

FERNANDA MARIA RIBEIRO DE ALENCAR

**MAPEANDO A MODELAGEM ORGANIZACIONAL
EM ESPECIFICAÇÕES PRECISAS**

**Tese apresentada à Coordenação da
Pós-Graduação em Ciência da
Computação do Centro de Ciências Exatas
e da Natureza da Universidade Federal de
Pernambuco, como parte dos requisitos
para a obtenção do título de Doutor em
Ciência da Computação.**

Orientador: Jaelson Freire Brelaz de Castro

Recife

Dezembro 1999

A **DEUS**

“INTELIGÊNCIA SUPREMA,
CAUSA PRIMÁRIA DE TODAS AS COISAS”,

A

IGOR NICHOLAI , ÍTALO DIMITRY

MEUS FILHOS,

E A

JOSÉ ROBERTO

MEU MARIDO

AGRADECIMENTOS

Meus agradecimentos, em primeiríssimo lugar, são endereçados a DEUS, por mais essa oportunidade que me proporcionou.

Em segundo lugar, meu especial agradecimento à minha família, nas pessoas de meus dois filhos, ainda bebês, que tanto têm sofrido pelas minhas constantes ausências, em razão dos esforços por concluir esse trabalho, e ao meu companheiro, amigo, revisor, meu marido, que nestes anos tem me incentivado sempre a continuar. Aos meus pais e avó, dedico homenagem e agradecimento. Em especial, a memória de meu pai que, ainda na minha infância para adolescência, me orientava para o aprofundamento nos estudos. Agradeço à minha mãe pela dedicação, esforço e colaboração, sem a qual a realização deste trabalho teria sido muito mais difícil. A minha avó pelas preces e torcida.

Agradeço especialmente aos colegas Victor Santander, pelas horas dedicadas à ajuda de revisão; Marco Toranzo, Márcio Bueno e Márcia Cabral, pela paciente atenção à exposição da proposta do trabalho de Tese, e às suas observações; a Gilberto Cysneiros Filho, que contribuiu para mudanças no rumo de nosso trabalho, colaborando conosco nos estudos realizados; a Fabrízia Souza, pela parceria em diversos trabalhos publicados, mas principalmente, pela força impulsionadora e atenção num momento tão importante; aos meus dois companheiros de início de jornada, mas eternos amigos, Thalles Cerqueira de Mello e Ana Cristina de Carvalho Moura Cordeiro; aos demais colegas, que direta ou indiretamente contribuíram para a minha escalada.

Aos colaboradores do Departamento de Informática, agradeço inicialmente ao Prof. Sílvio Lemos Meira, pela oportunidade que foi dada de ingressar no programa de doutorado, e hoje, passados alguns anos, à Profa. Teresa Ludemir, que em muito contribuiu para nosso engrandecimento, mesmo que talvez desconheça esse fato. Não poderia esquecer, os inúmeros momentos que “torramos a paciência” das secretárias da

pós-graduação do Departamento de Informática, Maria Lília Freitas e Ivoneide Ribeiro, às quais dedico meu apreço.

Agradeço aos colegas do Departamento de Eletrônica e Sistemas, em particular ao Prof. Odilon Maroja por sua colaboração e ao Prof. Frederico Dias Nunes, pelo exemplo de “garra” e por ter acreditado e me dado acolhida, num momento difícil e crucial para meu futuro, promovendo a tranquilidade necessária ao trabalho.

Aos membros da Banca de Defesa da Proposta de Tese - Prof. Alexandre Marcos Lins de Vasconcelos, Prof. Daltro José Nunes, Prof. Edson Costa de Barros Carvalho Filho e Prof. Francisco de Assis Cartaxo Pinheiro -, agradeço pela participação e pelas informações disponibilizadas para o enriquecimento deste trabalho.

Em especial, meu agradecimento à amiga Andréa Tenório Pinto, pelo apreço e estímulo, pela participação, força e torcida para o sucesso desse processo.

Lembro e agradeço às muitas outras pessoas que também deram sua parcela de colaboração, não havendo aqui, infelizmente, condições de a todos nominar.

Porém, meu AGRADECIMENTO MAIOR é direcionado ao meu orientador, Prof. Jaelson Freire Brelaz de Castro, que antes de tudo, sabe ser GENTE, independente dos títulos que já possua e dos que venha a possuir. Agradeço a amizade deste SER HUMANO espetacular, de enorme coração. Agradeço pela oportunidade de conviver e aprender com o EXCELENTE PROFISSIONAL, haja visto o reconhecimento que lhe é dado pela comunidade científica nacional e internacional. Graças a Deus tive a sorte de tê-lo em meu caminho, orientando-me, lutando comigo para conseguirmos nossos objetivos, participando sempre. Torço para que o vínculo que construímos permaneça no futuro. Não poderia ter tido melhor orientador.

A todos, MEU MUITO OBRIGADO e que Deus os abençoe, retribuindo o carinho e a energia renovadora que a mim dispensaram.

SUMÁRIO

	Pag.
LISTA DE FIGURAS	x
LISTAS DE TABELAS	xv
LISTAS DE ABREVIACÕES	xvi
RESUMO	xviii
ABSTRACT	xx
 CAPÍTULO I - INTRODUÇÃO	
1.1 – <u>CONTEXTO</u>	01
1.2 – <u>MOTIVAÇÃO DO TRABALHO</u>	03
1.3 – <u>OBJETIVOS E ABORDAGEM</u>	08
1.4 – <u>ESTRUTURA DO TRABALHO</u>	11
 CAPÍTULO II - A ENGENHARIA DE REQUISITOS	
2.1– <u>PRINCIPAIS CONCEITOS DA ENGENHARIA DE REQUISITOS</u>	13
2.1.1 – DEFINIÇÕES	13
2.1.2 – MOTIVAÇÕES PARA O ESTUDO DA ENGENHARIA DE REQUISITOS	16
2.1.3 – ATIVIDADES DO PROCESSO DA ENGENHARIA DE REQUISITOS	20
2.2 – <u>CLASSIFICAÇÃO DOS REQUISITOS</u>	26
2.3 – <u>A MODELAGEM DE REQUISITOS</u>	29
2.3.1 – TÉCNICAS PARA A MODELAGEM DE REQUISITOS FUNCIONAIS	30
2.3.1.1 – <u>A Técnica de Modelagem Unificada – UML</u>	32
2.3.2 - TÉCNICAS PARA A MODELAGEM DE REQUISITOS NÃO FUNCIONAIS	36

2.4 – <u>PROBLEMAS EM ABERTO NA ÁREA DE ENGENHARIA DE REQUISITOS</u>	37
--	----

<u>CONSIDERAÇÕES</u>	38
----------------------------	----

CAPÍTULO III – TÉCNICAS DE MODELAGEM ORGANIZACIONAL

3.1– <u>PRINCIPAIS TÉCNICAS DE MODELAGEM DE REQUISITOS ORGANIZACIONAIS</u>	41
--	----

3.2 – <u>A TÉCNICA DE MODELAGEM ORGANIZACIONAL I*</u>	46
---	----

3.2.1 – O MODELO DE DEPENDÊNCIAS ESTRATÉGICAS – (SD)	50
--	----

3.2.1.1 – <u>Construtores da Representação</u>	55
--	----

3.2.1.2 – <u>Aspectos de Precisão do Modelo de Dependências Estratégicas (SD)</u>	58
---	----

3.2.1.2.1 - Aspectos de Precisão com Relação ao Ator de quem se Depende	63
---	----

3.2.1.2.2 - Aspectos de Precisão com Relação ao Ator Dependente	64
---	----

3.2.2 – O MODELO DE RAZÕES ESTRATÉGICAS – (SR)	68
--	----

3.2.2.1 - <u>Construtores da Representação</u>	74
--	----

3.2.2.2 – <u>Aspectos de Precisão do Modelo de Razões Estratégicas – (SR)</u>	76
---	----

<u>CONSIDERAÇÕES</u>	80
----------------------------	----

CAPÍTULO IV - ESPECIFICAÇÃO PRECISA DE REQUISITOS

4.1 – <u>PRINCIPAIS CONCEITOS</u>	83
---	----

4.2 – <u>OS MÉTODOS DE ESPECIFICAÇÃO PRECISA</u>	89
--	----

4.2.1 – MÉTODOS BASEADOS EM PROPRIEDADES	90
--	----

4.2.2 – MÉTODOS BASEADOS EM MODELOS	92
---	----

4.3 – <u>AS LINGUAGENS PRECISAS UTILIZADAS NO DESENVOLVIMENTO PROPOSTO</u>	95
--	----

4.3.1 – A LÓGICA MODAL DE AÇÕES – MAL	95
---	----

4.3.1.1 – <u>A Linguagem MAL na Versão Original</u>	96
4.3.1.2 – <u>A Linguagem Structured MAL</u>	99
4.3.2 - A LINGUAGEM DE RESTRIÇÕES DE OBJETOS - OCL	112
<u>CONSIDERAÇÕES</u>	122

CAPÍTULO V – MAPEAMENTO DA MODELAGEM ORGANIZACIONAL COM A ESPECIFICAÇÃO PRECISA DE REQUISITOS

5.1 – <u>CONTEXTO DO PROBLEMA</u>	125
5.2 – <u>MECANISMOS DE MAPEAMENTO</u>	131
5.2.1 – DIRETRIZES DE MAPEAMENTO DE I* PARA MAL ESTRUTURADO	133
5.2.2 - DIRETRIZES DE MAPEAMENTO DE I* PARA UML/OCL	169
5.3 – <u>RESULTADOS ALCANÇADOS</u>	195
5.3.1 - LIMITAÇÕES DOS RESULTADOS DO MAPEAMENTO DA TÉCNICA I* PARA A LÓGICA MODAL DE AÇÕES ESTRUTURADA	196
5.3.2 – LIMITAÇÕES DOS RESULTADOS DO MAPEAMENTO DA TÉCNICA I* PARA A LINGUAGEM DE RESTRIÇÕES DE OBJETOS (OCL)	197
<u>CONSIDERAÇÕES</u>	198

CAPÍTULO VI - ESTUDO DE CASOS

6.1 – <u>EXEMPLOS DE DESCRIÇÃO DO MAPEAMENTO ENTRE A TÉCNICA I* E A LINGUAGEM STRUCTURED MAL</u>	201
6.1.1 – ENVASADORA DE ÁGUA MINERAL	201
6.1.2 – SISTEMA BANCÁRIO DE TRANSFERÊNCIA POR TELEFONE COM SEGURANÇA POR CRIPTOGRAFIA	225
6.2 – <u>EXEMPLO DE DESCRIÇÃO DO MAPEAMENTO ENTRE A TÉCNICA I* E A LINGUAGEM OCL</u>	248

6.2.1 – LOJA DE VENDA DE CD'S POR ENCOMENDA	248
<u>CONSIDERAÇÕES</u>	271
 CAPÍTULO VII - CONCLUSÕES	
7.1 – <u>SUMÁRIO DOS RESULTADOS</u>	273
7.2 – <u>TRABALHOS RELACIONADOS</u>	279
7.3 - <u>CONTRIBUIÇÕES</u>	280
7.4 – <u>FUTUROS DIRECIONAMENTOS</u>	281
 <u>REFERÊNCIAS BIBLIOGRÁFICAS</u>	 284

LISTA DE FIGURAS

	Pag.
Capítulo II - A ENGENHARIA DE REQUISITOS	
Figura 2.1 - Aspectos Motivadores ao Estudo da Engenharia de Requisitos	17
Figura 2.2 - Percentuais de Erros Referentes às Diferentes Fases do Desenvolvimento de Software	18
Figura 2.3 - Percentuais de Custo com a Correção de Erros Relativos às Diferentes Fases do Desenvolvimento de Software	19
Figura 2.4 - Custo de Correção de Um Erro Durante a Especificação dos Requisitos	19
Figura 2.5 - Entradas e Saídas do Processo da Engenharia de Requisitos	21
Figura 2.6 - Atividades do Processo da Engenharia de Requisitos	23
Figura 2.7 - Cenário Típico de Caso de Uso para a Loja de Venda de CD's, em UML	33
Figura 2.8 - Diagrama de Classes da Loja de Venda de CD's com Agregação e Generalização	34
Capítulo III - TÉCNICAS DE MODELAGEM ORGANIZACIONAL	
Figura 3.1 - Estrutura Organizacional e Cadeia de Valores	44
Figura 3.2 - Modelo de Dependências Estratégicas (SD) da Loja de Venda de CD's	54
Figura 3.3 - Esquema Parcial para o Modelo de Dependências Estratégicas (SD) da Loja de Venda de CD's por Encomenda	57
Figura 3.4a - Ligações de Decomposição de Tarefas	69
Figura 3.4b - Ligações Meio-Fim	69
Figura 3.5 - Modelo de Razões Estratégicas (SR) da Loja de Venda de CD's	70
Figura 3.6 - Esquema Parcial para o Modelo de Razões Estratégicas (SR) em TELOS	75

Capítulo IV -ESPECIFICAÇÃO PRECISA DE REQUISITOS

Figura 4.1 -	Abordagem para o Refinamento da Especificação até o Programa Executável	85
Figura 4.2 -	Sintaxe das Especificações em Structured MAL	103
Figura 4.3 -	Sintaxe da Definição de Tipo em Structured MAL	103
Figura 4.4 -	Estruturação Gráfica de parte do Sistema de Controle da Venda de CD por Encomenda	109
Figura 4.5 -	Escopo Geral de Especificação em Structured MAL	110
Figura 4.6 -	Definição do Agente “Registro de Encomenda de CD” em MAL	111
Figura 4.7 –	Diagrama de Classes UML da Loja de Venda de CD’s	115

Capítulo V - MAPEAMENTO DA MODELAGEM ORGANIZACIONAL COM A ESPECIFICAÇÃO PRECISA DE REQUISITOS

Figura 5.1 -	Modelo de Dependências Estratégicas da Loja de Vendas de CD’s	135
Figura 5.2 -	Mapeamento de Atores em i*, em Agentes Structured MAL	139
Figura 5.3 -	Definição em Structured MAL da Loja de Venda de CD’s com seus Agentes Componentes	140
Figura 5.4 -	Modelo de Razões Estratégicas para a Loja de Venda de CD’s	141
Figura 5.5 -	Especificando Tarefas, Modelo SD, como Ações Compartilhadas em Structured MAL	144
Figura 5.6 -	Especificação de Tarefa Interna, Modelo SR, em Ação Local, em Structured MAL	145
Figura 5.7 -	Especificando Recursos em i*, como Agentes em Structured MAL	147
Figura 5.8 -	Especificando Objetivos e Objetivos-Soft em i* como Atributos do Tipo Booleano e Enumerado Compartilhados, em Structured MAL	151
Figura 5.9 -	Especificando Objetivos e Objetivos-Soft em i* como Atributos do Tipo Booleano e Enumerado Locais, em Structured MAL	152

Figura 5.10 -	Um Relacionamento Externo Representado em MAL Estruturado através de Axiomas	157
Figura 5.11 -	As Ligações de Decomposição de Tarefas em i*, Especificadas em Structured MAL através de Axiomas com Conectivos AND e Operadores Deônticos	161
Figura 5.12 -	As Ligações Meio-Fim em i*, Especificadas em Structured MAL através de Axiomas com Conectivos OR e Operadores Deônticos	167
Figura 5.13 -	Classes em pUML correspondentes aos Atores em i* do Modelo SD (Sistema Empresa de Venda de CD)	173
Figura 5.14 -	Composição em i*, Mapeada como Agregação de Classe, em UML	174
Figura 5.15 -	Tarefas Mapeadas em Operações de Classe em UML	178
Figura 5.16 -	Recurso em i*, Mapeado em uma Classe UML	179
Figura 5.17 -	Objetivos e Ojetivos-Soft Mapeados como Atributos Booleanos	184
Figura 5.18 -	Aplicação de Pré e Pós Condições, em OCL	185
Figura 5.19 -	Relacionamento de Decomposição de Tarefa em i*, Expresso por Pós-Condições em OCL	189
Figura 5.20 -	Relacionamento Meio-Fim em i*, onde Meio e Fim são Tarefas	192
Figura 5.21 -	Ligação Meio-Fim Representada como Invariante de Classe em OCL	195
 Capítulo VI - ESTUDO DE CASOS		
Figura 6.1 -	Organograma Setorial da Envasadora de Água Mineral	202
Figura 6.2 -	Modelo de Dependências Estratégicas da Envasadora de Água Mineral	204
Figura 6.3 -	Definição em Structured MAL da de Água Mineral com seus Agentes Componentes	207
Figura 6.4 -	Modelo de Razões Estratégicas da Envasadora de Água Mineral	208
Figura 6.5 -	Especificando Tarefas em i*, da Envasadora de Água Mineral, como Ações em Structured MAL	211
Figura 6.6 -	Especificando Recursos em i*, da Envasadora de Água Mineral, como Agentes em Structured MAL	213

Figura 6.7 -	Especificando Objetivos e Objetivos-Soft em i*, da Envasadora de Água Mineral, como Atributos Booleanos e Enumerados, em Structured MAL	215
Figura 6.8 -	Um Relacionamento Externo Representado em MAL através de Axiomas	216
Figura 6.9 -	Axiomas para as Ligações de Decomposição de Tarefa do Agente Inspeção, da Envasadora de Água Mineral	218
Figura 6.10 -	Axiomas para as Ligações de Decomposição de Tarefa do Agente Limpeza_Externa, da Envasadora de Água Mineral	219
Figura 6.11 -	Axiomas para as Ligações Meio-Fim do Agente Inspeção, da Envasadora de Água Mineral	222
Figura 6.12 -	Axiomas para as Ligações Meio-Fim do Agente Limpeza_Externa, da Envasadora de Água Mineral	224
Figura 6.13 -	Modelo de Dependências Estratégicas de um Sistema Bancário de Transferência por Telefone	225
Figura 6.14 -	Modelo de Dependências Estratégicas de um Sistema Bancário de Transferência por Telefone com um Centro Autorizador	227
Figura 6.15 -	Definição do Agente Sistema Bancário	228
Figura 6.16 -	Modelo de Razões Estratégicas para o Sistema Bancário de Transferência por Telefone	229
Figura 6.17 -	Especificando Tarefas em i*, dos atores Cliente, Caixa e Sistema_Bancário_por_Telefone como Ações em Structured MAL	231
Figura 6.18 -	Recursos do Sistema Bancário Mapeados em Agentes em Structured MAL	233
Figura 6.19 -	Especificando Objetivos e Objetivos-Soft em i*, para o Sistema Bancário, como Atributos Booleanos, em Structured MAL	235
Figura 6.20 -	Relacionamento entre Agentes em i*, de um Sistema Bancário, Expresso por Axiomas em Structured MAL	236

Figura 6.21 -	As Ligações de Decomposição de Tarefa em i*, para o Sistema Bancário, Especificadas em Structured MAL	239
Figura 6.22 -	As Ligações de Meio-Fim para o Agente Cliente, do Sistema Bancário, Especificadas em Structured MAL	247
Figura 6.23 -	Modelo de Dependências Estratégicas para a Loja de Venda de CD's	250
Figura 6.24 -	Esboço Inicial do Diagrama de Classes, em UML, para a Loja de Venda de CD's	251
Figura 6.25 -	Modelo de Razões Estratégicas para a Loja de Venda de CD's	252
Figura 6.26 -	Tarefas em i*, mapeadas em Operações sobre as Classes	253
Figura 6.27 -	Recursos em i*, mapeados como Classes em UML	255
Figura 6.28 -	Objetivos e Objetivos-Soft em i*, mapeados como Atributos do Tipo Booleano e Enumerado	257
Figura 6.29a -	Relacionamento de Decomposição de Tarefa em i*, Expresso por Restrições em OCL, para a Classe Loja	260
Figura 6.29b -	Relacionamento de Decomposição de Tarefa em i*, Expresso por Restrições em OCL, para a Classe Atendente	261
Figura 6.29c -	Relacionamento de Decomposição de Tarefa em i*, Expresso por Restrições em OCL, para a Classe Apoio	263
Figura 6.30a -	Relacionamento Meio-Fim em i*, Expresso por Restrições em OCL, para a Classe Loja	267
Figura 6.30b -	Relacionamento Meio-Fim em i*, Expresso por Restrições em OCL, para a Classe Atendente	268
Figura 6.30c -	Relacionamento Meio-Fim em i*, Expresso por Restrições em OCL, para a Classe Apoio	269
Figura 6.31 -	Restrições em OCL sobre Classe Loja	270

LISTA DE TABELAS

	Pag.
Capítulo II - A ENGENHARIA DE REQUISITOS	
Tabela 2.1 - Tabela Resumida com os Principais Modelos Semi-Precisos	31
Capítulo IV - ESPECIFICAÇÃO PRECISA DE REQUISITOS	
Tabela 4.1 - Operações Básicas sobre Coleções	116
Tabela 4.2 - Mapeamento de Tipos, Atributos e Associações OCL	121
Tabela 4.3 - Definição do Mapeamento μ	122

LISTA DE ABREVIACÕES

ALBERT	Agent-oriented Language for Building and Eliciting Real-Time Requirements
BON	Better Object Notation ou Business Object Notation
CASE	Computer Aided Software Engineering
CBD	Component Based Development
CCS	Calculus of Communication Systems
CIM	Computer Integrated Manufacture
CORE	Controlled Requirements Expression
CSP	Communication Sequential Processes
DFD	Data Flow Diagram
EI	Enterprise Integration
ER	Entity-Relation Diagrams
ESPRIT	European Strategic Programme on Research in Information Technology
FOREST	Formal Requirements Specification Techniques
GG	Goal-Goal
GT	Goal-Task
IEEE	Institute of Electrical and Electronics Engineers
JSD	Jackson System Development
KAOS	Knowledge Acquisition in autOmated Specification
LOTOS	Language of Temporal Ordering Specification
MA	Modelo de Atores
MAL	Modal Action Logic
MAU	Modelo de Atividades e Uso
MC	Modelo de Conceitos
MIT	Massachusetts Institute of Technology

MO	Modelo de Objetivos
MooZ	Modular Object Oriented Z
MRSI	Modelo de Requisitos do Sistema de Informações
NASA	National Aeronautics and Space Administration
NATURE	Novel Approaches to Theories Underling Requirements Engineering
OCL	Object Constraint Language
OMT	Object Oriented Modeling Technique
OOA	Object Oriented Analysis
OOSE	Object-Oriented Software Engineering
ORDIT	Organizational Requirements Definition for Information Technology
PIN	Personal Identification Number
RLP	Requirements Language Processor
RT	Resource-Task
SADT	Structured Analysis and Design Technique
SASS	Structured Analysis and System Specification
SD	Strategic Dependency
SDL	Specification and Description Language
SR	Strategic Rationale
SRD	Structured Requirements Definition
SREM	Software Requirements Engineering Methodology
SS	Soft-Goal – Soft-Goal
SSADM	Structured Systems Analysis and Design Methodology
SSM	Software Systems Methodology
TT	Task-Task
UML	Unified Modeling Language
VDM	Vienna Development Method
VSCS	Very Structured Common Sense

RESUMO

A tendência atual da modelagem de sistemas é encontrar metodologias que enfoquem aspectos relativos ao contexto onde estes estarão inseridos, tratando questões relacionadas com estratégia empresarial, funções e dados. Assim, sob a ótica da Administração, a formação dos profissionais de negócio é falha. Faltam-lhes a compreensão mais detalhada dos objetivos, metas, estruturação da empresa, bem como das funções de seus empregados, de nada adiantando processadores fabulosos e vastas redes de comunicação e informação, usados para automatizar modos de trabalho obsoletos ou incoerentes. Por outro lado, sob a ótica da Ciência da Computação, também é preciso que os profissionais desenvolvedores de sistemas de informática tenham a compreensão do contexto organizacional no qual estes sistemas estarão e como poderão ser úteis aos processos do negócio. As técnicas tradicionais da Engenharia de Requisitos não contemplam diretamente os aspectos organizacionais. Hoje existem técnicas específicas para a modelagem organizacional, que exteriorizam os requisitos relacionados com os atores envolvidos na organização, seus relacionamentos, intenções e razões existentes, determinantes de alternativas estratégicas para o negócio.

Importância imprescindível deve ser dada às descrições de requisitos, para que sejam mais precisas e se possa garantir a corretude e consistência daquilo que foi especificado. Surge assim, a necessidade da formalização dos requisitos, para que seja possível provar-se propriedades e possibilitar animações, por exemplo. Todavia, as técnicas tradicionais de formalização preocupam-se com os aspectos da funcionalidade dos sistemas, também não tratando dos aspectos fundamentalmente organizacionais.

Neste trabalho, estamos propondo diretrizes para o mapeamento dos modelos organizacionais, que explicitam os requisitos organizacionais em componentes estruturais das linguagens de especificação precisas. Objetivamos com esta proposta acrescentar às especificações precisas, os componentes organizacionais que lhes faltem. Fizemos uso a técnica de modelagem i^* , para levantar os aspectos organizacionais, e as linguagens Structured MAL (Structured Modal Action Logic) e OCL (Object Constraint Language) associada com UML (Unified Modeling Language), para a modelagem

precisa. Como forma de ratificar a aplicabilidade das diretrizes propostas, apresentamos três estudos de casos.

ABSTRACT

Nowadays the trend is to find methodologies that focus on to the context where the systems are inserted, analyzing issues related to business strategy, functions and data. From the point of view of administration, the background of business professionals lack a deep understanding of the objectives, company structure and role of employees. And it is worthless to use high processing and communication and information networks to automate outdated working habits. On the other hand, from the point of view of Information Science it is necessary that the developers of computer systems understand the context where the systems can be useful for the business. The traditional techniques of Requirements Engineering do not observe directly the organizational aspects. Nowadays there are specific techniques for organizational modeling that express the requirements related with the actor on the organization, their relations, intentions and existing reasons that determine the strategic alternatives for the business.

On the other hand, a precise description of the requirements is necessary to guarantee correctness and consistency of what was specified. It becomes necessary to state the requirements to make it possible to prove properties and make animations, for example. Nevertheless the traditional techniques have dealt with the functionality aspects of the systems, and have not addressed the organizational fundamentals.

We are proposing directives for the mapping of the organizational models, that explain the organizational requirements, into structural components of the precise specification languages. Our aim is to add the organizational components to the precise specifications. We are using the modeling technique *i** to find the organizational aspects, and the languages MAL (Structured Modal Action Logic) and OCL (Object Constraint Language) associated with UML (Unified Modeling Language), for the precise modeling. The proposed directives are applied to three cases for validation.

CAPÍTULO I

INTRODUÇÃO

Torna-se fundamental nos dias atuais, dada a complexidade cada vez maior dos sistemas de informação baseados em computadores, a necessidade de se modelar as organizações, não só internamente, mas objetivando apontar estratégias alternativas que ajudem ao entendimento do próprio negócio e a especificação de sistemas que ajudarão no negócio. Por outro lado, é fundamental procurar fazer com que o modelo construído seja o mais correto possível e, sobretudo, a depender das características do sistema desejado, seja o menos ambíguo possível. Com base nesses dois pontos de vista, procuraremos elucidar o contexto motivador desse trabalho, os objetivos a serem alcançados, o meio encontrado para atingir estes objetivos, bem como a estruturação do trabalho.

1.1- CONTEXTO

Atualmente, os sistemas de computação estão difundidos em todos os setores da vida moderna. Atividades antes executadas manualmente por pessoas, hoje são realizadas, em parte ou no todo, através de computadores, resultando na ampla melhoria da qualidade e rapidez dos serviços e produtos oferecidos pelas organizações a seus usuários. Desta forma, a tecnologia da informação evidencia-se como um componente importante em muitas iniciativas atuais e em novas tecnologias, que torna possível a meta de se melhorar, continuamente, as atividades desenvolvidas, identificando e selecionando, inclusive, alternativas que antes não tinham sido objeto de preocupação.

Toda essa corrida ao mundo dos softwares adveio do que se costumou chamar de a *Crise do Software*, também denominada por alguns pesquisadores, como a *Doença do Software* [PRE96] – ainda um problema atual –, surgida como resultado do acelerado desenvolvimento dos dispositivos de hardware e seu respectivo barateamento de custo. No afã de se acompanhar este desenvolvimento de hardware, passou-se a desenvolver “software” de forma desordenada. O custo do software cresceu de forma inversa ao do hardware, vertiginosamente ascendente, à medida que se dispunha de sistemas muito

mais complexos. O fato é que, de modo geral, as pessoas envolvidas na área de desenvolvimento de software, ainda não estavam, como ainda não estão, amadurecidas o suficiente, principalmente para tratar, sistematicamente, com os sistemas complexos e de grande porte que, em grande escala, entram na vida pessoal, governamental e comercial; as técnicas, quando aplicadas, não são apropriadas para esses tipos de sistemas. Daí advém um conjunto de problemas que até hoje se perpetuam, tais como: estimativas de custos e cronogramas mal dimensionados; produtividade inferior a demanda; e baixa qualidade dos serviços e produtos. Hoje nos deparamos com problemas oriundos de sistemas mal desenvolvidos, que poderão ocasionar desastres os mais variados, inclusive até com perdas de vidas, como o caso da pane no sistema de controle de ambulâncias de Londres [FIN93, LAS].

Como forma de equacionar os problemas oriundos da Crise do Software, surgiu a *Engenharia de Software*, tendo como preocupação o estabelecimento de princípios de engenharia, para obter softwares mais confiáveis, eficientes e passíveis de certificação. Constituindo-se, este ramo da Engenharia, de três elementos básicos: (a) teoria; (b) técnicas (procedimentos ou metodologias, conforme seja o termo desejado) para garantir a qualidade dos softwares; e (c) ferramentas para a automação desses métodos.

A aplicação da Engenharia de Software ressaltou a necessidade de se conhecer e se definir adequadamente os requisitos do sistema, quer de software ou não, que se deseja desenvolver. Passa-se a considerar que erros oriundos de uma má definição de requisitos, nos estágios iniciais do processo de desenvolvimento, podem resultar em altos custos na manutenção dos sistemas, em total rejeição do sistema e/ou em perdas econômicas, sociais ou ambientais. Sob esse ponto de vista é que a Engenharia de Requisitos surge como forma de contribuir para a melhoria dos problemas encontrados nos processos de desenvolvimento de sistemas [JIR94, CAS95], independente de qual metodologia esteja sendo utilizada na implementação do sistema. Para se realizar um bom trabalho de Engenharia de Requisitos faz-se necessário o entendimento dos aspectos sociais, organizacionais, técnicos, econômicos e jurídicos envolvidos no domínio do desenvolvimento, de forma a minimizar riscos e custos.

Nesse contexto, é de fundamental importância a utilização de metodologias, a fim de compreender os requisitos dos clientes e capturá-los de forma apropriada. Em geral, não é uma tarefa fácil a compreensão e captura dos requisitos, pois tratamos, muitas vezes, com desejos dos clientes que, em sua maioria, não sabem exatamente o que pretendem [BER94]. Para tanto, faz-se necessário uma compreensão mais ampla, procurando enfocar todos os atores (com suas especializações em agentes, posições e papéis) e objetos que se interrelacionam dentro da empresa onde o sistema a ser desenvolvido se localizará. É essencial a modelagem do negócio, procurando-se uma descrição dos “porquês”, das intenções que existem nas relações entre os agentes da organização.

Os modelos tradicionalmente utilizados, sob o ponto de vista da Ciência da Computação, são extremamente relacionados com aspectos da funcionalidade dos sistemas; em sua grande maioria, estão preocupados com o “que” e “como” fazer, apenas permitindo descrições de entidades e atividades, e não com o “porque” fazer, envolvidos nos processos decisórios existentes. Por outro lado, sob o ponto de vista da Administração, as técnicas visam sobretudo, modelar a estrutura interna da organização, evidenciando suas funções, dados e estratégias, tendo-se hoje como inovador a modelagem de processos.

Desta forma, é premente a necessidade de modelos capazes de reconhecer motivações, intenções e razões sobre as características dos processos existentes, a fim de que, ao final do processo de desenvolvimento, tenha-se uma documentação apropriada, que permita mais facilmente a compreensão das características essenciais do sistema, bem como as alternativas de processos que venham contribuir para a melhoria e eficácia dos sistemas, sobretudo, os de informação.

Neste contexto é que estamos direcionando nosso trabalho, apontando na Seção seguinte as razões que o impulsionaram.

1.2 – MOTIVAÇÃO DO TRABALHO

A forte tendência, nesse final de século e possivelmente, no início do próximo, está em se determinar formas de como melhorar e garantir qualidade no desenvolvimento de sistemas. Essa propensão está interrelacionada com a necessidade

de melhor compreensão do domínio da aplicação [JAC95a], nas primeiras fases do desenvolvimento, mais especificamente com a fase de definição dos requisitos, para que seja possível se agregar valor aos sistemas desenvolvidos; e com a busca pela formalização dos requisitos iniciais pretendidos pelos clientes, de forma a diminuir as ambigüidades e incertezas, objetivando-se a satisfação dos requisitos dos sistemas com as necessidades de seus usuários e com as especificações desenvolvidas. Temos consciência de que isso não eliminará todos os problemas, mas os reduzirá.

Na visão tradicional, uma especificação identifica, principalmente, aspectos relativos a funcionalidade (aos serviços almejados) do sistema requerido, preocupando-se com a descrição das funções fundamentais dos componentes de software que formarão o sistema. Nela, interessam as transformações que devam ser realizadas pelos componentes nas entradas, a fim de que algumas saídas sejam produzidas. Dificilmente, aspectos não funcionais, tais como confiabilidade, segurança, desempenho, portabilidade, disponibilidade, qualidade, etc., são tratados pelos métodos de modelagem (especificação) de requisitos tradicionais. Segundo ainda essa visão tradicional, aspectos relativos aos objetivos do sistema, possíveis relacionamentos entre estes e os objetivos e metas da própria empresa ou, ainda, aqueles aspectos não funcionais, não são considerados. A maioria das técnicas atuais focam suas atenções na definição das propriedades desejadas, em lugar de observar a informação de uma forma mais ampla, começando com as necessidades do próprio negócio, ou dos objetivos dos sistemas nele embutidos. As técnicas de modelagem utilizadas, tais como, análise estruturada, diagramas de fluxo de dados e modelagem entidade relacionamento, focam na modelagem das atividades e entidades, importantes conceitos para o desenvolvimento de sistemas, pouco oferecendo para ajudar na busca por soluções alternativas inovadoras aos problemas do negócio. Desta forma, é comum que os sistemas desenvolvidos não satisfaçam, realmente, às necessidades de seus usuários, muito embora seus desenvolvedores acreditem – ilusoriamente – que estejam tecnicamente certos. Segundo afirma Peter Drucker, em seu artigo [DRU99], “é bem possível que a informática tenha prejudicado seriamente a administração, porque ela é muito boa para conseguir informações adicionais do tipo errado. A informática gera mais dados sobre o lado interno da organização; praticamente não produz informações sobre qualquer coisa que

acontece fora da empresa”. Pensamento parecido é o de Cezar Taurion, que em seu artigo [TAU99] comenta o fato da necessidade da Tecnologia da Informação, no âmbito dos negócios, por ser extremamente estratégica. Todavia, afirma que “sem mudanças na maneira de pensar e usar a tecnologia dentro da empresa, para adicionar valor ao negócio, será difícil relacionar os investimentos com a Tecnologia da Informação e o aumento da rentabilidade e produtividade.”; e também, “os computadores por si não vão transformar uma empresa ineficiente em uma moderna e ágil organização”. Neste sentido, achamos de fundamental importância a cooperação entre o pessoal de negócio e a equipe de desenvolvedores de software, hardware e/ou sistemas. Deve ser de primordial importância a preocupação com os aspectos da organização, os internos - relativos aos dados do próprio negócio - e os externos - representados pelas intenções e estratégias do mercado - a serem considerados, quando do desenvolvimento de sistemas, objetivando melhorar o nível de aceitação desses sistemas. A insatisfação dos clientes com relação aos sistemas desenvolvidos ainda é um problema para a Engenharia de Requisitos, pois quando se consegue chegar ao final de um projeto, o especificado pode não refletir as reais necessidades do cliente. A consideração dos requisitos organizacionais, torna-se de fundamental ajuda para a melhoria da qualidade dos software desenvolvidos. Pois uma vez consideradas a estrutura organizacional, as metas da organização e seus objetivos estratégicos estaremos eliminando (ou minimizando) contradições que possam existir, nos sistemas em desenvolvimento, com relação a realidade da organização e aos seus objetivos de mercado.

Os projetistas, durante muito tempo, ignoraram a importância dos aspectos organizacionais. Só mais recentemente começaram a mudar sua visão [ZAV97, FOX97, YU96, YU97b, YU99], buscando ir mais além da simples descrição da funcionalidade do sistema. As técnicas mais modernas procuram preencher a visão tradicional com, pelo menos, dois outros aspectos relativos à compreensão: (a) do domínio, dentro do qual o sistema desejado se encontra; e, (b) das restrições que possam existir no sistema, no seu ambiente ou no seu desenvolvimento. Há uma tendência: a necessidade de aumentar o conhecimento acerca dos requisitos organizacionais [BUB94, KIR94, LEO98, LEI98, YU99] que influenciam as primeiras fases do desenvolvimento de que trata a Engenharia de Requisitos. Os sistemas desenvolvidos considerando-se esse tipo

de requisitos estarão conformes com a realidade e necessidades dos clientes, provavelmente com um índice de aceitação muito maior, refletindo a qualidade do desenvolvimento.

Pensando na necessidade de melhor compreender as relações entre o ambiente do negócio, as pessoas nele envolvidas e o sistema a ser desenvolvido, resolvemos nos agregar à linha de pesquisadores que objetivam tratar, especificamente, com os requisitos organizacionais [LEI95, LOU95, MYL95, YU95b, YU97b]. Nos estudos que realizamos, identificamos a existência de algumas técnicas específicas para tratar com a modelagem desse tipo de requisitos, sob a ótica da Ciência da Computação e sob a ótica da Administração. É inegável a necessidade de se aprofundar o conhecimento e se ampliar o contexto do problema, na tentativa de capturar requisitos organizacionais, que muito influenciarão nos sistemas requeridos, e que, tipicamente, dizem respeito às organizações (metas, objetivos, políticas, corpo de pessoal, etc.). Podemos citar, os fatores que nos motivam a estudar esse tipo modelagem:

- (i) A pobre compreensão acerca do domínio, identificada nas técnicas tradicionais, mas que se constitui em causa primária de falha dos projetos finais, em prazos extrapolados e em projetos não concluídos ou que não atendem a realidade de seus usuários;
- (ii) A necessidade de que os desenvolvedores entendam o que os usuários desejam, para que possam acrescentar informações adicionais, identificando, muitas vezes, as várias alternativas de desenvolvimento;
- (iii) A necessidade de que os usuários entendam o que os sistemas podem fazer para melhorar a qualidade de seus negócios. Neste ponto, a depender da técnica utilizada, faz-se necessário um maior esclarecimento ao cliente, sobre o modelo desenvolvido; e
- (iv) A necessidade de que o conhecimento da estrutura organizacional e o conhecimento estratégico estejam bem organizados, a fim de que o compartilhamento e o reuso sejam facilitados.

Por outro lado, para alguns tipos de sistemas (por exemplo, sistemas críticos e de segurança, dentre outros) também é premente a necessidade de se ter uma especificação formal dos requisitos, logo nas primeiras fases do processo de desenvolvimento, para

que se possa garantir que a definição de requisitos estabelecida, seja realmente o elicitado pelo analista e reflita o desejado pelo cliente. Para isso, são utilizadas linguagens dotadas de rigor matemático, de forma a possibilitar o uso de provas e testes de completude e consistência. Deve-se ressaltar que existem limitações em se retratar certas propriedades do sistema [MAI87, BOW94a], tendo em vista as características do sistema matemático escolhido, além do fato de serem estes métodos formais, em sua maioria, “não atraentes” a alguns usuários e desenvolvedores de sistemas [WARM99]. Essa dificuldade natural ocorre pela necessidade de uma base matemática mais aprofundada, para que se possa empregar e usufruir de toda a técnica formal. As linguagens de especificação formal, em geral são muito complexas e destinadas a pessoas que, realmente, dispõem de um conhecimento matemático mais profundo, não sendo este o caso, em geral, dos engenheiros de software. Felizmente, hoje já existem linguagens de especificação formal [WARM99] cuja proposta é a de serem mais amigáveis, principalmente para o pessoal de desenvolvimento, no sentido de não exigirem conhecimento matemático muito aprofundado, bem como alguns ambientes que suportam desenvolvimento formal [CAS96, REI97].

Pela reconhecida importância que tem a formalização dos requisitos, alguns pesquisadores de renome, na área da Engenharia de Requisitos, falam em integração do informal com o formal [BUB94b, LAM95, WIE98, DUB98, CAS97]. Por outro lado, pouco encontramos, durante nossas investigações, trabalhos da comunidade de Métodos Formais, que expressem a *necessidade de integração do formal com o informal*. Quando muito, verificamos alguma preocupação [MAI87] com o ambiente no qual o sistema a ser especificado será inserido, mas não com o contexto geral, principalmente com respeito aos aspectos organizacionais. Desta forma, acreditamos ser oportuna uma proposta para ajudar o processo de formalização de requisitos, cujos benefícios incluem:

- (i) Uma forma de se garantir a corretude e consistência daquilo o que foi capturado e especificado;
- (ii) A possibilidade de provas de propriedades importantes, capturadas pela modelagem organizacional; e
- (iii) A possibilidade de animação dos requisitos, como forma de validação do documento de requisitos obtido.

Assim sendo, propomos estreitar essa distância, oferecendo meios para que o campo das especificações formais que, em geral, lida com os aspectos funcionais dos sistemas, possa ter a sua disposição, mais informações relativas ao campo das organizações, no tocante aos requisitos organizacionais que em muito podem complementar as especificações, apontando estratégias, alternativas e objetivos a serem seguidos. Como forma de implementar esse estreitamento entre os dois campos, serão sugeridos alguns mecanismos de mapeamento, que nos conduzam do campo organizacional (mais alto nível) para o campo formal (mais preciso).

1.3 - OBJETIVOS E ABORDAGEM

Tendo levantado, na Seção anterior, a necessidade de se ter, logo nas primeiras fases do desenvolvimento, a modelagem de aspectos relativos ao negócio e, por outro lado - para se ter uma especificação mais precisa e correta, desses requisitos - a formalização dos requisitos, e identificado a distância entre as duas formas de abordagem, objetivamos fazer um mapeamento, inicialmente unidirecional (vide Seção 7.4), entre esse dois mundos.

Como benefício deste mapeamento destacamos a possibilidade de incorporarmos, às especificações existentes, aspectos relativos à organização que determinarão efetivamente a adequação do sistema a sua real necessidade de implementação e implantação. Além disso, sob o ponto de vista da modelagem da organização o mapeamento, possibilitará a animação das especificações de requisitos segundo as ferramentas disponíveis nas linguagens precisas adotadas neste trabalho.

Neste trabalho são propostas, assim, algumas diretrizes de mapeamento, que possibilitem gerar uma parte de especificação formal, facilitando o tratamento dos requisitos organizacionais, relacionados com campo das empresas, pelo mundo da modelagem formal.

Tendo em foco o objetivo de modelar e tratar requisitos, faz-se necessário, primeiramente, conhecer as técnicas de modelagem convencionais existentes, verificando de que tipos de requisitos tratam.

Como segundo passo no encaminhamento de nosso objetivo, tendo identificado o fato de que as técnicas convencionais existentes, em geral, não tratam os requisitos organizacionais adequadamente, faz-se necessário investigar quais técnicas de modelagem disponíveis na literatura tratam, especificamente, desses requisitos.

Isto posto, como terceiro passo, selecionamos uma técnica específica para a modelagem dos requisitos organizacionais, no caso a técnica i^* , proposta por Eric Yu [YU95a]. A escolha fundamentou-se nas razões:

- (i) ser uma técnica moderna, que trata das razões e dependências estratégicas envolvidas no desenvolvimento, cuja importância é também considerada por outros pesquisadores [MAL94, LEO98];
- (ii) ser uma técnica orientada a agentes [YU97c, YU98], a quem se atribui a intencionalidade das ações;
- (iii) por ter aspectos de orientação a objeto [MYL99], tendência atual muito forte; (iv) pelo fato de caracterizarmos i^* como uma técnica graficamente concisa [ALE99b] - apresentando apenas dois modelos - e oferecer notações diferenciadas para cada aspecto a ser tratado (agente, objetivo, recursos, tarefas, objetivos não funcionais, etc.); e
- (iv) pela aplicabilidade desta técnica, [YU95b, DUB98], conforme descrito na Seção 7.2, adiante.

Alguns sistemas, tais como, sistemas críticos, sistemas em tempo-real, sistemas de segurança e sistemas distribuídos, precisam ser formalmente especificados. Assim como quarto passo da abordagem, identificamos as técnicas de especificação formal existentes, optando por aquelas que melhor se adequem aos nossos objetivos.

Como quinto passo, precisamos, dentre as técnicas formais existentes, optar por aquelas linguagens apropriadas ao nosso objetivo. Neste sentido, elegemos duas linguagens: Structured MAL (Structured Modal Action Logic) e OCL (Object Constraint Language).

A primeira é uma linguagem de especificação formal baseada em lógica modal de ações, conhecida como MAL. Essa escolha, fundamentalmente, deveu-se:

- (i) ao fato de ser MAL uma linguagem que foi usada no campo industrial, haja visto ter sido parte do projeto FOREST [MAI87], do qual resultou uma nova lógica estruturada em termos de orientação a objeto [RYAN90], assim como num novo método de elicitação de requisitos, o VSCS (Very Structured Common Sense) [CAS91, CAS92], levando em consideração os novos mecanismos de estruturação;
- (ii) ao fato de ter sido desenvolvido o ambiente Multiview [CAS96, CAS97] no Departamento de Informática da Universidade Federal de Pernambuco, o qual permite a geração de especificações em MAL, tendo por base o método VSCS; e
- (iii) a possibilidade de descrever a ordem de ocorrência das ações, fato esse que pode ser útil, no momento da ordenação das tarefas, dentro do modelo organizacional selecionado.

A segunda linguagem de especificação formal escolhida, OCL, baseia-se em modelos, conhecida como a Linguagem de Restrições a Objetos, OCL (Object Constraint Language) [WARM99], utilizada para complementar os modelos orientados a objeto. Essa escolha fundamenta-se no fato da linguagem:

- (i) principalmente, fazer parte do padrão de modelagem orientado a objeto, UML (Unified Modeling Language) [JACO99], em uso por toda a comunidade de orientação a objetos; e
- (ii) ter como proposta a garantia de ser mais voltada aos engenheiros que lidam com desenvolvimento de sistemas e que, por isso, são profissionais que muitas vezes não têm uma fundamentação matemática tão rigorosa.

Com base na técnica de modelagem organizacional i^* , e em duas linguagens formais, Structured MAL e OCL, propomos, finalmente, o estabelecimento de um mapeamento que possibilite passar os aspectos capturados pela modelagem organizacional, para o mundo das especificações formais, que utilizem as linguagens

Structured MAL e OCL. São produzidas diretrizes que ajudam, de forma mais prática, o mapeamento dos conceitos levantados, de forma unidirecional, com a técnica de modelagem organizacional i^* e os construtores das linguagens de formalização adotadas. Com isso, buscamos facilitar a possibilidade de formalização dos requisitos organizacionais que, até então, não são considerados de forma apropriada na literatura especializada.

1.4 - ESTRUTURA DO TRABALHO

Nesta Seção, resumidamente, apresentamos como se encontra estruturado este trabalho de tese, de forma que se tenha, inicialmente, uma compreensão teórica do que o trabalho aborda.

No Capítulo 2, é feita uma revisão sobre a origem da Engenharia de Requisitos, principais conceitos e definições, principais atividades e técnicas correlatas, a classificação dos requisitos em funcionais, não funcionais e organizacionais, sendo evidenciada a necessidade de modelá-los sistematicamente; são apresentadas as principais técnicas de modelagem funcional e não funcional, merecendo algum destaque a técnica UML; culminando com os principais problemas na área de Requisitos, sobretudo aqueles que ainda são aspectos de pesquisa em aberto.

Na última década, tem sido grande o número de abordagens que procuram estabelecer técnicas, métodos, linguagens para a descrição e produção de requisitos, de forma a ter uma sintaxe mais precisa: em geral, gráficos acompanhados por informações textuais. Destacamos assim, no Capítulo 3, as principais técnicas de modelagem de requisitos organizacionais, que tratam de metas e objetivos dos ambientes organizacionais, nos quais se encontram inseridos os sistemas desejados, logo nas primeiras fases do desenvolvimento. Dentre as técnicas existentes, destacamos a técnica de modelagem organizacional i^* , objeto de nosso estudo, para a qual apontamos aspectos formais considerados pelos autores.

Na Engenharia de Requisitos, o uso das linguagens de especificação formal permite que as especificações de requisitos tenham mais rigor, podendo esses requisitos serem checados segundo o conjunto de propriedades desejadas. Essas especificações formais podem ainda ser utilizadas para a geração de um protótipo de implementação executável. Neste sentido, no Capítulo 4, são apresentados os principais conceitos de especificação formal, destacando-se as principais técnicas e as características das linguagens Structured MAL e OCL, eleitas, especificamente, para o propósito deste trabalho.

O que se deseja é levar em consideração, no momento da especificação dos requisitos, aspectos relativos à intenções, objetivos e metas, normalmente relacionados com a estrutura organizacional no qual o sistema será embutido, e que, em geral, ficam esquecidos pelos modeladores. Por outro lado, espera-se que essa mesma especificação tenha o rigor necessário. Assim sendo, no Capítulo 5 procuramos estabelecer diretrizes para que se tenha um mapeamento unidirecional dos requisitos organizacionais capturados e expressos através da técnica i^* , com os formalismos oferecidos pelas linguagens de especificação formal Structured MAL e OCL.

No Capítulo 6 são apresentados 3 (três) estudos de caso, como meio de validarmos a aplicação das diretrizes apontadas no Capítulo anterior, para o estabelecimento do mapeamento entre o mundo da modelagem organizacional e o mundo das especificações formais. Apresentamos dois casos relativos a integração da modelagem em i^* com a linguagem MAL e um terceiro caso relativo a integração da modelagem em i^* com a linguagem OCL. Esses três casos utilizados, fazem parte de trabalhos apresentados e publicados em eventos nacionais e internacionais [ALE97a, ALE97b, ALE98a, ALE98b, ALE99a, ALE99b].

Por fim, no Capítulo 7 são apresentadas as contribuições, os trabalhos relacionados e futuros, oriundos de possíveis extensões deste trabalho.

CAPÍTULO II

A ENGENHARIA DE REQUISITOS

O passo primário e fundamental para o sucesso, em qualquer modelo de desenvolvimento de software, é o da *Definição e Análise dos Requisitos*, pois mesmo que se tenha um sistema bem projetado e codificado, se ele foi mal especificado, com certeza causará prejuízos e transtornos para o cliente e para os desenvolvedores. Assim, é prudente se utilizar das metodologias oferecidas pela Engenharia de Requisitos, para um adequado processo de desenvolvimento de definição e análise de requisitos.

Neste Capítulo, apresentamos os conceitos e as atividades da Engenharia de Requisitos, a classificação para os requisitos e suas principais técnicas de modelagem, além de apontar possíveis problemas relacionados que ainda se encontram em aberto.

2.1 – PRINCIPAIS CONCEITOS DA ENGENHARIA DE REQUISITOS

No processo de desenvolvimento de software, estabelecer requisitos que sejam compreensíveis por todas as partes envolvidas – clientes, modelador, desenvolvedor, etc. – é um fator básico, ao tempo que é, também, um problema de difícil solução. Assim, faz-se necessária uma abordagem sistemática de obtenção dos requisitos que assegure a compreensão dos requisitos do usuário e a produção de um sistema utilizável a um custo bom. Essa abordagem deve seguir princípios de engenharia, apresentando de forma adequada métodos, técnicas e ferramentas que dêem suporte a essa etapa do processo de desenvolvimento. Nesse sentido é que foi criada a Engenharia de Requisitos, termo para o qual encontramos algumas definições.

