições no rastreamento de sequências de eventos. A proposta desta dissertação não altera a metodologia OMT no tocante à atividade diagramática do modelador. O modelo aqui proposto é gerado automaticamente a medida que esta atividade se desenvolve e o mecanismo integrador baseia-se nas interações detectadas entre elementos modelados em diagramas diferentes.

A proposta de Hayes e Coleman [HAY91] fundamenta-se na consideração de que pré e pós-condições nos modelos Funcional e Dinâmico sejam equivalentes, o que é difícil de garantir. De qualquer forma, assumida a equivalência, os autores sugerem a verificação de consistência através da seqüência de eventos (no modelo Dinâmico) quando comparada com as pré e pós-condições definidas para o modelo Funcional. Esta comparação pode ser conseguida na proposta desta dissertação através da leitura seqüencial da matriz Square, envolvendo o comportamento do sistema em termos de eventos e o reconhecimento, nesta leitura, das operações e dos processos envolvidos e, além disto, das classes participantes.

Ramackers e Verrijn-Stuart [RAM91], no trabalho apresentado na seção 4.3, consideram três perspectivas utilizadas para modelar um sistema de informação: perspectiva dos dados, perspectiva do comportamento e perspectiva dos processos. De forma semelhante, Rumbaugh et al [RUM91] consideram a perspectiva dos aspectos estáticos e estruturais dos dados (no Modelo de Objetos), a perspectiva comportamental (no Modelo Dinâmico) e a perspectiva das

transformações das funções do sistema (no Modelo Funcional).

Na proposta de Ramackers e Verrijn-Stuart [RAM91], as perspectivas são tratadas como visões num contexto integrado, com duas alternativas: a primeira, onde uma visão pode corresponder a uma das três perspectivas de forma complementar e conjugada, e a segunda, onde pode corresponder a uma combinação dos aspectos de dados, comportamento ou processos. Na proposta desta dissertação, as perspectivas da OMT podem ser visualizadas num modelo único, que sofre um processo de abstração onde há múltiplas alternativas em função do filtro escolhido pelo modelador. Além disto, são previstas regras de consistência para verificação geral do modelo.

Dedene e Snoeck [DED94], no trabalho apresentado na seção 4.4, descrevem a metodologia chamada M.E.R.O.DE que não tem muito em comum com a OMT. De qualquer forma, fica evidenciada a necessidade de integração dos modelos gerados e os autores desta proposta tentam a integração pela verificação de consistência, viabilizada pela formalização. São apresentados diversos pontos de verificação mas os autores não informam exatamente qual o mecanismo que executa esta consistência ou de que forma ela é realizada. Na proposta desta dissertação, são utilizados como pontos de verificação as conexões explícitas que serão detalhadas no próximo capítulo e estas conexões, conforme será visto, são verificadas visualmente através do modelo matricial ou de forma automatizada através de regras de consistência.

No próximo capítulo são apresentadas considerações sobre o mecanismo de integração proposto, baseado em operações, e são descritos fundamentos para a interação entre os modelos da OMT, concretizada através de conexões explícitas entre eles.

5 INTEGRAÇÃO DOS MODELOS DA OMT

> "*Os diferentes modelos não são completamente independentes ... mas cada modelo pode ser examinado e entendido isoladamente num contexto amplo. As interconexões entre os diferentes modelos são limitadas e explícitas.*"
> Rumbaugh et al [RUM91]

5.1 Introdução

Neste capítulo é discutida a integração dos modelos da OMT. A seção 5.2 identifica as atividades consideradas neste trabalho, como contidas na fase de análise, e define a importância de um dicionário de dados e do relacionamento entre operações e processos e entre operações e eventos para a solução proposta nesta dissertação para a integração. A seção 5.3 apresenta os postulados que norteiam esta proposta, baseados na descrição da metodologia feita pelos seus autores, identifica as conexões explícitas para a integração dos modelos da OMT e apresenta as premissas do modelo matricial aqui proposto.

5.2 Considerações para Integração

De acordo com os autores da metodologia OMT, os três modelos foram idealizados para darem perspectivas independentes sobre o sistema. Esta independência, é claro, tem limites. Eles mesmos definem pontos comuns de conexão e é através destes que se viabiliza a verificação de consistência entre os modelos criados.

Além dos pontos de conexão entre os modelos que a metodologia apresenta na fase de análise, alguns outros são previstos na fase denominada "projeto de objetos" na OMT. Estes são considerados, neste trabalho, como inclusos na fase de análise para melhorar a verificação de consistência. O conjunto assim formado por estes vínculos permite a integração dos modelos. Das etapas propostas para a OMT, as seguintes fazem parte da fase de análise no escopo desta dissertação:

- Construir o modelo de Objetos: identificar classes; iniciar a geração de um dicionário de dados; acrescentar associações entre classes; acrescentar atributos para

objetos e ligações; organizar e simplificar as classes através de herança. Resultado: modelo de Objetos = diagrama de Objetos + dicionário de dados.

- Construir o modelo Dinâmico: identificar eventos; preparar diagramas de Fluxo de Eventos; desenvolver um diagrama de Estados para cada classe com comportamento dinâmico importante. Resultado: modelo Dinâmico = diagramas de Fluxo de Eventos + diagramas de Estados.
- Construir o modelo Funcional: identificar valores de entrada e saída; utilizar diagramas de Fluxo de Dados para mostrar dependências funcionais. Resultado: modelo Funcional = diagramas de Fluxo de Dados.
- Incluir etapas do projeto de objetos: definir uma operação para cada processo-folha do DFD; definir uma operação-causa e uma operação-efeito para cada evento do diagrama de Fluxo de Eventos; acrescentar ao modelo de Objetos as operações descobertas durante o desenvolvimento dos outros modelos; ajustar a estrutura de classes para aperfeiçoar a herança; verificar, repetir e refinar os três modelos. Resultado: modelo de Objetos detalhado + modelo Dinâmico detalhado + modelo Funcional detalhado.

Para que a proposta de integração aqui apresentada seja efetiva, devem ser reforçados dois pontos básicos na atividade de modelagem da OMT: (1) um dicionário de dados eficiente e o mais completo possível e (2) a enfatização do mapeamento entre processos e operações e entre eventos e operações.

5.2.1 Dicionário de Dados

Um dicionário de dados deve ser construído para completar o modelo de Objetos, complementando o diagrama de Objetos [RUM91]. É fundamental, para a verificação de consistência entre os modelos, que seja construído um dicionário de dados não só para as classes de objetos mas também para seus atributos, para os argumentos de eventos e de operações e para os dados dos fluxos de dados. Esta dissertação, entretanto, não entrará em detalhes sobre a construção de um dicionário de dados que apoie a metodologia.

Um dicionário de dados pode viabilizar a comparação, por exemplo, entre os argumentos de um evento, transmitindo informação entre objetos, e os dados de um fluxo de dados, transmitindo

informação entre processos. Este tipo de comparação detecta ou não a coerência, no caso do exemplo, dos dados que fluem no modelo funcional com aqueles que são transmitidos no modelo dinâmico, confrontando-os com os atributos das classes de objetos envolvidas.

5.2.2 Operações e Processos-folha

Rumbaugh et al [RUM91] declaram que cada processo de nível mais baixo, atômico, é uma operação. O ideal, em termos de consistência, seria que a declaração acima fosse seguida com rigor. Entretanto os próprios autores da OMT sugerem que algumas vezes este relacionamento um-para-um não é verdadeiro. Por exemplo, um modelador pode achar desnecessário o detalhamento de um processo como “Gerar Nota Fiscal”, ainda que inclua operações como “ler_nome_produto” e “calcular_total”.

De qualquer modo, para viabilizar a integração de modelos como é aqui pretendida, é necessária a definição do vínculo dos processos-folha estabelecidos no DFD com operações das classes de objetos. Mesmo que cada processo-folha não corresponda a uma única operação.

5.2.3 Operações e Eventos

No modelo Dinâmico interessa a seqüência de operações determinadas pelos estímulos individuais (eventos) de um objeto para outro [RUM91]. Esta seqüência de operações inevitavelmente acarreta um outro tipo de seqüência: a dos eventos. Em outras palavras, pode-se dizer que uma seqüência de eventos é determinada pelas operações envolvidas.

Neste trabalho, ao serem construídos os diagramas de Fluxo de Eventos, no modelo Dinâmico, a tarefa de definição destes eventos deve ser completada informando-se as operações envolvidas. Associadas a cada evento devem ser informadas uma operação-causa, de responsabilidade da classe de objetos de onde se origina o evento, e uma operação-efeito, de responsabilidade da classe para onde se destina o evento. A primeira ativa o evento e a segunda é ativada por ele.

Algumas classes de objetos representam a interferência do usuário ou, de forma mais geral, do ambiente externo ao sistema. As operações destas classes de objetos não são especificadas. Estas classes de objetos aparecem, entretanto, no diagrama de Fluxo de Eventos (assim como são

representadas como atores nos DFDs). Como esses objetos ficam além da fronteira que limita o sistema, quando um deles constitui a fonte do evento, não interessa a operação-causa. E no caso de ser o destino do evento, não interessa a operação-efeito. Nestes casos o evento será referenciado pela classe que o envia e a operação-efeito ou pela operação-causa e pela classe que o recebe.

5.3 Integração

Recordando o que foi apresentado no capítulo sobre o estado da arte, verifica-se o seguinte: Hayes e Coleman [HAY91], refazendo a OMT com modelos coerentes, usam pré e pós-condições e eventos (usando a definição original da OMT) como elementos integradores; Ramackers e Verrijn-Stuart [RAM91], modelando sistemas integrados e formais, usam eventos (definidos como a execução de uma ação) para integrar modelos; Dedene e Snoeck [DED94], em MERODE, usam (principalmente) eventos, sem distinguí-los claramente de métodos (ou de operações no caso da OMT), para a verificação de consistência entre os modelos.

A integração pretendida aqui utiliza as operações como elemento comum. Rumbaugh et al [RUM91] comentam a inclusão das operações na modelagem. Mesmo despreocupando-se inicialmente com a definição das operações, mostram a validade da escolha delas como elemento integrador:

> "*As operações na programação orientada a objetos podem corresponder a consultas sobre atributos ou associações no modelo de objetos ..., a eventos no modelo dinâmico ... e a funções no modelo funcional ... Consideramos mais útil distinguir estes vários tipos de operações durante a análise.*" [RUM91]

Pelo enfoque de uma operação, portanto, pode-se fazer referência a todas as dimensões modeladas. No modelo de Objetos as operações estão presentes na descrição das classes de objetos. Uma operação é de responsabilidade de uma classe de objetos, acessando ou alterando seus atributos e suas ligações com outros objetos. No modelo Dinâmico as operações podem causar transformações de estados e estão presentes como ações ou atividades no ciclo de vida dos objetos. Estão intimamente ligadas aos eventos ativando-os ou sendo ativadas por eles. No modelo Funcional as operações estão presentes nos processos, sendo implementadas através de métodos, como funções do sistema.

A proposta de integração através da Matriz Square assume que existem conexões explícitas entre os três modelos da OMT. Estas conexões, isto é, os relacionamentos entre seus elementos, são definidas pelos postulados apresentados a seguir que, por sua vez, fundamentam-se no que é ditado pelos autores da OMT [RUM91], ao estabelecer e definir os elementos a modelar, ao relacionar e combinar os modelos entre si e ao sugerir verificações de consistência entre eles.

5.3.1 Postulados para Integração

O modelo de Objetos influencia e restringe a estrutura do modelo Dinâmico, que especifica seqüências possíveis de modificações nos objetos do modelo de Objetos. [RUM91, p.110].

Os estados podem ter subestados que herdam as transições de seus superestados, assim como as classes têm subclasses que herdam os seus atributos [RUM91, p.97]. O modelo Dinâmico descreve um conjunto de objetos concorrentes, cada um com seu diagrama de Estados próprio. Agregação também implica em concorrência. A concorrência no interior de um objeto ocorre quando ele pode ser particionado em subconjuntos de atributos ou ligações. O estado do objeto compreende um estado de cada componente [RUM91, p.99].

Algumas operações, para obter suas informações, devem percorrer os caminhos de uma rede de associações, isto é, conjuntos de ligações entre os objetos envolvidos [RUM91, p.236]. As associações são a “cola” do modelo de Objetos, providenciando caminhos de acesso entre os objetos [RUM91, p.245].

- **Postulado 1**. A estrutura do modelo Dinâmico é influenciada e restringida pela estrutura do modelo de Objetos.
  - **a)** As hierarquias de classes definem generalizações de estados e as agregações definem concorrência entre os estados dos objetos.
  - **b)** Um evento de um objeto para outro implica na existência de uma ligação (instância de associação ou agregação), entre os objetos envolvidos, providenciando um caminho de acesso entre eles, por onde podem ser rastreados os efeitos das operações e a seqüência dos eventos.

O modelo Dinâmico descreve aspectos relacionados à seqüência das operações, incluindo os eventos e a seqüência em que eles ocorrem [RUM91, p.18]. Uma descrição comportamental de um objeto deve especificar o que o objeto faz em resposta aos eventos. As operações vinculadas a estados ou transições são executadas em resposta aos correspondentes estados ou eventos [RUM91, p.92].

Um objeto pode executar a ação de enviar um evento para outro objeto e, assim, um sistema de objetos interage pelo intercâmbio de eventos [RUM91, p.103]. Cada evento enviado a um objeto corresponde a uma operação no objeto [RUM91, p.184].

- **Postulado 2**. O modelo Dinâmico mostra, quanto ao comportamento individual dos objetos, os eventos que uma classe de Objetos recebe e em que ordem as operações, definidas para essa classe, alteram seus estados.

- **Postulado 3**. O modelo Dinâmico, quanto ao comportamento global do sistema, mostra a seqüência em que as operações são executadas.
  a) Um evento ativa ou é ativado por operações definidas para as classes de objetos envolvidos.

Um evento é algo que acontece num determinado momento. Alguns eventos podem ser simples sinais de que alguma coisa ocorreu, enquanto outros transportam valores de dados ao ativar operações [RUM91, p.85].

- **Postulado 4**. Um evento, quando possui atributos, transporta valores de dados de um objeto para outro, e estes servem de argumentos para as operações a serem executadas.

No modelo Funcional, os processos-folha são operações sobre os objetos. Enquanto o modelo de Objetos mostra “quem” faz, o modelo Funcional mostra “o que” o sistema deve fazer. Nem sempre há correspondência direta entre processos e operações [RUM91, p.137]. Os processos do modelo Funcional devem em algum momento ser implementados como operações sobre objetos. Cada processo do nível mais baixo é uma operação [RUM91, p.130]. Na prática, entretanto, alguns processos implementam diversas operações enquanto, por outro lado, algumas operações necessitam de diversos processos para serem implementadas [RUM91, p.137].

- **Postulado 5**. Processos correspondem a operações de objetos.
  a) Por vezes um processo corresponde a diversas operações e, em outros casos, uma operação corresponde a diversos processos.
  b) Cada processo atômico de nível mais baixo é uma operação.

As operações não-triviais podem ser divididas em três categorias: consultas, ações e atividades. Uma consulta é uma operação sem efeitos colaterais [RUM91, p.131]. Ao contrário das ações e das atividades, as consultas não aparecem explicitadas nos diagramas; entretanto, algumas operações modeladas são consultas. Identificadas no diagrama de Estados, ações são operações instantâneas associadas a eventos e atividades são operações que consomem tempo para se completar associadas a estados [RUM91, p.92,184]

- **Postulado 6**. Uma operação especificada para uma classe de objetos pode ser uma consulta, uma atividade ou uma ação.
  a) Consultas, atividades e ações são implementadas em processos e tem suas seqüências definidas por eventos.
  b) Uma atividade ocorre associada a um estado e uma ação é executada em decorrência de uma transição de estado.

Os processos do modelo Funcional mostram objetos que estão relacionados por função [RUM91, p.137]. Os atores são vinculados às entradas e saídas de um diagrama de Fluxo de Dados, agindo de forma ativa ao produzir e consumir valores [RUM91, p.126]. Diferentemente de um ator, um depósito de dados não gera por si mesmo qualquer operação e apenas atende a solicitações de armazenamento e de acesso aos seus dados [RUM91, p.127].

- **Postulado 7**. Um processo envolve objetos que estão relacionados por função.
  a) Classes de objetos participam de processos de forma ativa ou passiva, estimulando, executando ou sofrendo a ação das operações ali implementadas.
- **Postulado 8**. Um ator é um objeto ativo gerando e consumindo valores de dados nos extremos do DFD.
- **Postulado 9**. Um depósito de dados é um objeto passivo, atendendo solicitações para acessar ou armazenar dados.

Um evento é uma informação unidirecional de um objeto para outro [RUM91, p.85]. Os fluxos de dados do modelo Funcional, que também transmitem informação, são valores no modelo de Objetos. Alguns fluxos de dados são valores puros, como números, strings ou listas de valores, outros representam objetos [RUM91, p.138]. No modelo Funcional, um fluxo de dados interliga a saída de um objeto ou processo à saída de outro objeto ou processo, representando informações sendo transmitidas [RUM91, p.126].

- **Postulado 10**. Um fluxo de dados transporta dados passando-os como argumentos de operações, assim como um evento, quando tratado como uma transmissão unidirecional de informação.
  - **a)** Alguns dados dos fluxos de dados correspondem a atributos dos objetos e outros são valores intermediários de uma computação.

Alguns fluxos de dados representam argumentos para operações, outros têm seus dados como "alvo" para as operações. Muitas vezes um fluxo de entrada de um processo é um objeto chamado de "alvo" (aquele que executa ou solicita a execução das operações implementadas no processo) enquanto os outros fluxos são apenas argumentos. Se o fluxo de saída de um processo aponta para um depósito de dados, então este representa o "alvo". Se os fluxos de entrada ou de saída originam-se ou dirigem-se a um ator, então este é o "alvo". Se o fluxo de entrada de um processo é um fluxo de objeto e o fluxo de saída contiver atributos dele ou de um objeto que tem ligações com ele, então aquele objeto é o "alvo" [RUM91, p.138].

Cada fluxo de dados que se dirige a um depósito de dados é uma operação de atualização. Cada fluxo de dados que parte de um depósito de dados é uma operação de consulta, sem efeitos colaterais sobre ele. Os fluxos de dados de ou para atores representam operações sobre estes fluxos. Os valores destes fluxos de dados são os argumentos ou os resultados das operações [RUM91, p.138].

- **Postulado 11**. Um fluxo de dados corresponde a uma ou mais operações.
  - **a)** Um fluxo de dados que chega a um depósito de dados corresponde a uma operação de atualização.
  - **b)** Um fluxo de dados que sai de um depósito de dados corresponde a uma operação de consulta.
  - **c)** Um fluxo de dados que chega a um ator corresponde a operações cujos resultados

são os dados componentes do fluxo.

d) Um fluxo de dados que sai de um ator corresponde a operações cujos argumentos são os dados componentes do fluxo.

e) Um fluxo de objeto corresponde a uma operação de criação de um objeto, sendo portanto uma operação de classe.

Um diagrama de Fluxo de Dados mostra todos os caminhos possíveis de computação de valores, sem identificar quais são ou não executados e nem em que ordem. No modelo Dinâmico modelam-se as decisões e as seqüências. Uma decisão resolve se uma ou mais funções serão executadas. Às vezes é útil incluir estes controles no modelo Funcional para que não sejam esquecidos e para que as dependências de seus dados sejam mostradas. Isto é feito através dos fluxos de controle, que é um valor booleano que afeta o modo como um processo é avaliado [RUM91, p.129]. Uma condição é uma função booleana de valores de objetos, sendo válida num intervalo de tempo. As condição controlam as transições no diagrama de Estados, disparando eventos correspondentes [RUM91, p.91]. A seqüência dos eventos determinam a seqüência das operações que ativam [RUM91, p.18] que, por sua vez, são implementadas pelos processos [RUM91, p.130,137].

- **Postulado 12**. Decisões e seqüências são questões de controle que fazem parte do modelo Dinâmico, mas podem ser incluídas no modelo Funcional através de fluxos de controle.

  a) Uma condição controla operações nas transições de estado assim como um fluxo de controle o faz com processos e, portanto, com as operações neles implementadas.

  b) Um evento determina a seqüência das operações assim como um fluxo de controle o faz com processos e, portanto, com as operações neles implementadas.

5.3.2 Vínculos entre os Elementos Modelados

Os postulados apresentados descrevem interações entre os elementos modelados. Para a integração proposta, feita pela existência de conexões explícitas determinadas por estas interações, é importante que se considere os vértices e os arcos, representados tradicionalmente nos diagramas da OMT [RUM91], e que se acrescente, tomando como base os postulados, um vértice e novos arcos, principalmente ligando diagramas diferentes. Os elementos acrescidos não interferem visualmente nos diagramas tradicionais, mas são considerados pela Matriz Square como componentes do que seria um diagrama OMT único. Arcos e vértices são representados na matriz como elementos-arco e elementos-vértice respectivamente.

Os diagramas utilizados pela OMT modelam como vértices:

- **classes**, no diagrama de Objetos e no diagrama de Fluxo de Eventos, sendo que pela Matriz Square vértices num e noutro diagrama referentes à mesma classe são assumidos como elemento-vértice único;
- **atores** ou **depósitos de dados**, no diagrama de Fluxo de Dados, sendo que pela Matriz Square um vértice ator (ou depósito de dados) será assumido como o mesmo elemento-vértice da correspondente classe;
- **estados**, no diagrama de Estados e
- **processos**, no diagrama de Fluxo de Dados.

E como arcos:

- relacionamentos (**generalizações**, **associações** ou **agregações**), no diagrama de Objetos;
- **eventos**, no diagrama de Eventos, sendo que pela Matriz Square não são considerados como elementos-arco entre classes, mas entre suas operações-causa e efeito ou, na ausência de uma delas, entre a classe que envia o evento e a operação-efeito ou entre a operação-causa e a classe que recebe o evento;
- **transições** de estado, nos diagramas de Estados, sendo que pela Matriz Square são substituídos pelos seus componentes então tratados como elementos-arco conforme

será detalhado a seguir, no caso de condições e ações e conforme foi visto, no caso de eventos;

- **fluxos de dados**, **fluxos de objeto** ou **fluxos de controle**, no diagrama de Fluxo de Dados.

Para viabilizar, então, de forma mais consistente a integração que é proposta nesta dissertação, a Matriz Square considera também como elemento-vértice:

- **operações**, já que são assumidas como elemento integrador. Como operação, entenda-se consulta, ação ou atividade. As duas últimas são modeladas nos diagramas de Estados: ações em arcos de transições e atividades em vértices junto a estados. As consultas ficam inclusas como operações em vértices junto a classes no diagrama de Objetos (não são identificadas explicitamente na modelagem pelos autores da OMT [RUM91]). Entenda-se também como operações, naturalmente, as operações-causa e efeito de um evento.

E como elementos-arco:

- **condições**, entre estados (quando há transição apenas por condição) ou entre operações-efeito e estados (quanto há evento na transição) e entre operações-efeito e ações (quando há evento na transição e ação associada à transição);
- **vínculos de responsabilidade**, entre as classes e as suas operações;
- **vínculos de evento**, entre as classes que enviam e recebem eventos (a inclusão destes vínculos é justificada porque a modelagem dos eventos, assim como é proposta, define operações-causa e operações-efeito que, como operações, são consideradas elementos-vértice, e porque os eventos são tratados como elementos-arco entre classes, conforme foi visto);
- **vínculos de implementação**, entre as operações e os processos que as implementam.
- **vínculos de interação**, entre as operações (de classes diferentes) implementadas num mesmo processo ou implementadas em processos diferentes mas com fluxo de dados ou de controle entre si.
- **vínculos de estado**, entre as classes e os seus estados;
- **vínculos de atividade**, entre os estados e as operações consideradas como suas

atividades;

- **vínculos de subestado**, entre os superestados e os seus subestados;
- **vínculos de concorrência**, entre os estados concorrentes num único objeto e
- **vínculos de subprocesso**, entre processos e subprocessos.

Os elementos-vértice (classes, estados, operações e processos) são representados na matriz na diagonal principal e os elementos-arco (relacionamentos, eventos, fluxos de dados, fluxos de objeto, fluxos de controle, condições e vínculos de responsabilidade, de evento, de interação, de estado, de atividade, de subestado, de concorrência e de implementação) são representados fora da diagonal principal entre dois elementos-vértice, conforme a figura 5.1. Nela são vistos dois elementos-vértice na diagonal principal e dois elementos-arco definindo a interação entre os elementos-vértice.

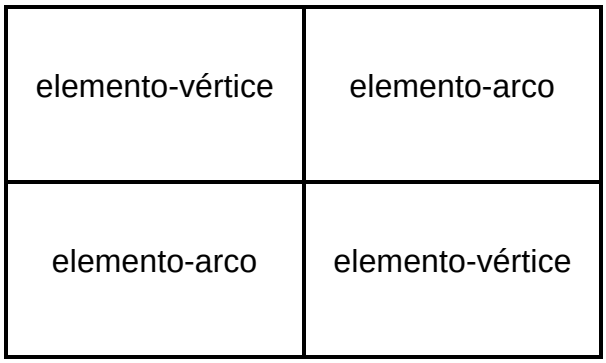


Figura 5.1. Esquema de Matriz Square com elementos-vértice e elementos-arco

5.3.3 Conexões Explícitas

Os postulados apresentados evidenciam a importância do papel das operações como agente integrador. Eles se refletem no metamodelo da figura 5.2. Nele foram considerados apenas os relacionamentos não confinados a cada modelo. Esta restrição é feita para enfatizar as conexões explícitas de integração entre os modelos.

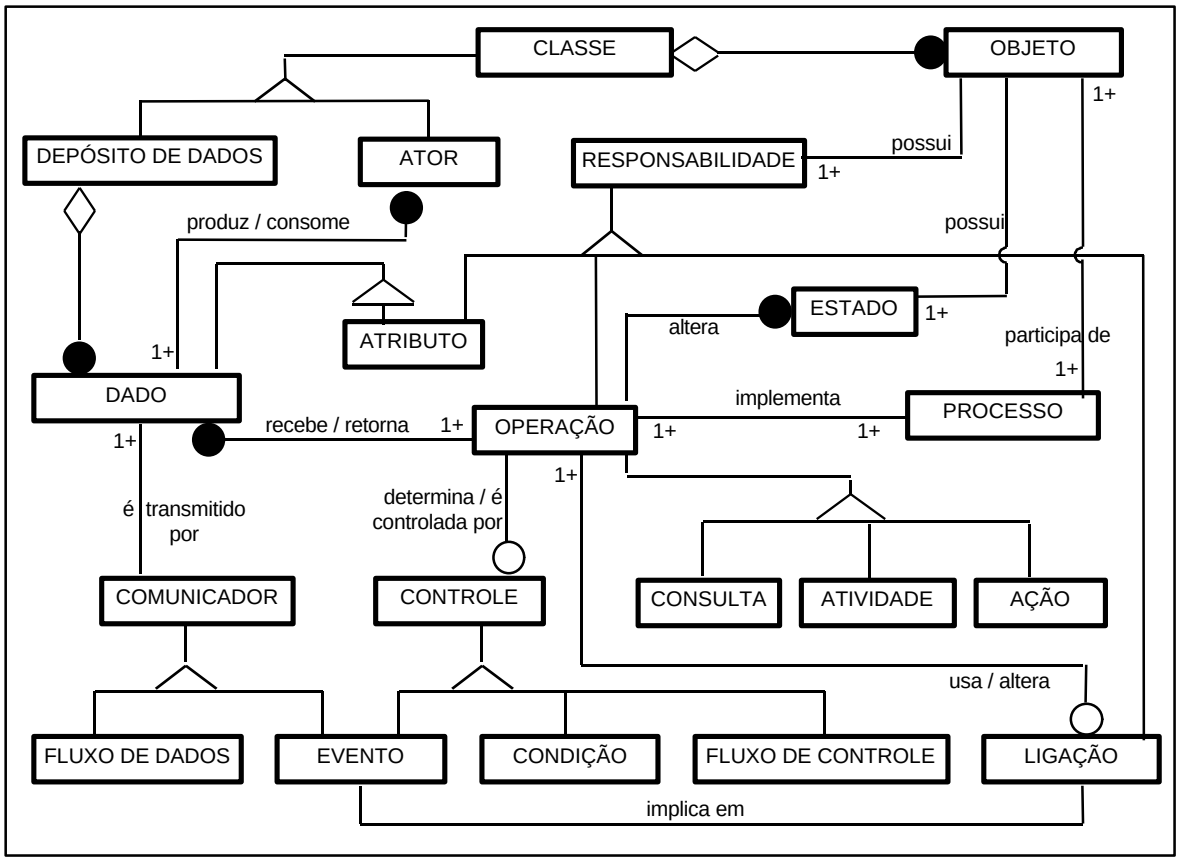


Figura 5.2. Metamodelo que integra os conceitos utilizados nos modelos OMT

Da análise da figura 5.2 pode-se verificar que uma operação estabelece sete classes de conexões explícitas diferentes que definem sua participação como agente integrador. Uma operação (1) é de responsabilidade de um objeto, (2) usa ou altera ligações entre objetos, (3) altera estados de objetos, (4) recebe ou retorna dados, (5) determina ou é controlada por eventos, (6) determina ou é controlada por condições ou fluxos de controle e (7) é implementada por processos. Estas sete classes de conexões explícitas, com participação efetiva das operações, são:

- **Conexão de Responsabilidade**. Esta conexão mostra que uma operação é uma responsabilidade dos objetos de uma classe. Ela é modelada na Matriz Square por vínculos de responsabilidade entre classes e operações. Uma classe estará vinculada diretamente também aos eventos que envia a outras classes pois estes estarão na mesma coluna do vínculo de responsabilidade e da operação correspondentes na matriz.
- **Conexão de Ligação**. Uma operação, sendo consulta, atividade ou ação, como uma responsabilidade de uma classe de objetos, colabora para que objetos de outras classes cumpram, também, suas responsabilidades. Para isto, usa e/ou altera ligações entre eles. Esta conexão mostra como os objetos interagem e esta interação, que é modelada pelas

associações e agregações entre as classes, é confirmada pelos vínculos de eventos e pelos vínculos de interação. Vínculos de evento representam as colaborações de uma classe em relação a outras, decorrentes dos eventos que recebe. Vínculos de interação representam indiretamente colaborações já que mostram interações entre operações de classes diferentes. Vínculos de eventos entre classes e de interação entre operações, além de confirmar associações e agregações, ratificam eventos entre as operações daquelas classes e, portanto, integram associações e agregações com eventos e, também, com os processos que implementam aquelas operações.

- **Conexão de Estado**. Esta conexão mostra que uma operação altera um estado de um objeto ou está associada, como uma atividade, a um estado. São determinados, assim, vínculos de estado e vínculos de atividade. Vínculos de estado são modelados quando, ao ser criado um diagrama de Estados, descrevem-se os estados válidos para uma classe de objetos. Estes vínculos determinam, em última instância, quais operações (justamente aquelas dos objetos da classe em questão) podem afetar, em razão de eventos e condições, as transições de estados modeladas. Vínculos de atividade são modelados quando associam-se operações a estados determinando que estas são atividades executadas pelos objetos naqueles estados.
- **Conexão de Comunicação**. Uma operação pode receber ou retornar dados de/para outras operações ou de/para atores ou, ainda, acessar ou alterar depósitos de dados. Uma transmissão de informação é modelada por esta conexão que é determinada através de elementos comunicadores como eventos (quando enviam dados de um objeto a outro) e fluxos de dados (ou fluxos de objeto). Pode ser detectado onde um dado é encontrado na comunicação entre objetos e como aparece em argumentos de operações ou atributos de eventos e como componente de fluxos de dados.
- **Conexão de Controle por Evento**. Esta conexão mostra que uma operação pode ativar ou ser ativada por um evento. Este tipo de controle ocorre como um estímulo entre objetos. Um evento é, portanto, um elemento de controle das operações determinando a seqüência em que elas ocorrem, assim como um fluxo de controle o faz com os processos que implementam estas operações. Por outro lado, um evento também, quando elemento comunicador, transmite informações necessárias à execução de operações, assim como um fluxo de dados o faz entre os processos que implementam estas operações.
- **Conexão de Controle por Condição**. Esta conexão mostra que uma operação pode determinar ou ser controlada por uma condição. Este tipo de controle não fica restrito

à seqüência das operações entre objetos, mas também à seqüência de operações de um único objeto. Este tipo de controle é importante na modelagem OMT [RUM91] porque, na ausência de eventos, são as condições que estabelecem transições de estados e seqüências entre operações. Uma condição é, portanto, um elemento de controle de operações, assim como um fluxo de controle o faz com os processos que implementam estas operações.

- **Conexão de Processo**. Esta conexão mostra que uma operação é implementada por um ou mais processos. São determinados, assim, vínculos de implementação entre operações e processos. Indiretamente vinculam-se objetos (aqueles responsáveis pelas operações) e processos.

A tabela 5.1 mostra os postulados que fundamentam as sete conexões explícitas vistas anteriormente. Uma oitava conexão (esta não tendo a participação de operações) é a que vincula classes de objetos (dos diagramas de Objeto e de Fluxo de Eventos) com atores e depósitos de dados (do diagrama de Fluxo de Dados). Esta conexão passará a ser chamada de **conexão de classe**. Ela é estabelecida quando uma classe e um ator ou uma classe e um depósito de dados, com mesmo nome, são considerados como elemento-vértice único.

postulados	conexões explícitas						
	de Responsa-bilidade	de Ligação	de Estado	de Comuni-cação	de Controle por Evento	de Controle por Condição	de Processo
1		■	■		■		
2	■				■		
3	■				■		
4				■	■		
5							■
6	■		■		■		■
7	■						■
8	■			■			
9	■			■			
10				■	■		
11	■			■			
12					■	■	■

Tabela 5.1. Conexões explícitas fundamentadas por postulado
(o sombreamento indica o postulado como fundamento para a conexão)

A tabela 5.2 mostra as classe de conexões e os elementos integradores entre os modelos de Objeto (MO), Dinâmico (MD) e Funcional (MF). As tabelas 5.3, 5.4 e 5.5 mostram a integração entre elementos de modelos diferentes, como é pretendida pela Matriz Square, indicando os elementos integradores em cada caso.

	conexões explícitas							
Integração	**de Responsabilidade**	**de Ligação**	**de Estado**	**de Comunicação**	**de Controle por Evento**	**de Controle por Condição**	**de Processo**	**de Classe**
MO x MD	vínculos de responsabilidade	vínculos de evento e interação	vínculos de estado e de atividade	vínculos de responsabilidade	eventos	condições		(mesmo vértice)
MO x MF				fluxos de dados e de objeto			vínculos de implementação	
MD x MF			vínculos de estado e de atividade	vínculos de responsabilidade	eventos	condições		

Tabela 5.2. Conexões explícitas e seus elementos integradores entre modelos

modelo de Objetos	**modelo Dinâmico**					
	classe	**estado**	**atividade**	**ação**	**evento**	**condição**
classe	nome (mesmo vértice)	vínculo de estado	vínculos de estado e responsabilidade	vínculo de responsabilidade	vínculo de responsabilidade	
operação	vínculo de responsabilidade	condição, evento e vínculo de atividade			evento	condição
generalização						
associação					vínculos de even-to, interação e responsabilidade	
agregação					vínculos de even-to, interação e responsabilidade	

Tabela 5.3. Integração entre os modelos de Objetos e Dinâmico

modelo de Objetos	**modelo Funcional**					
	ator	**depósito de dados**	**processo**	**fluxo de dados**	**fluxo de objeto**	**fluxo de controle**
classe	nome (mesmo vértice)	nome (mesmo vértice)	fluxos de dados e de objeto e vínculos de responsabilidade e implementação	nome do depósito de dados ou do ator	nome do depósito de dados	
operação	vínculo de responsabilidade	vínculo de responsabilidade	vínculo de implementação	vínculos de responsabilidade e implementação	vínculos de responsabilidade e implementação	vínculo de implementação
generalização						
associação			vínculos de even-to, interação, responsabilidade e implementação			
agregação			vínculos de even-to, interação, responsabilidade e implementação			

Tabela 5.4. Integração entre os modelos de Objetos e Funcional

modelo Dinâmico	**modelo Funcional**					
	ator	**depósito de dados**	**processo**	**fluxo de dados**	**fluxo de objeto**	**fluxo de controle**
classe	nome (mesmo vértice)	nome (mesmo vértice)	fluxos de dados e de objetos			
estado	vínculo de estado	vínculo de estado	evento, condição, vínculos de atividade e implementação			
atividade		vínculo de responsabilidade	vínculo de implementação	vínculos de implementação	vínculos de implementação	
ação			vínculo de implementação	vínculos de implementação	vínculos de implementação	
evento			vínculo de implementação	vínculos de implementação	vínculos de implementação	vínculo de implementação
condição						vínculos de atividade e implementação

Tabela 5.5. Integração entre os modelos Dinâmico e Funcional

As combinações dos elementos integradores que constituem as conexões explícitas (ver tabela 5.2) permitem rastreamentos, na Matriz Square, que resultam em correlacionamentos importantes como: classes que participam de processos, eventos associados a processos, transições de estados que ocorrem em processos, eventos correspondentes a fluxos de dados ou a fluxos de controle, condições correspondentes a fluxos de controle e outras, conforme é mostrado nas tabelas 5.3, 5.4 e 5.5.

5.3.4 Premissas do Modelo Matricial

O objetivo do mecanismo integrador aqui proposto, a Matriz Square, é consistir as três dimensões da OMT que descrevem os objetos, o comportamento individual e global destes objetos e as funções do sistema do qual eles fazem parte, integrando estas informações de forma a modelar coerentemente este sistema. Assim como os modelos de Objeto, Dinâmico e Funcional são construídos a partir dos diagramas usados na metodologia OMT, o modelo matricial é construído a partir da Matriz Square [GON96]. Para isto, são adotadas as seguintes premissas básicas:

- O modelo matricial é constituído através da Matriz Square de forma a permitir a verificação de consistência entre os modelos da OMT, através das conexões explícitas, de dois modos:
  - localmente, a partir da visualização da matriz filtrada através de um processo de seleção por filtros, e
  - globalmente, pela aplicação de regras de consistência.

Nos dois casos, o objetivo é a verificação das conexões explícitas, através dos elementos integradores que são seus componentes.

- A integração da modelagem é conseguida pelo modelo matricial através da unificação dos elementos representados. Esta representação unificada, que permite a coerência do todo, não possibilita a visão geral do modelo. Primeiro, porque isto se torna impraticável a medida que o sistema cresce. Segundo, porque a consistência visual é viabilizada por fragmentos da matriz, filtrados adequadamente.
- O modelo matricial não dispensa uma ferramenta que automatize a construção da Matriz Square. São muitos os aspectos sintáticos a considerar no momento em que um elemento é modelado nos diagramas e traduzido para a matriz.

- Pela mesma razão, a Matriz Square como ferramenta gráfica não é destinada à entrada de dados. Os diagramas de Objeto, de Fluxo de Eventos, de Estados e de Fluxo de Dados não podem, por isso, serem dispensados. É através deles que é feita a entrada de dados para a construção da Matriz Square.
- O objetivo do modelo matricial não é descrever, mas integrar descrições de forma a torná-las coerentes entre si. Por isso, não substitui os modelos de Objetos, Dinâmico e Funcional. Não tem a intenção de ser um modelo que, sozinho, constrói a especificação de um sistema.

5.4 Conclusões

Em relação às atividades desenvolvidas pelo modelador, a integração dos modelos da OMT pretendida pelo modelo matricial necessita do apoio de um dicionário de dados mais completo e eficiente e de uma vinculação mais forte de operações com processos e eventos. A Matriz Square, que será descrita no próximo capítulo, considera vínculos entre os elementos modelados acrescidos àqueles representados tradicionalmente nos diagramas da OMT. Desta forma, elementos integradores constituem conexões explícitas entre os modelos que podem, então, ser integrados consistentemente. A proposta identifica os elementos dos três modelos que podem ser integrados através das conexões explícitas. Esta integração, centrada nas operações, pode ser feita visualmente ou com a aplicação de regras de consistência.

Os próximos quatro capítulos detalham a proposta de integração apresentando, no capítulo 6, a definição de Matriz Square com a notação utilizada e as vantagens da representação unificada; caracterizando, no capítulo 7, o que é necessário para uma ferramenta automatizar a construção do modelo matricial, suportando a metodologia OMT; e discutindo, nos capítulos 8 e 9, a integração dos modelos da OMT através do uso de filtros sobre a Matriz Square e da aplicação de regras de consistência.

6 A MATRIZ SQUARE

> "*Matrizes constituem ferramentas convenientes para sistematizar cálculos trabalhosos provendo uma notação compacta para armazenar informação e para descrever relacionamentos complicados.*"
> Noble e Daniel [NOB88]

6.1 Introdução

Neste capítulo é proposta uma nova ferramenta gráfica, a Matriz Square, como mecanismo integrador dos modelos da OMT. Na seção 6.2 são apresentados conceitos, como de matriz associada a um grafo e de N^2Chart, que deram origem à Matriz Square, e as definições dos diagramas utilizados na OMT que servem de base para a formalização apresentada. Na seção 6.3 é apresentada a notação dos elementos da metodologia OMT na matriz. Na seção 6.4 são discutidas as vantagens da representação unificada.

6.2 Definição e Origem da Matriz Square

As definições apresentadas nesta dissertação empregam a terminologia de linguagens de primeira ordem, como apresentada por exemplo por Casanova et al [CAS87].

6.2.1 Matriz Associada a um Grafo

A teoria dos grafos teve marcado seu início, em meados do século XIX, através da formulação de problemas algorítmicos [SZW84]. O objetivo principal era (e continua sendo) a busca da solução mais eficiente possível (ou uma delas). Na modelagem de software os grafos são utilizados com o objetivo de buscar a visualização de uma solução, tanto para testar sua eficiência como para viabilizar comunicação e documentação.

Definição 1: (grafo) (SZW84)
Um **grafo** G(V,E) é composto por
(1) um conjunto finito não-vazio V, sendo $\boldsymbol{v} \in$ V um **vértice**.
(2) um conjunto finito E, sendo $\boldsymbol{e} = (\boldsymbol{v_i},\boldsymbol{v_j}) \in$ E um **arco**.
O grafo G é dito **não direcionado** quando $(\boldsymbol{v_i},\boldsymbol{v_j}) \in$ E é um par não ordenado.
O grafo G é dito **direcionado** quando $(\boldsymbol{v_i},\boldsymbol{v_j}) \in$ E é um par ordenado e, neste caso, cada arco possui uma só direção de $\boldsymbol{v_i}$ para $\boldsymbol{v_j}$.

Uma matriz associada a um grafo (figura 6.1) — ou matriz de adjacências — é uma matriz nxn (n = número de vértices do grafo) onde o elemento ij, na matriz, vale 1 se houver arco (aresta) entre os vértices i e j e vale 0 se não houver [MAC74].

Definição 2: (matriz) [STE72]
Uma **matriz m por n** denotada por $A = [r_{ij}]$ é um quadro de mxn elementos dispostos em m linhas e n colunas. O elemento r_{ij} de A pertence à linha i e à coluna j.

Definição 3: (matriz quadrada) [STE72]
A matriz $Q = [r_{ij}]$ com n linhas e n colunas é dita **quadrada**. Diz-se que a matriz Q é de ordem n.

A matriz associada a um grafo, também chamada de matriz de adjacências representa um grafo sem ambigüidades [SZW84].

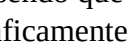
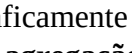
Definição 4: (matriz de adjacências) [SZW84]
Dado um grafo G(V,E), com n vértices $v \in V$, uma **matriz de adjacências** é uma matriz quadrada $A = [r_{ij}]$ de ordem n associada ao grafo G, tal que:
(1) se $r_{ij} = 1$ então $(v_i,v_j) \in E$
(2) se $r_{ij} = 0$ então $(v_i,v_j) \notin E$

Figura 6.1. Exemplos de Grafo e de Matriz de Adjacências.

6.2.2 Em Direção à Formalização

As definições 5, 6, 7 e 8, dos diagramas utilizados na metodologia OMT, baseiam-se nos conceitos apresentados anteriormente no capítulo 4, referentes à metodologia OMT, e no capítulo 5, com relação às considerações feitas para a integração dos modelos. Estas definições servem de

base para a construção da definição da matriz Square. Serão assumidos como princípios ontológicos que: "*o mundo é composto por coisas e todas as coisas possuem propriedades*" [WAN89]. Estas coisas são chamadas *objetos*. Eles têm identidades discretas e distintas; suas propriedades são seus atributos, suas operações e suas ligações com outros objetos; os objetos que possuem as mesmas propriedades são agrupados numa classe e estas classes são organizadas hierarquicamente, compartilhando propriedades com base em relacionamentos de generalização [RUM91].

Definição 5: (diagrama de Objetos)
Um **diagrama de Objetos** é um grafo direcionado $D^O(C,R)$ composto por:
(1) um conjunto finito não-vazio C, tal que
$\forall \boldsymbol{c} \in C$, $\boldsymbol{c} = (\boldsymbol{n}, P)$ é uma **classe**, sendo
$\boldsymbol{n}$ um **nome de classe** e
$P = (\{a_1, ..., a_f\},\{\boldsymbol{m_1}, ..., \boldsymbol{m_g}\})$ **propriedades de classe** , onde
$a_1, ..., a_f$ são f **atributos**, e
$\boldsymbol{m_1} = (\boldsymbol{nm_1}, A_1, T_1)$, ..., $\boldsymbol{m_g} = (\boldsymbol{nm_g}, A_g, T_g)$ são g **operações**, sendo
$\boldsymbol{nm_1}, ..., \boldsymbol{nm_g}$ **nomes de operação**,
$A_1, ..., A_g$ **argumentos** e
$T_1, ... T_g$ **tipos de dados de retorno**.
(2) um conjunto finito R de arcos, tal que
$\forall(\boldsymbol{c_i},\boldsymbol{c_j}) \in R$, $\boldsymbol{c_i} \in C$, $\boldsymbol{c_j} \in C$, sendo que
(a) Se $(\boldsymbol{c_i},\boldsymbol{c_j})$ é denotado graficamente por  de $\boldsymbol{c_i}$ para $\boldsymbol{c_j}$ então
$(\boldsymbol{c_i},\boldsymbol{c_j})$ é uma **generalização**, $\boldsymbol{c_i}$ é uma **subclasse** e $\boldsymbol{c_j}$ é uma **superclasse**.
(b) Se $(\boldsymbol{c_i},\boldsymbol{c_j})$ é denotado graficamente por  de $\boldsymbol{c_i}$ para $\boldsymbol{c_j}$ então
$(\boldsymbol{c_i},\boldsymbol{c_j})$ é uma **generalização com classes não-disjuntas**, $\boldsymbol{c_i}$ é uma **subclasse** e
$\boldsymbol{c_j}$ é uma **superclasse**.
(c) Se $(\boldsymbol{c_i},\boldsymbol{c_j})$ é denotado graficamente por de $\boldsymbol{c_i}$ para $\boldsymbol{c_j}$ então
$(\boldsymbol{c_i},\boldsymbol{c_j}) = (as, pp, cdn)$ é uma **associação**, onde
as é um **nome de associação**, $pp = (pp_i, pp_j)$ são **papéis** e
$cdn = (cdn_i, cdn_j)$ são **cardinalidades** respectivamente de $\boldsymbol{c_i}$ e $\boldsymbol{c_j}$,
Diz-se que uma associação é **reflexiva** se i = j.
Diz-se que uma associação é uma **associação como classe**
quando $(\boldsymbol{c_i},\boldsymbol{c_j}) = (as, pp, cdn, P)$, sendo P propriedades de classe.
(d) Se $(\boldsymbol{c_i},\boldsymbol{c_j})$ é denotado graficamente por  de $\boldsymbol{c_i}$ para $\boldsymbol{c_j}$ então
$(\boldsymbol{c_i},\boldsymbol{c_j}) = (ag, cdn_j)$ é uma **agregação**, onde
ag é um **nome de agregação** e cdn_j é a **cardinalidade** de $\boldsymbol{c_j}$, sendo
$\boldsymbol{c_i}$ é uma **classe agregada** e $\boldsymbol{c_j}$ é uma **classe componente**.

Comentário:
Um diagrama de Objetos possui um único tipo de vértice (classe) e três tipos de arcos (generalização, associação e agregação) que são diferenciados pela notação gráfica.

Os objetos que compõem um sistema comunicam-se e estimulam-se através de eventos. Estes são ativados por operações dos objetos de onde se originam e ativam operações dos objetos para

onde se destinam. Assim é definido o comportamento global do sistema.

Definição 6: (diagrama de Fluxo de Eventos)
Um diagrama de Fluxo de Eventos é um grafo direcionado $D^E(C,EV)$ composto por:
(1) um conjunto finito não-vazio C, tal que
$\forall \boldsymbol{c} \in$ C, $\boldsymbol{c}$ é uma classe, sendo **n** o seu nome de classe
(2) um conjunto finito EV de arcos, tal que
$\forall(\boldsymbol{c_i},\boldsymbol{c_j}) \in$ EV, $\boldsymbol{c_i} \in$ C, $\boldsymbol{c_j} \in$ C, $\boldsymbol{ev_{ij}} = (\boldsymbol{c_i}, \boldsymbol{c_j})$, sendo que
$\boldsymbol{ev_{ij}}$ é um **evento** de $\boldsymbol{c_i}$ para $\boldsymbol{c_j}$,
denotado por (ev, $\boldsymbol{m_i}$, $\boldsymbol{m_j}$) ou (ev, $\boldsymbol{m_i}$, $\boldsymbol{c_j}$) ou (ev, $\boldsymbol{c_i}$, $\boldsymbol{m_i}$), onde
ev é um **nome de evento**,
$\boldsymbol{m_i}$ é uma operação de $\boldsymbol{c_i}$ chamada **operação-causa** e
$\boldsymbol{m_j}$ é uma operação de $\boldsymbol{c_j}$ chamada **operação-efeito**.

Comentário:
Um diagrama de Fluxo de Eventos possui um único tipo de vértice (classe) e um único tipo de arco (evento).

O comportamento individual de cada objeto, em termos de transições dos seus estados, é definido através do seu ciclo de vida.

Definição 7: (diagrama de Estados)
Dado um diagrama de Fluxo de Eventos $D^E(C,EV)$, um **diagrama de Estados**, referente a $\boldsymbol{c} \in$ C, é um grafo direcionado $D^T(S,TR)$ composto por:
(1) um conjunto finito não-vazio S, tal que
$\forall \boldsymbol{s} \in$ S, $\boldsymbol{s}$ = (s, $\boldsymbol{m}$) é um **estado** de uma classe $\boldsymbol{c}$, sendo que
s é um **nome de estado** e
$\boldsymbol{m}$ é uma operação chamada **atividade** de $\boldsymbol{s}$.
(2) um conjunto finito TR de arcos, tal que
$\forall(\boldsymbol{s_i},\boldsymbol{s_j}) \in$ TR, $\boldsymbol{s_i} \in$ S, $\boldsymbol{s_j} \in$ S,
$(\boldsymbol{s_i},\boldsymbol{s_j})$ é uma **transição** de $\boldsymbol{s_i}$ para $\boldsymbol{s_j}$,
denotada por ($\boldsymbol{ev}$, $\boldsymbol{cn}$, $\boldsymbol{m}$) ou ($\boldsymbol{cn}$, $\boldsymbol{m}$) ou ($\boldsymbol{ev}$, $\boldsymbol{cn}$) ou ($\boldsymbol{cn}$), sendo que
$\boldsymbol{ev}$ é um evento,
$\boldsymbol{cn}$ é uma **condição** e
$\boldsymbol{m}$ é uma operação chamada **ação**.
Um $D^T(S,TR)$ é subdividido em n níveis $\boldsymbol{N^T}$, sendo que
se $\boldsymbol{s_i}$ está em $\boldsymbol{N^T_k}$ e $\boldsymbol{s_j}$ está em $\boldsymbol{N^T_{k+1}}$ então $\boldsymbol{s_j}$ é **subestado** de $\boldsymbol{s_i}$.
Um $D^T(S,TR)$ é subdividido em n subdiagramas de Estados, chamados **subdiagramas de concorrência** $\boldsymbol{N^c}$, sendo que
se $\boldsymbol{s_i}$ está em $\boldsymbol{N^c_k}$ e $\boldsymbol{s_j}$ está em $\boldsymbol{N^c_{k+1}}$ então $\boldsymbol{s_j}$ é **concorrente num único objeto** de $\boldsymbol{s_i}$.

Comentário:
Um diagrama de Estados possui um único tipo de vértice (estado) e um único tipo de arco (transição).

As funções do sistema determinam como os dados são transformados e quais os caminhos possíveis que podem tomar.

Definição 8 (diagrama de Fluxo de Dados)
Um diagrama de Fluxo de Dados é um grafo direcionado $D^D(P,FL)$ composto por
(1) um conjunto finito não-vazio P, tal que $\forall \boldsymbol{x} \in P$,
(a) Se $\boldsymbol{x}$ é denotado graficamente por (p) então
$\boldsymbol{x} = \boldsymbol{p}$ é um **processo** e p é um **nome de processo**
(b) Se $\boldsymbol{x}$ é denotado graficamente por ***n*** então
$\boldsymbol{x} = \boldsymbol{c^d}$ é um **depósito de dados** e $\boldsymbol{n}$ é um nome de classe
(c) Se $\boldsymbol{x}$ é denotado graficamente por [***n***] então
$\boldsymbol{x} = \boldsymbol{c^a}$ é um **ator** e $\boldsymbol{n}$ é um nome de classe
(2) um conjunto finito FL de arcos, tal que:
$\forall(\boldsymbol{x_i},\boldsymbol{x_j}) \in FL, \boldsymbol{x_i} \in P, \boldsymbol{x_j} \in P$, sendo que
(a) Se $(\boldsymbol{x_i},\boldsymbol{x_j})$ é denotado graficamente por ⟶▷ de $\boldsymbol{x_i}$ para $\boldsymbol{x_j}$ então
$(\boldsymbol{x_i},\boldsymbol{x_j})$ é um **fluxo de objeto**,
sendo $\boldsymbol{x_i} = \boldsymbol{p}$ um processo e $\boldsymbol{x_j} = \boldsymbol{c^d}$ um depósito de dados.
(b) Se $(\boldsymbol{x_i},\boldsymbol{x_j})$ é denotado graficamente por ⟶ de $\boldsymbol{x_i}$ para $\boldsymbol{x_j}$ então
$(\boldsymbol{x_i},\boldsymbol{x_j}) = \{d_1, ..., d_n\}$ é um **fluxo de dados**, onde $d_1, ..., d_n$ são **dados**,
sendo que
(b1) se $\boldsymbol{x_i} = \boldsymbol{c^d}$ é um depósito de dados ou $\boldsymbol{x_i} = \boldsymbol{c^a}$ é um ator
então $\boldsymbol{x_j} = \boldsymbol{p}$ é um processo
(b2) se $\boldsymbol{x_j} = \boldsymbol{c^d}$ é um depósito de dados ou $\boldsymbol{x_j} = \boldsymbol{c^a}$ é um ator
então $\boldsymbol{x_i} = \boldsymbol{p}$ é um processo
(c) Se $(\boldsymbol{x_i},\boldsymbol{x_j})$ é denotado graficamente por ⋯▸ de $\boldsymbol{x_i}$ para $\boldsymbol{x_j}$ então
$(\boldsymbol{x_i},\boldsymbol{x_j}) = \boldsymbol{cn}$ é um **fluxo de controle**, onde $\boldsymbol{cn}$ é uma condição,
sendo $\boldsymbol{x_i} = \boldsymbol{p}$ um processo e $\boldsymbol{x_j} = \boldsymbol{p}$ um processo.
Um $D^D(P,FL)$ é subdividido em n níveis $\boldsymbol{N^D}$, sendo que
se $\boldsymbol{p_i}$ está em $\boldsymbol{N^D_k}$ e $\boldsymbol{p_j}$ está em $\boldsymbol{N^D_{k+1}}$ então $\boldsymbol{p_j}$ é **subprocesso** de $\boldsymbol{p_i}$.

Comentário:
Um diagrama de Fluxo de Dados possui três tipos de vértices (processo, depósito de dados e ator) e três tipos de arcos (fluxo de objeto, fluxo de dados e fluxo de controle) e por isso vértices e arcos necessitam ser diferenciados, entre si, pela notação gráfica.

Esta dissertação, em termos de definições, não enfocará axiomas que determinem regras de consistência internas sobre os diagramas da OMT (por exemplo: balanceamento de fluxos de dados em níveis diferentes ou transição de estado obrigatória chegando a um estado existente). Esta preocupação não deve ser abandonada, entretanto, numa implementação completa de uma ferramenta que apoie a Matriz Square.

Definição 9: (diagrama OMT)

Dados os grafos $D^O(C,R)$, $D^E(C,EV)$, $D^T(S,TR)$ e $D^D(P,FL)$, diz-se que o grafo $G^{OMT}(V,E)$ é um **diagrama OMT**, tal que:

(1) $C \subset V$, $S \subset V$, $P \subset V$, $R \subset E$, $EV \subset E$, $TR \subset E$ e $FL \subset E$.

(2) Sejam uma classe $\boldsymbol{c_i} \in C$ com nome $\boldsymbol{n_i}$ e um depósito de dados $\boldsymbol{c^d_j} \in P$ com nome $\boldsymbol{n_j}$, se $\boldsymbol{n_i} = \boldsymbol{n_j}$ então $\boldsymbol{c_i} = \boldsymbol{c^d_j} = v \in V$.

(3) Sejam uma classe $\boldsymbol{c_i} \in C$ com nome $\boldsymbol{n_i}$ e um ator $\boldsymbol{c^a_j} \in P$ com nome $\boldsymbol{n_j}$, se $\boldsymbol{n_i} = \boldsymbol{n_j}$ então $\boldsymbol{c_i} = \boldsymbol{c^a_j} = v \in V$.

(4) $\forall$operação $\boldsymbol{m}$ de $\forall$classe $\boldsymbol{c} \in C$, $\boldsymbol{m} = v \in V$ e
A = $(\boldsymbol{c}, \boldsymbol{m}) \in E$ é um **vínculo de responsabilidade** onde A são os argumentos de $\boldsymbol{m}$, chamados **entrada** do vínculo de responsabilidade e
T = $(\boldsymbol{m}, \boldsymbol{c}) \in E$ é o **retorno** do vínculo de responsabilidade onde T são os tipos de dados de retorno de $\boldsymbol{m}$.

(5) $\forall(\boldsymbol{c_i}, \boldsymbol{c_j}) \in E$, se $\boldsymbol{c_i} \in C$ e $\boldsymbol{c_j} \in C$ são classes e $\boldsymbol{ev_{ij}}$ é um evento de $\boldsymbol{c_i}$ para $\boldsymbol{c_j}$, então $(\boldsymbol{c_i}, \boldsymbol{c_j})$ é um **vínculo de evento** e
(a) Se $\boldsymbol{ev_{ij}}$ é denotado por (ev, $\boldsymbol{m_f}$, $\boldsymbol{m_g}$) então $\boldsymbol{ev_{ij}} = (\boldsymbol{m_f}, \boldsymbol{m_g}) \in E$
(b) Se $\boldsymbol{ev_{ij}}$ é denotado por (ev, $\boldsymbol{m_f}$, $\boldsymbol{c_j}$) então $\boldsymbol{ev_{ij}} = (\boldsymbol{m_f}, \boldsymbol{c_j}) \in E$
(c) Se $\boldsymbol{ev_{ij}}$ é denotado por (ev, $\boldsymbol{c_i}$, $\boldsymbol{m_g}$) então $\boldsymbol{ev_{ij}} = (\boldsymbol{c_i}, \boldsymbol{m_g}) \in E$
onde a operação $\boldsymbol{m_f}$ de $\boldsymbol{c_i}$ é a operação-causa do evento $\boldsymbol{ev_i}$ e
a operação $\boldsymbol{m_g}$ de $\boldsymbol{c_j}$ é a operação-efeito do evento $\boldsymbol{ev_i}$

(6) $\forall(\boldsymbol{m}, \boldsymbol{p}) \in E$, sendo $\boldsymbol{m} \in V$ uma operação e $\boldsymbol{p} \in P$ um processo, $(\boldsymbol{m}, \boldsymbol{p})$ é um **vínculo de implementação**.

(7) $\forall(\boldsymbol{m_i}, \boldsymbol{m_j}) \in E$, sendo $\boldsymbol{m_i}$ uma operação de uma classe $\boldsymbol{c_h} \in C$, $\boldsymbol{m_j}$ uma operação de uma classe $\boldsymbol{c_k} \in C$ e $\boldsymbol{c_h} \neq \boldsymbol{c_k}$, se
(a) $\exists(\boldsymbol{p}, \boldsymbol{m_i})$ e $\exists(\boldsymbol{p}, \boldsymbol{m_j})$ como vínculos de implementação, sendo $\boldsymbol{p} \in P$ um processo, ou
(b) $\exists(\boldsymbol{p_f}, \boldsymbol{m_i})$ e $\exists(\boldsymbol{p_g}, \boldsymbol{m_j})$ como vínculos de implementação e $\exists(\boldsymbol{p_f}, \boldsymbol{p_g})$ como fluxo de dados, sendo $\boldsymbol{p_f} \in P$ e $\boldsymbol{p_g} \in P$ processos,
então $(\boldsymbol{m_i}, \boldsymbol{m_j})$ é um **vínculo de interação** entre as operações $\boldsymbol{m_i}$ e $\boldsymbol{m_j}$

(8) $\forall(\boldsymbol{c}, \boldsymbol{s}) \in E$, se $\boldsymbol{c} \in C$ é uma classe e $\boldsymbol{s} \in S$ é um estado de $\boldsymbol{c}$, então $(\boldsymbol{c}, \boldsymbol{s})$ é um **vínculo de estado**.

(9) $\forall(\boldsymbol{s}, \boldsymbol{m}) \in E$, se $\boldsymbol{s} \in S$ é um estado e $\boldsymbol{m}$ é uma operação chamada atividade de $\boldsymbol{s}$, então $(\boldsymbol{s}, \boldsymbol{m})$ é um **vínculo de atividade**.

(10) $\forall(\boldsymbol{s_j}, \boldsymbol{s_i}) \in E$, se $\boldsymbol{s_i} \in S$ é um estado e $\boldsymbol{s_j} \in S$ é um subestado de $\boldsymbol{s_i}$, então $(\boldsymbol{s_j}, \boldsymbol{s_i})$ é um **vínculo de subestado**.

(11) $\forall(\boldsymbol{s_i}, \boldsymbol{s_j}) \in E$, se $\boldsymbol{s_i} \in S$ e $\boldsymbol{s_j} \in S$ são estados concorrentes num único objeto, então $(\boldsymbol{s_i}, \boldsymbol{s_j})$ é um **vínculo de concorrência**.

(12) $\forall(\boldsymbol{p_i}, \boldsymbol{p_j}) \in E$, se $\boldsymbol{p_i} \in P$ é um subprocesso do processo $\boldsymbol{p_j} \in P$, então $(\boldsymbol{p_i}, \boldsymbol{p_j})$ é um **vínculo de subprocesso**.

(13) $\forall(\boldsymbol{s_i}, \boldsymbol{s_j}) \in E$, sendo que $\boldsymbol{s_i} \in S$ e $\boldsymbol{s_j} \in S$ são estados e $(\boldsymbol{s_i}, \boldsymbol{s_j})$ é uma transição,
(a) Se $(\boldsymbol{s_i}, \boldsymbol{s_j})$ é denotada por $(\boldsymbol{ev}, \boldsymbol{cn}, \boldsymbol{m})$ então
ev = $(\boldsymbol{s_i}, \boldsymbol{m_f}) \in E$, $\boldsymbol{cn} = (\boldsymbol{m_f}, \boldsymbol{s_j}) \in E$ e $\boldsymbol{cn} = (\boldsymbol{m_f}, \boldsymbol{m}) \in E$
(b) Se $(\boldsymbol{s_i}, \boldsymbol{s_j})$ é denotada por $(\boldsymbol{cn}, \boldsymbol{m})$ então $\boldsymbol{cn} = (\boldsymbol{s_i}, \boldsymbol{s_j}) \in E$ e $\boldsymbol{cn} = (\boldsymbol{s_i}, \boldsymbol{m}) \in E$
(c) Se $(\boldsymbol{s_i}, \boldsymbol{s_j})$ é denotada por $(\boldsymbol{ev}, \boldsymbol{cn})$ então ev = $(\boldsymbol{s_i}, \boldsymbol{m_f}) \in E$ e $\boldsymbol{cn} = (\boldsymbol{m_f}, \boldsymbol{s_j}) \in E$
(d) Se $(\boldsymbol{s_i}, \boldsymbol{s_j})$ é denotada por $(\boldsymbol{cn})$ então $\boldsymbol{cn} = (\boldsymbol{s_i}, \boldsymbol{s_j}) \in E$
onde $\boldsymbol{ev}$ é um evento, $\boldsymbol{cn}$ é uma condição, $\boldsymbol{m}$ é uma ação,
ev é o nome do evento $\boldsymbol{ev}$ e $\boldsymbol{m_f}$ é a operação-causa do evento $\boldsymbol{ev}$.

6.2.3 Definição de Matriz Square

A matriz Square é uma extensão do conceito de matriz de adjacências, onde o valor 1, indicando a existência de um arco, é substituído por aquilo que o arco representa. O grafo associado é formado pelo conjunto dos diagramas da metodologia OMT. Os elementos-vértice são colocados na diagonal principal da matriz e os elementos-arco fora dela, na interface entre dois elementos-vértice (figura 6.2). Os elementos-vértice são classes de objetos, estados, operações e processos. Os elementos-arco são os outros elementos modelados nos diagramas, conforme as definições 5 a 9.

Definição 10: (matriz de adjacências estendida)
Dado um grafo G(V,E), com n vértices $v \in V$, uma **matriz de adjacências estendida** é uma matriz quadrada $A = [r_{ij}]$ de ordem n associada ao grafo G, tal que:
(1) se $(v_i,v_j) \in E$ então $r_{ij} = (v_i,v_j)$ é um **elemento-arco**
(2) se $v_i \in V$ então $r_{ii} = v_i$ é um **elemento-vértice**

Definição 11: (matriz square)
Dado um grafo $G^{OMT}(V,E)$ com no classes $\boldsymbol{c} \in V$, ns estados $\boldsymbol{s} \in V$, nm operações $\boldsymbol{m}$ das classes $\boldsymbol{c} \in V$ e np processos $\boldsymbol{p} \in V$, sendo que q = no + ns + nm + np, diz-se que a matriz de adjacências estendida $\boldsymbol{S} = [r_{ij}]$ de ordem q é uma **matriz square** associada ao diagrama OMT $G^{OMT}(V,E)$, sendo que
(1) um elemento-arco $r_{ij} = (v_i, v_j)$ é
uma generalização ou
uma generalização com classes não-disjuntas ou
uma associação ou uma agregação ou
um evento ou uma condição ou um vínculo de interação ou
uma fluxo de dados ou um fluxo de objeto ou um fluxo de controle ou
um vínculo de responsabilidade ou seu retorno ou um vínculo de evento ou
um vínculo de implementação ou um vínculo de estado ou
um vínculo de atividade ou um vínculo de subestado ou
um vínculo de concorrência ou um vínculo de subprocesso e
(2) um elemento-vértice $r_{ii} = v_i$ é
uma classe ou uma classe com associação reflexiva ou
um depósito de dados ou um ator ou
um estado ou uma operação ou um processo.

Comentário:
Para simplificação, a definição considera que os índices de r, na matriz, e dos vértices dos diagramas são os mesmos. Na verdade o mapeamento de um índice de r para um índice de vértice, na ferramenta Square, é feito pela Tabela de Elementos.

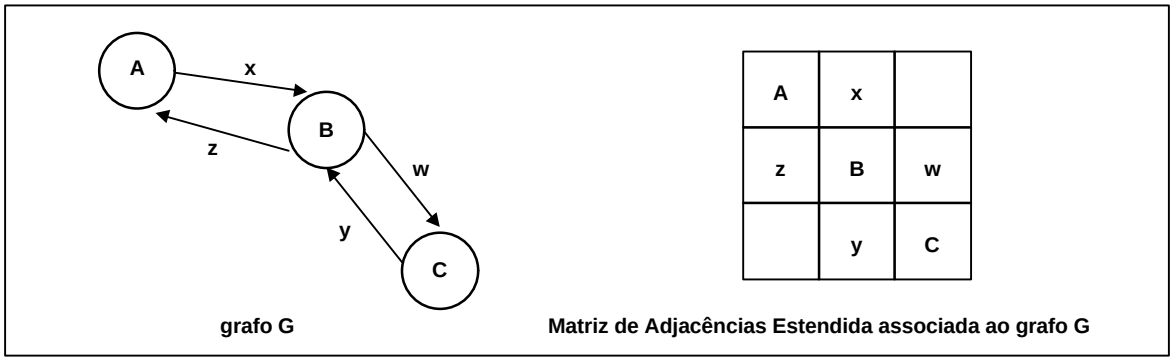


Figura 6.2. Exemplo de Matriz de Adjacências Estendida.

O nome da matriz vem da palavra inglesa “square” que significa, entre outras coisas [MIC90]:

- *quadrado* (como substantivo);
- *dividir em quadrados, pôr no esquadro, endireitar, ajustar* (como verbo);
- *quadrado, direito* (como adjetivo) e
- *honesto, justo* (em sentido figurado).

O significado do nome fica, assim, adequado à idéia (correção, ajuste, consistência) e à forma (quadrada).

6.2.4 Interface entre os Elementos da Matriz Square

Ao ser traduzido um diagrama da OMT, a Matriz Square (figura 6.2) representa os elementos-vértice na diagonal principal. Na linha de um elemento-vértice (na matriz) é representada a sua interface de saída (arcos que partem daquele vértice, no diagrama) e na coluna, a sua interface de entrada (arcos que chegam nele).

Definição 12: (interface de entrada e de saída)
Dada a matriz square $\boldsymbol{S} = [r_{ij}]$, de ordem q, e um elemento-arco r_{ij} então r_{ij} é uma **interface** entre r_{ii} e r_{jj}, sendo
(1) r_{ij} é uma **interface de entrada** de r_{jj} e
(2) r_{ij} é uma **interface de saída** de r_{ii}

A leitura é sempre feita no sentido horário. A figura 6.2 mostra a tradução simplificada de um

diagrama genérico da OMT: os elementos-arco x e y pertencem à interface de entrada do elemento-vértice B e os elementos-arco z e w pertencem à sua interface de saída. Por outro lado, os elementos-arco x e y pertencem, respectivamente, às interfaces de saída de A e C e os elementos-arco z e w pertencem, respectivamente, às interfaces de entrada de A e C.

6.2.5 Os Setores da Matriz Square

Os setores são idealizados para facilitar a localização dos elementos representados. Naturalmente, como em qualquer técnica ou ferramenta, a prática conseguida pelo uso continuado irá facilitar a leitura da matriz. O objetivo é que a interpretação da matriz, com os setores evidenciando áreas de representação, fique direcionada e, assim, mais ágil.

Definição 13: (diagonal principal da matriz square)
Dada a matriz square $\boldsymbol{S} = [r_{ij}]$, de ordem q, associada a um grafo $G^{OMT}(V,E)$, a **diagonal principal** de $\boldsymbol{S}$ é um conjunto ordenado
$\mathbf{d}(\boldsymbol{X},\boldsymbol{Y}) = \{r_{11}, r_{22}, \ldots, r_{XX}, \ldots, r_{YY}, \ldots, r_{qq}\}$, tal que:
(1) $\forall r_{ii}$, se r_{ii} é uma classe $\boldsymbol{c_i} \in V$ ou r_{ii} é um estado $\boldsymbol{s} \in V$ então $i < \boldsymbol{X}$
(2) $\forall r_{ii}$, se r_{ii} é uma operação $\boldsymbol{m} \in V$ então $i \geq \boldsymbol{X}$ e $i \leq \boldsymbol{Y}$
(3) $\forall r_{ii}$, se r_{ii} é um processo $\boldsymbol{p} \in V$ então $i > \boldsymbol{Y}$

Comentário:
A definição de diagonal principal é importante para estabelecer a localização dos elementos-vértice que, por sua vez, é necessária para localizar os setores na matriz. Além do que estabelece a definição, a ordem dos elementos-vértice é ditada pela ordem com que eles são diagramados.

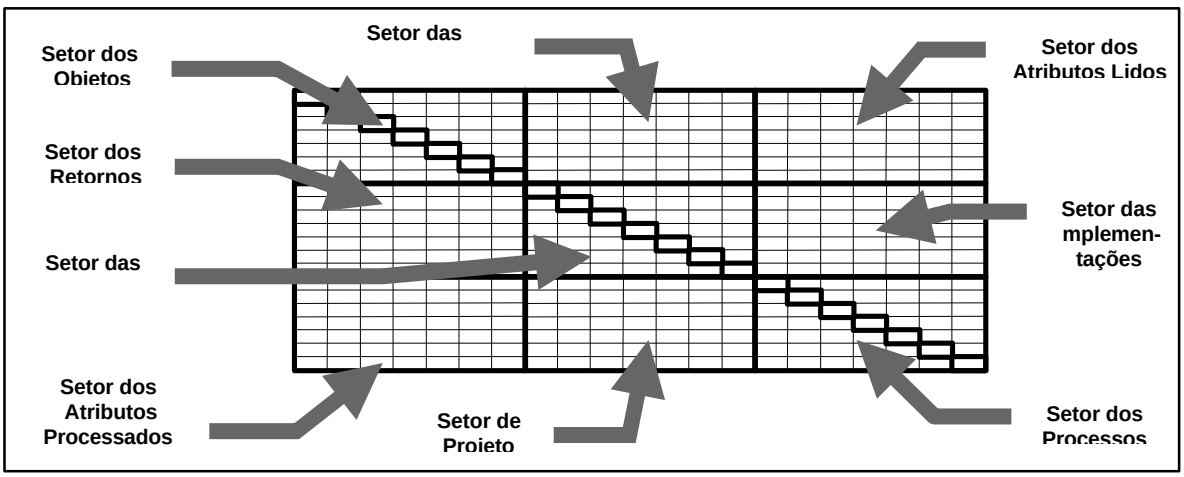


Figura 6.3. Setores da matriz Square

A matriz Square é dividida em nove setores, conforme é mostrado na figura 6.3. Como o agente integrador utilizado neste trabalho são as operações, elas estão centralizadas na diagonal principal da Matriz Square e determinam os limites dos nove setores. Os separadores verticais e horizontais localizam-se nos limites da área central onde são representadas as operações.

Os setores da matriz Square representam elementos específicos que são modelados nos diferentes diagramas da OMT. A seguir são apresentados os setores e os elementos neles representados, na tabela 6.1.

setor	elementos representados	interface entre
dos Objetos	classe	
	estado	
	generalização, associação, agregação	classes
	condição	estados
	vínculo de estado	classe e estado
	vínculo de subestado	estados
	vínculo de concorrência	estados
	vínculo de evento	classes
das Responsa-bilidades	evento	classe que envia o evento e operação-efeito
	entrada de vínculo de responsabilidade	classe e operação
	vínculo de atividade	estado e operação
dos Atributos Lidos	fluxo de dados	depósito de dados ou ator e processo
dos Retornos	evento	operação-causa e classe que recebe o evento
	retorno de vínculo de responsabilidade	operação e classe
	condição	operação-efeito de evento e estado
das Operações	operação	
	evento	operação-causa e operação-efeito
	condição	operação-efeito de evento e ação
	vínculo de interação	operações
das Imple-mentações	vínculo de implementação	operação e processo
dos Atributos Processados	fluxo de dados	processo e depósito de dados ou ator
	fluxo de objeto	processo e depósito de dados
de Projeto	nenhum na fase de análise, assim como é estabelecida aqui nesta dissertação	
dos Processos	processo	
	fluxo de dados	processos
	fluxo de controle	processos
	vínculo de subprocesso	processos

Tabela 6.1. Setores da matriz Square e elementos representados

Definição 14: (matriz particionada, submatrizes da partição) [NOB88]
A matriz A = $[r_{ij}]$ é dita **particionada** quando separadores verticais são desenhados entre

colunas selecionadas e separadores horizontais são desenhados entre linhas selecionadas na matriz. As matrizes menores formadas pelos elementos contidos nas áreas formadas por esses separadores são ditas **submatrizes da partição** de A.

Dada a matriz square $\boldsymbol{S} = [r_{ij}]$, de ordem q, associada a um grafo $G^{OMT}(V,E)$,), com diagonal principal **d(*X*,*Y*)**, definem-se os seus setores assim como segue:

Definição 15: (setor de objetos)
O **setor de objetos** denotado por $\boldsymbol{S^O} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, de ordem $\boldsymbol{X}$ - 1, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^O}$, $1 \leq i < \boldsymbol{X} \wedge 1 \leq j < \boldsymbol{X}$
(2) $\forall r'_{ii} \in \boldsymbol{S^O}$, r'_{ii} é uma classe ou r'_{ii} é um estado
(3) $\forall r'_{ij} \in \boldsymbol{S^O}$, se r'_{ij} é uma generalização ou r'_{ij} é uma associação ou r'_{ij} é uma agregação então r'_{ii} e r'_{jj} são classes.
(4) $\forall r'_{ij} \in \boldsymbol{S^O}$, se r'_{ij} é um vínculo de estado então r'_{ii} é uma classe e r'_{jj} é um estado.
(5) $\forall r'_{ij} \in \boldsymbol{S^O}$, se r'_{ij} é condição ou r'_{ij} é um vínculo de subestado ou r'_{ij} é um vínculo de concorrência então r'_{ii} e r'_{jj} são estados.
(6) $\forall r'_{ij} \in \boldsymbol{S^O}$, se r'_{ij} é um vínculo de evento então r'_{ii} e r'_{jj} são classes.

Definição 16: (setor das responsabilidades)
O **setor das responsabilidades** denotado por $\boldsymbol{S^{RP}} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, com $\boldsymbol{X}$ -1 linhas e $\boldsymbol{Y}$ - $\boldsymbol{X}$ + 1 colunas, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^{RP}}$, $1 \leq i < \boldsymbol{X} \wedge \boldsymbol{X} \leq j \leq \boldsymbol{Y} \wedge i < j$
(2) $\forall r'_{ij} \in \boldsymbol{S^{RP}}$, se r'_{ij} é um evento então r'_{ii} é uma classe e r'_{jj} é uma operação
(3) $\forall r'_{ij} \in \boldsymbol{S^{RP}}$, se r'_{ij} é uma entrada de um vínculo de responsabilidade então r'_{ii} é uma classe e r'_{jj} é uma operação.
(4) $\forall r'_{ij} \in \boldsymbol{S^{RP}}$, se r'_{ij} é um vínculo de atividade então r'_{ii} é um estado e r'_{jj} é uma operação, chamada atividade.

Definição 17: (setor dos atributos lidos)
O **setor dos atributos lidos** denotado por $\boldsymbol{S^{AL}} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, com $\boldsymbol{X}$ -1 linhas e q - $\boldsymbol{Y}$ colunas, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^{AL}}$, $1 \leq i < \boldsymbol{X} \wedge \boldsymbol{Y} < j \leq q \wedge i < j$
(2) $\forall r'_{ij} \in \boldsymbol{S^{AL}}$, se r'_{ij} é um fluxo de dados então r'_{ii} é uma classe e r'_{jj} é um processo

Definição 18: (setor dos retornos)
O **setor dos retornos** denotado por $\boldsymbol{S^{RT}} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$,

com $\boldsymbol{Y}$ - $\boldsymbol{X}$ + 1 linhas e $\boldsymbol{X}$ - 1 colunas, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^{AL}}$, $\boldsymbol{X} \leq i \leq \boldsymbol{Y} \wedge 1 \leq j < \boldsymbol{X} \wedge i > j$
(2) $\forall r'_{ij} \in \boldsymbol{S^{RT}}$, se r'_{ij} é um evento então r'_{ii} é uma operação e r'_{jj} é uma classe
(3) $\forall r'_{ij} \in \boldsymbol{S^{RT}}$, se r'_{ij} é um retorno de um vínculo de responsabilidade então r'_{ii} é uma operação e r'_{jj} é uma classe.
(4) $\forall r'_{ij} \in \boldsymbol{S^{RT}}$, se r'_{ij} é uma condição então r'_{ii} é uma operação e r'_{jj} é um estado.

Definição 19: (setor das operações)
O **setor das operações** denotado por $\boldsymbol{S^M} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, de ordem $\boldsymbol{Y}$ - $\boldsymbol{X}$ + 1, tal que:
(1) $\forall\ r_{ij} \in \boldsymbol{S^M}$, $\boldsymbol{X} \leq i \leq \boldsymbol{Y} \wedge \boldsymbol{X} \leq j \leq \boldsymbol{Y}$
(2) $\forall r'_{ii} \in \boldsymbol{S^M}$, r'_{ii} é uma operação
(3) $\forall r'_{ij} \in \boldsymbol{S^M}$, se r'_{ij} é um evento então r'_{ii} é uma operação-causa do evento e r'_{jj} é uma operação-efeito do evento
(4) $\forall r'_{ij} \in \boldsymbol{S^M}$, se r'_{ij} é uma condição então r'_{ii} é uma operação e r'_{jj} é uma ação
(5) $\forall r'_{ij} \in \boldsymbol{S^M}$, se r'_{ij} é um vínculo de interação então
r_{ih} e r_{jk} são vínculos de implementação, sendo que:
h = k ou
r_{hk} é fluxo de dados ou fluxo de controle

Definição 20: (setor das implementações)
O **setor das implementações** denotado por $\boldsymbol{S^{IN}} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, com $\boldsymbol{Y}$ - $\boldsymbol{X}$ +1 linhas e $\boldsymbol{X}$ - 1 colunas, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^{IN}}$, $\boldsymbol{X} \leq i \leq \boldsymbol{Y} \wedge \boldsymbol{Y} < j \leq q \wedge i < j$
(2) se $r'_{ij} \neq$ "" então r'_{ij} é um vínculo de implementação.