2.1.1 - DEFINIÇÕES

Inicialmente, é fundamental definirmos o que sejam *Requisitos*.

A idéia do que seja requisito varia muito. Por requisitos pode-se entender um conjunto de descrições de como o sistema pretendido deva se comportar ou um conjunto de propriedades, de atributos do sistema e/ou limitações do próprio processo de

desenvolvimento do sistema. Neste contexto, existem várias idéias sob diferentes pontos de vista.

Segundo Sommerville [SOM97, KOT98], um requisito pode descrever uma:

- (a) Facilidade a nível do usuário; por exemplo, um corretor de gramática e ortografia;
- (b) Propriedade muito geral do sistema; por exemplo, o sigilo de informações;
- (c) Restrição específica no sistema; por exemplo, o tempo de varredura de um sensor;
- (d) Restrição no desenvolvimento do sistema; por exemplo: a linguagem JAVA deverá ser utilizada para o desenvolvimento do sistema.

Nessa linha é que observamos a definição de requisitos, segundo o padrão 830 (1998) da IEEE [STD830-98], que descreve requisito como:

- 1) uma condição ou uma capacidade de que o usuário necessite, para que solucione um problema ou alcance um objetivo;
- 2) uma condição ou uma capacidade que deve ser alcançada ou possuída por um sistema ou componente de um sistema, para satisfazer um contrato, um padrão, uma especificação ou outros documentos impostos formalmente;
- 3) uma representação documentada de uma condição ou capacidade, conforme os itens (1) e (2).

A visão de requisitos, segundo Davis [DAV93], não se restringe a apenas o "quê" o sistema deva fazer, mas também a "como" deva fazer. De outra forma, trata-se do que Davis caracteriza como o "dilema do quê versus como". Para ele a definição do IEEE não é muito boa, pois só trata da visão do quê.

Uma definição bem simples para requisitos é dada por Macaulay [MAC96], para quem requisito é simplesmente algo de que o cliente necessita.

Segundo Jackson [JAC95a, JAC95b, ZAV97] requisitos são fenômenos ou propriedades do domínio da aplicação que devem ser executadas. Normalmente esses fenômenos ou propriedades são expressos em linguagem natural, diagrama informal ou outra notação apropriada ao entendimento.

Já a definição de requisitos tomada por Loucopoulos [LOU95, BUB95a], acrescentam à definição do IEEE, as informações relativas ao contexto organizacional do cliente onde o sistema deverá operar.

Uma segunda definição necessária é a da disciplina de Engenharia de Requisitos.

Segundo o IEEE [STD830], Engenharia de Requisitos corresponde ao processo de aquisição, refinamento e verificação das necessidades do cliente para um sistema de software, objetivando-se ter uma especificação completa e correta dos requisitos de software.

No guia STARTS, segundo Hofmann [HOF93], Engenharia de Requisitos compreende o processo pelo qual as intenções e requisitos escritos e falados de seu possuidor são transformadas em uma especificação precisa, consistente, não ambígua e completa do comportamento do sistema, incluindo funções, interfaces, desempenho e limitações.

Para Leite[LEI87], a Engenharia de Requisitos pode ser definida como um processo segundo diferentes pontos de vista, no qual "o que" é para fazer, é capturado e modelado. Neste processo é usada uma combinação de métodos, ferramentas e atores, sendo produzido, como resultado da modelagem, um documento de requisitos.

Meyer [MEY88] afirma que especificar o documento de requisitos de um software é definir, de forma completa e não ambígua, as características externas do software oferecidas aos usuários e a forma pela qual este será integrado ao sistema.

Boehm [BOE89] define Engenharia de Requisitos como uma atividade que objetiva desenvolver uma especificação completa, consistente, não ambígua e correta dos requisitos, que sirva, inclusive, de base para um acordo entre as partes envolvidas no processo de desenvolvimento do software, onde se pactue, de forma concisa, "o que" o produto irá fazer. Nesse sentido, a Engenharia de Requisitos seria um processo que requer um envolvimento muito grande entre cliente e desenvolvedores, evitando decisões de projetos durante a definição dos requisitos, ficando, assim, definidas, pelo menos, duas fases bem distintas: o que se tenta alcançar e como projetar o sistema para satisfazer os requisitos. Segundo Hofmann, essa definição parece separar a Engenharia de Requisitos de outras preocupações do desenvolvimento de software [HOF93].

Já segundo Davis [DAV93], a Engenharia de Requisitos corresponde a atividade de entendimento das necessidades do usuário no contexto do problema a ser resolvido, bem como das limitações impostas na solução.

De acordo com Loucopoulos [LOU95], a Engenharia de Requisitos corresponde ao processo sistemático de desenvolvimento de requisitos, através de um processo iterativo, cooperativo de análise do problema, documentação das observações resultantes, em uma variedade de formatos de representações e checagem da acurácia da compreensão ganha.

Segundo Sommerville [KOT98], trata-se de um termo relativamente novo, para cobrir todas as atividades envolvidas na descoberta, documentação e manutenção de um conjunto de requisitos, em um sistema baseado em computador; onde o termo Engenharia implica que técnicas sistemáticas e repetitivas são usadas para assegurar que os requisitos do sistema sejam completos, consistentes e relevantes.

O que observamos é que algumas definições procuram retratar a Engenharia de Requisitos como um processo sistemático de investigação das necessidades do cliente com relação ao sistema a ser desenvolvido, portanto, baseando-se fundamentalmente na percepção do problema para o qual se desenvolverá um sistema (software e/ou hardware). Todavia, recaem na questão da compreensão do problema por parte dos desenvolvedores.

2.1.2 – MOTIVAÇÕES PARA O ESTUDO DA ENGENHARIA DE REQUISITOS

Várias são as razões para se estudar e aplicar a Engenharia de Requisitos, como adiante iremos explicitar, mas o fato principal é a correlação existente entre os erros e inadequações do produto final, com as falhas ocorridas durante as fases de análise e especificação dos requisitos. Segundo relatórios da NASA Jet Propulsion Laboratory [LUT93, KEL92], essa situação cada vez mais tende a se agravar com as novas tecnologias.

Alguns estudos mostram que um grande número de sistemas de informação não são apropriados para as necessidades de seus usuários. Segundo [CAS95], o nível de aceitação dos sistemas de informação comerciais é da ordem de 40%, enquanto que, para sistemas de tempo real, este índice sobe para cerca de 75%. Este fato, em sua maioria, está correlacionando a erros e inadequações na produção do produto final, com falhas ocorridas

durante as fases de análise e especificação de requisitos. Ou seja, nos sistemas de tempo devido a sua própria natureza, a análise e a especificações de requisitos devem ser obrigatoriamente serem feitas com mais cuidado e precisão, demandando mais tempo e atenção. Daí o alto índice de aceitação dos sistemas de tempo real.

As motivações variam desde razões técnicas a, principalmente, aspectos econômicos e, mesmo, consequências legais e sócio-ambientais. A Figura 2.1 identifica, sinteticamente, os aspectos motivadores ao estudo da Engenharia de Requisitos.

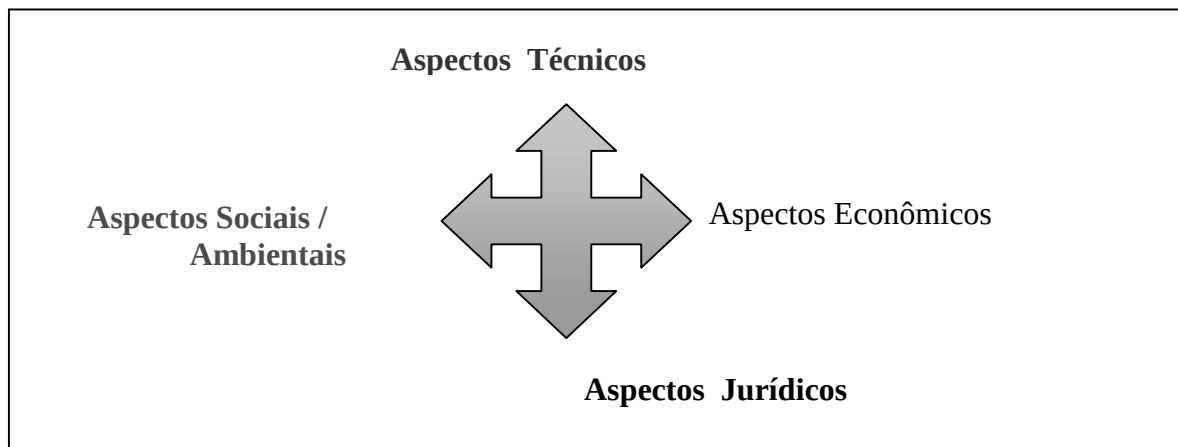


Figura 2.1 – Aspectos Motivadores ao Estudo da Engenharia de Requisitos

Segundo o *aspecto sócio/ambiental*, uma má especificação dos requisitos de um sistema ou produto pode ocasionar danos irreparáveis, como ilustra o caso do Royal Ambulance Service [FIN93], que, por uma falha no sistema, deixou em 1992, todo o serviço de ambulância na cidade de Londres virtualmente paralisado. A falta de definição e precisão nos requisitos pode acarretar riscos de vida, como falhas possíveis de ocorrer em sistemas de tempo-real, tais como a tele-medicina e o sistema aeroespacial, dentre outros; do mesmo modo, desastres ambientais podem ser ocasionados por uma má especificação de equipamentos de segurança ou sistemas de controle.

As falhas que possam existir em função da má especificação dos requisitos, nas fases iniciais do desenvolvimento, dizem respeito ao *ponto de vista técnico*. Observa-se

que cerca de 40% do percentual de erros detectados nos sistemas, deve-se a especificações mal feitas, Figura 2.2 [CAS95].

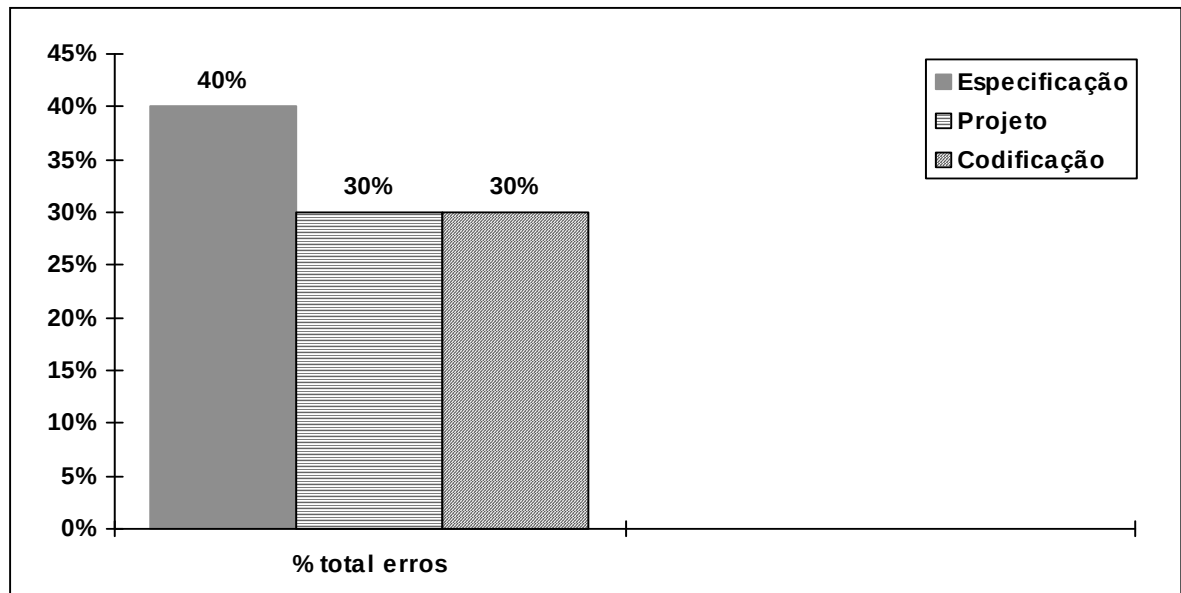


Figura 2.2 - Percentuais de Erros Referentes às Diferentes Fases do Desenvolvimento de Software

Por tratar de aspectos relacionados com a especificação dos requisitos de um sistema, a Engenharia de Requisitos utiliza técnicas, ferramentas e metodologias de desenvolvimento, a fim de serem mais corretamente tratados os objetivos e desejos quanto a funcionalidade do objeto proposto.

É preciso enfatizar também, que o cliente, normalmente, não sabe ou não tem certeza do que realmente quer [BER94]: este ponto é reconhecido pela Engenharia de Requisitos, no momento de capturar os requisitos. Além das próprias incertezas quanto ao desejado pela parte contratante e/ou desenvolvedora, os requisitos relativos a própria estrutura da organização, as metas e objetivos da empresa também devem ser considerados e modelados, a fim de que se tenha um bom planejamento, gerenciamento e desenvolvimento. No Capítulo 3, estes pontos serão tratados mais detalhadamente.

Sob o *ponto de vista econômico* existem fortes razões para se considerar a Engenharia de Requisitos, visto que altos custos durante a fase de manutenção são relativos a erros ocorridos durante a fase de definição e análise dos requisitos. Além dos dados

levantados - conforme Figura 2.2 - sobre o percentual de erros detectados nos sistemas, observa-se também que aproximadamente 66% do total do custo de correção de erros se encontra na fase de especificação dos requisitos, Figura 2.3.

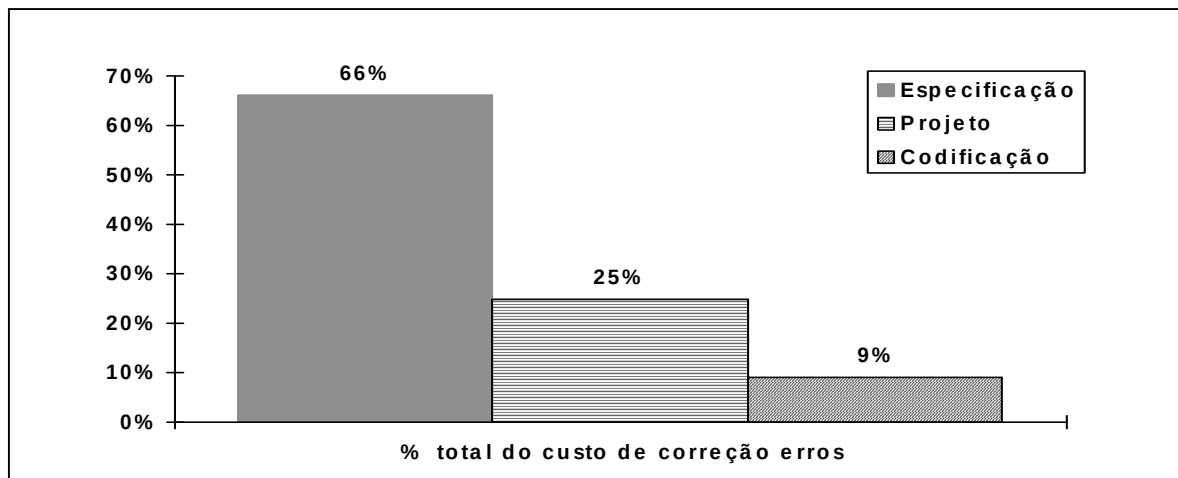


Figura 2.3 - Percentuais de Custo com a Correção de Erros Relativos às Diferentes Fases do Desenvolvimento de Software

Ainda nessa linha, quanto mais tarde for detectada uma decisão errada, mais caro será corrigi-la. Para se ter uma idéia, o custo de correção de um erro, introduzido durante a fase de especificação do Processo da Engenharia de Software, é na média *cinco vezes* maior que o custo de um único erro ocorrido durante a fase de codificação, Figura 2.4.

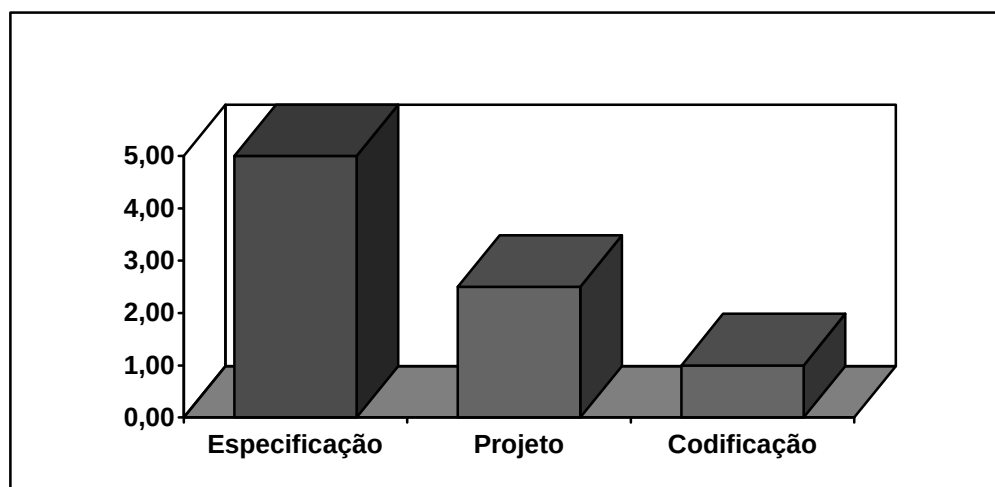


Figura 2.4 - Custo de Correção de Um Erro Durante a Especificação dos Requisitos

Segundo o *ponto de vista jurídico*, o documento de requisitos pode funcionar como um acordo contratual entre os clientes e a equipe de desenvolvedores. Assim, a utilização de técnicas apropriadas à obtenção da especificação dos requisitos do sistema contratado deve ser usado como suporte para correta elaboração do pacto. Outro ponto está nos aspectos legais quanto às responsabilidades pelos efeitos decorrentes dos possíveis danos ambientais, sociais e econômicos do sistema, cujos requisitos não tenham sido precisamente especificados e entendidos. Trata-se, de certa forma, dos requisitos regulamentares.

A Engenharia de Requisitos serve de suporte às partes contratantes - clientes e desenvolvedores -, para estabelecer criteriosamente os requisitos do sistema de software que se queira desenvolver e revisá-los no contexto do problema, obtendo-se uma especificação completa.

Caso o processo de requisitos seja inadequado, os usuários perderão a confiança na equipe de desenvolvedores e na própria disciplina de Engenharia de Requisitos. Consequentemente, trata-se de um processo difícil de ser bem realizado, mas de fundamental necessidade para o bom funcionamento do sistema desejado.

2.1.3 - ATIVIDADES DO PROCESSO DA ENGENHARIA DE REQUISITOS

De uma maneira geral, um processo corresponde a um conjunto organizado de atividades que transformam entradas em saídas. Segundo Sommerville [KOT98, SOM97], a Figura 2.5 caracteriza bem o processo de Engenharia de Requisitos. Nesta Figura, temos algumas entradas que sempre devem ser levadas em consideração:

- (a) Informações sobre os sistemas existentes;
- (b) Necessidades das partes envolvidas com o sistema a ser desenvolvido;
- (c) Padrões Organizacionais, que dizem respeito a própria estrutura da organização;
- (d) Regulamentações que possam vir a influenciar os requisitos do sistema; e
- (e) Informações sobre o Domínio da aplicação.

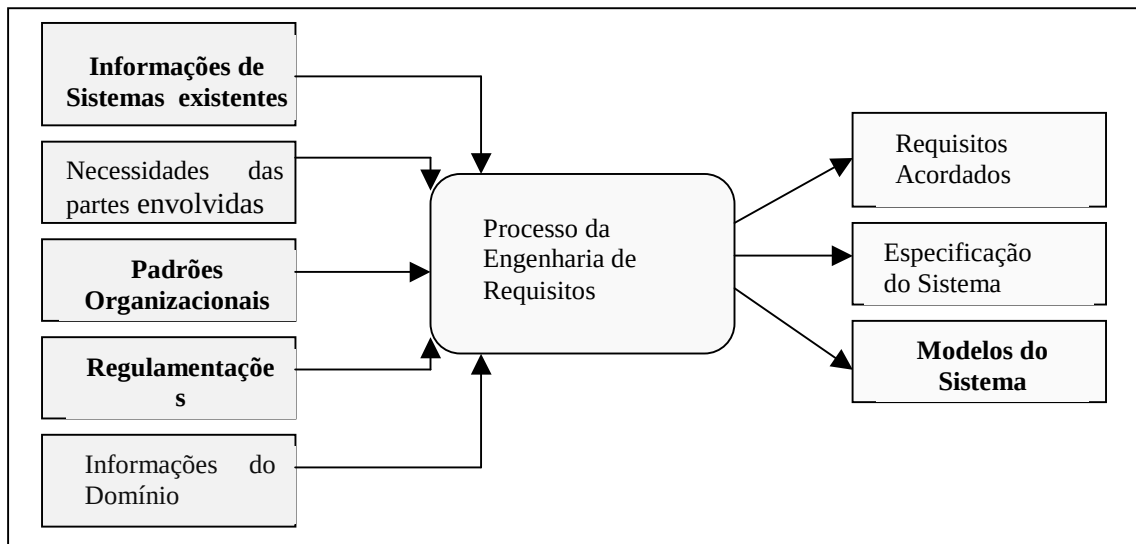


Figura 2.5 - Entradas e Saídas do Processo da Engenharia de Requisitos

Como podemos ver na Figura 2.5, Sommerville considera como relevante, ao profissional de computação, não só buscar informações sobre a funcionalidade dos sistemas ou descrever as necessidades dos usuários do sistema a ser desenvolvido, mas entender melhor o domínio da aplicação, considerar os aspectos relativos a própria estrutura da organização (padrões adotados, gerenciamento de qualidade, metas, estratégias, etc.) e inclusive considerar os aspectos legais (regulamentações externas) que se apliquem ao sistema a ser produzido. Destacando-se, assim, que não só cabe ao administrador conhecer ou modelar os requisitos organizacionais, mas também ao profissional que estará desenvolvendo o sistema a ser adotado pela instituição.

Sob nosso ponto de vista, ao considerarmos os requisitos organizacionais estaremos adequando-nos à realidade do próprio negócio. Requisitos que estavam sendo modelados passam a serem adequados às metas e interesses da organização, não se desenvolvendo sistemas que em parte ou no todo contrariem esses objetivos. Como resultado, por exemplo, teremos pelo menos:

- (i) custos de projetos bem ajustados, satisfazendo às partes (clientes e engenheiros de requisitos);
- (ii) prazos de execução dimensionados segundo a necessidade; e

- (iii) satisfação do cliente, visto que o especificado, realmente, molde-se (ou identifica, a depender da natureza da proposta de trabalho) a um problema ou deficiência real de sua organização.

Pelo observado na Figura 2.5, como resultado do processo da Engenharia de Requisitos espera-se ter, pelo menos três saídas importantes:

- (a) Requisitos Acordados;
- (b) Especificação do Sistema; e
- (c) Modelos do Sistema.

Essas saídas serão fundamentais para as demais etapas do desenvolvimento, dado que serão tomadas como base.

Todavia, não existe um processo ideal para a Engenharia de Requisitos [KOT98]. Várias são as formas de se caracterizar o processo da Engenharia de Requisitos, variando segundo fases e os autores; daí encontrar-se muitas inconsistências na terminologia utilizada na área [DAV93].

Segundo Kotonya [KOT98], os processos de requisitos variam radicalmente de uma organização para outra, existindo vários fatores que concorrem para isso, dentre os quais:

- (i) maturidade técnica;
- (ii) envolvimento disciplinar;
- (iii) cultura organizacional; e
- (iv) domínio da aplicação.

Já Loucopoulos [LOU95] afirma que existe uma aparente inabilidade ou mesmo má vontade em se fornecer uma descrição mais precisa da Engenharia de Requisitos, o que explica os diferentes modelos de requisitos existentes.

No modelo de atividades de alto nível de Sommerville [KOT98], tem-se fundamentalmente, quatro fases: Elicitação, Análise, Modelagem e Validação. Segundo ele, as atividades de Elicitação e Validação dos requisitos corresponderiam à fase de Aquisição de Requisitos, enquanto a Análise e a Modelagem dos requisitos corresponderiam à fase de Especificação de Requisitos. Já Loucopoulos [LOU95] formula o processo em termos de três principais atividades: Elicitação, Especificação e Validação, correspondendo, segundo este, às três preocupações importantes na Engenharia de Requisitos: Compreensão,

Descrição e Concordância de um problema. Concluímos então que, na verdade, o termo Especificação é empregado para sintetizar as atividades de Modelagem e Análise apontadas por Sommerville, conforme podemos ver resumidamente na Figura 2.6 [CAS95].

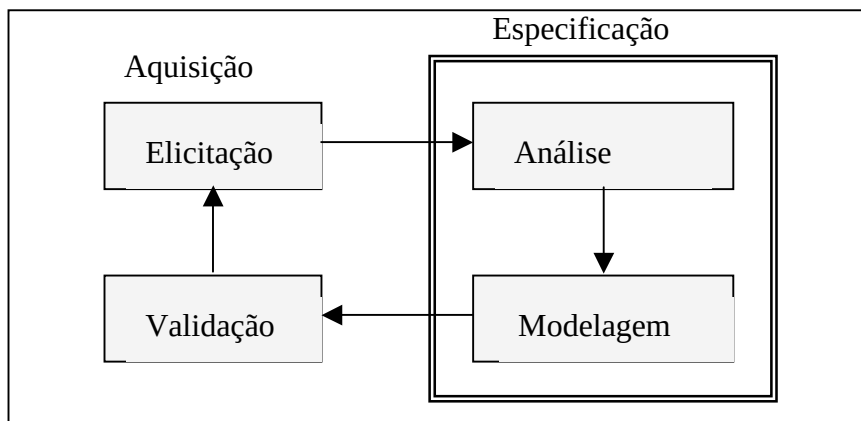


Figura 2.6 - Atividades do Processo da Engenharia de Requisitos

Fato é que sempre encontraremos o processo da Engenharia de Requisitos sendo aplicado nos modelos de desenvolvimento [PIN98], seja ele qual for.

A título de definição, utilizaremos a classificação adotada por Sommerville [KOT98], acrescentando, a cada uma das atividades possíveis, comentários associados, colhidos de outros autores, ou seja, abordaremos na ordem as fases de: Elicitação, Análise, Modelagem e Validação.

Na atividade de *elicitação de requisitos*, procura-se identificar os fatos que compõem os requisitos do sistema, objetivando-se obter conhecimento, o mais correto e mais completo possível, sobre o domínio do problema, de forma a prover entendimento do que será demandado e a possibilitar que esse conhecimento possa ser usado para produzir uma especificação mais precisa.

Para realizar a elicitação dos requisitos, entre as técnicas existentes podemos enumerar as seguintes:

- (a) Entrevista [LEI96];
- (b) Leitura de Documentos;
- (c) Questionários;
- (d) Análise de Protocolos;

- (e) Participação Ativa dos Usuários;
- (f) Cenários [MIL98, BRE98, WEI98, KAI98];
- (g) Métodos Soft Systems (SSM) [CHE89];
- (h) Observações e Análise Sociais [JIR98];
- (i) Reuso de Requisitos;
- (j) Prototipagem; e
- (k) Identificação de Intenções [LEI98, BRE98, WEI98, LEV98].

O profissional de Engenharia de Requisitos deve selecionar as técnicas a serem utilizadas e estabelecer de que maneira elas serão integradas, fato que dependerá do problema e da equipe participante. Como resultado desta fase temos uma declaração informal dos requisitos.

A *análise de requisitos* é um passo altamente crítico no processo de desenvolvimento, pois nesta ocasião são identificadas inadequações, contradições, ambigüidades e incompletudes que possam existir segundo a lista dos requisitos elicitados. O objetivo principal é a obtenção de uma especificação que seja o mais consistente e completa possível. Na verdade, a análise deve ser intercalada com a elicitação, pois, em geral, problemas são descobertos quando os requisitos estão sendo elicitados.

Para realizar a análise dos requisitos encontramos várias técnicas, das quais enumeraremos as que seguem, por serem as mais antigas e tradicionais referenciadas:

- (a) Diagramas de Fluxo de Dados – DFD [DeMA79];
- (b) Técnica de Análise e Projeto Estruturado – SADT [ROS77];
- (c) Metodologia para Engenharia de Requisitos de Software – SREM [BEL77];
- (d) Método de Ward-Mellor [WAR85, WAR86];
- (e) Análise Orientada a Objeto – OOA [COA91];e
- (f) Método de Shlaer-Mellor [SHL92].

Associado a análise tem-se a *modelagem dos requisitos*. De forma geral, um modelo é, normalmente, fruto de uma linguagem de representação de requisitos ou combinação de linguagens de representação de requisitos (linguagens naturais, rigorosas ou formais).

O que se objetiva é obter, como resultado desta fase do processo, um documento formal contendo uma definição oficial do que seja necessário aos desenvolvedores do sistema, o “*Documento de Requisitos*”. Devendo-se ressaltar que os requisitos sempre evoluirão a medida que se tenha um melhor entendimento das necessidades do usuário e mudanças nos objetivos da empresa, sendo essencial planejar, para que as possíveis mudanças de requisitos, durante o desenvolvimento e uso do sistema, não levem à necessidade de re-escritas extensas no Documento de Requisitos.

Diversas são as técnicas para a modelagem de requisitos existentes, que variam conforme a classificação dada aos requisitos (Seção 2.2), duas das quais – técnicas para a modelagem de requisitos funcionais e técnicas para a modelagem de requisitos não funcionais -, serão tratadas na Seção 2.3, enquanto a modelagem de requisitos organizacionais será abordada no Capítulo 3.

A *validação* dos requisitos corresponde ao processo de certificação do modelo de requisitos quanto a correteza em relação à intenção do usuário. Caso a validação não seja adequada, poderão ser propagados para as fases de projeto e implementação, erros e mal entendidos cometidos nas fases anteriores.

Sendo sugerido ainda por Sommerville [KOT98] e por Loucopoulos [LOU95], podemos citar algumas técnicas de validação:

- (a) Prototipagem;
- (b) Simulação e Animação;
- (c) Para-fraseamento da Linguagem Natural; e
- (d) Abordagens de Sistemas Inteligentes.

Dentre as técnicas apontadas, a Prototipagem apresenta-se como uma das mais efetivas para validação de requisitos. Quando dispomos de uma especificação de requisitos executável, a validação fica facilitada, pois aquela já pode ser diretamente utilizada como teste pelos próprios usuários. Com isto pode-se ter redução dos custos e dos prazos de desenvolvimento dos sistemas. Daí também a importância do desenvolvimento de protótipos nesta fase.

2.2 - CLASSIFICAÇÃO DOS REQUISITOS

Os ambientes empresariais atuais são caracterizados por sistemas interligados de características as mais diversas. Assim, os softwares e/ou os sistemas, para essas organizações, precisam ser construídos considerando-se o ambiente no qual estarão inseridos, assim como os aspectos relacionados com estrutura e política empresarial. Essa tendência - em se observar aspectos relativos ao ambiente organizacional - já vem se manifestando desde o final da década passada [DOB89, DOB90, DOB92, DOB94a, DOB94b]. Assim, durante o processo de especificação dos requisitos, surge a necessidade de se estabelecer o tipo de requisito (funcional, não funcional ou organizacional) de que se esteja tratando, a fim de melhorar a compreensão das necessidades do cliente, bem como melhor modelar esta necessidade.

De forma geral, podemos categorizar os requisitos, em três classes básicas distintas, mas que podem estar relacionadas: funcionais, não funcionais e organizacionais [LOU95, KOT98, DAV93]. Alguns autores preferem classificar os requisitos segundo somente os dois primeiros tipos, funcionais e não funcionais.

Os *requisitos funcionais* dizem respeito à definição das funções que um sistema ou um componente de sistema deva fazer. Ou seja, descrevem as transformações a serem realizadas nas entradas de um sistema ou em um de seus componentes, a fim de que se produza saídas.

Os *requisitos não funcionais* dizem respeito a restrições, aspectos de desempenho, interfaces com o usuário, confiabilidade, segurança, manutenibilidade, portabilidade, padrões, etc., bem como aspectos sociais e políticos. Alguns desses requisitos, com o curso do desenvolvimento, são provavelmente traduzidos em funções ou operacionalizados [CHU99].

De forma geral, a diferença entre requisitos funcionais e não funcionais está no fato dos primeiros descreverem *o que* o sistema deve fazer, enquanto que os outros fixam restrições sobre *como* os requisitos funcionais serão implementados.

Os requisitos funcionais e não funcionais, em princípio devem ser distinguidos numa especificação de requisitos. Todavia, nem sempre é tão simples estabelecer a diferença entre o que seja um requisito funcional ou não funcional. Fazer uma separação

bem definida fica um tanto quanto difícil, pois alguns requisitos podem exprimir necessidades do sistema como um todo e não limitações em funções individuais.

Os *requisitos organizacionais* dizem respeito às metas da empresa, suas políticas estratégicas adotadas, os empregados da empresa com seus respectivos objetivos; enfim toda a estrutura da organização.

Encontra-se ainda sugestões de que requisitos possam ser vistos como sendo "o que" o sistema deve fazer, associado com "o porque" deva fazer, em lugar do "como" o sistema deva fazer. Procura-se ampliar a visão tradicional dos requisitos que trata apenas de aspectos funcionais e não funcionais, com informações organizacionais que abordam a intencionalidade dos fatos, ou seja os requisitos organizacionais [YU94, LOU95, MYL95, YU95b, YU97b].

Segundo essa nova visão, a Engenharia de Requisitos pode ser dita como: *A área do conhecimento preocupada na comunicação com agentes organizacionais, com respeito a suas visões, intenções e atividades relativas as suas necessidades de suporte de computadores, desenvolvimento e manutenção de uma especificação de requisitos adequada a um sistema* [LOU95]. Isso sugere que a Engenharia de Requisitos também inclua problemas e aspectos gerenciais, organizacionais, econômicos, técnicos, sociais e ambientais. Nesse contexto, [BUB95a] salienta que a própria definição da Engenharia de Requisitos evoluiu, ao longo dos anos.

A visão tradicional [LOU95], em geral, esquece aspectos importantes, que influenciam diretamente no sistema que se deseja especificar, tais como:

- (i) os objetivos do próprio sistema e o relacionamento desses com os objetivos das organizações; ou mesmo
- (ii) outras propriedades dos sistemas relacionadas com desempenho, segurança, restrições, etc.; apegando-se em especificar a funcionalidade.

Esta visão mais ampla sobre os requisitos, serve para caracterizar situações nas quais sejam necessárias resoluções de conflitos de requisitos a nível organizacional, optando-se por fornecer prioridades aos requisitos especificados ou por avaliar as alternativas que possam satisfazer estes mesmos requisitos.

Deve ser levado em consideração o fato de que o sistema que esteja sendo especificado é, apenas, um dentre os vários sistemas no ambiente da organização. Para ser compatível com o ambiente e para se adequar aos padrões e preocupações organizacionais, deve-se, também, especificar políticas de implementação que restringem as opções dos projetistas por outros aspectos. Ou seja, os requisitos devem, invariavelmente, abarcar as informações do problema, as diretrizes sobre o comportamento do sistema e as restrições sobre suas propriedades, projeto e construção.

Na verdade, os requisitos organizacionais desempenham papel fundamental no projeto e no desenvolvimento de um sistema. Para Dobson [DOB94b], requisitos organizacionais são aqueles que resultam de um sistema pretendido, considerando-se o contexto social no qual se encontre inserido. Como exemplos, podemos citar: (a) obrigações e responsabilidades, (b) controle e autonomia, (c) valores e éticas, etc., normalmente, embutidos na estrutura e na política da empresa, de forma que não são diretamente observados ou facilmente articulados.

A hipótese básica, para se considerar os requisitos organizacionais, está na ajuda à compreensão de interações complexas entre as empresas (organizações) e as pessoas envolvidas. Assim, esta ajuda torna-se essencial, por oferecer meios:

- (i) para a descrição dos processos da organização; e
- (ii) para a definição e estruturação da informação de suporte a estas aplicações [DAR93, MYL92, MYL95, WIE98].

Embora as muitas classificações propostas tenham muitos pontos em comum, elas diferem de acordo com o ponto de vista dos autores, visto que uma dada classificação pode enfatizar mais um aspecto do que outro. Todavia, a tendência atual aponta para a necessidade de tratar, especificamente, dos requisitos organizacionais [DAR93, LOU95, MYL95, WIE98, YU99].

Até pouco tempo, os projetistas e desenvolvedores ignoravam a importância dos aspectos organizacionais para o desenvolvimento de seus produtos. Só mais recentemente é que esses aspectos começaram a ser observados e foram considerados essenciais ao sucesso das implementações. Hoje, tem-se, nos modelos, ausência de componentes intencionais (por exemplo, as razões, as motivações, os porquês). Os modelos que consideram esses componentes serão objetos de nossas investigações no Capítulo 3.

Na Seção que segue são apresentadas as principais técnicas de modelagem dos requisitos.

2.3 – A MODELAGEM DE REQUISITOS

A modelagem de requisitos realiza importantes papéis:

- (i) ajuda no entendimento da informação, da função e do comportamento de um sistema, tornando a análise de requisitos mais fácil e mais sistemática;
- (ii) é o principal foco para a revisão de requisitos, influenciando na determinação da completude, consistência e precisão da especificação; e
- (iii) é a base para o projeto, fornecendo uma representação essencial do software.

Os modelos variam em função da área de aplicação, todavia devem ser capazes de representar:

- (i) as informações que o sistema a ser especificado transforme;
- (ii) as funções que possibilitam a ocorrência das transformações; e
- (iii) o comportamento do sistema, quando se está desenvolvendo estas transformações.

Em função da natureza cognitiva da modelagem de requisitos, as notações empregadas são conhecidos como modelos conceituais. Uma especificação, desenvolvida em termos de um modelo conceitual, representa abstrações, suposições e limitações sobre um domínio de aplicação.

Apesar de oferecerem uma notação gráfica, que lhes dão uma certa precisão, em geral, os modelos apresentam descrições complementares em linguagem natural, a fim de permitir um maior entendimento do sistema e consequentemente, elucidar mais requisitos. Nesse sentido é que Loucopoulos, afirma que a modelagem conceitual é um facilitador à modelagem de quaisquer especificação de requisitos [LOU95] e, também por isso, é que Mylopoulos define o processo de modelagem conceitual como uma atividade que, formalmente, expõe com precisão aspectos do mundo físico e social, com o propósito de melhorar o entendimento e a comunicação [MYL92]. Salientam que a modelagem

conceitual pode ser usada para todos os tipos de sistemas: informação, embutidos, críticos, de tempo-real, etc.

Os modelos necessários ao desenvolvimento de um sistema dependem do tipo de sistema a ser construído e das pessoas que irão utilizá-los.

De qualquer forma, o desenvolvimento do modelo de requisitos é um processo iterativo, requerendo que o analista busque esclarecimentos dos participantes, do que deve ser acoplado à elicitación dos requisitos. O modelo terá, assim, um duplo papel:

- (i) o de facilitar o entendimento do analista quanto ao problema; e
- (ii) o de ajudar os usuários a verificarem se a compreensão do analista equivale às suas proposições.

Segundo vários autores da área de Engenharia de Requisitos [LOU95, DAV93, SOM97, CAS95], os modelos podem variar desde (i) simples descrições em linguagem natural, passando por (ii) práticas mais rigorosas com modelos semi-precisos, e até (iii) modelos precisos, com toda uma linguagem matemática bem fundamentada. Trataremos neste Capítulo, dos modelos semi-precisos.

Um quadro de classificação, segundo as visões dos vários autores apontados anteriormente, pode ser visto na Tabela 2.1, na qual são apresentados, de forma resumida, os principais modelos semi-formais mais referenciados.

Quanto aos modelos precisos, serão tratados no Capítulo 4, deste trabalho. Nas próximas seções, serão apresentadas as técnicas para a modelagem de requisitos funcionais e não funcionais.

2.3.1 – TÉCNICAS PARA A MODELAGEM DE REQUISITOS FUNCIONAIS

Como visto na Seção anterior, são vários os modelos para a representação dos requisitos de uma dada aplicação. Tradicionalmente, esses modelos costumam ser categorizados, em função dos diferentes pontos de vista, em: (i) estrutural e (ii) comportamental. De forma geral, a tendência verificada é a integração da modelagem de requisitos funcionais sob os pontos de vista estrutural e orientado a objeto.

Principais Notações de Modelagem Semi Precisas				
Métodos Estruturados				
Sommerville [KOT98]	Loucopoulos [LOU95]	Davis [DAV93]	Castro [CAS95]	Rumbaugh [RUM99]
SADT	SADT	SADT	SADT	
Ward-Mellor	MacDonald	Statecharts	CORE	
SSADM	JSD*	Rede de Petri	SREM	
VOSE		DFD		
VORD		SRD		
		SASS		
		RLP	RLP	
Métodos Orientados a Objeto				
Booch	Booch	JSD*	Booch	UML
OMT	OMT		OMT	
	Shlaer-Mellor		Sintropy	
OOSE			BON	

* Alguns autores classificam-na como tradicional, outros como orientada a objeto

Tabela 2.1– Tabela Resumida com os Principais Modelos Semi-Precisos

Quanto às técnicas estruturadas de modelagem de requisitos funcionais, destacamos as técnicas:

- (i) CORE (Controlled Requirements Expression) [MUL79, CAS95];
- (ii) VOSE (Viewpoint-oriented System Engineering) [FIN92]; e
- (iii) VORD (Viewpoint-oriented Requirements Definition) [KOT96].

Dentre as técnicas para a modelagem dos requisitos funcionais, destacamos ainda as técnicas ditas orientadas a objeto, as quais classificamos quanto a geração cronológica como:

- (i) *técnicas de primeira geração*, onde destacamos as técnicas: a JSD (Jackson Systems Development) de Michael Jackson [JAC83], OMT (Object Modeling Technique) de Rumbaugh [RUM91], a OOSE (Object-Oriented

Software Engineering) de Jacobson [JACO92] e a de Shlaer-Mellor [SHL92];

- (ii) *técnicas de segunda geração* onde destacamos as técnicas: de Booch [WHI94], BON [WAL95] e Sintropy [COO94]; e
- (iii) *técnicas de terceira geração*, onde destacamos a técnica UML (Unified Modeling Language) [JACO99, BOO99, RUM99].

Associada à técnica UML existe a Linguagem de Restrições de Objetos (OCL), que será descrita na Subseção 4.4.2.

2.3.1.1 – A Técnica de Modelagem Unificada – UML

A técnica UML (Unified Modeling Language) [JACO99, BOO99, RUM99] além de ser hoje um padrão para especificar, visualizar, documentar e construir artefatos de um sistema, pode ser utilizada em todos os processos, ao longo do ciclo de desenvolvimento e, por meio de diferentes técnicas de implementação.

A primeira iniciativa de unificação dos métodos orientados a objeto, aconteceu por volta de 1995, com James Rumbaugh (OMT) e Grady Booch, através de iniciativa da Rational Corporation [BOO95]. Posteriormente, juntou-se a eles o Ivar Jacobson. Juntos criaram a linguagem de modelagem unificada, UML [JACO99, BOO99, RUM99], para tratar aspectos de grande escala, próprios de sistemas complexos e críticos, poderosa o bastante para modelar qualquer aplicação de tempo real, cliente/servidor, dentre outras.

Vale ressaltar que não se trata de uma metodologia, mas sim de uma linguagem de modelagem, que possui um meta-modelo bem definido. Atualmente, começam a surgir propostas de metodologias para desenvolvimento com a linguagem UML, tais como:

- (i) RUP (Rational Unified Process) [KRU98]; e
- (ii) Catalysis [DSOU99].

Quanto aos diagramas propostos em UML, temos:

- (i) Diagrama de Caso de Uso;
- (ii) Diagrama de Classe;
- (iii) Diagrama de Objeto;

- (iv) Diagrama de Seqüência;
- (v) Diagrama de Colaboração;
- (vi) Diagrama de Statechart;
- (vii) Diagrama de Atividade;
- (viii) Diagrama de Componente; e
- (ix) Diagrama de Implantação.

A título de exemplificação abordamos apenas os dois primeiros tipos de diagramas. Um leitor interessado no assunto deve consultar [JACO99, BOO99, RUM99]. O primeiro diagrama trata dos Caso de Uso. Conforme pode ser visto na Figura 2.7, temos um exemplo típico de Caso de Uso, que pode caracterizar uma situação do sistema de controle para uma Loja de Venda de CD's, que entre outro serviços permite que clientes previamente cadastrados possam consultar o acervo de CD's existente na Loja.

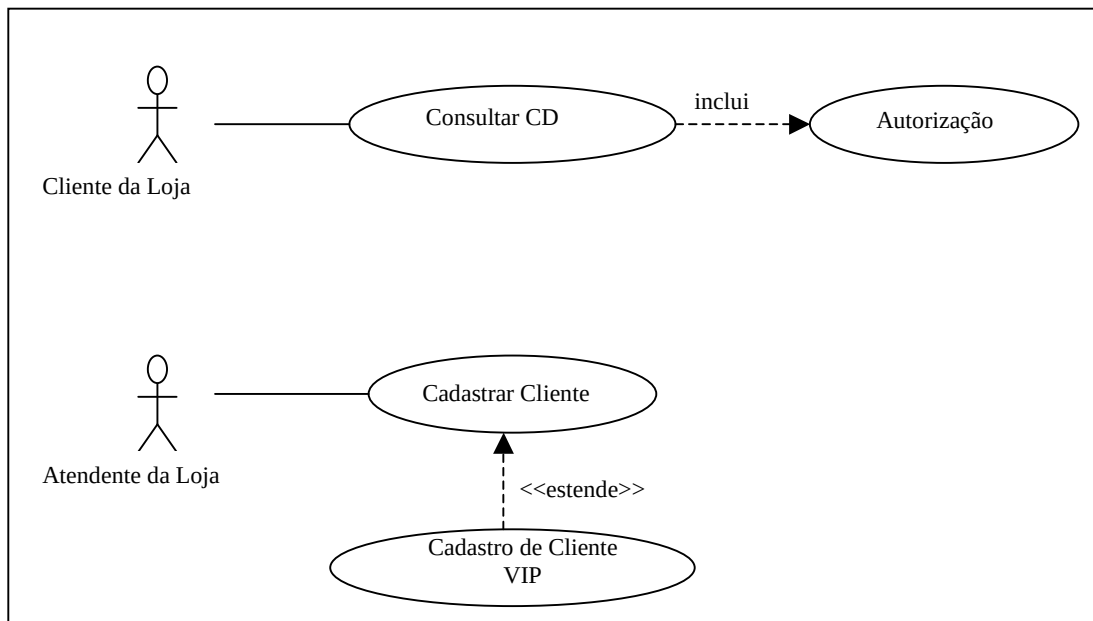


Figura 2.7 – Cenário Típico de Caso de Uso para a Loja de Venda de CD's, em UML

Neste Diagrama de Caso de Uso, o cliente deseja consultar o sistema de controle da loja, a fim de identificar o CD desejado. Todavia para realizar essa atividade, faz-se necessário que ele possua uma autorização, pois nem todos clientes da loja terão acesso ao sistema. Para que isso aconteça é preciso que um funcionário da loja cadastre o cliente

como um cliente. Clientes especiais terão certos privilégios ou regalias. Portanto, se faz uma extensão do Caso de Uso “Cadastrar Cliente”, para contemplar os clientes VIPs.

O Diagrama de Classes, apresenta as classes com os atributos e operações que cada uma de suas instâncias possui, arranjadas em hierarquias que compartilham estrutura e comportamento que estão associados a outras classes. São também levados em consideração os conceitos fundamentais de estruturação que incluem dentre outros a agregação e a generalização, conforme pode ser visto na Figura 2.8.

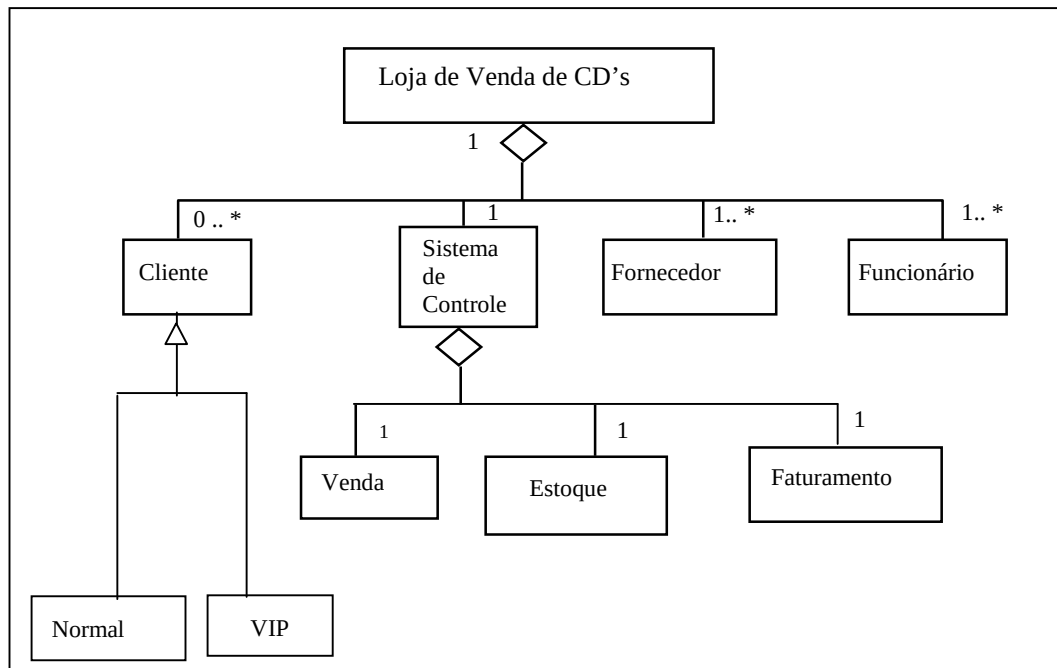


Figura 2.8 – Diagrama de Classes da Loja de Venda de CD's com Agregação e Generalização

Nesta Figura, podemos ver que o “Sistema de Controle” da loja é composto por partes específicas para controle de venda, estoque e faturamento e que, por outro lado, precisa diferenciar o que seja um cliente “Normal” e um cliente “VIP”.

Em UML são definidos elementos cujas notações podem ser empregadas nos diversos diagramas, sendo os principais: pacote, estereótipo e anotação. Pacote é um mecanismo para organizar elementos do modelo em grupos, tendo por base as categorias de classe. Estereótipo significa a introdução de novos elementos no meta-modelo, para permitir que os usuários estendam a capacidade de modelagem da linguagem. E

finalmente, anotação é um comentário, colocado em um diagrama, sem qualquer conteúdo semântico. Maiores detalhes podem ser encontrados em [BOO99, JACO99, RUM99].

Os responsáveis pela padronização da Linguagem de Modelagem Unificada (UML) definiram a linguagem através de uma abordagem de meta-modelo para tornar sua sintaxe mais precisa [KEN99]. Na verdade, um meta-modelo para uma linguagem é uma definição da linguagem escrita em termos da mesma. O meta-modelo UML tem pouco ou nenhum conteúdo semântico, embora seja acompanhado por comentários em linguagem natural, que dá, informalmente, os significados dos vários construtores.

Um aspecto importante na linguagem UML é o reconhecimento, por parte de seus autores, da necessidade de uma descrição precisa de sua semântica. Isso resultou num documento semântico [UML99], controlado pela OMG (Object Management Group), que se constitui em parte importante na definição padrão da linguagem. Esse documento semântico significa um passo importante na direção de se fornecer uma descrição precisa de UML. A abordagem tomada no documento semântico é a de fornecer uma descrição em meta modelo da linguagem, sendo apresentada em termos de três visões:

- (i) *a sintaxe abstrata*, expressa usando-se um subconjunto das notações de modelagem estática de UML;
- (ii) *um conjunto de regras bem formadas*, expressas na Linguagem de Restrições de Objetos (OCL); e
- (iii) *a semântica dos elementos de modelagem*, descrita em linguagem natural.

O significado da linguagem, descrita em linguagem natural, é muito informal. Evans e Kent descrevem em [EVA99] uma proposta, que está em execução por um grupo de pesquisadores internacionais, o grupo do UML preciso (Precise UML), visando uma estratégia de formalização que se concentra em fornecer uma semântica denotacional precisa, para os elementos do núcleo (“core”) de UML [pG99]. Como UML tem um amplo escopo, é natural que tentemos nos concentrar nos conceitos essenciais da linguagem, para construir uma fundamentação mais clara e precisa, como base para a formalização. Neste sentido é que a proposta do grupo pUML concentra-se na identificação e formalização de um modelo semântico do núcleo (core) de UML. Surge então, uma pergunta natural: Que partes são incluídas como núcleo de UML? Essa questão, segundo Evans e Kent, foi

parcialmente respondida no documento da semântica de UML [UML99]. Neste documento são identificados:

- (i) um pacote para Relacionamentos-Núcleo; e
- (ii) vários pacotes para o Comportamento Comum.

O pacote de Relacionamento Núcleo define um conjunto de elementos do modelo que são comuns a todos os diagramas de UML, tais como:

- (a) relacionamento;
- (b) classificador;
- (c) associação; e
- (d) generalização.

Já o pacote de Comportamento Comum fornece um significado denotacional parcial aos elementos do modelo, no pacote núcleo. Segundo o exemplo fornecido em [EVA99], este pacote descreve uma associação entre classificadores e instâncias, estabelecendo a conexão entre a representação de um classificador e seu significado, que é uma coleção de instâncias.

O grupo de pUML também está buscando dar uma semântica para a linguagem de restrições de objeto – OCL [WARM99], considerando apenas os aspectos das restrições estruturais em UML. No Capítulo 4, abordaremos com mais profundidade esse aspecto.

2.3.2 – TÉCNICAS PARA A MODELAGEM DE REQUISITOS NÃO FUNCIONAIS

Sabe-se que, durante o desenvolvimento, um sistema de software é determinado, fundamentalmente, por sua funcionalidade e, parcialmente, pelos requisitos globais relacionados com custo operacional, desempenho, segurança, confiabilidade, manutenibilidade, portabilidade, robustez e outras mais.

Indubitavelmente, os requisitos não funcionais assumem papel crucial no processo de desenvolvimento e têm grande influência na qualidade do sistema a ser desenvolvido. Há casos em que os requisitos não funcionais representam a própria essência do sistema, como seria o caso dos sistemas integrados em grandes sistemas - os chamados sistemas críticos e de tempo real -, tais como: (i) sistemas de controle de tráfego aéreo; (ii) sistemas

de controle de processos; e (iii) sistemas de monitoração de pacientes. Assim, podemos dizer que, em geral, principalmente em sistemas mais complexos e de tempo-real, os requisitos não funcionais são de extrema importância.

Embora esse tipo de requisitos seja de reconhecida relevância, quando se trata de técnicas para a especificação, dominam, quase que, totalmente, as abordagens que não contemplam esses requisitos. Poucas são as técnicas que fornecem algumas diretrizes para a sua captura e representação, ou mesmo que mostrem a influência que esses requisitos têm no processo [CHU99]. Fato é que os requisitos não funcionais não são, adequadamente, cobertos pelas várias técnicas da Engenharia de Requisitos propostas e têm merecido pouca atenção por parte dos pesquisadores, pelo que, são mal compreendidos.

Algumas propostas para tratar os requisitos não funcionais, foram feitas: (i) Chung [LEE91] propôs um modelo orientado a objetivos; (ii) Dobson [DOB90, DOB94a] sugeriu o desenvolvimento de modelos lógicos para tratar os requisitos não funcionais do sistema; (iii) Kotonya [KOT98], juntamente, com o Sommerville, propuseram uma ferramenta baseada em pontos de vista que integra requisitos funcionais e não funcionais; e (iv) Mylopoulos [MYL92] propôs uma ferramenta com base no modelo de Chung e no ambiente DAIDA [JAR92]. Todas as propostas objetivam transformar objetivos de alto nível e desejos, em propriedades mensuráveis.

Destacamos uma iniciativa para a modelagem de requisitos não funcionais apresentada por Mylopoulos [MYL92] com base no trabalho de Lee [LEE91], cuja força é medida através de vários estudos de caso, para sistemas de informação, realizados por diferentes autores: [CYS97, CYS98, LAW95]. Como aplicação da ferramenta desenvolvida por Mylopoulos, tem-se alguns trabalhos relacionados:

- i. O que trata de *metas de desempenho* para sistemas de informação [NIX93];
- ii. O que trata *requisitos de segurança* como uma classe de requisitos não funcionais [LAW93];
- iii. O que trata *requisitos de qualidade* [LAW95]; e
- iv. O que trata de *requisitos não funcionais* para análise de modelos orientados a objeto [CYS97, CYS98].

2.4 – PROBLEMAS EM ABERTO NA ÁREA DE ENGENHARIA DE REQUISITOS

Embora a área de Engenharia de Requisitos tenha apresentado um grande avanço nos últimos anos, vários problemas ainda persistem, como por exemplo:

1. *Necessidade de Envolvimento das Partes Participantes;*

Apesar dos processos da Engenharia de Requisitos objetivarem identificar as necessidades dos clientes, na prática ainda se sofre com problemas de identificação das reais necessidades e interesses das diferentes pessoas que influenciarão ou serão influenciadas pelo sistema que está sendo especificando. Isso porque muitas vezes não há envolvimento completo entre as partes.

2. *Necessidade de Gerenciamento e Rastreamento de Requisitos;*

Os processos existentes, não incluem técnicas efetivas para o gerenciamento e rastreamento dos requisitos, significando que alterações podem ser introduzidas de forma aleatória, conduzindo a perdas de tempo e esforço muito grande em se tentar compreender as possíveis mudanças dos requisitos [TOR99].

3. *Necessidade de Definição das Responsabilidades;*

Em geral, nos processos da Engenharia de Requisitos existem diferentes pessoas envolvidas que, normalmente, desconhecem suas reais responsabilidades, significando que existe a possibilidade de que tarefas não sejam executadas ou quando executadas, não sejam da forma adequada, devido a essa falta de definição.

4. *Problemas de Comunicação com as Partes Participantes;*

Em função do grande número de pessoas envolvidas, as partes podem não se comunicar eficientemente. Esse fato pode resultar na elaboração de um documento de requisitos que não seja compreendido por todos, além da possibilidade de haver requisitos incompletos ou mesmo incorretos.

5. *Os Aspectos da Organização não são Considerados.*

Em geral, os processos da Engenharia de Requisitos estão preocupados apenas com os aspectos técnicos referentes aos sistemas que se deseja especificar. Todavia, por não serem considerados aspectos relacionados com as organizações, pode-se deixar de compreender amplamente as reais necessidades e intenções dessas organizações, o que concorre, em muito, para que os requisitos especificados não satisfaçam as reais necessidades do cliente.

Tendo identificado esses problemas, resolvemos focar este último, identificando as técnicas especificamente relacionadas com os *Requisitos Organizacionais*, objetivando formalizar meios para que esses requisitos possam ser incorporados às especificações de requisitos existentes, que em geral, tratam dos aspectos técnicos dos sistemas a serem especificados e desenvolvidos.

CONSIDERAÇÕES

Neste Capítulo apresentamos uma retrospectiva das principais prerrogativas, conceitos e etapas do processo da Engenharia de Requisitos, onde levantamos a necessidade de melhor compreensão: do domínio da aplicação; do problema específico a ser resolvido; das necessidades e limitações organizacionais; e das facilidades específicas, necessárias às partes interessadas.

Na Engenharia de Requisitos, destacamos que o ponto básico recai na compreensão daquilo que se deseja desenvolver, ou seja, na necessidade de se definir e classificar o que se espera do sistema. Observamos então, que os requisitos tradicionalmente só poderiam ser de duas naturezas: funcional e não funcional, mas que trabalhos recentes identificam uma categoria muito importante de requisitos, os chamados requisitos organizacionais.

Particularmente, nosso trabalho encontra-se inserido na fase de modelagem de requisitos, no que se refere aos requisitos organizacionais, segundo a classificação das atividades do processo da Engenharia de Requisitos estabelecida por Sommerville

[KOT98], vide Figura 2.6, questionando que os aspectos da organização precisam ser considerados.

No próximo Capítulo trataremos especificamente das técnicas para a modelagem de requisitos organizacionais por representarem parte do foco central de nosso trabalho.

CAPÍTULO III

TÉCNICAS DE MODELAGEM ORGANIZACIONAL

Reservamos este Capítulo para tratarmos exclusivamente das técnicas de modelagem dos requisitos organizacionais, em razão deste nosso trabalho basear-se nesta atividade do processo da Engenharia de Requisitos, especificamente, com respeito a esse tipo de requisito.

Conforme visto no Capítulo anterior, o processo de modelagem é central ao processo de especificação dos requisitos. Modelos são criados com o objetivo de se ter uma melhor compreensão de uma entidade do mundo real, a qual se deseja construir.

Nas seções que se seguem discutiremos sobre a modelagem, em especial como se classificam os modelos e as técnicas para a modelagem dos requisitos organizacionais. Destacaremos as técnicas de modelagem de requisitos organizacionais, por serem esses tipos de requisitos o principal objeto de nosso trabalho.

3.1 – PRINCIPAIS TÉCNICAS DE MODELAGEM DE REQUISITOS ORGANIZACIONAIS

O principal obstáculo à captura correta dos requisitos de um sistema tem sido a dificuldade em se obter uma compreensão mais aprofundada do domínio da aplicação. Neste contexto, a modelagem organizacional facilita a compreensão do ambiente empresarial e é reconhecida como atividade valiosa, pela Engenharia de Requisitos. A Engenharia de Requisitos, atuando sobre os problemas reais das organizações, verifica a importância de se modelar o ambiente organizacional para se obter requisitos úteis à estrutura do negócio [NEL94, MYL95, REM95, PET97, SOU98, JUR99, YU99].

O *modelo organizacional* é uma representação da estrutura, atividades, processos, informações, recursos, pessoal, comportamento, objetivos e restrições das empresas comerciais, governamentais ou de outra natureza, a fim de ajudar a compreender as complexas interações entre as organizações e as pessoas. Ressalta-se

que, em cada organização, os processos de negócios apresentam características próprias e, por esta razão, é importante se dar atenção à modelagem e à reengenharia desses processos [NIC96, FOX98]. Assim, destaca-se que os objetivos da modelagem de negócio são os de:

- i. Fornecer um objeto que seja uma representação compartilhável e reusável da cadeia de fornecimento de informação e conhecimento;
- ii. Suportar tarefas da cadeia de fornecimento, pela habilitação de respostas a questionamentos que não estão explicitamente representados no modelo;
- iii. Definir os objetos de maneira precisa, de forma que sejam consistentemente aplicados, através dos domínios e interpretados pelos usuários;
- iv. Suportar visualização do modelo, de forma intuitiva, simples e consistente.

Do ponto de vista da Engenharia de Requisitos, muitas são as técnicas propostas. Dobson [DOB94a] descreve os objetivos, política e estrutura da organização tendo como foco principal os aspectos sociais. Já outros autores [YU93, BUB94a, YU94] preferem trabalhar com múltiplas visões, ou mesmo, com a análise de metas e objetivos para a modelagem da organização [MYL92, DAR93, YU94, MYL99, YU99]. As organizações, segundo essas técnicas, são representadas a partir da elaboração de modelos. Para estes autores, a modelagem organizacional torna-se uma tarefa facilitadora para se chegar, gradualmente, às especificações de requisitos, funcionais e não funcionais, melhores e mais próximas dos requisitos das próprias organizações. Cada técnica propõe uma nomenclatura específica, gerando uma variedade de notações, o que dificulta a uniformização da representação de informações pelas diferentes partes envolvidoras do sistema [CAS97].

Destacamos algumas técnicas de modelagem organizacionais existentes, na área da Engenharia de Requisitos:

- i. A técnica ORDIT (Organisational Requirements Definition of Information Technology Systems) [ORD93, DOB94b, EAS97], para ajudar os participantes das organizações a definirem alternativas técnicas e o futuro organizacional, e, consequentemente, o papel da tecnologia da informação, fornecendo um processo sistemático e aproveitável, que seja capaz de suportar gerações de requisitos organizacionais e fornecer métodos e ferramentas associadas que suportem o processo;
- ii. A técnica de Bubenko, apresentada, originariamente, em [BUB83], destaca áreas de conhecimento da organização. É constituída por cinco submodelos [KIR94], elaborados a partir de objetivos (*Modelo de Objetivos* - MO), atores (*Modelo de Atores* - MA), atividades e uso (*Modelo de Atividades e Uso* -MAU), conceitos (*Modelo de Conceitos* - MC) e requisitos (*Modelo de Requisitos do Sistema de Informação* - MRSI)];
- iii. A técnica KAOS (Knowledge Acquisition in autOmated Specification) [DAR93, LAM95] é uma metodologia da Engenharia de Requisitos que fornece uma linguagem formal, precisar de mesmo nome, rica para a captura de requisitos funcionais, não funcionais e organizacionais, capaz de fornecer facilidades a descrição de objetivos, agentes, alternativas, eventos, ações, modalidade de existência e responsabilidades de agentes. Trata-se de uma ferramenta para aquisição/elaboração de requisitos direcionada a objetivos, onde é forte o conceito de agentes. Explora muitos dos construtores utilizados pela notação da Análise Orientada a Objetos, cujo projeto final consiste na atribuição de responsabilidades a agentes.

Na mesma linha de trabalho da técnica KAOS, mas com uma visão mais ampla sobre objetivos e sobre o relacionamento estratégico entre os vários agentes, temos a técnica i* [YU95a], que será abordada na Seção 3.2.

Na área das ciências administrativas encontramos, como técnicas de modelagem da organização, fundamentalmente:

- i. A Cadeia de Valores, instrumento básico para diagnosticar a vantagem competitiva e descobrir maneiras de criá-la e sustentá-la, mas que também pode ter papel valioso no projeto da estrutura organizacional. Isso porque a Cadeia de Valores oferece uma forma sistemática de dividir uma empresa em suas atividades distintas, podendo ser utilizada para examinar como são as atividades em uma empresa e como poderiam ser agrupadas [POR99]. Na Figura 3.1, podemos visualizar uma Cadeia de Valores com a superposição de uma estrutura organizacional típica, onde na horizontal, temos as atividades gerais a toda a empresa (elos horizontais) – Infra-estrutura, Recursos Humanos, Desenvolvimento de Valores com a superposição de uma estrutura organizacional típica, onde na horizontal, temos as atividades gerais a toda a empresa (elos horizontais) – Infra-estrutura, Recursos Humanos, Desenvolvimento de

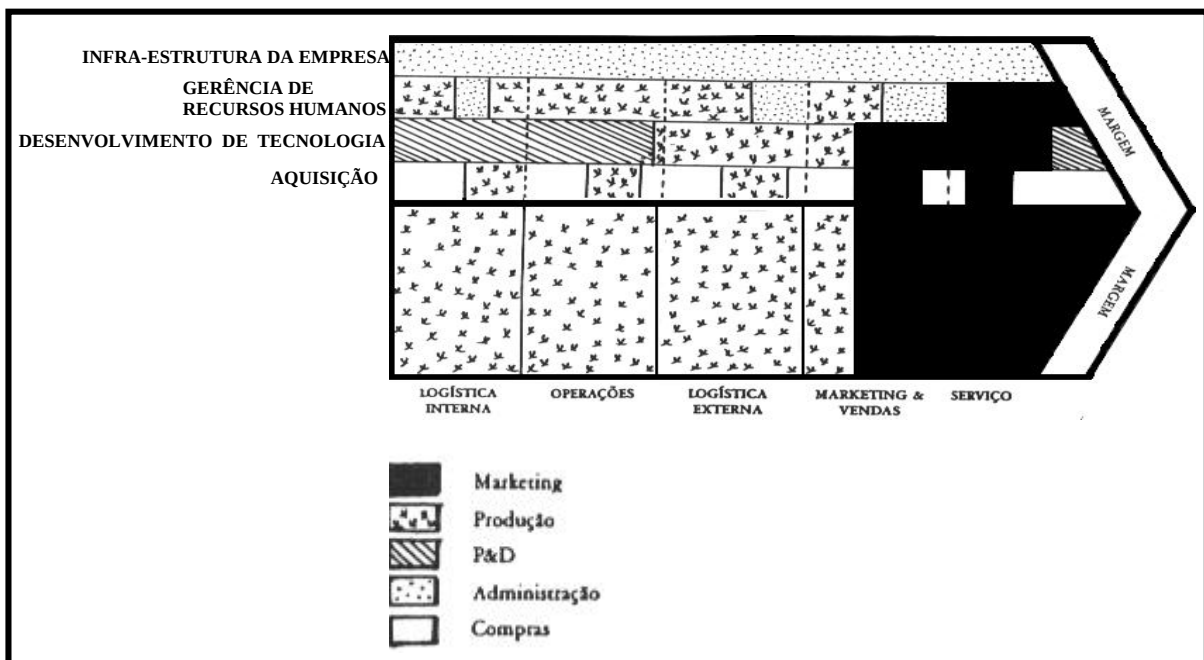


Figura 3.1 – Estrutura Organizacional e Cadeia de Valores

Tecnologia e Aquisição - enquanto na vertical temos as atividades específicas (elos verticais), dentro de um processo - Logística Interna, Operações, Logística Externa, Marketing e Vendas, Serviço.

- ii. Organogramas:
São diagramas que descrevem as posições dentro da organização, em termos das funções de negócio e do nível de gerenciamento existentes.
- iii. Diagramas de Fluxo de Trabalho e suas variações:
São diagramas que descrevem como as coisas são feitas, identificando as entidades presentes e os passos envolvidos em suas execuções, o que caracteriza a descrição dos processos. A nível de usuário, um diagrama de fluxo de trabalho é criado através de entrevistas realizadas com os usuários do sistema e envolvem apenas entidades (internas e externas) e fluxo de trabalho (não dados). Segundo [NIC96], os diagramas de fluxo de trabalho podem ser classificados como: (a) Ad hoc; (b) Administrativo (Orientados a Pessoas); (c) Produção (Orientados a Sistemas); e (d) Transacional.
- iv. Modelagem Objeto do Negócio:
Esse tipo de modelagem identifica os candidatos a classes, para os objetos do sistema. Esses modelos agrupam os candidatos a classes por: (a) Processo; (b) Área Funcional; e (c) Outro tópico relevante. Existe, segundo Mike Heroux [HER99], duas perspectivas que ajudam na identificação de classes: (a) Dados; e (b) Eventos.

Estas técnicas, do ponto de vista da Ciência da Administração, são utilizadas para a modelagem organizacional, sendo utilizadas, por exemplo, no desenvolvimento dos sistemas ERP (Enterprise Resource Planning) [HAB99]. Tem sido prática comum nas organizações empresarias a utilização desses sistemas. ERP é um termo genérico para um conjunto de atividades que são executadas por um software multi-modular, cujo objetivo é o de ajudar o gestor de uma empresa nas diversas fases do seu negócio. Esse tipo de software também permite incluir: (i) desenvolvimento de produto; (ii) compra de itens; (iii) interação com fornecedores; (iv) serviços a cliente e acompanhamento de ordens de produção; e (v) módulos de aplicativos para os aspectos financeiros e de gestão de recursos humanos. Esse tipo de sistema permite que as empresas padronizem

seus sistemas de informação e reconhece as necessidades das empresas em reduzir o tempo de resposta ao mercado de produtos e serviços. Os sistemas ERP são desenvolvidos para responder instantaneamente ao surgimento de novas necessidades não previstas originariamente.

Para a construção desse tipo de sistemas, surgiu a SAP (Sistemas, Aplicações e Produtos para Processamento de Dados) [SAP], uma das empresas especializadas no fornecimento de software para aplicações em negócios. Hoje, dentre os vários produtos da SAP, tem-se o sistema R/3. Este sistema permite integrar todas as operações do negócio, por exemplo: (i) contabilidade e controle; (ii) gerenciamento da produção e dos materiais; (iii) gerenciamento da qualidade; (iv) gerenciamento do planejamento e salários; (v) gerenciamento de recursos humanos; e (vi) gerenciamento de projetos. Trata-se de um sistema de extrema aplicabilidade e fundamental no mundo dos negócios. Hoje, as empresas precisam de sistemas cada vez mais “inteligentes”. Neste caso, os sistemas de software passam a ser de extrema necessidade e precisam ser bem desenvolvidos para garantir sua qualidade. Assim, é importante destacarmos a necessidade de que os desenvolvedores de sistemas de software possuam uma compreensão mais aprofundada sobre a organização, seus objetivos, metas e estratégias de mercado. Deste modo, os sistemas construídos se adequarão às reais necessidades da empresa.

Por outro lado, devemos lembrar que sob o ponto de vista da ciência da Administração, não basta apenas encher-se as empresas com equipamentos ou sistema de informação de última geração, sem que haja preocupação de atualização dos próprios processos do negócio. É preciso uma reavaliação dos processos de negócio. Neste sentido é que, dentre as técnicas da ciência da Computação, encontramos a técnica i^* , que nos permite não só melhor entender requisitos organizacionais que interferirão nos sistemas, mas também identificar alternativas para os vários processos da organização. Trata-se de uma técnica, ao nosso entender, mais rica. Nas seções que se seguem, abordaremos especificamente a técnica i^* , adotada neste trabalho.

3.2 - A TÉCNICA DE MODELAGEM ORGANIZACIONAL I*

Observamos, antes de tudo, que ao procurarmos compreender uma organização, a fim de melhor desenvolvermos o sistema desejado, os conhecimentos capturados nos diversos modelos existentes (DFD, ER, Statechart, etc.) não são suficientes, porque a maioria destes modelos descrevem apenas entidades, funções, fluxo de dados e estados do sistema. Eles não conseguem expressar as razões envolvidas no processo, ou seja, o porquê fazer uma determinada ação ou tomar uma decisão (alternativas para o “como fazer”). Assim, faz-se necessário uma ontologia mais rica, capaz de reconhecer motivações, intenções, raciocínios sobre as características de um processo e que facilite os esforços da Engenharia de Requisitos. A técnica i* foi proposta por Eric Yu em sua tese de doutorado em [YU95a], com esse objetivo. Ela permite a descrição dessa ontologia e é utilizada para: (i) obter uma melhor compreensão sobre os relacionamentos da organização, entre os vários atores do sistema; (ii) entender as razões envolvidas nos processos de decisões; e (iii) ilustrar as várias características de modelagem que podem ser apropriadas à Engenharia de Requisitos, principalmente, na fase inicial da especificação dos requisitos [YU95b,YU97a].

Em seu trabalho, Yu aplica a técnica i* em quatro áreas, fazendo uma comparação com as ferramentas existentes em cada uma dessas áreas, a saber:

(i) Engenharia de Requisitos

Um dos principais obstáculos à obtenção dos requisitos certos está na dificuldade de compreensão, mais aprofundada, do domínio da aplicação. Os modelos de requisitos que existem, normalmente quando descrevem um ambiente organizacional, apenas o fazem em termos de entidades e atividades, não capturando muitas preocupações que os usuários possam ter quanto as implicações em se adotar uma solução em lugar de outra. Com as motivações e razões, explicitamente capturadas num modelo de requisitos, fica mais fácil evoluir o sistema para alcançar as mudanças nas necessidades do usuário.

(ii) Reengenharia de Processos de Negócio

A comunidade na área de administração e negócio, tornou-se atenta para o fato de que a Tecnologia da Informação não deve apenas ser utilizada para a automação dos processos de negócio existentes, mas também servir de base para a reformulação desses processos, a fim de encontrar objetivos de negócio mais amplos (concorrência, competitividade, estratégias etc.). Neste sentido, Peter Drucker [DRU99] diz que “a administração não precisa de mais informações sobre o que está acontecendo internamente. Precisa de mais informações sobre o que está acontecendo lá fora”; prevendo que “o principal desafio da informática nos próximos 30 anos será o de organizar o fornecimento sistemático de informações externas significativas”. Assim é que, segundo Yu, um princípio central da reengenharia está na necessidade de perguntar o “porquê” dos fatos. Sem o claro entendimento das razões por detrás das práticas e das estruturas existentes, não se pode decidir, facilmente, que mudanças poderão ser feitas nos processos de negócio. Os tipos predominantes de modelos utilizados são variações dos diagramas de fluxo de trabalho e atividades.

(iii) Análise de Impactos Organizacionais

O pessoal participante de um Sistema de Informação pode ter um amplo conjunto de preocupações quanto ao sistema e como este poderá alterar o ambiente de trabalho, pois podem existir funções que poderão ser parcial ou totalmente afetadas pela incorporação de um novo sistema de informação. Esse fato, inclusive, pode aumentar ainda mais a pressão social e a disciplina da organização. Os modelos convencionais usados no desenvolvimento de sistemas, segundo Yu, foram primariamente projetados para a descrição de sistemas técnicos e não para fornecer descrições mais ricas sobre as organizações sócio-humanas. A técnica i* foi concebida com a idéia de se considerar os aspectos organizacionais, facilitando os esforços durante o desenvolvimento do sistema de

informação e permitindo que a análise organizacional seja melhor integrada aos processos de desenvolvimento do sistema.

(iv) Modelagem do Processo de Software

É de conhecimento geral que existem muitos esforços no desenvolvimento de modelos de processos de desenvolvimento de software. Todavia, deve-se ter em mente que, para se alcançar o sucesso no desenvolvimento de um software de alta qualidade, é importante se levar em conta não só as ferramentas para o suporte técnico, mas, também, o ambiente organizacional. Assim, um modelo de processo de software que capture motivações e razões, envoltas nas atividades e fluxos, facilitará as análises sistemáticas e o projeto dos processos de software. A diferença com relação à primeira aplicação está no fato daquela se prender especificamente à fase de requisitos, no processo de desenvolvimento de software, a fim de que se tenha uma compreensão mais aprofundada.

Nesta técnica, as organizações são vistas como se consistissem de unidades semi autônomas chamadas de atores, cujo comportamento destes não é totalmente controlável ou previsível, mas regulamentado por relacionamentos sociais. Os atores ou participantes do processo têm liberdade de ação, dentro do contexto de restrições sociais, e possuem propriedades intencionais, tais como, objetivos, crenças, habilidades e compromissos. Muito crucialmente, esses atores dependem uns dos outros para terem seus objetivos alcançados e suas tarefas realizadas, além de objetivos que, individualmente, não seriam alcançados, mas que tratados em conjunto podem passar a ser. Nesse contexto, tem-se uma técnica que procura a compreensão mais aprofundada do processo através de uma visão intencional e estratégica. Assim, podemos afirmar que um *ator é intencional* quando não executa simplesmente uma atividade e/ou produz ou consome uma entidade, mas possui motivações, intentos e razões por detrás de suas ações. Um *ator é estratégico* quando não foca apenas seu objetivo imediato, mas quando preocupa-se com as implicações de seu relacionamento estrutural com outros

atores, ou seja, as oportunidades e vulnerabilidades presentes, devido a falhas que possam existir na relação social. Quando tratamos com técnicas que não consideram o contexto intencional, temos: (i) atividades simplesmente sendo executadas; (ii) a não existência da noção de sucesso ou falha; ou (iii) falta de noção de objetivos que possam ser alcançados por diferentes meios.

A técnica i^* é composta por dois modelos: o *Modelo de Dependências Estratégicas (SD)* e o *Modelo de Razões Estratégicas (SR)*. O Modelo de Dependências Estratégicas (SD) descreve as relações de dependências externas entre os atores da organização, enquanto o Modelo de Razões Estratégicas (SR) descreve interesses e conceitos dos participantes e as direções que podem seguir.

3.2.1 – O MODELO DE DEPENDÊNCIAS ESTRATÉGICAS (SD)

O Modelo de Dependências Estratégicas (SD) consiste em um conjunto de nós e ligações entre esses, onde os nós representam os atores e cada ligação indica uma dependência entre dois atores. Assim, um processo é descrito em termos de uma malha de relacionamentos de dependências entre os vários atores, capturando-se as motivações e os desejos envolvidos nas atividades. Neste modelo, distingue-se quatro tipos de dependências:

- (i) Recursos, que estão relacionados com a noção de entidades;
- (ii) Tarefas, que dizem respeito às atividades;
- (iii) Objetivos, que são afirmações; e
- (iv) Objetivos-Soft, que estão relacionados com a noção de requisitos não funcionais, visto na Seção 2.2.

Na *dependência de recurso*, o agente depende da disponibilidade de uma entidade física ou de uma informação. Por *recurso*, entendemos o produto final de alguma ação, em um processo, que estará ou não disponível. Neste tipo de dependência, assume-se que não haja aspectos pendentes a serem tratados ou decisões a serem tomadas. Por exemplo, o Cliente para receber a encomenda de CD's precisa pagar por esta, assim o pagamento é um recurso necessário a efetuação da compra.

Na *dependência de tarefa*, um agente informa ao outro o que deve ser feito, sem haver preocupação em se informar o “porque” fazer. A descrição de uma tarefa em i^* não tem por intenção ser uma completa especificação dos passos necessários à execução dessa tarefa: relaciona-se com as atividades a serem realizadas. Por exemplo, o fornecedor deve *enviar os CD solicitados*.

Na *dependência de objetivo*, um agente depende de outro para que uma determinada intenção sua seja satisfeita, não importando a maneira como seja alcançado esse objetivo. Por *objetivo* se entende uma condição ou estado do mundo, que um ator (agente) gostaria de alcançar. Em linguagem natural, um objetivo é expresso como uma desejo, uma afirmação; por exemplo, a loja de venda de CD's, deseja *aumentar a clientela*.

A *dependência de objetivo-soft* é similar à dependência de objetivo, exceto pelo fato de que a condição de sucesso não é, precisamente, definida a priori, ou seja, os objetivos envolvem aspectos subjetivos que vão sendo esclarecidos, através de métodos que são escolhidos, durante o desenvolvimento do processo. Esse tipo de dependência fornece um importante elo de ligação entre dois aspectos importantes: (i) o lado técnico e (ii) o lado gerencial. Por exemplo, o Cliente para voltar à Loja espera ser bem atendido, portanto atendimento bom é um objetivo-soft pois não se consegue definir precisamente o que é ser bem atendido.

De uma forma geral, esses tipos de dependências são usados para indicar a natureza de liberdade e controle existentes no relacionamento entre dois atores, ou seja, os quatro tipos de dependências caracterizam como as decisões recaem sobre um dos lados da dependência, indicando quem resolverá os problemas que surgirem. Graficamente, essas dependências são expressas como exemplifica a Figura 3.2, onde se tem vários atores. Considere os relacionamentos entre o ator *Loja* e o ator *Cliente*. A Loja depende do Cliente para que este faça a compra de CD's na Loja; sendo assim, há o relacionamento do tipo objetivo - “Compre CD's”. O Cliente depende da Loja para receber o disco solicitado, no caso uma dependência do tipo recurso - “CD Solicitado”. Outro objetivo importante para o Cliente é que a Loja tenha os títulos que ele esteja interessado, no caso uma dependência do tipo objetivo - “Ter CD Desejado”. Outro

objetivo do Cliente, desta vez, não tão preciso, é com respeito a um serviço de qualidade, no caso uma dependência de objetivo-soft - “Qualidade [Serviço]”.

Os atores envolvidos numa relação de dependência foram denominados por Yu como:

- (i) “depender” ou ator dependente, se depende que outro ator satisfaça a dependência; e
- (ii) “dependee” ou ator de quem se depende, se é o responsável por satisfazer a dependência.

Desta forma, quando o ator “dependee”,
numa relação de dependência:

- (i) disponibilizar um recurso necessário;
- (ii) executar a tarefa de sua responsabilidade;
- (iii) atender ao objetivo do ator dependente;
- (iv) realizar alguma tarefa para alcançar o objetivo-soft;

diremos que a relação de dependência foi satisfeita pelo “dependee”. Ainda nesse Modelo de Dependências Estratégicas (SD):

- (i) podem ser identificados diferentes graus de dependências:
 - (a) aberta (open), graficamente expressa por "O", quando os intentos, em caso de falha da relação de dependência, são afetados, mas sem conseqüências sérias. Por exemplo, no relacionamento entre Cliente e Atendente, se o Cliente não souber as informações sobre o CD desejado, não tem sérias conseqüências pois o Atendente pode procurar essas informações em outra fonte;
 - (b) compromissada (committed), graficamente não possui sinal associado, quando os intentos são, significativamente, afetados em caso de falha, levando-se a questionar a viabilidade da dependência. Por exemplo, no relacionamento do Caixa com Cliente, caso este último não pague pelos serviços haverá um comprometimento, pois o Cliente encomendou o CD, e não honrou seu compromisso; e
 - (c) crítica (critical), graficamente explicitada por meio do sinal "X", quando os intentos são seriamente afetados em caso de falhas,

levando-se a uma preocupação quanto a viabilidade de toda a rede de relacionamentos e dependências [YU95b]. No exemplo da Loja de Venda de CD's não existe um relacionamento crítico. Todavia consideremos por exemplo, o fato do Cliente Ter que passar para o Atendente o número de seu cartão de crédito sem ter a garantia de segurança, sob este ponto de vista a garantia na segurança do número do cartão seria um relacionamento crítico.

(ii) distinguem-se, ainda, um ator complexo, através de três diferentes tipos de sub-unidades, onde cada uma é um ator num sentido mais especializados, graficamente expressos com nomenclatura apropriada:

(a) *agente*, é um ator que possui manifestações físicas concretas, tal como um ser humano, muito embora esse termo ainda possa ser utilizado em relação a agentes artificiais (hardware, software ou uma combinação). Por exemplo, no modelo da Figura 3.2 todos os atores, a exceção da Loja, poderiam ser agentes.

(b) *papel*, caracterização abstrata do comportamento de um ator social dentro de alguns contextos especializados ou domínio de empenho, ou seja, as funções possíveis de serem exercidas por um agente dentro de uma organização. Dependências estão associadas com um papel quando essas dependências se aplicam independente de quem execute o papel. Por exemplo, os atores Atendente e Caixa, poderiam ser entendidos como papéis, dado que são tipicamente funções dentro da Loja; e

(c) *posição*, em abstração, é algo intermediário entre um agente e um papel. Trata-se de um conjunto de papéis que são tipicamente ocupados por um agente, ou seja, representa uma posição dentro da organização, onde o agente possa desempenhar várias funções (neste caso, algumas vezes a modelagem e análise podem ficar um pouco mais complexa). Diz-se que uma posição cobre um papel. Por

exemplo, os papéis de Caixa e Atendente, estariam associados a uma posição “Frente de Loja”, indicando que os funcionários que exercem aquelas funções estão tratando diretamente com o Cliente.

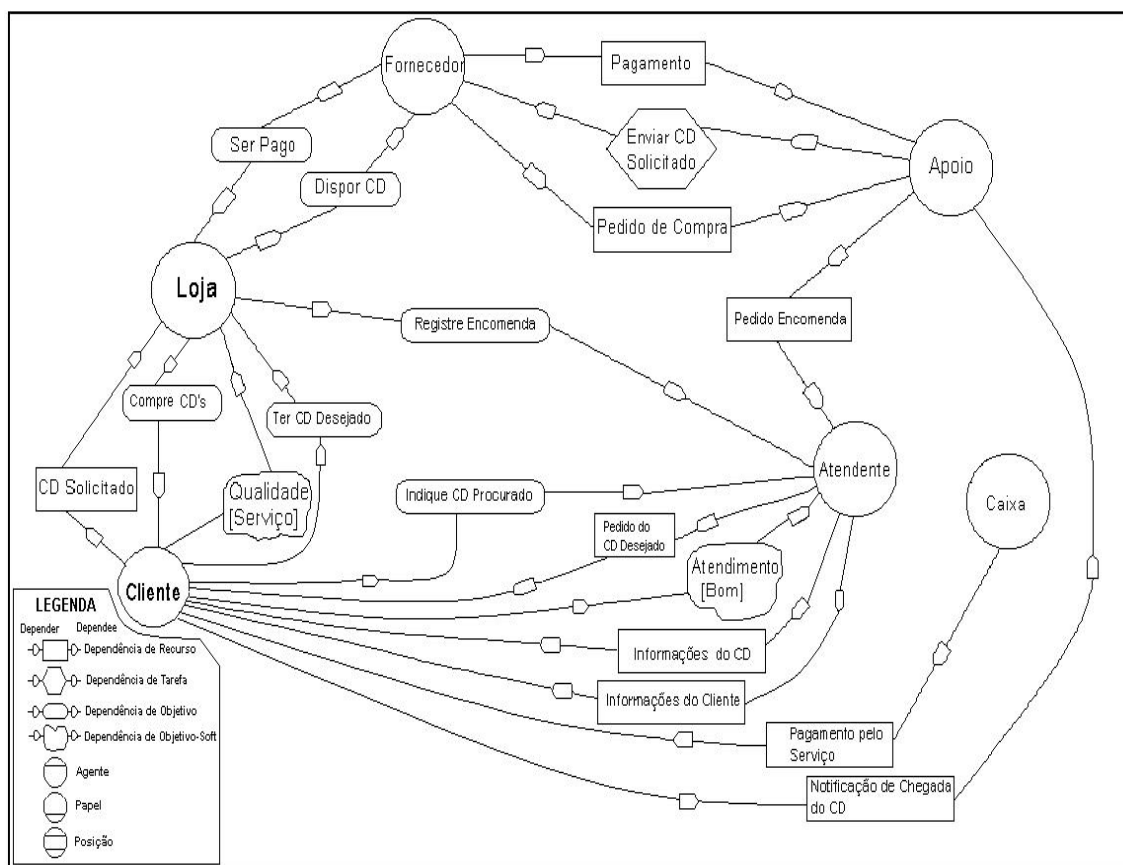


Figura3.2 – Modelo de Dependências Estratégicas (SD) da Loja de Venda de CD's.

- (iii) pode-se fazer uma análise em termos de oportunidades aberta aos agentes e vulnerabilidades que enfrentam nos relacionamentos [YU97c]. As oportunidades estão relacionadas com desejos e habilidades que possua um agente. No exemplo da Loja de venda de CD's, existem Clientes dos mais variados gostos e temperamentos, todos almejando serem bem atendidos. Da mesma forma, existem funcionários com anseios de alcançarem uma promoção, e podem alcançar ou não esse desejo no trato com os Clientes, desde que estão envolvidos num relacionamento de dependência do tipo objetivo-soft – “Atendimento[Bom]”. Assim este

relacionamento pode representar uma oportunidade de ascensão para o Atendente. Da mesma forma, no caso de falha desse relacionamento de dependência, não só um anseio pessoal de um dos atores poderá falhar, mas também parte dos relacionamentos podem ficar comprometidos, no caso, a Loja não conseguiria aumentar a clientela, por uma falha na qualidade do atendimento a seus Clientes.

O Modelo de Dependências Estratégicas (SD) oferece uma compreensão mais aprofundada sobre o relacionamento externo entre os agentes, muito mais do que informam as demais técnicas de modelagem existentes. É usado para expressar a rede de intenções, ou seja, de desejos, esperanças e compromissos existentes nos relacionamentos estratégicos entre os vários atores do processo. Estratégicos porque indicam as oportunidades existentes, possibilitando apontar as habilidades de cada ator e as possibilidades de falha que possam existir. Dado um Modelo de Dependências Estratégicas (SD), pode-se: (a) perguntar quais novos relacionamentos entre os atores são possíveis; (b) igualar os desejos de um agente com as habilidades do agente do qual dependa, para se explorar as oportunidades disponíveis a esses atores; e (c) identificar a viabilidade ou não das dependências.

Em síntese, podemos dizer que o Modelo de Dependências Estratégicas (SD) modela os processos em termos de relacionamentos externos.

3.2.1.1- Construtores da Representação

Eric Yu, em sua tese [YU95a], procurou definir mais claramente o modelo SD, usando uma linguagem de modelagem conceitual chamada TELOS [MYL90]. TELOS não é uma linguagem de especificação formal. Trata-se de uma linguagem centrada em objetos, para suportar, fundamentalmente, o desenvolvimento de sistemas de informação, baseando seus princípios na premissa de que o desenvolvimento de sistema de informação é conhecido de forma intensiva e, portanto, fundamentando-se nos conceitos da representação do conhecimento. Desta forma, TELOS é apropriada para a representação do conhecimento sob uma variedade de mundos (da aplicação, do usuário,

do sistema e do desenvolvimento) que estejam relacionados com um sistema de informação. Esta linguagem apresenta, ainda, características especiais, tais como:

- (i) uma ferramenta centrada em objetos que suporta agregação, generalização e classificação;
- (ii) um novo tratamento de atributos;
- (iii) uma representação explícita de tempo;
- (iv) facilidades para especificação de restrições de integridade e regras dedutivas; e
- (v) possuem três protótipos de implementação, todas em PROLOG.

Nesse sentido podemos dizer que foi obtida uma ferramenta de representação orientada a objeto, com classificação, generalização, agregação, atribuição e tempo. O que diferencia, a representação do *i** em TELOS, da nossa proposta, é que estamos objetivando utilizar linguagens precisas, a fim de nos utilizarmos do poder de expressão dessas, e observar inconsistências e ambigüidades, tornando a técnica *i** mais precisa.

O esquema para o Modelo de Dependências Estratégicas (SD) é definido a nível de metaclasses, em TELOS, vide Figura 3.3.

Classes Domínio são definidas como instâncias de alguma metaclasses, enquanto que estas últimas são instâncias de metametaclasses. Dessa forma, a Classe Ator é uma instância de uma metametaclasse chamada Classe. Essa facilidade de metaclasses em TELOS, permite que o esquema seja expresso, dentro da mesma ferramenta, como objetos do domínio.

Uma instância da Classe Ator pode ter como um atributo alguma instância da Classe_Depende_de ou da Classe_Dependente_de, sendo usado isto como construção básica para representação das dependências estratégicas do ator. O esquema para isso é definido pelas ligações rotuladas “depende_de” (se o ator nominado é o ator chamado por Yu como “dependeer”, Seção 3.2.1) e “dependente_de” (se o ator nominado é o ator chamado por Yu como “dependee”, Seção 3.2.1).

Os quatro tipos de dependências encontradas no modelo de dependência estratégica (SD) – recurso, objetivo, tarefa e objetivo-soft – são definidas como

Um compromisso é representado como um atributo na Classe Ator com o valor do atributo pertencendo à classe Classe_Dependum; onde “dependum” foi o nome dado por Yu aos elementos da relação de dependência (recurso, tarefa, objetivo e objetivo-soft).

3.2.1.2 - Aspectos de Precisão do Modelo de Dependências Estratégicas (SD)

Na modelagem conceitual, o propósito de se tomar uma abordagem mais formal é o de oferecer um significado mais preciso aos conceitos, de forma a ser capaz de julgar entre as diferentes interpretações de um dado modelo, assim como o de oferecer base para ferramentas de construção que ajudem com os modelos.

Os conceitos formais de objetivo, crença, habilidade e compromisso, que, normalmente, caracterizam os agentes artificiais, não são, diretamente, aplicáveis à técnica i^* , precisam ser adaptados, pois:

- i. A técnica i^* trata com relacionamento intencional entre agentes e não apenas com as propriedades intencionais de um agente isolado;
- ii. Os modelos da técnica i^* são descritivos, assumindo-se que o comportamento de um agente estratégico pode ser completamente imprevisível e aberto. Já os modelos de agentes em Inteligência Artificial são, primariamente, especificações prescritivas.
- iii. A técnica i^* tem um componente semi-formal, envolvendo conceitos de satisfação, que não estão prontamente aptos à formalização usando técnicas convencionais, pois a(s) interpretação (ões) dos conceitos de tipos são, tipicamente, dependente(s) do contexto.

Os aspectos de formalização dos conceitos do Modelo de Dependências Estratégicas (SD) são caracterizados com relação aos conceitos adotados em Inteligência Artificial, todavia não se objetiva uma integração total desses, em um sistema formal em particular. As ferramentas de suporte à técnica i^* , tal como o OME [OME3], precisam fornecer facilidades que sejam consistentes com a caracterização dada nos axiomas. A ferramenta OME começou a ser escrita por Chen Wang, como parte de sua tese de

Mestrado, tendo sido posteriormente, substancialmente modificada por Godfrey Cheng, Seyil Yoon e Mike Bissener.

Faz-se necessário destacar alguns conceitos preliminares, antes de se partir para a caracterização formal, tais como:

- (i) Rotina;
- (ii) Habilidade (adaptado de IA);
- (iii) Viabilidade; e
- (iv) Compromisso (adaptado de IA).

Os conceitos de Rotina e Viabilidade foram introduzidos para se lidar com necessidades de pensamento do ator estratégico.

As semânticas dos relacionamentos, no Modelo de Dependências Estratégicas (SD), são caracterizadas em termos de algumas supostas características internas intencionais do agente (semântica axiomática, conforme veremos no Capítulo 4) Algumas das quais são explicitamente modeladas no segundo tipo de modelo previsto na técnica i^* , Modelo de Razões Estratégicas (SR) que será visto na Seção 3.2.2.

Na estrutura interna intencional de um agente, onde todas as variáveis livres nas fórmulas são entendidas como universalmente quantificadas, inclui-se, pelo menos, os três componentes seguintes: (a) U_x – um conjunto de rotinas; (b) H_x – um conjunto de regras meio-fim; e (c) E_x – um conjunto de elementos primitivamente manipuláveis.

Passemos a observar alguns conceitos preliminares:

(i) *Rotinas*

A caracterização interna de um agente está centrada nas rotinas sustentadas pelo agente e os elementos que constituem a rotina. Uma rotina, na verdade, é o meio através do qual um agente pode alcançar o que deseja, consistindo de elementos que incluem sub-objetivos, subtarefas, recursos e objetivos soft, e de relacionamentos entre os elementos que sejam especificados como restrições, tal como precedência temporal. Um conjunto de rotinas de um agente é explicitado no Modelo de Razões Estratégicas (SR), a ser apresentado na Seção 3.2.2. No caso da loja de Venda de CD's, a rigor teríamos que nos reportar ao Modelo de

Razões Estratégicas (SR) da Figura 3.5, todavia podemos dizer que uma rotina com relação ao ator Apoio seria, “Fazer_Compra” através das sub-tarefas “Comprar”, enviando “Pedido_de_Compra” e recebendo o pedido solicitado, “Receber_Pedido”.

(ii) *Habilidade*

Quando um agente possui uma rotina que sirva como meio para que este alcance, por exemplo um objetivo, diz-se que o agente tem habilidade para alcançar aquele objetivo, sendo utilizado o predicado $H(y, \eta)$ para indicar que o agente y tem habilidade para produzir ou alcançar o elemento intencional η (recurso, tarefa, objetivo ou objetivo-soft). Assim, H_e , representa a habilidade do agente do qual se depende (dependee):

- $H_e: H(y, \eta) \equiv \exists u (U_y(u) \wedge \text{propósito}(u, \eta))$

O agente y tem a habilidade de alcançar η se somente se y tem, em seu repertório de rotinas U_y , uma rotina u cujo propósito é a realização de η . Observa-se, que em oposição ao raciocínio operacional, não é exigida qualquer garantia de que o agente y possa alcançar o objetivo através de alguma seqüência de ações. No caso da loja de venda de CD's por exemplo, o ator Apoio só poderá notificar o Cliente da chegada do CD, se o ator Apoio receber o CD solicitado ao Fornecedor.

(iii) *Viabilidade*

A fim de fornecer uma noção forte de habilidade, adequada ao raciocínio estratégico, é introduzida a noção de viabilidade, indicando que um agente acredita que alguma rotina funciona, mesmo que seja incompletamente especificada ou conhecida. Durante o raciocínio estratégico, um agente está satisfeito em reduzir uma solução a um nível no qual todos os componentes são viáveis. Neste sentido, uma rotina é julgada ser viável se cada um de seus elementos, explicitamente mencionados, é viável e se todas as suas restrições se mantêm. Diz-se que um elemento η é viável para um agente y , $V(y, \eta)$ se aquele agente tem

uma rotina viável para produzir η . No caso da loja, notificar o Cliente da chegada do CD solicitado é viável, dado que ele tem uma rotina capaz de produzir a notificação, conforme visto nos itens anteriores.

(iv) *Compromisso*

A viabilidade é uma propriedade local, ou seja, é o julgamento de um agente sobre se suas próprias rotinas seriam postas em prática ou não. Quando uma rotina envolve dependências de outros agentes, o agente dependente (chamado por Yu de “depender”), normalmente, não pode fazer julgamentos sobre a viabilidade das rotinas nos agentes dos quais se depende (chamado por Yu de “dependee”), pois este agente não conhece o bastante sobre aquelas rotinas ou a sua viabilidade dos elementos. Contudo, é importante, para os agentes, estarem aptos a acessar a viabilidade das rotinas que envolvem as dependências nos outros agentes. É tomada a visão de que, se um agente é capaz de alcançar η (recurso, tarefa, objetivo ou objetivo-soft) e está comprometido em fazê-lo, então η é viável para aquele agente. Neste sentido, o conceito de compromisso fornece uma abstração, que permite que a viabilidade seja julgada sem se ter o conhecimento sobre as rotinas usadas para alcançar η , ou mesmo a necessidade de julgar a viabilidade dos elementos que compõem aquelas rotinas. Comprometimento é, pois, a propriedade que faz a ligação entre habilidade e viabilidade.

$$\blacksquare \text{ HCV: } H(y, \eta) \wedge \exists x C(y, x, \eta) \supset V(y, \eta)$$

Se o agente y é capaz de alcançar η e y está comprometido em alcançar η para x , então, η é viável para y . Esse fato é chamado de *Assertiva Viabilidade-Compromisso*. Para o raciocínio estratégico, apenas preocupa a viabilidade $V(y, \eta)$ e não o requisito de que η seja, atualmente, alcançado. No Modelo de Dependências Estratégicas (SD), a *assertiva viabilidade-compromisso*, tipicamente aparece dentro do contexto de crença de algum agente. Tal crença pode necessitar ser justificada, se o

agente está preocupado com a viabilidade da dependência. Por exemplo, o ator Apoio, da Loja de venda de CD's, está compromissado como o Cliente, em enviar *Notificação de Chegada do CD* desejado.

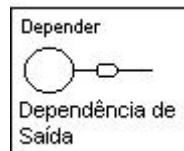
Com estes conceitos preliminares, está-se pronto para a caracterização axiomática dos conceitos de dependência. O Modelo de Dependência Estratégica (SD) fornece uma caracterização externa de um agente, em termos de dois conjuntos de dependências:

- i) Dependência de entrada – quando o agente em questão é aquele de quem se depende (“dependee”). Graficamente, tem-se



Neste caso, utiliza-se a notação de uma meia seta da direita para a esquerda ($\leftarrow$); e

- ii) Dependência de saída - quando o agente em questão é o agente dependente (“dependor”).



Neste caso, utiliza-se a notação de uma meia seta da esquerda para a direita ($\rightarrow$).

O símbolo η é utilizado para denotar um elemento objeto da dependência genérico (chamado por Yu de “dependum”, que pode ser recurso, tarefa, objetivo e objetivo-soft). Axiomas para os diferentes tipos de elementos são variações do genérico. Por exemplo, para:

- (i) a dependência de objetivos, η é uma afirmativa representando o objetivo g ;

- (ii) a dependência de tarefa, supondo uma tarefa t , diz-se que η representa o fato da tarefa ser realizada, ou seja, $\text{faz}(t)$;
- (iii) a dependência de recurso, supondo um recurso r , η é disponível (r); e
- (iv) a dependência de objetivo soft, supondo um objetivo-soft s , η é satisfeito (s).

Vimos que a dependência entre os atores pode ser de três tipos: aberta, compromissada e crítica. Nas subseções que seguem, abordaremos os operadores sobre cada um desses tipos de dependência, considerando estarem relacionados com um ator de quem se depende (“dependee”) ou ator que seja o dependente (“dependor”).