Definição 21: (setor dos atributos processados)
O **setor dos atributos processados** denotado por $\boldsymbol{S^{AP}} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, com q - $\boldsymbol{Y}$ linhas e $\boldsymbol{X}$ - 1 colunas, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^{AL}}$, $\boldsymbol{Y} < i \leq q \wedge 1 \leq j < \boldsymbol{X} \wedge i > j$
(2) $\forall r'_{ij} \in \boldsymbol{S^{AL}}$, se r'_{ij} é um fluxo de dados ou um fluxo de objeto então r'_{ii} é um processo e r'_{jj} é uma classe.

Definição 22: (setor de projeto)
O **setor de projeto** denotado por $\boldsymbol{S^{PJ}} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, com q - $\boldsymbol{Y}$ linhas e $\boldsymbol{Y}$ - $\boldsymbol{X}$ + 1 colunas, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^{AL}}$, $\boldsymbol{Y} < i \leq q \wedge \boldsymbol{X} \leq j \leq \boldsymbol{Y} \wedge i > j$
(2) $\forall r'_{ij} \in \boldsymbol{S^{PJ}}$, $r'_{ij} =$ ""

Definição 23: (setor dos processos)
O **setor dos processos** denotado por $\boldsymbol{S^P} = [r'_{ij}]$ de $\boldsymbol{S}$ é uma submatriz da partição de $\boldsymbol{S}$, de

ordem q - ***Y***, tal que:
(1) $\forall r_{ij} \in \boldsymbol{S^P}$, $\boldsymbol{Y} < i \leq q \wedge \boldsymbol{Y} < j \leq q$
(2) $\forall r'_{ij} \in \boldsymbol{S^P}$, se r'_{ij} é um fluxo de dados ou um fluxo de controle então r'_{ii} é o processo de origem do fluxo e r'_{jj} é o processo de destino do fluxo.
(3) $\forall r'_{ij} \in \boldsymbol{S^P}$, se r'_{ij} é um vínculo de subprocesso então r'_{jj} é um processo e r'_{ii} é um subprocesso seu.

6.2.6 N²Chart

A matriz Square, além de ter características semelhantes às matrizes de adjacências, tem também origem no conceito de "N²Chart". Idealizado por Robert Lano em 1979 [LOY93], N²Chart é uma matriz quadrada que representa os elementos modelados por um DFD (figura 6.4). Processos, depósitos de dados e terminadores são colocados na diagonal principal. Os fluxos de dados são posicionados para representar a interface entre os elementos da diagonal principal.

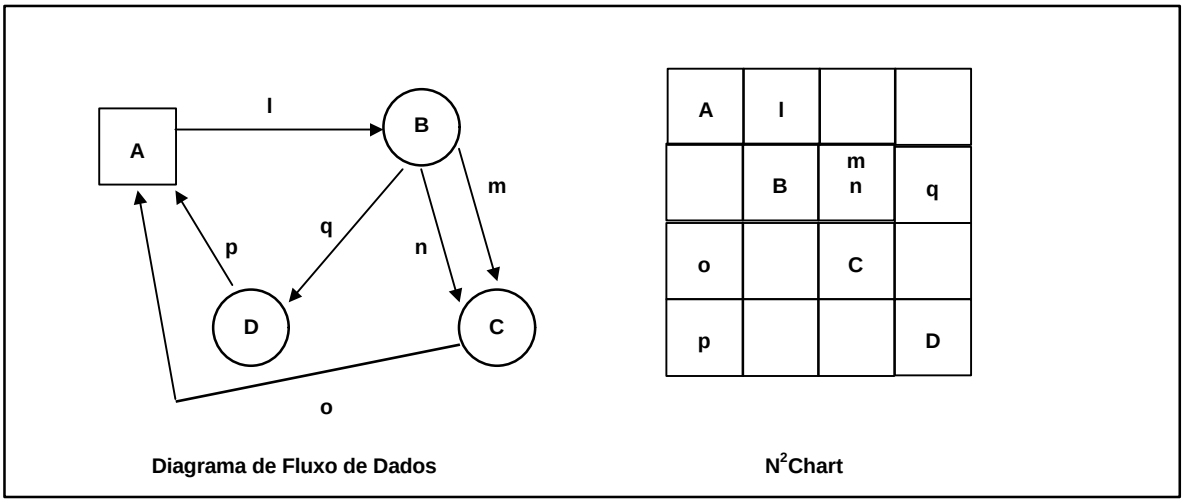


Figura 6.4. DFD e N²Chart [LOY93].

Loy e Stapp [LOY93] testaram o impacto de ferramentas gráficas (N²Chart e DFD) diferentes quanto a: (a) aprendizado, (b) interpretação e (c) criação. Cada componente de um grupo de 19 pessoas, sem conhecimento anterior das ferramentas, depois de instruído usou cada uma delas em problemas de igual complexidade. Tempos iguais foram destinados tanto para o aprendizado como para o uso. O grupo foi divido em dois. Um deles passou pela experiência primeiro com N²Chart e depois com DFD. O outro fez o contrário. As seguintes conclusões foram apresentadas

pelos autores:

- A utilização das duas técnicas parece não ter trazido confusão aos usuários, quanto à passagem de uma para outra. Os autores acham que o aprendizado das regras de construção do DFD fica fortalecido em combinação com o aprendizado de N^2Chart e vice-versa.
- Ao fim das duas horas do teste todas os participantes tinham um entendimento básico sólido dos mecanismos das duas técnicas.
- Os DFDs foram entendidos e interpretados um pouco mais facilmente. Deduzem os autores que isto se deve ao fato dos fluxos de dados serem mostrados explicitamente (a direção e as posições iniciais e finais) enquanto que N^2Charts requerem que as regras estruturais de comunicação entre os elementos estejam sempre presentes.
- N^2Charts foram criados com um pouco mais de facilidade. Imaginam os autores que a estrutura matricial beneficia muito a “carga criativa” pela formação da arquitetura do esquema. Os usuários têm uma idéia mais clara da localização dos componentes e da interface entre eles.
- Alguns participantes (não foi mencionada a proporção) que declararam preferir interpretar DFDs, na realidade interpretaram melhor N^2Charts.

Finalmente, Loy e Stapp [LOY93] defendem a validade, tanto para o aprendizado como para propósitos de aplicação, no contexto da modelagem de sistemas, da utilização das duas técnicas (N^2Chart e DFD) de forma integrada.

A matriz Square, comparada a N^2Chart, possui características de representação diferentes, conforme são descritas nesta dissertação, para viabilizar a tradução (para o formato matricial) dos diagramas da OMT: diagrama de Objetos, diagrama de Fluxo de Eventos, diagrama de Estados, além de diagrama de Fluxo de Dados. Além disso N^2Chart não se destina à consistência, mas à representação, embora seja óbvio que a idéia de consistência visual esteja presente tanto num N^2Chart como num diagrama de Fluxo de Dados ou em qualquer outra ferramenta gráfica.

6.3 Notação dos Elementos da OMT na Matriz Square

A seguir são apresentadas as notações dos elementos modelados na OMT. As figuras que seguem mostram sempre a notação Square à esquerda e a notação OMT à direita. Traços horizontais e verticais na matriz separam seus setores.

A matriz da figura 6.5 mostra uma classe que possui vínculos de responsabilidades com duas operações. À direita da classe estão as entradas e abaixo dela estão os retornos destes vínculos, respectivamente para cada operação.

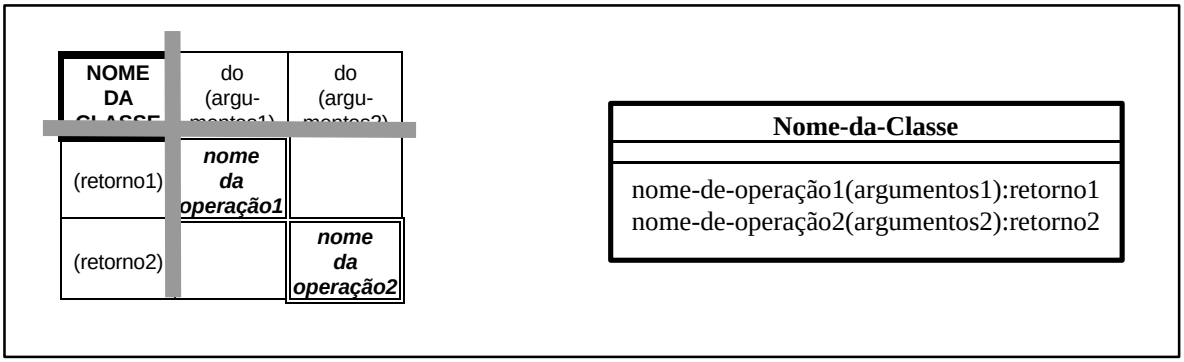


Figura 6.5. Classe de objetos e suas operações

- **Classe**. É representada numa célula (com borda grossa) da diagonal principal da matriz, no setor de objetos (figuras 6.5 e 6.6).
- **Operação**. É representada numa célula (com borda dupla) da diagonal principal da matriz, no setor das operações (figura 6.5). A assinatura correspondente à operação aparece nas suas interfaces de entrada (na sua coluna, com a indicação "**do**", representando uma entrada de vínculo de responsabilidade) e de saída (na sua linha, representando um retorno de vínculo de responsabilidade) vinculadas à classe.

A matriz da figura 6.6 mostra uma associação entre as classes "Classe1" e "Classe2" e entre as classes "Classe1" e "Classe3". Mostra também uma associação reflexiva da classe "Classe2" e uma associação como classe entre as classes "Classe3" e "Classe4", representada pela "Classe5".

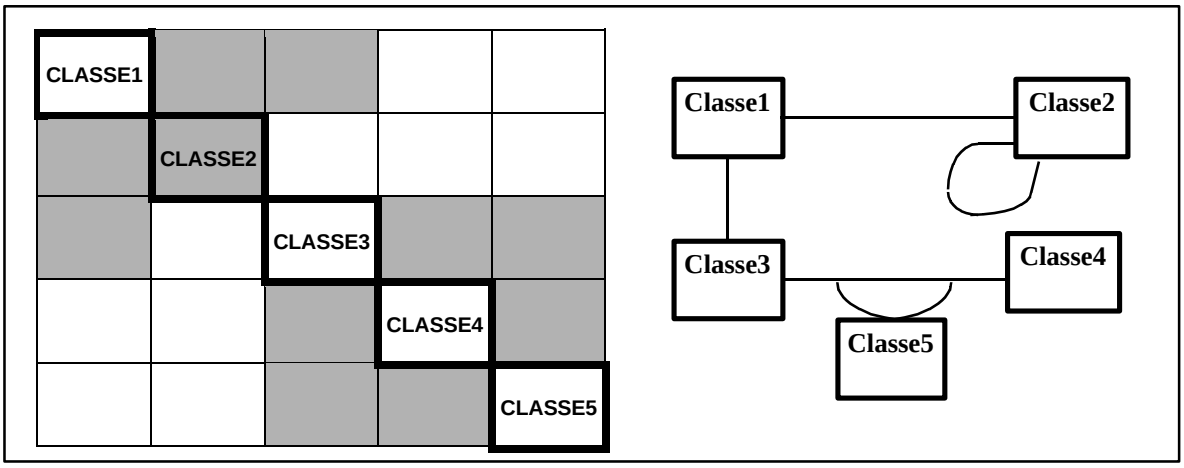


Figura 6.6. Associação, associação reflexiva e associação como classe.

- **Associação**. É representada no setor dos objetos por sombreamento das células da interface das classes relacionadas (figura 6.6). Como uma associação é bi-direcional, as células que amarram as duas classes são representadas exatamente da mesma forma. O nome da associação, papéis e cardinalidades (representados no diagrama) não são representados na matriz. Esta dissertação não trata da consistência de papéis e cardinalidades já que são informações restritas ao modelo de Objetos.
- **Associação ternária**. É representada como associações simples entre as classes participantes.
- **Associação reflexiva**. É representada na própria célula da classe, sendo esta célula sombreada (figura 6.6).
- **Associação como Classe**. Tem, na matriz, representação dupla: como associação e como classe (figura 6.6).

A matriz da figura 6.7 mostra uma generalização entre as classes “Classe1” e “Classe3” e uma generalização com classes não-disjuntas entre as classes “Classe2” e “Classe4”. Também é mostrada uma agregação, onde a “Classe3” é agregada e a “Classe4” é componente.

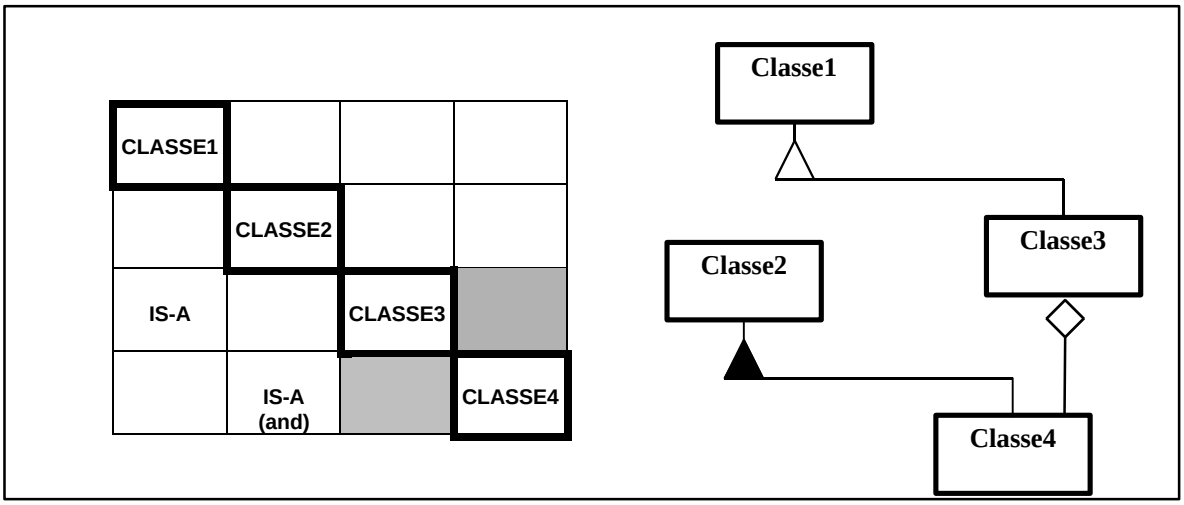


Figura 6.7. Agregação e generalização.

- **Agregação**. É representada no setor de objetos nas células das interfaces das classes relacionadas (figura 6.7). A célula da interface de saída (na linha) da classe agregada e de entrada (na coluna) da classe componente aparece com sombreamento. A célula da interface de saída (na linha) da classe componente e de entrada (na coluna) da classe agregada aparece com hachura horizontal.
- **Generalização**. É representada pela indicação **IS-A** no setor de objetos na célula da interface de entrada (na coluna) da classe mais genérica e de saída (na linha) da classe mais específica (figura 6.7).
- **Generalização com Classes Não-Disjuntas**. É representada como uma generalização (figura 6.7), acrescentando-se a expressão **and** entre parênteses, logo abaixo de **IS-A**.

Informações adicionais que devem ser dadas pelo modelador ao construir um diagrama de Fluxo de Eventos são: operação-causa e operação-efeito para cada evento, a não ser que uma das classes envolvidas represente uma interferência externa ao sistema e, assim, não são definidas operações para ela. É o caso da “Classe1”, na figura 6.8, sem operação-causa para “evento1” e sem operação-efeito para “evento4”. Nestes casos, “evento1” aparece na matriz da figura 6.8 entre “Classe1” e sua operação-efeito “operaçãoA” e “evento4” aparece entre a sua operação-causa “operaçãoB” e a “Classe1”. Por outro lado, “evento2” aparece entre a sua operação-causa “operaçãoA” e sua operação-efeito “operaçãoC” e “evento2” aparece entre a sua operação-causa “operaçãoC” e a sua operação-efeito “operaçãoB”. A matriz da figura 6.8 também mostra vínculos de evento entre as classes “Classe1” e “Classe2” e entre as classes “Classe2” e “Classe3”, através de células com borda grossa nas interfaces entre estas classes.

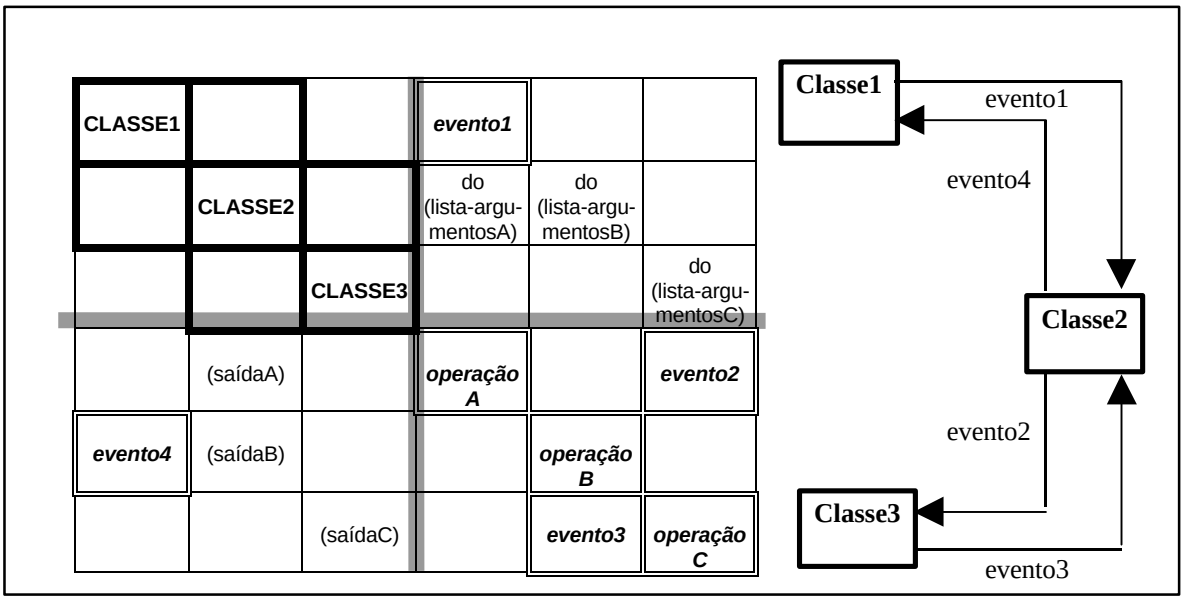


Figura 6.8. Eventos entre classes.

- **Evento**. Pode ser representado nos setores (a) das responsabilidades, (b) dos retornos ou (c) das operações:

(a) Sendo um evento de origem externa, é representado no setor das responsabilidades na célula da interface de saída (na linha) da classe de objetos de origem e da interface de entrada (na coluna) da operação-efeito da classe de objetos de destino (evento1 na figura 6.8).

(b) Sendo um evento de resposta ao ambiente externo, é representado no setor dos retornos na célula da interface de saída (na linha) da operação-causa da classe de objetos de origem e da interface de entrada (na coluna) da classe de objetos de destino (evento4 na figura 6.8).

(c) Sendo um evento interno, é representado no setor das operações na célula da interface de saída (na linha) da operação-causa e da interface de entrada (na coluna) da operação-efeito (evento2 e evento 3 na figura 6.8).

De qualquer forma, a célula onde o evento é representado tem borda dupla e, no setor dos objetos, a célula da interface de saída (na linha) da classe de objetos de origem e da interface de entrada (na coluna) da classe de objetos de destino tem borda grossa, representando um vínculo de evento entre elas.

A matriz da figura 6.9 mostra uma classe com três estados. O “estado1” tem um vínculo de atividade (“do”) com a “atividade”. A primeira transição é modelada pela seguinte seqüência:

"estado1", "nome do evento", "operação", "[condição1]", "estado2". A segunda transição é modelada por: "estado2", "[condição2]", "estado3". As ações associadas às transições desencadeiam-se em decorrência da execução da "operação", que é a operação-efeito do evento na primeira transição (a "ação1" fica vinculada à "operação" através da "[condição1]"), ou em decorrência da própria condição que acarreta a transição, no segundo caso (a "ação2" fica vinculada ao "estado2" através da "[condição2]").

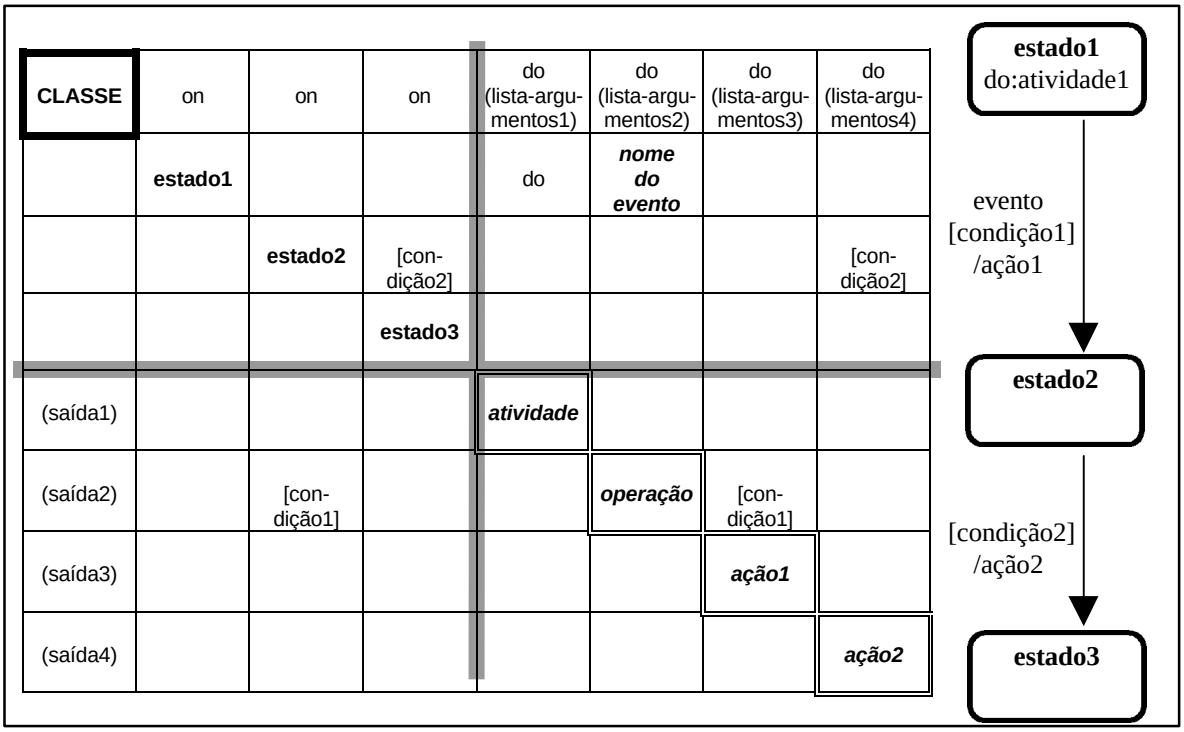


Figura 6.9. Estados e transições de estados.

- **Estado**. É representado por seu nome (é necessária a identificação dos estados pelo nome) no setor dos objetos numa célula da diagonal principal da matriz, tendo a indicação "**on**", representando um vínculo de estado, na célula da sua interface de entrada (na sua coluna) vinculada à classe correspondente (figura 6.9)**.** Os vínculos de estado "**on**" definem na linha de uma classe, então, quais são os estados válidos dos objetos desta classe.
- **Atividade**. É representada como uma operação no setor das operações (figura 6.9). A indicação "**do**", representando um vínculo de atividade, na célula da interface de saída (na linha) do estado e da interface de entrada (na coluna) da operação (atividade), da classe correspondente, significa que esta operação é uma atividade executada pelos objetos desta classe no estado em questão.
- **Transição**. É representada através de seus componentes: (a) evento, (b) condição com

evento, (c) condição sem evento e (d) ação.

(a) Um **evento** associado a uma transição de estados é indicado, no setor das responsabilidades (figura 6.9), pelo seu nome na célula da interface de saída (na linha) do estado anterior e da interface de entrada (na coluna) da operação-efeito do evento.

(b) Uma **condição** associada a uma transição com evento é representada entre colchetes (ou somente pelos colchetes na ausência da condição), no setor dos retornos (condição1 na figura 6.9), na célula da interface de saída (na linha) da operação-efeito do evento correspondente e da interface de entrada (na coluna) do estado posterior.

(c) Uma **condição** associada a uma transição sem evento é representada entre colchetes (ou somente pelos colchetes na ausência da condição), no setor dos objetos (condição2 na figura 6.9), na célula da interface de saída (na linha) do estado anterior e da interface de entrada (na coluna) do estado posterior.

(d) Uma **ação** associada a uma transição é representada como uma operação, no setor das operações. Uma condição entre colchetes (ou só colchetes na ausência desta) é representada na célula da interface de entrada (na coluna) da ação e da interface de saída (na coluna) da operação-efeito (no caso de presença de evento na transição, como na condição1 na figura 6.9) ou do estado anterior (no caso de ausência de evento na transição, como na condição2 na figura 6.9).

A matriz da figura 6.10 mostra um estado, chamado subestado, com vínculo de subestado com outro estado.

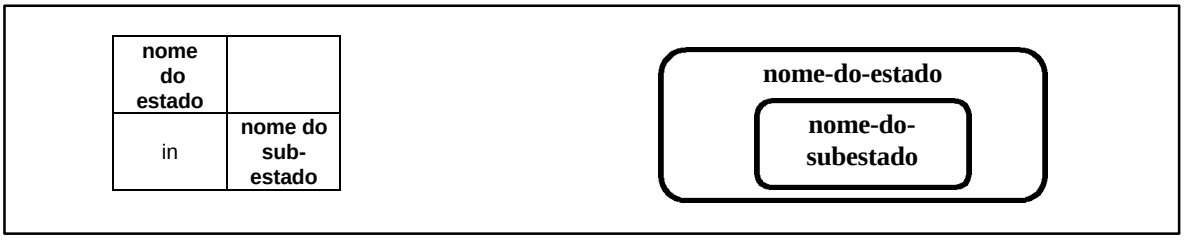


Figura 6.10. Superestado e subestado.

- **Subestado.** É modelado como um estado (figura 6.10) com a indicação "**in**", representando um vínculo de subestado, na célula da sua interface de saída (na sua linha) e da interface de entrada (na coluna) do estado (superestado).

A matriz da figura 6.11 mostra os estados "estado1" e "estado2" com vínculos de concorrência entre si.

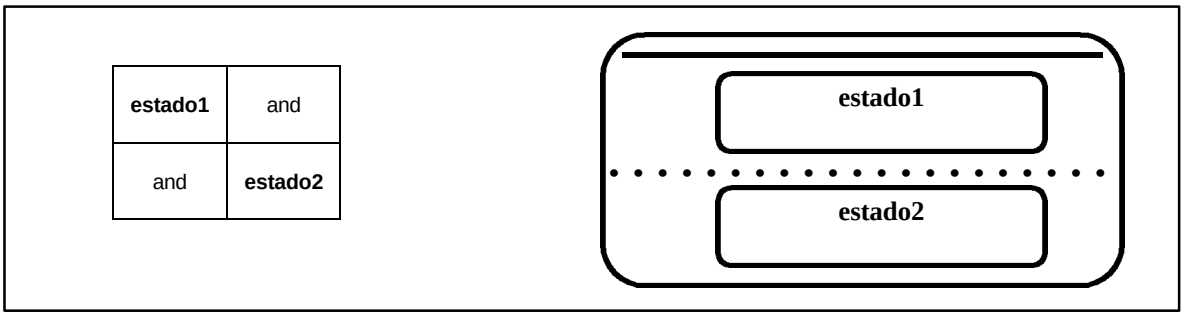


Figura 6.11. Estados concorrentes num único objeto.

- **Estados Concorrentes (em agregação)**. São representados como estados no setor dos objetos. A concorrência é representada em decorrência do relacionamento de agregação existente entre as respectivas classes de objetos.
- **Estados Concorrentes (num único objeto)**. São representados como estados no setor dos objetos. A concorrência é modelada pela indicação "**and**", representando um vínculo de concorrência, nas células das interfaces entre os estados concorrentes (figura 6.11).

A matriz da figura 6.12 mostra um vínculo de subprocesso entre os processos "subprocesso" e "processo".

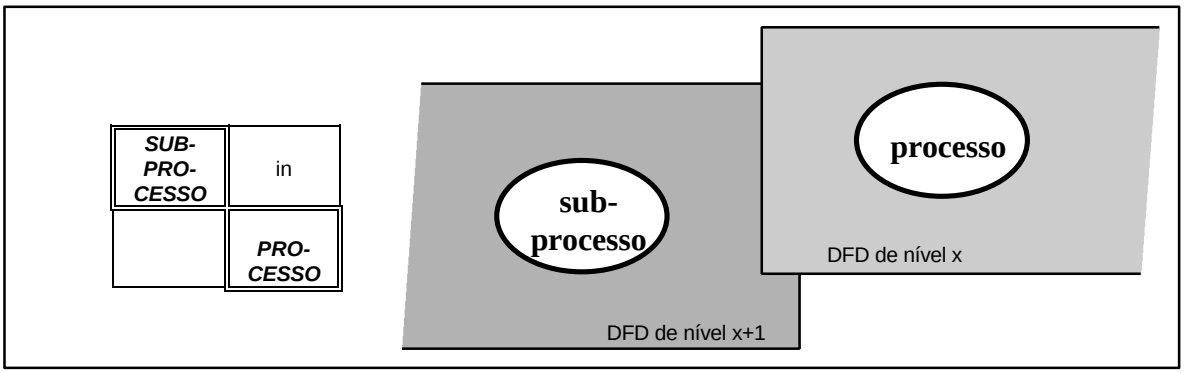


Figura 6.12. Processo e subprocesso.

- **Processo**. É representado numa célula (com borda dupla) da diagonal principal da matriz, no setor dos processos (figuras 6.12).
- **Subprocesso**. É modelado como um processo (figura 6.12), com a indicação "**in**", representando um vínculo de subprocesso, na célula da sua interface de saída (na sua linha) e da interface de entrada (na coluna) do superprocesso.

A matriz da figura 6.13 mostra a “Classe1” como um ator e a “Classe2” como um depósito de dados. O “processo1” tem como interface de entrada o fluxo de dados “lista1 de dados” que recebe de “Classe1” e como saída um fluxo de controle representado pela “[condição]” e um fluxo de objeto representado pela borda grossa na célula da coluna de “Classe2”. O “processo2” tem como entrada o fluxo de dados “lista2 de dados” e o fluxo de controle representado pela “[condição]”, ambos na sua coluna, e como saída, na sua linha, o fluxo de dados “lista3 de dados” que, por sua vez, é interface de entrada do “processo3”, estando em sua coluna. O “processo3” ainda possui como saída o fluxo de dados “lista4 de dados”, vinculado como entrada da “Classe1”.

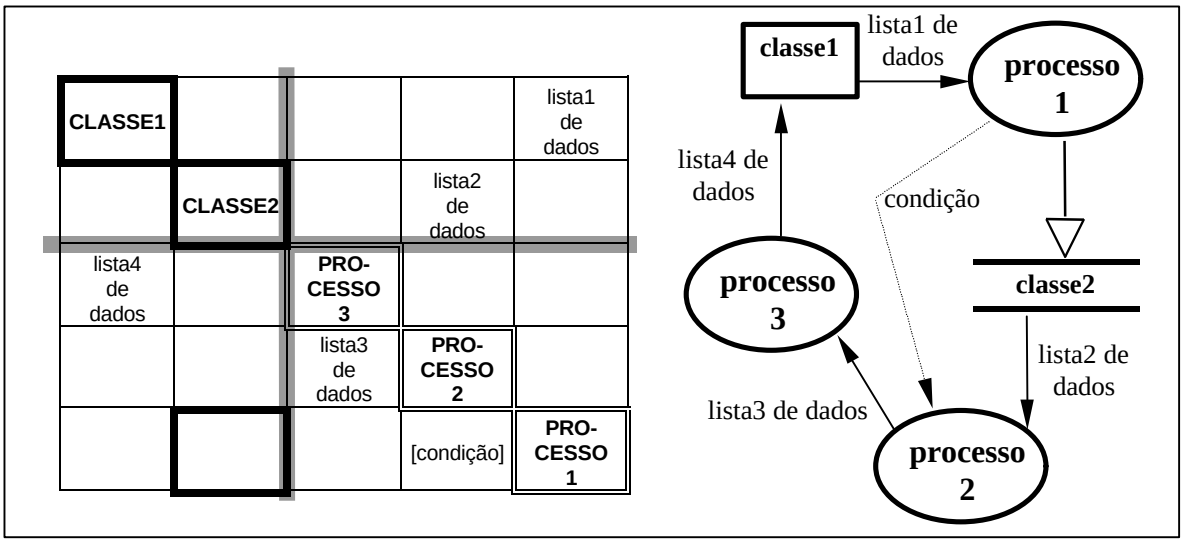


Figura 6.13. Depósito de dados, ator, fluxo de dados, fluxo de objetos e fluxo de controle.

- **Depósito de dados**. É representado como a classe de objetos (classe2 na figura 6.13).
- **Ator**. É representado como a classe de objetos (classe1 na figura 6.13).
- **Fluxo de dados**. É representado pela lista dos seus dados na célula da interface de saída (na linha) do elemento de origem e da interface de entrada (na coluna) do elemento destino. Originando-se em depósito de dados ou ator é representado no setor dos atributos lidos (lista1 e lista2 de dados na figura 6.13). Dirigido a depósito de dados ou ator é representado no setor dos atributos processados (lista4 de dados na figura 6.13). Localizado entre processos é representado no setor dos processos (lista3 de dados na figura 6.13).
- **Fluxo de objeto**, É representado pela borda grossa na célula da interface de saída (na linha) do processo de origem e da interface de entrada (na coluna) do depósito de

dados (classe) de destino (figura 6.13).

- **Fluxo de controle**. É representado no setor dos processos pela condição entre colchetes na célula da interface de saída (na linha) do processo de origem e da interface de entrada (na coluna) do processo de destino (figura 6.13).

A matriz da figura 6.14 mostra operações sendo implementadas por processos. As operações "operação1", com vínculo de responsabilidade com a "Classe1", e "operação2", com vínculo de responsabilidade com a "Classe2", são implementadas pelo mesmo processo "processo1", pois possuem vínculo de implementação ("in") com ele. Sendo assim, possuem vínculo de interação entre si (representado pelas células sombreadas). Por outro lado, o vínculo de interação entre as operações "operação2" e "operação3" existe por que são operações de classes diferentes (respectivamente "Classe2" e "Classe1") e são implementadas pelos processos "processo1" e "processo2", que possuem fluxo de dados (poderia ser de controle) entre si. Uma tarefa importante do modelador, complementando a construção do DFD, é a indicação das operações que são implementadas pelos processos.

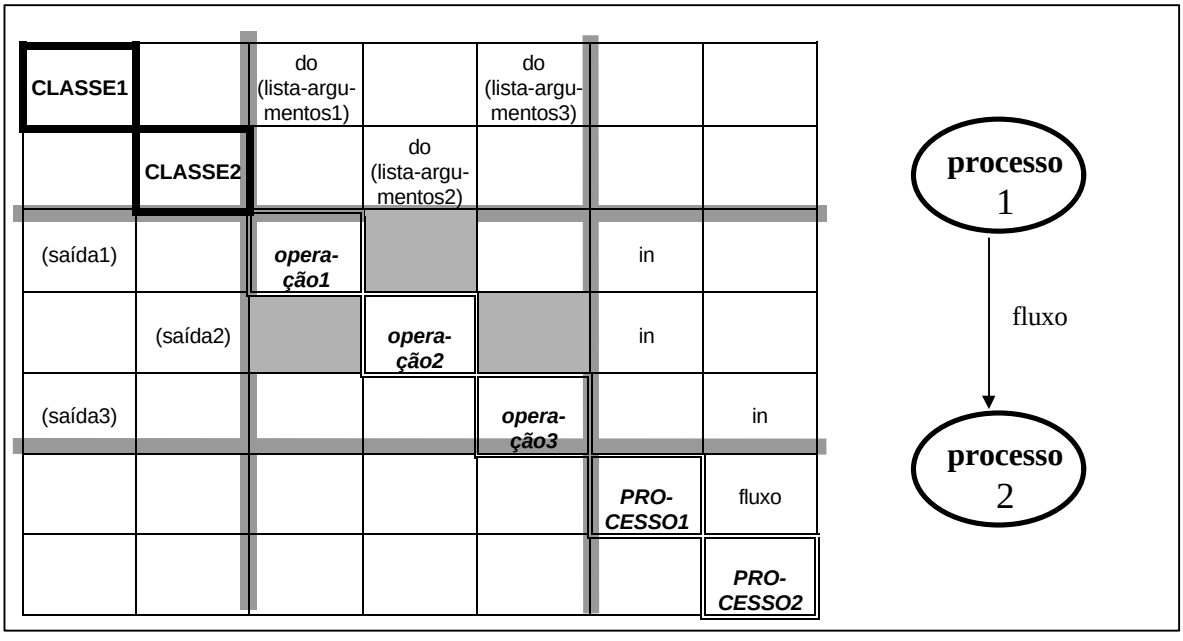


Figura 6.14. Implementação de operação por processo.

- **Implementação de operação por processo**. É modelada pela indicação "**in**", representando um vínculo de implementação, no setor das implementações (figura 6.14) na célula da interface de saída (na linha) da operação e da interface de entrada (na coluna) do processo-folha. Vínculos de interação entre operações de classes de objetos

diferentes são representados por células sombreadas em dois casos:

(a) Existindo fluxo de dados ou de controle entre os processos, é sombreada a célula da interface de saída (na linha) da operação implementada pelo processo de origem e da interface de entrada (na coluna) da operação implementada pelo processo de destino do fluxo (entre “operação2” e “operação3” na figura 6.14).

(b) São sombreadas as células das interfaces entre as operações implementadas pelo mesmo processo (entre “operação1” e “operação2” na figura 6.14).

6.4 Vantagens da Representação Unificada

Dado um grafo $G^{OMT}(V,E)$, a matriz Square $\boldsymbol{S} = [r_{ij}]$ gera um modelo matricial com as seguintes propriedades:

- **Propriedade 1**: Todos os elementos modelados nos diagramas são representados na Matriz Square.
- **Propriedade 2**: Cada um dos elementos modelados nos diagramas é representado numa única posição da Matriz Square, mesmo que estejam presentes em mais de um diagrama.

A propriedade 2 possui uma exceção: uma condição, componente de uma transição de estados com uma ação associada, tem representação dupla na coluna do estado posterior e na coluna da ação. Isto é necessário porque a condição determina dois fatos: a transição de um estado para outro e a execução da ação associada.

Classes de objetos, eventos e operações, que são referidos em mais de um diagrama, são representados na matriz Square atendendo a propriedade 2. Uma classe de objetos é modelada nos diagramas de Objeto e de Fluxo de Eventos e pode aparecer também como ator ou depósito de dados no diagrama de Fluxo de Dados. Um evento é modelado no diagrama de Fluxo de Eventos e pode aparecer também em transição de estado no diagrama de Estados (neste caso, na matriz, o evento representado na coluna da sua operação-efeito é indicado pelo nome, nesta mesma coluna, como componente da transição). Uma operação é definida para uma classe de objetos no diagrama de Objetos e pode aparecer também em transição de estado, como ações, ou nos estados, como atividades, no diagrama de Estados, além de ser indicada como operação-causa e/ou operação-efeito no diagrama de Eventos. Todas estas referências apontam para um

único local na matriz.

As propriedades 1 e 2 devem ser obedecidas pela ferramenta que apoie a Matriz Square. Através do Tradutor todo elemento modelado na atividade diagramática deve ser imediatamente traduzido e representado na matriz, obedecendo a propriedade 1. A Tabela de Elementos encarrega-se da representação unificada, principalmente quanto à propriedade 2. Classes de objetos, atores e depósitos de dados (assim como todos os outros elementos) têm referência única. No momento em que se modela um depósito de dados, por exemplo, é referenciada uma classe já existente ou é criada outra. No segundo caso, quando a nova classe for modelada nos diagramas em que ainda não existe, assumirá a mesma referência do depósito de dados na Tabela de Elementos. Este mesmo procedimento é adotado para eventos e operações. A Tabela de Elementos deve possuir uma entrada para cada elemento, indicando em que diagramas ele está modelado e em que posição da matriz Square está representado. A expressão “representação unificada” possui, portanto, significado duplo:

- o modelo matricial representa todos os elementos, sendo portanto um modelo único e
- cada um dos elementos tem uma única representação no modelo, isto é, aparece numa única posição da matriz.

A representação unificada favorece a consistência pela eliminação de ambigüidades devido à ausência de duplicidade de informação. Outro benefício é decorrente da existência de vínculos diretos ou indiretos, no modelo único, providas pelas interfaces de entrada e saída entre os elementos. Elementos de diagramas diferentes ficam vinculados. Estes vínculos estabelecem padrões, auxiliando a verificação visual de coerência entre os modelos, e são usados também para a aplicação das regras de consistência.

Num processo de modelagem onde diversos elementos aparecem em diagramas diferentes, o modelador tem de exercitar mecanismos de abstração para limitar a informação presente de forma a verificar a consistência, em termos de duplicação, ambigüidade, ausência e erros sintáticos e semânticos. O problema é que, na modelagem, não é utilizado um único diagrama. Ao contrário, são gerados vários e geralmente extensos e complexos. Naturalmente o problema cresce em sistemas maiores.