3.2.1.2.1 - Aspectos de Precisão com Relação ao Ator de quem se Depende

Consideremos inicialmente, uma *dependência aberta*, no lado daquele de quem se depende (dependee), denotado por “De”. Neste caso, se o agente y é o agente do qual outro agente depende, tendo η como objeto da dependência (“dependum”), espera-se, pelo menos, que y tenha uma rotina para alcançar η , isto é, y seja capaz de alcançar η . Por exemplo, no Modelo de Dependências Estratégicas (SD) Figura 3.2, no relacionamento entre os agentes Loja e o Cliente, tomando-se η como o objetivo “Compre CD’s”, o Cliente será o agente do qual a Loja depende para ter η alcançado. A princípio trata-se de uma relação de dependência aberta, cujo operador pode ser definido como

$$\text{De: } \overleftarrow{D}(y, \eta) \supset H(y, \eta)$$

Intuitivamente, este operador indica que o agente y , no caso o Cliente, é capaz e tem habilidade para alcançar η , no exemplo, o objetivo “Compre CD’s”. Este operador, portanto, caracterizará a dependência aberta ou não compromissada, no lado do ator de quem se depende - “dependee”, no exemplo, o Cliente. Observe que o operador $\overleftarrow{D}(y, \eta)$ é tomado com uma seta da direita para esquerda, caracterizando uma dependência de entrada, ou seja, o agente y é um agente do qual outro agente depende.

Por outro lado, quando existe um compromisso, a dependência aberta pode se tornar uma *dependência compromissada*. No caso da loja de venda de CD's, por exemplo, se o Cliente se compromete em comprar o CD fazendo o pedido deste, mas desisti da compra, então não será alcançando o objetivo da Loja, “Comprar CD's). Neste caso teremos caracterizada uma dependência compromissada. O objetivo da Loja será significativamente afetado. Podemos definir a dependência compromissada, no lado do ator de quem se depende (“dependee”), CDe, como uma conjunção de compromisso e um predicado dependente, através de $\overleftarrow{CD}(y, \eta)$.

$$\text{CDe: } \overleftarrow{CD}(y, x, \eta) \equiv \overleftarrow{D}(y, \eta) \wedge C(y, x, \eta)$$

Neste caso, uma dependência aberta $\overleftarrow{D}(y, \eta)$ é transformada em uma dependência compromissada $\overleftarrow{CD}(y, x, \eta)$, pela inclusão de um compromisso do agente y com relação ao agente x, tendo η como elemento desse compromisso, $C(y, x, \eta)$.

Os autores não apontam um operador do lado do agente “dependee” para o terceiro tipo de dependência, a crítica. No modelo de nosso exemplo, também não dispomos de uma dependência desse tipo. Assim, na próxima Seção, observaremos a força das dependências do lado do agente dependente (“dependee”).

3.2.1.2.2 - Aspectos de Precisão com Relação ao Ator Dependente

No lado do agente dependente (“dependee”), aqui denotado por “Dr”, a situação é um pouco mais complexa, fazendo-se necessário que se definam alguns conceitos intermediários, de forma a tratar separadamente os aspectos de oportunidade e vulnerabilidade.

Intuitivamente, o aspecto de oportunidade (Opor) de uma dependência de saída (o agente em questão é um agente dependente) é caracterizado como a crença do agente x de que existe algum outro agente y, que esteja se oferecendo como um ator “dependee”. Caso o agente y comprometa-se com x, então η será viável para x. No exemplo da loja de venda de CD's, se o Cliente se compromete a adquirir o CD

encomendado, então o objetivo da Loja, “Compre CD’s”, é viável. Define-se então um operador para denotar a oportunidade do agente x em alcançar η , $\overrightarrow{D}_{opor}(x, \eta)$,

$$\square \quad \text{DrOpor} \overrightarrow{D}_{opor}(x, \eta) \supset B(x, \overleftarrow{D}(y, \eta) \wedge (C(y, x, \eta) \supset V(x, \eta)))$$

Sendo B algum operador de crença, tal qual os usados em inteligência artificial, temos que a oportunidade é definida dado que existe uma crença de que a dependência compromissada do agente y com relação ao elemento η , esteja entre as vulnerabilidades do agente x, com relação a esse mesmo η . Todavia, essa caracterização não permite que η seja usada como uma dependência de saída, caso a crença acima não seja estabelecida no agente x. Assim, é adotada uma abordagem mais flexível, tratando-se $\overrightarrow{D}_{opor}(x, \eta)$ como uma propriedade primitiva intencional de x.

Isso posto, podemos considerar os diferentes graus das dependências. Para uma dependência aberta, no lado do ator dependente (“depender”), o agente dependente reconhece uma oportunidade e tem o nível aberto com relação à vulnerabilidade, $\overrightarrow{AD}_{vul}(x, \eta)$,

$$\square \quad \text{AD} \overrightarrow{D}(x, \eta) \overrightarrow{D}_{opor}(x, \eta) \overrightarrow{DA}_{vul}(x, \eta)$$

No exemplo da loja de venda de CD’s, o agente Loja reconhece que tem uma oportunidade de que seu objetivo “Compre CD’s” seja alcançado, mas ao mesmo tempo está vulnerável com respeito a esse objetivo, pois depende do agente Cliente para alcançar este objetivo.

Consideremos a noção de dependência compromissada no lado do ator dependente. Neste caso dividiremos esta em duas partes:

- i) o agente x reconhece a oportunidade que está sendo oferecida pelo agente y; e
- ii) o agente x é auto comprometido no uso de η , sendo assim vulnerável com relação a η .

Assim, pode-se definir um operador de dependência compromissada para o agente dependente, como segue

$$\text{CDr: } C\overrightarrow{D}(x, y, \eta) \equiv C\overrightarrow{D}_{\text{opor}}(x, y, \eta) \wedge C\overrightarrow{D}_{\text{vul}}(x, \eta)$$

onde $\overrightarrow{CD}_{\text{opor}}$ é similar ao operador $\overrightarrow{D}_{\text{opor}}$, só que para um agente y em particular. A primeira parte da expressão corresponde ao item (i), enquanto a segunda parte ao item (ii), acima mencionados. No caso da loja, o compromisso do Cliente em pagar pelo CD e levá-lo, para que a Loja possa ter o objetivo “Compre CD’s “ alcançado. Como existe um compromisso, existe também uma vulnerabilidade.

Já em uma dependência crítica, o ator dependente é mais vulnerável do que em uma dependência compromissada, mas, por outro lado, tem-se as mesmas características, diferindo no tocante à criticalidade da vulnerabilidade.

$$\text{XDr: } \overrightarrow{XD}(x, y, \eta) \equiv \overrightarrow{CD}(x, y, \eta) \wedge \overrightarrow{XD}_{\text{vul}}(x, \eta)$$

Uma dependência é dita crítica se existe o compromisso do agente y para com o agente x, no alcance do elemento da dependência η , mas se por outro lado o agente x é extremamente vulnerável a uma falha na obtenção de η . A vulnerabilidade baseia-se no fato de que a viabilidade de η afetaria a viabilidade da rotina, na qual η é suposta servir. No modelo do exemplo da Loja de venda de CD’s, não existe dependência crítica, pois será possível encontrar-se alternativas em caso de falha de qualquer das dependências modeladas.

Para formalmente se distinguir dentre os três graus de vulnerabilidade, oferecidos pelo Modelo de Dependências Estratégicas (SD) – aberta, compromissada e crítica - pode-se usar o conceito de rotinas, onde é assumido que o ator dependente (“dependor”) esteja usando o “dependum” η , em alguma rotina.

Uma dependência é aberta se o objeto da dependência (“dependum”) desabilita apenas parte de uma rotina, por exemplo, se o elemento da dependência não é viável, então ele torna alguma parte da rotina não viável.

$$\text{ADrVul: } A\overrightarrow{D}_{\text{vul}}(x, \eta, u) \supset \exists u'(U_x(u) \wedge \text{subrotina}(u', u) \wedge (\neg V(x, \eta) \supset \neg V(x, u'))))$$

$\overrightarrow{D}$

$\overrightarrow{D}$

- ADrVul2: $A_{vul}(x, \eta) \equiv \exists u (U_x(u) \wedge A_{vul}(x, \eta, u))$

Lembremos que U_x representa um conjunto de subrotinas, enquanto u e u' são subrotinas.

Por outro lado, dizemos que um agente x dependente, depende de η a nível compromissado, se a rotina se torna não viável, dado que η é não viável.

- CDrVul: $\vec{C}_{D_{vul}}(x, \eta, u) \supset (\neg V(x, \eta) \supset (\neg V(x, u)))$
- CDrVu2: $\vec{C}_{D_{vul}}(x, \eta) \equiv \exists u (U_x(u) \wedge \vec{C}_{D_{vul}}(x, \eta, u))$

O nível compromissado de vulnerabilidade tem um efeito desabilitador em uma rotina.

Uma dependência mais forte resulta se um elemento da dependência η puder ter um efeito desabilitador em todas as rotinas que o agente possua. Essa dependência, então, é chamada de crítica.

- XDrVul: $\vec{X}_{D_{vul}}(x, \eta) \supset \forall \eta' \forall u \forall \alpha (U_x(u) \wedge (H_x C, u, \alpha) \wedge \neg V(x, \eta)) \supset \neg V(x, \eta')$

Para todas as rotinas de x que têm η' como um fim, um elemento de dependência η , não viável, conduziria a um η' não viável, dado o conjunto de regras meio-fim H_x do agente. Em outras palavras, se o agente x não tem outra forma de fazer η' viável, toda a rede de relacionamento cai.

Tendo observado os aspectos estruturais e de precisão centrais do Modelo de Dependências Estratégicas (SD), passaremos ao segundo modelo da técnica i^* , Modelo de Razões Estratégicas (SR), na Seção que segue.

3.2.2 – O MODELO DE RAZÕES ESTRATÉGICAS (SR)

Enquanto o modelo SD trata apenas dos relacionamentos externos entre os atores, o segundo modelo que compõe a técnica i^* , o Modelo de Razões Estratégicas (SR) é usado para:

- (i) descrever os interesses, preocupações e motivações dos participantes de um processo;

- (ii) possibilitar avaliação das possíveis alternativas de definição do processo;
e,
- (iii) investigar, mais detalhadamente, as razões existentes por detrás das dependências entre os vários atores.

Ou seja, o Modelo de Razões Estratégicas (SR) permite uma compreensão mais aprofundada sobre as razões do processo.

Esse modelo, graficamente, também é composto por nós e elos que, juntos, fornecem uma estrutura para expressar as razões envolvidas nos processos. Existem:

- (i) quatro tipos de nós, que se baseiam nos tipos de dependências do modelo dependências estratégicas (SD): *objetivo* (goal), *tarefa* (task), *recurso* (resource) e *objetivos-soft* (soft-goal); e
- (ii) dois tipos principais de relacionamentos que interconectam os nós:
 - (a) *meio-fim* (means-ends); e
 - (b) *decomposição de tarefas* (task-decomposition).

Os *relacionamentos meio-fim*, indicam um relacionamento entre um fim – que pode ser um objetivo a ser alcançado, uma tarefa a ser realizada, um recurso a ser produzido, ou um objetivo soft a ser satisfeito – e um meio, para se atender a esse fim, ou seja, sugerem que podem existir outros meios de alcançar o mesmo fim, exprimindo as alternativas existentes.

Os *relacionamentos de decomposição de tarefa* exprimem o que deve ser feito para se ter a tarefa realizada e podem ser de quatro tipos: (a) *sub-objetivo*, (b) *sub-tarefa*; (c) *recursoPara*; ou (d) *objetivo-softPara*. Esses tipos de relacionamento podem interligar-se aos relacionamentos de dependência do modelo de dependência estratégica (SD), quando as razões ultrapassam os limites do ator. Nas Figuras 3.4a e 3.4b podemos, resumidamente, identificar os principais tipos de ligações existentes no modelo de razões estratégias (SR).

Na Figura 3.4a temos os diversos tipos de decomposição de uma tarefa, onde podemos ver que esta pode ser decomposta em: um sub-objetivo; em uma subtarefa; em

um recurso necessário a execução da tarefa; ou em um objetivo-soft, que também representa um refinamento da tarefa.

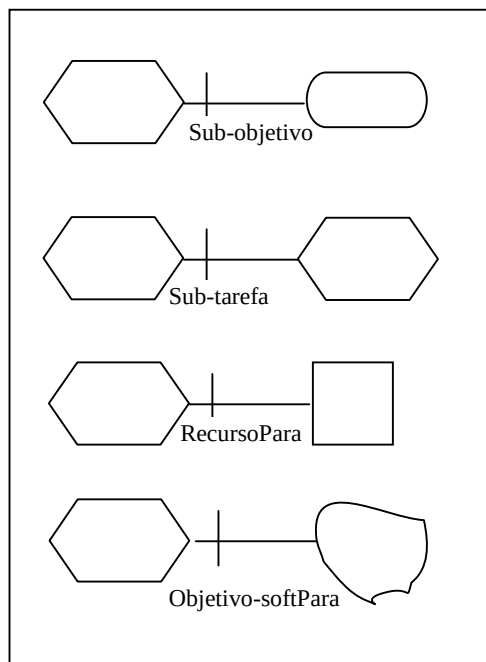


Figura 3.4a - Ligações de Decomposição de Tarefas

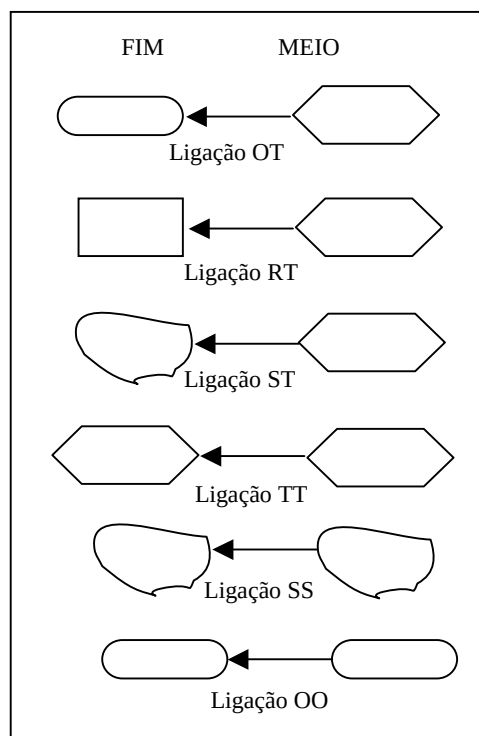


Figura 3.4b – Ligações Meio-Fim

Na Figura 3.4b temos os diversos tipos de relacionamento meio-fim, onde, em geral, o meio de se conseguir um fim é uma tarefa e os fins podem ser: um objetivo, um recurso, um objetivo-soft ou uma tarefa. Todavia existem dois outros tipos: (a) o meio e fim são objetivos; e (b) o meio e o fim são objetivos-soft.

A partir do Modelo de Dependências Estratégicas (SD) é feito um detalhamento de alguns dos atores envolvidos. No caso do estudo de caso (vide Figura 3.5), os atores Loja, Apoio e Atendente são descritos com mais detalhes através do Modelo de Razões Estratégicas (SR) para a Loja de Venda de CD's. Neste modelo, podemos identificar no comportamento interno dos atores detalhados, ligações do tipo decomposição de tarefa (Figuras 3.4a) e ligações do tipo meio-fim (Figura 3.4b), por exemplo:

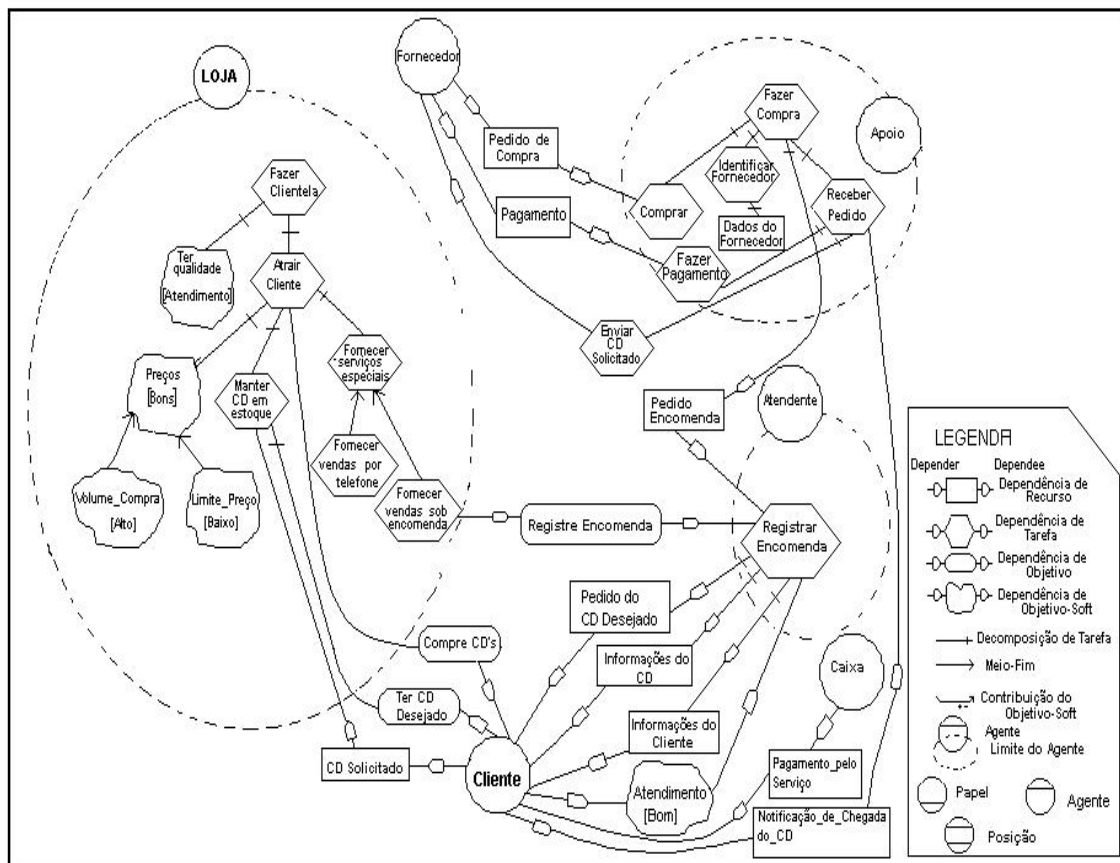


Figura 3.5 – Modelo de Razões Estratégicas (SR) da Loja de Venda de CD's

- (i) No agente *Apoio*, temos uma decomposição de tarefa - a tarefa “*Fazer Compra*” decomposta em três subtarefas: *Comprar*, *Receber Pedido* e *Identificar Fornecedor*. Esta última tarefa é ainda decomposta, em um recurso *Dados do Fornecedor*; e
- (ii) No agente *Loja*, teremos um relacionamento meio-fim , ou seja, existem dois meios de se alcançar a tarefa *Fornecer Serviços Especiais*. Uma opção é *Fornecer Vendas por Telefone* e outra opção é *Fornecer Vendas por Encomenda*.

No Modelo de Razões Estratégicas (SR), quando da existência de um relacionamento de decomposição de tarefa, uma sub-tarefa restringe a tarefa original a um curso particular de uma ação. No caso de um objetivo-soft ser um componente de uma tarefa, serve como um objetivo de qualidade para esta tarefa, guiando ou restringindo a seleção entre alternativas em adicionais decomposições desta tarefa. Por exemplo no modelo da Figura 3.5, temos que no ator Loja, a subtarefa Atrair Cliente restringe a tarefa original que é Fazer Clientela, e ainda o objetivo-soft Ter_Qualidade[Atendimento] restringe essa tarefa, guiando-a para o fato de que não pode ser feita aleatoriamente.

Em especial, nesse Modelo de Razões Estratégicas (SR), as ligações que envolvem objetivos soft requerem um atributo extra para indicar o tipo de contribuição relacionada com esses objetivos:

- (i) positiva (+), como o próprio nome sugere se o objetivo-soft influenciará positivamente;
- (ii) negativa (-), como o próprio nome sugere se o objetivo-soft influenciará negativamente;
- (iii) bastante (sup), como o próprio nome sugere se o objetivo-soft influenciará de forma suficiente;
- (iv) não bastante (sub), como o próprio nome sugere se o objetivo-soft influenciará de forma não suficiente; ou
- (v) desconhecida ou indefinida (?), caso onde não se sabe qual será a influência do objetivo-soft em questão.

No exemplo da Loja de Vendas de CD's, não nos interessou explicitar as contribuições relativas aos objetivos-soft identificados.

No relacionamento de decomposição de tarefas, tem-se uma descrição hierárquica dos elementos intencionais, tal qual um refinamento, onde cada elemento é importante para o sucesso da tarefa. Por exemplo, no ator Loja, Fazer Clientela significa *Atrair Cliente*, *Fornecer Serviços Especiais*, donde podemos chegar a *Vendas_sob_Encomenda*. Assim para *Fazer Clientela* é preciso vender sob encomenda.

Na verdade, o Modelo de Razões Estratégicas (SR) fornece um rico conjunto de conceitos para a modelagem de processos e as razões por detrás desses processos, bem como a possibilidade de compreensão mais aprofundada, quando comparada com as demais formas de modelagem existentes. Todavia, possíveis restrições temporais não aparecem graficamente neste modelo.

Os modelos convencionais vêem um processo como atividades com fluxos entre estas, não capturando a dimensão do “porquê” de um processo. Através do relacionamento meio-fim, o Modelo de Razões Estratégicas (SR) fornece uma visão de processo orientada a objetivo, ou seja, em qualquer nó, quando se faz uma questionamento do “como”, um meio é explorado, com o nó atual sendo seu fim desejado. Da mesma forma, quando um questionamento do “porquê” é realizado, explora-se descobrir o fim para que o nó atual seja o meio. Por exemplo, *Fornecer Vendas sob Encomenda* é um dos meio para se ter um fim alcançado, *Fornecer Serviços Especiais*.

Conceitos, tais como, *rotina*, *regras*, *crenças* aparecem explicitamente no Modelo de Razões Estratégicas (SR). Assim sendo, consideraremos cada um destes três conceitos em separado.

Para se referir a um conjunto particular de escolhas, no Modelo SR, é introduzida a noção de *rotina*. Assim, uma rotina é um sub-gráfico do Modelo SR, com uma única ligação a um nó fim, a partir de cada nó meio. Por exemplo, na Figura 3.5, com relação ao ator Loja podemos identificar uma rotina que envolve as tarefas:

- (a) “Fornecer Vendas por Encomenda” que representa um meio;
- (b) “Fornecer_Serviços_Especiais” que representa um fim ; e
- (c) “Atrair Cliente” que é a uma sub-tarefa que também está sendo decomposta.

Desta forma, representa um curso particular de ação entre múltiplas alternativas apresentadas em cada nó do tipo OU (OR). A noção de rotina é usada para se referir a um processo e suas razões. Rotinas, tipicamente, têm conexões com outros atores, através das ligações de dependências no Modelo de Dependências Estratégicas (SD). O termo rotina é usado para cobrir a natureza tipicamente recorrente de um processo de

trabalho. Descrevendo uma rotina em termos de elementos intencionais, o Modelo de Razões Estratégicas (SR) reconhece que a solução adicional do problema, normalmente, acontece quando uma rotina é executada. Isso contrasta com a noção mecanicista de procedimento ou o conceito de plano, como uma série de ações primitivas na clássica Inteligência Artificial [YU95].

As ligações meio-fim, no Modelo de Razões Estratégicas (SR), são vistas como inseridas em um contexto particular: são raciocínios. Todavia, essas ligações podem ser vistas como aplicações de relacionamentos mais genéricos, que dizem que sempre que se tenha algum elemento como fim, pode-se usar algum outro elemento como um meio para aquele fim. Por exemplo, para se atingir o fim “Fornecer Serviços Especiais” utilizamos “Fornecer Vendas sob Encomenda” ou “Fornecer Vendas por Telefone” como meios de se alcançar aquele fim. Uma regra é, pois, uma ligação meio-fim que não foi ainda limitada, consistindo de:

- (i) uma condição de aplicabilidade;
- (ii) um meio; e
- (iii) um fim.

Por exemplo, “Fornecer Vendas sob Encomenda a Clientes VIPs” representa uma regra do tipo Tarefa-Tarefa (Regra TT) que é aplicada à ligação meio-fim “Fornecer Serviços Especiais” e “Fornecer Vendas sob Encomenda”, onde a condição de aplicabilidade é “Identificar Clientes VIPs” que supomos que a Loja possua; o meio é “Fornecer Vendas sob Encomenda” e o fim é fornecer “Fornecer Serviços Especiais”.

Elementos do raciocínio, tais como, as condições de aplicabilidade em regras, são informações suplementares que não são diretamente parte da hierarquia meio-fim. Oferecem evidências de suporte a escolha de rotinas ou evidências contra as possíveis alternativas de rotinas. Essas são modeladas como crenças, constituindo uma rede suplementar de nós e ligações, em um estilo argumentativo de raciocínio.

3.2.2.1 - Construtores da Representação

Na Figura 3.6 tem-se um esquema parcial para o Modelo de Razões Estratégicas (SR) em TELOS [MYL90], onde são mostradas as ligações adicionais a esse modelo: ligações meio-fim e ligações de decomposição de tarefa.

A Seção central da Figura 3.6 trata com as ligações de decomposição de tarefa (vide Figura 3.4a) e suas correspondentes ligações de dependência. Como visto na Seção anterior, uma tarefa pode ser decomposta em sub-objetivos, subtarefas, recursos e objetivos soft. Cada uma dessas dependências tem uma ligação de dependência oposta. Quanto as ligações meio-fim e as regras, por questão de espaço apenas os casos de ligações objetivo-tarefa (GT) e recurso-tarefa (RT) são mostrados.

O ator de quem se depende (“dependee”) é designado como um atributo da ligação, a partir da tarefa até seu elemento objeto da dependência (“dependum”).

Uma dependência de entrada é uma ligação de um ator a um “dependum”, com o ator dependente (“dependor”) representado como um atributo da ligação.

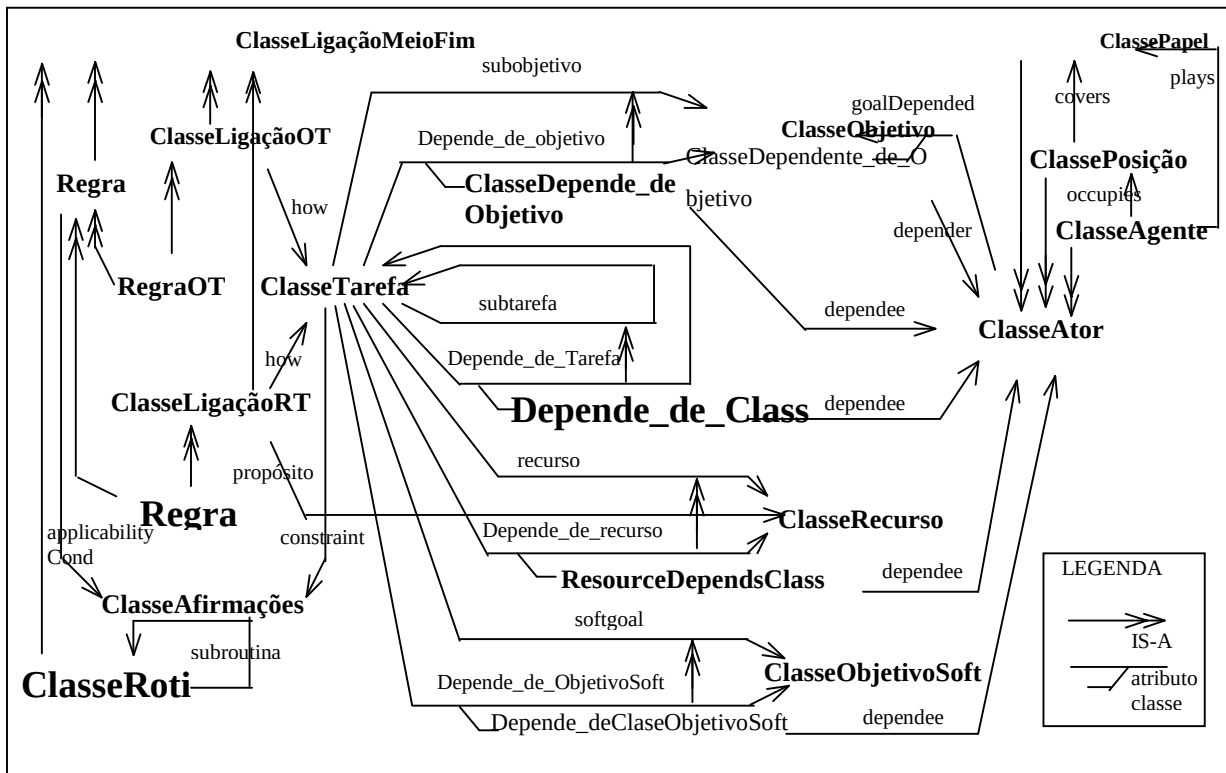


Figura 3.6 – Esquema Parcial para o Modelo de Razões Estratégicas (SR) em TELOS

A Seção da esquerda da Figura 3.6 mostra os relacionamentos entre ligações meio-fim, regras e rotinas. As ligações meio-fim têm um propósito e um “como”. Cada tipo de ligação meio-fim é uma especialização disso, com tipos diferentes de elemento intencional como propósito e como. Na Figura 3.6, só são mostrados os tipo de ligação meio-fim Objetivo-Tarefa (ClasseLigaçãoOT) e recurso-tarefa (ClasseLigaçãoRT)

Para se fazer uma análise, é necessário ter-se acesso à coleção de rotinas, regras e elementos, primitivamente viáveis, que um ator possua. Assim, o construtor de ator, no modelo de raciocínios estratégicos (SR), necessita ter atributos adicionais, além daqueles existentes no Modelo de Dependências Estratégicas (SD). Esses são definidos na metaclasses ClasseAtorSR.

3.2.2.2 - Aspectos de Precisão do Modelo de Razões Estratégicas (SR)

Os axiomas são projetados para fornecer uma caracterização mais formal aos conceitos do que a representação informal e para servir de base para a construção de ferramentas de suporte à técnica.

Assim como no Modelo de Dependências Estratégicas (SD) (vide Seção 3.2.1.2.2) usamos a variável η para denotar um elemento intencional, sendo necessário fazer uma distinção entre os diferentes tipos de elementos intencionais. Para isso são utilizados o , t , r e s para, respectivamente, denotarem objetivo, tarefa, recurso e objetivo soft.

Uma ligação de decomposição de tarefa no Modelo de Razões Estratégicas (SR) é denotada pelo predicado $dt(\eta, t)$, onde se tem o elemento η associado à tarefa t . Desta forma, os diferentes tipos de ligações de decomposição de tarefas mencionados na Seção precedente são denotados como:

- (a) $subobjetivo(t,o)$ indicando que a tarefa t é decomposta no elemento objetivo o ;
- (b) $subtarefa(t, t')$ indicando que a tarefa t é decomposta no elemento tarefa t' ;
- (c) $recursoPara(t,r)$ indicando que a tarefa t é decomposta no elemento recurso r ;
- e

- (d) objetivo-softPara(t,s) indicando que a tarefa t é decomposta no elemento objetivo-soft s.

O predicado $rx(t, k)$ indica que k é uma restrição que aplica-se à tarefa t. Enquanto os predicados $odt(x, \eta)$ e $cdt(x, \eta)$ são usados para indicar, respectivamente, ligações de decomposição de tarefa aberta e compromissada.

O segundo tipo de ligação existente no Modelo de Razões Estratégicas (SR) é a ligação meio-fim. Esta ligação meio-fim é representada pelo predicado $lmf(l, \eta, u)$ onde l representa a ligação, o elemento intencional η é o fim e a rotina u é o meio para se chegar ao elemento intencional.

Na Seção 3.3.1.2.2, vimos alguns conceitos preliminares, por exemplo que o repertório de rotinas que um ator x possui é denotado por U_x ; assim, uma rotina u que esteja no repertório de x é escrita como $U_x(u)$. O conjunto de elementos do ator x, primitivamente viáveis é representado por E_x . As regras meio-fim de um ator x são denotadas por $H_x(\eta, u, \alpha)$, onde η representa o propósito (o fim), u representa o “como” (o meio) e α é a condição de aplicabilidade.

O conjunto de crenças primitivas do ator x, isto é, das crenças que não precisam ter justificativas adicionais, é representado por B_x . As regras de redução-crença, usadas para justificar as crenças de um ator x são denotadas por $I_x(b, b', \alpha)$, onde b' é a crença de suporte para a crença b e α é a condição de aplicabilidade.

Uma subrotina também é uma rotina, assim o propósito de uma sub-rotina deve estar de acordo com o “como” da rotina, embora possa haver ligações decomposição de tarefas entre essas.

$$\begin{aligned} \blacksquare \quad & (\text{subrotina}(u', u) \wedge \text{propósito}(u', \eta)) \supset \\ & (\text{como}(u, \eta) \vee \exists t (\text{como}(u, t) \wedge \text{subel}(\eta, t))) \end{aligned}$$

onde $\text{subel}(\eta, t)$ significa que η é um elemento em alguma decomposição ou sub-decomposição da tarefa t.

$$\blacksquare \quad \text{subel}(\eta, t) \subset (\text{el}(\eta, t) \vee \exists t' (\text{el}(t', t) \wedge \text{subel}(\eta, t')))$$

Utilizando-se o Modelo de Razões Estratégicas (SR) num processo de análise, faz-se necessário se alguns conceitos e descrevê-los por meio de axiomas, como segue:

1. *Habilidade de um Agente*

Diz-se que um agente x tem habilidade para alcançar o elemento η se somente se este possui uma rotina para fazer isso. Ou seja,

$$\text{Habe:} \quad \text{Hab}(x, \eta) \equiv \exists u (U_x(u) \wedge \text{propósito}(u, \eta))$$

Por exemplo, no caso da loja de venda de CD's o agente Loja tem habilidade para alcançar "Atrair Cliente", dado que possui uma rotina para alcançar este propósito ("Fornecer Vendas sob Encomenda", "Fornecer Serviços Especiais" e "Atrair Cliente").

2. *Viabilidade da Tarefa*

Uma tarefa é dita ser viável se todos os seus elementos são viáveis e se todas as suas restrições acredita-se que se mantenham. Ou seja,

$$\text{Vt:} \quad V(x, t) \subset \forall \eta (dt(\eta, t) \supset V(x, \eta)) \wedge \forall k (rx(k, t) \supset B(x, k))$$

No caso da loja, a tarefa "Atrair Cliente" é viável se todos os seus elementos ("Preços[Bons]", "Manter CD em Estoque" e "Fornecer Serviços Especiais") são individualmente elementos viáveis, desde que também se mantenham suas possíveis restrições.

3. *Viabilidade de um Elemento*

Consideremos a viabilidade para cada elemento individualmente. O critério geral para que um elemento seja viável depende se ele é um elemento aberto ou comprometido com relação à tarefa. Ou seja,

$$\text{Ve:} \quad V(x, \eta) \subset (odt(\eta) \wedge V_o(x, \eta)) \vee (cdt(\eta) \wedge V_c(x, \eta))$$

Assim, a depender de ser um elemento aberto ou compromissado, teremos dois axiomas:

(a) Um elemento aberto η é viável se η é uma dependência aberta ou se ele é viável sob o critério de um elemento compromissado, ou seja,

$$\text{▪ Veo: } V_o(x, \eta) \subset (\overrightarrow{D}_{opo}(x, \eta) \vee V_c(x, \eta))$$

Intuitivamente, para um elemento aberto ou x sabe como alcançá-lo ou x conhece alguém que possa alcançá-lo.

(b) Um elemento compromissado η é viável se η é primitivamente viável; ou se existe alguma ligação meio-fim viável associando η a uma rotina viável ou se η é uma dependência de saída e existe um outro agente y comprometido em produzir η para x .

$$\begin{aligned} \text{▪ Vec: } V_c(x, \eta) \subset E_x(\eta) \\ \vee \exists l \exists u (lmf(l, \eta, u) \wedge V(x, l) \wedge U_x(u) \wedge V(x, u)) \\ \vee (\overrightarrow{D}_{opp}(x, \eta) \wedge \exists y B(x, C \overleftarrow{D}(y, x, \eta))) \end{aligned}$$

Intuitivamente, para um elemento compromissado η , ou x sabe como alcançá-lo; ou x deve obter um comprometimento de alguém que possa alcançá-lo; ou x deve reduzir η através de uma rotina até este seja viável.

4. Viabilidade de uma Rotina

Uma rotina u é viável se todos os elementos especificados nela, como seus atributos, é individualmente viável e todas as subrotinas são viáveis.

$$\begin{aligned} \text{▪ Vu: } V(x, u) \subset (U_x(u) \wedge \forall \eta' (\text{como}(u, \eta') \supset V(x, \eta')) \\ \wedge \forall u' (\text{subrotina}(u', u) \supset V(x, u'))) \end{aligned}$$

Desta forma a viabilidade pode ser avaliada recursivamente.

5. Viabilidade de uma Ligação Meio-Fim

Uma ligação meio-fim l , com u como o meio e η como o fim, é viável se o agente possui uma regra para os relacionamentos meio-fim e o agente acredita que a condição de aplicabilidade α para esta regra se mantenha.

$$\text{VI: } V(x, l) \subset \exists \eta \exists u (U_x(u) \wedge \text{lmf}(l, \eta, u) \supset \exists \alpha (H_x(\eta, u, \alpha) \wedge B(x, \alpha)))$$

Desta forma, nesta Seção introduzimos axiomas que tratam principalmente com as duas principais ligações encontradas no Modelo de Razões Estratégicas (SR): a ligação de decomposição de tarefa e a ligação meio-fim.

CONSIDERAÇÕES

Neste Capítulo, foram observadas as principais técnicas para a modelagem dos requisitos organizacionais sob o ponto de vista da ciência da computação e da administração.

Com base na literatura pesquisada, podemos dizer que poucas técnicas tradicionais, na Ciência da Computação, tratam sobre os requisitos organizacionais (estudo de viabilidade e análise de restrições). Já na área de Administração, as técnicas não são sofisticadas, do ponto de vista de serem graficamente pobres e sem uma definição precisa.

Todavia, temos que considerar que, a tentativa de se compreender o contexto organizacional e as razões que levam às várias alternativas, podem ser importantes para o sucesso do sistema. A maioria dos modelos existentes fornecem uma descrição dos *quês*, mas não dos *porquês* em um processo, sendo mal equipados para ajudar a responder algumas perguntas que objetivam atingir estes *porquês*. E responder os *porquês* é mais importante, não apenas por ajudar, em primeira instância, no desenvolvimento, com sucesso, do sistema, mas também por facilitar o desenvolvimento de cooperação com outros sistemas, bem como a evolução dos mesmos.

Hoje, vê-se um grande número de trabalhos na área da Engenharia de Requisitos, que se direcionam para a necessidade de modelos conceituais capazes de lidar com

agentes, objetivos e intenções [DAR93, LOU95, MYL95, YU97c, MYL99], de forma a retratar, com mais precisão, as necessidades do mundo real, bem como há outros trabalhos que buscam tornar as abordagens cada vez mais precisas [YU97a, LEV98, DARI97, DUB98]. Segundo esse aspecto, disse [GUS95] que a definição de Engenharia de Requisitos evoluiu, antes, tratada como um ramo da Engenharia de Software, passou a ter maior destaque, sobretudo por se preocupar com a comunicação com atores organizacionais. Assim, a modelagem organizacional e a classificação dos requisitos em funcional, não funcional e organizacional, passam a ser importantes por várias razões:

- (i) levar à incorporação dos requisitos de cada pessoa envolvida com o sistema, considerando os objetivos e metas do ambiente empresarial;
- (ii) justificar e explicar a presença dos requisitos de cada pessoa envolvida com o sistema;
- (iii) poder ser utilizada para a designação de responsabilidades a agentes e de compromissos dos requisitos dos participantes, com respeito ao sistema desejado; e
- (iv) fornecer informação básica para a detecção e resolução de conflitos que possam surgir, a partir dos múltiplos pontos de vistas envolvidos.

Do ponto de vista da administração [TEI99], “a pura experiência, se não for acompanhada por modelos de raciocínio que permitam uma reflexão consciente, não irá além da repetição sistemática de fórmulas que um dia, eventualmente, foram bem sucedidas. Contudo, nada garante que a experiência anterior se aplique ao que vem pela frente”. Neste sentido é que apontamos algumas das técnicas utilizadas para a modelagem da organização, sob a ótica da Administração. Vimos que os modelos de fluxo de trabalho e suas variações são os basicamente mais utilizados para a modelagem organizacional, sob essa visão. Não obstante, apresentam apenas as entidades e atividades envolvidas num processo de trabalho, mas não evidenciam as razões para as suas existências ou relacionamentos, não abrangendo tipos de soluções para problemas que possam estar envolvidos na execução do trabalho.

Do ponto de vista da Engenharia de Requisitos, avaliamos o poder de expressão de algumas técnicas de modelagem organizacional. Uma avaliação mais criteriosa foi

realizada recentemente entre as técnicas i^* e Bubenko [ALE99b]. Nesta avaliação, alguns pontos característicos foram levados em consideração:

- (i) o elemento principal;
- (ii) os objetivos;
- (iii) estabelecimento de domínios;
- (iv) modelos propostos;
- (v) entendimento dos modelos;
- (vi) ferramental gráfico;
- (vii) tratamento de vulnerabilidades;
- (viii) termos utilizados;
- (ix) dicionário de conceitos; e
- (x) expressividade.

A conclusão obtida foi a de que as técnicas i^* e Bubenko são similares.

Com isso, podemos afirmar que, em linhas gerais, para o objetivo deste trabalho, o uso de qualquer uma das técnicas de modelagem organizacional seria adequado; evidentemente, que em função das peculiaridades de cada uma, pequenas adaptações poderiam ser necessárias.

Nem todos os sistemas exigem uma especificação de requisitos precisa, todavia existem certos sistemas, tais como, sistemas críticos, sistemas distribuídos e sistemas de segurança, nos quais é fundamental a utilização de técnicas de especificação rigorosa. Assim sendo, buscamos apresentar os aspectos formais encontrados na técnica i^* , todavia destacando que, segundo seus próprios autores afirmaram, em comunicações privadas conosco estabelecidas, que propositadamente deixaram em aberto a questão da semântica, por não desejarem restringirem-se a uma única interpretação, tendo como proposta ser uma técnica de mais alto nível e quando necessário, reportam-se a sua definição em termos de TELOS.

No próximo Capítulo, trataremos especificamente sobre os conceitos, métodos e técnicas para descrição precisa de requisitos existentes. Objetivamos discorrer sobre a necessidade da precisão e eleger algumas linguagens para que seja alcançado o objetivo

do nosso trabalho, ou seja, a integração dos requisitos organizacionais às especificações precisas de requisitos.

CAPÍTULO IV

ESPECIFICAÇÃO PRECISA DE REQUISITOS

Os métodos formais, suportados por várias ferramentas, permitem que os engenheiros possam capturar e especificar requisitos de forma mais cuidadosa e precisa, reduzindo o custo de desenvolvimento e melhorando a qualidade dos softwares, de forma substancial. São vários os trabalhos de pesquisa que se esforçam em desenvolver linguagens de especificação precisa. Todavia, em sua maioria, têm como foco principal os requisitos funcionais. Aspectos não funcionais e organizacionais, em geral, não recebem tratamento preciso. Hoje tenta-se oferecer suporte formal às técnicas gráficas de modelagem de requisitos (documentação dos requisitos), uma vez que as descrições contidas nesses métodos são, normalmente, ambíguas e imprecisas, permitindo diferentes interpretações para uma mesma especificação. Importantes informações podem estar escondidas entre detalhes considerados irrelevantes, podendo ocorrer que falhas de requisitos só sejam descobertas muito mais tarde, quando da implementação dos sistemas, tornando suas correções muito caras ou mesmo impossíveis. Como solução, desde algum tempo, existe uma forte tendência na utilização de métodos formais [BOW94a, BOW94b, JAC98], baseados em matemática, na busca da produção de documentações precisas e não ambíguas.

Neste Capítulo abordaremos conceitos fundamentais da especificação formal e precisa, bem como algumas linguagens de especificação formal, detendo-nos naquelas que serão utilizadas na integração com a modelagem organizacional.

4.1- PRINCIPAIS CONCEITOS

Fato indiscutível, até então, é que os sistemas reais devem ser modelados, a fim de que suas implementações possam ser realizadas através de sistemas de computação. Uma *especificação* nada mais é do que um modelo de um sistema, em um determinado nível de abstração, de forma a ser possível descrever não só o comportamento do sistema, mas também os tipos de dados, caso seja necessário.

Os métodos de especificação formal, segundo [SOM97], diferem dos métodos de análise de requisitos. Isso porque esses últimos são informações não precisas, resultantes da fase de elicitação, que irão ser analisadas para averiguação de inconsistências e incompletudes, considerando o que se espera do sistema. Trata-se de uma checagem sob os requisitos capturados. Essas informações ajudarão o analista a entender o sistema e seu contexto dentro do ambiente. Já os métodos formais modelam matematicamente o comportamento do sistema, quase sempre o comportamento funcional, procurando eliminar inconsistências e ambigüidades, através das provas de propriedades e animações. Neste caso, métodos formais ajudam a precisar os métodos informais e semi-formais, garantindo especificações mais corretas e seguras.

Os métodos formais permitem que o engenheiro de software especifique, desenvolva e verifique o sistema a ser desenvolvido, através da aplicação de uma rigorosa notação matemática, possibilitando assegurar consistência, completude e correte. Quando os métodos formais são usados durante o desenvolvimento permitem descobrir e corrigir, mais fácil e rapidamente, os problemas existentes.

A especificação formal de um objeto está profundamente relacionada com o conceito de abstração. Uma especificação deve explicitar, de forma exata e não ambígua, todas as condições a que um objeto deve satisfazer, expressando o essencial e omitindo o não essencial, relativos às decisões futuras de implementação.

O escopo limitado das linguagens de especificação formal resulta do restrito conjunto de abstrações incorporado, considerando-se quais requisitos deverão ser capturados. A obtenção de uma especificação que represente todos os aspectos de um sistema, principalmente quanto aos aspectos sócio-técnicos da empresa (requisitos organizacionais), é muito difícil, devido à complexidade das atividades realizadas. Muitos dos aspectos organizacionais não são especificados, por dificuldades de representação através dos métodos de modelagem semi-precisos tradicionais e, também, através dos métodos precisos.

Segundo [EDE94], os componentes essenciais em um método de especificação são: (i) os *modelos*, com seus conjuntos de conceitos e regras quanto à utilização do método e tipo de abstração; (ii) as *linguagens*, descrições formais segundo os modelos; (iii) a *seqüência de aplicação do método*; e (iv) as *ferramentas* disponíveis que possam

auxiliar na especificação, documentação, avaliação, tradução e simulação da especificação.

No desenvolvimento de software, tem-se como objetivo fundamental, obter-se um programa escrito em uma determinada linguagem que satisfaça aos requisitos inicialmente, propostos [SAN93]. O que se procura fazer é centrar a atenção em métodos de desenvolvimento de programas a partir de especificações, de forma que se possa garantir que o programa resultante esteja correto por construção. Para isso, trabalha-se com a idéia de refinamento passo a passo. A cada passo é capturada uma decisão de projeto e provada sua corretude, conforme pode ser visto na Figura 4.1.

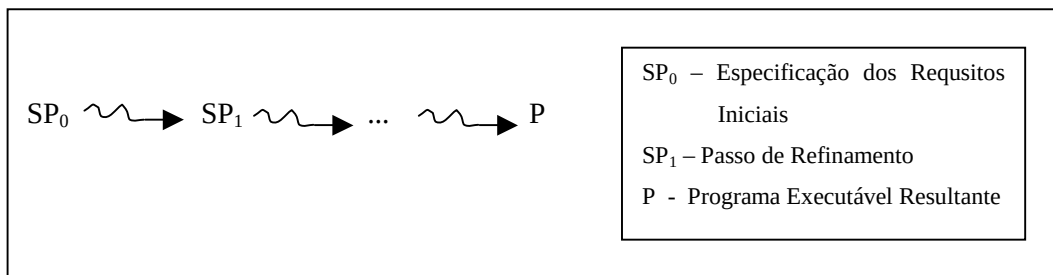


Figura 4.1 – Abordagem para o Refinamento da Especificação até o Programa Executável

O modelo de Transformação Formal, proposto por Balzer em 1983 [KOT98], objetiva a criação de um objeto nos primeiros estágios do ciclo de vida do desenvolvimento, que possa ser usado como meio de compreensão e validação dos requisitos dos usuários. Esse modelo de desenvolvimento requer, como ponto de partida, uma especificação formal do sistema (definição matemática, não ambígua). Este modelo baseia-se na transformação da especificação formal encontrada em um programa, através de métodos matemáticos, contrariando, por outro lado, o princípio de se ter especificações que sejam de fácil compreensão por parte dos clientes/usuários. Tais transformações devem preservar a corretude, ou seja, o programa deve satisfazer sua especificação. Isso é feito através de vários passos de refinamento. Cada passo de refinamento transforma especificações abstratas em outras especificações mais concretas, através de regras matemáticas pré-definidas. Teoricamente, a cada passo de refinamento, deve-se provar que as novas especificações mantêm todas as propriedades das anteriores. Dessa forma, fica garantido que o sistema final estará correto e que o

único objeto a ser validado será a especificação. Aplicações sucessivas de passos de refinamento são realizadas, encerrando-se quando a especificação gerada for um programa executável.

A motivação desse modelo está na possibilidade de se gerar, automaticamente, programas que sejam corretos por construção. Isso porque o próprio processo de desenvolvimento garantiria que o programa faz, exatamente, o que foi especificado. Todavia, o uso de refinamento ainda não é muito adequado para o desenvolvimento de sistemas de grande porte, pois as especificações, em cada passo, tendem a se tornar maiores e de prova mais difícil. O que se costuma fazer é a decomposição das especificações em unidades menores que serão refinadas, de forma independente, durante o processo de desenvolvimento, sem contudo ser removida a possibilidade de insensatas decisões de projeto.

O processo de refinamento, segundo [COR94], envolve refinamento de dados e refinamento de operações. O refinamento dos dados preocupa-se com as transformações de tipos, tais como conjuntos, relações e mapeamentos em tipos mais concretos, por exemplo: arrays e registros (possíveis de serem implementados). No segundo tipo de refinamento, as operações nos tipos abstratos são refinadas em estruturas de controle, que atuam no correspondente tipo concreto.

Poucos métodos formais são métodos, no sentido de propiciarem formas organizadas para melhorar o desenvolvimento, tais como notações e diretrizes de processo de aplicação sistemática. Segundo [SOM97], alguns são simplesmente notações.

O que se objetiva com as técnicas formais é clareza, não ambigüidade, independência de implementação e completude da especificação. Por meio de um sistema de inferência é possível deduzir propriedades do sistema e chegar-se a especificações executáveis.

A especificação formal, constitui-se numa coleção de diferentes técnicas, mas tendo em comum o uso da matemática para especificar o comportamento dos sistemas. Os custos e benefícios de uma especificação formal variam segundo o modo de utilizá-la; por exemplo, o modo mais simples e uso mais completo é o de utilizá-la como meio para a documentação e comunicação dos requisitos, com o intuito de reduzir

ambigüidades. Todavia, assim como existem na Engenharia de Requisitos alguns mitos, na área dos métodos formais eles também existem [BOW94a, BOW95], como por exemplo:

1. *Métodos formais podem garantir que o software seja perfeito*, eliminando a necessidade de teste. Isso não só é um absurdo, mas pode trazer sérias consequências se tomado a sério;
2. *Métodos formais descrevem apenas provas de corretude de programas*. Não é verdade, pois existem outras possibilidades, tais como, animação de especificações e provas de propriedades;
3. *Métodos formais são apenas úteis para sistemas críticos*. Existem várias aplicações de sucesso em sistemas não críticos, por exemplo as aplicações CICS da IBM [MUN90] e uma especificação de um Osciloscópio [DES90];
4. *Métodos formais requerem matemáticos altamente treinados*. Não é de todo verdade; alguns métodos não requerem uma forte base matemática por parte de quem os utilizam, por exemplo está é a proposta de OCL (Object Constraint Language) [WARM99];
5. *Métodos formais aumentam o custo do desenvolvimento*. Não é verdade; pelo contrário, a sua não utilização pode conduzir a altos custos em função de uma má especificação;
6. *Métodos formais são insuportáveis para os usuários*. Depende do grau de maturidade destes, no sentido de compreender os benefícios que advêm do uso de métodos formais;
7. *Métodos formais não são utilizados em softwares reais de grande escala*. Não é verdade, pois, por exemplo, sistemas complexos, críticos e de tempo-real são os que mais necessitam ser precisamente especificados;
8. *Métodos formais atrasam o processo de desenvolvimento*. Os atrasos se dão não apenas em função da inabilidade para utilização desses métodos, mas também pela falta de experiência na determinação dos prazos e

custo na medida certa. Deve-se salientar que a estimativa de custo e prazo não é algo fácil de se realizar;

9. *Métodos formais não são suportados por ferramentas.* Não corresponde à realidade Podemos citar alguns exemplos de ferramentas; o *Multiview* [CAS96]; o Prosoft [REI97] e ForMooZ [VAS95], além de [PIN96, SCH97]
10. *Métodos formais significam desistência dos métodos tradicionais de projeto da engenharia.* No contexto da engenharia, um método descreve a forma na qual um processo será conduzido. Segundo esse aspecto, é verdade que alguns métodos formais não possuem essa característica, mas não significam desistência dos demais métodos;
11. *Métodos formais apenas se aplicam a software.* Podem também ser aplicados igualmente a hardware, exemplo disso tem-se no provador de teorema HOL, usado para verificar partes do microprocessador Viper; assim como outros sistemas de provas têm sido aplicados a hardware [HAL90];
12. *Métodos formais não são desejáveis.* Isso não corresponde a realidade de fato, pois muitas agências internacionais, tais como, o International Electrotechnical Commission, o Ministério da Defesa do Reino-Unido (MoD), a European Space Agency e o Atomic Energy Control Board do Canada, sugerem a aplicação de métodos formais.
13. *Não existe apoio ao uso dos Métodos formais.* Inicialmente, o desenvolvimento foi um esforço isolado, todavia nos dias atuais o interesse pelo uso dos métodos formais cresceu vertiginosamente. Muitos têm sido os casos de sua utilização para atender as necessidades particulares de determinados usuários, sendo preciso adicionar-se operadores, estruturas de dados e estender-se as linguagens de forma mais breve e elegante. Portanto, há um esforço muito grande na comunidade científica/acadêmica e na indústria para o desenvolvimento e padronização de métodos formais.

14. *Pessoas de métodos formais sempre usam métodos formais.* Não é verdade, pois é reconhecido que existem certos aspectos de um sistema que podem ser melhor especificados através de outras abordagens.

Alguns desses mitos podem conter verdade em parte, outros são completamente equivocados, todavia, todos eles podem ser melhor caracterizados e tratados em [HAL90,BOW94a, BOW95].

4.2 – OS MÉTODOS DE ESPECIFICAÇÃO PRECISA

Especificamente, para a Engenharia de Requisitos, o uso das linguagens de especificação formal permite que as especificações de requisitos sejam manipuladas com rigor formal, podendo ser checadas segundo o conjunto de propriedades desejadas, bem como serem utilizadas na geração de um protótipo de implementação e obtidas por meio de vários passos de refinamento.

As linguagens de especificação, devido a sua complexidade e necessidade de real experiência matemática, são, em geral, de difícil compreensão e utilização por parte dos engenheiros de requisitos. Diferem, segundo [SOM97], de acordo com o paradigma utilizado, ou seja, de acordo com critérios de expressividade e estruturação, segundo o aspecto que modelam, tais com:

- (i) modelagem da funcionalidade;
- (ii) modelagem de propriedades de concorrência e distribuição; e
- (iii) modelagem de aspectos temporais.

Uma das classificações encontradas para os métodos formais é a apresentada por [Som97,SAN93, McDE96], que considera:

- (i) Métodos de Especificação Algébrica;
- (ii) Métodos Baseados em Modelos; e
- (iii) Métodos Baseados em Álgebra de Processos.

Entretanto, a classificação mais amplamente utilizada na área é a de [COH86], que considera os métodos formais sob duas categorias:

- (i) Métodos baseados em Modelos; e
- (ii) Métodos baseados em Propriedades.

Não obstante as várias classificações encontradas na literatura, é consenso que todos os métodos devam possuir aspectos básicos comuns que caracterizem uma boa descrição formal, ou seja:

- (a) *Fundamentação Matemática.* Deve existir um modelo matemático formal que dê sustentáculo aos elementos de definição e de especificação da técnica formal, permitindo que haja possibilidade de análise e verificação de especificações, prototipagem e implementação de testes de conformidade dos sistemas especificados;
- (b) *Descrição Mínima e Precisa.* Deve descrever de forma precisa todas as propriedades, de forma a se ter uma descrição com alto grau de abstração;
- (c) *Poder de Expressão.* Deve possuir capacidade de expressar de forma concisa, uma grande quantidade de propriedades importantes;
- (d) *Flexibilidade.* Fornecer condições para que haja modularidade, de forma a permitir pequenas alterações nas propriedades a serem especificadas;
- (e) *Facilidade de Entendimento.* No sentido de prover descrições facilmente compreensíveis.

O desafio na atualidade recai, cada vez mais, na implantação de sistemas complexos, porém confiáveis. Para tanto, temos necessidade de existência de métodos de modelagem precisos, completos, consistentes, não ambíguos, concisos e claros, mas que possam ser utilizados como parte do desenvolvimento. Nas seções que seguem, apresentamos alguns métodos formais existentes, tendo por base a classificação adotada em [COH86].

4.2.1 – MÉTODOS BASEADOS EM PROPRIEDADES

Nos métodos baseados em propriedades um sistema é modelado em termos de suas propriedades, segundo a abordagem algébrica. Nestes métodos não é construído um modelo explícito, mas faz-se uso de um conjunto de tipos e operações sobre estes tipos.

Cada tipo será considerado como um conjunto de valores e as operações serão modeladas como funções sobre esse conjunto de valores. Todas as operações são definidas por axiomas que expressam as restrições que devem ser satisfeitas pelas operações.

A estrutura de uma especificação algébrica é composta por duas partes bem definidas: (a) uma assinatura – um conjunto de tipos e um conjunto de funções com seus tipos – representando a parte da sintaxe; e (b) um conjunto de axiomas que expressam as restrições sobre as funções - a parte semântica.

Outra característica encontrada nos métodos algébricos é que muitos destes permitem definir tipos que poderão ser importados por outros tipos.

Como exemplos desse tipo de método, destacamos:

- (i) Larch [GUT93], que representa uma família de linguagens de especificação desenvolvida no MIT e Xerox PARC, cujo objetivo é suportar uma variedade de diferentes linguagens de programação, que usa lógica de predicado de primeira ordem. Esta linguagem é suportado pela ferramenta LP e é de ampla aceitação;
- (ii) OBJ que permite que especificações algébricas sejam testadas. É suportada pelo OBJ3 [GOG88]; e
- (iii) Ainda vários outros métodos, tais como Extended ML [SAN85], CLEAR [BUR80], CIP-L [BAU85], ASL [WIR86] , ACT ONE [EHR85], etc.;

Os métodos classificados segundo [SOM97, SAN93, McDE96] como baseados em álgebra de processos, são considerados dentro da classificação de Cohen, como baseados em propriedades. Estes métodos cujo poder de expressão, a concisão (propriedades sintáticas) e as propriedades semânticas (modelo matemático auto-contido) são importantes para descrições formais e independentes de implementação de sistemas concorrentes e distribuídos; são considerados, também, como métodos de especificação de concorrência [COH86], uma vez que estão relacionados com a modelagem das interações entre os processos concorrentes e distribuídos, bem como com a descrição de protocolos, comunicação e sincronismo nessa classe de sistemas. Esse tipo de método de especificação tem sido amplamente utilizado nas áreas de

telecomunicações, sistemas distribuídos e concorrentes, sistemas cooperativos, com a finalidade de especificar protocolos, dado que esses tipos de sistemas, por suas complexidade e não trivialidade, requerem sua concepção antes de serem implementados, de forma mais rigorosa e menos ambígua. São vários os trabalhos observados nesta área, onde citamos: [REI97, REI98, LIM98]. Dentre os métodos existentes destacamos:

- (i) CSP (Communication Sequential Processes) [HOA78];
- (ii) CCS (Calculus of Communication Systems) [MILN80]; e
- (iii) LOTOS (Language Of Temporal Ordering Specification) [IS88].

4.2.2 – MÉTODOS BASEADOS EM MODELOS

Esses métodos são os mais conhecidos e têm por base a teoria dos conjuntos e a lógica de primeira ordem. As especificações são modelos matemáticos abstratos do sistema a ser especificado. O modelo é constituído por objetos - que representam a entrada, saída e o estado interno do sistema -, bem como por operações e funções que manipulam estes objetos.

Nestes modelos, (i) o estado de um sistema é modelado em função de seus componentes, os quais são representados como conjuntos, funções, etc., (ii) as condições invariantes do estado são especificadas como predicados sobre os componentes; e (iii) as operações que manipulam o estado, são expressas como predicados que relacionam diferentes estados dos componentes. A explícita especificação do estado e os efeitos das operações sobre este, contrastam com os métodos algébricos que tratam da modelagem de tipos abstratos e suas operações.

Os métodos variam quanto à semântica, mas em geral fornecem meios para se construir especificações modulares, através da nomeação de coleções de componentes e predicados associados. Todavia, varia o cálculo utilizado para a manipulação dessas entidades, destinadas à prova de pré e pós condições.

Os métodos baseados em modelos preocupam-se, principalmente, com requisitos funcionais não críticos e com o comportamento seqüencial do sistema desejado.

Nesta categoria de métodos formais, enquadram-se:

- (i) A linguagem de especificação Z, proposta por Jean-Raymond Abrial [ABR81] tendo por base o princípio de que programas e dados podem ser descritos usando-se a lógica de primeira ordem tipada e a teoria dos conjuntos. Uma versão mais nova, [SPI89], adota uma notação mais elegante e concisa, na qual uma especificação é representada como uma coleção de esquemas. Segundo [WOO96] o conjunto de teorias utilizado inclui operadores padrão de conjuntos, compreensão de conjunto, produtos cartesianos e conjuntos das partes, enquanto que a lógica matemática utilizada é o cálculo de predicado de primeira ordem. Conforme dissemos no Capítulo 3, Syntropy [COO94] é um exemplo de técnica de modelagem que incorpora aspectos de formalização de requisitos na linguagem Z. Segundo [SAN93], Z tem sido utilizada para especificar sistemas reais, todavia não para descrever propriedades não funcionais, tais como usabilidade, desempenho, confiabilidade, etc.; tão pouco para a descrição de comportamento temporal ou concorrente, existindo para esse fim outras linguagens mais adequadas;
- (ii) A linguagem MooZ (Modular Object Oriented Z), extensão orientada a objeto da linguagem Z [MEI90], de forma a melhorar a manutenção, dispondo-se de meios adicionais para a modularização das especificações, bem como tornar possível a reusabilidade dessas especificações. Uma especificação MooZ consiste de um conjunto de definições de classe que podem estar relacionadas por alguma hierarquia; onde uma das classes assume o nome do sistema que se encontra sob definição, cujos estados e operações dos objetos dessa classe especificam o sistema em análise. Como extensão de Z, incorpora todos os construtores desta linguagem (given set, esquemas, descrições axiomáticas, tipos livres e restrições) e, adicionalmente, novos construtores: (a) classe; (b) esquema anônimo; (c) operações semânticas; e (d) e envio de mensagens. A semântica formal de MooZ pode ser encontrada em [MIN93], bem como uma extensão de MooZ com a linguagem Eiffel [COR94]. Quanto a ferramentas, já foi desenvolvido

um sistema que suporta desenvolvimento formal em MooZ, o ForMooZ [VAS95]

- (iii) VDM (The Vienna Development Method) [JON90], que se baseia no uso de conjuntos e mapeamentos finitos. Uma especificação em VDM consiste de duas partes: (i) a definição de variáveis abstratas usadas para representar o estado interno do modelo do sistema; e (ii) a definição de operações e funções que atuam nas variáveis definidas.
- (iv) A Linguagem ALBERT (Agent-oriented Language for Building and Eliciting Requirements for real-Time systems) projetada por Michaël Petit [PET90]. Hoje, trabalha-se com a segunda versão dessa linguagem - ALBERT II, proposta por Eric Du Bois [BOIS95, BOIS97, DUB98], que tem por base uma variante da lógica temporal de tempo-real, tratando-se de uma linguagem matemática apropriada para a descrição de seqüências de estados. Essa lógica é uma extensão da lógica de primeira ordem multi-tipada, envolvendo conceitos de variáveis, predicados, funções, tipos e agentes (vistos como especialização do conceito de objeto). O ALBERT II suporta, basicamente, a modelagem de requisitos funcionais, em termos de uma coleção de agentes que interagem, a fim de fornecerem serviços necessários à organização. Em [DUB95, BOIS97] pode ser encontrado o uso de uma ferramenta baseada em ALBERT, no contexto da segurança de sistemas de informação.
- (v) A Linguagem KAOS, foi proposta a partir do meta-modelo KAOS [DAR93], citado no Capítulo 3, com o objetivo de capturar aspectos de requisitos relativos a *quando*, *quem* e *porque*, em adição ao tradicional *que*. Essa linguagem fornece condições para a captura dos requisitos em termos de objetivos, restrições, objetos, ações e agentes. A semântica formal da linguagem de especificação formal KAOS pode ser encontrada em [DAR93, LAM98b], assim como tentativas de adequar esta linguagem à realidade cada vez mais exigente em [LAM98a];

- (vi) A Lógica Modal de Ações (MAL) [MAI87] que iremos tratar especificamente na Seção 4.3.1; e
- (vii) A Linguagem de Restrições de Objetos (OCL) [WARM99], a qual trataremos especificamente na Seção 4.3.2;

4.3 – AS LINGUAGENS PRECISAS UTILIZADAS NO DESENVOLVIMENTO PROPOSTO

Destacamos, na sequência, as linguagens de especificação precisa que serão utilizadas no processo de integração dos requisitos organizacionais às especificações formais de requisitos funcionais dos sistemas desejados. Assim sendo nas subseções que se seguem estaremos descrevendo mais amíúde as linguagens Structured MAL (Structured Modal Action Logic) [RYAN91] e OCL (Object Constraints Language) [WARM99], linguagens de especificação precisa baseada em modelos.

4.3.1 – A LÓGICA MODAL DE AÇÕES – MAL

A Lógica Modal de Ações (MAL) [MAI87], também conhecida com lógica dinâmica ou lógica multi-modal, foi desenvolvida dentro do projeto FOREST (Formal Requirements Specification Techniques) - projeto de cooperação entre o Imperial College, GEC Research, GEC Avionics e o Atomic Energy Research Establishment - como meio para se:

- (i) produzir um formalismo matemático à especificação de requisitos, sem a necessidade de que os especialistas tivessem muita habilidade e treinamento matemático apurado; e
- (ii) produzir um documento de especificação dos requisitos de sistemas de software baseados em estados, em tempo real. Na verdade, destinava-se à especificação de requisitos funcionais e de tempo, oferecendo meios para a representação de custos, requisitos de usabilidade ou mesmo restrições de implementação.

A idéia básica da Lógica Modal de Ações – MAL é a de representar as ações (atividades executadas pelas entidades do mundo real, chamadas de agentes), que fazem um sistema passar de um estado para outro, através de uma modalidade. Esta lógica tem por base a Lógica de Primeira Ordem Tipada, incluindo tipos, variáveis, símbolos lógicos, predicados, símbolos funcionais, símbolos constante, termos fórmulas atômicas e um certo número de axiomas e regras de inferências. São necessários dois tipos de informação sobre os agentes (entidades do mundo real) que estão executando ações:

- (i) Regras Estruturais, que indicam em que contexto uma ação realizada por um agente terá um contexto resultante em particular. Trata-se das pré e pós condições; e
- (ii) Regras Comportamentais, que fornecem informações sobre que seqüências de ações de um agente ou combinações de agentes são permitidas. Essas informações são reveladas pelo componente deôntico.

Devemos salientar que existem duas versões de MAL: (i) a versão inicialmente proposta [MAI87] e (ii) uma extensão da primeira, o Structured MAL [RYAN91]. Por tanto, resolvemos subdividir esta Seção, onde abordaremos inicialmente o MAL tradicional e em seguida o Structured MAL.

4.3.1.1 – A Linguagem MAL na Versão Original

Nesta Seção, serão abordados os principais aspectos da linguagem baseada em lógica modal de ações, proposta por Maibaum [MAI87]. Assim, como principais características da Lógica Modal de Ações, podemos citar :

- (a) *Agentes e Ações* - tipos pré-definidos que, respectivamente, definem as entidades do mundo real (qualquer componente do sistema que esteja sendo descrito, que tem existência independente; podendo ser ativo ou passivo e descrevem os processos (atividades) que os agentes podem executar;
- (b) *Modalidade* - representada por [], com o objetivo de capturar o efeito da ocorrência de ações, ou seja, [ação *a*] pode ser considerada como a pós-condição ou o resultado de uma ação *a* que tenha sido completada;

- (c) *Operadores deônticos* – no caso dois operadores *per* (permissão) e *obl* (obrigação) que permitem o controle sobre qual ação pode ser executada pelos agentes;
- (d) *Combinadores* a serem utilizados na composição de seqüências de ações (transações que podem ser vistas como novas ações atômicas); e
- (e) *Lógica temporal de intervalo*, do ramo de tipo linear.

Quanto a modalidade, a lógica modal tem uma família de operadores modais, um operador para cada ação que o sistema possa suportar, sendo sua semântica expressa por um conjunto de estados e uma família de relações sobre esses estados, cada uma interpretando a modalidade. Ou seja, através de axiomas representaremos situações do tipo

$$\phi \rightarrow [A,a]\phi$$

onde podemos ler: “ Sendo satisfeita a condição ϕ , então o agente A realiza a ação a e a condição ϕ também será satisfeita após a ocorrência de a ”. Neste caso, ϕ representa a pré-condição e ϕ a pós-condição, que na lógica proposta são fórmulas que também podem ter modalidade.

O formalismo introduzido por Maibaum, trata não apenas com a informação de comportamento, relativa ao desempenho de ações, mas também incorpora uma noção de causalidade, de forma a refletir idéias a serem feitas, com um agente ordenando (*obl*), proibindo (*ref*) ou permitindo (*PER*) outro agente a executar uma ação ou cancelando tais licenças. Dessa forma é que foi introduzida à lógica, um componente deôntico para possibilitar a prescrição de obrigação, proibição e permissão, através dos operadores deônticos *obl*, *ref* e *PER*, respectivamente. A axiomatização desses operadores, imperfeitamente, capturará a intuição sobre as estruturas de permissão e de como elas mudam. As regras de formação desses operadores são:

- (i) Se A é um conjunto de ações e c é um agente, então, $PER(c, A)$ é uma fórmula;
- (ii) Se $a \in Ac$ (Tipo ação) e c é um agente, então, $obl(c,a)$ é uma fórmula; e

- (iii) Se A é um conjunto de ações, então, $\text{ref}(A)$ é também um conjunto de ações.

Existe também um outro operador deôntico, equivalente a PER, relativo às ações; ao invés de termos um conjunto de ações, temos apenas uma ação. Trata-se do operador per, de forma que:

- (i) Se $a \in Ac$ e $c \in Ag$, então, $\text{per}(c,a)$ é uma fórmula.

Se só tivermos universos finitos de ações, os diferentes papéis de PER e per não são necessários.

Um outro operador é o de obrigação de seqüência, OBLS, no qual tem-se que

$$|- \text{OBLS } (s) \leftrightarrow \text{obl } (s_1) \wedge ((\wedge 1 \leq i \leq (n-1)[s_j] \ (1 \leq j \leq i) \text{obl}(s_{+1}))$$

onde s é uma seqüência de ações cujo comprimento é n . Semanticamente, temos que uma seqüência de ações s é obrigada, se somente se a primeira ação é obrigada e após realizar essa primeira ação, a segunda é obrigada e assim por diante.

Quanto à combinação, ações podem ser compostas usando-se os combinadores:

1. ‘;’ para indicar composição seqüencial, na forma $______ ; ______ : Ac \times Ac \rightarrow Ac$;
2. ‘||’ para indicar composição paralela, na forma $______ || ______ : Ac \times Ac \rightarrow Ac$; e
3. ‘+’ para indicar composição não determinística, na forma $______ + ______ : Ac \times Ac \rightarrow Ac$.

A semântica de combinadores seria, por exemplo, $\text{pré-condição} \Rightarrow \text{obl}(A, (a; b) \parallel c)$ indicando que sendo a pré-condição satisfeita, o agente A seria obrigado a realizar a ação a , seguida da b e em paralelo a ação c . Já as regras de inferência para os três tipos de combinadores, segundo Maibaum [MAI87], são:

$$\frac{\Gamma \vdash \alpha \rightarrow [c,a]\beta, \Gamma \vdash \beta \rightarrow [c,b]\gamma}{\Gamma \vdash \alpha \rightarrow [c, a;b] \gamma} \text{ (Seq.)}$$

$$\frac{\Gamma \vdash \alpha \rightarrow [c,a]\beta_1, \Gamma \vdash \beta_2 \rightarrow [c,b]\gamma, \Gamma \vdash \beta_1 \rightarrow \beta_2}{\Gamma \vdash \alpha \rightarrow [c,b]\gamma} \text{ (ExtSeq.)}$$

$$\begin{array}{c}
\Gamma \vdash \alpha \rightarrow [c, a;b] \gamma \\
\hline
\Gamma \vdash \alpha \rightarrow [c,a]\beta, \Gamma \vdash \alpha \rightarrow [c,b]\gamma \quad \text{(Nond.)} \\
\hline
\Gamma \vdash \alpha \rightarrow [c, a+b] (\beta \vee \gamma) \\
\hline
\Gamma \vdash \alpha \rightarrow [c,a]\beta, \Gamma \vdash \alpha \rightarrow [c,b]\gamma \quad \text{(Par.)} \\
\hline
\Gamma \vdash \alpha \rightarrow [c, a \parallel b] (\beta \wedge \gamma)
\end{array}$$

Quanto a extensão temporal, a noção primitiva de tempo é o intervalo, não existindo a noção de instantes de tempo. Em termos sintáticos as extensões temporais se enquadram em duas categorias:

- (i) *qualificação temporal de obrigações*, na qual tem-se expressões do tipo
 $\text{obl}(X, A, P)$
 significando que X é obrigado a realizar A após $\sim P$ ser verdade. P indica o intervalo que termina, quando se torna falso. Uma variação seria
 $\text{obl}(X, A, [Y,B])$
 significando que X é obrigado a fazer A, antes de Y terminar de fazer B;
 e
- (ii) *operadores temporais* para indicar a duração de fórmulas ou desempenho de ações. Vários são os operadores, por exemplo, BEFORE, DURING, etc.

As definições formais dos aspectos da lógica podem ser encontradas em [KEN91, KEN92, KEN93].

4.3.1.2 – A Linguagem Structured MAL

O Structured MAL (Lógica Modal de Ações Estruturada) resultou de uma extensão do MAL [RYAN91], de forma a permitir que a lógica de ações fosse estruturado. Essa extensão do MAL, incorporou os principais aspectos da lógica original MAL, a exceção de combinadores de ações, passando a ter algumas características adicionais e a se denominar de MAL⁺⁺ ou, simplesmente, Structured MAL. Esta extensão – Estruturado – assim se denomina pelo fato de que foi introduzida a idéia de dividir as especificações em partes menores com as quais se possa pensar a

nível “local”, de forma que seja possível se derivar propriedades do comportamento de um agente, usando-se apenas os axiomas atribuídos a este agente.

Desta forma, uma especificação Structured MAL corresponde a um conjunto de descrições de agentes (objeto) que realizam ações. Essas descrições consistem de um conjunto de declarações e axiomas que definem o comportamento dos agentes e que podem interagir compartilhando atributos e ações, rotuladas por *s* (*shared*) ou apenas com efeitos locais e por isso rotuladas com *l* (*local*).

Os agentes são as unidades da estrutura e tomam o papel de objetos, como em linguagem de especificação orientada a objeto, podendo ser ativo ou passivo. Aproveitando-se da estruturação, uma ação de modalidade anteriormente escrita como [Ac, Ag], em Structured MAL não mais precisa do nome do agente, ficando apenas [Ac]. Como resultado ainda da estruturação, temos que os atributos locais (rotulados por *l*) de um agente, só podem ser afetados por ações desse agente, sendo isso expresso por meio de axioma. Da mesma forma, se uma ação é local a um agente, um axioma é gerado, fixando que esta ação só pode atualizar os atributos desse agente.

O Structured MAL proposto não possui regras de construção; em seu lugar, tem-se os axiomas de localidade, que controlam a comunicação entre os agentes. Para ser capaz de fornecer os axiomas de localidade, a linguagem deve ser suficientemente rica em comparar ações e atributos.

Um componente, em termos lógicos, dentro de uma especificação é uma assinatura em conjunto com uma teoria sobre a assinatura. Em MAL uma assinatura é um par consistindo de dois conjuntos

$$S = \langle A, P \rangle$$

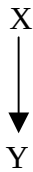
onde *A* representa um conjunto de ações e *P* um conjunto de proposições. Podemos dizer ainda que uma assinatura é a linguagem extra-lógica na representação da teoria, isto é, símbolos atributos, ações, símbolos de tipo usados e das funções usuais. Dada uma assinatura, proposições atômicas podem ser compostas a fim de serem formadas sentenças mais complexas. A composição é realizada por meio dos operadores booleanos convencionais $\wedge$, $\vee$, $\rightarrow$, $\neg$, etc., bem como o próprio construtor $[\]$, usado para descrever o efeito de ações. Existem ainda os quantificadores *Existencial* (dado um

tipo $s \in S$, $\exists s$) e *Universal* (dado um tipo $s \in S$, $\forall s$) utilizados nos axiomas. Todavia, deixou-se de ter os combinadores de ações.

Para permitir interações entre agentes, são definidos morfismos, relação primitiva entre os agentes, que representam mapeamentos entre duas assinaturas, mostrando-nos como renomear os elementos da assinatura a fim de demonstrar a relação de subagentes. Em Structured MAL encontramos três tipos de morfismos:

- (i) Herança;
- (ii) Agregação; e
- (iii) Cliente-Servidor.

A herança é capturada por um único morfismo, ou seja, se Y herda X, então existe um morfismo da especificação de X para a especificação de Y. Graficamente,

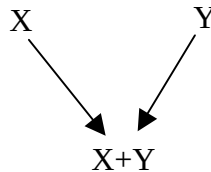


Na especificação do agente, em caso de haver herança, existe a cláusula **includes**, cuja expressão é

includes nome do subagente

que implicitamente declara todos os símbolos da assinatura e axiomas do subagente, no agente que está sendo especificado, mas renomeia os símbolos da assinatura, prefixando-os por meio do nome do morfismo.

A agregação é capturada por dois morfismos, ou seja, se $X+Y$ é uma agregação de X e Y, então existem morfismos de X para $X+Y$ e de Y para $X+Y$. Graficamente,

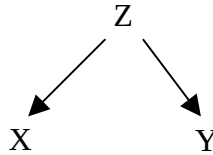


Em caso de agregação, a cláusula é a mesma,

includes nome do subagente

a diferença da cláusula de herança reside no fato de não ser possível acrescentar-se informações novas, como acontece com herança.

Cliente-Servidor é também capturado por dois morfismos, ou seja, se X é um cliente de Y, isto é, herda seletivamente de Y, então é construído morfismos com se segue. Seja Z a parte de Y que é herdada por X. Então, tem-se um morfismo de Z para X e um de Z para Y. Graficamente,



Em caso de haver uma estruturação cliente-servidor, é colocada no agente Cliente a cláusula

Servers nome do subagente servidor

A forma geral de uma especificação de um agente, na linguagem de especificação Structured MAL, é composto pelo nome do agente, a assinatura (uses, includes, servers, types, interfaces, atributos, ações, funções e variáveis) e por uma parte relativa ao conjunto de axiomas, conforme pode ser visto na Figura 4.2.

O uso da cláusula **uses** inclui os tipos definidos pelo usuário, na assinatura. Esses tipos são especificados conforme a Figura 4.3.

```

agent name
uses sort-name, ...
servers subagent-name
includes subagent-name via morphism-name
where
    subagent-action is agent-action;
    subagent-attribute is agent-attribute;

    localising action, ..., attribute, ...;
    actions
    [ls] act-sym: sort x ... x sort;
    ⋮
attributes
    [ls] attr-sym: sort x ... x sort → sort;
    ⋮
axioms
    axiom;
end agent-name

```

Figura 4.2 – Sintaxe das Especificações em Structured MAL

```

sort sort-name
fn : arb-sort x arb-sort ..... x arb-sort → sort-name;
:

```

Figura 4.3 – Sintaxe da Definição de Tipo em Structured MAL

Os parâmetros das funções não precisam ser do tipo que se esteja definindo e constantes, podem ser simplesmente, funções de aridade zero.

Na linguagem existem *termos*, *fórmulas atômicas* e *fórmulas*. Os *termos* são formados usando-se:

- (i) variáveis de tipos;
- (ii) símbolos função de tipo S;
- (iii) símbolos constante, na verdade funções de aridade zero;
- (iv) símbolos atributos;
- (v) símbolos ação, com zero ou mais termos parâmetro;
- (vi) para cada termo atributo **A**:s. existe um termo **&A: ref_s**; e
- (vii) para cada termo α : **ref_s** existe um termo $*\alpha$: s, seu conteúdo.

As *fórmulas atômicas* são igualdades de termos e fórmulas deônticas (per (a) e obl(a)), onde “a” é um termo ação. As *fórmulas* são compostas a partir de fórmulas menores com os conectivos usuais e os operadores de modalidade [.]; assim, “se ϕ e Ψ são fórmulas e a é um termo ação, então $\phi \wedge \Psi$, $\phi \vee \Psi$, $\phi \rightarrow \Psi$, $\neg \phi$ e $[a] \phi$ são fórmulas”.

Além disso, em Structured MAL, encontramos quatro tipos de axiomas:

- (i) axiomas de ação;
- (ii) axiomas com nenhuma ação;
- (iii) axiomas de permissão e
- (iv) axioma do tipo regra.

Nesta linguagem os axiomas de ação possuem a sintaxe,

pré-condição $\rightarrow$ [a] pós-condição

ou simplesmente,

[a] pós-condição

Em [GOL92, RYAN91] é encontrada toda a sintaxe para Structured MAL, segundo a qual nos basearemos. Já os axiomas com nenhuma ação possuem a sintaxe,

pré-condição $\rightarrow$ pós-condição

Nesta sintaxe, podemos incluir axiomas do tipo que envolve componente deôntico

pré-condição $\rightarrow$ obl (a)

De forma que se $\text{cond}_1 \rightarrow \text{obl}(a)$ e $\text{cond}_2 \rightarrow \text{obl}(a)$, então $\text{cond}_1 \vee \text{cond}_2 \rightarrow \text{obl}(a)$. Outro tipo de axiomas é o de permissão, que possuem a sintaxe

per (a) $\rightarrow$ pós-condição

Da mesma forma que no caso anterior, se $\text{per}(a) \rightarrow \text{cond}_1$, $\text{per}(a) \rightarrow \text{cond}_2$ e $\text{per}(a) \rightarrow \text{cond}_3$, então como resultado $\text{per}(a) \rightarrow \text{cond}_1 \wedge \text{cond}_2 \wedge \text{cond}_3$. O último tipo de axioma é a regra, que, simplesmente, é um termo atributo; onde um termo atributo tem a seguinte sintaxe,

nome-atributo (valores-parâmetros) = valor

De forma geral, quanto a semântica, existem diferentes abordagens. Alguns autores [MAN86, HOL93, McDE96] apontam três tipos de definição formal de semântica: (i) Denotacional; (ii) Operacional; e (iii) Afirmativa ou Axiomática. Em particular as estruturas semânticas da lógica modal de ações são do tipo denotacional, onde temos os modelos Kripke como é usual [CHEL80]. Para cada agente existe uma construção Kripke. Já uma composição de agentes corresponde a tomar o produto de construções Kripke, módulo de ações e atributos compartilhados. As palavras possíveis em uma construção são estados do agente. Os estados interpretam fórmulas não modais, localmente, avaliando-se os termos nos conjuntos que interpretam os tipos, para proposições atômicas, e pelas definições usuais da tabela verdade, para os conectivos não modais.

Uma estrutura de interpretação para uma assinatura com tipos S, consiste de:

- (i) Para cada tipo s em S , um domínio não vazio de indivíduos D_s ; e para cada símbolo função $f: s_a \times \dots \times s_n \rightarrow s$, uma função $[f] : D_{s1} \times \dots \times D_{sn} \rightarrow D_s$;
- (ii) Um conjunto de estados W e um estado inicial designado $w_0 \in W$;
- (iii) Um conjunto de $ACT \subseteq W \rightarrow W$, que será usado para interpretar termos do tipo ação;
- (iv) Para cada tipo s , um conjunto $REF_s \subseteq W \rightarrow D_s$. Esse será usado para interpretar termos do tipo referência (ref_s);
- (v) Para cada símbolo ação $a: s_1 \times \dots \times s_n \rightarrow ação$, uma função $[a] : D_{s1} \times \dots \times D_{sn} \rightarrow ACT$, com antes;
- (vi) Para cada símbolo atributo $A: s_1 \times \dots \times s_n \rightarrow s$, uma função $[A] : D_{s1} \times \dots \times D_{sn} \rightarrow REF_s$;
- (vii) Um par de conjuntos P (permissão) e O (obrigação). Ambos são subconjuntos de $ACT \times W$. Sendo usados para interpretar fórmulas deônticas.

Segundo [RYAN91] uma estrutura de interpretação é, portanto, um conjunto de estados e um meio de se interpretar termos ação como funções entre estados, assim como um meio de avaliação de fórmulas dependentes dos estados dentro dos estados. Para interpretar os quantificadores, cada variável da linguagem é atribuída a um indivíduo de tipo apropriado. Uma atribuição A é um mapeamento a partir de variáveis para indivíduos, no conjunto suporte de tipos das variáveis, ou seja,

- $x : s$ é atribuído a $A(x) \in D_s$;
- $\alpha : ref_s$ é atribuído a $A(\alpha) \in REF_s$;
- $x : ação$ é atribuído a $A(x) \in ACT$.

Para se avaliar fórmulas relativas a uma atribuição A nos estados, primeiro é feita uma avaliação de termos como se segue:

1. Variáveis

Se x é uma variável do tipo s , então sua interpretação no estado w , é escrita como

$$[x]_w^A \text{ é } A(x) \text{ em } D_s$$

Similarmente, se $\alpha : \text{ref}_s$, então

$$[\alpha]_w^A = A(\alpha) \text{ e}$$

Se x : ação, então

$$[x]_w^A = A(x)$$

2. Atributos

Se $A(t_1, \dots, t_n)$ é um termo atribuído do tipo s , então sua interpretação no estado w , escrita como

$$[A(t_1, \dots, t_n)]_w^A \text{ é } [A]([t_1]_w^A, \dots, [t_n]_w^A)(w)$$

Isto é, para interpretar $A(t_1, \dots, t_n) : s$ em w , aplica-se $[A]$ às interpretações $t_1, \dots, t_n$ em w , dando uma função de $W \rightarrow D_s$. Esta função é a referência do atributo envolvido, que é aplicado a w para conceder o resultado desejado.

3. Ações

Se $a(t_1, \dots, t_n) : \text{ação}$, então

$$[a(t_1, \dots, t_n)]_w^A = [a]([t_1]_w^A, \dots, [t_n]_w^A) \in \text{ACT}.$$

4. Referências

Onde $\&A$ diz respeito ao endereço de A . Assim,

$$[\&A(t_1, \dots, t_n)]_w^A = [A]([t_1]_w^A, \dots, [t_n]_w^A)$$

5. Valores

Onde $*\alpha$, corresponde ao valor de α . Assim,

$$[*\alpha]_w^A = [\alpha]_w^A(w)$$

A interpretação dos termos referência é coberta pelo item 4 acima e por este item. Não existe constantes ou funções do tipo ref_s . Os únicos termos do tipo ref_s são termos tais como, $\&A(t_1, \dots, t_n)$ e variáveis.

Quanto às fórmulas, tem-se que estas são verdadeiras ou falsas em um determinado estado, escrevendo-se $[\phi]^A$ para o conjunto de estados no qual ϕ é verdadeiro. Inicialmente, serão tratadas fórmulas atômicas, que são igualdades entre termos, permissões e obrigações de ações.

1. A fórmula atômica $t_1 = t_2$ é verdadeira, no estado w , se t_1 e t_2 têm a mesma interpretação em w . Formalmente,

$$w \in [t_1 = t_2]^A \text{ se só se } [t_1]^A_w = [t_2]^A_w$$
2. A fórmula atômica $\text{per}(a)$ é verdadeira em w se $\langle [a]^A_w, w \rangle$ está em P , onde P é o conjunto especificando quais ações são permitidas em quais estados e a é um termo do tipo ação.
3. Similarmente, $w \in \text{obl}(a)$ se $\langle [a]^A_w, w \rangle$ está em O , onde O é o conjunto especificando que ações são obrigadas em quais estados e a é um termo do tipo ação.

No tocante a interpretação dos conectivos, tem-se:

1. $w \in [\phi \wedge \Psi]^A$ se só se $w \in [\phi]^A$ e $w \in [\Psi]^A$.
2. $w \in [\phi \vee \Psi]^A$ se só se $w \in [\phi]^A$ ou $w \in [\Psi]^A$.
3. $w \in [\phi \rightarrow \Psi]^A$ se só se $w \notin [\phi]^A$ ou $w \in [\Psi]^A$.
4. $w \in [\neg \phi]^A$ se só se $w \notin [\phi]^A$
5. $w \in [[a]\phi]^A$ se só se $[a]^A_w(w) \in [\phi]^A$. Ou seja, para avaliar se $[a]\phi$ é verdade em w , avalia-se primeiro a ação a em w . O resultado é uma função em ACT. Então, aplica-se esta função a w , dando outro estado $[a]^A_w(w)$. A resposta é então dada se ϕ é verdade neste novo estado.

6. $w \in [\forall x.\phi]^A$ se $w \in [\phi]^{A'}$ para todas as atribuições A' , diferente de A no máximo pela atribuição de x .
7. $w \in [\exists x.\phi]^A$ se $w \in [\phi]^{A'}$ para pelo menos uma atribuição A' que difere de A , no máximo pela atribuição de x .

O tratamento dos quantificadores é dado nas duas últimas cláusulas. Se a fórmula ϕ não tem variáveis livres, então $[\phi]^A$ é independente da atribuição A .

Visto um pouco da sintaxe e da semântica de Structured MAL, apresentamos uma rápida exemplificação de um escopo de especificação, para o caso de um sistema de controle de vendas por encomenda, para uma Loja de Venda de CD's. Quando tratamos com sistemas onde existe um grande número de agentes e ações envolvidos, faz sentido dividir as assinaturas em outras menores e especificar estas separadamente, não só melhorando a escrita e a leitura da especificação, mas também fazendo com que condições tenham efeitos locais. Esse princípio de encapsulamento parece ter sido herdado das especificações orientadas a objeto e é o que caracteriza o aspecto estruturado da especificação. Uma estruturação gráfica, para o sistema de controle de vendas de uma Loja de CD's, pode ser encontrado na Figura 4.4.

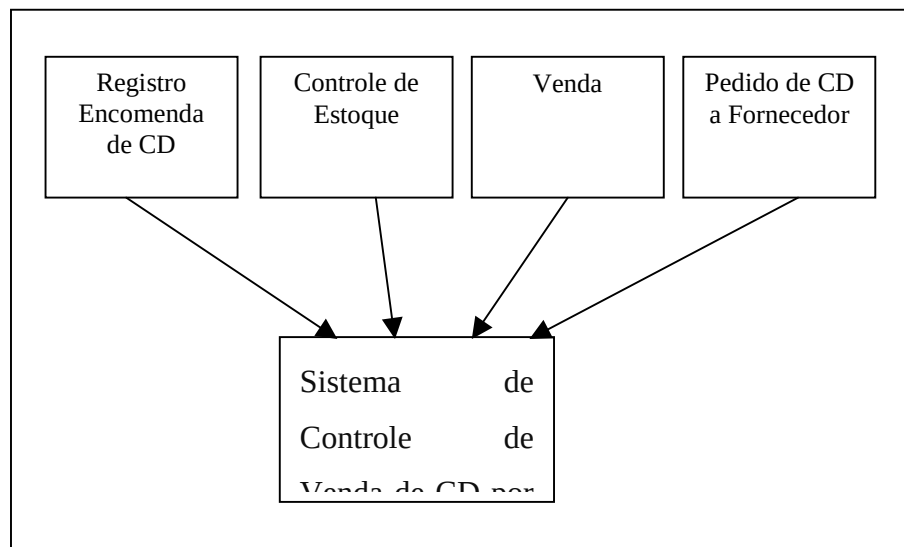


Figura 4.4 – Estruturação Gráfica de parte do Sistema de Controle da Venda de CD por Encomenda

Nesta estruturação gráfica, a confluência das setas indica agregação, ou seja, Sistema de Controle é composto por módulos que:

- (a) Registra a Encomenda de CD;
- (b) Controla o Estoque de CD na Loja;
- (c) Efetiva a Venda; e
- (d) Emite Pedido de CD a Fornecedor.

Na Figura 4.5, tem-se o escopo geral da especificação em Structured MAL, relativa a agregação do sistema de Controle de Venda por Encomenda da Figura 4.4.

Structured MAL não é em si uma linguagem de especificação, apesar de dispor de uma sintaxe concreta, daí o escopo de uma especificação em Structured MAL poder ser constituído por declarações e axiomas mantidos em separado, classificando-se cada um desses em diferentes categorias:

- (a) declarações de agentes;
- (b) declarações de ações;
- (c) declarações de tipos de entidades;
- (d) assinaturas de funções;
- (e) assinaturas de predicados;
- (f) assinaturas de ações;
- (g) declarações de constantes;
- (h) declarações de variáveis;
- (i) definições de ações;
- (j) condições de habilitação; e
- (k) axiomas estruturais.


```

agent Sistema de Controle de Venda de CD por Encomenda
  includes Registro Encomenda de CD;
  includes Controle de Estoque;
  includes Venda;
  includes Pedido de CD a Fornecedor;

  actions
    /* definição das ações*/
  end

  atributes
    /* definição dos atributos*/
  end

  axioms
    end

end agent Sistema de Controle de Venda de CD por Encomenda

```

Figura 4.5 – Escopo Geral de Especificação em Structured MAL

Cada uma das partes apontadas pela palavra reservada **includes** deve ser também especificadas como agentes, com a mesma estrutura da Figura 4.5. Por exemplo, na Figura 4.6, temos o agente Registro Encomenda de CD.

```

agent Registro Encomenda de CD
includes Cliente;
actions
  l Pegar Dados do Cliente;
  s Gerar Pedido
end
atributes
  s título do CD;
  s quantidade
end
axioms
  .....
end
end agent Registro Encomenda de CD.

```

Figura 4.6 – Definição do Agente “Registro Encomenda de CD” em Structured MAL

Esse agente possui :

- (i) atributos que podem ser modificados por ações locais ou externas, conforme sejam esses atributos locais (**l**) ou compartilhados (**s**), respectivamente:

- (a) título do CD; e
- (b) quantidade, que podem ser modificados por outros agentes; e
- (ii) ações que também podem ser locais ou compartilhadas:
 - (a) Pegar Dados do Cliente, que acessa os dados compartilhados do agente Cliente; e
 - (b) Gerar Pedido.

Destaca-se ainda que os relacionamentos podem ser modelados por funções ou predicados. Alguns trabalhos foram publicados com respeito a utilização de MAL em especificação de requisitos [FIA92, KEN93, RYAN93, ALE97b, ALE98a, ALE98b, ALE99a].

4.3.2 - A LINGUAGEM DE RESTRIÇÕES DE OBJETOS – OCL

A linguagem OCL (Object Constraints Language) começou a ser desenvolvida por volta de 1995 por Jos Warmer e Steve Cook, na IBM, como uma linguagem de modelagem de negócio, porém só foi formalmente definida como parte do padrão UML (Unified Modeling Language), segundo [WARM99], em 1997, pelo OMG (Object Management Group). Os usuários de UML (Unified Modeling Language) e outras linguagens de modelagem, podem utilizar OCL afim de especificar restrições e outras expressões vinculadas a cada modelo. Em [PAI99] podemos encontrar um estudo comparativo feito com relação a outras linguagens e OCL (Object Constraints Language).

OCL é uma linguagem de expressões textuais e precisas, utilizada na descrição de restrições em modelos orientados a objeto, como forma de complementar a parte gráfica dos modelos, para descrever restrições, que não conseguem ser diagramaticamente representadas.

Esta linguagem foi fortemente influenciada pelo método Syntropy – linguagem de modelagem orientada a objeto que combina OMT com a formalidade de um subconjunto da linguagem formal Z. Dessa forma, por essa influência de Z, classificamos OCL como uma linguagem formal baseada em modelos, muito embora seja uma linguagem que adota uma sintaxe simples, não simbólica, e que se utiliza de símbolos matemáticos mais simples da teoria de conjuntos e lógica, numa proposta de

ser precisa, porém de fácil compreensão (escrita-leitura) por quem não possua conhecimento matemático aprofundado.

Um dos mais importantes aspectos de OCL é que se tornou parte do padrão UML (Unified Modeling Language) [JACO99, BOO99, RUM99] para especificação. O seu objetivo foi o de acrescentar à definição de UML (Unified Modeling Language) [JACO99, BOO99, RUM99] a possibilidade de especificar restrições aos modelos UML.

Embora vários métodos de modelagem, tais como Syntropy [COO94] e BON [WAL95], utilizem o conceito de restrições, esta é uma idéia relativamente nova. Por *restrições* pode-se entender limitações em um ou mais valores de parte ou do todo de um objeto, em um modelo orientado a objeto ou de um sistema. Na prática são detalhes e informações extras, que não conseguem ser expressos pelos modelos gráficos. Antes de OCL estas restrições eram descritas em linguagem natural, resultando daí ambigüidades e imprecisões.

A utilização da idéia de restrições acrescenta algumas vantagens, donde podemos destacar:

- i. *Melhor Documentação* por acrescentar aos modelos, informações sobre os elementos e seus relacionamentos. Um modelo gráfico pode conter algumas restrições, como é o caso da multiplicidade nos relacionamentos; no entanto, alguns detalhes não são possíveis de serem representados;
- ii. *Precisão Aumentada*, dado que as restrições não podem ser interpretadas de forma diferente por várias pessoas, são não ambíguas, tornando o modelo ou sistema, sobre o qual se aplicam, mais preciso; e
- iii. *Comunicação sem Enganos*: em função de existência de precisão na descrição das informações, os desenvolvedores passam a ser capazes de comunicar seus intentos de forma não ambígua, dirimindo mais cedo, no processo de desenvolvimento, possíveis defeitos, salvando-se desperdício de dinheiro e evitando-se frustrações.

Apesar da existência de várias interpretações para o termo *restrição*, todos parecem estar de acordo com a necessidade de expressá-las nos objetos do sistema. O que difere

é a forma de como reagir quando da existência de quebra de restrição. Podemos classificar as restrições em:

- i. *Declarativa*, quando são apenas, como sugere o nome, uma declaração do que deve ser verdadeiro e não do como deve ser feito. Esse tipo é o utilizado em UML; e
- ii. *Operacionais*, quando inicia a execução de uma operação em caso de quebra de restrição, ou seja, quando prevê o tratamento de exceções.

Segundo essa classificação, OCL é uma linguagem declarativa, dado que as restrições não conduzem a efeitos colaterais, ou seja, o estado do sistema não mudará em função da avaliação de uma expressão em OCL, muito embora essa possa ser utilizada para especificar uma pós-condição. Desse fato, podem advir vantagens:

- (a) o modelador não precisar decidir como manipular a quebra de uma restrição;
- (b) as restrições não mudariam ao longo do tempo; e
- (c) a checagem de ocorrência de quebra de restrição é feita atomicamente.

Desta forma, a especificação de restrições é feita a nível conceitual, onde os aspectos de implementação são considerados como irrelevantes.

Esta linguagem de restrições pode ser utilizada para vários propósitos dentro do princípio de ser uma linguagem de modelagem a ser usada, entre outros, em conjunto com diagramas da UML:

- (i) para especificar invariantes de classes e tipos, no Modelo de Classes;
- (ii) para especificar invariantes de tipo, para Estereótipos previstas na UML;
- (iii) para descrever pré e pós condições de métodos;
- (iv) como uma linguagem de navegação;
- (v) para especificar restrições em operações.

O núcleo central de OCL está na navegação, ou seja, no relacionamento, entre os objetos. Desta forma, as expressões OCL permitem escrevermos restrições no comportamento dos objetos identificados, navegando-se entre estes objetos. A nível de especificação, as expressões aparecem nos invariantes, pré e pós condições.

Invariantes são restrições que estabelecem uma condição que sempre deve ser encontrada, ser verdadeira, por todas as instâncias da classe, tipo ou interface. São descritas usando-se uma expressão que será avaliada como verdadeira, se o invariante é satisfeito durante todo o tempo.

As *Pré-Condições* especificam as condições que devem ser verdadeiras (satisfeitas) no momento em que a operação vá ser executada. As *Pós-Condições* especificam as condições obrigatórias que devem ser verdadeiras assim que a operação acabar de ser executada.

Apresentaremos um pouco da sintaxe de OCL, com base no diagrama de classes da Loja de Venda de CD's, Figura 4.7 (vide Figura 2.8, Capítulo 2). Nesta Figura são apresentados os principais elementos envolvidos, onde vemos uma especialização da Classe Cliente, em Clientes Normais e Clientes VIPs cujas compras trimestrais ultrapassam o volume de 10 unidades. Da mesma forma que também podemos identificar uma especialização no tipo de vendas realizadas pela Loja. O sistema de controle atuará sobre a Venda, o Estoque e o Faturamento da Empresa. Esse diagrama serve apenas como ilustração para os elementos construtores da linguagem OCL.