Na prática é difícil comparar fragmentos de diagramas. Ocorrem algumas complicações, como por exemplo: como selecionar o que interessa? como operacionalizar esta seleção? como definir o que é importante? A análise direta sobre os diagramas é trabalhosa, exigindo esforço e tempo.

Com o modelo matricial (com representação unificada), um filtro adequado sobre a Matriz Square proporciona o mecanismo de abstração eficiente para a seleção dos elementos que interessam dentro de um determinado contexto. Padrões e regras de consistência possibilitam o auxílio necessário para a análise que objetiva a integração das informações modeladas.

6.5 Conclusões

A Matriz Square é uma ferramenta gráfica que gera o modelo matricial assim como os diagramas de Objeto, de Fluxo de Evento, de Estado e de Fluxo de Dados geram os modelos de Objeto, Dinâmico e Funcional. Como todos os elementos, que são representados nos diagramas, são também representados na matriz, viabiliza-se a integração das informações num único modelo. Isto permite que o modelador verifique as vinculações existentes entre os elementos representados, estabelecidas diretamente nos diagramas ou indiretamente através da interação entre os modelos.

Como já foi visto, os modelos da análise devem ser intuitivos, expressivos e precisos e, para isto, não podem ser ambíguos, devem ter poder de abstração e devem ser consistentes [HAY91]. A ambigüidade é evitada, com o uso da Matriz Square, porque sua notação exige interpretação única com a posição dos elementos-vértice na diagonal principal e dos elementos-arco como interface entre eles, definindo rigidamente entradas, pelas colunas, e saídas, pelas linhas. O poder de abstração é conseguido pelo uso de filtros sobre a matriz como será visto no capítulo 8 e a consistência é permitida no modelo matricial pela integração de elementos desvinculados totalmente em virtude da sua localização original em diagramas diferentes.

A presença de arcos e vértices nos diagramas estabelece intuitividade, enquanto a disposição destes elementos na estrutura matricial exige maior atenção do modelador, diminuindo a carga intuitiva da leitura. Esta dificuldade é contornada pelo fato dos diagramas continuarem tendo papel importante na modelagem e toda a entrada de dados ser feita ainda por eles. A construção do modelo matricial deve ocorrer de forma automatizada e paralela à atividade diagramática através de uma ferramenta que apoie a Matriz Square, conforme é discutido no próximo capítulo.

7 UMA FERRAMENTA DE APOIO À MATRIZ SQUARE

> "*Como em vários campos, os projetos de software crescem sobremaneira em tamanho e complexidade, sendo grande a necessidade de análise e geração de software com o auxílio do computador*."
> Fischer [FIS90]

7.1 Introdução

Neste capítulo são apresentadas as características mínimas necessárias de uma ferramenta que apoie a Matriz Square para a integração dos modelos da metodologia OMT. Na seção 7.2 são apresentadas as premissas da ferramenta e a sua arquitetura.

7.2 Fundamentos da Ferramenta

A metodologia OMT possui uma ferramenta de modelagem de objetos específica, construída por um de seus autores. A OMTool é um editor gráfico para construção de diagramas de Objetos. Ela permite a criação, recuperação, edição, arquivamento e impressão de diagramas de Objetos, em interação com o usuário [RUM91]. A ferramenta OMTool gera um modelo gráfico, que armazena a figura que está desenhada na tela, e um modelo lógico, que armazena o significado subjacente da figura, isto é, as classes, os atributos, as operações e seus relacionamentos.

Uma ferramenta de apoio à Matriz Square é caracterizada aqui para apoiar a fase de análise da metodologia OMT, com o objetivo principal de integrar seus modelos numa descrição única e coerente que permita a verificação de consistência entre os modelos.

7.2.1 Premissas Básicas da Ferramenta

- A ferramenta deve suportar a metodologia OMT na fase de análise, levando em conta o que foi apresentado na seção 5.2 (Considerações para Integração).
- O suporte à metodologia OMT direciona-se, fundamentalmente, ao problema da integração dos modelos concebidos na fase de análise. A verificação de consistência

neste sentido toma como diretrizes básicas o que foi abordado na seção 5.3 (Integração).

- A diagramação e a verificação de consistência intra-modelos são tarefas que devem ser suportadas pela ferramenta mas não fazem parte do objetivo desta dissertação. Elas são ocasionalmente consideradas neste trabalho apenas por serem inerentes a atividade de modelagem, como é o caso da OMT.
- A ferramenta não deve prescindir de um dicionário de dados que lhe dê suporte. É de grande importância naqueles aspectos da consistência que envolvem comparações de dados a nível de atributos e argumentos. A especificação e estruturação de um dicionário de dados também não faz parte desta dissertação.
- A ferramenta adota os conceitos do paradigma de orientação a objetos, nos moldes como é apresentado por Rumbaugh et al [RUM91], para que seja efetivo o suporte dado pela ferramenta à metodologia OMT.

7.2.2 Arquitetura da Ferramenta

A ferramenta de apoio à Matriz Square, como já foi mencionado, deve suportar a metodologia OMT, consistindo seus modelos de forma a integrá-los, na fase de análise, numa descrição coerente. Para que isso aconteça, paralelamente à atividade diagramática deve ser construído um repositório de informações sobre os elementos que vão sendo modelados. Deste repositório devem fazer parte um dicionário de dados, a Matriz Square e uma tabela de elementos.

O dicionário de dados deve fazer referência aos dados modelados, incluindo classes de objetos, seus atributos, argumentos de suas operações, argumentos de eventos e dados de fluxos de dados.

A matriz Square armazena os elementos dos diagramas representando-os em posições únicas. Forma um modelo também único do sistema no formato matricial. Suas características, notação e uso constituem o tema central desta dissertação. Ela incorpora o núcleo fundamental da ferramenta caracterizada aqui e que deve construir o modelo matricial possibilitando a integração dos modelos gerados pelos diagramas de forma visual ou através de regras de consistência.

A tabela de elementos deve vincular os elementos representados na matriz com a representação nos diagramas e complementar o modelo matricial em termos de repositório de informações.

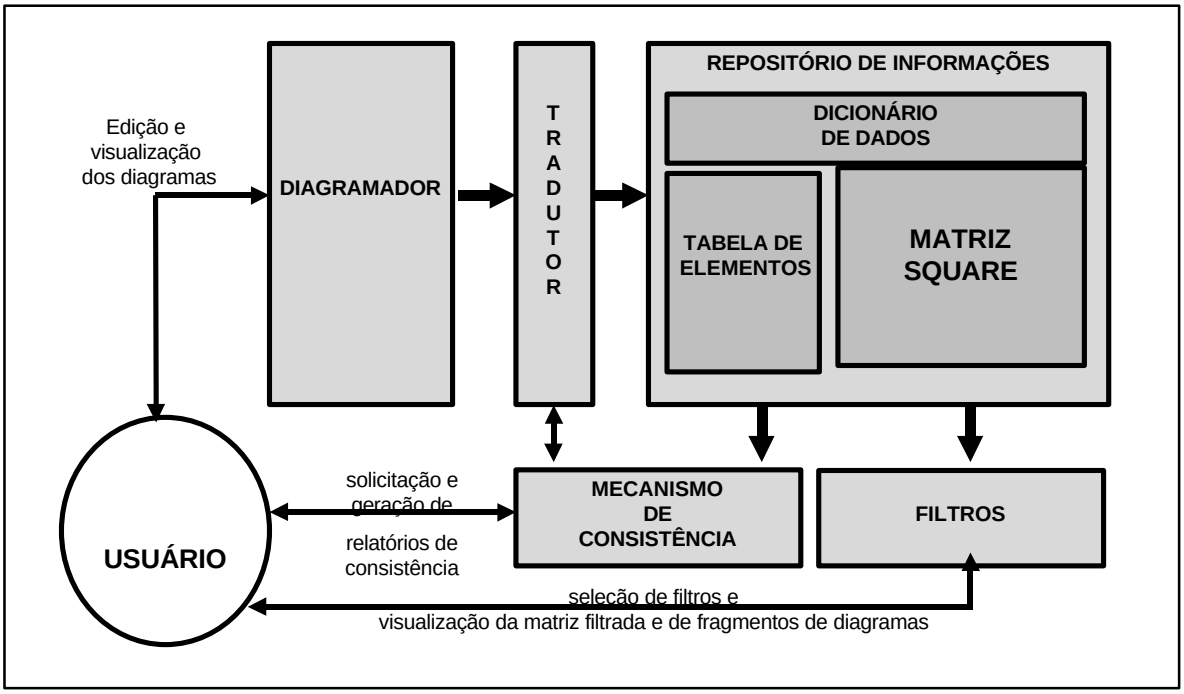


Figura 7.1. Arquitetura básica de uma ferramenta de apoio à Matriz Square.

A seguir são descritos os componentes de uma ferramenta de apoio à Matriz Square, conforme é mostrado na figura 7.1.

- **Diagramador**. Deve possuir um editor gráfico e uma interface gráfica para suportar a atividade diagramática. Através dele o modelador constrói os diagramas de Objetos, de Fluxo de Eventos, de Estados e de Fluxo de Dados. O Diagramador deve permitir a entrada de todas as informações que a ferramenta irá gerenciar. O gerenciamento destes dados, incluindo visualização, armazenagem e edição de diagramas, é um assunto bastante abordado em diversos trabalhos e por ferramentas CASE. Por isso não será desenvolvido nesta dissertação.
- **Tradutor**. Este componente deve traduzir aquilo que é informado através do Diagramador e armazenar estas informações traduzidas no Repositório de Informações. Sua principal função é traduzir a notação dos diagramas para a notação do modelo matricial. Dependendo da implementação da ferramenta, um maior ou menor grau de consistência pode ser feito durante este processo. Para isto deve haver a interferência do Mecanismo de Consistência. É interessante que as informações traduzidas sejam inseridas no Repositório de Informações com um nível de consistência satisfatório. Nem tão elevado, limitando em demasia a liberdade de criação do modelador, nem tão

baixo que traga dificuldades indesejadas para futuras correções.

- **Repositório de Informações**. Deve constitui-se do dicionário de dados, da Tabela de Elementos e da Matriz Square. Os mecanismos que acessam o Repositório de Informações, dependendo da necessidade, devem buscar informações nos três componentes que o compõe.
- **Dicionário de Dados**. Deve conter referências a classes de objetos, a seus atributos, a argumentos de suas operações, a argumentos de eventos e a dados de fluxos de dados. Embora seja reconhecida a sua importância, como já foi mencionado, não é objetivo desta dissertação a especificação do Dicionário de Dados. Interessa à ferramenta que seja possível a comparação de itens de dados com o objetivo de concluir se referem-se ao mesmo objeto de dado ou não. O gerenciamento do dicionário deve prever a recuperação: (1) de aliases referentes a um mesmo objeto de dado e (2) dos componentes de um dado agregado. Desse modo um dado pesquisado pode ser comparado com sinônimos (no primeiro caso) ou com componentes (no segundo caso) de um dado encontrado e que deve ser alvo de comparação.
- **Tabela de Elementos**. Deve conter informações sobre os elementos modelados principalmente para vincular suas posições no modelo matricial e nos diagramas. Os elementos devem ser referenciados através de identificadores únicos para possibilitar acesso, troca de nome, inclusão, eliminação e outras alterações. O objetivo é não perder a individualidade de cada elemento e seus vínculos e, principalmente, possibilitar a representação única no modelo matricial.
- **Matriz Square**. Deve armazenar os elementos no formato matricial, como é definido no capítulo 6, com o objetivo de integrar os modelos num local único, permitindo a consistência visual sob este novo formato e otimizando a aplicação de regras de consistência a partir dos vínculos que a matriz possibilita.
- **Mecanismo de Consistência**. Baseado em regras de consistência definidas no capítulo 9, deve realizar a verificação sobre o modelo matricial, apoiado pelo Dicionário de Dados e pela Tabela de Elementos. Deve gerar relatórios a pedido do modelador ou no momento de sua interferência no processo de tradução, desde que necessário.
- **Filtros**. A matriz Square, como um modelo único, assume o compromisso de representar todos os elementos modelados. Por isto, a medida que o sistema torna-se complexo, perde poder de comunicação em razão do seu tamanho. Por outro lado, o objetivo não é a visualização completa da matriz. O benefício da consistência visual só

é alcançado pela seleção adequada dos fragmentos de informações presentes no modelo matricial. Esse processo de seleção deve ser feito através dos filtros. A ferramenta deve possibilitar que, a qualquer momento, o modelador os utilize e obtenha a matriz filtrada e fragmentos correspondentes dos diagramas. Os elementos, assim selecionados e combinados num modelo único, atendem o princípio da simplificação da complexidade, permitindo a verificação de consistência visual.

A seguir são descritas as funções mínimas exigidas para uma ferramenta de apoio à Matriz Square para serem utilizadas pelo usuário:

- **Edição e visualização dos diagramas**. Estas funções devem resultar na construção dos modelos de Objetos, Dinâmico e Funcional. Permitem a edição dos diagramas de Objetos, de Fluxo de Eventos, de Estados e de Fluxo de Dados. Além da visualização dos diagramas, aqui é feita, portanto, toda a entrada de dados necessária para o cumprimento das etapas listadas na seção 5.2 (Considerações para Integração) para a fase de análise.
- **Solicitação e geração de relatórios de consistência**. A geração de relatórios deve ocorrer: (a) quando o processo de tradução, ao utilizar o mecanismo de consistência, detecta algo que deve ser comunicado ou (b) quando o usuário solicita uma verificação de consistência sobre o repositório de informações. No primeiro caso, o usuário deve tomar alguma decisão, principalmente sobre o elemento que está sendo modelado. No segundo caso, o usuário usará o relatório para fundamentar suas decisões futuras em relação à modelagem como um todo.
- **Seleção de filtros e visualização da matriz filtrada e de fragmentos de diagramas**. Estas funções devem consistir em selecionar filtros adequados e obter visões parciais da matriz Square e dos diagramas. Deste modo, o usuário terá subsídios para tomar decisões sobre problemas localizados de inconsistência, se detectados. Ou, ainda, verificar a ausência de elementos importantes perdidos possivelmente em razão da complexidade do sistema.

7.3 Conclusões

Uma ferramenta de apoie a Matriz Square, suportando a metodologia OMT, deve construir o modelo matricial de modo automatizado. Esta construção deve ocorrer paralelamente à atividade diagramática, quando o modelador usa a ferramenta para a entrada dos elementos nos diagramas de Objeto, de Fluxo de Eventos, de Estados e de Fluxo de Dados. As informações que são geradas desta forma devem ser incluídas num repositório formado por um dicionário de dados, uma tabela chamada Tabela de Elementos e pela própria Matriz Square.

A ferramenta deve possibilitar que o modelador verifique a consistência da modelagem pela aplicação de regras de consistência, apresentadas no capítulo 9, e a qualquer momento visualize a matriz, concentrando-se em detalhes que serão importantes conforme o contexto de interesse do momento, usando filtros conforme é discutido no próximo capítulo.

8 A INTEGRAÇÃO DOS MODELOS DA OMT ATRAVÉS DE FILTROS

> "*Abstração é o exame seletivo de determinados aspectos de um problema. O objetivo da abstração é isolar os aspectos que sejam importantes para algum propósito e suprimir os que não o forem.*"
> Rumbaugh et al [RUM91]

8.1 Introdução

Este capítulo discute modelo matricial visto sob filtros. A seção 8.2 define os filtros selecionáveis pelo usuário para obter fragmentos da Matriz Square para consistência visual. A seção 8.3 apresenta um ensaio sobre padrões de modelagem na matriz, mostrando como as conexões explícitas são verificadas através de filtros. A seção 8.4 discute o uso dos filtros. A seção 8.5 mostra a aplicação de cenários.

8.2 Filtros

O modelo matricial é visualizado através de filtros, segundo o esquema mostrado na figura 8.1. A matriz filtrada é uma submatriz da matriz Square, sempre contendo a diagonal principal. O filtro escolhido pelo usuário seleciona os elementos-vértice (da diagonal principal) e estes determinam os elementos-arco que estarão na matriz filtrada.

A matriz filtrada descarta alguns segmentos de linhas e colunas (da matriz Square) que ficam ocultos. Este mecanismo é bastante utilizado em planilhas eletrônicas para "esconder" informações não relevantes num determinado contexto de visualização de uma planilha.

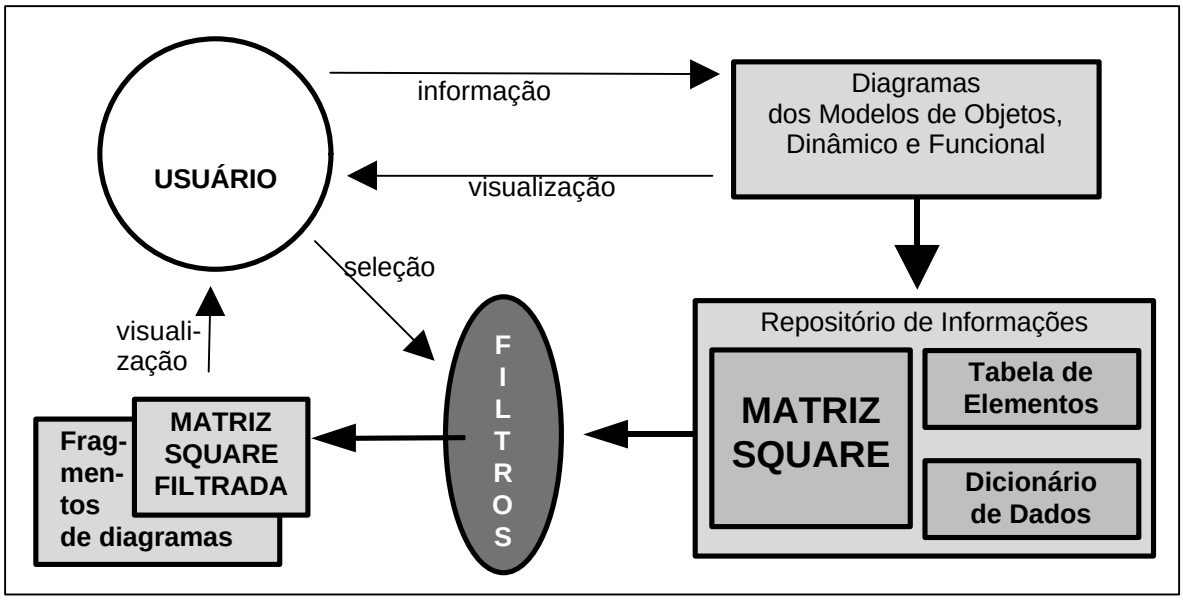


Figura 8.1. O papel dos filtros na visualização do modelo matricial.

Dependendo dos elementos visíveis na matriz filtrada resultante, é permitida também a visualização de fragmentos dos diagramas correspondentes. Este processo é viabilizado pela Tabela de Elementos que mapeia a posição dos elementos na matriz e nos diagramas. O uso de filtros gera um processo de abstração que auxilia a construção dos modelos. Dependendo do filtro selecionado, o modelador pode verificar a coerência entre informações originadas em modelos diferentes, realizando uma consistência visual.

Existem sete tipos de filtros: Filtro por responsabilidade, por estado, por comunicação, por evento, por condição, por processo e por ligação. Cada um deles privilegia uma classe de conexão explícita, onde a participação das operações é fundamental para a integração. Os filtros são apresentados a seguir, considerando os seguintes itens:

- **elementos filtrantes primários**: determinados pelo modelador ao selecionar o filtro;
- **elementos filtrantes secundários**: filtrados a partir dos primários;
- **setores pesquisados**;
- **elemento integrador**: elementos-arco que a partir dos quais são encontrados os elementos-vértice filtrados e
- **elementos-vértice filtrados**: determinam, juntamente com os elementos-vértice filtrantes, a diagonal principal da matriz filtrada.

Para as definições 24 a 30, a seguir, considera-se uma matriz square $\boldsymbol{S} = [r_{ij}]$, de ordem q, com

setores $\boldsymbol{S^O}$, $\boldsymbol{S^M}$, $\boldsymbol{S^P}$, $\boldsymbol{S^{RP}}$, $\boldsymbol{S^{AL}}$, $\boldsymbol{S^{RT}}$, $\boldsymbol{S^{AP}}$ e $\boldsymbol{S^{IN}}$ e com diagonal principal **d(*X,Y*)**, associada a um grafo $G^{OMT}(V,E)$.

8.2.1 Filtro por Responsabilidade

Um filtro por responsabilidade tem como elemento filtrante primário uma classe de objetos e seu objetivo é detectar as conexões de responsabilidade desta classe no modelo.

A matriz filtrada por Responsabilidades mostra em relação a uma classe de objetos (ver tabela 8.1, com as referências):

- as operações (a) sob sua responsabilidade;
- as operações (b, e) que são ativadas pelos eventos que estimula;
- os processos (c, d) com os quais atores ou depósitos de dados correspondentes têm fluxo de dados;
- os processos (f) que implementam suas operações e
- as outras classes de objetos (g) responsabilizadas pelas operações que atendem as necessidades daquela especificada.

elemento filtrante		**setor pesquisado**	**elemento integrador**	**elemento-vértice filtrado**
primário	**secundário**			
classe de objetos		das responsabilidades	v.de responsab.	operação (a)
			evento	operação (b)
		dos atributos lidos	fluxo de dados	processo (c)
		dos atributos processados	fluxo de dados	processo (d)
	operação (a)	das operações	evento	operação (e)
		das implementações	v.de implement.	processo (f)
	operação (b, e)	das responsabilidades	v.de responsab.	classe (g)

Tabela 8.1. Pesquisa referente ao filtro por responsabilidade.

Definição 24: (matriz filtrada por responsabilidade)
Uma matriz ***S*** **filtrada por responsabilidade** para uma classe $r_{ff} = \boldsymbol{c_f} \in C$, é uma matriz square denotada por $\boldsymbol{S_r} = [r_{i'j'}]$, de ordem q', com diagonal principal **d(*X,Y*)'** tal que:
se i = f
ou $r_{fi} \in \boldsymbol{S^{RP}}$ ou $r_{fi} \in \boldsymbol{S^{AL}}$ ou $r_{if} \in \boldsymbol{S^{AP}}$ são interfaces
ou $r_{ji} \in \boldsymbol{S^M}$ é interface e $r_{fj} \in \boldsymbol{S^{RP}}$ é entrada de vínculo de responsabilidade
ou $r_{ij} \in \boldsymbol{S^{RP}}$ é entrada de vínculo de responsabilidade e $r_{jj} \in$ **d(*X,Y*)'**
ou $r_{ji} \in \boldsymbol{S^{IN}}$ é vínculo de implementação e $r_{fj} \in \boldsymbol{S^{RP}}$ é entrada de vínculo de responsabilidade
então $r_{ii} \in$ **d(*X,Y*)'**

8.2.2 Filtro por Ligação

Um filtro por ligação tem como elemento filtrante primário uma classe de objetos e seu objetivo é detectar as conexões de ligação desta classe no modelo.

A matriz filtrada por ligação mostra em relação a uma classe de objetos (ver tabela 8.2, com as referências):

- as operações (a) sob sua responsabilidade e que atendem eventos estimulados por outras classes de objetos (operações-efeito);
- as classes de objetos (b, d) que estimulam esses eventos;
- as operações (c) que ativam esses eventos (operações-causa) e
- os processos (e) que implementam as operações-causa

elemento filtrante		**setor pesquisado**	**elemento integrador**	**elemento-vértice filtrado**
primário	**secundário**			
classe de objetos		das responsabilidades e das operações	v.de responsab. e evento	operação (a)(*)
	operação (a)	das responsabilidades	evento	classe (b)
		das operações		operação (c)
	operação (c)	das responsabilidades	v.de responsab.	classe (d)
		das implementações	v.de implement.	processo (e)

Tabela 8.2. Pesquisa referente ao filtro por ligação.
(*) não são filtradas operações que não tem evento que as ative diretamente

Definição 25: (matriz filtrada por ligação)
Uma matriz ***S*** **filtrada por ligação** para a classe $r_{ff} = \boldsymbol{c_f} \in O$, é uma matriz square denotada por $\boldsymbol{S_{col}} = [r_{i'j'}]$, de ordem q', com diagonal principal **d(*X,Y*)'** tal que:
se i = f ou
$r_{fg} \in \boldsymbol{S^{RP}}$ é entrada de vínculo de responsabilidade, r_{kg} é evento, $k \leq \boldsymbol{Y}$, tal que
(a) i = g ou
(b) i = k ou
(c) $r_{ik} \in \boldsymbol{S^{RP}}$ é entrada de vínculo de responsabilidade ou
(d) $r_{ki} \in \boldsymbol{S^{IN}}$ é vínculo de implementação
então $r_{ii} \in$ **d(*X,Y*)'**

8.2.3 Filtro por Estado

Um filtro por estado tem como elemento filtrante primário um estado de uma classe de objetos e seu objetivo é detectar as conexões deste estado no modelo.

A matriz filtrada por estado mostra em relação a um estado de uma classe de objetos (ver tabela 8.3, com as referências):

- outros estados (a, e) que possuem transições de ou para o estado especificado;
- as operações (b) que são executadas nestas transições;
- a atividade (c) que é executada neste estado;
- as ações (d) que são ativadas pelas transições consideradas e
- os processos (f) que implementam as operações envolvidas, associadas e acionadas.

elemento filtrante		**setor pesquisado**	**elemento integrador**	**elemento-vértice filtrado**
primário	**secundário**			
estado		dos objetos	condição	estado (a)
		das responsabilidades	evento	operação (b)
			condição	
			v.de responsab.	operação (c)
	operação (b)	das operações	condição	operação (d)
		dos retornos		estado (e)
	operação (c)	das implementações	v.de implement.	processo (f)
	operação (d)			

Tabela 8.3. Pesquisa referente ao filtro por estado.

Definição 26: (matriz filtrada por estado)
Uma matriz ***S*** **filtrada por estado** para o estado $r_{ff} = \boldsymbol{s_f} \in$ S, válido para uma classe $r_{gg} = \boldsymbol{c_g} \in$ C, é uma matriz square denotada por $\boldsymbol{S_s} = [r_{i'j'}]$, de ordem q', com diagonal principal **d(*X,Y*)'** tal que:
se i = f ou i = g
ou r_{fi} é interface, tal que $r_{fi} \in \boldsymbol{S^O}$ ou $r_{fi} \in \boldsymbol{S^{RP}}$
ou $r_{ji} \in \boldsymbol{S^M}$ é condição e $r_{fj} \in \boldsymbol{S^{RP}}$ é interface
ou $r_{ji} \in \boldsymbol{S^{IN}}$ é vínculo de implementação e $r_{jj} \in$ **d(*X,Y*)'**
ou $r_{ji} \in \boldsymbol{S^{RT}}$ é condição e r_{fj} é interface, tal que $r_{fj} \in \boldsymbol{S^O}$ ou $r_{fj} \in \boldsymbol{S^{RP}}$
então $r_{ii} \in$ **d(*X,Y*)'**

8.2.4 Filtro por Comunicação

Um filtro por comunicação tem como elemento filtrante primário um dado constante do dicionário de dados e seu objetivo é detectar as conexões deste dado no modelo.

A matriz filtrada por comunicação mostra em relação a um dado (ver tabela 8.4, com as referências):

- a classe de objetos (a) que contém o dado como seu atributo;
- operações que ativam (b) ou são ativadas (c,g) por eventos que transmitam o dado como atributo entre objetos de classes que geram (f) e consomem (d) o dado;
- classes (d, n) e suas operações (e) que recebam o dado como argumento, sem vínculo direto com evento ou que sejam responsáveis pelas operações-causa e efeito;
- processos que originam (l, j) e que recebam (i, m) fluxo de dados, onde o dado apareça, com depósitos de dados ou atores (h, k).

elemento filtrante		**setor pesquisado**	**elemento integrador**	**elemento-vértice filtrado**
primário	**secundário**			
dado		(*)		classe (a)
		das responsabilidades	v.de responsab. e evento	operação (b) operação (c)
			v.de responsab.	classe (d) operação (e)
		das responsabilidades e das operações	v.de responsab. e evento	classe (f) operação (g)
		dos atributos lidos	fluxo de dados	classe (h) processo (i)
		dos atributos processados		processo (j) classe (k)
		dos processos		processo (l) processo (m)
	operação (b)	das responsabilidades	v.de responsab.	classe (n)
	operação (c)			

Tabela 8.4. Pesquisa referente ao filtro por comunicação.
(*) esta pesquisa é feita no dicionário de dados e não na matriz

Definição 27: (matriz filtrada por comunicação)

Um **item de dado**, denotado por ***dd*** é um atributo de classe ou um elemento da entrada de um vínculo de responsabilidade ou um dado de fluxo de dados

Uma matriz ***S*** **filtrada por comunicação** para o item de dado dd_f, é uma matriz square denotada por $S_a = [r_{i'j'}]$, de ordem q', com diagonal principal **d(*X,Y*)'** tal que:

se r_{ii} é classe com atributos $\{a_1, ..., a_n\}$ e $dd_f \in \{a_1, ..., a_n\}$

ou $r_{jk} \in S^{RP}$ é entrada A de vínculo de responsabilidade e $dd_f \in$ A, tal que
i = j ou i = k

ou r_{ij} é evento e $r_{jj} \in$ **d(*X,Y*)'**, tal que $r_{ij} \in S^{RP}$ ou $r_{ij} \in S^{M}$

ou r_{jk} é fluxo de dados com dados $\{d_1, ..., d_n\}$ e $dd_f \in \{d_1, ..., d_n\}$, tal que
(a) i = j ou i = k
(b) $r_{jk} \in S^{AL}$ ou $r_{jk} \in S^{AP}$

ou $r_{jk} \in S^{P}$ é fluxo de dados com dados $\{d_1, ..., d_n\}$ e $dd_f \in \{d_1, ..., d_n\}$, tal que
i = j ou i = k

então $r_{ii} \in$ **d(*X,Y*)'**

8.2.5 Filtro por Evento

Um filtro por evento tem como elementos filtrantes primários um evento e as classes de objetos de origem e de destino. Seu objetivo é detectar as conexões deste evento no modelo.

A matriz filtrada por evento mostra em relação a um evento (ver tabela 8.5, com as referências):

- as operações envolvidas (operações-causa e efeito) ativando (b, c) ou sendo ativada (a, d) pelo evento;
- os estados (e, f) afetados pelo evento e
- os processos (g) que implementam as operações-causa e efeito .

elemento filtrante		**setor pesquisado**	**elemento integrador**	**elemento-vértice filtrado**
primário	**secundário**			
evento		das responsabilidades	v.de responsab. e evento	operação (a)
		dos retornos	retorno de v.de responsab. e evento	operação (b)
classe-origem		das operações	evento	operação (c)
classe-destino		das operações		operação (d)
	operação (a, d)	das responsabilidades	evento ou condição	estado (e)
		dos retornos	condição	estado (f)
	operação (b, c)	das implementações	v.de implement.	processo (g)

Tabela 8.5. Pesquisa referente ao filtro por evento.

Definição 28: (matriz filtrada por evento)
Uma matriz ***S*** **filtrada por evento** para o evento $r_{fg} = \boldsymbol{ev_{fg}} \in$ EV, da classe $r_{oo} = \boldsymbol{c_o} \in$ C para a classe $r_{dd} = \boldsymbol{c_d} \in$ C, é uma matriz square denotada por $\boldsymbol{S_{ev}} = [r_{i'j'}]$, de ordem q', com diagonal principal **d(*X*,*Y*)'** tal que:
se i = o ou i = d
ou o = f e i = g
ou d = g e i = f
ou o ≠ f e d ≠ g, tal que i = f ou i = g
ou $r_{ig} \in \boldsymbol{S^{RP}}$ é interface
ou $r_{gi} \in \boldsymbol{S^{RT}}$ é interface
ou $r_{ji} \in \boldsymbol{S^{IN}}$ é vínculo de implementação e $r_{jj} \in$ **d(*X*,*Y*)'**
então $r_{ii} \in$ **d(*X*,*Y*)'**

8.2.6 Filtro por Condição

Um filtro por condição tem como elemento filtrante primário uma condição e seu objetivo é detectar as conexões desta condição no modelo.

A matriz filtrada por condição mostra em relação a uma condição (ver tabela 8.6, com as referências):

- as operações (a, b, d, e) e os processos (i, j), que tornam a condição verdadeira ou a tomam como uma pré-condição;
- os estados (c, f, g, h) afetados pela condição e
- os processos (k) que implementam as operações que definem a condição como verdadeira ou não (a) ou que dependem dela (b, d, e).

elemento filtrante		**setor pesquisado**	**elemento integrador**	**elemento-vértice filtrado**
primário	**secundário**			
condição		das operações	condição	operação (a) operação (b)
		das responsabilidades		estado (c) operação (d)
		dos retornos		operação (e) estado (f)
		dos objetos		estado (g) estado (h)
		dos processos	fluxo de controle	processo (i) processo (j)
	operação (a)	das implementações	v.de implement.	processo (k)
	operação (b, d, e)			

Tabela 8.6. Pesquisa referente ao filtro por condição.

Definição 29: (matriz filtrada por condição)
Uma matriz ***S*** **filtrada por condição** para a condição r_{fg} = ***cn*** ∈ TR, é uma matriz square denotada por $\boldsymbol{S}_{\boldsymbol{cn}}$= $[r_{i'j'}]$, de ordem q', com diagonal principal **d(*X*,*Y*)'** tal que:
se i = f ou i = g
ou r_{jk} é fluxo de controle ***cn'*** = ***cn***, tal que i = j ou i = k
ou $r_{ji} \in \boldsymbol{S}^{\boldsymbol{IN}}$ é vínculo de implementação e $r_{jj} \in$ **d(*X*,*Y*)'**
então $r_{ii} \in$ **d(*X*,*Y*)'**

8.2.7 Filtro por Processo

Um filtro por processo tem como elemento filtrante primário um processo e seu objetivo é detectar as conexões deste processo no modelo.

A matriz filtrada por processo mostra em relação a um processo (ver tabela 8.7, com as referências):

- os processos (a) com os quais o filtrante é ligado por algum tipo de fluxo;
- as operações (c) que o processo filtrante implementa;
- as classes de objetos (d, e) que representam atores ou depósitos de dados com as quais o processo filtrante é ligado por fluxo de dados;
- as classes de objetos que estimulam (f) ou recebem (h) eventos que têm as operações implementadas como operações-causa ou efeito;
- os estados (g, i) com transição causada pelas operações implementadas e
- as operações (j, k) vinculadas às operações implementadas por evento ou condição.

elemento filtrante		**setor pesquisado**	**vínculo pesquisado**	**elemento-vértice filtrado**
primário	**secundário**			
processo		das processos	fluxos	processo (a)
			v.de subprocesso	subprocesso* (b)
		das implementações		operação (c)
		dos atributos lidos	fluxos de dados	classe (d)
		dos atributos processados		classe (e)
	operação	das responsabilidades	evento	classe (f)
			evento ou condição	estado (g)
		dos retornos	evento	classe (h)
			condição	estado (i)
		das operações	evento ou condição	operação-causa (j) operação-efeito (k)
	subprocesso	repete o mesma pesquisa como processo		

Tabela 8.7. Pesquisa referente ao filtro por processo.
(*) os subprocessos são usados como elementos-filtrantes secundários, sem fazer parte da matriz filtrada

Definição 30: (matriz filtrada por processo)
Uma matriz ***S* filtrada por processo** para o processo $r_{ff} = \boldsymbol{p_f} \in P$, é uma matriz square denotada por $\boldsymbol{S_p} = [r_{i'j'}]$, de ordem q', com diagonal principal **d(*X*,*Y*)'** tal que:
se i = f
ou g = f ou r_{gf} é vínculo de subprocesso, tal que:
(a) $r_{gi} \in \boldsymbol{S^{AP}}$ é interface ou
(b) $r_{ig} \in \boldsymbol{S^{AL}}$ é interface ou
(c) $r_{ig} \in \boldsymbol{S^{IN}}$ é vínculo de implementação ou
(d) r_{ij} é interface ou r_{ji} é interface, tal que
i < ***Y*** e $r_{jg} \in \boldsymbol{S^{IN}}$ é vínculo de implementação ou
(e) $r_{ig} \in \boldsymbol{S^{P}}$ é fluxo de dados ou fluxo de controle ou
(f) $r_{gi} \in \boldsymbol{S^{P}}$ é fluxo de dados ou fluxo de controle
então $r_{ii} \in$ **d(*X*,*Y*)'**

8.3 Padrões

Um padrão é aquilo que serve de base ou norma para a avaliação de qualidade ou quantidade; é medida, modelo, exemplo, protótipo, arquétipo, gabarito [FER86]. Padrões são utilizados na música, na arquitetura, na psicologia, na lingüística, na indústria e em diversas outras áreas do conhecimento humano.

Padrões são utilizados em metodologias que adotam o paradigma orientado a objetos porque trabalham com elementos como classes e seus objetos e para encontrar padrões entre estes elementos necessitamos observar como eles se relacionam [COA92].

Padrões podem ser reconhecidos pelo modelador no modelo matricial, no momento em que algum filtro é selecionado. Eles colocam em evidência as conexões explícitas para a integração. A escolha de um filtro possibilita a visualização de padrões. A presença de um padrão, numa matriz filtrada, é um indício de existência de conexão e de integração e, portanto, de consistência. Um padrão incompleto ou a sua ausência indica a ausência de uma conexão e, portanto, uma provável inconsistência.

8.3.1 Um Ensaio sobre Padrões no Modelo Matricial

Durante a preparação desta dissertação foi possível exercitar o uso do modelo matricial através de alguns estudos de casos. Além daqueles apresentados no capítulo 10, outros à nível de pequenos testes também foram examinados. Durante estes exercícios foi possível verificar alguns padrões que se repetem. Eles são apresentados aqui na forma de um ensaio.

Um estudo mais aprofundado sobre padrões no modelo matricial é uma tarefa que exige uma pesquisa demorada que deve ser aplicada sobre um grande número de casos. É provável que aplicações com características semelhantes produzam padrões semelhantes. Neste ensaio são considerados dez padrões, apresentados a seguir. Eles tiveram como ponto de partida as classes de conexões explícitas, onde as operações têm papel fundamental para a integração.

Neste ensaio são encontrados dois padrões correspondentes às conexões de Estado, de Controle por Condição e de Processo e apenas um padrão para as outras classes de conexões consideradas.

O padrão duplo, quando encontrado, é decorrente da presença forte que a dimensão funcional tem nestes casos. A existência ou não de evento na transição entre estados é decisiva, pois a modelagem altera-se significativamente de uma ou de outra maneira. A seguir são descritos os padrões encontrados.

- **Padrão de Responsabilidade**: Um padrão de responsabilidade de uma classe ("Classe1", na figura 8.2) mostra as operações que esta classe executa, os eventos que estimula e os processos em que participa.

Os vínculos de responsabilidade mostram que a "Classe1", na figura 8.2, executa as operações "operação1" e "operação2". A localização do "evento" na linha da "operação2" indica que esta é sua operação-causa e, em decorrência, que é estimulado pela "Classe1", pois esta é responsável pela "operação2". A "operação 3" (sem classe responsável visível no filtro) apresenta-se como operação-efeito do "evento", pois está em sua coluna. A figura 8.2 mostra, também, que a "Classe1" participa dos processos "Processo1" e "Processo2" pois suas operações são implementadas por eles. O vínculo de interação (célula sombreada onde está o "evento") entre a "operação2" e a "operação3" indica que o "Processo2" (implementador de "operação2") recebe um fluxo de dados ou de controle vindo de um processo (não mostrado) que implementa a "operação3" e, portanto, justifica o "evento" entre as operações.

CLASSE 1	do ( )	do (args)			fluxo2 de dados
(saída1)	***operação 1***			in	
(saída2)		***operação 2***	***evento***		in
			operação 3		
fluxo1 de dados				***PROCESSO 1***	
					PROCESSO 2

Figura 8.2. Padrão de responsabilidade da "Classe1"

- **Padrão de Ligação**: O padrão de ligação de uma classe de objetos (“Classe1”, na figura 8.3) mostra os eventos que atende, as operações que executa para atender esses eventos, as operações de outras classes de objetos que causam esses eventos, os relacionamentos estabelecidos com essas outras classes e os processos que implementam essas operações.

A figura 8.3 mostra que as classes “Classe1” e “Classe2” possuem um vínculo de evento (célula de borda grossa) entre elas, devido ao “evento” solicitado pela “Classe2”, através da “operação2” (operação-causa). O “evento” é atendido pela “Classe1” através da “operação1” (operação-efeito). Existe coerência entre a localização do vínculo de evento (da segunda para a primeira classe) e a localização do evento (da operação da segunda para a operação da primeira classe). Também existe coerência entre o vínculo de evento e a associação (células sombreadas) entre as classes, assim como entre o evento e o vínculo de interação (célula sombreada) entre as operações. A associação (modelada no diagrama de Objetos), o evento (modelado nos diagramas de Fluxo de Eventos), o vínculo de evento (decorrente da modelagem do evento) e o vínculo de interação (decorrente da implementação das operações pelo mesmo processo ou, como deve ser o caso presente, por processos ligados por fluxo de dados ou de controle) confirmam a colaboração existente entre as classes representadas e que se origina de informações modeladas em diferentes modelos.

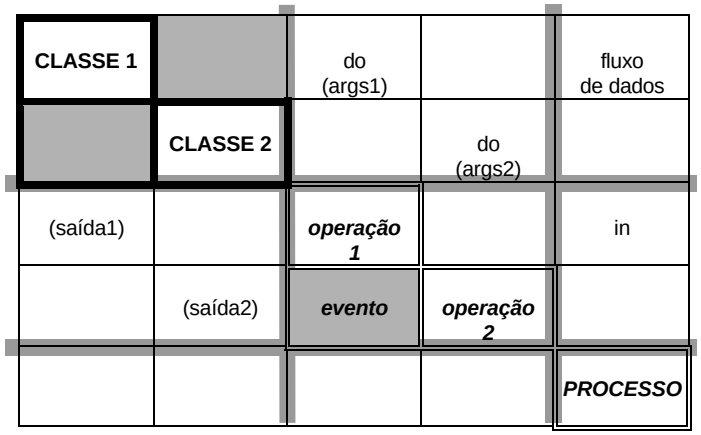


Figura 8.3. Padrão de Ligação da “Classe1”

- **Padrão I de Estado**: Um padrão I de estado (“estado x”, na figura 8.4) mostra os estados com os quais este tem transição por evento, as operações envolvidas, os processos que implementam estas operações e os dados alterados.

CLASSE	on	on	on	do (args1)	do (args2		
	estado anterior			evento 1			
		estado x			evento 2		
			estado posterior				
(saída1)		[condição1]		***operação 1***		in	
(saída2)			[condição2]		***operação 2***		in
fluxo de dados 1						***PROCESSO 1***	
fluxo de dados 2							***PROCESSO 2***

Figura 8.4. Padrão I de estado para “estado x”

Este padrão de estado tem a presença de eventos e na figura 8.4 visualizam-se os estados “estado anterior”, “estado x” e “estado posterior” com vínculo de estado (“on”) com a “Classe”. Há uma transição entre “estado anterior” e “estado x”, que é mostrada pela seguinte leitura: “estado anterior”, “evento1”, “operação1”, “[condição1]” e “estado x”. E há uma transição entre “estado x” e “estado posterior”, definida assim: “estado x”, “evento2”, “operação2”, “[condição2]” e “estado posterior”. A primeira transação seria marcada por “evento1 [condição1]” e a segunda por “evento2 [condição2]” no diagrama de Estados.

As operações mostradas na figura 8.4 são operações-efeito dos eventos modelados. Os vínculos de implementação mostram que estas operações são implementadas pelos processos “Processo1” e “Processo2”. Os fluxos de dados e os vínculos de responsabilidade mostram os dados considerados nestas transições. Os tipos de dados de “(saída1)” e de “(saída2)” devem ser coerentes com os dados de “fluxo de dados1” e de “fluxo de dados2”, respectivamente.

- **Padrão II de Estado**: Um padrão II de estado (“estado x”, na figura 8.5) mostra os outros estados com os quais este tem transição por condição, as atividades e ações envolvidas e os processos que implementam estas operações.

CLASSE	on	on	on	do (args1)	do (args2)			
	estado anterior	[condição1]						
		estado x	[condição2]	do	[condição2]			
			estado posterior					
				atividade			in	
					ação			in
						PROCESSO 1	[condição1]	
							PROCESSO 2	[condição2]
								PROCESSO 3

Figura 8.5. Padrão II de estado para o “estado x”.

Na figura 8.5 aparecem os mesmos estados e a mesma classe da figura 8.4 para possibilitar a comparação entre os dois padrões de estado. No padrão II não há eventos interferindo nas transições entre os estados. A leitura da matriz quanto aos estados (“estado anterior”, “[condição1]”, “estado x”, “[condição2]”, “estado posterior”) mostra as transições do “estado x”. Aqui não temos operações-efeito já que não temos eventos. A participação da “Classe” em processos não pode ser verificada, portanto, pela implementação de operações-efeito. Por outro lado, a participação da “Classe” em processos pode ser constatada pelas implementações de operações do tipo atividade e ação, que podem ser modeladas em diagramas de Estado. Notar o vínculo de atividade (“do”) entre o “estado x” e a “atividade” e a “[condição2]” entre “estado x” e a “ação”.

No caso da figura 8.5 podem ser comparados os fluxos de controle, já que estão representados com as condições das transições. Tomando-se como exemplo a “[condição1]” pode-se fazer a seguinte consistência: se a “[condição1]” desencadeia a passagem dos objetos da “Classe” para o “estado x” e esses objetos neste estado executam a operação definida como “atividade”, então a “[condição1]” é uma pré-condição para esta operação; se a “atividade” é implementada pelo “Processo2” então a mesma “[condição1]” deve compor o fluxo de controle modelado com destino neste processo. Constata-se que realmente entre “Processo1” e “Processo2” está

representado o fluxo de controle “[condição1]”. Ficam desta maneira consistidos a transição, a atividade associada, a implementação e o fluxo de controle.

- **Padrão de Comunicação**: Um padrão de comunicação de um dado (“dado x” na figura 8.6) mostra os eventos, às operações, os fluxos de dados e as classes de objetos que têm algo a ver com a transmissão, a recepção, a transformação ou o conhecimento do dado em questão e os relacionamentos que estabelece entre as classes de objetos envolvidas.

A figura 8.6 mostra que a “Classe1” é responsável pela “operação1” que é operação-causa de “evento”. O destino do “evento” é a “Classe2”, ativando a operação-efeito “operação2”. Isto é descrito pelo vínculo de evento (célula com borda grossa entre as classes), pelos vínculos de responsabilidade (“do...”) e pela posição do “evento” na linha de “operação1” e na coluna de “operação2”. Os argumentos de um evento estão presentes no vínculo de responsabilidade da sua operação-efeito, portanto o argumento de “evento” é “dado x”. Este evento, deste modo, pode ser correlacionado ao fluxo de dados (“dado x”) representado entre os processos “Processo1” e “Processo2”; o “evento” entre as operações “operação1” e “operação2” mostra que há uma passagem de informação entre elas (seu argumento, “dado x”) e como “Processo1” implementa “operação1” e “Processo2” implementa “operação2”, a mesma passagem de informação (fluxo de dados “dado x”) é coerentemente modelada entre estes processos.

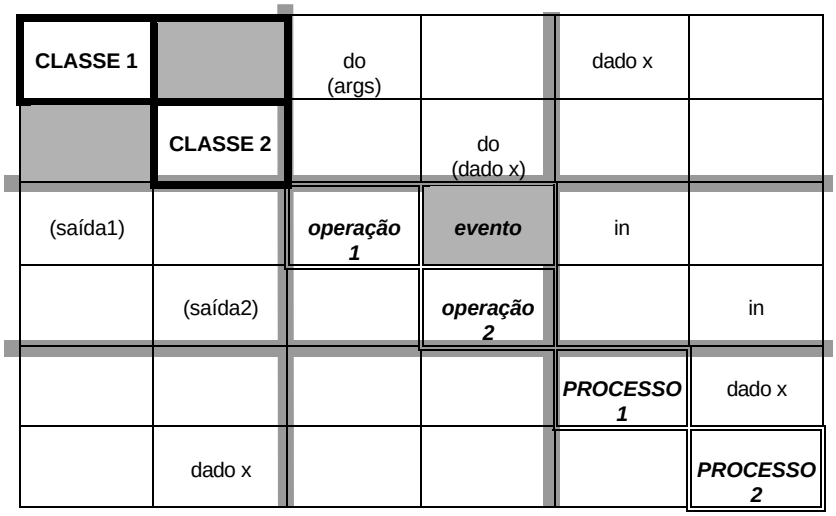


Figura 8.6. Padrão de comunicação para “dado x”.
(dado x pode aparecer em um ou mais de um dos fluxos de dados representados nos setores de atributos lidos e de atributos processados)

Na figura 8.6, a leitura definida pelos fluxos de dados é a seguinte: “Classe1”, “dado x”, “Processo1”, “dado x”, “Processo2”, “dado x”, “Classe2”. Assim, a entrada de “Processo1” é o fluxo de dados “dado x” e como os argumentos de “operação1”, implementada por este processo, é “args” (ver vínculo de responsabilidade entre “Classe1” e “operação1”) pode-se consistir “args” com “dado x”, com o auxílio de um dicionário de dados. De forma semelhante pode-se consistir o tipo de dado de retorno “(saída2)” da “operação2” com o tipo de dado de “dado x”, já que “operação2” é implementada por “Processo2”.

- **Padrão de Evento**: Um padrão de evento (“evento x”, na figura 8.7) mostra as ligações que se estabelecem entre objetos de classes diferentes, a seqüência de operações destes objetos e de processos que implementam estas operações e a sua interferência nas transições de estado destes objetos.

Na figura 8.7 observa-se a “Classe1” com os estados “estado” e “estado posterior” (ver vínculos de estado “on”) e a “Classe2”. Entre estas classes existe um vínculo de evento mostrado pela célula com borda grossa. A “operação1” da “Classe1” é operação-causa do “evento x” e a “operação2” da “Classe2” é operação-efeito do mesmo evento. Estas operações são implementadas respectivamente pelos processos “Processo1” e “Processo2” (ver vínculos de implementação).

Através da figura 8.7 é possível verificar que o “evento x” do diagrama de Eventos enviado para a “Classe2” participa de uma transição entre estados desta classe, conforme a leitura “estado”, “evento”, “operação2”, “[condição]” e “estado posterior”. Verifica-se, assim, que os objetos da “Classe2” passam para o “estado posterior” ao ser executada a “operação2”. Observa-se que o “Processo2” tem a participação da “Classe2” com a execução da “operação2”, implementada por ele, e pode-se ver que os objetos da “Classe2” participam do “Processo2” quando estão no estado chamado “estado posterior”.

Na figura 8.7, ainda é possível fazer a correlação do “fluxo” (entre “Processo1” e “Processo2”) com o “evento x” (entre “operação1” e “operação2”), em razão de que o primeiro vincula os processos e o último vincula as operações e constatam-se vínculos de implementação entre eles (operações e processos).

Nas figura 8.6 e 8.7, assim como na 8.3 onde foi feita a referência, nota-se que associações

(células sombreadas entre as classes), eventos (identificados pelo nome), vínculos de evento (células de borda grossa entre as classes) e vínculos de interação (células sombreadas entre as operações) confirmam colaborações entre as classes envolvidas.

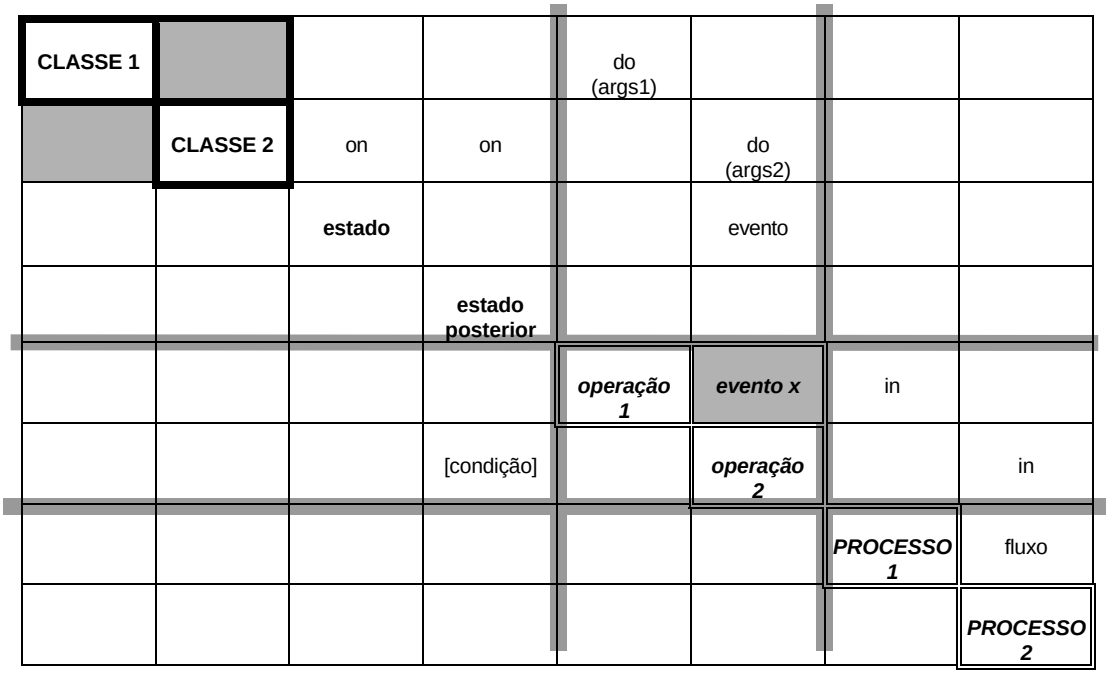


Figura 8.7. Padrão de evento para “evento x”.

- **Padrão I de Condição**: Um padrão I de condição (“condição x”, na figura 8.8) mostra a seqüência das operações dos objetos de uma classe e dos processos que implementam estas operações, numa transição de estados desses objetos causada por um evento e efetivada pela condição considerada.

Na figura 8.8 aparece o padrão I de Condição, com um evento participando da transição entre estados lá mostrada, e na figura 8.9 aparece o padrão II de Condição, sem a participação de um evento.

Na figura 8.8 observa-se a “Classe” com os estados “estado1” e “estado2” e a transição entre eles que é feita por “evento [condição x] / ação” como deveria aparecer no diagrama de Estados. Na matriz, a leitura é feita da seguinte maneira: “estado1”, “evento”, “operação”, “[condição x]”, “estado2”, indicando a transição e sendo a “operação” tida como operação-efeito do “evento”. A “[condição x]” entre as operações “operação” e “ação” estabelece a seqüência de execução entre elas, isto é, a transição entre os estados (efetuada pela execução da “operação”) acarreta a execução da “ação”.

Na figura 8.8, assim como no padrão de Evento da figura 8.7, também se observa a participação da “Classe” em processos, no caso nos processos “Processo1” e “Processo2”, estando os objetos desta classe no estado “estado2”. Isto é verificado pelos vínculos de implementação entre as operações e os processos. Esta integração possibilita que a condição “[condição x]” seja correlacionada ao fluxo de controle “[condição x]”, isto é, a “ação” só é executada se a condição for verdadeira e, do mesmo modo, o “Processo2” só é executado se a condição do fluxo de controle que chega até ele for verdadeira.

CLASSE	on	on	do (args1)	do (args2)		
	estado 1		evento			
		estado 2				
(saída1)		[condição x]	***operação***	[condição x]	in	
(saída2)				***ação***		in
					PROCESSO 1	[condição x]
						PROCESSO 2

Figura 8.8. Padrão I de condição para “condição x”

- **Padrão II de Condição**: Um padrão II de condição (“condição x”, na figura 8.9) mostra a sua interferência na transição de estados dos objetos de uma classe, determinando a seqüência das operações (atividades) que possam estar associadas aos estados desses objetos e dos processos que implementam essas operações.

Na figura 8.9 observa-se a “Classe” com os estados “estado1” e “estado2” que tem transição feita pela “[condição x]”, sem participação de evento. Este é um padrão semelhante ao padrão II de Estado, com a diferença de que lá a ênfase é em um estado com transições com outros e aqui a ênfase é na condição que determina uma transição. Como aqui não há um estado sendo

destacado, podem aparecer as atividades de todos os envolvidos: “atividade1” do “estado1” e “atividade2” do “estado2” (ver vínculos de atividades).

Esta característica de mostrar as atividades dos estados faz com que seja possível a comparação, na figura 8.9, da condição que causa a transição com o fluxo de controle que liga os processos que implementam (ver vínculos de implementação) as operações, consideradas como atividades. Integram-se as seguintes informações: os objetos da “Classe” no “estado1” executam a “atividade1” participando do “Processo1” e, sendo a “[condição x]” verdadeira passarão para o “estado2” e, desta forma, executam a “atividade2” participando do “Processo2”. O “Processo2”, nota-se a coerência, só é executado quando a condição do fluxo de controle que chega até ele (“[condição x]”) for verdadeira.

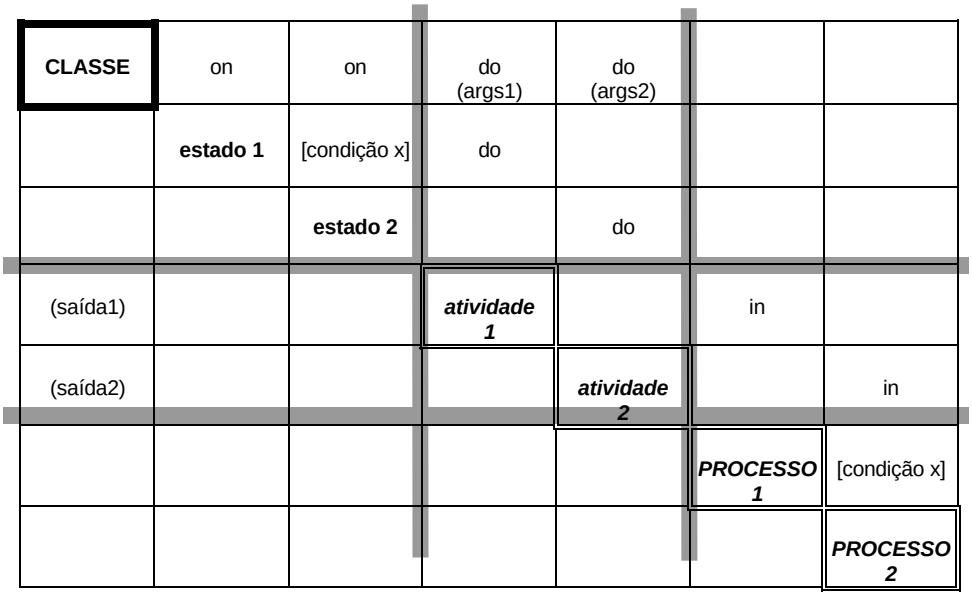


Figura 8.9. Padrão II de condição para “condição x”

- **Padrão I de Processo**: Um padrão I de processo (“Processo X”, na figura 8.10) mostra as operações (dos objetos de uma classe) que implementa, os fluxos de dados de entrada e de saída e a transição de estados que pode estar associada a essas operações.

No padrão I de Processo (figura 8.10) há a participação de evento externo ao processo e no padrão II de Processo (figura 8.11) o evento é interno (entre as operações implementadas no processo).

A figura 8.10 mostra a “Classe1” com os estados “estado” e “estado posterior” (ver vínculos de estados) com a transição conforme a seguinte leitura: “estado”, “evento”, “operação”, “[condição]”, “estado posterior”. Um padrão I de Processo é semelhante a um padrão I de Condição, sendo que a diferença é que aqui a ênfase é no processo e por isso podem ser observadas todas as operações implementadas por ele, no caso o “Processo X”. Este processo implementa as operações “operação” e “ação” e, portanto, pode ser associado diretamente à transição entre os estados dos objetos da “Classe1”.

A figura 8.10 mostra, então, a participação da “Classe1” no “Processo X”. Os fluxos de dados que chegam ao “Processo X” podem ser comparados com os argumentos de entrada das operações por ele implementadas. Da mesma forma, os fluxos de dados que saem devem ser coerentes em termos de tipo de dado, quando correlacionados com os retornos das operações.

CLASSE 1	on	on	do (dados1)	do (args)	dados1
	estado		evento		
		estado posterior			
(tipos de dados2)		[condição]	***operação***	[condição]	in
(saída)				***ação***	in
dados2					***PROCESSO X***

Figura 8.10. Padrão I de processo para “Processo X”.

- **Padrão II de Processo**: Um padrão II de processo (“Processo X”, na figura 8.11) mostra as operações que implementa, as classes de objetos que executam ou solicitam essas operações, os relacionamentos estabelecidos entre essas classes, as transições de estado que essas operações acarretam e os fluxos de dados envolvidos.

Neste padrão de processo, assim como mostra a figura 8.11, podem ser observadas as participações de mais de uma classe num determinado processo, já que, havendo a ocorrência de um evento, deve existir uma classe que o envia e outra que o recebe. O “Processo X”, então, tem a participação da “Classe1”, que envia o “evento” através da operação-causa “operação1” de sua responsabilidade, e da “Classe2”, que recebe o “evento” através da operação-efeito “operação2”.

Esta constatação é feita devido ao vínculo de evento entre as classes (célula de borda grossa), aos vínculos de responsabilidade (“do...”) entre as classes e as operações, ao próprio evento entre as operações e aos vínculos de implementação (“in”) entre as operações e o processo. Novamente é possível confirmar a colaboração entre as classes envolvidas, tendo como ênfase agora a participação delas num único processo. Esta colaboração é ratificada pela associação (célula sombreada), pelo evento (identificado pelo nome, entre as operações), pelo vínculo de evento (célula de borda grossa entre as classes) e pelo vínculo de interação (célula sombreada entre as operações).

Na figura 8.11 também é possível observar que a transição entre os estados “estado” e “estado posterior”, da “Classe2” (ver vínculos de estado), ocorre durante a execução do “Processo X”, pois a operação-efeito (“operação2”) do “evento”, que causa a transição, é implementada por este processo. Alem disto, consistências entre fluxos de dados que chegam e saem do processo e argumentos e retornos das operações podem ser feitas.

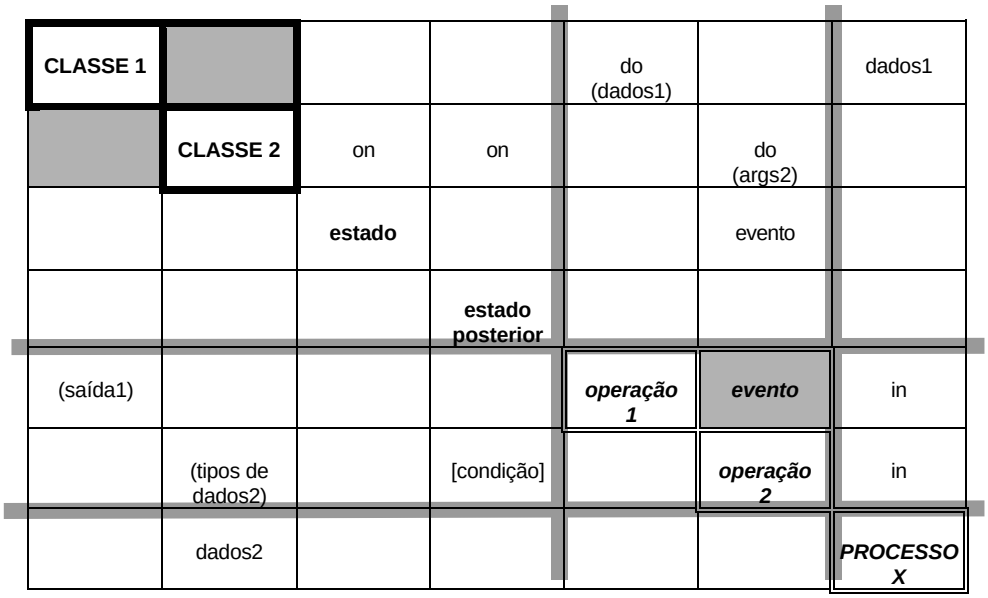


Figura 8.11. Padrão II de processo para o “Processo X”.

8.4 O Uso de Filtros

Para prover um mecanismo de abstração adequado, uma ferramenta que apoie a Matriz Square deve prever a seleção de filtros a qualquer momento durante a atividade diagramática e a partir de qualquer diagrama. Também deve ser possível a filtragem de uma matriz já filtrada, isto é, a aplicação de filtro sobre filtro. As funções básicas desejadas são:

- **Seleção de um Filtro.** Um filtro deve ser selecionado no ambiente onde a atividade diagramática se desenvolve. O modelador seleciona um elemento filtrante, destacando-o no diagrama e, em seguida, através do menu, por exemplo, escolhe o filtro desejado. Selecionados o filtro e o elemento filtrante, seria conveniente que o modelador visualizasse, em janelas diferentes, a matriz filtrada e fragmentos correspondentes dos diagramas de Objetos, de Fluxo de Eventos, de Estados e de Fluxos de Dados.
- **Sobreposição de Filtros.** Tendo obtido fragmentos de matriz e de diagramas, deve ser permitido ao modelador a seleção de um novo filtro e de um novo elemento filtrante correspondente. Um filtro sobreposto pode ser de três tipos: (a) por adição, com o acréscimo de novos elementos aos já filtrados; (b) por eliminação, com a manutenção somente dos elementos filtrados que satisfazem ao novo filtro ou (c) por eliminação seletiva, com a filtragem seletiva conforme a tabela 8.8. Desta maneira podem ser obtidos fragmentos de fragmentos já filtrados, aumentando o detalhamento e, portanto, melhorando as condições de verificação visual de consistência.

Filtro	Elemento Filtrante	Elementos eliminados da matriz filtrada
por Responsabilidade	classe de objetos	classes de objetos que não se relacionam com a filtrante
por Ligação	classe de objetos	classes de objetos que não se relacionam com a filtrante
por Estado	estado	estados que não têm transição direta com o filtrante
por Comunicação	dado	dados que não são elementos filtrantes em nenhum dos filtros
por Evento	evento	eventos diferentes do filtrante na linha deste
por Condição	condição	condições diferentes da filtrante na linha desta
por Processo	processo	processos que não têm fluxo de dado ou de controle com o filtrante

Tabela 8.8. Elementos eliminados na filtragem sobreposta por eliminação seletiva

8.5 Cenário

A seleção de um filtro durante a atividade diagramática e a sobreposição destes filtros permite a visualização de cenários sobre a matriz filtrada resultante. Rumbaugh et al [RUM91] referem-se, na metodologia OMT, a cenários e diagramas de Eventos. Um cenário é definido como uma seqüência de eventos que ocorrem durante uma execução particular de um sistema. Diagrama de Eventos é um cenário ampliado, onde a seqüência de eventos e os objetos que permutam estes eventos podem ser mostrados.

Rumbaugh et al [RUM91] sugerem a criação de cenários e, após, de diagramas de Eventos como etapa inicial na construção do modelo Dinâmico. Se isto for feito, esta formulação inicial do comportamento dinâmico de alguns objetos pode ser encontrada (ou não) como uma decorrência do que foi efetivamente modelado nos diagramas. Cenário é um nome apropriado para a visão propiciada por uma matriz filtrada.

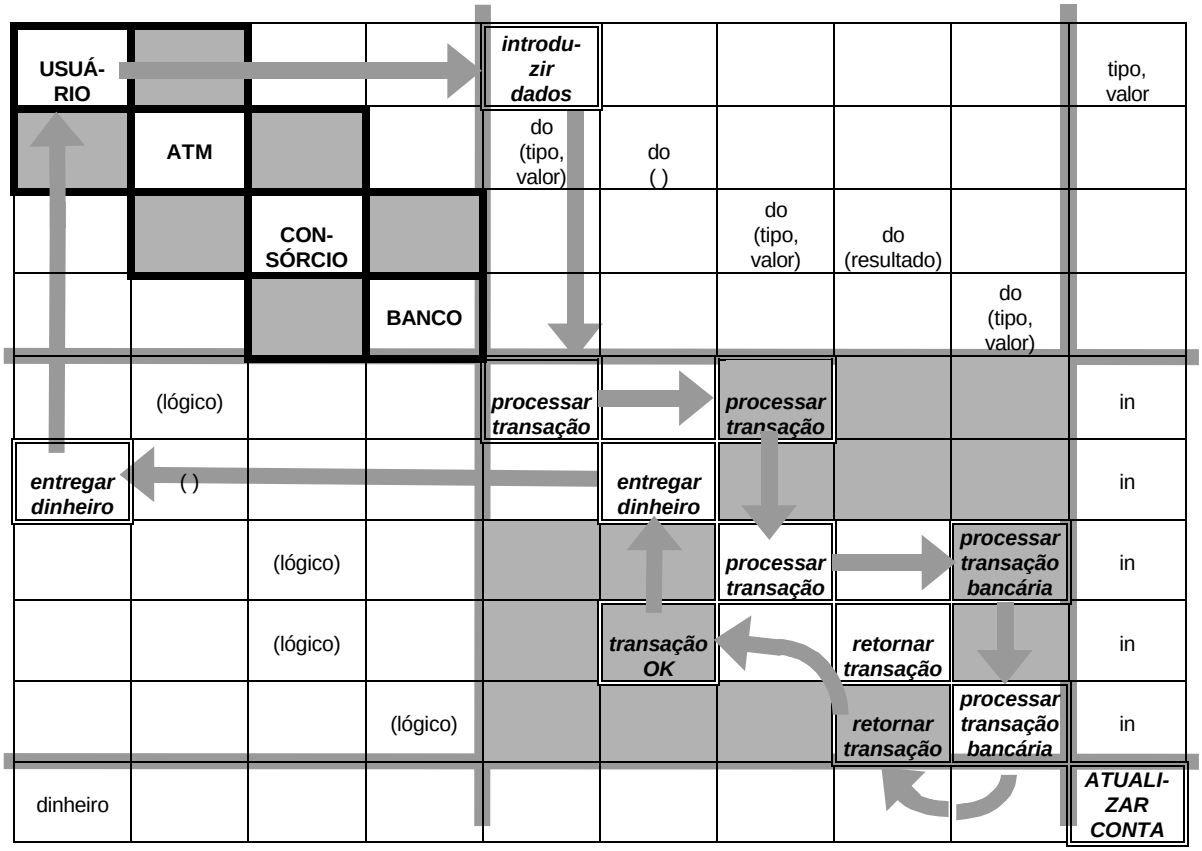


Figura 8.12. Matriz filtrada mostrando o processamento de uma transação em ATM.

A figura 8.12 mostra, através de um fragmento de Matriz Square, como um cenário é representado e como uma seqüência de eventos é interpretada no modelo matricial. É utilizada aqui a aplicação “Máquina de Caixa Automático (ATM)”, tomada como um dos estudos de caso nesta dissertação. O cenário, apresentado na figura 8.12, mostra a interação entre “Usuário”, “ATM”, “Consórcio” e “Banco”, quanto ao processamento de uma transação, implementado pelo processo “Atualizar Conta”. A aplicação de filtros, solicitando o contexto do processo “Atualizar Conta”, enfatiza a situação de uma transação executada retornando OK. Isto pode ser conseguido através da aplicação de um filtro por processo e outro por evento, com sobreposição por eliminação seletiva. Elimina-se, neste caso, toda a parte do cenário com a descrição de falha na transação.

As setas na figura 8.12, com início e término em “usuário” mostram a leitura da seqüência de eventos. Esta leitura indica que: “usuário” introduz tipo e valor em “ATM”; “ATM” solicita processar transação ao “consórcio”; “consórcio” solicita o mesmo processamento ao “banco”; “banco” processa transação e retorna o resultado ao “consórcio”; “consórcio” avisa transação OK à “ATM”; “ATM” entrega dinheiro ao “usuário”.

8.6 Conclusões

O mecanismo de integração proposto para os modelos da OMT deve executar a verificação das conexões explícitas, através dos elementos integradores que são seus componentes. Um dos modos desta verificação é a visualização da Matriz Square, com filtros, à procura de padrões que detectem a consistência do que está sendo modelado. A seleção de fragmentos da matriz através do uso de filtros provê um mecanismo de abstração para tornar viável a leitura do modelo matricial e para enfatizar vínculos e cenários que, a partir da observação dos diagramas é muito difícil. Há vínculos que são estabelecidos diretamente pela atividade diagramática, enquanto que outros ficam definidos de maneira implícita em razão dos primeiros. De acordo com o filtro utilizado, os vínculos que constituem as conexões explícitas podem ser detectados e os que são de interesse do modelador podem ser verificados na Matriz Square.

A tabela 8.9 mostra as conexões explícitas que podem ser verificadas por filtro na Matriz Square. Cada filtro permite a observação de um padrão de modelagem principal e, em alguns casos,

podem ainda aparecer padrões incompletos. Nesta tabela, para cada conexão há duas linhas: a superior indica o padrão principal e a inferior indica os padrões incompletos. Estes padrões permitem a verificação da existência das conexões explícitas de acordo com a representação dos elementos integradores que compõe cada conexão, conforme foi estabelecido na seção 5.3 do capítulo 5. Deste modo, a verificação de consistência entre os modelos da OMT, que é o objetivo do modelo matricial, é feita localmente.

conexão de ↓	**Filtros**						
	por Respon-sabilidade	**por Ligação**	**por Estado**	**por Comuni-cação**	**por Evento**	**por Condição**	**por Processo**
responsa-bilidade	resp	lig	est(I) ou est(II)	comu	ev	cond(I) ou cond(II)	proc(I) ou proc(II)
		resp	resp	resp	est(I), resp	est(I), est(II), resp	resp, est(I), est(II), comu
ligação		lig		comu	ev		proc(II)
				lig	lig		comu
estado			est(I) ou est(II)		ev	cond(I) ou cond(II)	
					est(I)	est(I), est(II)	
comuni-cação	resp	lig	est(I) ou est(II)	comu			proc(I) ou proc(II)
		comu	resp	resp, lig, proc(I), proc(II)			resp, est(I), est(II), comu
controle por evento	resp	lig	est(II)	comu	ev	cond(II)	proc(II)
		comu	resp	resp	est(I), resp, proc(I), proc(II)	est(I), resp, proc(I, proc(II)	resp, est(I), comu
controle por condição	resp		est(I) ou est(II)		ev	cond(I), ou cond(II)	proc(I) ou proc(II)
			resp		est(I)	est(I), est(II), resp, proc(I), proc(II)	resp, est(I), est(II), comu
processo	resp	lig	est(I) ou est(II)	comu	ev	cond(I) ou cond(II)	proc(I) ou proc(II)
	proc(I), proc(II)	proc(I), proc(II)	proc(I), proc(II)	proc(I), proc(II)	est(I), proc(I), proc(II)	est(I), est(II), proc(I), proc(II)	resp, est(I), est(II), comu

Tabela 8.9. Conexões explícitas identificadas em filtros através de padrões
(Padrões: resp=de Responsabilidade, lig= de Ligação, est(I)=I de Estado, est(II)=II de Estado, Comu=de Comunicação, ev=de Evento, cond(I)=I de Condição, cond(II)=II de Condição, proc(I)=I de Processo, proc(II)=II de Processo)

O outro modo de verificação das conexões explícitas previsto pela proposta é a aplicação de regras de consistência sobre a Matriz Square, que é o assunto do próximo capítulo.

9 A INTEGRAÇÃO DOS MODELOS DA OMT ATRAVÉS DE REGRAS DE CONSISTÊNCIA

"Mas que na forma se disfarce o emprego
Do esforço; e a trama viva se construa
De tal modo, que a imagem fique nua,
Rica, mas sóbria, como um templo grego"
Olavo Bilac

9.1 Introdução

Neste capítulo é discutida a aplicação das regras de consistência como um dos modos de verificação da integração entre os modelos da OMT. A seção 9.2 apresenta as regras de consistência utilizadas para consistir a Matriz Square, agrupadas em relação aos modelos que consistem. A seção 9.3 discute a aplicação das regras de consistência no contexto de um elemento ou globalmente.

9.2 As Regras de Consistência

Uma regra de consistência verifica as conexões explícitas entre os modelos. Estas conexões podem ser:

- **por identidade**, quando o mesmo elemento é representado em modelos diferentes, ou
- **por correspondência**, quando há um vínculo entre elementos diferentes modelados em modelos diferentes causado por algum tipo de conexão.

As regras de consistência são agrupadas de acordo com o seu contexto, ou seja, em relação aos modelos em que atuam. Assim, a seguir elas são apresentadas em três grupos: modelo de Objetos x modelo Dinâmico, modelo de Objetos x modelo Funcional e modelo Dinâmico x modelo Funcional. Cada uma das regras de consistência toma como base os postulados, apresentados no capítulo 5, e orienta-se pelas definições 5, 6, 7, 8, 9 e 11 (dos diagramas e da Matriz Square), apresentadas no capítulo 6.

9.2.1 Modelo de Objetos x Modelo Dinâmico

A seguir são apresentadas as regras de consistência entre o modelo de Objetos (MO) e o modelo Dinâmico (MD).

Pelo postulado 1, a estrutura do modelo Dinâmico é definida pela estrutura do modelo de Objetos; assim, as classes do modelo de Objetos aparecem trocando eventos no modelo Dinâmico.

- **Regra 1**. Uma classe de objetos no MD é uma classe de objetos no MO

Pelo postulado 1, um evento de um objeto para outro implica na existência de associação ou agregação entre as classes destes objetos..

- **Regra 2**. Um evento no MD implica em um relacionamento de associação ou agregação no MO entre as classes de objetos envolvidas

Pelo postulado 3, um evento ativa ou é ativado por operações definidas para as classes envolvidas e, assim, a seqüência de eventos estabelece a seqüência de operações.

- **Regra 3**. Um evento no MD corresponde a duas operações no MO: uma operação-causa, definida para a classe de objetos que estimula o evento, e uma operação-efeito, definida para a classe de objetos que recebe o evento.

Pelo postulado 2, os eventos determinam a ordem com que as operações são executadas. Pelo postulado 4, os atributos de um evento, transportando valores de dados, servem de argumentos para as operações a serem executadas. Como, pelo postulado 3, os eventos são ativados por operações, então, se estes eventos transportam informações, elas devem ser geradas pelas operações que os ativam.

- **Regra 4**. Os dados constituintes dos atributos de um evento do MD são os mesmos dos argumentos da sua operação-efeito, assim como é definido no MO

- **Regra 5**. Os tipos dos dados que constituem os atributos de um evento do MD são os mesmos tipos de dados que constituem a saída da sua operação-causa, assim como é definido no MO

Pelo postulado 12, uma condição controla operações nas transições de estado; o paradigma de orientação a objetos estabelece que um objeto muda de estado por alteração de seus atributos e que estes atributos só são afetados por operações executadas pelo próprio objeto; portanto uma condição que afeta uma transição deve considerar dados de conhecimento do objeto para que ele ative ou não a operação que acarretará a alteração de estado.

- **Regra 6**. Uma condição componente de transição de estado de uma classe de objetos no MD envolve atributos definidos para esta classe de objetos no MO

Pelo postulado 6, uma atividade é uma operação definida para uma classe e ocorre associada a um estado válido para os objetos desta classe, enquanto que uma ação, que também é uma operação definida para uma classe, é executada em decorrência de uma transição de estado.

- **Regra 7**. Uma atividade associada a um estado de uma classe de objetos no MD é uma operação definida para essa classe de objetos no MO
- **Regra 8**. Uma ação componente de transição de estados de uma classe de objetos no MD é uma operação definida para essa classe de objetos no MO

Pelos postulados 2 e 3, os eventos que uma classe recebe ativam operações nesta classe e, assim, definem a ordem em que as operações desta classe são executadas alterando, em decorrência, os estados válidos para os objetos desta classe.

- **Regra 9**. Um evento componente de transição de estados de uma classe de objetos no MD tem uma operação-efeito definida para essa classe de objetos no MO
- **Regra 10**. Um evento que uma classe de objetos recebe deve estar presente numa de suas transições de estado.

9.2.2 Modelo de Objetos x Modelo Funcional

A seguir são apresentadas as regras de consistência entre o modelo de Objetos (MO) e o modelo Funcional (MF).

Pelos postulados 8 e 9, há um mapeamento direto de atores e depósitos de dados para classes de objetos.

- **Regra 11**. Um depósito de dados no MF é uma classe de objetos no MO
- **Regra 12**. Um ator no MF é uma classe de objetos no MO

Pelo postulado 7, um processo envolve objetos que estão relacionados por função, sendo que as classes de objetos dele participam estimulando, executando ou sofrendo a execução de operações por ele implementadas.

- **Regra 13**. Um processo no MF tem a participação de pelo menos uma classe de objetos do MO, através das operações que implementa, e uma classe de objetos participa de pelo menos um processo

Pelo postulado 5, processos correspondem a operações de objetos, sendo que cada processo-folha é uma operação embora nem sempre esta relação um-para-um ocorra.

- **Regra 14**. Um processo no MF implementa pelo menos uma operação, definida para uma classe de objetos no MO, e é implementada por pelo menos um processo

Pelo postulado 11, fluxos de dados ligados a depósitos de dados correspondem a operações de atualização, os que chegam, e de consulta, os que saem; fluxos de dados ligados a atores correspondem a operações cujos resultados são dados do fluxo que chega ou a operações cujos argumentos são dados do fluxo que sai. Como depósitos de dados e atores têm fluxos de dados ligados obrigatoriamente a processos, estes são os implementadores das operações correspondentes aos fluxos e geram ou consomem os dados de entrada e saída destas operações.

- **Regra 15**. Os dados dos fluxos de dados que chegam a um processo no MF devem[2] estar contidos nos argumentos das operações implementadas nele, assim como é definido para as classes de objetos do MO
- **Regra 16**. Os dados dos fluxos de dados que saem de um processo no MF devem[1] corresponder em tipo à saída das operações implementadas nele, assim como é definido para as classes de objetos do MO

Pelo postulado 7, um processo relaciona por função as classes de objetos que dele participam; esta participação ocorre, de um lado, estimulando e, de outro, executando ou sofrendo a execução de operações. Teríamos de um lado uma classe enviando um evento e de outro uma classe recebendo este evento. Além disto, um evento pode ser relacionado a um fluxo de dados, pelo postulado 10, ao transportar informação, ou a um fluxo de controle, pelo postulado 12, como elemento de controle. Um evento então pode ser localizado dentro de um processo ou entre processos. Pelos postulados 1 e 3, a execução de uma operação de uma classe é estimulada através de um evento enviado a partir de outra classe e isto implica na existência de associação ou agregação entre as classes envolvidas.