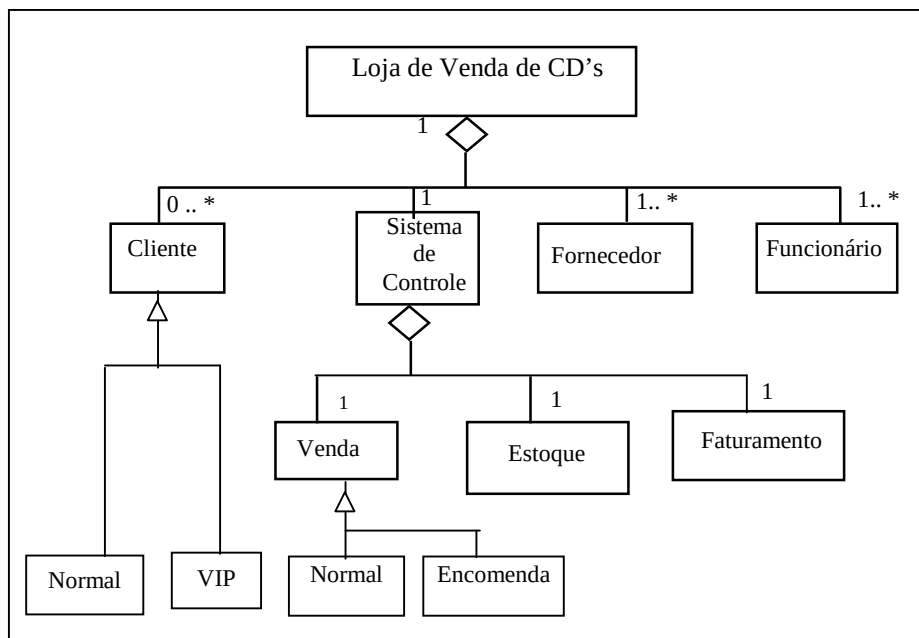


Figura 4.7 – Diagrama de Classes UML da Loja de Venda de CD's

Resumidamente, pois, dizemos que OCL (Object Constraints Language) é uma linguagem precisa, fácil de ser lida e escrita, mas fortemente tipada, onde podemos classificar os tipos como:

- (i) *pré-definidos*, onde se enquadram:
 - (a) *os tipos ditos básicos* – *Integer* são os números inteiros tal qual na matemática; *Real* são os números reais tal qual na matemática; *String* são as sequências de caracteres, por exemplo ‘OCL’; e *Boolean* com apenas os valores ‘true’ e ‘false’ -, que independem do modelo UML . Segundo [RIC98] o tipo enumerado é uma extensão dos tipos básicos através de uma enumeração de valores literais, tal como, *enum{valore₁, ..., valor_n}*; e
 - (b) *os tipos coleção*, cujos construtores são: *Set*, *Bag* e *Sequence*. Um *Set* é uma coleção que contem instâncias de um tipo OCL válido, não contendo elementos duplicados. Um *Bag* é uma coleção que contém elementos duplicados. Enquanto que uma *Sequence* é uma coleção de elementos duplicados, mas ordenados. As operações básicas sobre todos os tipos coleção podem ser vistas na Tabela 4.1.

OPERAÇÃO	DESCRIÇÃO
size	O número de elementos na coleção.
count (objeto)	O número de ocorrência do objeto na coleção.
includes (objeto)	Verdadeiro se o objeto é um elemento da coleção.
includesAll(coleção)	Verdadeiro se todos os elementos da coleção parâmetro estão presentes na coleção atual.
isEmpty	Verdadeiro se a coleção não contém elementos.
notEmpty	Verdadeiro se a coleção contém um ou mais elementos.
iterate(expressão)	Expressão é avaliada para todos os elementos na coleção. O tipo do resultado depende da expressão.
sum()	A adição de todos os elementos na coleção. Os elementos devem do tipo que admita adição.
exists(expressão)	Verdadeiro se a expressão é verdadeira para pelo menos um elemento na coleção.
forAll(expressão)	Verdadeiro se para todos os elementos a expressão é verdadeira.

Tabela 4.1 – Operações Básicas sobre Coleções

- (ii) tipos do modelo – todos os elementos presentes num modelo UML define tipos em OCL. Esses tipos são chamados de modelos que podem ser tipos, classes, interfaces, associações de classes, atores, casos de uso e tipos de dados definidos no modelo UML. No exemplo da Loja de venda de CD's, podemos definir o tipo Cliente.

Todos os tipos definidos no modelo UML (Unified Modeling Language) ou pré-definidos dentro de OCL (Object Constraints Language), recebem o tipo `OclType`. Vários são os operadores sobre cada um dos tipos.

Uma expressão em OCL é válida se for escrita segundo as regras de OCL, existindo um tipo associado a cada expressão. As expressões básicas em OCL podem ser uma das que segue:

- (i) a palavra chave *self*, utilizada para se referir ao objeto cuja operação está sendo realizada. No exemplo da Figura 4.7, poderíamos ter

VIP

self.nome = 'Fabrízia'

significando que Fabrízia é uma instância da classe VIP.

- (ii) A palavra chave *result* usada em uma pós-condição. Indica o valor de retorno da operação;
- (iii) Uma variável;
- (iv) A aplicação de uma operação, tais como:
 - Operações pré-definidas, como por exemplo, +, -, *, <, >, ...
 - Operações em atributos, como por exemplo, self.nome
 - Constantes;
- (iv) As operações sobre coleção, tais como: select, reject, collect, iterate, forAll e exists.

Se uma expressão OCL que segue uma declaração do contexto é usada como um invariante, isso significa que para todas as instâncias daquele contexto a expressão será avaliada como verdadeira. Por exemplo, no caso da loja os clientes são chamados de VIP se por trimestre compra mais que 10 unidades. Este fato pode ser colocado pelo invariante

VIP

CompraTrimestral >= 10;

O contexto de um invariante é, normalmente, uma classe, uma interface ou um tipo. As expressões podem ser utilizadas ainda, nas pré e pós-condições aplicadas às operações ou métodos, cuja sintaxe geral é

Tipo1:: operação (arg: Tipo2) : TipoRetorno

pre: expressãoOCL₁

post: expressãoOCL₂ ou

post: result = expressãoOCL₃

Por exemplo, no caso da loja de venda de CD se considerar que uma das operações dos Sistema de Controle é “Registrar Cliente”, então

Sistema de Controle:: Registrar Cliente (c: Cliente)

pre: not cliente -> includes (c)

post: cliente = cliente @pre -> including(c)

A pré-condição diz que o cliente que será registrado não deve já ser cadastrado, enquanto que a pós-condição diz que o conjunto de clientes após a operação de registro é idêntico ao conjunto de clientes antes da operação acrescido do novo cliente. O símbolo @ seguido da palavra chave pre indica o valor de um atributo ou associação no início da execução da operação. O contexto das pré e pós-condições sempre será uma operação ou método.

Apresentamos a sintaxe de alguns elementos de OCL, todavia maiores detalhes podem ser encontrados em [WARM99, JACO99, BOO99, RUM99].

Quanto à semântica da linguagem OCL, encontramos grupos de pesquisadores internacionais trabalhando no sentido de formular uma semântica precisa. Neste sentido, destacamos, fundamentalmente, duas propostas:

- (i) A de Hamie, Howse e Kent, integrantes do grupo de pUML [pG99], cuja proposta [HAM98a] é a de fornecer uma semântica para checar que OCL é não ambíguo e para melhorar a própria linguagem OCL. Este grupo escolheu trabalhar com a linguagem Larch [GUT93], para servir de referência inicial à semântica das expressões OCL que propõem. Objetivam uma semântica que possibilite uma base para a construção de ferramentas CASE, que suportem checagem de integridade de todos os modelos UML; e
- (ii) A de Richters e Gogolla, cuja proposta [RIC98] é a de fornecer um significado preciso aos conceitos de OCL, segundo alguns aspectos centrais dos modelos de classe em UML. Para isso, definem a sintaxe e a semântica de expressões OCL e apresentam exemplos de suas avaliações. Estes exemplos levam em consideração que a semântica formal facilita a verificação, validação e simulação de modelos, ajudando a melhorar a qualidade desses modelos e dos projetos de software relacionados, bem como o fato da formalização de OCL ainda não ser uma questão fechada.

O grupo do UML preciso (pUML) é um dos mais atuantes e que preparam uma proposta de definição precisa da UML e OCL a ser submetida à OMG (Object Management Group). Por este motivo optamos por relatar a proposta do referido grupo. Em [HAM98a], afirma-se que, como OCL não existe isoladamente, mas associado às partes de um modelo, definidas através de diagramas - no nosso caso, modelados em UML. Necessita-se de uma semântica relacionada com o núcleo da notação diagramática de UML. No caso específico, trabalham com o diagrama de classes.

A escolha pela linguagem Larch foi motivada, segundo os autores, pelo desejo de não estarem comprometidos no projeto de lógicas e raciocínio de sistemas, mas com o foco voltado para a elaboração do significado das próprias notações de modelagem. Ou seja, a semântica proposta, tendo por base Larch, será usada, pelos autores, como ponto de partida e referência ao desenvolvimento de ferramentas de checagem e animação, usando Prolog.

A semântica das expressões OCL, usando-se Larch, é alcançada fornecendo-se primeiro a semântica dos tipos de objetos, atributos e associações. A linguagem Larch (LSL) usa, segundo [HAM98a], módulos de especificação chamados de “tratados”, para descrever os tipos abstratos de dados e teorias.

Um tipo objeto é uma descrição de um conjunto de objetos, em termos de propriedades e comportamento que todos compartilham. No formalismo adotado, um tipo objeto **A** está associado com um tipo básico em Larch (LSL), consistindo de elementos que, de forma única, representam objetos do tipo. Os identificadores dos objetos são denotados por A . É definido um mapeamento $\tau: \text{OclType} \rightarrow \text{lslSort}$, onde $\tau(\mathbf{A}) = A$. Já os atributos de um tipo objeto são formalizados como funções sobre os tipos que representam esse tipo objeto, ou seja, a cada atributo **att** do tipo **T** de um objeto do tipo **A**, será associado um símbolo função denotado por att , sendo definido um mapeamento α_A que transforma um símbolo atributo em um símbolo função em LSL, através de $\alpha_A(\mathbf{att}:\mathbf{T}) =_{\text{def}} \text{att}: A, \Sigma \rightarrow T$.

Os valores dos tipos básicos em OCL são mapeados diretamente a tipos pré-definidos em LSL. Por exemplo, o tipo **Boolean** é mapeado para o tipo Bool, ou seja, $\tau(\mathbf{Boolean}) =_{\text{def}} \text{Bool}$; o tipo **Integer**, $\tau(\mathbf{Integer}) =_{\text{def}} \text{Integer}$; de forma similar, o tipo **String**, $\tau(\mathbf{String}) =_{\text{def}} \text{String}$; e, os tipos Enumerados, são diretamente mapeados em tipos

enumerados de LSL. Para os tipos coleção – conjuntos, bags e seqüências –, a linguagem Larch fornece tratados básicos que especificam operações básicas nessas estruturas. Os tipos **Set(T)**, **Bag(T)** e **Sequence(T)** são mapeados, em LSL, nos tipos, **Set[T]**, **Bag[T]** e **Sequence[T]**, respectivamente.

Operações ou solicitações no tipo **A**, tais como, **op(S):T** do tipo **T**, são mapeados como: $\alpha_A(\mathbf{op(S):T}) =_{\text{def}} \text{op: } A, S, \Sigma \rightarrow T$.

Para cada papel da associação **r**, de uma associação entre dois tipos **A** e **B**, é associado um símbolo função em LSL, ρ_A para o mapeamento $\rho_A(\mathbf{r:set[B]}) =_{\text{def}} \mathbf{r:set[A]}, \Sigma \rightarrow \text{set[B]}$. Todos os mapeamentos acima mencionados são sumarizados na Tabela 4.2.

OCL	LSL
A (tipo objeto)	A (tipo de identidades de objeto)
T (tipo valor)	T (tipo de valores)
Boolean	Bool
String	String
Integer	Integer
Collection(T)	Collection[T]
Set(T)	Set[T]
Bag(T)	Bag[T]
Sequence(T)	Seq[T]
att:T (atributo)	att:A, $\Sigma \rightarrow T$
op(S):T	op:A, S, $\Sigma \rightarrow T$
r:set[B] (papel)	r:Set[A], $\Sigma \rightarrow \text{Set[B]}$
r(S):set[B] (papel)	r:Set[A], S, $\Sigma \rightarrow \text{Set[B]}$
r:B (papel)	r:Set[A], $\Sigma \rightarrow \text{Set[B]}$
r(S):B (papel)	r:Set[A], S, $\Sigma \rightarrow \text{Set[B]}$

Tabela 4.2 – Mapeamento de Tipos, Atributos e Associações OCL

Em [HAM98a] também é definido um mapeamento μ com a assinatura $\mu: \text{OclExpression} \rightarrow \text{lslExpression}$, que mapea expressões OCL em expressões LSL com base nos mapeamentos expressos na Tabela 4.2. A definição do mapeamento μ é dado

na Tabela 4.3. A interpretação de uma expressão OCL, como em μ , é dado no momento correspondente a um estado do sistema σ . Nesta definição, variáveis em OCL são mapeadas em variáveis em LSL, ou seja, $\mu(v) = v$. Expressões da forma **v.att** são mapeadas em $\text{att}(a, \sigma)$. Expressões da forma **c→collect(v: T | expr-v)**, onde **c** é uma coleção e **expr-v** é uma expressão envolvendo **v**, são mapeadas em $\text{collect}_f(c, \sigma)$, onde $\mu(c) = c$ é a interpretação de **c**, e $f(v) = \mu(\text{expr-v})$. A única exceção é onde a expressão é um nome de papel, sendo, neste caso, a expressão **c→collect(r)** interpretada como $r(c, \sigma)$.

Expressão OCL	Expressão LSL
v (variável)	v (variável)
v.att	$\text{att}(v, \sigma)$
v.op(v')	$\text{op}(v, v', \sigma)$
v.r (r nome do papel)	$r(v, \sigma)$
c→select(v b-expr-v)	$\text{select}_p(c, \sigma)$ onde $p(v, \sigma) = \mu(b\text{-expr-v})$
c→reject(v b-expr-v)	$\text{reject}_p(c, \sigma)$ onde $p(v, \sigma) = \mu(b\text{-exp-v})$
c→collect(v expr-v)	$\text{collect}_f(c, \sigma)$ onde $f(v, \sigma) = \mu(\text{exp-v})$
c→iterate(v; acc=exp expr-v-acc)	$\text{Iterate}_{f, \text{exp}}(c, \sigma)$ onde $f(v, \text{acc}, \sigma) = \mu(b\text{-expr-v-acc})$
c.r(coleção c, papel r)	$r(c, \sigma), (\mu(c) = c)$

Tabela 4.3 – Definição do Mapeamento μ

Os valores **true** e **false** das expressões são mapeados, respectivamente, em **true** e **false**. As expressões de conjunto, tais como, $\text{Set}\{\}$ e $\text{Set}\{1,2\}$, são mapeadas, respectivamente, em $\{\}$ e $\{1,2\}$, sendo os demais valores de expressões mapeados de forma similar.

Desta forma, é proposta uma semântica para um subconjunto de expressões OCL em conjunto com a semântica de um núcleo central da notação diagramática de UML, em termos da linguagem Larch. Em [KEN99], ainda encontramos uma

semântica de meta-modelo para restrições estruturais em UML, considerando partes do diagrama de classes e invariantes, no estilo de OCL.

CONSIDERAÇÕES

De forma geral, uma linguagem de especificação precisa e formal é composta por:

- (1) uma *sintaxe* que define a notação própria da especificação e que permite que os requisitos sejam interpretados de forma única;
- (2) uma *semântica* que ajuda a definir um universo de objetos que serão utilizados na descrição do sistema; e
- (3) um *conjunto de relações* que define as regras que indicam que objetos satisfazem a especificação.

As propriedades de uma especificação precisa – ausência de ambigüidade, consistência e completude – constituem-se nos objetivos de todos os métodos de especificação.

Os métodos formais para a especificação de requisitos emergem lentamente e aos poucos são utilizados pela indústria do desenvolvimento de software, especialmente quando se trata de sistemas críticos que envolvam aspectos de segurança [BOW94a, VAS95, PIN96, CAS96, REI97, REI98, LIM98]. Todavia, salientamos que os métodos formais preocupam-se fundamentalmente com os requisitos funcionais do sistema que se deseja especificar.

Apesar de ser evidente a necessidade de utilização dos métodos formais, sua ampla utilização entre os desenvolvedores ainda é baixa, devido a fatores tais como:

- (i) dificuldade de entendimento das notações;
- (ii) dificuldade na formalização de certos aspectos de requisitos; e
- (iii) conservadorismo dos desenvolvedores de software.

É importante lembrar ainda, que não existe um único método perfeitamente adaptado a uma aplicação, em particular, sobretudo devido à limitações na expressividade, à habilidade de manipular grandes especificações, à disponibilidade de

ferramentas de suporte e outras restrições. A escolha de um ou outro método a ser adotado depende de quais propriedades serão modeladas e a praticidade em se estender o método. Assim, a utilização de linguagens formais para a formalização de requisitos em sistemas complexos pode recair em duas principais dificuldades [CAS96, CAS97, ALE99a]:

- (i) o escopo limitado das linguagens; e
- (ii) a aquisição preliminar dos requisitos relevantes do sistema, tais como, aspectos comportamentais, principalmente no tocante aos requisitos organizacionais, que não são considerados nas especificações.

Para evitar as limitações do escopo são propostos construtores mais ricos de forma que possam capturar os requisitos, enquanto que os aspectos comportamentais nos quais aspectos tais como segurança e confiabilidade são críticos, podem ser tratados pela modelagem dos requisitos organizacionais, através das técnicas apresentadas no Capítulo 3.

Neste sentido é que propomos, no próximo Capítulo, diretrizes que possibilitem mapear os requisitos organizacionais, capturados pela técnica de modelagem organizacional i^* , conforme visto no Capítulo 3, em aspectos de especificações precisas e formais, utilizando Structured MAL e OCL.

CAPÍTULO V

MAPEAMENTO ENTRE A MODELAGEM ORGANIZACIONAL E A ESPECIFICAÇÃO PRECISA DE REQUISITOS

O mundo da modelagem organizacional considera requisitos importantes que influenciam diretamente no desenvolvimento de sistemas, mas que não são tratados, em geral, pelos métodos tradicionais de modelagem, sob o ponto de vista da Ciência da Computação. Por outro lado, os métodos formais utilizados para tornar as especificações dos sistemas mais precisas e menos ambíguas, em geral, também não conseguem capturar os tipos de requisitos de que tratam especificamente as técnicas organizacionais, detendo-se, basicamente, na formalização da funcionalidade e do comportamento do sistema a ser desenvolvido. Desta forma, estamos propondo diretrizes de mapeamento para que os requisitos do tipo organizacionais, possam ser considerados no mundo dos métodos formais, objetivando que as especificações possam ser enriquecidas, a fim de serem mais condizentes com às necessidades dos clientes ou do mercado, conforme seja o tipo de sistema a ser desenvolvido.

5.1 – CONTEXTO DO PROBLEMA

No Capítulo 2, identificamos que, um ponto a ser melhor trabalhado dentro da Engenharia de Requisitos, está relacionado com a modelagem dos requisitos classificados como organizacionais. Isto porque, em qualquer desenvolvimento de software, faz-se necessário melhor identificar e mais profundamente compreender o domínio da aplicação, na medida do possível, objetivando com isso que se possa encontrar os requisitos esperados pelo cliente, descrevendo-os de forma precisa, explícita e clara.

O *domínio da aplicação*, segundo Michael Jackson [JAC95a], está relacionado com qualquer coisa que seja específica ao problema, podendo ser algo intangível, tal como legislações, regras de segurança, interesses dos participantes, metas e objetivos da própria organização, etc., sendo, pois, parte fundamental no processo da Engenharia de

Requisitos. Em outras palavras, podemos dizer que a fase inicial do processo de requisitos, que exige uma compreensão mais apurada sobre o domínio, é a mais importante de todo o processo, porque:

- (i) os sistemas sob desenvolvimento estão relacionados a uma grande quantidade de suposições relativas ao ambiente, no qual estes estarão embutidos. Assim, uma pobre compreensão acerca do domínio se constitui em causa primária de falha de projeto;
- (ii) os desenvolvedores precisam entender o que os usuários desejam e os usuários precisam explicitar o que os sistemas devem fazer, devendo ser evidenciadas as várias alternativas;
- (iii) mudanças nos requisitos são a principal fonte de problemas, sendo importante nesse contexto, o rastreamento dos requisitos [PIN96, PIN98b,CAS99, TOR99] ; e
- (iv) o conhecimento da estrutura organizacional e o conhecimento estratégico devem estar bem organizados, a fim de que o compartilhamento e o reuso dos requisitos sejam facilitados.

Considerando estes aspectos, no Capítulo 3 investigamos especificamente as técnicas para a modelagem dos requisitos organizacionais, que nos propiciam oportunidade de aprofundar o conhecimento sobre o sistema que desejamos modelar, tendo em vista que este é o ponto de nosso maior interesse.

Dentre as técnicas de modelagem organizacional elegemos a técnica i^* , por concentrar-se na análise das implicações sob o ponto de vista dos relacionamentos de dependências entre os atores de uma organização. Mesmo considerando essas evidências apresentadas por Eric Yu, entendemos que as demais técnicas para a modelagem dos requisitos organizacionais respondem satisfatoriamente ao seu fim, variando principalmente quanto a notação ou concisão gráfica. Consideramos que, por parte dos profissionais técnicos, aptos a lidarem com outras técnicas organizacionais, o uso da nossa proposta demandaria apenas uma pequena familiarização com a nova nomenclatura proposta.

Conforme mencionamos nas considerações do Capítulo 3, quando, na ocasião, apresentamos alguns pontos da comparação entre as técnica i^* e Bubenko [ALE99b], as técnicas de modelagem organizacional são de igual expressividade, pelo que poderíamos ter utilizado qualquer uma das técnicas, ou seja, nossa proposta transcende a técnica i^* , podendo, com pequenas adaptações, ser aplicada à outras técnicas de modelagem organizacional, tais como Bubenko [BUB83] e Kaos [DAR93].

A técnica i^* , dispõe de uma rica ontologia, cujo objetivo principal é o de ajudar na compreensão do ambiente da organização, na exploração de propostas alternativas de sistemas e na análise dos impactos das várias alternativas na organização, enriquecendo conceitualmente as técnicas de modelagem organizacional. Neste contexto:

- (i) a modelagem do ambiente organizacional i^* , em termos dos relacionamentos intencionais, propicia um modelagem mais rica, em comparação com as técnicas existentes, que em geral se baseiam somente em entidades e atividades (principais componentes das técnicas utilizada pela Administração);
- (ii) as interdependências entre os atores, em seu Modelo de Dependências Estratégicas (SD), são analisadas em termos de oportunidades e vulnerabilidades;
- (iii) a modelagem explícita das razões, em seu Modelo de Razões Estratégicas (SR), leva a uma compreensão mais aprofundada sobre o porquê dos sistemas estarem na organização em uma dada disposição;
- (iv) o uso de uma abordagem baseada no conhecimento facilita a coleta, organização, uso e reuso do domínio do conhecimento.

De forma geral, Yu apontou ainda algumas diferenças básicas entre i^* e as demais técnicas de Engenharia de Requisitos existentes, as quais iremos aqui apenas mencionar [YU95a], e que nos motivaram a escolha dessa técnica em detrimento das demais:

- (i) *Visão prescritiva versus descritiva.* As técnicas existentes propõem uma especificação de requisitos prescritiva. Nesse tipo de abordagem os agentes se

comportam de acordo com uma prescrição. O i^* adota uma visão descritiva, onde os agentes se comprometem com outros agentes, havendo um relacionamento multi-lateral entre os envolvidos;

- (ii) *Compleitude e Consistência.* A grande maioria dos esforços na Engenharia de Requisitos visam obter documentos de requisitos completos e consistentes. É natural, todavia que nas fase iniciais as informações coletadas ainda sejam bastante incompletas e algumas até conflitantes. Assim, durante a modelagem essas inconsistências podem ser úteis no sentido de apontarem para alternativas competitivas. Da mesma forma, a incopleitude pode Ter suas vantagens, uma vez que durante as fases iniciais, grande quantidade de informações é necessária, pois podem não estar relacionadas com a escolha das alternativas de processo.
- (iii) *A noção de processo.* Muitos modelos de requisitos tratam um processo como uma série de passos chamados de atividades onde o mecanismo básico de abstração é a decomposição. . Contudo, deve-se salientar que, na fase inicial dos requisitos, algumas atividades não são conhecidas de antemão ou não têm todas as restrições concisamente estabelecidas. Assim, no Modelo de Dependências Estratégicas (SD) um processo é descrito em termos de uma rede de relacionamentos intencionais entre os agentes. Aspectos não essenciais do processo são escondidos pelo uso da abstração de agente. Já os aspectos essenciais do processo são modelados com os relacionamentos de dependência entre determinados agentes, o se faz uso de diferentes tipos de dependências. Essas dependências são usadas para se expressar como os agentes restringem o comportamento dos outros agentes. Essas características não podem ser expressas usando-se apenas decomposição. Já o Modelo de Razões Estratégicas (SR) contem uma concepção de processo mais detalhada, permitindo descrever os elementos do processo e suas razões. Desta forma, o pensamento dos agentes sobre alternativas de processo podem ser explicitamente representado.

- (iv) *A noção de agente* [YU97c, YU98]. Na maioria das técnicas existentes a noção de agente (quando existe, sendo que Bubenko [BUB93b] não utiliza esta abordagem) está fracamente ligada à noção de atividades. Tipicamente, as atividades ou ações envolvidas são mapeadas para agentes, como acontece, por exemplo, com o KAOS [DAR93]. Em i^* , em função da visão intencional, o agente ou ator é fonte de intencionalidade. O termo ator é usado em i^* para expressar uma entidade genérica, que possui intencionalidade; e
- (v) *Engenharia de Requisitos orientada a objetivos*. Atualmente, vários trabalhos têm focado a orientação a objetivos [DAR93, DUB95, GUS95, LAM95, LAM98a, LAM98b, YU98, MYL99]. A idéia é que os sistemas sejam vistos como preenchendo alguns objetivos de alto nível em ambientes complexos. Em i^* , vemos essa noção, só que diferindo das técnicas específicas dessa área, em alguns pontos:
- (a) a modelagem é o ponto enfatizado, pois procura fornecer características de modelagem que permitam que o fenômeno do mundo real seja expresso em um modelo, apesar de também fornecer suporte à análise e ao projeto sem, necessariamente, fornecer indicações de automação;
 - (b) os objetivos de projeto precisam ser descobertos, como resultado do questionamento dos “porquês” ou das “razões”, pois não se conhece, a priori, o ambiente no qual o sistema estará incluso;
 - (c) apesar de algumas técnicas, tais como KAOS [DAR93], advogarem a estratégia da “redução” ou decomposição de objetivos, em muitas ocasiões isto não é possível. Nestas circunstâncias, a técnica i^* seria mais apropriada, pois objetivos não ajudam a reduzir totalmente outros objetivos;
 - (d) os atores sociais são vistos como sendo estratégicos;

- (e) a intencionalidade é distribuída, ou seja, cada ator possui o seu próprio interesse estratégico; e
- (f) o foco está nos requisitos iniciais, conduzindo a uma postura descritiva na modelagem, enquanto que a maioria das técnicas orientadas a objetivos são prescritivas. Neste sentido, i* pressupõe que um passo adicional teria de ser tomado a fim de se chegar a uma especificação de requisitos prescritiva.

Observamos todas as técnicas de modelagem de requisitos (funcionais, não funcionais ou organizacionais), onde notamos uma grande preocupação com a consistência e com a corretude dos requisitos. Daí derivar-se a importância dos métodos precisos como forma de reduzir, através de um formalismo matemático rigoroso, as incertezas, próprias das linguagens textuais que, em geral, acompanham os modelos gráficos que especificam os requisitos do sistema desejado. Neste sentido é que desde muito tempo, importantes pesquisadores da área da Engenharia de Requisitos ressaltam a importância da formalização dos requisitos [MYL90, BUB94b, BUB95b, DUB95, DARI97, PET97, SOM97, JAC98, WIE98].

Nesta linha de raciocínio, no Capítulo 4, procuramos investigar técnicas precisas de especificação de requisitos. Identificamos que estas são aplicadas sobretudo a requisitos classificados como funcionais. Em geral, os aspectos organizacionais que influenciam diretamente na funcionalidade ou no comportamento do sistema não são incorporados explicitamente pelas técnicas formais.

Neste Capítulo estamos propondo um conjunto de diretrizes de mapeamento para estreitar a distância entre o mundo dos requisitos organizacionais e o mundo das técnicas precisas de descrição de requisitos. Ressaltamos que a nossa proposta suporta o mapeamento apenas em uma direção – de requisitos organizacionais para requisitos precisos. Para o mapeamento “no outro sentido”, isto é, de requisitos precisos para requisitos organizacionais, outras pesquisas se fazem necessárias.

5. 2 – MECANISMOS DE MAPEAMENTO

Nesta Seção, trabalharemos com a questão específica de como reduzir a distância entre o mundo organizacional abstrato e o mundo preciso, procurando meios para integração desses mundos. Para tanto, trabalhamos com a técnica i^* para modelar aspectos do mundo organizacional, enquanto que para representar os aspectos precisos, utilizaremos as linguagens Structured MAL [KEN93] e a linguagem de restrições OCL [WARM99], utilizada como complementação da modelagem orientada a objeto, como forma de introduzir maior precisão.

Gostaríamos de salientar ao leitor, de antemão, que o trabalho ora proposto tem algumas limitações; entre elas, a de suportar o mapeamento em apenas uma direção (de requisitos organizacionais para requisitos “precisos”) e de tratar, apenas, de um subconjunto da técnica i^* . O mapeamento objetiva oferecer meios para que os requisitos organizacionais sejam mapeados e inseridos, na especificação precisa do documento de requisitos. Estamos tratando de elementos (requisitos) a nível mais alto e tentando transformá-los em algo mais concreto (mais próximo da implementação). Isto é natural, uma vez que, de maneira geral, os requisitos nas fases iniciais, independente de classificação, são ainda “abstratos”. Neste momento, técnicas organizacionais, tais como i^* são bastante úteis. A medida que se amadurece o conhecimento, os requisitos capturados nas fases iniciais vão ficando mais bem definidos e mais precisos, podendo, então, serem detalhados, através de uma técnica precisa, tal como o Structured MAL ou OCL.

Pensamos em fazer o mapeamento no sentido reverso, ou seja, partindo dos requisitos especificados precisamente, chegarmos aos requisitos mais imprecisos da fase inicial (organizacional). Contudo, esta abordagem não é simples, pois os requisitos sofreram refinamentos, sendo em várias situações acrescentadas restrições para tornarem-se requisitos concretos. Este mapeamento “de volta” será melhor analisado em trabalhos futuros (vide Seção 7.4). A segunda limitação do nosso trabalho está em não suportar os mecanismos de especializações de atores em i^* - Agente, Posição ou Papel por não haver um claro entendimento sobre o que realmente significam (os autores do i^* não definiram,

claramente, esta especialização sendo este um tópico atual de pesquisa do grupo de Toronto).

Como forma de esclarecer e melhor precisar o mapeamento entre i^* e as linguagens Structured MAL e pUML/OCL, introduzimos as “funções”¹ “ F_a ” para o mapeamento com Structured MAL e “ G_a ” para o mapeamento com pUML/OCL. Na notação adotada, “a” representará, em cada mapeamento, o elemento que se deseje expressar. Por exemplo, “a” poderá ser um ator (Act), uma composição de atores (ActComp), uma tarefa (Ta), um recurso (Re), um objetivo (Obj), um objetivo-soft (ObjSoft), ou um relacionamento do Modelo Estratégico SD (Relac_SD).

Outro ponto a ser esclarecido, é o fato de adotarmos um tipo enumerado para representar, nas linguagens precisas Structured MAL e pUML/OCL, o elemento objetivo-soft, relacionado aos requisitos não funcionais. Não utilizamos o tipo booleano, pelo fato da necessidade de expressarmos os vários tipos de contribuição desses objetivos-soft, previstas na modelagem segundo i^* (vide seção 3.2.2). Assim, nas especificações poder-se-á adotar um tipo enumerado² com as possibilidades: positiva (posit), negativa (negat), indefinida (indef), bastante (sup) e não bastante (naosub). Salientamos que nas linguagens precisas adotadas existe a possibilidade de se definir um tipo enumerado.

Nas duas seções que se seguem, 5.2.1 e 5.2.2, respectivamente, tratamos das diretrizes de mapeamento entre os elementos componentes da técnica de i^* com os elementos da linguagem de especificação Structured MAL e com os elementos da linguagem OCL. Para elucidarmos as diretrizes faremos uso do mesmo exemplo utilizado em todos os Capítulos anteriores, a Loja de Vendas de CD’s por encomenda.

¹ Não se trata de funções matemáticas formalmente falando, com domínio e contradomínio, apenas uma notação informal para dar mais precisão aos mapeamentos.

² Apesar da modelagem i^* contemplar cinco possibilidades de contribuições para um objetivo soft, nos estudos de caso apenas estaremos considerando as três primeiras possibilidades.

5.2.1 – DIRETRIZES DE MAPEAMENTO DE I* PARA STRUCTURED MAL

Na sequência, apresentamos diretrizes que ajudarão no processo de integração de descrições feitas através da técnica i*, como forma de completar uma especificação realizada com Structured MAL (Lógica Modal de Ações Estruturada), usado tradicionalmente para a especificação dos requisitos funcionais do sistema. As diretrizes ajudarão no estabelecimento do relacionamento entre fragmentos da especificação formal em MAL e os objetivos organizacionais descritos em i*. Como MAL possui o conceito de agente e pela natureza orientada a agente da técnica i*, muitos conceitos são relativamente diretos no processo de integração com o Structured MAL [KEN93].

Neste processo de mapeamento, vamos considerar os componentes estruturais dos Modelos de Dependências Estratégicas (SD) e do Modelo de Razões Estratégicas (SR), anteriormente apontados nas Seções 3.2.1 e 3.2.2. Para facilitar o entendimento das diretrizes a serem propostas, faremos uso dos modelos estratégicos da Loja de Venda de CD's por encomenda.

Até o momento, estabelecemos sete diretrizes de mapeamento, algumas das quais são subdivididas, considerando o mapeamento no sentido $i^* \rightarrow$ Structured MAL. Nos trabalhos [ALE98a, ALE98b, ALE99a] são apresentados os primeiros esforços de determinação dessas diretrizes.

Diretriz M1: Relacionada com o mapeamento dos atores do modelo i*;

SubDiretriz M1.1: Relacionada com o mapeamento da composição dos vários atores;

Diretriz M2: Relacionada com o componente tarefa, nos modelos da técnica i*;

SubDiretriz M2.1: Relacionada com componente tarefa no Modelo de Dependências Estratégicas (SD);

SubDiretriz M2.2: Relacionada com componente tarefa no Modelo de Razões Estratégicas (SR);

Diretriz M3: Relacionada com o componente recurso, nos Modelos Estratégicos da técnica i*;

Diretriz M4: Relacionada com os componentes objetivos bem definidos e objetivos soft, nos modelos da técnica i*;

SubDiretriz M4.1: Relacionada com objetivos bem definidos e objetivos-soft no Modelo de Dependências Estratégicas (SD);

SubDiretriz M4.2: Relacionada com objetivos bem definidos e objetivos-soft no Modelo de Razões Estratégicas (SR);

Diretriz M5: Trata dos relacionamentos externos entre os atores do i*;

Diretriz M6: Relacionada com as ligações do tipo decomposição de tarefa no Modelo de Razões Estratégicas (SR);

Diretriz M7: Relacionada às ligações do tipo meio-fim no Modelo de Razões Estratégicas (SR);

SubDiretriz M7.1: Relacionada às ligações do tipo meio-fim, onde o meio é uma tarefa e o fim pode ser tarefa, objetivo bem definido, objetivo-soft ou recurso;

SubDiretriz M7.2: Relacionada às ligações do tipo meio-fim, onde o meio é um objetivo (ou um objetivo-soft) e o fim é um objetivo bem definido ou um objetivo-soft.

Consideremos, inicialmente, o Modelo de Dependências Estratégicas (SD), já definido anteriormente e repetido na Figura 5.1, por conveniência. Neste modelo, são identificados os atores envolvidos e os relacionamentos externos entre esses atores. No caso, são ao todo seis atores: Loja, Cliente, Fornecedor, Atendente, Caixa e Apoio. Esses atores mantêm entre si uma rede de relacionamentos de dependências, onde estão envolvidos os elementos recurso, tarefa, objetivo e objetivo-soft, segundo visto na Seção 3.2.1. O ator que depende que um outro ator realize uma tarefa, forneça-lhe um recurso, satisfaça um objetivo ou ofereça meios para se alcançar um objetivo-soft, é chamado de dependente ou “dependeer”. Já o ator responsável por atender ao ator dependente, é chamado de “dependee”. No caso da Figura 5.1, vemos que a Loja objetiva que o Cliente atenda seu objetivo, “Compre CD’s”, uma dependência de objetivo. Nesta

situação a Loja é o ator dependente. Já o Cliente espera que: (i) a Loja forneça-lhe o “CD Solicitado”, uma dependência de recurso; (ii) a Loja disponha do CD desejado por este, “Ter CD Desejado”, uma dependência de objetivo; e (iii) a Loja ofereça um serviço com qualidade, desejo expresso na dependência de objetivo-soft “Qualidade[Serviço]”. Na situação apontada, o Cliente é um ator dependente do ator Loja. A mesma avaliação pode ser feita nos demais relacionamentos entre os atores. Em linhas gerais, a Loja espera aumentar sua clientela. Para isso pretende atrair clientes oferecendo o serviço de venda sob encomenda, mas para Clientes Especiais, cujo volume de compra trimestral seja de 10 unidades. No processo estão envolvidos o pessoal de Frente de Loja, que lida diretamente com o Cliente, representados no modelo pelos atores Atendente e Caixa, e o pessoal de Apoio que cuida da parte burocrática que envolve o serviço de encomenda e contato com Fornecedores. Tendo observado no modelo o comportamento externo entre os atores do processo, passaremos a apresentar as diretrizes propostas. Ressaltando que antes de cada diretriz, serão feitas algumas considerações, a respeito dos elementos envolvidos no mapeamento proposto.

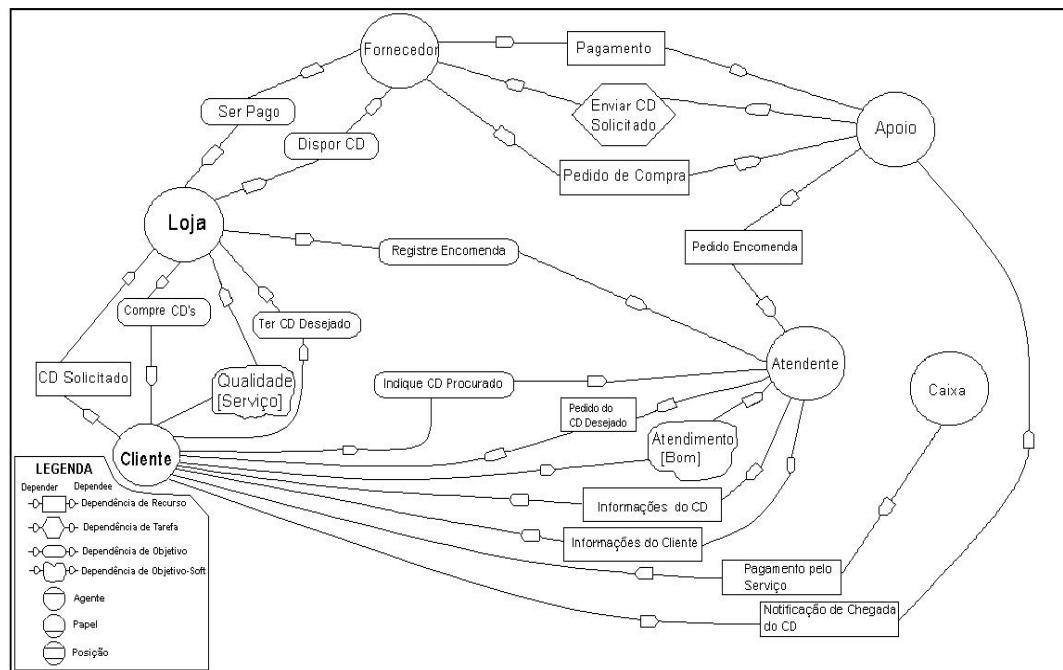


Figura 5.1 – Modelo de Dependências Estratégicas da Loja de Venda de CD's

Considerações M1: Relacionada com o mapeamento dos atores do modelo i^* .

Segundo a técnica i^* , as unidades semi-autônomas de um processo são chamadas de *atores* ou participantes desse processo. Os atores são considerados:

- (i) intencionais quando não executam simplesmente uma atividade e/ou consomem ou produzem uma entidade; e
- (ii) estratégicos quando não focam apenas seu objetivo imediato, mas se preocupam com as implicações de seu relacionamento com outros atores.

A nível de i^* estamos ainda nos primeiros estágios da modelagem, no Processo da Engenharia de Requisitos, onde os atores, individualmente, têm metas, intenções e capacidades para desenvolverem tarefas e terem objetivos bem definidos, mas também objetivos-soft, ainda não bem delimitados, em geral relacionados a requisitos não funcionais. Além disso os atores estarão relacionados entre si através das relações de dependências do tipo tarefas, de recursos, objetivos e objetivos-soft (Capítulo 3). Portanto, nesse nível de modelagem nem todos os atores são implementáveis sob o ponto de vista de uma implementação computacional.

Em MAL Tradicional, as entidades do mundo real, ou seja qualquer componente do sistema, passivo ou ativo, são chamadas de *agentes*. Onde cada agente é capaz de realizar ações, ter atributos que definem suas características e comportamentos expressos através dos axiomas. Além disso é possível tratar-se:

- (i) dos efeitos da ocorrência das ações, através da modalidade;
- (ii) dos aspectos Deônticos, que estabelecem permissões e obrigações na realização das ações; e
- (iii) dos aspectos Temporais, que tratam de duração de propriedades ou de ações (Neste trabalho não tratamos dos aspectos temporais).

Estamos, porém utilizando uma variação do MAL tradicional, o Structured MAL, onde as especificações são divididas em estruturas menores de serem tratadas e onde os agentes são as unidades da estrutura que tomam o papel de objetos, como nas linguagens

orientadas a objeto. Fundamentalmente, cada agente possui ações e atributos associados, incorporando os mesmos conceitos de Modalidade, Deônticos e Temporais.

Atores em i^* e Agentes em Structured MAL poderão (ou não) serem implementáveis do ponto de vista computacional. Sugerimos a inclusão no Agente de um atributo local do tipo enumerado $Atrb_{implementável}$, para definir esta característica. Os valores possíveis para o atributo são {verd, falso, indef}. A definição desta propriedade fica sob responsabilidade do engenheiro de requisitos, que poderá ser consultado quando necessário.

Inicialmente, o atributo $Atrb_{implementável}$ assume o valor indefinido (indef). Atores humanos, não são implementáveis do ponto de vista computacional. Assim, quando do mapeamento para Structured MAL, o atributo $Atrb_{implementável}$ assume o valor falso (falso).

Caso, seja necessário futuramente acessar informações referentes aos Atores humanos estes poderão ser modelados como agentes implementáveis. Assim, quando do mapeamento para Structured MAL, o atributo $Atrb_{implementável}$ assume o valor verdadeiro (verd).

Outro aspecto que somente estará presente nas descrições mais precisas em Structured MAL são os aspectos dêonticos e temporais. Em i^* estas características não são representadas. Com isso há um enriquecimento na especificação, pois os atores passarão a ter permissões e obrigações, bem como ser estabelecidas durações em suas tarefas.

Diretriz M1: Relacionada com o mapeamento dos atores do modelo i^* .

Os Atores (Act) nos modelos estratégicos da técnica i^* podem ser entendidos como Agentes (Ag) em Structured MAL.

$$F_{Act}: i^* \rightarrow \text{Structured MAL}$$

$$Act \rightarrow \text{agent Ag}$$

Desta forma, o construtor gráfico “”, que representa um ator no modelo i^* , será mapeado com um componente agente, “Ag”, em Structured MAL, com toda a estrutura que a linguagem permite. Esse mapeamento é direto.

Considerações M1.1: Relacionada com o mapeamento da composição dos vários atores.

Embora no Diagrama SD não haja o conceito explícito de decomposição, uma organização é na realidade composta (agregada) por vários atores. Em Structured MAL, o Diagrama SD será representado por um agente global que inclui os vários subagentes (atores do Diagrama SD). Na sintaxe de Structured MAL (Seção 4.3.1.2) esta situação será expressa pela inclusão da cláusula:

includes nome _do_subagente

para cada um dos sub-agentes que será incorporado pelo agente global, que representa a composição (vide figura 4.5, Seção 4.3.1.2).

SubDiretriz M1.1: Relacionada com o mapeamento da composição dos vários atores.

A Composição de Atores (ActComp) que representam um sistema em i^* , é mapeada em MAL como morfismo de Agregação de Agentes, expresso no agente composto.

$$F_{\text{ActComp}}: i^* \rightarrow \text{Structured MAL}$$

$$\text{Act} = \bigcup_{i=1}^n \text{Act}_i \rightarrow \text{agent Ag} \left(\bigwedge_{i=1}^n \text{includes (agent } A_i) \right)$$

Ou seja, o ator composto Act é representado pela união dos vários atores que o compõe nos modelos em i^* , considerando que cada um dos atores, Act_i , seja mapeado em Structured MAL, segundo M1, em um agente Ag_i . Assim, teremos que, se $\text{Act} = \bigcup_{i=1}^n \text{Act}_i$, então o agente composto em Structured MAL Ag incluirá em sua especificação, todos os agentes resultantes da utilização do mapeamento segundo M1.

Como forma de verificação das diretrizes expostas acima, de acordo com a Figura 5.1, utilizamos a loja de venda de CD's que é composta por seis atores: Loja, Cliente,

Fornecedor, Atendente, Caixa e Apoio (representa funcionário de apoio). Neste caso, aplicamos simplesmente a diretriz M1 sem nos preocupar, a esse nível de modelagem de requisitos, com a questão da computacionalidade ou não dos atores identificados. Assim, tem-se o mapeamento da Figura 5.2.

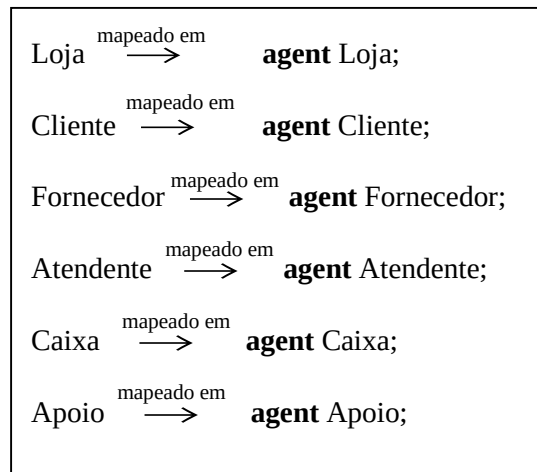


Figura 5.2 – Mapeamento de Atores em i* em Agentes Structured MAL

Cada um desses agentes em Structured MAL possui um escopo de especificação bem definido, com um nome correspondente, uma parte de assinatura e uma parte de axiomas, conforme pode ser visto na Figura 5.3. Usando a Subdiretriz M1.1, temos que os seis agentes mapeados (Loja, Cliente, Fornecedor, Atendente, Caixa e Apoio) fazem parte do agente que representa o sistema em questão (“Loja_Venda_CD”). Desta forma, tem-se, para a Loja de Venda de CD’s, a especificação em Structured MAL também apresentada na Figura 5.3.

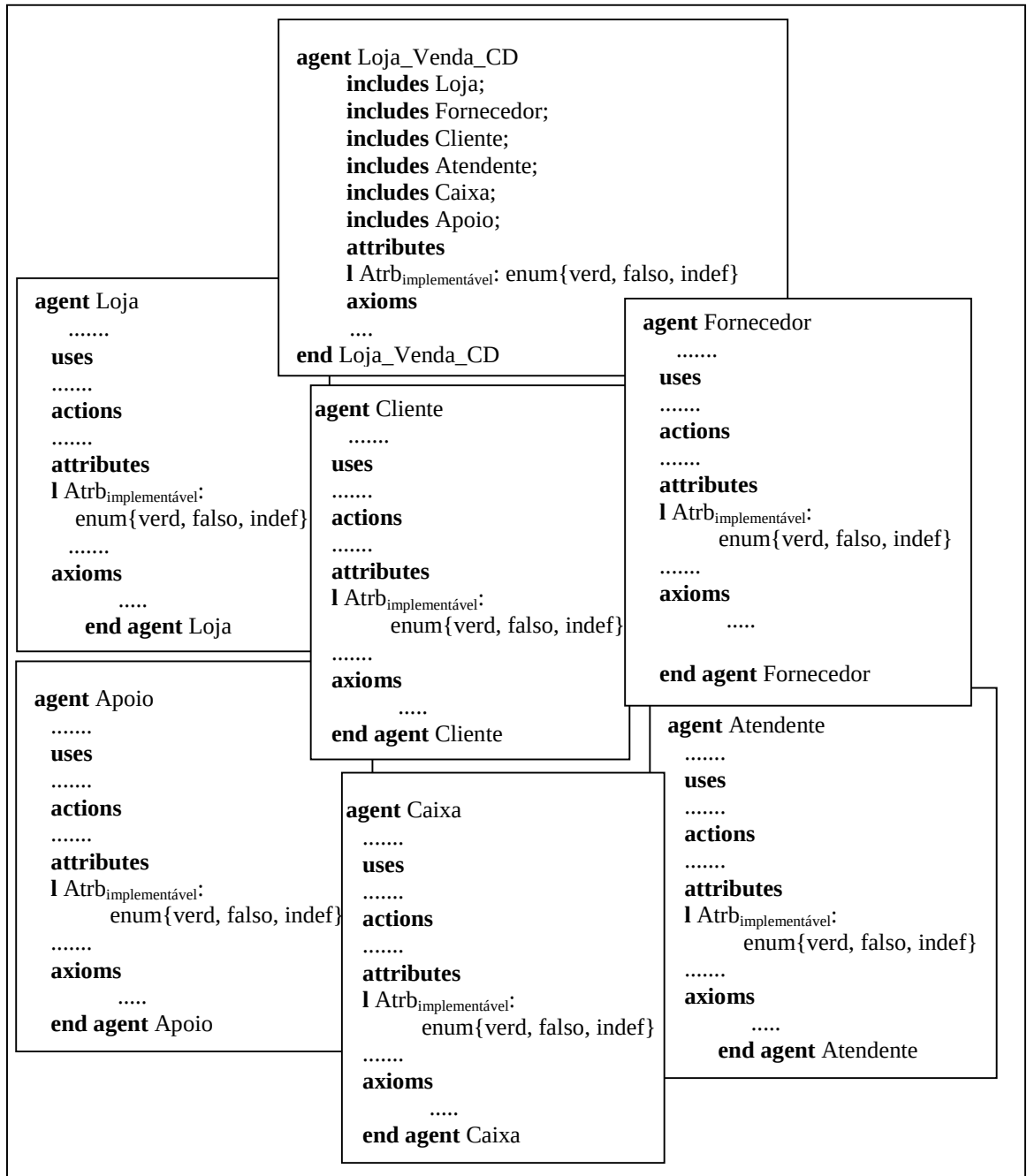


Figura 5.3 – Definição em Structured MAL da Loja de Venda de CD's com seus Agentes Componentes

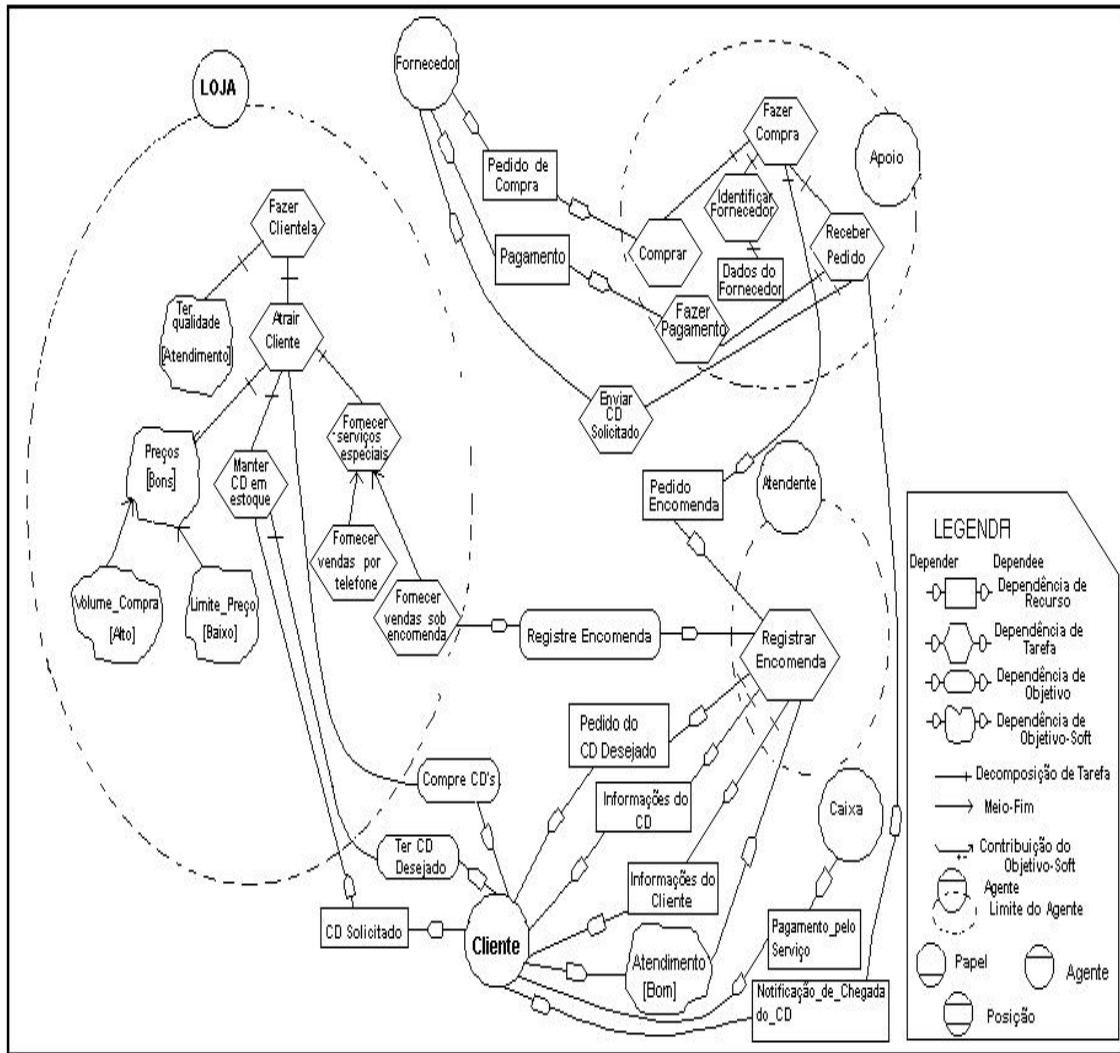


Figura 5.4 – Modelo de Razões Estratégicas para a Loja de Venda de CD's

Tendo observado os aspectos relativos aos Atores, presentes na organização, passamos a observar os relacionamentos entre estes, explicitados pelo Modelo de Dependências Estratégicas (SD) e o comportamento interno de cada ator, expresso pelo Modelo de Razões Estratégicas (SR). Consideremos o modelo (SR) já descrito anteriormente (vide Seção 3.2.2), mas repetido na Figura 5.4. por conveniência.