- **Regra 17**. As classes de objetos do MO, que participam de um processo-folha comum no MF, possuem associação ou agregação entre si
- **Regra 18**. As classes de objetos do MO, que participam de processos-folha do MF ligados por fluxo de dados ou de controle, possuem associação ou agregação entre si

Pelo postulado 11, quando ligado a um depósito de dados, um fluxo de dados corresponde a uma operação de atualização ou de consulta e um fluxo de objeto a uma operação de criação.

- **Regra 19**. Um fluxo de dados que liga um processo a um depósito de dados no MF implica na implementação de uma operação, definida para a classe de objetos correspondente do MO, pelo processo considerado. A implicação da operação no fluxo de dados também é verdadeira.

[2] Podem existir operações implementadas (quando houver mais de uma) num processo cujos argumentos são internos ao processo e, portanto, não aparecem nos fluxos de dados.

9.2.3 Modelo Dinâmico x Modelo Funcional

A seguir são apresentadas as regras de consistência entre o modelo Dinâmico (MD) e o modelo Funcional (MF).

Pelo postulado 3, um evento ativa ou é ativado por operações e, pelos postulados 5 e 6, processos correspondem a operações implementando-as como consultas, atividades ou ações; portanto as operações estabelecem mapeamentos entre eventos e processos.

- **Regra 20**. Um evento do MD está associado, através das operações envolvidas, a pelo menos um processo do MF

Uma operação altera o estado de um objeto ativada por um evento, pelo postulado 2, ou controlada por uma condição, pelo postulado 12. Como as operações, pelos postulados 5 e 6, são implementadas por processos, então as transições de estado podem ser amarradas a processos.

- **Regra 21**. Uma transição de estados do MF está associada, através das operações envolvidas, a pelo menos um processo do MF

Pelo postulado 12, decisões e seqüências são questões de controle modeladas por condições e eventos ou por fluxos de controle; condições e eventos controlam e determinam a seqüência de operações e fluxos de controle fazem o mesmo com processos que, pelo postulado 6, implementam as operações.

- **Regra 22**. Um fluxo de controle do MF é uma condição componente de uma transição de estado ou um evento, como agente de controle, do MD e vice-versa

Pelo postulado 10, um fluxo de dados transporta dados assim como um evento quando transmite informação. Pelo postulado 3, um evento ativa ou é ativado por operações. Assim um evento pode servir de veículo para a transferência de dados entre operações como um fluxo de dados o faz entre processos. Como, pelo postulado 5, um processo por vezes corresponde a diversas operações e, em outros casos, uma operação corresponde a diversos processos, os fluxos de dados que entram e saem de um processo devem corresponder aos argumentos e aos retornos das operações nele implementadas, excluindo-se argumentos e retornos internos ao processo.

- **Regra 23**. Os dados como atributos de um evento do MD estão contidos ou contém os dados de pelos menos um fluxo de dados do MF que chega ao processo que implementa a operação-efeito do evento.
- **Regra 24**. Os tipos de dados dos atributos de um evento do MD correspondem aos tipos de dados dos componentes de pelo menos um fluxo de dados do MF que sai do processo que implementa a operação-causa do evento.

9.3 Aplicação das Regras de Consistência

A aplicação de uma regra de consistência deve ser permitida (a) por elemento ou (b) global. Esta aplicação diferente se dá devido a sua ativação em dois momentos:

- **por elemento**: quando um elemento é modelado num diagrama. Neste caso devem ser aplicadas regras de consistência específicas relacionadas àquele elemento e ao diagrama utilizado.

- **global**: quando o modelador solicita um relatório de consistência. Neste caso o mecanismo de consistência deve examinar toda a Matriz Square relatando os casos em que as regras são desobedecidas.

Deve ser previsto pela ferramenta que apoie a Matriz Square que a violação de uma regra de consistência aplicada por elemento possa gerar um aviso, um impedimento ou uma correção automática:

- **Aviso**. Uma mensagem informa que alguma ação deve ser tomada pelo usuário para corrigir ou aceitar que a regra seja desobedecida, conforme o caso.

- **Impedimento**. Uma mensagem informa que o usuário está impedido de executar a ação que causou a violação da regra.

- **Correção automática**. Uma correção automática é feita nos diagramas e, também na Matriz Square adequando os modelos à regra de consistência em decorrência de alguma atividade executada pelo modelador.

Estes três tipos de reação à violação de uma regra aplicada por elemento devem ser permitidos para tornar a modelagem adequadamente flexível e disciplinada. A decisão desta gradação deve ser tomada na implementação de uma ferramenta que apoie a Matriz Square.

A tabela 9.1 mostra as regras de consistência conforme a aplicação em relação aos setores da matriz Square. A posição sombreada indica o setor em que a regra é aplicada.

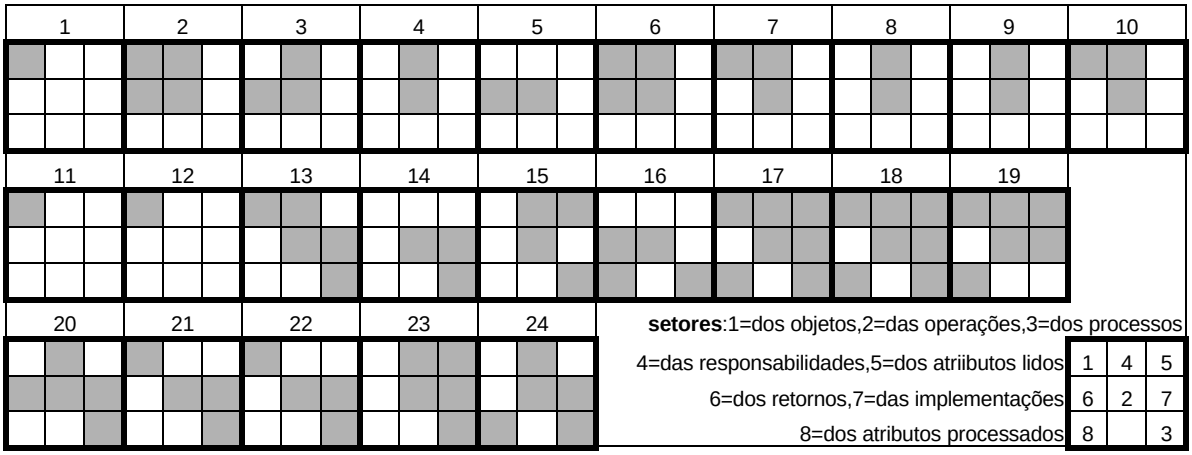


Tabela 9.1. Aplicação das regras de consistência por setor

Cada regra não necessita pesquisar toda a matriz para ser aplicada, conforme mostra a tabela 9.1. A pesquisa da matriz deve ser auxiliada sempre pela Tabela de Elementos, principalmente quanto à vinculação do elemento ao diagrama utilizado para modelá-lo.

9.4 Conclusões

Um dos modos da verificação de integração entre os modelos da OMT é a visualização da Matriz Square através de filtros, como foi visto no capítulo anterior. Esta consistência é feita enfatizando contextos limitados em relação ao todo da modelagem. O mecanismo de integração proposto prevê que a verificação seja feita, também, globalmente. Isto é conseguido pela aplicação de regras de consistência sobre as conexões explícitas existentes entre os três modelos da OMT.

As tabelas 9.2, 9.3 e 9.4 mostram as conexões explícitas que podem ser consistidas pela aplicação das regras de consistência sobre a Matriz Square. As tabelas 9.2 e 9.3 mostram também a conexão de classe que não tem a participação de operações como agente integrador. Cada regra consiste os elementos integradores que compõem cada conexão, conforme foi estabelecido na seção 5.3 do capítulo 5. Deste modo, a verificação de consistência entre os modelos da OMT, que é o objetivo do modelo matricial, é feita globalmente.

	conexões						
regras de consistência	**de respon-sabilidade**	**de ligação**	**de estado**	**de comuni-cação**	**de controle por evento**	**de controle p/ condição**	**de classe**
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

Tabela 9.2. Conexões explícitas entre os modelos de Objetos e Dinâmico verificadas pelas regras de consistência (o sombreamento indica a aplicação da regra sobre a conexão)

	conexões				
regras de consistência	**de respon-sabilidade**	**de ligação**	**de comuni-cação**	**de processo**	**de classe**
11					
12					
13					
14					
15					
16					
17					
18					
19					

Tabela 9.3. Conexões explícitas entre os modelos de Objetos e Funcional verificadas pelas regras de consistência (o sombreamento indica a aplicação da regra sobre a conexão)

	conexões				
regras de consistência	**de estado**	**de comuni-cação**	**de controle por evento**	**de controle por condição**	**de processo**
20					
21					
22					
23					
24					

Tabela 9.4. Conexões explícitas entre os modelos Dinâmico e Funcional verificadas pelas regras de consistência (o sombreamento indica a aplicação da regra sobre a conexão)

No próximo capítulo são apresentados dois estudos de caso, onde são mostrados os resultados da aplicação das regras de consistência e do uso de filtros sobre os modelos matriciais, construídos a partir da modelagem que foi realizada em cada caso.

10 ESTUDOS DE CASO

"Prova, Olha, Toca, Cheira, Escuta
Cada sentido é um dom divino."
Manuel Bandeira

10.1 Introdução

Neste capítulo são apresentados dois estudos de caso utilizando a Matriz Square para consistência dos modelos construídos. A seção 10.2 mostra a preparação de conferência de trabalho da IFIP, cuja modelagem é feita através da metodologia OMT, por Reinoso [REI94]. A seção 10.3 mostra a máquina de caixa automática (ATM), cuja modelagem é apresentada por Rumbaugh et al [RUM91].

Procurou-se ser o mais fiel possível aos autores, incluindo-se apenas informações indispensáveis ao tratamento do caso. Como os modelos construídos não foram detalhados suficientemente, a consistência não pode aprofundar-se como deveria, entretanto é possível constatar inconsistências em termos de integração de modelos.

A seguir são apresentados, para cada caso, a definição do problema, os objetos da aplicação, os modelos de Objetos, Dinâmico e Funcional desenvolvidos pelos autores, as relações operação x processo e operação x evento que interessam (deduzidas das miniespecificações dos processos e das definições dos problemas) e a integração.

10.2 Preparação de Conferência de Trabalho da IFIP

Será utilizado o relatório de pesquisa de Guilhermo Bustos Reinoso, realizado no CPGCC da UFRGS, em junho de 1994 [REI94]. Sobre a modelagem feita neste relatório são mostrados a seguir filtros do modelo matricial correspondente, com a devida consistência. Os problemas de inconsistência encontrados não têm grande repercussão no resultado geral da modelagem executada, pela sua pequena dimensão (com poucas classes de objetos e poucas funções) e, provavelmente, devido à qualidade do trabalho do modelador que, deve ser salientado, não é o alvo desta análise. A falta de um detalhamento maior com a construção de níveis inferiores no DFD também impedem uma consistência mais forte.

10.2.1 Definição do Problema

A definição do problema, originalmente apresentada por T.W.Olle [OLL82], a seguir é incluída conforme está descrita por Reinoso [REI94]:

> "1. Antecedentes
>
> Uma Conferência de Trabalho da IFIP é uma conferência internacional que tenta reunir especialistas de todos os países da IFIP para discutir aspectos técnicos de interesse específico para um ou mais Grupos de Trabalho da IFIP. O procedimento usual, que será considerado para este propósito, é o de uma conferência convidada que não é aberta para qualquer um. Para tais conferências, parte do problema é assegurar que todos os membros relacionados com o(s) Grupo(s) de Trabalho e Comitê(s) Técnico(s) da IFIP sejam convidados, mesmo que eles não possam comparecer. Além disso, é importante assegurar suficientes assistentes à conferência de tal modo que a viabilidade financeira seja atingida sem exceder o máximo indicado pelas facilidades disponíveis.
>
> As políticas da IFIP sobre Conferências de Trabalho sugerem a nomeação de um Comitê de Programa para tratar com o conteúdo técnico da conferência e um Comitê de Organização para tratar das matérias financeiras, preparação do local, convites e publicidade. É claro que estes comitês precisam trabalhar juntos, possuem necessidades comuns de informação e precisam manter registros de informação consistentes e atualizados.
>
> 2. Sistema de informação a ser projetado
>
> O sistema de informação a ser projetado deve suportar as atividades dos comitês de Programa e Organização vinculados à preparação de uma Conferência de Trabalho da IFIP. O envolvimento destes dois comitês é considerado análogo a duas entidade organizacionais em estrutura corporativa que usam informações em comum.
>
> Devem ser suportadas as seguintes atividades dos comitês:
>
> Comitê de Programa:
>
> 1. Preparar uma lista de pessoas para enviar a chamada de trabalhos.
> 2. Cadastrar cartas de intenção em resposta à chamada.
> 3. Cadastrar os artigos de contribuição recebidos.
> 4. Distribuir artigos entre aqueles que realizam a avaliação.
> 5. Coletar os relatórios dos avaliadores e selecionar os artigos para inclusão no programa.
> 6. Agrupar os artigos aceitos em sessões para apresentação e selecionar o moderador para cada sessão.
>
> Comitê de Organização:
>
> 1. Preparar uma lista de pessoas que convite para a conferência.
> 2. Imprimir convites prioritários para Representantes Nacionais, membros do Grupo de Trabalho e membros de grupos de trabalho associados.
> 3. Assegurar que todos os autores de artigos aceitos recebam um convite.
> 4. Assegurar que os autores de artigos rejeitados recebam um convite.
> 5. Evitar o envio de convites duplicados para qualquer pessoa.
> 6. Registrar a aceitação dos convites.
> 7. Gerar a lista final dos participantes.
>
> 3. Fronteiras do sistema

> Deve ser notado que os aspectos financeiros e de cronograma de trabalho do Comitê de Organização, a planificação de ambos os comitês, as acomodações em hotéis para os participantes e as matérias relativas à preparação de cópias dos anais tem sido omitidos deste exercício, porém uma submissão pode incluir algum ou todos estes aspectos adicionais se os autores sentem-se motivados."

10.2.2 Objetos da Aplicação

Pela definição do problema são consideradas as seguintes classes de objetos:

- Pessoa: generalização de receptor Chamada Trabalhos e/ou de Convidado
- Carta Intenção: carta com a intenção de submissão de artigo
- Resposta Convite: resposta a um convite à conferência
- Receptor Chamada Trabalhos: pessoa que recebe uma chamada para trabalhos
- Convidado: pessoa convidada para a conferência
- Participante: pessoa que participa da conferência, sendo uma generalização de Moderador, Autor, Avaliador e/ou Membro Grupo
- Convidado Prioritário: Convidado que recebe convite prioritário, sendo uma generalização de Membro Grupo ou de Representante Nacional
- Moderador: Participante que atua como moderador de uma Sessão
- Autor: Participante como autor de artigo submetido à conferência
- Avaliador: Participante como avaliador de artigos submetidos à conferência
- Membro Grupo: Participante como membro de grupo de trabalho
- Representante Nacional: Convidado Prioritário como representante de país
- Sessão: sessão da conferência
- Artigo: artigo submetido à conferência
- Relatório: relatório de avaliação de artigo
- Artigo Aceito: artigo submetido e aceito para a conferência
- Artigo Rejeitado: artigo submetido e rejeitado para a conferência

10.2.3 Modelo de Objetos

A figura 10.1 mostra o diagrama de Objetos para a IFIP.

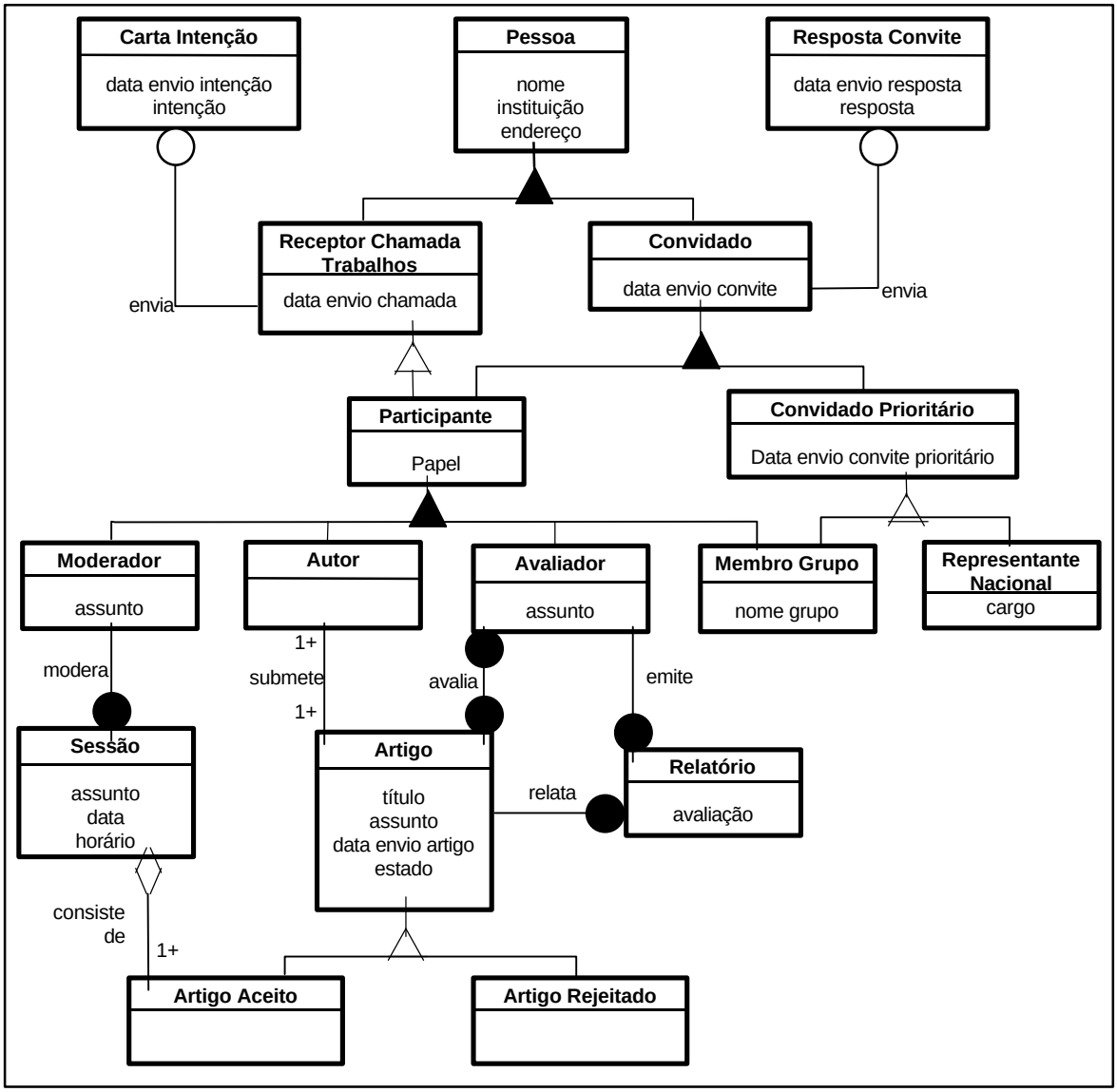


Figura 10.1. Diagrama de Objetos em IFIP [REI94]

10.2.4 Modelo Dinâmico

A figura 10.2 mostra o diagrama de Fluxo de Eventos e a figura 10.3 mostra o diagrama de Estados de Artigo para IFIP.

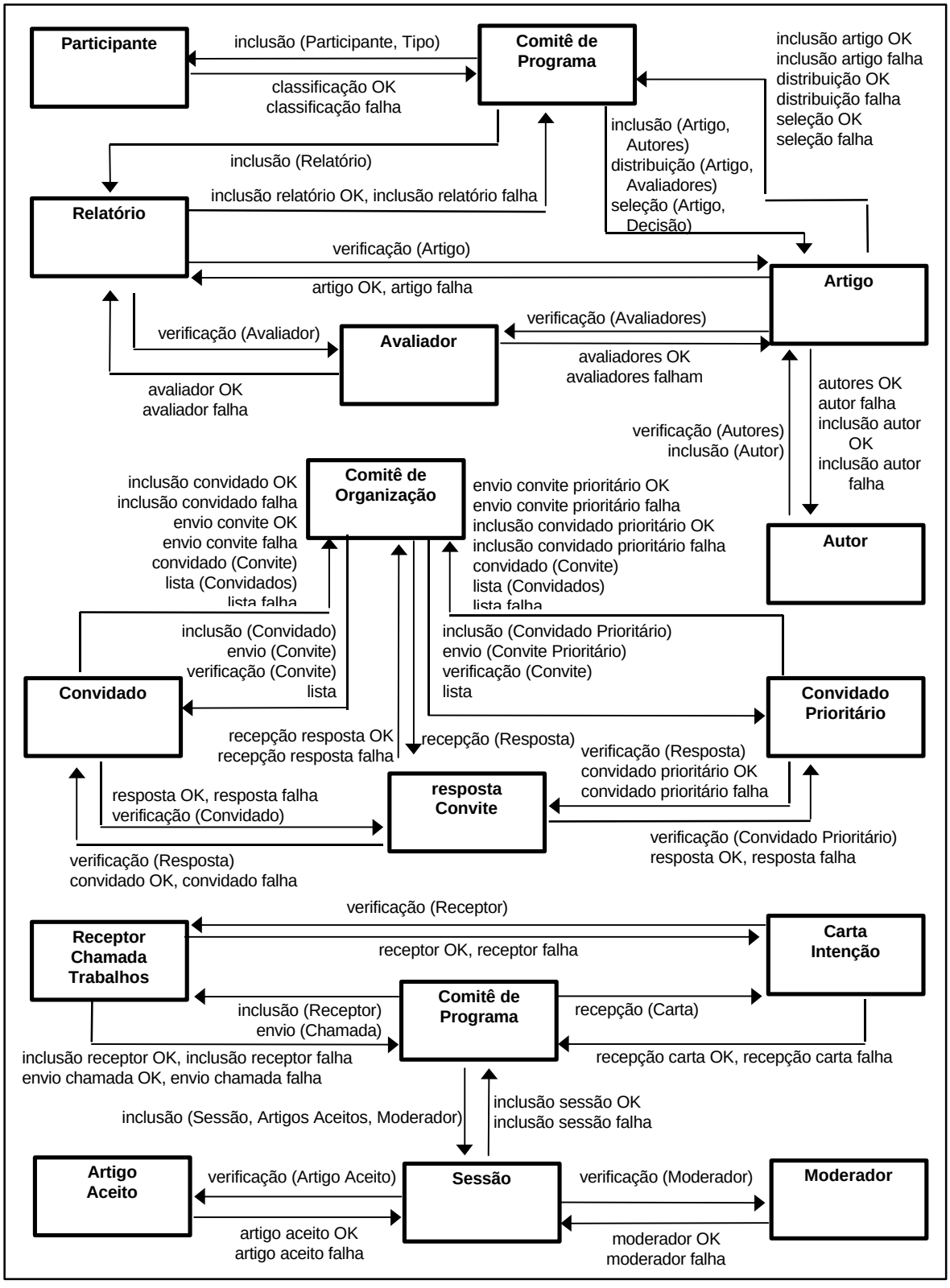
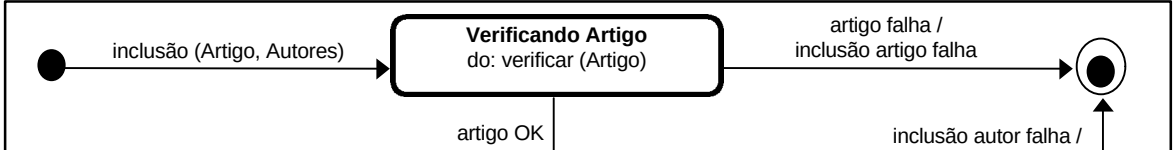


Figura 10.2. Diagrama de Fluxo de Eventos em IFIP [REI94]



10.2.5 Modelo Funcional

A figura 10.4 mostra o diagrama de Fluxo de Dados do nível 0 para IFIP e as figuras 10.5, 10.6 e 10.7 mostram os DFDs para o detalhamento dos processos do nível 0.

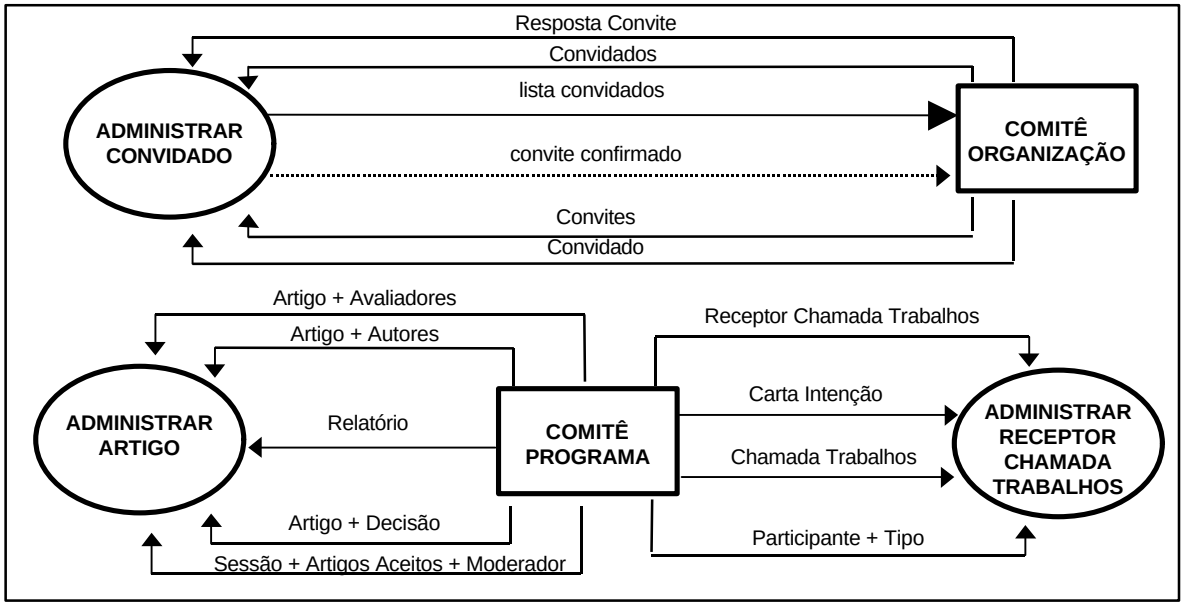
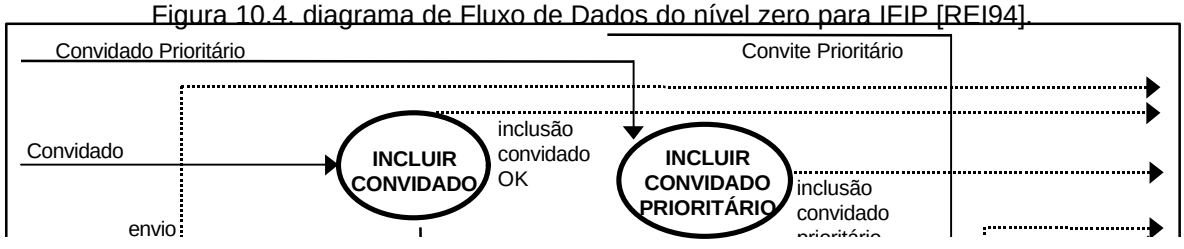


Figura 10.4. diagrama de Fluxo de Dados do nível zero para IFIP [REI94].



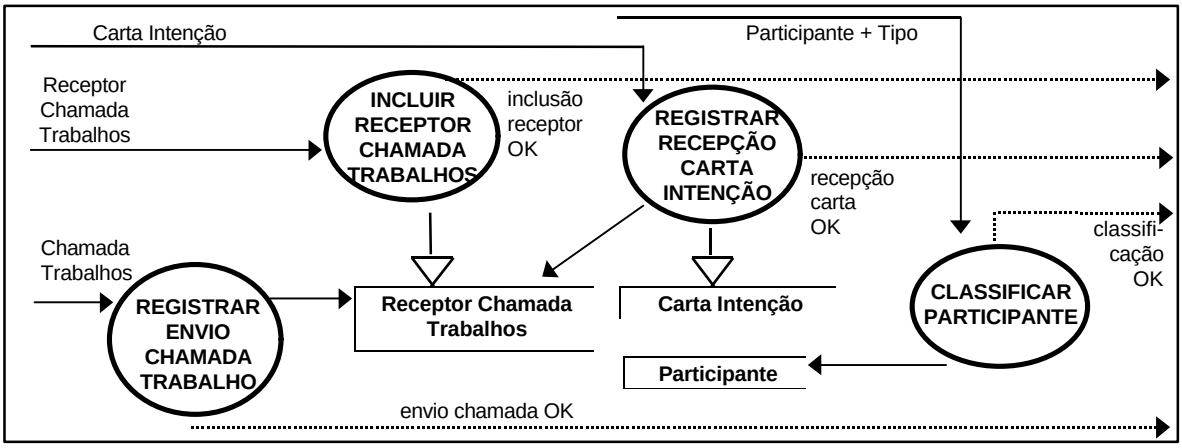


Figura 10.6. diagrama de Fluxo de Dados de “Adm. Receptor Chamada Trab.” para IFIP [REI94].

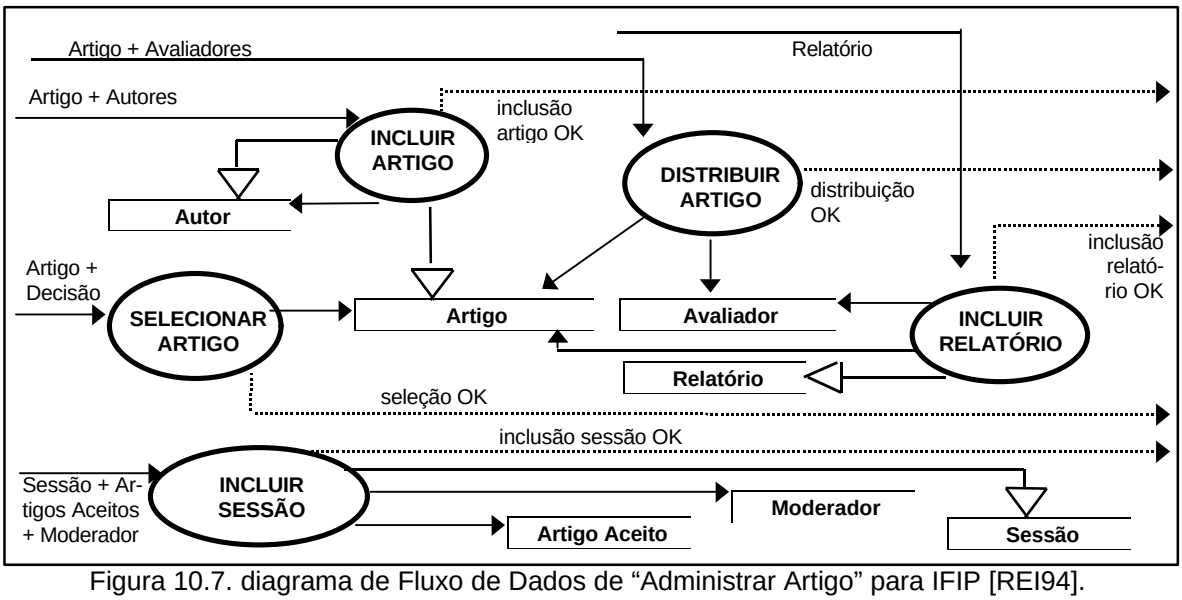


Figura 10.7. diagrama de Fluxo de Dados de “Administrar Artigo” para IFIP [REI94].

10.2.6 Operações x Processos

As operações implementadas pelos processos-folha referentes ao processo “Administrar Convidado” são mostradas na tabela 10.1 e as referentes ao processo “Administrar Artigo” são mostradas na tabela 10.2.

PROCESSO-FOLHA	OPERAÇÕES	CLASSE DE OBJETOS
Incluir Convidado	incluir (Convidado)	Convidado
Registrar Envio Convite	enviar (Convite)	
Incluir Convidado Prioritário	incluir (Convidado Prioritário)	Convidado Prioritário
Registrar Envio Convite Prioritário	enviar (Convite Prioritário)	
Verificar Convite	verificar (Convidado)	Convidado
	verificar (Convidado Prioritário)	Convidado Prioritário
Registrar Recepção Resposta Convite	incluir (Resposta Convite)	Resposta Convite
Listar Convidados	listar (Convidados)	Convidado

Tabela 10.1. Operações implementadas no processo “Administrar Convidado”

PROCESSO-FOLHA	OPERAÇÕES	CLASSE DE OBJETOS
Incluir Artigo	verificar (Autores)	Autor
	incluir (Autor)	
	incluir (Artigo, Autores)	Artigo
Selecionar Artigo	verificar (Artigo)	
	selecionar (Artigo, Decisão)	
Distribuir Artigo	verAvaliadores(Avaliadores)	
	verificar (Avaliadores)	Avaliador
	verificar (Artigo)	Artigo
	distribuir (Artigo, Avaliadores)	
Incluir relatório	verificar (Artigo)	
	incluir (Relatório)	Relatório
Incluir Sessão	verificar (Artigo Aceito)	Artigo Aceito
	verificar (Moderador)	Moderador
	incluir (Sessão, Artigos Aceitos, Moderador)	Sessão

Tabela 10.2. Operações implementadas no processo “Administrar Artigo”

10.2.7 Operações x Eventos

A tabela 10.3 mostra os eventos referidos na integração relatada a seguir, com as classes de origem e de destino e as operações-causa e efeito correspondentes.

CLASSE ORIGEM	operação-causa	EVENTO	operação-efeito	CLASSE DESTINO
Comitê Organiz.	—	inclusão	incluir	Convidado
		seleção	selecionar	Artigo
		distribuição	distribuir	
Convidado	incluir	inclusão convidado OK	—	Comitê Organiz.
Artigo	distribuição OK	distribuição OK		
	distribuição falha	distribuição falha		
	seleção falha	seleção falha		
Artigo	verAvaliadores	verificação	verificar	Avaliadores
Avaliador	verificar	avaliadores OK	distribuição OK	Artigo
		avaliadores falham	distribuição falha	

Tabela 10.3. Eventos, operações e classes envolvidas.

10.2.8 Integração dos Modelos

A figura 10.8 mostra a ausência de associação (ver seta indicadora na figura 10.8) modelada entre "Comitê Organização" e "Convidado". A primeira classe representada no diagrama de fluxo de eventos não foi modelada no diagrama de objetos e, por isto, a associação também não, talvez por decisão do modelador. O padrão do processo "Incluir Convidado" e do evento "Inclusão" ficam, desse modo, incompletos, como mostra a figura 10.8. Foram violadas as regras de consistência números 1 e 2.

Nos DFDs apresentados, o modelador utiliza diversas vezes fluxos de controle entre processos e atores. Como isto não está de acordo com o que a OMT estabelece, estes fluxos de controle são entendidos neste estudo de caso como fluxos de dados e suas condições como mensagens. De outro modo, como fluxo de controle, estes elementos não conseguiriam ser representados na matriz Square, violando suas regras de sintaxe que não são expressas como regras de consistência mas colaboram decisivamente para a integração dos modelos. Na figura 10.8, "inclusão convidado OK" é um destes casos.

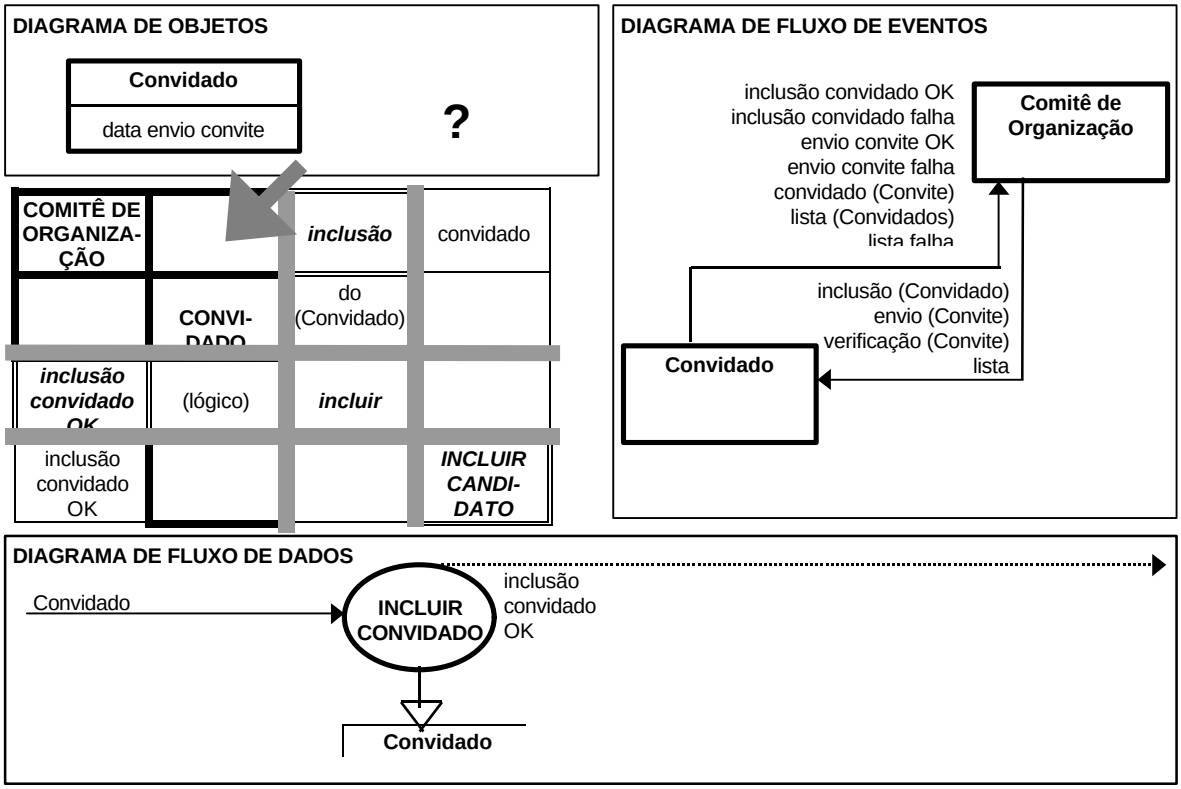


Figura 10.8. Ausência de associação entre Comitê Organização e Convidado

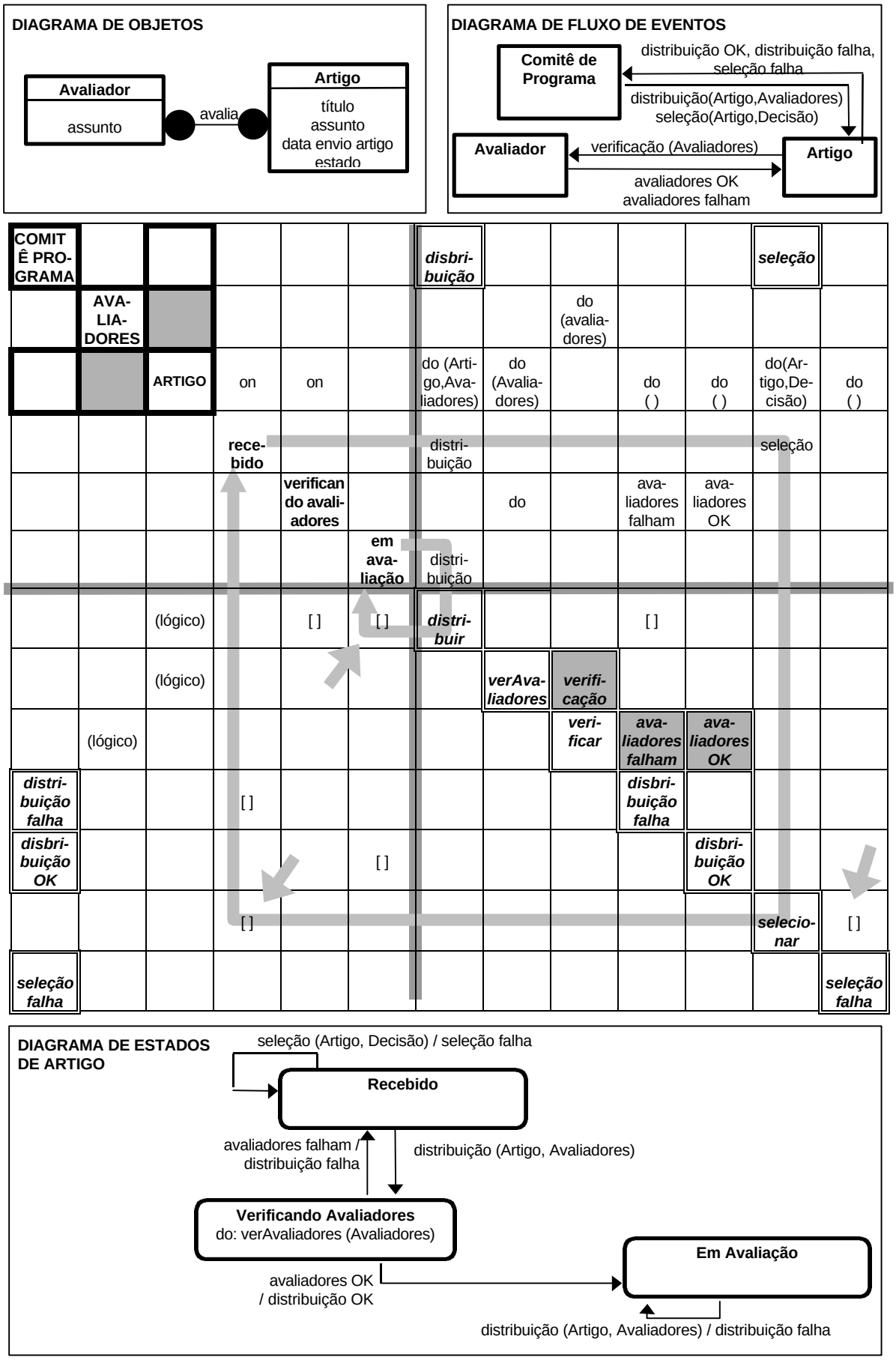


Figura 10.9. Eventos e transições de Artigo

A figura 10.9 mostra os estados de “Artigo” com transições de “recebido” para “verificando avaliadores” e deste para “em avaliação”. Um padrão I de Estado para o estado “verificando avaliadores” mostraria o que é visto na figura 10.9. Verifica-se que o evento “seleção” faz com que “Artigo” permaneça no estado “recebido” e execute a ação “seleção falha”, obrigatoriamente, pois não há condição nenhuma associada. O mesmo ocorre com o evento “distribuição” sobre o estado “em avaliação”: é executada a ação “distribuição falha” incondicionalmente. Nos dois casos isto fica marcado pela condição representada por “[ ]” colocada automaticamente em função da diagramação na linha da operação “selecionar” e entre a operação “distribuir” e o estado “em avaliação”. Uma possível solução é apresentada a seguir na figura 10.10.

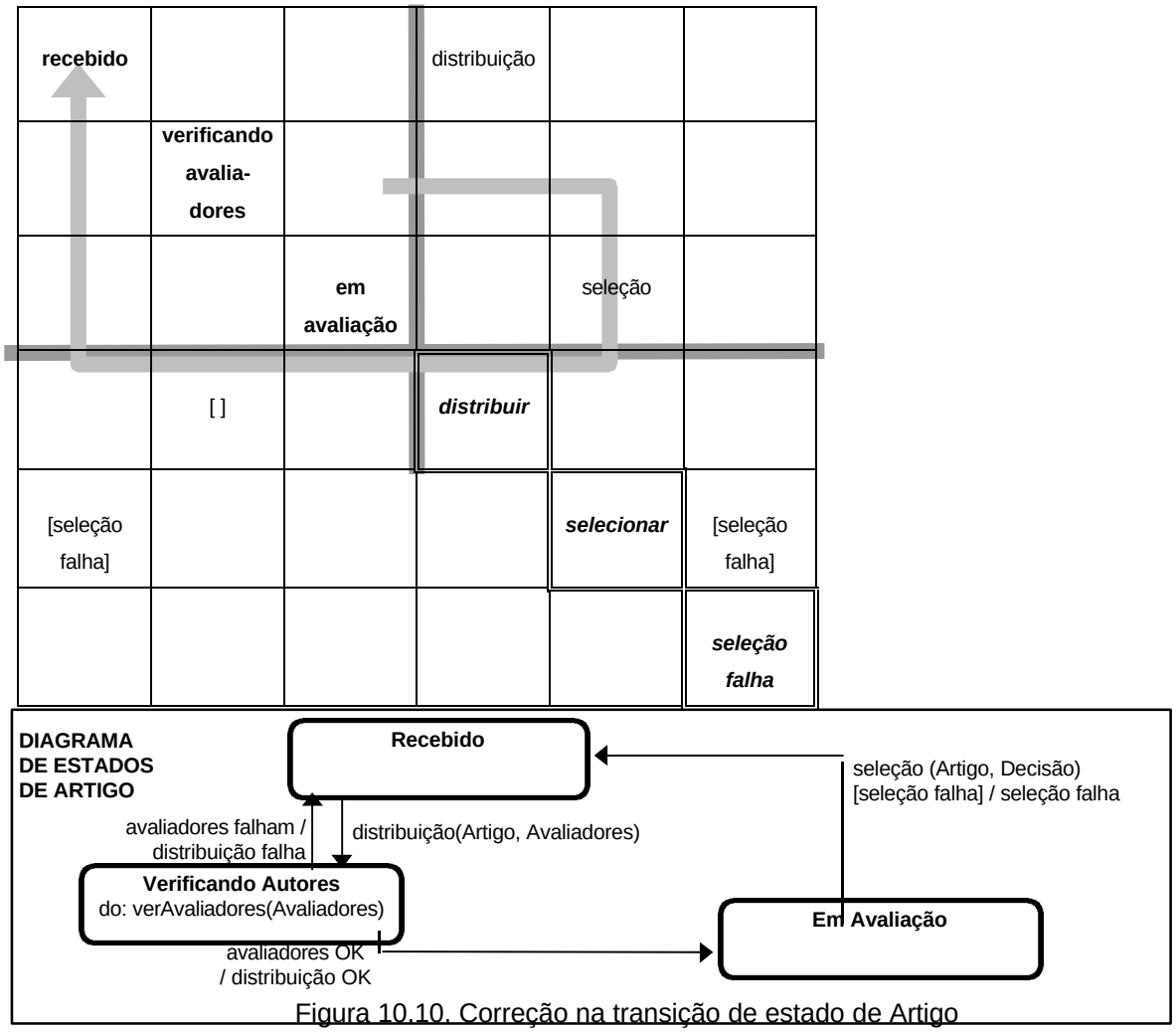


Figura 10.10. Correção na transição de estado de Artigo

10.3 Máquina de Caixa Automático (ATM)

Será utilizada a modelagem apresentada por Rumbaugh et al no capítulo 8 (Analysis) de “Object-Oriented Modeling and Design” [RUM91] sobre o uso de máquinas de caixa automático

(“automatic teller machines” — ATMs) num consórcio de bancos. Esta modelagem é apresentada pelos autores da OMT para exemplificar e discutir a fase de análise abordada pela metodologia.

10.3.1 Definição do Problema

A definição do problema a seguir é apresentada conforme está descrita por Rumbaugh et al [RUM91]:

> “Desenvolva o software para apoiar uma rede bancária computadorizada incluindo caixas humanos e máquinas de caixa automático (ATM) a serem compartilhadas por um consórcio de bancos. Cada banco provê seu próprio computador para manter suas contas e processar transações sobre elas. Os caixas tradicionais são propriedade dos bancos e se comunicam diretamente com os computadores de seus bancos proprietários. Os caixas humanos introduzem dados sobre contas e transações. Os caixas automáticos (ATM) comunicam-se com um computador central que liquida as transações com os bancos adequados. Um caixa automático recebe cartões magnéticos, interage com o usuário, comunica-se com o sistema central para executar transações, entrega dinheiro e imprime extratos. O sistema exige um adequado arquivamento de registros e reservas de segurança. O sistema deve manipular corretamente acessos concorrentes à mesma conta. Os bancos devem prover seu próprio software para seus próprios computadores; você deve projetar o software para as ATMs e para a rede. O custo do sistema compartilhado deve ser distribuído pelos bancos de acordo com o número de clientes com cartões magnéticos.”

10.3.2 Objetos da Aplicação

Em virtude da aplicação não ter sido completamente desenvolvida pelos autores [RUM91], serão utilizadas somente parte do modelo de Objetos por eles apresentado de forma a viabilizar sua consistência com os outros. Desse modo, serão considerados as seguintes classes de objetos:

- ATM: máquina de caixa automático para verificação de extratos de conta e retirada de dinheiro.
- Consórcio: organização de bancos que comissiona e opera a rede de ATMs.
- Banco: instituição financeira que mantém contas de clientes
- Conta: conta de banco na qual as transações podem ser aplicadas.
- Transação: solicitação de operações nas contas de um cliente.
- Cartão magnético: cartão de um cliente do banco que dá acesso às suas contas
- Cliente: o dono de uma ou mais contas em um banco.

10.3.3 Modelo de Objetos

A figura 10.11 mostra o diagrama de Objetos para ATM.

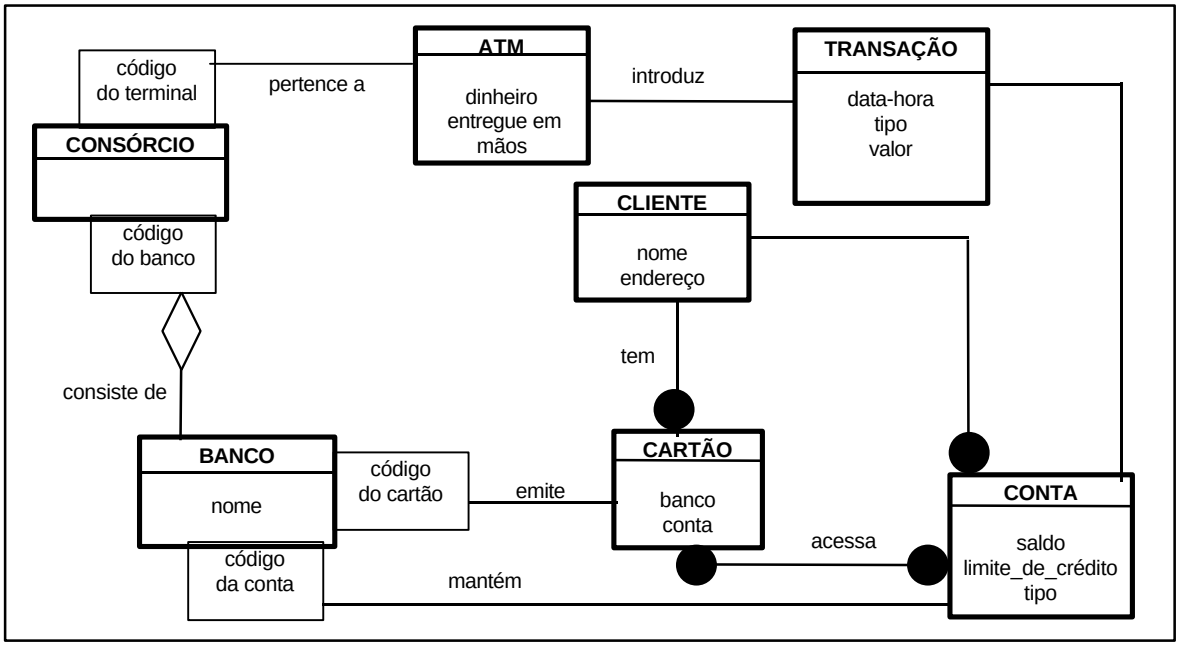


Figura 10.11. Modelo de Objetos em ATM [RUM91]

10.3.4 Modelo Dinâmico

A figura 10.12 mostra o diagrama de Fluxo de Eventos e as figuras 10.13, 10.14 e 10.15 mostram os diagramas de Estados para algumas classes em ATM.

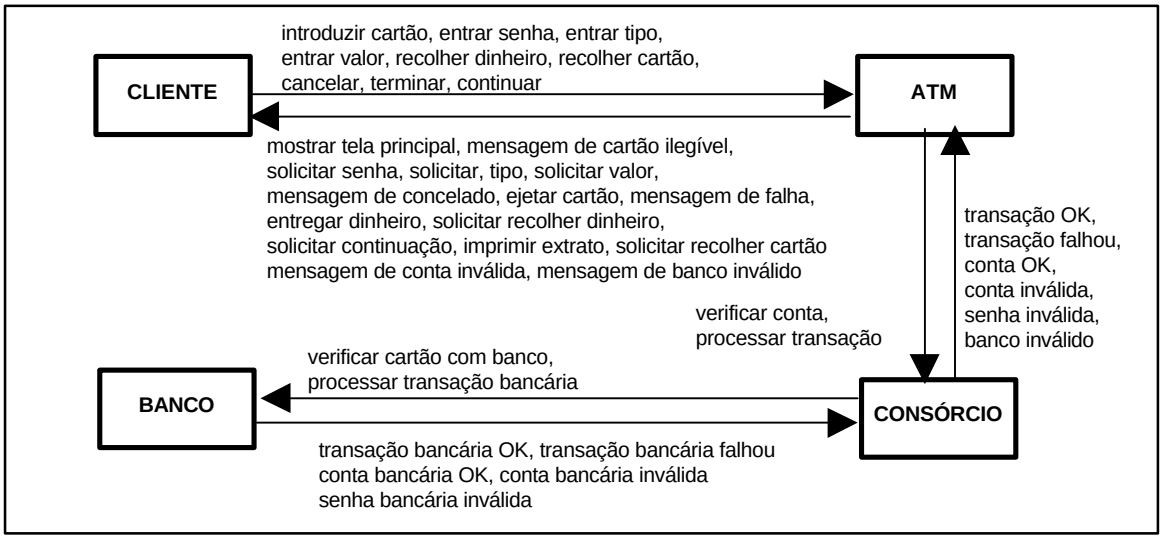


Figura 10.12. Diagrama de fluxo de eventos em ATM [RUM91]

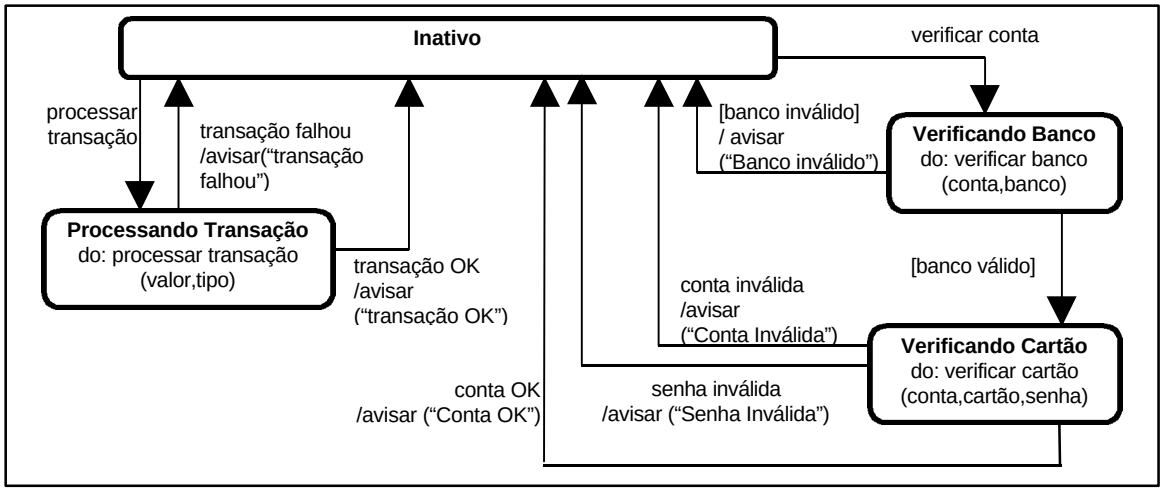


Figura 10.13. Diagrama de estados para a classe Consórcio em ATM [RUM91]

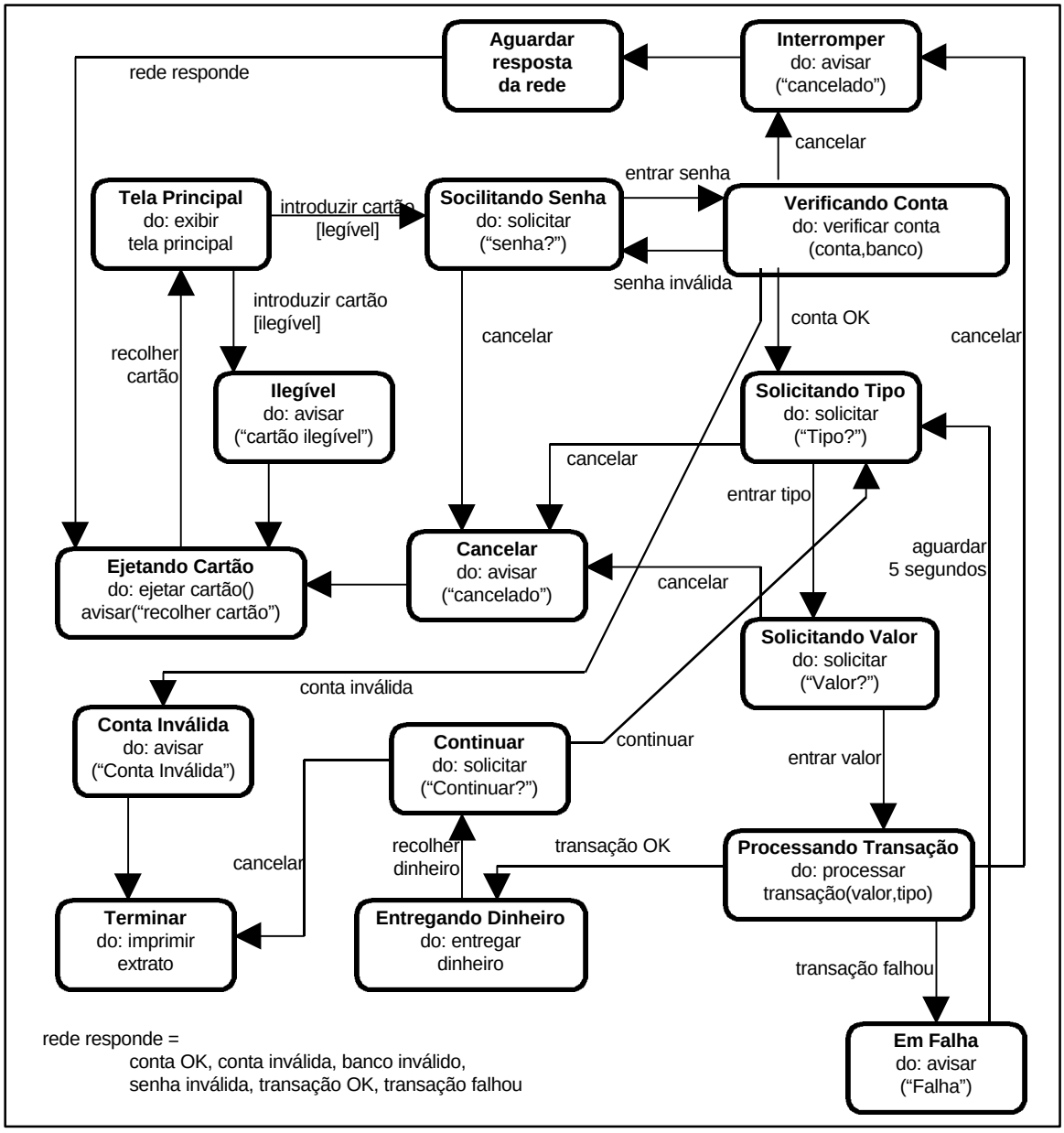


Figura 10.14. Diagrama de estados para a classe ATM [RUM91]

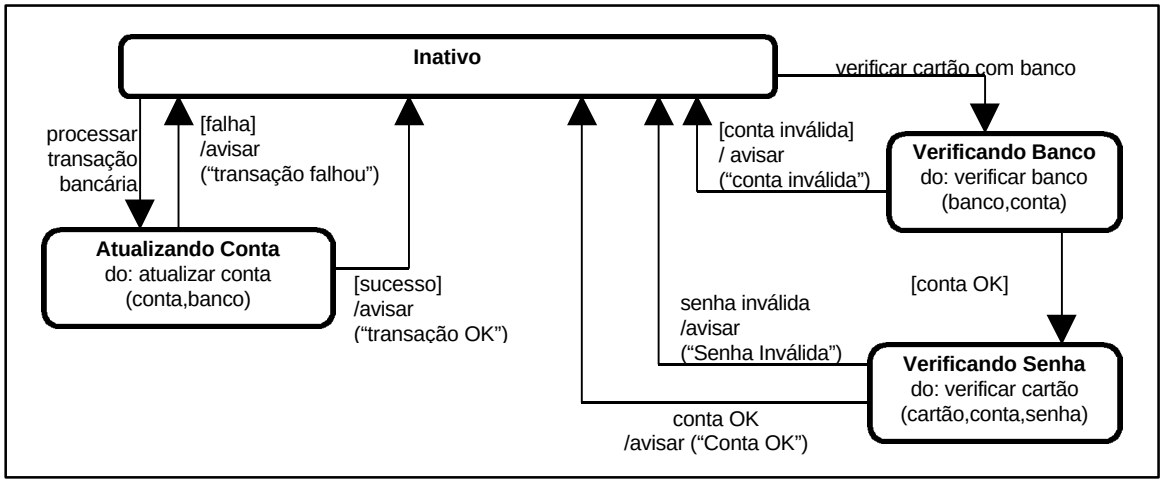


Figura 10.15. Diagrama de estados para a classe Banco em ATM [RUM91]

10.3.5 Modelo Funcional

As figuras 10.16 e 10.17 mostram diagramas de Fluxo de Dados para ATM.

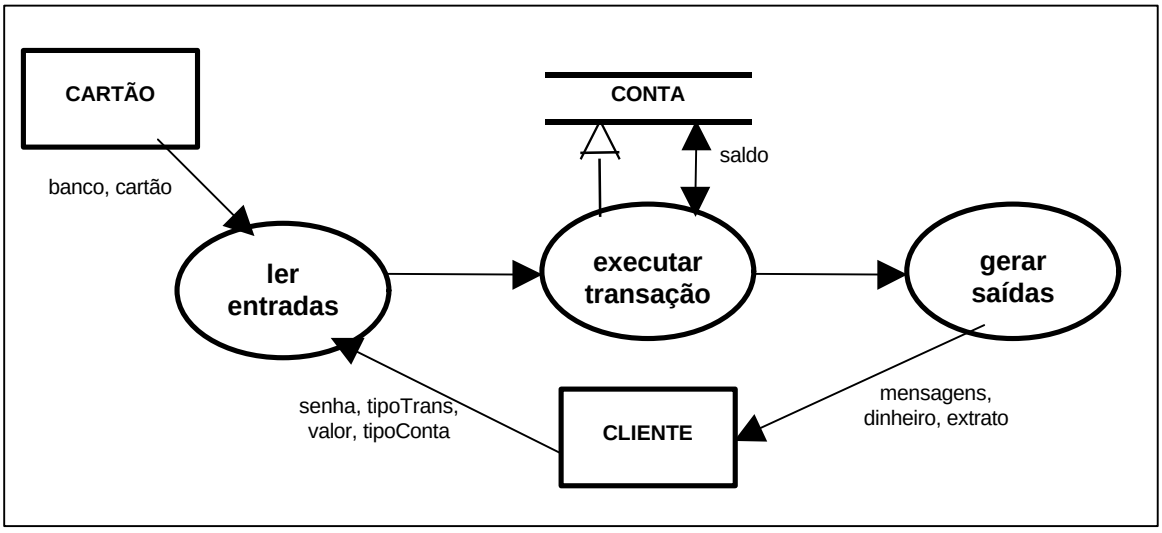


Figura 10.16. Diagrama de fluxo de dados de nível mais elevado para a ATM [RUM91]

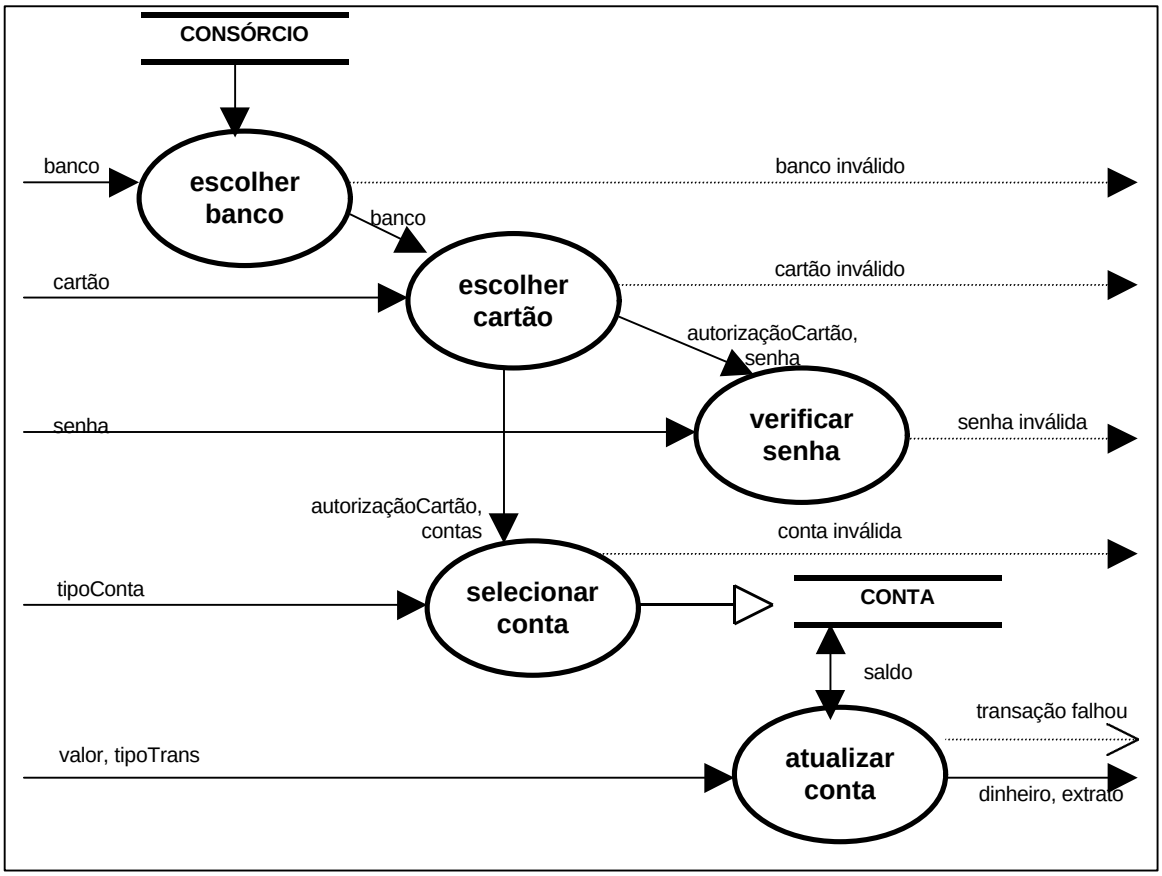


Figura 10.17. Diagrama de fluxo de dados para o processo “executar transação" da ATM [RUM91]

10.3.6 Operações x Processos

Operações implementadas pelos processos-folha referentes ao processo “Executar Transação” são mostradas na tabela 10.4.

PROCESSO-FOLHA	OPERAÇÕES	CLASSE DE OBJETOS
Selecionar Banco	verificar banco(conta,banco)	Consórcio
Selecionar Cartão	verificar cartão(cartão,conta,senha)	Consórcio
	verificar banco(conta,banco)	Banco
Verificar Senha	verificar cartão(cartão,conta,senha)	Banco
Selecionar Conta	verificar conta(conta,banco)	Banco
Atualizar Conta	processar transação(valor,tipo)	ATM
	processar transação(valor,tipo)	Consórcio
	atualizar conta(conta,valor,tipotrans)	Banco
	atualizar conta(conta,valor,tipotrans)	Conta

Tabela 10.4. Operações implementadas no processo “Executar Transação”

10.3.7 Operações x Eventos

A tabela 10.5 mostra os eventos referidos na integração relatada a seguir, com as classes de origem e de destino e as operações-causa e efeito correspondentes.

CLASSE ORIGEM	operação-causa	EVENTO	operação-efeito	CLASSE DESTINO
ATM	verificar conta	verificar conta	verificar banco	Consórcio
Consórcio	verificar banco	banco inválido	ejetar cartão	ATM
Consórcio	verificar senha	conta OK	processar transação	ATM
Banco	selecionar conta	atualizar conta	atualizar conta	Conta

Tabela 10.5. Eventos, operações e classes envolvidas.

10.3.8 Integração dos Modelos

A figura 10.18 mostra as classes de objetos “Consórcio” e “ATM” interagindo através do evento “Banco Inválido”. Esta figura mostra a ausência do evento (ver seta indicadora com a posição marcada por “a” na figura 10.18) na transição de estado de ATM entre os estados “Verificando Conta” e “Ejetando Cartão”. A regra de consistência número 10 é violada por esta ausência. Há um evento (“Banco Inválido”, no original “bad bank code”) recebido por ATM, conforme o diagrama de Fluxo de Eventos, com a operação-efeito “Ejetar Cartão”, que não consta do diagrama de Estados entre “Verificando Conta” e “Ejetando Cartão”.

Na figura 10.18, as indicações marcada por “a” e “b” mostram a transição de estado entre “Verificando Conta” e “Ejetando Cartão”. Em “a” o nome do evento (“Banco Inválido”) deveria indicar sua presença na transição e em “b” colchetes (sem condição) vincularia a transição com o estado posterior, conforme a notação da matriz.

No setor das operações da matriz filtrada, mostrado na figura 10.18, é possível observar a seqüência “Verificar Conta (operação) - Verificar Conta (evento) - Verificar Banco - Banco Inválido - Ejetar Cartão”. Pode-se supor que, se um processo iniciado pela atividade associada a um estado (“Verificando Conta”) terminar com a atividade associada a outro estado (“Ejetando

Cartão”), deve existir uma transição entre eles.

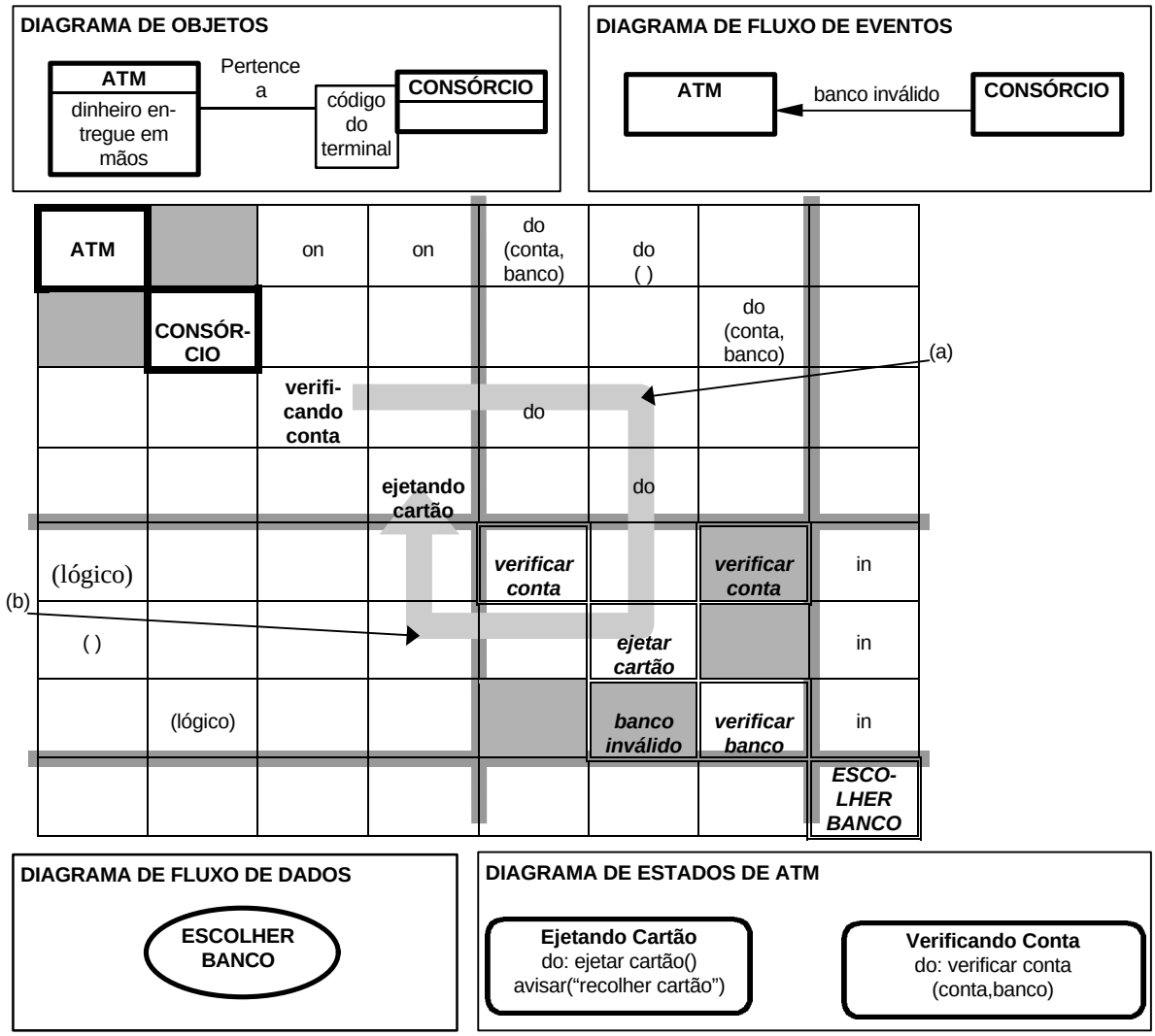


Figura 10.18. Ausência do evento “banco inválido” na transição de estado de ATM

A figura 10.19 mostra a ausência de um fluxo de controle entre os processos “Verificar Senha” e “Atualizar Conta”. A regra de consistência número 22 foi violada com esta ausência. O evento (“Senha OK”) que vincula a operação “Verificar Senha”, implementada pelo primeiro processo, com a operação “Atualizar Conta”, implementada pelo segundo processo, não tem correspondência na dimensão funcional através de um fluxo de controle.

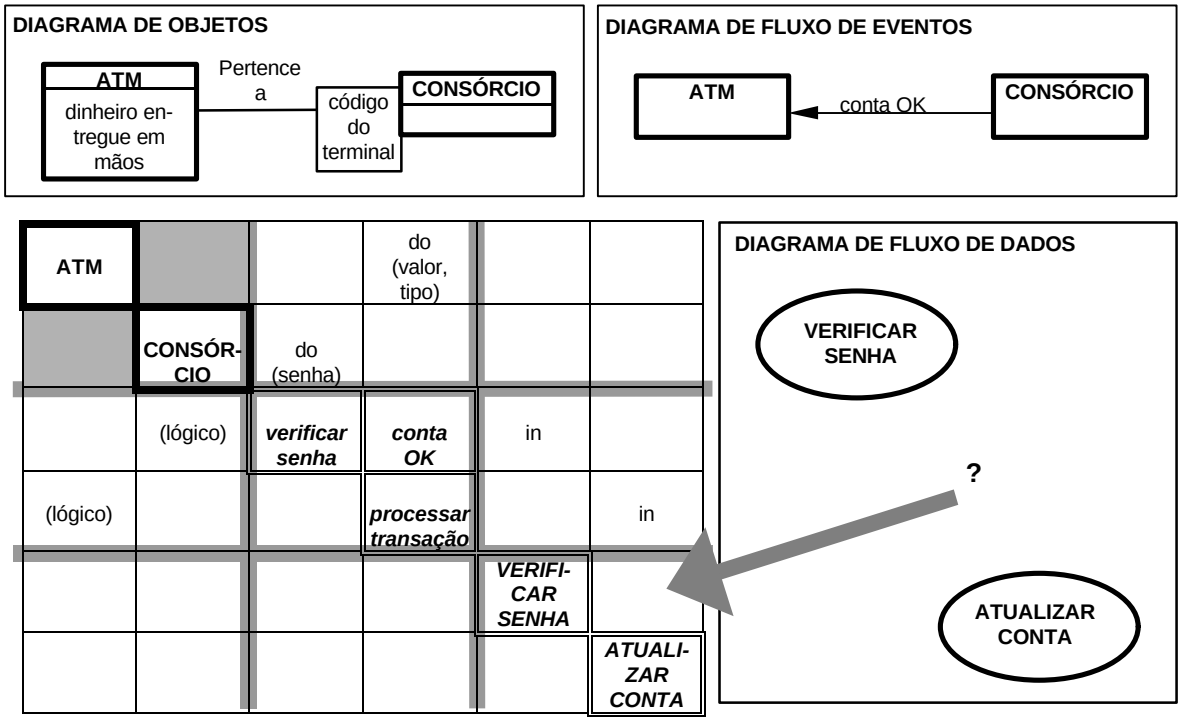


Figura 10.19. Ausência fluxo de controle “senha OK”

A figura 10.20 mostra o evento “Atualizar Conta” entre as operações “Selecionar Conta” e “Atualizar Conta”. Novamente a regra de consistência 22 é violada, pois não existe um fluxo de dados correspondente ao evento (“Atualizar Conta”) que, neste caso, transmite informação indicando qual a conta a ser atualizada.

Pela figura 10.20 observa-se que os fluxos de dados que chegam ao processo “Atualizar Conta” trazem informações do processo “Ler Entradas” e do depósito de dados “Conta”. A partir de “Ler Entradas” chegam o valor e o tipo da transação. Do depósito de dados fica difícil imaginar que seja conhecida a conta a atualizar. Mesmo porque o processo “Selecionar Conta” tem apenas um fluxo de objeto até o depósito de dados “Conta” criando um novo objeto (uma nova conta), conforme a notação da metodologia OMT. Faltaria, então, a indicação da conta selecionada para

que pudesse ser atualizada.

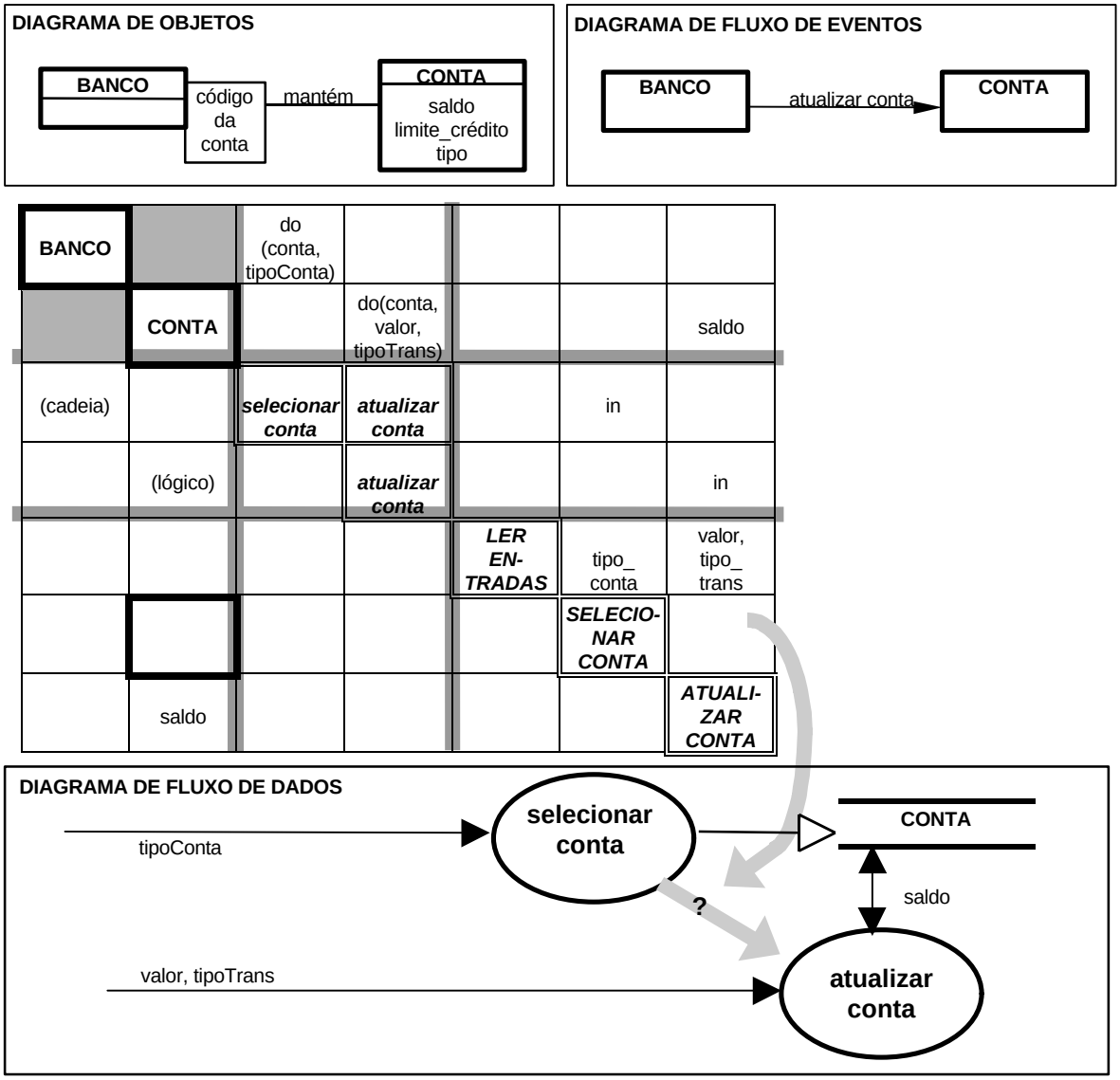


Figura 10.20. Insuficiência de informações para atualizar conta

Observe-se que a operação “Selecionar Conta” tem como entrada a conta a ser atualizada, indicando que esta informação chega ao processo “Selecionar Conta” que a implementa. Se fosse necessário um filtro diferente mostraria que o processo “Selecionar Cartão” passa esta informação para o processo “Selecionar Conta”.