Considerações M2: Relacionadas com o componente **tarefa**, nos modelos da técnica i*.

Neste ponto, precisamos relembrar o conceito de tarefas segundo a técnica de modelagem organizacional. Em i^* as tarefas são as atividades a serem realizadas pelos diversos atores da organização descrita. Como estamos tratando com a modelagem organizacional e a nível mais abstrato, é possível que nem todas as tarefas, possam ser implementadas computacionalmente. Muito embora esperamos que em se tratando de atores implementáveis as tarefas sejam implementáveis.

Em Structured MAL, encontramos o conceito de ação, representando atividade (processo) que os agentes podem executar.

Em ambos os mundos, os conceitos dizem respeito a atividades que devem ser realizadas. Obviamente, tarefas relacionadas com atores implementáveis serão mapeadas através de ações em Structured MAL.

Destacamos ainda que na modelagem organizacional em i^* não é possível descrever restrições ou limitações quanto à ocorrência de tarefas. No entanto, quando mapeadas em ações de um agente em Structured MAL, podemos ser mais precisos e acrescentar informações sobre:

- (i) O efeito da ocorrência das ações, através da *Modalidade* [ação];
- (ii) Quando uma ação é possível ou obrigada ser executada pelos agentes através dos *operadores deônticos* *per* (permissão) e *obl* (obrigação), respectivamente;
- (iii) *Duração Temporal* (aspecto não tratado nesta tese).

Para tratar do mapeamento das tarefas, propomos a Diretriz M2.

Diretriz M2: Relacionada com o componente tarefa, nos modelos da técnica i^* .

Todas as tarefas implementáveis computacionalmente, dos modelos estratégicos i^* , serão traduzidas como ações de um agente em Structured MAL.

De forma que

$$F_{ta}: i^* \rightarrow \text{Structured MAL}$$

$$\text{Tarefa} \rightarrow \text{Ação}$$

Considerações M2.1 e M2.2: Encontram-se relacionadas com o componente tarefa nos Modelos de Dependências Estratégicas (SD) e Razões Estratégicas (SR), respectivamente

Um dos aspectos que precisa ser especificamente tratado é a questão da visibilidade das ações em Structured MAL, onde existe explicitamente o conceito este conceito. A localidade de uma ação significa que esta só pode ser utilizada pelo agente no qual se encontra definida, sendo rotulada com l (local). O compartilhamento de uma ação significa que esta pode ser utilizada por outros agentes, sendo rotulada com s (shared). Embora, não encontremos explicitamente o conceito de visibilidade em i^* , nos valeremos desse conceito para identificar se o componente da dependência pertence ao Modelo de Dependências Estratégicas (SD) ou ao Modelo de Razões Estratégicas (SR). Assim, propomos a subdiretriz M2.1, para tratar todas as tarefas encontradas no Modelo de Dependências Estratégicas (SD), associando a estas o conceito de visibilidade compartilhada; e a subdiretriz M2.2 para tratar às tarefas no Modelo de Razões Estratégicas (SR), associando a estas o conceito de visibilidade local.

SubDiretriz M2.1: Relacionada com o componente tarefa no Modelo de Dependências Estratégicas (SD).

As tarefas são mapeadas como ações compartilhadas do agente, em Structured MAL. Esse agente (denominado por Yu como “dependee”) é aquele que tem a responsabilidade por executá-la, em i^* .

As ações devem ser compartilhadas porque, em se tratando do modelo de dependências estratégicas (SD), as tarefas são elementos componentes da relação de dependência entre atores, ou seja, existe um ator que dependerá de outro ator, para que a tarefa seja realizada. O ator executor é o indicado pelo sentido da seta. Assim, se um ator A_1 depende de que um ator A_2 execute uma tarefa Ta_1 , então esta tarefa, segundo a diretriz acima, será mapeada como uma ação compartilha, (s Ac_1), que pertence ao agente Ag_2 , mapeado de A_2 , segundo M1, ou seja, teremos

$$F_{ac}: i^* \rightarrow \text{Structured MAL}$$

$Ta_1 \rightarrow (actions \ s \ Ac_1) \in Ag_2$

Na Figura 5.1, observando, por exemplo, o relacionamento de dependência entre os atores “Fornecedor” e “Apoio”, vemos uma dependência de tarefa, isto é, o ator Apoio depende do ator Fornecedor para que a tarefa “Enviar CD Solicitado” seja realizada. Na Figura 5.5 apresentamos um trecho de especificação em Structured MAL, construído de acordo com a diretriz M2.1.

```

agent Fornecedor
  ....
includes
  ....
actions
  s Enviar_CD_Solicitado;

attributes
  l Atrbimplementável: enum{verd, falso, indef}
  ....
axioms
  ....
end agent Fornecedor

```

Figura 5.5– Especificando Tarefas, Modelo SD, como Ações Compartilhadas em Structured MAL

Na Figura 5.5, vemos que a ação de Enviar_CD_Solicitado é uma ação que pode ser compartilhada por outro agente, mas que, no caso, deve ser realizada pelo agente Fornecedor. Salientamos que, conforme seja necessário, teremos que nos preocupar com a sintaxe própria da linguagem Structured MAL.

SubDiretriz M2.2: Relacionada com o componente **tarefa** no Modelo de Razões Estratégicas (SR).

*As **tarefas** pertencentes a um ator que está sendo detalhado pelo modelo SR são mapeadas como **ações locais** ao agente, em Structured MAL.*

Quando trabalhamos com o Modelo de Razões Estratégicas (SR), visto na Seção 3.2.2, observamos o comportamento interno de cada ator. Nesta situação, as tarefas modeladas

internamente são realizadas pelo ator que se esteja detalhando o comportamento. Portanto, quando avaliamos o comportamento interno, se um ator A_1 executa uma tarefa Ta_1 , então essa tarefa será mapeada como uma ação local, Ac_1 , pertencente à especificação em Structured MAL do agente Ag_1 (mapeado de A_1), segundo M1. Teremos

$F_{ac}: i^* \rightarrow \text{Structured MAL}$

$Ta_1 \rightarrow (\text{action } l \text{ } Ac_1) \in Ag_1$

Observamos que, no Modelo de Razões Estratégicas da Figura 5.4, somente três atores foram detalhados: Loja, Apoio e Atendente. Destacamos dentre estes, o ator Atendente, que tem, por obrigação, a realização da tarefa de “Registrar_Encomenda”. Segundo a diretriz M2.2, esta tarefa será traduzida em um trecho da especificação em Structured MAL, como visualizado na Figura 5.6.

```

agent Atendente
  actions
    l Registrar_Encomenda;
  attributes
    l Atrbimplementável: enum{verd, falso, indef}
    ....
  axioms
    ....
end agent Atendente

```

Figura 5.6 – Especificação de Tarefa Interna, Modelo SR, em **Ação Local, em Structured MAL**

Considerações M3: Relacionadas com o componente **recurso**, nos modelos da técnica i^* .

O elemento recurso, nos modelos estratégicos da técnica i^* , corresponde a uma entidade física do mundo real ou uma informação, um objeto que existe concretamente. Neste caso, os recursos a nível de implementação serão entidades implementáveis, dado que são objetos concretos que são manipulados pelos atores, segundo suas necessidades.

Em Structured MAL, o elemento que tem vida própria é o agente, pois este também pode ser tomado como uma unidade que assume o papel de um objeto, à semelhança dos princípios da orientação a objetos

O mapeamento do recurso em um agente em Structured MAL, dá ao recurso uma capacidade de expressão muito maior do que a necessária, dado que pode possuir atributos e realizar ações. A nível da modelagem organizacional em i*, interessa-nos apenas a informação da disponibilidade (ou não) do recurso. Assim, esta situação é representada através da inclusão de um atributo do tipo booleano, no agente correspondente ao recurso modelado. Caso o atributo seja verdadeiro (true) significa a disponibilidade do recurso, caso contrário (false), o recurso não estará disponível. O nome desse atributo fica de livre escolha, apenas sugerimos que seja o mais próximo do elemento recurso, unicamente por uma questão de simplicidade. Desta forma, propomos a diretriz M3.

Diretriz M3: Relacionada com o componente **recurso**, nos Modelos Estratégicos da técnica i*.

*Os **recursos** dos modelos estratégicos serão traduzidos como **agentes** em Structured MAL, com um **atributo compartilhado** ($atbr_{rec}$) **do tipo booleano**, que indique a disponibilidade desse recurso.*

Assim, se **Re** é um recurso, em qualquer dos modelos estratégicos do i*, então será mapeado como um agente de mesmo nome,

$$\begin{aligned} F_{Re}: i^* &\rightarrow \text{Structured MAL} \\ Re &\rightarrow \text{agent Re} \end{aligned}$$

devendo existir um atributo, ($atbr_{Re} : \text{boolean}$) no (**agent Re**), que indicará a disponibilidade (ou não) do recurso, para a tarefa que o esteja solicitando.

Desta forma, considerando os modelos de dependências estratégicas (SD), Figura 5.1, e de razões estratégicas (SR), Figura 5.4, observamos os recursos: *Pagamento*, *Pedido_de_Compra*, *Pedido_Encomenda*, *Pedido_do_CD_Desejado*, *Informações_do_CD*, *Informações_do_Cliente*, *Pagamento_pelo_Serviço*, *Notificação_de_Chegada*

do_CD, *CD_Solicitado* e *Dados_do_Fornecedor*. Na Figura 5.7, temos algumas destas classes.

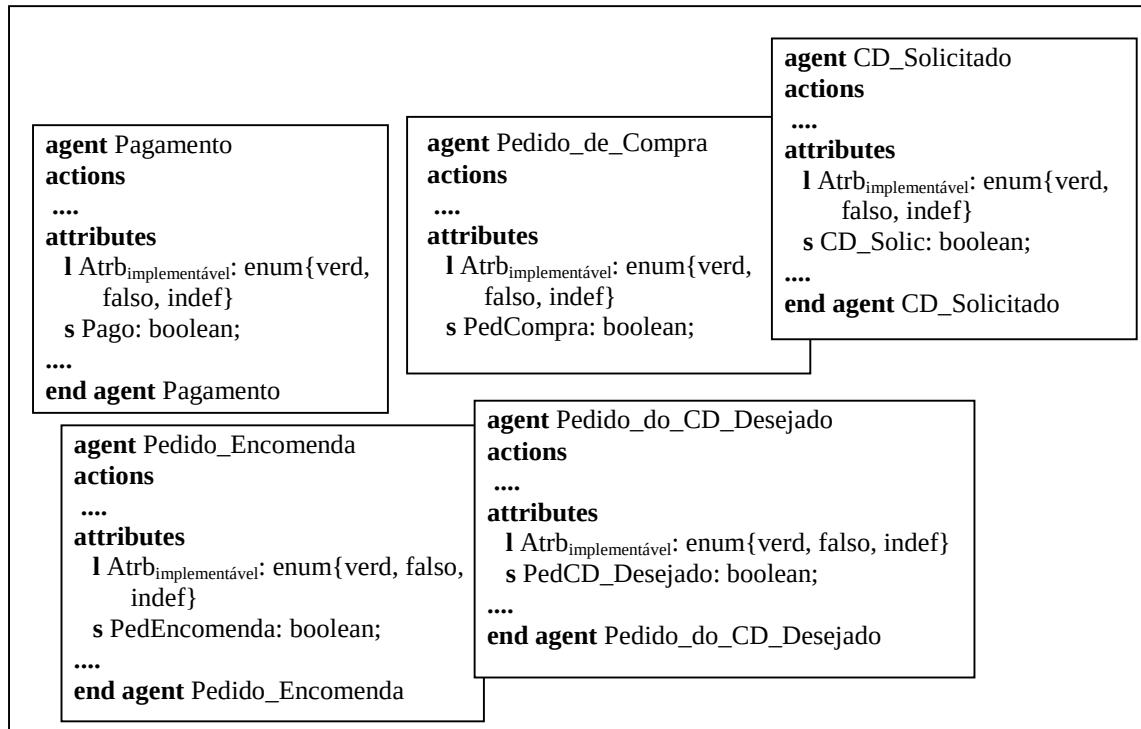


Figura 5.7 – Especificando Recursos em i*, como Agentes em Structured MAL

Considerações M4: Relacionadas com os componentes **objetivos** bem definidos e **objetivos soft**, nos modelos da técnica i*.

Outros dois componentes importantes, nos modelos estratégicos de i*, dizem respeito a objetivos. Estes objetivos podem ser bem definidos ou então tratarem de metas vagas, normalmente relacionadas a requisitos não funcionais, denominados de objetivos-soft, conforme abordado no Capítulo 3. Esses objetivos, a nível dos modelos estratégicos, podem ainda estar muito vagos; uma vez que é dito o que se quer mas não como se espera alcançar.

Um *objetivo* em i* corresponde a uma condição ou estado do mundo que um ator gostaria de alcançar. O conceito de *objetivo-soft* é similar ao de objetivo, exceto pelo fato de que a condição de sucesso não é precisamente definida a priori. Os objetivos, a nível

de i^* , envolvem aspectos subjetivos que vão sendo esclarecidos durante o desenvolvimento do processo.

Do ponto de vista do Structured MAL, podemos associar aos conceitos de objetivo e objetivo-soft o conceito de atributo. Como em i^* , objetivos e objetivos-soft representam desejos, metas ou intenções dos atores, propomos representar os objetivos e objetivos-soft de um ator em i^* como atributos em Structured MAL.

Cada um dos objetivos (bem definidos) será representado por um atributo do tipo booleano para sinalizar a satisfação (true) ou não (false) destes objetivos. Obviamente que alguns desses objetivos não são implementáveis, porque podem representar aspectos almejados por um ator, mas que não conseguiremos traduzir em alguma rotina executável, sob o ponto de vista de implementação. Deixamos para um passo a posteriori do desenvolvimento a preocupação com este aspecto, o que não invalida a diretrizes ora propostas. Destacamos que é esperado que os atores (agentes) implementáveis possuam objetivos bem definidos implementáveis.

Com relação aos objetivos-soft, relacionados a requisitos não funcionais, serão associados atributos do tipo enumerado. Os valores possíveis para o atributo são {posit, negat, indef, sup, naosub}, representando as possíveis contribuições positiva, negativa, indefinida, bastante e não bastante, respectivamente, desses objetivos, expressos no Modelo de Razões Estratégicas (SR), (Seção 3.2.2). Nos axiomas em que apareça, o valor do atributo associado a um objetivo-soft deve ser avaliado. Ao longo do desenvolvimento é natural que os objetivos-soft sejam melhor explorados, para tanto, sugerimos aplicar as técnicas e estudos relacionados em [CHU99] e aplicáveis a requisitos não funcionais.

Diretriz M4: Relacionada com os componentes **objetivos bem definidos** e **objetivos soft**, nos modelos da técnica i^* .

*Os **objetivos bem definidos** e os **objetivos soft** dos modelos estratégicos serão traduzidos como **atributos do tipo booleano e do tipo enumerado**, respectivamente, em um agente Structured MAL;*

De forma que, os objetivos bem definidos serão mapeados como atributos do tipo booleano

$$F_{ob}: i^* \rightarrow \text{Structured MAL}$$

Objetivos → Atributo: boolean

De forma semelhante, os objetivos-soft são mapeados como atributos do tipo enumerado.

$F_{\text{ob-Soft}}: i^* \rightarrow \text{Structured MAL}$

Objetivos-Soft → Atributo: enum {posit, negat, indef, sub, naosub}

Considerações M4.1 e M4.2: Relacionadas com os componentes **objetivo bem definido** e **objetivo soft**, nos Modelos de Dependências Estratégicas (SD) e Razões Estratégicas (SR), respectivamente.

Já mencionamos que em i^* não existe o conceito de visibilidade de elementos, assim, faremos uso do conceito encontrado em Structured MAL.

Em Structured MAL, o conceito de visibilidade também se aplica aos atributos. De forma geral, podemos dizer que um atributo será de uso local (apenas pode ser utilizado pelo agente no qual se encontra definido), neste caso rotulado com **l** (local) ou será compartilhado com outros agentes e rotulado com **s** (shared), significando que outros agentes podem utilizá-lo.

Desta forma, propomos as subdiretrizes:

M4.1 para os componentes objetivos bem definidos e objetivos-soft encontrados no Modelo de Dependências Estratégicas (SD) onde estes podem ser compartilhado entre os vários atores (agentes) envolvidos; e

M4.2 para esses componentes objetivos e objetivos-soft no Modelo Razões Estratégicas (SR), onde estes são locais ao ator (agente) cujo comportamento se esteja descrevendo.

SubDiretriz M4.1: Relacionada com os componentes **objetivo bem definido** e **objetivo soft**, no Modelo de Dependências Estratégicas (SD).

*Os **objetivos bem definidos** e os **objetivos soft** envolvidos num relacionamento do modelo SD são mapeadas como **atributos do tipo booleano compartilhados** e **atributos do tipo enumerado compartilhados**, respectivamente, pertencentes ao agente (denominado por Yu como “depende”, em i^*), em Structured MAL, que objetiva alcançá-los.*

Assim, se um ator A_1 depende que um ator A_2 satisfaça um objetivo bem definido Ob_1 , então, este objetivo, segundo a diretriz acima, será mapeado como um atributo

compartilhado, (s Atrb₁), que pertence ao agente Ag₁, mapeado de A₁, segundo M1, ou seja, teremos

F_{Ob}: i* → Structured MAL

Ob₁ → (attributes s Atrb₁: boolean) no agentAg₁

Caso tenhamos o ator A₁ dependendo que o ator A₂ satisfaça um objetivo-soft, ObSoft₁, então, este objetivo, segundo a diretriz acima, também será mapeado como um atributo compartilhado, (s Atrb₁), que pertence ao agente Ag₁, ou seja, teremos

F_{ObSof}: i* → Structured MAL

ObSoft₁ → (attributes s Atrb₁: enum {posit, negat, indef, sub, naosub}))

no agentAg₁

Na Figura 5.8, segundo o modelo SD, apresentado na figura 5.1, podemos observar, para o relacionamento entre o agente Cliente e o agente Loja, a ocorrência do que indica a subdiretriz M4.1, com respeito a um objetivo bem definido e a um objetivo soft.

```

agent Cliente
.....
actions
.....
attribute
l Atrbimplementável: enum{verd, falso, indef}
s Ter_CD_Desejado: boolean;

/*Representa o objetivo que tem o Cliente de que a Loja possua o
CD que ele deseja comprar */

s Qualidade[Serviço]: enum{posit, negat, indef, sub, naosub}

/*Representa um objetivo soft que está relacionado com o desejo que
tem o Cliente de que os serviços praticados pela Loja sejam de boa
qualidade */
.....
axioms
.....
end agent Cliente

```

Figura 5.8 – Especificando Objetivos e Objetivos-Soft em i* como Atributos do Tipo Booleano e Enumerado Compartilhados, em Structured MAL

SubDiretriz M4.2: Relacionada com os componente **objetivo bem definido** e **objetivo soft**, no modelo estratégico SR.

Os **objetivos bem definidos** e os **objetivos soft** internos a um ator que se esteja detalhando em i^* , são mapeado, em Structured MAL, como **atributos do tipo booleano locais** e **atributos do tipo enumerado locais**, respectivamente, pertencentes ao ator(agente) que se esteja observando o comportamento interno.

Assim se um a ator A_1 , quando avaliamos seu comportamento interno, tenciona alcançar um objetivo bem definido Ob_1 , então esse objetivo, segundo a diretriz acima, será mapeado como um atributo local do tipo booleano, $Atrb_1$, que pertence ao agente Ag_1 , mapeado de A_1 , segundo M1. Ou seja, teremos

$F_{Ob}: i^* \rightarrow \text{Structured MAL}$

$Ob_1 \rightarrow (\text{attributes } | \text{ } Atrb_1: \text{boolean}) \text{ no agent } Ag_1$

Da mesma forma, em se tratando de um objetivo-soft, se a ator A_1 , quando avaliamos seu comportamento interno, tenciona alcançar um objetivo-soft $ObSoft_1$, então este objetivo, segundo a diretriz acima, será mapeado como um atributo local do tipo enumerado, $Atrb_1$, que pertence ao agente Ag_1 , ou seja, teremos

$F_{ObSof}: i^* \rightarrow \text{Structured MAL}$

$ObSoft_1 \rightarrow (\text{attributes } | \text{ } Atrb_1: \text{enum } \{\text{posit, negat, indef, sub, naosub}\})$

no agent Ag_1

Já na Figura 5.9, segundo o modelo SR, Figura 5.4, podemos observar, para o agente Loja, a ocorrência do que indica a subdiretriz M4.2, com respeito aos objetivos-soft.

```

agent Loja
.....
actions
.....
attributes
  l Atrbimplementável: enum{verd, falso, indef};
  l Preços[Bons] : enum {posit, negat, indef, sub, naosub};

/* Representa o objetivo-soft que tem a Loja de que os preços sejam bons
para atrair cliente*/

  l Volume_Compras [Alto]: enum{posit, negat, indef, sub, naosub};

/* Representa o objetivo-soft que tem a Loja de que o patamar de consumo
do cliente seja alto */

  l Limite_Preço[Baixo] : enum{posit, negat, indef, sub, naosub};

/* Representa o objetivo-soft que tem a Loja de que o limite de preços seja
baixo para atrair */

  l Ter_Qualidade[Atendimento]: enum{posit, negat, indef, sub, naosub};

/*Representa um objetivo soft que está relacionado com o desejo que tem a Loja
de fornecer boa qualidade no atendimento prestado */

.....
axioms
.....
end agent Cliente

```

Figura 5.9 – Especificando Objetivos e Objetivos-Soft em i* como Atributos do Tipo Booleano e Enumerado Locais, em Structured MAL

Passemos a considerar os relacionamentos externos entre os agentes, identificados no modelo SD da técnica i*, Figura 5.1. Esses relacionamentos são tratados pela diretriz M5.

Considerações M5: Sobre os relacionamentos externos entre os atores do i*;

Na técnica i* (vide Capítulo 3), o Modelo de Dependências Estratégicas (SD) representa o comportamento externo entre os atores do sistema, sendo mostrado os vários relacionamentos de dependências (de tarefas, de recursos, de objetivos e de objetivos-soft).

Podemos mapear os vários relacionamentos de dependência entre um ator dependente (“dependor”) e um ator de quem se depende (“dependee”), através de um axioma para cada tipo de relacionamento, em Structured MAL.

Os axiomas, em uma especificação Structured MAL, estabelecem o comportamento dos agentes (objetos). Por exemplo, em se tratando de uma dependência de tarefa, estaremos lidando com ações em Structured MAL, que contarão com os aspectos:

de modalidade ([]), cuja utilidade é a de capturar o efeito produzido pela ocorrência de ações;

dos operadores deônticos **per** (permissão) e **obl** (obrigação), utilizados para definir o fato de ser uma ação permitida ou obrigada a acontecer; e temporais, que estabelecem a duração da ocorrência de uma ação (não tratados neste trabalho).

Estaremos acrescentando informações adicionais ao relacionamento de dependência. Teremos também que considerar se a tarefa será implementável ou não, nos mesmos moldes mencionados nas considerações anteriores.

Em se tratando dos objetivos bem definidos, que serão representados por atributos do tipo booleano, o valor verdadeiro não precisa ser explicitado quando de seu aparecimento nos axiomas, pois um axioma já representa uma tautologia. Contudo, com relação aos objetivos-soft, representados por atributos do tipo enumerado, os valores dos atributos precisam ser avaliados, sendo necessário explicitar esta avaliação, atribuindo-se um valor a esses atributos nos axiomas em que apareçam.

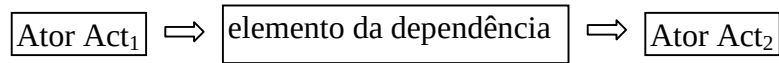
Em se tratando de recurso, este será mapeado como um agente e considerado o atributo booleano que representa sua disponibilidade.

De modo geral, o mapeamento não deve ser apenas uma tradução sintática. Casos existem em que será preciso incluir elementos que não estão representados nos Modelos Estratégicos, como meio de melhor precisar o comportamento de um agente ou torná-lo implementável. Este enriquecimento dificultará o mapeamento reverso, no sentido de que não recuperaríamos o mesmo modelo caso caminhássemos no sentido contrário.

Diretriz M5: Trata dos relacionamentos entre os atores do i*;

Cada um dos **relacionamentos** no Modelo de Dependências Estratégicos (SD) da técnica i^* pode ser tratado por **um axioma**, em Structured MAL, no agente do qual se depende.

Para a determinação do agente do qual se depende, consideraremos a direção das setas no Modelo de Dependências Estratégicos (SD). Normalmente, o sentido do relacionamento é dado pelo sentido das setas encontradas no relacionamento entre os atores. Por exemplo, se temos



então, em função do elemento da dependência, tem-se o seguinte:

- (i) Se o elemento é uma tarefa T_c , mapeada em uma ação Ac , e o ator Act_1 depende que o ator Act_2 execute esta tarefa, então no correspondente agente (ator) de quem se depende, Ag_2 em Structured MAL, ter-se-á o axioma

$$[Ac](\text{pós-cond})$$

significando que a ocorrência de Ac determinará um resultado. Por exemplo, na figura 5.1, vemos que o ator Apoio depende que o ator Fornecedor realizem uma tarefa (“Enviar CD Solicitado”). Na especificação em Structured MAL, do agente Fornecedor, teremos um axioma

$$[\text{Enviar_CD_Solicitado}]\text{pós-condição}$$

onde esta pós-condição é dependente do restante da especificação;

- (ii) Se o elemento é um recurso Re , mapeado como um agente Re em Structured MAL com o respectivo atributo correspondente $Atrb_{Re}$, e o ator Act_1 depende que o ator Act_2 forneça este recurso, então no corpo da especificação do agente Ag_2 haverá o seguinte axioma:

$$\text{pre-cond} \rightarrow Atrb_{Re}$$

Por exemplo, na figura 5.1, podemos ver uma dependência de recurso entre o agente Loja e o agente Cliente. Este último, espera que o agente Loja tenha disponível o “CD Solicitado”. Assim, um dos axiomas do agente Loja, em Structured MAL, será

$$\text{pre-cond} \rightarrow \text{CD_Solic}$$

onde a pre-cond, dependerá do restante da especificação, por exemplo, o Cliente realizar a ação de “Levar CD” (não expressa no modelo da figura 5.1), conforme pode ser visto no axioma 2 da figura 5.10;

- (iii) Se o elemento é um objetivo bem definido, mapeado como atributo booleano (Atrb_{obj}) em Structured MAL, e o ator Act_1 depende do ator Act_2 para alcançá-los, então no corpo da especificação do agente Ag_2 haverá o seguinte axioma:

$$\text{pre-cond} \rightarrow \text{Atrb}_{\text{obj}}$$

Na figura 5.1, identificamos entre os atores Loja e Cliente uma dependência do tipo objetivo bem definido. Por exemplo, a Loja espera ter satisfeito o objetivo “Compre CD’s”. Um trecho de axioma, em Structured MAL do agente Cliente, seria

$$\text{pre-cond} \rightarrow \text{Compre_CD's}$$

onde a pre-cond, depende do restante da especificação.

- (iv) Se o elemento é um objetivo-soft, mapeado como atributo do tipo enumerado ($\text{Atrb}_{\text{objSoft}}$) em Structured MAL, e o ator Act_1 , depende do ator Act_2 para alcançá-los, então no corpo da especificação do agente Ag_2 haverá o seguinte axioma:

$$\text{pre-cond} \rightarrow \text{Atrb}_{\text{objSoft}}$$

Na figura 5.1, identificamos entre os atores Loja e Cliente uma dependência do tipo objetivo-soft. Por exemplo, o Cliente espera que a Loja tenha “Qualidade[Serviço]”. Este objetivo, pode ser expresso pelo axioma 3, do trecho da especificação do agente Loja da figura 5.10,

$$\text{pre-cond} \rightarrow \text{Qualidade[Serviço]} = \text{posit}$$

Neste caso, como se trata de um objetivo-soft, associamos um atributo do tipo enumerado, cujo valor deve ser avaliado. No caso, atribuímos o valor positivo, para dizermos que a qualidade do serviço é boa.

De forma geral, tem-se

$$F_{\text{Relac (SD)}} : i^* \rightarrow \text{Structured MAL}$$

Relacionamentos entre Atores $\rightarrow$ Axiomas Específico no Agente de
quem se Depende

Na Figura 5.10, é apresentado um fragmento de axiomas, relativo ao comportamento do agente Loja com relação ao Cliente. Observe que o agente Loja estará compartilhando uma ação, “Comprar CD”, que deve ser realizada pelo agente Cliente, e que, portanto, deve ser detalhada no corpo da especificação, em Structured MAL, do agente Cliente. Todavia, esta tarefa ainda pode não estar descrita nos modelos estratégicos em i^* , pois ainda estamos no estágio inicial dos requisitos. Outro fato que também devemos observar, é a inclusão de um atributo estoque, que representa a quantidade de CD’s existentes na Loja. Esse atributo teve que ser acrescido, para que pudéssemos, concretamente, retratar o relacionamento entre os atores.

Novamente, esta informação ainda não se encontra expressa, no Modelo de Dependências Estratégicas i^* , não significando que não possamos acrescentá-la, também, ao modelo, como recurso. O importante a notar é que essa inclusão dificulta, ao final da especificação precisa em Structured MAL, recuperar totalmente os modelos estratégicos em i^* , na sua forma original. Por esta razão estamos neste trabalho apenas tratando do mapeamento $i^* \rightarrow \text{Structured MAL}$ e não o contrário.

```

agent Loja
    .....
    includes
    .....
    attribute
    I Atrbimplementável: enum{verd, falso, indef};
    I Preços[Bons] : enum{posit, negat, indef, sub, naosub};
    /* Representa o objetivo-soft */
    I Volume_Compras [Alto]: enum{posit, negat, indef, sub, naosub};
    /* Representa o objetivo-soft */
    I Limite_Preço[Baixo] : enum{posit, negat, indef, sub, naosub};
    /* Representa o objetivo-soft */
    I Ter_Qualidade[Atendimento]: enum{posit, negat, indef, sub, naosub};
    s Compre_CD: boolean;

    I estoque : nat;
    /*Representa a quantidade de CD's residentes no estoque da Loja*/
    .....
    axioms
    1.    [Levar_CD] ((estoque = estoque -1)  $\wedge$  Compre_CD);
    /*Significando que o objetivo bem definido da loja será verdadeiro, dado que
    o Cliente leve o CD, comportamento interno do agente Cliente não expresso
    aqui.*/
    2.    Levar_CD  $\rightarrow$  CD_Solic;
    3.    Ter_Qualidade[Atendimento] = posit  $\rightarrow$ 
                                                Qualidade[Serviço]

```

Figura 5.10 – Um Relacionamento Externo Representado em Structured MAL através de Axiomas

No tocante ao Modelo de Razões Estratégicas (SR), segundo o que foi abordado no Capítulo 3, identificamos o comportamento interno dos atores, no exemplo da Figura 5.4, em que podem ser vistos três atores: “Loja”, “Atendente” e “Apoio”. De modo geral, neste modelo SR são encontrados os mesmos elementos componentes das dependências (tarefa, recuso, objetivo e objetivo-soft) e dois outros relacionamentos adicionais: do tipo decomposição de tarefa e do tipo meio-fim.

Para representar esses dois novos tipos de relacionamento, propomos a diretriz M6 e M7.

Considerações M6: Relacionadas com as ligações do tipo decomposição de tarefa no modelo SR

Conforme mencionado anteriormente, trata-se de uma ligação encontrada no Modelo de Razões Estratégicas (SR) que representa o comportamento interno de cada um dos atores. Os *relacionamentos de decomposição de tarefa* exprimem o que deve ser feito para se ter a tarefa realizada, podendo a tarefa ser decomposta nos quatro tipos:

- (a) *sub-objetivo*,
- (b) *sub-tarefa*;
- (c) *recursoPara*; ou
- (d) *objetivo-softPara*.

Essas situações serão representadas em Structured MAL pelos axiomas dos agentes. Devemos considerar, novamente, o fato de que nem todos os elementos são implementáveis. Nesta fase do desenvolvimento, não estamos preocupados com este aspecto.

No mapeamento para Structured MAL, dos elementos de dependência existentes em i^* , no tocante à sub-tarefas (subações) serão utilizados operadores deônticos de permissão (per) ou obrigação (obl), para melhor precisar a realização da subação. Trata-se de informação adicional, pois em i^* , não se tem modelado esse aspecto. Para refletir a conjunção de elementos no qual uma tarefa pode ser decomposta, em i^* , far-se-á uso do conectivo lógico AND, em Structured MAL.

Destacamos ainda que a lógica temporal encontrada em Structured MAL em associação com a parte deôntica, poderá dar a ordem de ocorrência dos elementos na ligação de decomposição de tarefa, em i^* . Por exemplo, na Figura 5.4, observando um pequeno fragmento do agente Loja, identificamos que a tarefa “raiz”, “*Fazer Clientela*”, é decomposta em uma sub-tarefa, “*Atrair Cliente*”, e em um objetivo-softPara, que diz respeito a “*Ter qualidade de atendimento*”, que representam as folhas dessa raiz. Pelo Modelo de Razões Estratégicas não se tem informação do que seja preciso primeiro. Por exemplo: “*Atrair Cliente*” para depois se “*Ter qualidade de atendimento*”, ou “*Ter qualidade de atendimento*” para depois “*Atrair Cliente*”. Na modelagem i^* essa interpretação é deixada a cargo de quem esteja descrevendo o sistema. É proposta de trabalho futuro (vide Capítulo 7) utilizarmos-nos do componente Deôntico e do aspecto

Temporal encontrado em Structured MAL, para atribuir uma ordem, nos casos que sejam necessários.

Diretriz M6: Relacionada com as ligações do tipo decomposição de tarefa no Modelo de Razões Estratégicas (SR);

*Uma ligação do tipo **decomposição de tarefa** é representada, em Structured MAL, através de um **axioma de ação** contendo os elementos da decomposição (sub-objetivo, sub-tarefa, recursoPara ou objetivo-softPara) interligados pelo conectivo lógico **AND**, aplicando-se o operador deôntico **obl às subações** (subtarefas) que possam existir, conforme seja a decomposição da tarefa.*

Conforme afirmamos anteriormente, os axiomas ditarão o comportamento dos agentes. Neste caso específico, estamos tratando de uma ação de decomposição. A sintaxe de um axioma de ação em Structured MAL (Seção 4.4.1.2) é dada por

[ação]pós-condição

na qual, segundo a linguagem, uma pós-condição pode ser uma fórmula, que pode ser fórmula atômica ou uma fórmula com termos, ligadas pelos conectivos lógicos (Seção 4.4.1.2). Assim, se em Structured MAL a ação Ac (correspondente ao ator At , em i^*) é decomposta em uma sub-ação Ac_1 , uma sub-objetivo Ob_1 , um recurso Re_1 e um objetivo-soft $ObSoft_1$, então, segundo a diretriz acima, tem-se:

$$\text{axiom } [Ac] (\text{obl}(Ac_1) \wedge (\text{atbr}_{Re_1}) \wedge (\text{Atrb}_{Ob_1}) \wedge (\text{Atrb}_{ObSoft_1} = \text{'valor'}))$$

onde **obl(Ac_1)** corresponde a obrigação da realização de uma sub-ação da ação em avaliação, **atbr_{Re1}** corresponde ao atributo compartilhado do agente recurso Re_1 ; **Atrb_{obj1}** corresponde ao atributo local, do próprio agente, correspondente a um objetivo bem definido; e **Atrb_{objSoft1}** corresponde ao atributo local, do próprio agente, correspondente a um objetivo-soft.

No exemplo da Figura 5.4, observando-se o comportamento interno do ator Loja, identificamos um relacionamento de decomposição de Tarefa com relação a “Fazer

Clientela” e com relação a “Atrair Cliente”. Especificamente com relação à tarefa “Fazer Clientela”, tem-se um objetivo-soft “Ter Qualidade[Atendimento]” significando uma necessidade, e uma sub-tarefa “Atrair Cliente”. Essa última tarefa será decomposta em um objetivo-soft “Preços[Bons]” e duas outras tarefas: “Manter CD em Estoque” e “Fornecer Serviços Especiais”. Assim, na Figura 5.11, podemos observar a decomposição da tarefa “Fazer Clientela”, conforme nos indica a diretriz M6, onde o mapeamento ditado por esta diretriz encontra-se especificado no axioma 4.

```

Agent Loja
...
includes
actions
    l Fazer_Clientela;
    l Atrair_Cliente;
attributes
    l Atribimplementável: enum{verd, falso, indef};
    l Preços[Bons]: enum{posit, negat, indef, sub, naosub};
    l Volume_Compras [Alto]: enum{posit, negat, indef, sub, naosub};
    l Limite_Preço[Baixo] : enum{posit, negat, indef, sub, naosub};
    l Ter_Qualidade[Atendimento]: enum{posit, negat, indef, sub, naosub};
    s Compre_CD: boolean;
    l estoque : nat;
axioms
    1. [Levar_CD] (estoque = estoque -1)  $\wedge$  Vender CD;
    /*Significando que o objetivo bem definido da loja (Vender CD) verdadeiro,
    dado que o Cliente realizou a tarefa de comprar o CD e que foi dado baixa no
    estoque */
    2. Levar_CD  $\rightarrow$  CD_Solic;
    3. Ter_Qualidade[Atendimento] = posit  $\rightarrow$ 
        Qualidade[Serviço] = posit;
    4. [Fazer_Clientela] (obl (Atrair_Cliente)  $\wedge$ 
        (Ter_Qualidade[Atendimento] = posit));
    /*Indica que a tarefa será uma conjunção de outros elementos. Representa um
    refinamento da tarefa Fazer_Clientela em realizar a ação de atrair cliente
    (obl(Atrair_Cliente)), mantendo uma boa qualidade no atendimento (Ter-
    Qualidade[Atendimento] = posit)*/
end Agent Loja

```

Figura 5.11 – As Ligações de Decomposição de Tarefa em i*,
Especificadas em Structured MAL através de Axiomas
com Conectivos AND e Operadores Deônticos

De forma similar, no Modelo de Razões Estratégicas (SR), Figura 5.4, podemos identificar outros relacionamentos de decomposição de tarefa para o mesmo agente Loja, todavia, o procedimento é o mesmo.

Considerações M7: Relacionadas às ligações do tipo meio-fim no Modelo de Razões Estratégicas (SR).

Em i^* , os *relacionamentos meio-fim* indicam um relacionamento entre um *fim* que pode ser:

- (a) um objetivo a ser alcançado;
- (b) uma tarefa a ser realizada;
- (c) um recurso a ser produzido; ou
- (d) um objetivo soft a ser satisfeito

e um meio, para se atender a esses fins. Neste tipo de relacionamento, o meio para se alcançar um fim sempre é uma tarefa. Exceto em duas situações, conforme pode ser visto no Capítulo 3, onde o fim e o meio são ambos, objetivos ou objetivos-soft. Em geral, um relacionamento meio-fim sugerem a existência de meios para se alcançar o mesmo fim, exprimindo as alternativas existentes.

Em Structured MAL, esta situação será expressa nos axiomas, através da disjunção (utilização do conectivo lógico OR) dos meios que se disponha para se alcançar um fim.

No modelo da Figura 5.4, por exemplo, com relação ao ator Loja, tem-se por exemplo, dois relacionamentos meio-fim:

- (i) envolvendo a tarefa “Fornecer Serviços Especiais”, onde identificamos duas alternativas: “Fornecer Vendas por Telefone” ou “Fornecer Vendas sob Encomenda”;
- (ii) envolvendo o objetivo-soft “ Preços[Bons], onde identificamos duas possibilidades: “Volume_Compra[Alto]” ou “Limete_Preço[Baixo]”.

Diretriz M7: Relacionada às ligações do tipo meio-fim no Modelo de Razões Estratégicas (SR);

Uma ligação do tipo **meio-fim** é representada, em Structured MAL, pela disjunção dos elementos meio utilizando o conectivo lógico OR, expressa através de **axiomas**.

Em se tratando da ligação meio-fim, temos que considerar os tipos existentes, conforme Figura 3.4b, Capítulo 3. De forma geral, esse tipo de ligação é expressa através dos axiomas de Structured MAL que podem ser de dois tipos:

- (i) considerando-se que o meio é uma tarefa e o fim uma tarefa, objetivo bem definido, objetivo-soft ou recurso; e
- (ii) considerando-se a situação em que o meio é um objetivo bem definido ou objetivo-soft, e o fim também é um objetivo bem definido ou objetivo-soft.

Assim, para cada uma das possíveis ligações meio-fim existentes, propomos as subdiretrizes que se seguem.

Considerações M7.1: Relacionadas com ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é tarefa, objetivo, objetivo-soft ou recurso.

Relembramos que os elementos meio e fim são mapeados segundo as diretrizes já tratadas, valendo todas as considerações realizadas.

Para exprimir as alternativas existentes, nesse tipo de ligação, utilizaremos os axiomas em Structured MAL. Neste caso, em específico, utilizaremos o axioma de ação, cuja sintaxe é dada por

[ação]pós-condição

onde, segundo a linguagem, uma pós-condição pode ser uma fórmula, que pode ser tanto atômica ou ser uma outra fórmula com termos, ligadas pelos conectivos lógicos (Seção 4.4.1.2).

SubDiretriz M7.1: Relacionada às ligações do tipo meio-fim, onde o meio é uma tarefa e o fim pode ser tarefa, objetivo bem definido, objetivo-soft ou recurso.

Uma ligação do tipo meio-fim, onde o meio é uma tarefa, é representada, em Structured MAL, através de um axioma de ação contendo o conectivo lógico OR e o operador deôntico obl aplicado às ações que possam existir, em função de poder ser o fim uma tarefa, um objetivo bem definido, um objetivo-soft ou um recurso.

Assim, se na ligação meio-fim temos:

- (i) um objetivo bem definido como um fim, mapeado para Structured MAL, segundo D.4, $Atrb_{obj}$, e como meio tenhamos n tarefas, mapeadas para Structured MAL, segundo D.2 ($Ac_1, Ac_2, ..., Ac_n$), então teremos

$$\text{axiom } [Ac_1] Atrb_{obj} \vee [Ac_2] Atrb_{obj} \vee ... [Ac_n] Atrb_{obj}$$

onde o axioma nos diz que o fim $Atrb_{obj}$ será alcançado, ocorrendo pelo menos uma das ações Ac_i ;

- (ii) um objetivo-soft como um fim, mapeado para Structured MAL, segundo D.4, $Atrb_{objSoft}$, e como meio tenhamos n tarefas, mapeadas para Structured MAL, segundo D.2 ($Ac_1, Ac_2, ..., Ac_n$), então teremos

$$\text{axiom } [Ac_1] (Atrb_{objSoft} = \text{'valor}_1\text{'}) \vee [Ac_2] (Atrb_{objSoft} = \text{'valor}_2\text{'}) \vee ... \vee [Ac_n] (Atrb_{objSoft} = \text{'valor}_n\text{'})$$

onde o axioma nos diz que o fim $Atrb_{objSoft}$ será alcançado, ocorrendo pelo menos uma das ações Ac_i ;

- (iii) um recurso como um fim, mapeado para Structured MAL, segundo D.3, em um agente Re , onde consideramos o atributo desse agente, $atbr_{rec}$, e como meio tenhamos n tarefas, mapeadas para Structured MAL, segundo D.2 ($Ac_1, Ac_2, ..., Ac_n$), então teremos

$$\text{axiom } [Ac_1] \text{ } atrb_{rec} \vee [Ac_2] \text{ } atrb_{rec} \vee \dots [Ac_n] \text{ } atrb_{rec}$$

onde o axioma nos diz que o fim $Atrb_{rec}$ será alcançado, ocorrendo pelo menos uma das ações Ac_i ;

- (iv) uma tarefa como um fim, mapeada para Structured MAL, segundo M2, em uma ação, Ac , e como meio tenhamos n tarefas, mapeadas para Structured MAL, segundo M2 ($Ac_1, Ac_2, \dots, Ac_n$), então teremos

$$\text{axiom } [Ac] (\text{obl}(Ac_1) \vee \text{obl}(Ac_2) \vee \dots \text{obl}(Ac_n))$$

onde o axioma nos diz que o fim Ac será alcançado, tendo-se como obrigação a ocorrência de pelo menos uma das ações Ac_i ; Com este axioma, procuramos mostrar que existem n alternativas para se executar uma ação.

- (v) vários fins tais com, uma tarefa mapeada como uma ação Ac_1 , um objetivo bem definido mapeado como um atributo $Atrb_{obj}$, um recurso mapeado como um atributo de uma classe $Atrb_{rec}$, e um objetivo-soft mapeado como $Atrb_{objSoft}$, segundo as diretrizes de M2 a M4 e como meio uma tarefa, segundo M2, mapeada em uma ação, Ac , então teremos

$$\text{axiom } [Ac] (\text{obl}(Ac_1) \wedge Atrb_{obj} \wedge Atrb_{rec} \wedge (Atrb_{objSoft} = \text{'valor'}))$$

Considerações M7.2: Relacionadas com as ligações do tipo meio-fim, onde o meio é um objetivo (ou um objetivo-soft) e o fim é um objetivo bem definido ou um objetivo-soft.

Neste caso, os elementos meio e fim são do mesmo tipo. Individualmente, cada um dos elementos será mapeado conforme as diretrizes propostas e consideradas anteriormente. Especificamente utilizaremos o axioma com nenhuma ação (non action axiom), cuja sintaxe em Structured MAL (Seção 4.4.1.2), é dada por

$$\text{pré-condição} \rightarrow \text{pós-condição}$$

Destacamos apenas que embora estes tipos de ligações meio-fim não sejam freqüentes, estão previstos na técnica i*.

SubDiretriz M7.2: Relacionada às ligações do tipo meio-fim, onde o meio é um objetivo (ou um objetivo-soft) e o fim é um objetivo bem definido ou um objetivo-soft..

*Uma ligação do tipo **meio-fim**, onde o **meio** é **objetivo** (ou **objetivo-soft**) e o **fim** é um **objetivo** ou **objetivo-soft**, é representada, em Structured MAL, através de um **axioma de não ação** (no-action-axiom) e utilizando-se o conectivo **OR**.*

Como os objetivos são definidos como atributos booleanos, locais ou compartilhados, conforme M4, então se considerarmos como fim o objetivo **Ob** (**Atrb_{obj}**) e n objetivos (**Atrb_{obj1}**, **Atrb_{obj2}**, ... , **Atrb_{objn}**) como meio, teremos

$$\text{axiom} (\text{Atrb}_{\text{obj1}} \vee \text{Atrb}_{\text{obj2}} \vee \dots \vee \text{Atrb}_{\text{objn}}) \rightarrow \text{Atrb}_{\text{obj}}$$

Da mesma forma, em se tratando de um objetivo-soft, ou seja, se considerarmos como fim o objetivo **ObSoft** (**Atrb_{objSoft}**) e n objetivos (**Atrb_{objSoft1}**, **Atrb_{objSoft2}**, ... , **Atrb_{objSoftn}**) como meio, teremos

$$\begin{aligned} \text{axiom} ((\text{Atrb}_{\text{objSoft1}} = \text{'valor}_1') \vee (\text{Atrb}_{\text{objSoft2}} = \text{'valor}_2') \vee \dots \\ \vee \text{Atrb}_{\text{objSoftn}} = \text{'valor}_n')) \rightarrow (\text{Atrb}_{\text{objSoft}} = \text{'valor'}) \end{aligned}$$

Como exemplo do relacionamento meio-fim, observe o Modelo de Razões Estratégicas para a Loja de Venda de CD's, Figura 5.4, onde identificamos que, no agente Loja, a tarefa de “*Fornecer_serviços_especiais*” pode ser realizada de duas maneiras: ou a loja “*Fornece_vendas_por_telefone*” ou “*Fornece_vendas_sob_encomenda*”. Assim, essa última opção, influenciará positivamente para que seja atendido o objetivo soft do Cliente (Figura 5.1), que requer que a Loja tenha qualidade de serviços. Na Figura 5.12 podemos observar em MAL, para o agente Loja, que o axioma 5 está conforme nos indicam a subdiretriz M7.1, item (iv), para a representação das duas

alternativas para se obter o fim, neste caso representado no Modelo de Razões Estratégicas, Figura 5.4, pela tarefa *Fornecer_serviços_especiais*.

```

Agent Loja
includes
actions
    l Fazer_Clientela;
    l Atrair_Cliente;
    l Fornecer_serviços_especiais;
    l Fornecer_vendas_por_telefone;
    l Fornecer_vendas_sob_encomenda;
attributes
    l Atribimplementável: enum{verd, falso, indef};
    l Preços[Bons] : enum{posit, negat, indef, sub, naosub};
    l Volume_Compras [Alto]: enum{posit, negat, indef, sub, naosub};
    l Limite_Preço[Baixo] : enum{posit, negat, indef, sub, naosub};
    l Ter_Qualidade[Atendimento]: enum{posit, negat, indef, sub, naosub};
    s Compre_CD: boolean;
    l estoque : nat;
axioms
    1. [Levar_CD] (estoque = estoque -1)  $\wedge$  Vender CD;
    2. Levar_CD  $\rightarrow$  CD_Solic;
    3. Ter_Qualidade[Atendimento] = posit  $\rightarrow$  Qualidade[Serviço] = posit;
    4. [Fazer_Clientela] (obl (Atrair_Cliente)  $\wedge$ 
        (Ter_Qualidade[Atendimento] = posit);
    5. [Fornecer_serviços_especiais] (obl (Fornecer_vendas_por_telefone)  $\vee$ 
        obl (Fornecer_vendas_sob_encomenda));
    /*A loja poderá fornecer serviços especiais de duas maneiras: fornecendo vendas por
    telefone (obl (Fornecer_vendas_por_telefone)) ou fornecendo vendas por encomenda (obl
    (Fornecer_vendas_sob_encomenda)). Este axioma representa a aplicação da diretriz M7.1.*/
    6. (Limite_Preço[Baixo] = posit)  $\vee$  (Volume_Compras [Alto] = posit)  $\rightarrow$ 
        Preços[Bons] = posit;
    /* Se a loja dispõe de limites de preço baixo (Limite_Preço[Baixo] = posit) e se o volume
    de compras é alto (Volume_Compras [Alto] = posit), então terá (Preços[Bons] = posit). Este
    axioma representa a aplicação da diretriz M7.2.*/

end Agent Loja

```

Figura 5.12 – As Ligações Meio-Fim em i*, especificadas em MAL Estruturado através de Axiomas com Conectivo OR e Operadores Deônticos

Como exemplo de um relacionamento meio-fim, em que o meio e o fim são objetivos-soft, considere no modelo SR da loja de CD's da Figura 5.4, o relacionamento que envolve: “Preços [Bons]”, “Volume_Compra [Alto]” e “Limite_Preço [Baixo]”. O

axioma 6 da Figura 5.12 representa a aplicação da subdiretriz M7.2, em um trecho de especificação em Structured MAL para o agente Loja.