Pela observação da matriz filtrada, na figura 10.20, o evento (“Atualizar Conta”) entre as operações implementadas pelos processos “Selecionar Conta” e “Atualizar Conta” indica a ausência do fluxo de dados entre eles. Note-se que somente o diagrama de Fluxo de Eventos apresentado na figura 10.19 é uma inferência, não sendo mostrado na modelagem original.

10.4 Conclusões

Dois estudos de caso são utilizados para mostrar como o mecanismo de integração proposto integra os modelos gerados, dentro dos limites definidos pela modelagem realizada em cada caso. Além dos objetos da aplicação e dos modelos de Objeto, Dinâmico e Funcional desenvolvidos pelos autores de cada modelagem, são incluídas informações sobre o mapeamento das operações em relação aos processos e aos eventos, conforme é necessário para dar maior eficiência na integração. São mostradas algumas inconsistências encontradas após o uso de filtros e a aplicação das regras de consistência sobre a Matriz Square em cada caso.

Nestes estudos de caso foi utilizada a ferramenta que apoia a Matriz Square e suporta a metodologia OMT, desenvolvida na forma de protótipo para esta dissertação. Ela é descrita no próximo capítulo.

11 PROTÓTIPO DE UMA FERRAMENTA DE APOIO À MATRIZ SQUARE

> "*Ferramentas devem simplificar a construção de diagramas e as entradas em dicionários. Mas, porque a maioria das modelagens envolve iterações e refinamentos, um de seus atributos mais mencionados é a facilidade de revisão.*"
> Rodney Bell [BEL94]

11.1 Introdução

Neste capítulo é comentada a implementação do protótipo de uma ferramenta que apoia a Matriz Square e que é referida a partir de agora como "ferramenta Square". A seção 11.2 informa a linguagem utilizada e a simplificação da notação da metodologia OMT que foi implantada na parte de diagramação. A seção 11.3 apresenta as funções implementadas no protótipo. A seção 11.4 mostra as características do repositório de informações quanto à Tabela de Elementos, ao Dicionário de Dados e à Matriz Square. A seção 11.5 aborda a integração dos modelos assim como está implementada na ferramenta-protótipo, através dos filtros e das regras de consistência.

11.2 O Protótipo e a Diagramação

Para o desenvolvimento deste protótipo da ferramenta Square foi utilizada a linguagem Visual Basic, disponível através do Microsoft Excel, versão 5.0a. Esta linguagem foi escolhida por estar associada à planilha eletrônica Excel. Isto facilitou a utilização da planilha para a construção da matriz Square e também para o desenho de caixas e linhas, necessárias na construção dos diagramas.

Não foi objetivo da implementação conseguir boa performance em termos de velocidade de processamento da atividade diagramática e das funções de tradução e de consistência no protótipo.

A notação da metodologia OMT foi simplificada para facilitar a implementação do diagramador, que não é o objetivo principal da ferramenta. A notação utilizada é apresentada na figura 11.1. A simplificação adotada procurou não incluir alterações significativas na notação da metodologia OMT que pudessem afetar a tradução para o formato matricial, ou seja, que modificassem a notação da matriz Square.

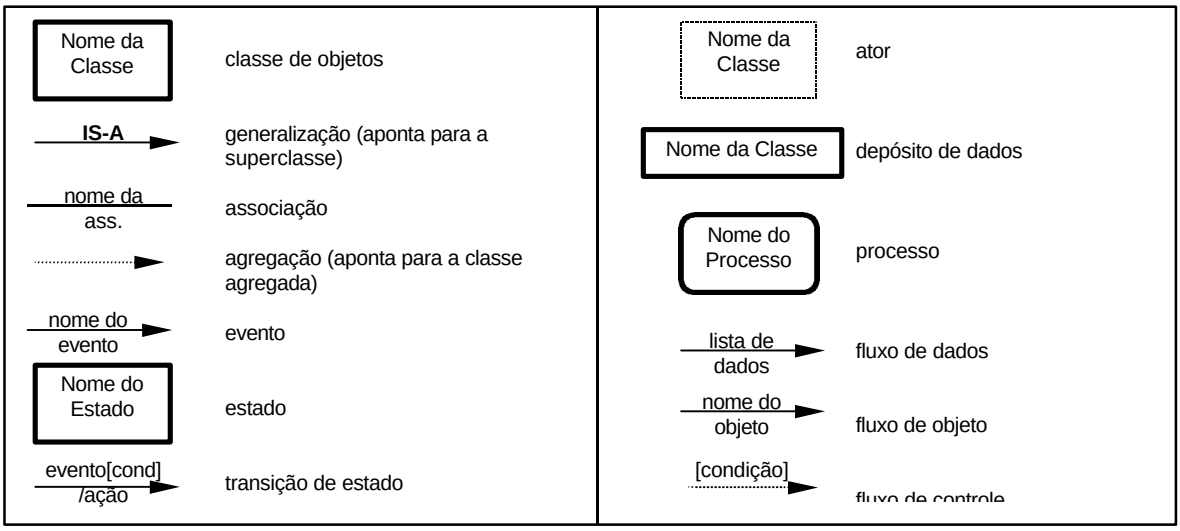


Figura 11.1. Notação OMT simplificada para a ferramenta-protótipo

Figura 11.2. Caixa de diálogo para inclusão de classe de objetos no diagrama de Objetos.

11.3 Funções Implementadas

A ferramenta é organizada em duas pastas de trabalho: uma armazena os diagramas, o dicionário de dados, a tabela de elementos e a Matriz Square, sendo específica para cada sistema modelado, e a outra contém as instruções executáveis da ferramenta. As opções são apresentadas na forma de menus e de janelas de diálogo. A seguir são descritos os elementos da ferramenta implementados no protótipo.

- **Diagramador**. Os elementos são modelados através da entrada das informações referentes a eles em caixas de diálogo como a mostrada na figura 11.2. É destinada uma planilha para cada diagrama: uma para o diagrama de objetos, uma para o diagrama de fluxo de eventos, uma para cada diagrama de estados e uma para cada um dos diagramas de fluxo de dados. Os diagramas de estados são acessados pela seleção de uma classe de objetos no diagrama de fluxo de eventos. Os níveis mais inferiores dos DFDs são acessados pela seleção dos processos neles detalhados.

- **Tradutor**. No momento em que o modelador opta pela entrada de um elemento num diagrama e informa todos os dados relativos a este elemento, estes dados são armazenados na Tabela de Elementos e na matriz Square, com a respectiva tradução, obedecendo as regras sintáticas e, inclusive, algumas regras de consistência que são imediatamente aplicadas.
- **Dicionário de Dados**. Um dicionário de sinônimos é implementado para realizar uma "amarração" simples entre termos diferentes com o mesmo significado. Ele é acionado sempre que um elemento de dado é informado na diagramação.
- **Tabela de Elementos**. Armazena informações para localização do elemento no diagrama correspondente e na matriz Square.
- **Matriz Square**. É construída numa planilha conforme as regras sintáticas decorrentes de sua definição.
- **Filtros**. Estão disponíveis através de menus. No momento em que o modelador escolhe um filtro e o elemento filtrante correspondente, fragmentos da matriz Square podem ser visualizados. Não será implementada a visualização de fragmentos de diagramas juntamente com a matriz filtrada.
- **Mecanismo de Consistência**. Atua durante a atividade diagramática realizando consistência durante a entrada de dados e também na solicitação de relatório de consistência.

11.4 Repositório de Informações

11.4.1 Tabela de Elementos

A tabela 11.1 mostra, como exemplo, um trecho da Tabela de Elementos assim como está implementada na ferramenta-protótipo. O exemplo é retirado do estudo de caso sobre a preparação de conferência de trabalhos da IFIP.

	Square		DO		DFE		DE			DFD		
Identificador	**V1**	**V2**	**L**	**C**	**L**	**C**	**Nome**	**L**	**C**	**Nome**	**L**	**C**
CC1			11	3	11	3						
CC2			11	12	11	12						
CE1							DE-ARTIGO	3	9			
CE2							DE-ARTIGO	7	8			
CE3							DE-ARTIGO	11	7			
CE4							DE-ARTIGO	11	13			
CE5							DE-ARTIGO	16	5			
CC3			5	12	5	12						
ARTIGO.distribuir												
ARTIGO.incluir												
ARTIGO.incluir_autor												
ARTIGO.inclusão_falha												
ARTIGO.inclusão_ok												
ARTIGO.ver_autores												
AUTOR.autor_falha												
AUTOR.autor_ok												
AUTOR.incluir												
AUTOR.inclusão_falha												
AUTOR.inclusão_ok												
AUTOR.verificar												
ARTIGO.verificar												
CP7										DFD-INCLUIR ARTIGO	12	3
CP6										DFD-INCLUIR ARTIGO	12	8
CP5										DFD-INCLUIR ARTIGO	12	14
CP4										DFD-INCLUIR ARTIGO	3	14
CP3										DFD-INCLUIR ARTIGO	3	8
CP2										DFD-ADMINISTRAR ARTIGO	3	9
CP1										DFD-0	9	3
!!!												
Le1	CC1	CC2			14	4						
Le2	CC2	CC3			8	11						
CC1	X									DFD-0	9	13
Lf1	CC1	CP1								DFD-0	7	8
CC1	X									DFD-ADMINISTRAR ARTIGO	14	2
Lf2	CC1	CP2								DFD-ADMINISTRAR ARTIGO	6	4
Lf3	CP2	CC1								DFD-ADMINISTRAR ARTIGO	4	2
CC3	D									DFD-ADMINISTRAR ARTIGO	3	15
Lf4	CP2	CC3								DFD-ADMINISTRAR ARTIGO	2	12
Lf5	CP2	CC3								DFD-ADMINISTRAR ARTIGO	4	12
Lt3	CE1	CE2					DE-ARTIGO	5	7			
Le4	CC2	CC1			15	4						

Tabela 11.1. Um exemplo de Tabela de Elementos

Os títulos das colunas da tabela 11.1 são a seguir descritos:

- **Identificador**: código que identifica cada elemento-vértice e cada elemento-arco modelados nos diagramas e representados na matriz Square. É gerado automaticamente pela ferramenta com a seguinte composição: uma primeira letra, que pode ser C (identificando uma “caixa”, isto é, um elemento-vértice) ou L (identificando uma “linha”, isto é, um elemento-arco); uma segunda letra, que identifica o elemento (C = classe de objetos, E = estado, P = processo, f = fluxo de dados, ...) e um número, que é acrescentado seqüencialmente durante a diagramação. As operações, como não são vértices nos diagramas, têm como identificador o nome da classe de objetos responsável e o seu nome, separados por ponto.
- **Square**: informação em duas colunas para os elementos-arco contendo os identificadores dos dois elementos-vértice, **V1** e **V2**, ligados por eles. Para os elementos-vértice a coluna **V1** pode conter X (ator) ou D (depósito de dados) indicando que o elemento-vértice já foi modelado como classe de objetos.
- **DO**: localização do elemento no diagrama de Objetos através da linha (**L**) e da coluna (**C**).
- **DFE**: localização do elemento no diagrama de Fluxo de Eventos através da linha (**L**) e da coluna (**C**).
- **DE**: localização do elemento nos diagramas de Estados através do nome do diagrama (**Nome**), da linha (**L**) e da coluna (**C**).
- **DFD**: localização do elemento nos diagramas de Fluxo de Dados através do nome do diagrama (**Nome**), da linha (**L**) e da coluna (**C**).

11.4.2 Dicionário de Dados

A tabela 11.2 mostra, como exemplo, um trecho do Dicionário de Dados assim como está implementado na ferramenta-protótipo. O exemplo é retirado do estudo de caso sobre a preparação de conferência de trabalhos da IFIP.

A seguir são descritos os títulos das colunas da tabela 11.2:

- **Identificador**: identificador (conforme a Tabela de Elementos) do elemento onde está presente o dado considerado.
- **Nome**: nome do dado, como é visível para o modelador.
- **Referência**: identificador (conforme a Tabela de Elementos) do elemento onde está presente um dado cujo nome é considerado como um sinônimo para aquele considerado. Os identificadores iniciados por "L" têm, como segundo dígito, o valor 1. Isto ocorre apenas para facilitar a implementação. Não há maiores implicações já que para o usuário isto é invisível.
- **Sinônimo ou Tipo**: nome identificado na referência como sinônimo ou, no caso de não haver sinônimo, o tipo do dado considerado.

Identificador	Nome	Referência	Sinônimo ou Tipo
	cadeia		
	inteiro		
	real		
	data		
	lógico		
	objeto		
CC2	assunto		cadeia
CC2	data_envio_artigo		data
CC2	estado		cadeia
CC2	título		cadeia
CC3	nome		cadeia
L1f1	artigo	CC2	título
L1f1	artigos_aceitos	CC2	título
L1f1	autores	CC3	nome
L1f1	avaliadores		cadeia
L1f1	sessão		cadeia
L1f2	artigo	CC2	título
L1f2	autores	CC3	nome
L1f3	[inclusão_artigo_ok]		lógico
L1f4	autor		objeto
L1f5	dados_autor		cadeia
L1f6	artigo	CC2	título
L1f6	autores	CC3	nome
L1f7	autores	CC3	nome

Tabela 11.2. Um exemplo de Dicionário de Dados

Os tipos de dados (cadeia, inteiro, real, data, lógico, objeto) são incluídos no dicionário para aparecerem na listagem oferecida ao usuário ao inserir um dado no Dicionário de Dados. Através desta listagem, o usuário pode referenciar o dado a inserir com tipo ou com sinônimo. Referenciando com sinônimo estará indiretamente indicando o tipo.

O Dicionário de Dados, na ferramenta-protótipo, é utilizado apenas em três momentos: (1) na inclusão do dado no dicionário; (2) na seleção de um dado como elemento filtrante para um filtro por comunicação e (3) para pesquisar, neste mesmo filtro, um atributo de uma classe de objetos.

11.4.3 Matriz Square

A matriz Square ocupa uma planilha na pasta (arquivo) referente ao sistema que está sendo modelado. Esta é a única planilha de trabalho do usuário em que são utilizados os limites de células.

A matriz utiliza diversos recursos disponíveis na planilha eletrônica Excel. Sempre que um elemento novo é modelado e representado na planilha, são usados os recursos para inserir linha e coluna. O protótipo não prevê a eliminação de um elemento diagramado. Os filtros são facilitados pelos comandos de ocultamento de linhas e colunas da planilha. Bordas, sombreamentos, fontes para letras (por exemplo: classes de objetos aparecem em negrito, operações em negrito e itálico, ...) e outras formatações também são recursos diretamente ativados através da planilha eletrônica.

Todo o repositório de informações (Tabela de Elementos, Dicionário de Dados e Matriz) pode ser salvo em disco como um arquivo independente. Assim podem ser trabalhadas aplicações diferentes pela ferramenta e a qualquer momento retomadas para alterações.

11.5 Integração dos Modelos

O protótipo permite a verificação visual, através dos filtros, e a aplicação das regras de consistência. A tabela 11.3 mostra os filtros que são selecionados através de menus e estão implementados no protótipo. A partir de cada diagrama o usuário ativa filtros específicos.

diagrama	**filtro**
de Objetos	por ligação, por responsabilidade e por comunicação
de Fluxo de Eventos	por evento, por comunicação, por ligação e por responsabilidade
de Estados	por evento, por estado e por comunicação
de Fluxo de Dados	por processo e por comunicação

Tabela 11.3. Filtros implementados no protótipo

Após a seleção de um filtro, são ocultas as linhas e as colunas adequadas e é apresentada a matriz filtrada. Não está implementada no protótipo a apresentação de fragmentos de diagramas.

Também não está implementada a sobreposição de filtros. Por outro lado, o usuário pode solicitar a exclusão das linhas e colunas que julgar necessário na matriz filtrada, indicando um ou mais elementos da diagonal principal.

Durante a atividade diagramática, em diversos momentos são ativadas algumas regras de consistência, conforme é mostrado a seguir:

- Uma classe de objetos ao ser modelada no diagrama de Objetos, é modelada também no diagrama de Fluxo de Eventos, automaticamente (regra 1)
- Só é possível incluir um evento no diagrama de Fluxo de Eventos se existir relacionamento entre as classes de objetos envolvidas (regra 2).
- Não há entrada de atributos na caixa de diálogo para inclusão de evento no diagrama de Fluxo de Eventos. Assume-se, por sintaxe, a correspondência com retornos e argumentos das operações-causa e efeito (regra 4 e 5).
- Ao incluir uma atividade associada a um estado, o usuário seleciona a partir da lista de operações da classe de objetos correspondente (regra 7).
- Ao incluir uma ação de uma transição de estados, o usuário seleciona a partir da lista de operações da classe de objetos correspondente (regra 8)
- Ao incluir um evento numa transição de estados, o usuário seleciona a partir da lista de eventos recebidos pela classe de objetos correspondente (regra 9).
- Ao incluir um depósito de dados ou um ator num DFD, o usuário seleciona a partir da lista das classes de objetos modeladas. (regra 11 e 12)

Desta forma, não são aplicadas na entrada de dados da atividade diagramática: (a) poucas regras de consistência definidas entre os modelos de Objetos e Dinâmico, (b) a maioria das definidas entre os modelos de Objetos e Funcional e (c) todas as definidas entre os modelos Dinâmico e Funcional. Algumas destas regras são, no protótipo, utilizadas para a geração do relatório de consistência, quando solicitado pelo usuário.

A tabela 11.4 mostra um exemplo de relatório de consistência, sobre uma modelagem parcial da preparação de conferência de trabalhos da IFIP, listando eventos sem operação-causa ou efeito (regra 3), eventos ausentes de transições de estados (regra 10), classes não participantes de processos (regra 13), operações são implementadas por processos (regra 14) e processos sem

operações implementadas (regra 14).

R E L A T Ó R I O D E C O N S I S T Ê N C I A
==========
O evento INCLUSÃO da classe COMITÊ_DE_PROGRAMA para a classe ARTIGO não possui operação-causa
O evento INCLUSÃO_ARTIGO_FALHA da classe ARTIGO para a classe COMITÊ_PROGRAMA não possui operação-efeito
O evento INCLUSÃO_ARTIGO_OK da classe ARTIGO para a classe COMITÊ_PROGRAMA não possui operação-efeito
O evento VERIFICAR da classe COMITÊ_DE_PROGRAMA para a classe ARTIGO não possui operação-causa
==========
O evento VERIFICAR recebido por ARTIGO não está presente no diagrama de Estados
==========
A classe COMITÊ_DE_PROGRAMA não participa de nenhum processo
A operação DISTRIBUIR da classe ARTIGO não é implementada por nenhum processo
==========
O processo LISTAR não implementa nenhuma operação
==========
========== f i m d o r e l a t ó r i o ==========

Tabela 11.4. Exemplo de relatório de consistência

Algumas mensagens do relatório são justificáveis e o usuário as recebe apenas como advertência. No exemplo da tabela 11.4, a classe “Comitê-Programa” gera diversas mensagens justificadas porque representa uma interferência externa ao sistema e, por isto, não possui operações definidas na modelagem.

11.6 Conclusões

Uma ferramenta de apoio à Matriz Square deve ser uma ferramenta CASE que suporte a metodologia OMT [RUM91] incluindo a atividade diagramática e a construção de um repositório de informação com: dicionário de dados, Tabela de Elementos e Matriz Square. O protótipo desenvolvido utilizou uma notação simplificada e, por isto, diferente em alguns aspectos em relação àquela da metodologia OMT original. Foi tomado o cuidado, entretanto, para que estas diferenças não desfigurassem a metodologia. O protótipo gera um dicionário de dados mínimo, uma Tabela de Elementos e a própria Matriz Square, que é o núcleo da presente proposta de integração. Podem ser testados o uso de filtros sobre a matriz, de modo simples e sem sobreposição, e a aplicação de boa parte das regras de consistência. O protótipo foi utilizado nos estudos de caso vistos no capítulo anterior e permitiu a detecção das inconsistências que foram lá apresentadas. Como este mecanismo de integração depende muito de uma ferramenta que automatize seu uso, é justo esperar que, com uma ferramenta completa, os estudos de caso apresentados fossem mais profundamente examinados.

12 CONCLUSÕES FINAIS

> "*Sempre que uma tal contradição (que vem desmentir um mundo de concepções) é sentida com força e intensidade, experimentamos uma reação decisiva na maneira de interpretar o mundo.*"
> Albert Einstein

12.1 Contribuições para a Integração dos Modelos da OMT

A construção e utilização do modelo matricial são aqui propostos como uma contribuição para a integração dos modelos da fase de análise gerados pela metodologia OMT. A consistência e a coerência entre os modelos podem ser verificadas visualmente através de filtros sobre a Matriz Square ou pela aplicação de regras de consistência. O mecanismo de integração proposto necessita de uma ferramenta que o apoie e que suporte a metodologia OMT, permitindo a verificação de consistência existente entre seus modelos.

As sugestões de consistência apresentadas, direta ou indiretamente, pelos autores da metodologia OMT são adotadas e transformadas em postulados que fundamentam a proposta aqui apresentada para integração dos seus modelos.

As contribuições desta dissertação são apresentadas a seguir:

- **Postulados**. São relacionados 12 postulados que definem fundamentos para a integração entre os modelos da OMT, tomando como base aquilo que é estabelecido pelos autores da metodologia. Estes postulados são independentes do mecanismo de integração proposto.
- **Conexões explícitas**. São definidas classes de conexões explícitas, que servem de base para o mecanismo proposto para a integração dos modelos da metodologia OMT. Sete destas conexões explícitas centram-se em operações, tomadas como agente integrador principal. Uma oitava conexão considera a identidade de classes, atores e depósitos de

dados, vinculando-os por nome. As conexões são instanciadas através de elementos integradores; alguns deles (eventos, condições e fluxos de dados e de objeto) pertencem à metodologia e outros (vínculos de responsabilidade, de evento, de estado, de atividade, de interação e de implementação) são incluídos pelo mecanismo de integração proposto, sendo modelados indiretamente a partir da atividade diagramática.

- **Matriz Square**. É especificada uma ferramenta gráfica, a Matriz Square, que complementando os diagramas tradicionais da metodologia OMT, constitui o núcleo do mecanismo integrador dos seus modelos. É desenvolvida a formalização que define a Matriz Square e estabelece suas características.
- **Ferramenta de apoio**. É caracterizada uma ferramenta para automatizar a construção da Matriz Square, suportando a metodologia OMT. É desenvolvido um protótipo de uma ferramenta com estas características implementando algumas de suas funções.
- **Integração por filtros**. São definidos filtros aplicáveis sobre a Matriz Square, permitindo a alteração do nível de abstração, para a consistência visual dos modelos da metodologia OMT. É discutido o uso destes filtros, abordando a seleção e a sobreposição dos mesmos, para a verificação de padrões de modelagem e de cenários. A aplicação de filtros possibilita a integração dos elementos modelados localmente, isto é, verificando fragmentos da matriz.
- **Regras de consistência**. São definidas regras aplicáveis sobre o modelo matricial para verificação de consistência, visando a integração dos modelos da metodologia OMT de forma coerente. A aplicação de regras de consistência possibilita a integração dos elementos modelados globalmente, isto é, verificando toda a matriz.

12.2 Dificuldades para a Integração dos Modelos da OMT

As dificuldades que o mecanismo proposto encontra para obter a integração dos modelos da OMT referem-se ao grau de consistência possível. Isto significa que o grau de consistência é diretamente proporcional:

- às definições de implementação de operações por processos. A melhor situação seria a relação um processo-folha - uma operação. Quanto mais operações sem processos vinculados e quanto mais processos-folha sem operações vinculadas menor o grau de consistência possível.

- às definições de operações-causa e efeito em relação a eventos. A melhor situação seria a relação uma operação-causa - um evento - uma operação-efeito. Quanto mais eventos sem operações associadas menor o grau de consistência possível.
- ao aumento da disciplina na modelagem. Quanto maior o detalhamento e o volume de informações, seguindo as diretrizes da modelagem OMT e atendendo inclusive os itens anteriores, maior o grau de consistência possível.
- à diminuição da flexibilidade na modelagem. Quanto mais a atividade diagramática for executada com flexibilidade, permitindo que o modelador cometa pequenos desvios em relação ao que é preestabelecido pelas regras de sintaxe e de consistência, menor o grau de consistência possível.

Um grande volume de informações, em termos de nível de detalhamento, pode ser bom para a integração, isto é, para a verificação de consistência, mas ruim para a comunicação do modelo e, em conseqüência, para o seu entendimento. Entre a maior disciplina e a maior flexibilidade deve-se encontrar o grau de consistência possível ideal. Os objetivos da fase de análise, deste modo, serão cumpridos com qualidade, construindo modelos adequadamente integrados e precisos.

> "*Por certo, sempre é possível criar maus projetos nos quais os três modelos estejam tão interligados que não possam ser separados, mas um bom projeto isola os diferentes aspectos de um sistema e limita o acoplamento entre eles.*" [RUM91]

12.3 Verificação das Conexões Explícitas

A integração dos Modelos da OMT é buscada pelo mecanismo proposto através da verificação de conexões explícitas existentes entre eles. Estas conexões são instanciadas por elementos integradores que são incluídos pela modelagem direta ou indiretamente.

A tabela 12.1 mostra como são verificadas as classes de conexões em cada correlação entre os modelos de Objeto (MO), Dinâmico (MD) e Funcional (MF), localmente através de filtros sobre a matriz ou, globalmente, pela aplicação de regras de consistência. São excluídos da relação de cada conexão explícita os filtros e regras que pouco contribuem para a verificação ou não verificam efetivamente a referida conexão.

		conexões explícitas							
Modelos	**Modo**	**de Respon-sabilidade**	**de Ligação**	**de Estado**	**de Comu-nicação**	**de Controle por Evento**	**de Controle por Condição**	**de Processo**	**de Classe**
MO x MD	Filtros	todos	2, 4, 5	3, 5, 6	1 a 4	todos	1, 3, 5		todos
	Regras	3, 7 a 10	2	7 a 10	4, 5	3 a 5, 9, 10	6		1
MO x MF	Filtros	todos	2,4,7		1,2, 4,7			1,2, 4,7	todos
	Regras	13 a 19	17, 18		15, 16, 18, 19			13 a 19	11, 12, 19
MD x MF	Filtros	todos		3, 5, 6	1 a 4, 7	todos	1, 3, 5, 6, 7	todos	todos
	Regras			21, 22	23, 24	20, 23, 24	22	20, 21, 23, 24	

Tabela 12.1. Conexões explícitas verificadas entre modelos por filtros e regras de consistência
(Filtros: 1=por Responsabilidade, 2=por Ligação, 3=por Estado,
4=por Comunicação, 5=por Evento, 6=por Condição e 7=por Processo)

12.4 Alterações Propostas para a Metodologia OMT

Segundo os autores da OMT [RUM91], o objetivo da fase de análise é desenvolver um modelo do que o sistema irá fazer e que é expresso em termos de objetos e relacionamentos, fluxos de controle dinâmico e transformações funcionais. Para isto são previstas originalmente as seguintes etapas [RUM91]:

- **Etapa 1**. Escrever ou obter uma descrição inicial do problema (enunciado do problema)
- **Etapa 2**. Construir um Modelo de Objetos.
  - Identificar as classes de objetos.
  - Iniciar a geração de um dicionário de dados contendo descrições de classes, atributos e associações.
  - Acrescentar as associações entre classes.
  - Acrescentar os atributos para objetos e as ligações.
  - Organizar e simplificar as classes de objetos utilizando herança.
  - Testar os caminhos de acesso utilizando cenários e repetir os passos anteriores, se necessário.
  - Agrupar classes em módulos, com base em estreito acoplamento e em similaridade de função.

  Resultado: Modelo de Objetos = diagrama de Objetos + dicionário de dados.

- **Etapa 3**. Construir um Modelo Dinâmico.

- Preparar cenários das seqüências típicas de interação
- Identificar eventos entre objetos e preparar uma seqüência de eventos para cada cenário.
- Preparar um diagrama de Fluxo de Eventos.
- Desenvolver um diagrama de estados para cada classe que tenha comportamento dinâmico importante.
- Verificar a consistência e a completeza dos eventos compartilhados pelos diagramas de Estados.

Resultado: Modelo Dinâmico = diagramas de Estado + diagrama de Fluxo de Eventos.

- **Etapa 4**. Construir um Modelo Funcional.
  - Identificar os valores de entrada e saída.
  - Utilizar diagramas de Fluxo de Dados quando necessário para mostrar dependências funcionais.
  - Descrever o que cada função faz.
  - Identificar restrições.
  - Especificar os critérios de otimização.

  Resultado: Modelo Funcional = diagramas de Fluxo de Dados + restrições
- **Etapa 5**. Verificar, repetir e refinar os três modelos.
  - Acrescentar ao modelo de Objetos as operações-chave que foram descobertas durante a preparação do modelo funcional.
  - Verificar se classes, associações, atributos e operações estão consistentes e completos no nível escolhido de abstração. Comparar os três modelos com o enunciado do problema e conhecimentos relevantes do domínio. Testar os modelos utilizando cenários.
  - Desenvolver cenários detalhados (incluindo condições de erro) como variações dos cenários básicos. Utilizar cenários do tipo “o que - se” para verificar mais detalhadamente os três modelos.
  - Repetir as etapas anteriores tantas vezes quantas forem necessárias para completar a análise.

  Resultado: Documentos da Análise = Enunciado do Problema + Modelo de Objetos + Modelo Dinâmico + Modelo Funcional.

O mecanismo de integração proposto prevê alterações na metodologia OMT. Ao utilizar os recursos da Matriz Square através de uma ferramenta apropriada, que a apoie e que suporte a OMT, o modelador dever executar as seguintes etapas:

- **Etapa 1**. Escrever ou obter uma descrição inicial do problema (enunciado do problema).
- **Etapa 2**. Construir um Modelo de Objetos.
  - Atividades originais, com ênfase na geração do dicionário de dados
- **Etapa 3**. Construir um Modelo Dinâmico.
  - Atividades originais, com as seguintes alterações e inclusões:
    - Informar para cada evento as suas operações-causa e efeito.
    - Testar cenários elaborados pelo modelador com os padrões obtidos em filtros por Responsabilidade, por Ligação e por Evento.
    - Verificar as colaborações entre classes em filtros por Ligação, detectando classes sem colaborações e/ou vínculos de evento sem associações ou agregações (e vice-versa).
    - Testar a consistência e completeza dos eventos compartilhados pelos diagramas de Estados, em filtros por Evento.
- **Etapa 4**. Construir um Modelo Funcional.
  - Atividades originais, com as seguintes alterações e inclusões:
    - Informar a(s) operação(ões) implementada(s) em cada processo-folha.
    - Testar fluxos de dados em filtros por Comunicação, por Evento e por Processo.
    - Testar fluxos de controle em filtros por Estado, por Evento, por Condição e por Processo.
    - Verificar a participação de classes em processos através de filtros por Responsabilidade e por Processo.
- **Etapa 5**. Verificar, repetir e refinar os três modelos.
  - Executar atividades originais, aplicando filtros e regras de consistência sobre a Matriz Square, com as seguintes inclusões:
    - Verificar a colaboração entre classes com a participação delas em processos detalhando conexões de responsabilidade, de ligação e de processo.

- Verificar a coerência dos estados válidos para as classes em relação à participação delas em processos detalhando conexões de responsabilidade, de estado, de controle por evento, de controle por condição e de processo.
- Verificar a coerência dos dados estruturados e manipulados detalhando conexões de responsabilidade, de comunicação, de controle por evento e de processo.

12.5 Trabalhos Futuros

A proposta é apresentada de forma aberta possibilitando diversos trabalhos futuros que são exemplificados a seguir:

- Detalhamento de padrões de modelagem subdivididos em conjuntos adequados para classes de aplicações diferentes. Os padrões encontrados em sistemas em tempo real provavelmente sejam diferentes daqueles presentes em sistemas para um gerenciador de transações ou para simulação dinâmica, por exemplo.
- Utilização do “setor de projeto” provavelmente com aplicação na fase de projeto durante a seqüência do desenvolvimento de um sistema. A inclusão das informações referentes à fase de projeto talvez possa ser apoiada pela Matriz Square, dando continuidade ao desenvolvimento. No “setor de projeto” podem ser representados detalhamentos referentes aos métodos que implementam as operações dos objetos modelados, como controles de decisão e repetição.
- Detalhamento da organização e do uso do dicionário de dados apoiando a Matriz Square, de forma adequada às suas necessidades.
- Utilização de padrões de modelagem no processo de reutilização de especificações. Quando o usuário reutiliza especificações, ele realiza uma pesquisa levando em conta seu problema corrente tentando reaver soluções similares. A pesquisa recupera soluções em termos de um conjunto de componentes que interagem, tendo um certo grau de similaridade com o problema corrente [KAR89]. Esta comparação talvez possa ser viabilizada através do uso de padrões. Um padrão desejado seria procurado entre as especificações já realizadas e disponíveis, através da similaridade entre conexões explícitas.
- Implementação completa de uma ferramenta que apoie a Matriz Square.

Referências Bibliográficas

[BAI89] BAILIN, S. An Object-Oriented Requirements Specification Method. **Communications of ACM**, v.32 , n.5, p.608-623. Maio 1989.

[BEL94] BELL, R. Choosing Tools for Analysis and Design. **IEEE Software**, p.121-125. Maio 1994.

[BOO86] BOOCH, G. Object-Oriented Development. **IEEE Trans. Software Engineering**, v.12, n.2, p.211-21. 1986.

[BRO92] BROWN, A.W.; PENEDO, M.H. An Annotated Bibliography on Integration in Software Engineering Environments. **ACM SIGSOFT**, **Software Engineering Notes**, v.17, n.3, p.47-55. Julho 1992.

[BRU91] BRUNET, J. Modeling the World with Semantic Objects. In: **Object Oriented Approach in Information System**, F.Van Assche, B.Moulin, C.Rolland (Editores), North-Holland, Elsevier Science Publishers B.V, p.361-379.. 1991.

[BRU92] BRUEGGE, B.; BLYTHE, J.; JACKSON, J.; SHUFELT, J. Object-Oriented System Modeling with OMT. **ACM**, **OOPSLA'92 conference proceedings**, p.359-376. Setembro 1992.

[CAS87] CASANOVA, M.A.; GIORNO, F.A.C.; FURTADO, A.L. **Programação em Lógica e a Linguagem Prolog**. São Paulo, Ed. Edgard Blücher. 454 p. 1987

[CHA91] CHAMPEAUX, D.; AMERICA, P.; COLEMAN, D.; DUKE, R.; LEA, D.; LEAVENS, G. Formal Techiniques for OO Software Development (Panel). **OOPSLA'91 conference proceedings**, Addison-Wesley Publishing Company, p.166-170. Maio 1991.

[COA92] COAD, P. Object-Oriented Patterns. **Communications of the ACM**, v.15, n.9. Setembro 1992.

[COA92a] COAD, P.; YOURDON, E. **Análise Baseada em Objetos**. Trad. da 2ª edição americana, editora Campus, Série Yourdon Press. 1992.

[DED94] DEDENE, B.; SNOECK, M. M.E.R.O.DE.: A Model-Driven Entity-Relationship Object-Oriented Development Method. **ACM SIGSOFT**, **Software Engineering Notes**, v.19, n.3, p.51-61. Julho 1994.

[DEM89] DeMARCO, T. **Análise Estruturada e Especificação de Sistema**. Trad. de Maria Beatriz Gomes Soares Veiga de Carvalho, Rio de Janeiro, ed.Campus. 1989.

[EMB92] EMBLEY, D.; KURTZ, B.; WOODFIELD, S. **Object-Oriented System Analysis: A Model-Driven Approach**. Englewood Cliffs, Prentice-Hall. 1992.

[FER86] FERREIRA, A.B.H. **Novo Dicionário da Língua Portuguesa** (Aurélio). 2ª ed., ed. Nova Fronteira, Rio de Janeiro, 1839 p. 1986.

[FIS90] FISCHER, A.S. **CASE: Utilização de Ferramentas para Desenvolvimento de Software**. Rio de Janeiro, ed. Campus, 264 p. 1990.

[GHE91] GHEZZI, C.; JAZAYERI, M.; MANDRIOLI, D. **Fundamentals of Software Engineering**. Prentice-Hall International, Inc. 1991.

[GON96] GONZALEZ, M.; ORTH, A.I. Uma Ferramenta de Apoio à Metodologia OMT para Integração de Modelos. **Panel 96 - XXII Conferência Latinoamericana de Informática**. Colômbia. Junho 1996

[HAY91] HAYES, F.; COLEMAN, D. Coherent Models for Object-Oriented Analysis. **OOPSLA'91 conference proceedings**, p.171-183. 1991.

[HEN90] HENDERSON-SELLERS, B.; EDWARDS, J.M. The Object-Oriented Systems Life Cycle. **Communications of the ACM**, v.33, n.9. Setembro 1990.

[HEN93] HENDERSON-SELLERS, B.; EDWARDS, J.M. The O-O-O Methodology for the

Object-Oriented Life Cycle. **ACM SIGSOFT**, **Software Engineering Notes**, v.18, n.4, p.54-60. Outubro 1993.

[JAN94] JANKOWSKI, D,J. The Feasibility of CASE Structured Analysis Methodology Support. **ACM SIGSOFT**, **Software Engineering Notes**, v.19, n.2, p.72-82. Abril 1994.

[KAR89] KARAKOSTAS, V. Requirements for CASE Tools in Early Software Reuse. **ACM SIGSOFT**, **Software Engineering Notes**, v.14, n.2. Abril 1989.

[KOR90] KORSON, T.D.; MCGREGOR, J.D. Understanding Object-Oriented: A Unifying Paradigm. **Communications of the ACM**, v.33, n.9, p.40-60. Setembro 1990.

[LOY93] LOY, P.; STAPP, Y. DFD's vs. N^2Charts. **ACM SIGSOFT**, **Software Engineering Notes**, v.18, n.3. Julho 1993.

[MAC74] MACIEL, J. **Elementos de Teoria Geral dos Sistemas**. Pedtrópolis, editora Vozes Ltda. 1974.

[MAR88] MARTIN, C.F. Second-Generation CASE Tools: A Challenge to Vendors. **IEEE Software**, p.46-49. Março 1988.

[MIC94] Comp. Melhoramentos de SP. Michaelis. **Dicionário Inglês-Português Português-Inglês**. São Paulo. 1994.

[MON92] MONARCHI, D.D.; PUHR, G.I. A Research Typology for Object-Oriented Analysis and Design. **Communications of the ACM**, v.35, n.9, p.35-47. Setembro 1992.

[NER92] NERSON, J. Applying Object-Oriented Analysis & Design. **Communications of the ACM**, v.35, n.9, p.63-74. Setembro 1992.

[NOB88] NOBLE, B.; DANIEL, J.W. **Applied Linear Algegra**. New Jersey, Prentice-Hall, Englewood Cliffs. 1988.

[OLL82] OLLE, T.W. IFIP Comparative Review of Information Systems Design Methodologies: Problem Definition. In: Olle, T.W. et al (eds.). **Information Systems Design Methodologies:** A Comparative Review. Amsterdam: North-Holland, 648p. 1982.

[RAM91] RAMACKERS, G.J.; VERRIJN-STUART, A.A. Integrating Information System Perspectives with Objects. In: **Object Oriented Approach in Information System**. F.Van Assche, B.Moulin, C.Rolland (Editores), North-Holland, Elsevier Science Publishers B.V. 1991.

[REN82] RENTSCH, T. Object-Oriented Programming. **SIGPLAN Notices**, v.17, n.9, p.51-57. Setembro 1982.

[REI94] REINOSO, G.B. Aplicação da Técnica de Análise Object Modeling Technique (OMT) ao Problema da IFIP. **Relatório de pesquisa**, CPGCC da UFRGS. Junho 1994

[REI94a] REINOSO, G.B.; HEUSER, C.A. Comparando o Processo de Modelagem de técnicas de Análise Orientada a Objetos. Curitiba, **8º Simpósio Brasileiro de Engenharia de Software**, anais, p.177-193. 1994.

[RUB92] RUBIN, K.; GOLDBERG, A. Object Behavior Analysis. **Communications of the ACM**, v.35, n.9, p.48-62. Setembro 1992.

[RUM91] RUMBAUGH, J; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. **Object-Oriented Modeling and Design**. Englewood Cliffs, ed. Prentice-Hall, 500 p. 1991.

[SHL88] SHLAER, S.; MELLOR, S.J. **Object-Oriented Systems Analysis: Modeling the World in Data**. Prentice-Hall. 1988.

[SOM92] SOMMERVILLE, J. **Software Engineering**. 4ª edição, Addison-Wesley Publishing Co, 649 p. 1992.

[STE72] STEINBRUCH, A. **Álgebra Linear e Geometria Analítica**. Ed. McGraw-Hill. São Paulo. 1972.

[SZW84] SZWARCFITER, J. **Grafos e Algoritmos Computacionais**. Rio de Janeiro, ed. Campus. 1984.

[WAN89] WAND, Y. A Proposal for a Formal Model of Objects. In: **Object-Oriented Concepts, Databases, and Applications**. Kin, W.; Lochovsky, F.H. (editores), New York, ACM PRESS, p-537-559. 1989

[WIR90] WIRFS-BROCK, R; WILKERSON, B.; WIENER, L. **Designing Object-Oriented Software**. New Jersey, Prentice-Hall, Inc. Englewood Cliffs, 341 p. 1990.