Toda essas diretrizes (M1 a M7) serão apropriadas para a definição de um esboço de uma especificação formal dos requisitos organizacionais que, em conjunto com a especificação formal, já existente, dos requisitos funcionais do sistema desejado, tornaria mais completa a descrição do problema. Obviamente que, quando necessário (como aconteceu com a inclusão de um atributo adicional, por ocasião da aplicação da diretriz M5, Figura 5.10) serão inseridos à especificação maiores detalhes ou mesmo impostas restrições, não encontradas nos Modelos Estratégicos i^* , a fim de que se obtenha uma melhor precisão.

Dispondo-se agora de especificações precisas é possível se fazer provas de propriedades, através dos axiomas apresentados, verificar se o que foi especificado é realmente o que foi elicitado, e ter a possibilidade de animação das especificações. Tudo isso objetivando fazer com que as ambigüidades e inconsistências sejam eliminadas do documento de requisitos.

Voltamos a salientar que estamos tratando apenas com um núcleo de construtores da técnica i^* , não abordando, por exemplo as questões de força das dependências, nem da especialização dos atores em Agentes, Papéis e Posições; por serem conceitos que ainda estão sendo trabalhos pelos autores do i^* .

De outra forma, só tratamos do mapeamento em um sentido, da modelagem organizacional em i^* , para especificações precisas em Structured MAL, porque o sentido inverso significa retirarmos informações que foram acrescentadas a fim de chegar-se a implementação do sistema que se esteja especificado. Assim, resolvemos tratar esse aspecto como extensão desse trabalho (vide Seção 7.4).

Novamente, ressaltamos que, como técnica de modelagem organizacional, poderia ser utilizada qualquer daquelas apresentadas no Capítulo 3; precisando-se, conforme os elementos utilizados pela técnica, pode-se fazer pequenas adaptações às diretrizes de mapeamento, de acordo com a sintaxe do modelo.

Na Seção seguinte estaremos propondo diretrizes para o mapeamento dos requisitos organizacionais, expressos segundo a técnica i^* , para descrições especificadas na linguagem OCL (Object Constraints Language) e que faz parte do padrão de modelagem orientado a objeto, UML [BOO99, JACO99, RUM99].

5.2.2 – DIRETRIZES DE MAPEAMENTO DE i^* PARA UML/OCL

Um dos objetivos de se integrar descrições organizacionais expressas em i^* , com descrições em UML/OCL será a busca de precisão das especificações de requisitos.

Inicialmente lembramos a natureza orientada a objeto da técnica i^* [MYL99]. Como a linguagem de restrições a objetos se aplica ao padrão de modelagem objeto UML, tentaremos, da mesma forma que procedemos para MAL, procurar estabelecer diretrizes que orientem no mapeamento de i^* para um núcleo de UML (pUML ou Precise UML [EVA99]), onde serão descritas restrições na linguagem OCL. Portanto, estaremos caminhando no sentido $i^* \rightarrow \text{pUML} \rightarrow \text{OCL}$. Devemos lembrar que a especificação das restrições é feita a nível conceitual, onde os aspectos de implementação ainda não são considerados. A partir de descrições em pUML poderemos usar OCL para:

- (vi) para especificar invariantes de classes e tipos, no modelo de classes;
- (vii) para descrever pré e pós condições de métodos; e
- (viii) como uma linguagem de navegação.

Como linguagem de navegação, irá permitir que sejam escritas expressões de restrições (invariantes, pré e pós condições) ao comportamento das classe identificadas. De forma similar ao que foi feito com relação a Structured MAL, na Seção 5.2.1, propomos algumas diretrizes de mapeamento. Em relação ao que foi apresentado para Structured MAL, não cobriremos o mapeamento das dependências externas entre os atores (diretriz M5), pois estaríamos apenas fazendo uma tradução sintática para pUML, dado que as restrições em OCL são aplicadas às classes ou às operações de uma classe, não permitindo a atribuição de qualquer precisão adicional. As diretrizes propostas em pUML/OCL são:

Diretriz Oc1: Relacionada com o mapeamento dos atores do modelo i*;

SubDiretriz Oc1.1 : Relacionada com o mapeamento da composição de vários atores;

Diretriz Oc2: Relacionada com o mapeamento das tarefas dos modelos i*;

SubDiretriz Oc2.1: Relacionada com o mapeamento do componente tarefa no Modelo de Dependências Estratégicas (SD);

SubDiretriz Oc2.2: Relacionada com o mapeamento do componente tarefa no Modelo de Razões Estratégicas (SD);

Diretriz Oc3: Relacionada com o mapeamento dos recursos dos modelos i*;

Diretriz Oc4: Relacionada com o mapeamento dos objetivos bem definidos e objetivos- soft dos modelos i*;

SubDiretriz Oc4.1: Relacionada com o mapeamento dos objetivos bem definidos e objetivos- soft no Modelo de Dependências Estratégicas (SD);

SubDiretriz Oc4.2: Relacionada com o mapeamento dos objetivos bem definidos e objetivos- soft no Modelo de Razões Estratégicas (SR);

Diretriz Oc5: Relacionada com o mapeamento dos relacionamentos de decomposição de tarefa nos modelos i*;

Diretriz Oc6: Relacionada com as ligações do tipo meio-fim no Modelo de Razões Estratégicas (SR);

SubDiretriz Oc6.1: Relacionada com ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é uma tarefa.;

SubDiretriz Oc6.2: Relacionada com ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é um objetivo, um objetivo-soft ou um recurso;

SubDiretriz Oc6.3: Relacionada com a expressão dos relacionamentos meio-fim dos modelos i*, quando o meio e fim são, simultaneamente, objetivos bem definidos ou objetivos-soft;

Considerando os elementos componentes do Modelo de Dependências Estratégicas (SD), Figura 5.1, inicialmente trabalhamos com as diretrizes que mapeam esses elementos em pUML, diretrizes Oc1 a Oc4, dado que esses mesmos elementos irão compor o Modelo de Razões Estratégicas (SR), Figura 5.4. Sempre que preciso, faremos algumas considerações iniciais, antes de cada diretriz proposta.

Considerações Oc1: Relacionadas com o mapeamento dos atores do modelo i*.

Como vimos anteriormente, a nível de i* estamos ainda nos primeiros estágios da modelagem, no Processo da Engenharia de Requisitos e **nem** todos os atores **são implementáveis** sob o ponto de vista de uma implementação computacional.

Em pUML (Precise Unified Modeling Language), tem-se o conceito de objeto e classe, advindos da orientação a objeto. Por *objeto* entende-se uma unidade real ou abstrata, com identidade e estado, que modela um conceito da realidade e que ocupa espaço físico ou lógico. Enquanto a *classe* corresponde a uma representação de um conjunto de coisas reais ou abstratas (objetos) que são do mesmo tipo, por compartilharem as mesmas características de atributos, operações, relações e semânticas. Uma classe pode ser a descrição de um objeto em qualquer tipo de sistema, desde sistemas técnicos a de negócios. De modo que segundo[FUR98], sob uma perspectiva cognitiva, uma classe pode ser:

- (i) algo tangível e/ou visível, por exemplo, pessoas, equipamentos e construções;
- (ii) coisas intangíveis e/ou simbólicas, tais como, qualidade, planejamento e responsabilidades;
- (iii) alguma coisa que possa ser concebida intelectualmente à qual dedica-se ação ou pensamento; e
- (iv) eventos temporais ou externos.

Atores em i* e Classes em pUML poderão (ou não) serem implementáveis do ponto de vista computacional. Como na consideração M1, também sugerimos a inclusão na Classe de um atributo local do tipo enumerado `Atrbimplementável`, para definir esta

característica. Os valores possíveis para o atributo são {verd, falso, indef}. Em pUML, os atores considerados como não implementáveis, deverão ser considerados como atores no diagrama de Casos de Uso (Use Cases)

Caso, seja necessário futuramente **acessar** informações referentes aos Atores humanos estes poderão ser modelados como agentes implementáveis. Assim, quando do mapeamento para pUML, o atributo $Atrb_{implementável}$ assume o valor verdadeiro (verd).

Outro aspecto que somente estará presente nas descrições mais precisas em pUML são as restrições de OCL (Object Constraint Language):

- (i) *invariantes* sobre os atributos da classe, definindo as propriedades dos atributos que devem ser mantidas em todas as instâncias da Classe; e
- (ii) *pré e pós-condições* sobre as operações da classe, ou seja, significando as condições que habilitam a execução da operação e as condições previstas após a execução da mesma.

Com isso há um enriquecimento na especificação, pois os atores antes modelados em i^* passarão a ter restrições que melhor definirão seus relacionamentos com outros atores e seu comportamentos. A partir dessas propomos mapear os atores em i^* segundo a diretriz que se segue.

Diretriz Oc1: Relacionada com o mapeamento dos atores do modelo i^* .

*Os **Atores**, nos modelos estratégicos da técnica i^* , podem ser entendidos como **Classes** em pUML, submetidas às restrições em OCL.*

A existência do construtor gráfico , que representa um ator nos modelos estratégicos de i^* , que será mapeado como uma Classe em pUML, semelhante a diretriz M1. Esse mapeamento é direto. Neste trabalho, não estaremos considerando as possíveis “especializações” dos atores (Agentes, Posições e Papéis) por serem ainda motivo de melhor definição, conforme já discutido. Desta forma, o mapeamento seria

$$G_{Act} : i^* \rightarrow pUML$$
$$Ator \rightarrow Class \text{ nome-ator}$$

Mais adiante, a medida em que formos mapeando os elementos dos modelos estratégicos i*, poderemos falar melhor sobre estas restrições a serem descritas em OCL.

Considerando o modelo de dependências estratégicas (SD), Figura 5.1, identificamos seis atores - Loja, Cliente, Fornecedor, Atendente, Caixa e Apoio –, os quais serão representados como classes em pUML, cujas estruturas são apresentadas na Figura 5.13.

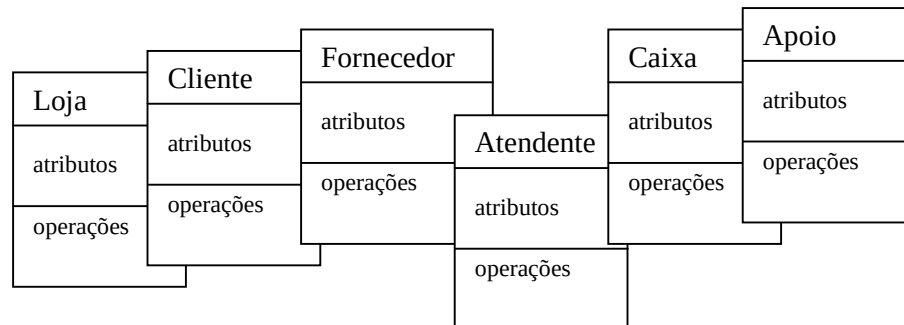


Figura 5.13 – Classes em pUML correspondentes aos Atores em i* do Modelo SD (Sistema Empresa de Venda de CD)

Considerações Oc1.1: Relacionadas com o mapeamento da composição de vários atores.

Da mesma forma que em Structured MAL, cabe uma subdiretriz para refletir a situação da composição de atores.

Embora no Diagrama SD não haja o conceito explícito de decomposição, uma organização é na realidade composta (agregada) por vários atores. Em pUML/OCL, o Diagrama SD será representado por uma classe global que inclui as várias classes (atores do Diagrama SD), através do mecanismo de *agregação*, que representa a composição de classes. Assim, na subdiretriz Oc1.1 tratamos dessa situação, valendo-nos desse mecanismo.

SubDiretriz Oc1.1: Relacionada com o mapeamento da composição de vários atores.

A **Composição de Atores**, que representam um sistema em i^* , é mapeada em pUML pelo mecanismo de **Agregação** aplicado às correspondentes classes.

$$G_{\text{ActComp}}: i^* \rightarrow \text{pUML}$$

$$\text{Act} = \bigcup_{i=1}^n \text{Act}_i \rightarrow \text{class } A_{\text{greg}} (\text{class } A_i) \text{ para } i = 1 \dots n$$

O ator composto **Act** será representado pela união dos vários atores (partes) que o compõe, nos modelos em i^* . Considera-se que cada um dos atores, **Act_i**, seja mapeado em pUML, segundo Oc1, em uma classe **A_i**. Portanto, teremos que se o ator **Act** = $\bigcup_{i=1}^n \text{Act}_i$, então, a classe resultante **A_{greg}**, em pUML, será o resultado da agregação das classes **A_i**.

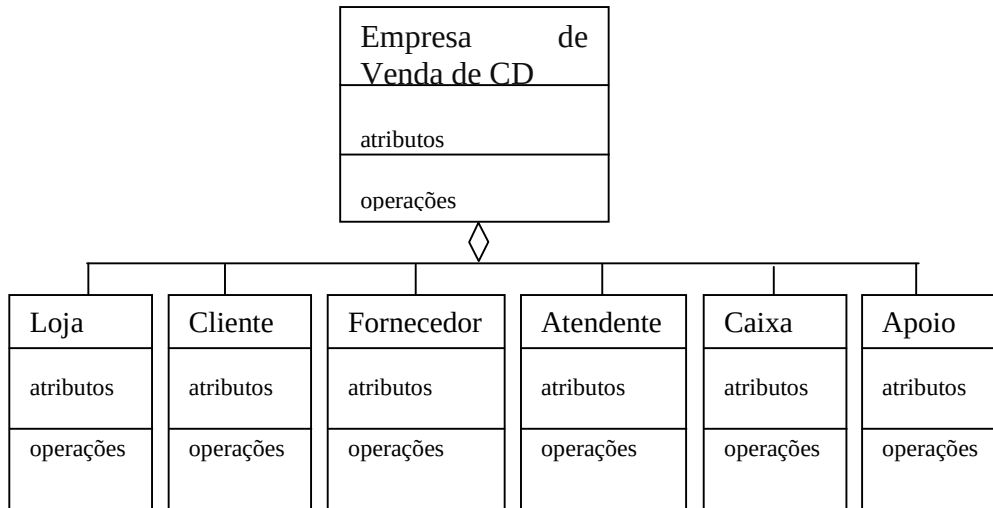


Figura 5.14 – Composição em i^* , Mapeada como Agregação de Classe, em UML

Na Figura 5.14, temos o resultado da aplicação da subdiretriz Oc1.1, onde tomamos a Empresa de Venda de CD, modelada como uma agregação das Classes Loja, Cliente, Fornecedor, Atendente, Caixa e Apoio, que correspondem aos atores do Modelo de Dependências Estratégicas, Figura 5.1, da Loja de Venda de CD's que foram mapeados em classes segundo a diretriz Oc1, Figura 5.13.

Novamente destacamos que, de forma geral, neste mapeamento não nos preocupamos com o fato de ser a classe implementável ou não, mas lembramos a

necessidade de fazermos essa identificação, caso desejemos continuar a descrição do sistema computacional desejado.

Considerações Oc2: Relacionadas com o mapeamento das tarefas dos modelos i*.

Vimos que em i* as tarefas são as atividades a serem realizadas pelos diversos atores da organização descrita. Como estamos tratando com a modelagem organizacional e a nível mais abstrato, é possível que nem todas as tarefas, possam ser implementadas.

Em Structured MAL, encontramos o conceito de operação sobre a classe, representando atividade (processo) que os objetos da classe podem executar.

Em ambos os mundos os conceitos dizem respeito a atividades que devem ser realizadas. Obviamente, Todavia, tarefas relacionadas com atores implementáveis serão mapeadas através de ações em Structured MAL.

Estamos propondo representar as tarefas como operações em pUML, a fim de que, a posteriori, possamos fazer restrições através de pré e pós-condições. Desta forma precisamos rever o conceito de operação relacionado a orientação a objeto. Assim, operações estão relacionadas a serviços de uma classe ou a comportamentos resultantes de um procedimento algorítmico.

Como já salientado, ao nível da modelagem que tratamos, propomos um mapeamento de tarefas em operações (diretriz Oc2), sem a preocupação de serem estas implementáveis ou não. Todavia, ressaltamos que para chegarmos a um código executável, faz-se necessário a preocupação sobre que tarefas serão ou não implementáveis, aplicando-se o mapeamento apenas àquelas que forem, mesmo porque, com as pré e pós-condições em OCL, aplicadas às operações, objetiva-se tornar essas operações mais precisas, em algumas vezes fazendo-se uso de expressões bem próximas das linguagens de programação conhecidas.

Diretriz Oc2: Relacionada com o mapeamento das tarefas dos modelos i*.

As **Tarefas**, nos modelos estratégicos da técnica i*, são mapeadas como *operações de uma classe em pUML*.

$$G_{Ta}: i^* \rightarrow pUML$$

Tarefa $\rightarrow$ Operação da Classe

Considerações Oc2.1 e Oc2.2: Relacionadas com o componente tarefa nos Modelos de Dependências Estratégicas (SD) e de Razões Estratégicas (SR), respectivamente.

Embora, não encontremos explicitamente o conceito de visibilidade em i^* , podemos associar em pUML/OCL, às tarefas encontradas no Modelo de Dependências o conceito de visibilidade pública; enquanto às tarefas no Modelo de Razões Estratégicas (SR), podemos associar o conceito de visibilidade privada.

Em pUML/OCL, existe explicitamente o conceito de visibilidade de operação. A visibilidade privada de uma operação significa que esta operação só pode ser utilizada pela classe na qual se encontra definida. Simbolicamente a visibilidade privada é expressa pelo símbolo “-”. A visibilidade pública de uma operação significa que esta pode ser utilizada por outras classes. Simbolicamente a visibilidade pública é expressa pelo símbolo “+”.

Quando do mapeamento para pUML/OCL estaremos acrescentando aos modelos i^* , informação de que as tarefas, agora operações, devem ser públicas, numa tentativa de representar o relacionamento externo entre os atores; ou privadas, para expressar o comportamento interno do ator que está sendo descrito (Modelo SR).

Desta forma, como no mapeamento para Structured MAL, cabe o estabelecimento de duas subdiretrizes, Oc2.1, que trata das tarefas com relação ao modelo SD, e Oc2.2, relacionada com as tarefas no modelo SR.

SubDiretriz Oc2.1: Relacionada com o mapeamento do componente tarefa no Modelo de Dependências Estratégicas (SD).

*Uma **Tarefa**, no Modelo de Dependências Estratégicas (SD), quando computacionalmente implementável, será entendida como uma **operação pública** da classe relativa ao ator que seja responsável por realizá-la.*

As operações devem ser compartilhadas porque, conforme visto anteriormente (seções 5.2.1 e 3.2.1), em se tratando do Modelo de Dependências Estratégicas (SD), as tarefas são elementos componentes da relação de dependência entre atores. Assim, se um ator A_1

depende que um ator A_2 execute uma tarefa Ta_1 , então essa tarefa, segundo a diretriz acima, será mapeada como uma operação compartilhada, $(+ Op_1)$, da classe A_2 , onde se mapeou o ator A_2 , segundo Oc1. Ou seja, teremos

$$G_{Ta}: i^* \rightarrow pUML$$

$Ta_1 \rightarrow (+ Op_1)$ da (class A_2)

No modelo da Figura 5.1 encontramos uma dependência de tarefa, “Enviar_CD_Solicitado”, entre os atores “Fornecedor” e “Apoio”, mas que é de responsabilidade do primeiro. Logo, esta tarefa deve aparecer como uma operação da classe Fornecedor, segundo Oc2.1, Figura 5.15 mais adiante.

SubDiretriz Oc2.2: Relacionada com o mapeamento do componente tarefa no Modelo de Razões Estratégicas (SR).

*Uma **Tarefa**, no Modelo de Razões Estratégicas (SR), quando computacionalmente implementável, é mapeada como uma **operação local** da classe relativa ao ator que se esteja detalhando o comportamento.*

Quando trabalhamos com o Modelo de Razões Estratégicas (SR), como visto na Seção 3.2.2, observamos o comportamento interno de cada ator. Assim, nesta situação, as tarefas, modeladas internamente segundo este modelo, são realizadas pelo ator que se esteja detalhando o comportamento. Desta forma, avaliando-se o comportamento interno do ator A_1 , se identificarmos uma tarefa, então esta tarefa Ta_1 será mapeada como uma *operação local*, Op_1 , da classe A_1 . Ou seja, teremos

$$G_{Ta}: i^* \rightarrow UML$$

$Ta_1 \rightarrow (+ Op_1)$ da (class A_1)

Considerando o ator “Atendente” no Modelo de Razões Estratégicas, Figura 5.4, vemos que este possui, em seu comportamento interno, a explícita tarefa de “Registrar_Encomenda”, que será traduzida como uma operação local da classe “Atendente”, segundo Oc2.2, Figura 5.15.

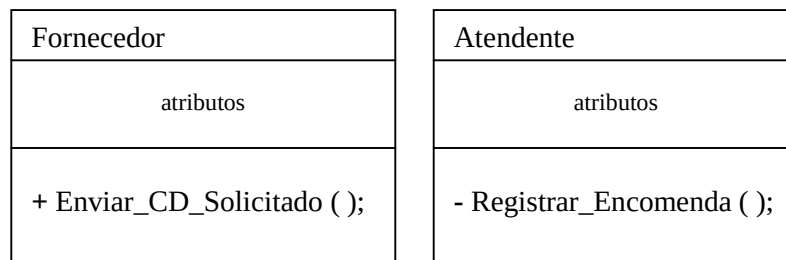


Figura 5.15 – Tarefas Mapeadas em Operações de Classe em UML

Considerações Oc3: Relacionadas com o mapeamento dos recursos dos modelos i*.

O elemento recurso nos modelos estratégicos da técnica i* corresponde a uma entidade física do mundo real ou uma informação, um objeto que existe. Neste caso, os recursos a nível de implementação serão entidades implementáveis, dado que são objetos concretos que são manipulados pelos atores, segundo suas necessidades.

Em pUML/OCL, o elemento que tem vida própria é a classe, visto que por definição é uma representação de um conjunto de coisas reais ou abstratas (objetos), a semelhança dos princípios da orientação a objetos.

O mapeamento do recurso em uma classe em pUML/OCL, dá ao mesmo uma capacidade de expressão muito maior do que a necessária, pois cada classe possui atributos associados, que expressam seu estado, e realiza operações. A nível da modelagem organizacional em i*, interessa-nos apenas a informação da disponibilidade (ou não) do recurso. Assim, esta situação é representada através da inclusão de um atributo do tipo booleano, na classe correspondente ao recurso modelado. Caso o atributo seja verdadeiro (true) significa a disponibilidade do recurso, caso contrário (false), o recurso não estará disponível. O nome desse atributo fica de livre escolha, apenas sugerimos que seja o mais próximo do elemento recurso, unicamente por uma questão de simplicidade. Desta forma, propomos a diretriz Oc3.

Diretriz Oc3: Relacionada com o mapeamento dos recursos dos modelos i*.

Os **Recursos** dos modelos estratégicos da técnica *i** são mapeados como uma **Classe** em *pUML*, cujo **atributo público** ($atbr_{rec}$) do tipo **booleano**, indica a disponibilidade desse recurso.

Conforme dissemos na Seção 5.2.1, este mapeamento em uma classe *pUML*, ocorre em virtude de um recurso ser uma entidade física ou uma informação concreta. Assim, se **Re** é um recurso, em qualquer dos modelos estratégicos do *i**, então será mapeado como uma classe de mesmo nome,

$$\begin{aligned} G_{Re}: i^* &\rightarrow pUML \\ Re &\rightarrow \text{class Re} \end{aligned}$$

devendo existir um atributo, ($atbr_{Re} : \text{boolean}$) da (**class Re**).

Observando os modelos estratégicos das Figuras 5.1 e 5.4, identificaremos vários recursos, os quais serão todos mapeados como classes. Na Figura 5.16, podemos visualizar a classe relativa aos recursos “Pedido_Encomenda”, “Pedido_de_Compra” e “CD_Solicitado”, onde apenas consideramos a parte da classe relativa aos atributos, não nos importando, para o caso em questão, se esta classe tem ou não operações, bem como outros atributos. Outros recursos do estudo de caso da Empresa de Venda de CD serão modelados no Capítulo 6.

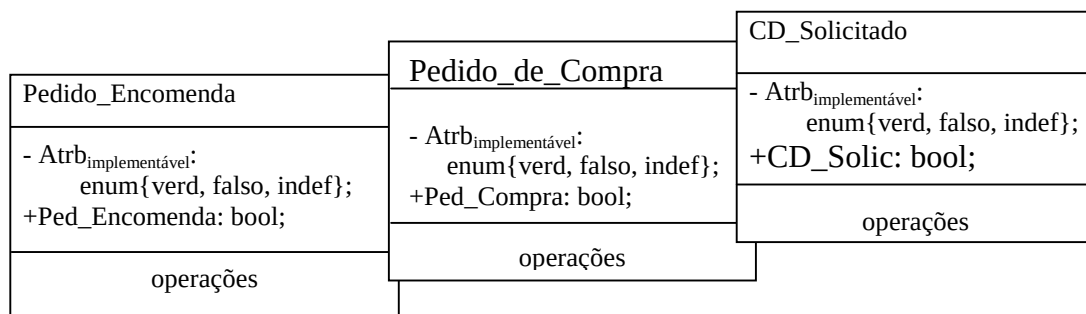


Figura 5.16 – Recurso em *i**, Mapeado em uma Classe UML

Considerações Oc4: Relacionadas com o mapeamento dos objetivos bem definidos e objetivos- soft dos modelos *i**.

Como anteriormente visto, um *objetivo* em *i** corresponde a uma condição ou estado do mundo que um ator gostaria de alcançar. Enquanto que o conceito de *objetivo-*

soft é similar ao de objetivo, exceto pelo fato de que a condição de sucesso não é, precisamente, definida a priori, ou seja, os objetivos envolvem aspectos subjetivos que vão sendo esclarecidos durante o desenvolvimento do processo.

Do ponto de vista do pUML/OCL, podemos associar aos conceitos de objetivo e objetivo-soft o conceito de atributo de uma classe. É importante relembrar que o atributo é parte do estado do sistema e que é a menor unidade que em si possui significado próprio e interrelacionado com o conceito lógico da classe à qual pertence. Como em i*, objetivos e objetivos-soft representam desejos, metas ou intenções dos atores, propomos representar os objetivos e objetivos-soft de um ator em i* como atributos de uma classe em pUML/OCL.

Os objetivos (bem definidos) serão representados por um atributo do tipo booleano para sinalizar a satisfação (true) ou não (false) destes objetivos. Obviamente, que alguns desses objetivos não são implementáveis, porque podem representar aspectos almejados por um ator, mas que não conseguiremos traduzir por alguma rotina executável, sob o ponto de vista de implementação. Deixamos para um passo a posteriori a preocupação com este aspecto o que não invalida a diretrizes ora propostas. Destacamos que se espera que os atores (agentes) implementáveis, também possuam objetivos implementáveis.

Com relação aos objetivos-soft, relacionados a requisitos não funcionais, serão associados atributos do tipo enumerado, cujos possíveis valores para o atributo são {posit, negat, indef, sub e naosub}, representando as possíveis contribuições positiva, negativa, indefinida, bastante e não bastante, respectivamente, desses objetivos, expressos no Modelo de Razões Estratégicas (SR), (Seção 3.2.2). Conforme dissemos na seção anterior, nos axiomas em que apareça, o valor do atributo associado a um objetivo-soft deve ser avaliado.

Da mesma forma que em Structured MAL, os objetivos e objetivos-soft serão mapeados como atributos de classe em pUML/OCL. Ressaltando-se que nem todos os objetivos ou objetivos-soft, poderão ser implementáveis. Espera-se que os atores implementáveis também possuam objetivos implementáveis.

Diretriz Oc4: Relacionada com o mapeamento dos objetivos bem definidos e objetivos-soft dos modelos i*.

Os *objetivos bem definidos* e os *objetivos soft* dos modelos estratégicos serão mapeados como *atributos do tipo booleano e do tipo enumerado*, respectivamente, de uma classe pUML;

Desta forma, os objetivos bem definidos são mapeados como atributos do tipo booleano

$G_{Obj}: i^* \rightarrow pUML$

Objetivos $\rightarrow$ Atributo: boolean

e os objetivos-soft são mapeados como atributos do tipo enumerado

$G_{ObjSoft}: i^* \rightarrow pUML$

Objetivos-Soft $\rightarrow$ Atributo: enum {posit, negat, indef, sub, naosub}

Considerações Oc4.1 e Oc4.2: Relacionadas com os objetivos e objetivos-soft nos Modelos de Dependências Estratégicas (SD) e de Razões Estratégicas (SR).

Como descrito nas outras diretrizes e já mencionamos que em i* não existe o conceito de visibilidade de elementos, faremos uso do conceito encontrado em pUML/OCL. Os objetivos e objetivos-soft devem ser considerados segundo o Modelo de Dependências Estratégicas (SD) ou Modelo de Razões Estratégicas (SR).

Como mencionado para as operações de uma classe, em pUML/OCL, o conceito de visibilidade também se aplica aos atributos. De forma geral, podemos dizer que um atributo será de uso privado (apenas podem ser utilizadas pela classe no qual se encontra definido), nesse caso rotulado com “-“(privado) ou será público, possível de ser utilizado por outras classes e rotulados com “+” (público).

Desta forma, propomos as duas subdiretrizes, M4.1 e M4.2, respectivamente, para os modelos SD, onde os objetivos e objetivos-soft podem ser compartilhado entre os vários atores (classes) envolvidos e, portanto, mapeados como atributos públicos; e SR, onde os objetivos e objetivos-soft são locais ao ator (classe) cujo comportamento se esteja

descrevendo, mapeados como atributos privados. Nesse ponto, salientamos que foram acrescentadas informações que enriqueceram os elementos encontrados nos modelos estratégicos da técnica i*.

SubDiretriz Oc4.1: Relacionada com o mapeamento dos objetivos bem definidos e objetivos-soft no Modelo de Dependências Estratégicas (SD).

Objetivos e Objetivos-Soft, no Modelo de Dependências Estratégicas (SD) da técnica i*, podem ser entendidos como **atributos booleanos público e enumerados públicos**, respectivamente, da Classe, em pUML, que corresponde ao ator a quem pertençam os objetivos.

Se um ator A_1 depende que um ator A_2 satisfaça um objetivo bem definido Ob_1 , então esse objetivo, segundo a diretriz acima, será mapeado como um atributo público, (+Atrb₁) da classe A_1 em UML, resultante do mapeamento do ator A_1 em uma classe, ou seja, teremos

$$\begin{aligned} G_{Obj}: i^* &\rightarrow \text{pUML} \\ Obj_1 &\rightarrow (+ \text{Atrb}_1: \text{boolean}) \text{ na (class } A_1) \end{aligned}$$

De forma semelhante, os objetivos-soft são mapeados como atributos de classe do tipo enumerado

$$\begin{aligned} G_{ObjSof}: i^* &\rightarrow \text{pUML} \\ ObjSoft_1 &\rightarrow (+ \text{Atrb}_1: \text{enum}\{\text{posit, negat, indef, sub, naosub}\}) \\ &\text{na (class } A_1) \end{aligned}$$

Por exemplo, considerando-se a Figura 5.1, observa-se que, no relacionamento entre os atores “Loja” e o “Atendente”, o primeiro espera que o último registre a encomenda de CD’s; esse fato é expresso pelo objetivo bem definido “Registre_Encomenda”, que será mapeado como um atributo do tipo booleano da classe *Loja*, apresentado na Figura 5.17. A característica de ser público encontra-se ligado à possibilidade deste ser influenciado pelo outro agente, envolvido na relação de dependência.

SubDiretriz Oc4.2: Relacionada com o mapeamento dos objetivos bem definidos e objetivos- soft no Modelo de Razões Estratégicas (SR).

*Os **objetivos bem definidos** e os **objetivos soft** internos a um ator que se esteja detalhando, em i^* , são mapeados como **atributos do tipo booleano local e do tipo enumerado** respectivamente, da classe correspondente em pUML que se esteja observando o comportamento interno.*

Assim se um ator A_1 , quando avaliamos seu comportamento interno, pretende alcançar um objetivo bem definido Obj_1 , então esse objetivo, segundo a diretriz acima, será mapeado como um atributo local do tipo booleano, $Atrb_1$, da classe A_1 , em pUML, ou seja, teremos

$$\begin{aligned} G_{Obj}: i^* &\rightarrow \text{UML} \\ Obj_1 &\rightarrow (- Atrb_1: \text{boolean}) \text{ na (class } A_1) \end{aligned}$$

Da mesma forma, os objetivos-soft serão mapeados como atributos privados do tipo enumerado.

$$\begin{aligned} G_{ObjSof}: i^* &\rightarrow \text{UML} \\ ObjSoft_1 &\rightarrow (- Atrb_1: \text{enum}\{\text{posit, negat, indef, sub, naosub}\}) \\ &\quad \text{na (class } A_1) \end{aligned}$$

Por exemplo, observando-se o Modelo de Razões Estratégicas, na Figura 5.4, identificamos que o agente Loja, possui alguns objetivos-soft – “Ter_Qualidade [Atendimento]”, “Volume_Compra[Alto]”, “Limite_Preço[Baixo]” e “Preços[Bons]” - e objetivos bem definidos – “Registre Encomenda” e “Compre_CD’s”. Aplicando-se a diretriz Oc4.1, incluímos nos atributos da classe Loja, atributos públicos rotulados por “+” e aplicando-se Oc4.2, incluímos nos atributos dessa mesma classe, atributos locais rotulados por “-”, Figura 5.17.

Loja
- Atrib_{implementável}: enum{verd, falso, indef}; + Registre Encomenda: boolean; + Compre_CD's: boolean; - Ter_Qualidade[Atendimento]: enum{posit,negat, indef, sub,naosub}; -Volume_Compra[Alto]: enum{posit,negat, indef, sub, naosub}; - Limite_Preço[Baixo]: enum{posit,negat, indef, sub, naosub}; - Preços[Bons]: enum{posit,negat, indef, sub, naosub};
- Fazer_Clientela (); - Atrair_Cliente (); - Fornecer_serviços_especiais (); - Fornecer_Vendas_Por_Telefone (); - Fornecer_Vendas_Por_Encomenda (); - Manter_CD_em_Estoque ();

Figura 5.17 – Objetivos e Ojetivos-Soft Mapeados como Atributos Booleanos

Na Figura 5.17 vemos que todos os atributos públicos correspondem aos objetivos bem definidos da Loja, que foram mapeados segundo a subdiretriz Oc4.1, enquanto todos os atributos locais correspondem aos objetivos-soft, mapeados segundo a subdiretriz Oc4.2.

Uma vez mapeados todos os elementos dos modelos estratégicos, podemos passar a observar as restrições que podem ser colocadas utilizando OCL.

A sintaxe de comentários em OCL é dada pelo sinal "--", no início do texto comentário. Assim, no momento, a partir da representação da classe Loja, Figura 5.17, poderemos observar que existem seis operações locais: (Fazer_Clientela (), Atrair_Cliente (), Fornecer_ser-viços_especiais (), Fornecer_Vendas_Por_Telefone (), Fornecer_Ven-das_Por_ Encomenda () e Manter_CD_em_Estoque ()).

Loja: : Manter_CD_em_Estoque () post: (CD_Solicitado.CD_Solic = posit) $\wedge$ (exist (CD)) -- Significa que o CD está disponível e que se encontra na Loja para ser vendido. pré: estoque < n -- Corresponde a ter em estoque uma quantidade menor que um limite de segurança CD.
Loja: : Atrair_Cliente () pre: existência_de_consumidores – condição não expressa em i*, que representa a existência de um mercado consumidor, aqui representado por um valor booleano que indica existência ou não. post : not cliente -> includes (cliente) -- O resultado de atrair cliente será a inclusão desse novo

Figura 5.18 – Aplicação de Pré e Pós Condições, em OCL.

Na figura 5.18, tem-se apenas duas das seis operações da classe Loja, com suas pré e pós-condições que, como pode ser observado, podem não vir dos modelos i*, mas de requisitos bem mais delimitados a serem definidos nas últimas fases do processo e que podem ser expressos em OCL. Por exemplo, a pré-condição para a realização da operação “Manter CD_em Estoque” é que a quantidade de CD solicitados (representado pelo atributo estoque) esteja inferior a um limite mínimo de segurança estipulado pela loja (representado pelo valor n). Como resultado desta ação teremos CD solicitados disponíveis e presentes na Loja (exist(CD)). Observemos que esta pós-condição se relaciona com um atributo de uma outra classe, no caso “CD_Solicitado” que corresponde, no modelo SR da Figura 5.4, à disponibilidade de CD. Este é um exemplo típico de navegação, principal ponto encontrado em OCL. Com relação a operação “Atrair Cliente”, a pré-condição é a existência de consumidores não modelada em i* no

momento, pelo que inserimos a *existência_de_consumidores*. Como resultado da ação “Atrair Cliente” teremos a inclusão de novos clientes na Loja.

Como vimos anteriormente na Seção 3.2.2, o modelo SR é usado para detalhar alguns dos atores do modelo SD. Surgem duas novas ligações no Modelo de Razões Estratégicas (SR): as ligações de decomposição de tarefas e as ligações meio-fim. Neste modelo os atores são detalhados, a fim de se observar seu comportamento interno. Assim, estabelecemos as diretrizes Oc5 e Oc6, respectivamente, para tratarem da decomposição de tarefas e do relacionamento meio-fim.

Considerações Oc5: Relacionada com o mapeamento dos relacionamentos de decomposição de tarefa nos modelos i*.

Conforme já mencionado anteriormente, a ligação *de decomposição de tarefa* é encontrada no Modelo de Razões Estratégicas (SR) e exprime o que deve ser feito para se ter a tarefa realizada, podendo a tarefa ser decomposta nos quatro tipos:

- (a) *sub-objetivo*,
- (b) *sub-tarefa*;
- (c) *recursoPara*; ou
- (d) *objetivo-softPara*.

Essas situações foram representadas em Structured MAL pelos axiomas dos agentes. Em pUML/OCL faremos uso das pré e pós-condições aplicadas à correspondente operação (em que foi mapeada a tarefa que está sendo decomposta). Os elementos em que a tarefa (operação) será decomposta serão considerados como pós-condições para esta. Todas as contribuições dos elementos, são tomados em conjunção, através do conectivo lógico AND. Como pré-condição da tarefa (operação) que está sendo decomposta deixamos apenas a indicação (pré-condOp) sobre a necessidade de se melhor definir, a posteriori. Ao nível de especificação que estamos tratando, algumas coisas não se encontram bem definidas.

Devemos considerar, novamente, o fato de que nem todos os elementos são implementáveis, ao nível de especificação em que trabalhamos. Esta preocupação é deixada para fases posteriores do desenvolvimento (later requirements).

Diretriz Oc5: Relacionada com o mapeamento dos relacionamentos de decomposição de tarefa nos modelos i*.

Uma ligação do tipo **decomposição de tarefa** é representada, em OCL, através de **pré- e pós-condição** na operação correspondente à tarefa que está sendo decomposta. Caso exista como elemento da composição uma sub-tarefa, suas pré-condições farão parte da pré condição da tarefa que está sendo decomposta. Na possibilidade de conjunção dos elementos, são utilizados os conectivos lógico **AND**.

Portanto, quando identificamos um Relacionamento de Decomposição de Tarefa no Modelo de Razões Estratégicos (SR) da técnica i*, os elementos em que a tarefa é decomposta representam os resultados da execução dessa tarefa. Assim, uma vez representados em pUML, através das diretrizes anteriores, estes elementos serão as pós-condições da operação que esteja sendo avaliada.

Precisamos considerar o caso de ser o elemento da decomposição uma tarefa. Por exemplo, na Figura 5.4, a tarefa “Fazer Clientela” possui, como um de seus elementos, a tarefa “Atrair Clientes”. Nesta situação, na pós-condição da operação correspondente a tarefa sob decomposição, constará o resultado da operação da tarefa correspondente à sub-tarefa, expresso por meio de sua pós-condição. Esse resultado, no momento, não nos interessa definir, apenas associá-lo-emos a uma designação, como por exemplo, pos-condOp₁; e, da mesma forma, na pré-condição da operação correspondente a tarefa sob decomposição, constará a pré-condição da operação da tarefa correspondente à sub-tarefa. Essas pré-condições no momento não nos interessa definir, apenas associá-las-emos a uma designação, como por exemplo, pre-condOp₁.

Assim, se a operação Op (relativa a tarefa mapeada T) é decomposta em uma sub-operação Op₁, uma sub-objetivo Ob₁, um recurso Re₁ e um objetivo-soft ObSoft₁, então, segundo a diretriz, tem-se em OCL:

Operação Op

pre: (pre-condOp₁)

post: (pos-condOp₁) and(Atbr_{Re1}) and(Atrb_{obj1}) and(Atrb_{objSoft1} = ‘valor’)

O termo **(pre-condOp₁)** corresponde à pré-condição da sub-operação em que é decomposta a operação em avaliação. Essa pré-condição deve fazer parte das pré-condições da operação que está sendo decomposta, dado que poderão haver outras pré-condições que no momento não nos interessam.

O termo **(pos-condOp₁)** corresponde ao resultado da realização da sub-operação em que é decomposta a operação em avaliação. O termo **atbr_{Re1}** corresponde ao atributo compartilhado da classe recurso Re₁. O termo **Atrb_{obj1}** corresponde ao atributo local, da própria classe, correspondente a um sub-objetivo bem definido, com o correspondente resultado de sua avaliação em verdadeiro, falso ou indefinido. O termo **Atrb_{objSoft1} = ‘valor’** corresponde ao atributo local da própria classe, correspondente a um objetivo-softPara.

Na Figura 5.4, observando um pequeno fragmento do agente Loja, identificamos que a tarefa “raiz”, “*Fazer Clientela*”, é decomposta em uma sub-tarefa, “*Atrair Cliente*”, e em um objetivo-softPara, que diz respeito a “*Ter qualidade de atendimento*”, que representam as folhas dessa raiz.. Da mesma forma, a tarefa “*Atrair Cliente*”, é decomposta em duas sub-tarefa: “Manter CD em Estoque” e “Fornecer Serviços Especiais” e em um sub-objetivo soft, “Preços[Bons]”. Na Figura 5.19, com base no Modelo de Razões Estratégicas (SR) da Figura 5.4, podemos observar estas decomposições, conforme nos indica a diretriz Oc5.

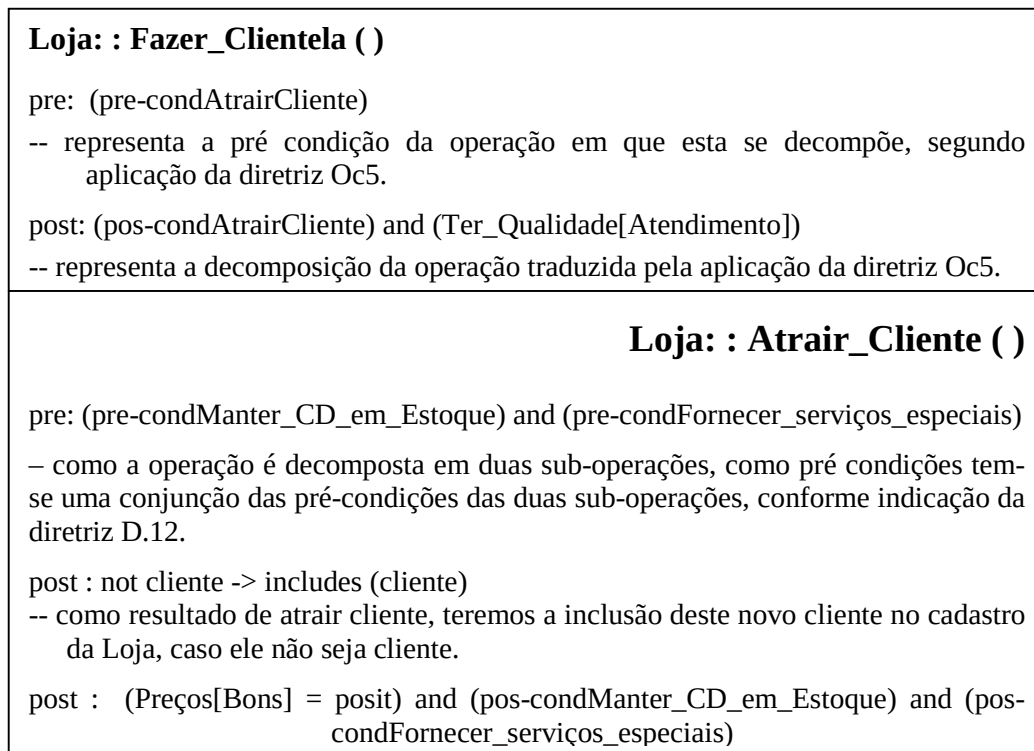


**Figura 5.19 – Relacionamento de Decomposição de Tarefa em i*,
Expresso por Pós-Condições em OCL.**

Na Figura 5.19, vimos a inclusão de pré-condições que não estão sendo representadas nos modelos estratégicos da técnica i*, mas que foram adicionados, como forma de refinamento da operação. Já as pós-condições representam a aplicação da diretriz proposta OC5. Apenas na operação “Atrair_Cliente” foi acrescida uma outra pós-condição adicional.

Considerações Oc6: Relacionadas com as ligações do tipo meio-fim no Modelo de Razões Estratégicas (SR)

Enfatizamos que em i*, os *relacionamentos meio-fim* indicam um relacionamento entre um *fim* que pode ser:

- (a) um objetivo a ser alcançado;
- (b) uma tarefa a ser realizada;
- (c) um recurso a ser produzido; ou

(d) um objetivo soft a ser satisfeito

e um meio, para se atender a esses fins. Neste tipo de relacionamento, o meio para se alcançar um fim sempre é uma tarefa. Exceto em duas situações, conforme pode ser visto no Capítulo 3, onde o fim e o meio são ambos, objetivos ou objetivos-soft. Em geral, um relacionamento meio-fim sugerem a existência de meios para se alcançar o mesmo fim, exprimindo as alternativas existentes.

Em pUML/OCL, esta situação será expressa nas restrições sobre as operações das classes e sobre os invariantes de classes, utilizando-se do conectivo lógico OR para exprimir a disjunção vários meios de que se disponha para se alcançar um fim. Consideraremos os vários tipos de relacionamento possíveis segundo sejam os fins, estabelecendo três subdiretrizes.

Diretriz Oc6: Relacionada com as ligações do tipo meio-fim no Modelo de Razões Estratégicas (SR);

Uma ligação do tipo **meio-fim** é representada, em OCL, através de uma disjunção dos vários meios, utilizando o conectivo **OR**, nas **pós-condições** das operações correspondente ou **invariantes** de classe.

Em se tratando da ligação meio-fim, temos que considerar os tipos existentes, conforme Figura 3.4b, Capítulo 3. De forma geral, esse tipo de ligação é expressa através de restrições em OCL, observando-se três tipos de situações:

- (i) quando o meio for uma tarefa e o fim uma tarefa;
- (ii) quando o meio for uma tarefa e o fim um objetivo bem definido, objetivo-soft ou recurso.
- (iii) quando o meio é um objetivo bem definido ou objetivo-soft, e o fim também é um objetivo bem definido ou objetivo-soft.

Para cobrir estas situações consideraremos as três seguintes subdiretrizes.

Considerações Oc6.1: Relacionadas com ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é uma tarefa.

Destacamos que todas as tarefas envolvidas serão mapeadas em operações, onde devemos considerar a questão de ser ou não implementável, caso desejemos chegar a um código executável. Ao nível em que estamos tratando, deixamos apenas indicações para a inclusão de pré e pós-condições nas operações meios, assim como pré e pós-condições da operação fim. O que caracterizará a ligação meio-fim será a utilização do conectivo lógico OR na pós-condição da operação fim, evidenciando as alternativas existentes.

SubDiretriz Oc6.1: Relacionada com ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é uma tarefa.

*Uma ligação do tipo **meio-fim**, onde o **meio é uma tarefa** e o **fim é uma tarefa**, é representada, em OCL, através da inclusão de uma **pré-condição** na operação correspondente à **tarefa fim**, que pode ser uma conjunção (**AND**) das pré-condições das subtarefas, e da inclusão de uma **pós-condição** na operação correspondente à **tarefa fim**, em associação com o conectivo lógico **OR**.*

Portanto, se temos uma tarefa fim, mapeada para pUML, segundo D.9, como a operação, **Op**, e como meio tenhamos n tarefas, mapeadas para pUML, segundo D.9 ($Op_1, Op_2, \dots, Op_n$), então teremos

Operação Op

pre: (pre-cond Op_1) and (pre-cond Op_2) and ... and (pre-cond Op_n)

post: (pos-cond Op_1) or (pos-cond Op_2) or ... or (pos-cond Op_n)

onde a pós-condição nos diz que o fim **Op** será alcançado, tendo-se como resultado pelo menos uma das pós-condições de ocorrência das operações Op_i . Com esta restrição, procuramos mostrar que existem n alternativas para se executar uma operação.

Desta forma, tomando-se o exemplo da Loja de venda de CD, Figura 5.4, identificamos uma ligação meio-fim onde o fim é a tarefa “Fornecer Serviços Especiais” e como meios tem-se as tarefas: “Fornecer Vendas por Telefone” e “Fornecer Vendas sob Encomenda”. Na Figura 5.20 temos a aplicação da subdiretriz Oc6.1, para a operação “Fornecer Serviços Especiais”.

Loja: : Fornecer_serviços_especiais

pre: (pre-condFornecer_vendas_por_telefone) and
(pre-condFornecer_vendas_sob_encomenda)

-- representa aplicação da subdiretriz Oc6.1, considerando as pré-condições das sub-operações em que se decompõe a operação em questão.

post: (pos-condFornecer_vendas_por_telefone) or
(pos-
condFornecer_vendas_sob_encomenda)

Figura 5.20 - Relacionamento Meio-Fim em i*, onde Meio e Fim são Tarefas.

Na Figura 5.20, vemos que a pós-condição para “Fornecer_serviços_Especiais” depende das pós-condições das operações “Fornecer_vendas_por_telefone” ou “Fornecer_Vendas_sob_Encomenda”, que representam as alternativas para a realização da operação fim, segundo a técnica i*, mas também dependem da ocorrência das pré-condições de cada uma das operações possíveis.

Considerações Oc6.2: Relacionadas com ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é um objetivo, um objetivo-soft ou um recurso.

Como as restrições são colocadas sobre as operações, neste caso serão feitas sobre a operação que representa o meio, considerando-se os vários possíveis fins. A principal consideração deve ser feita sobre a questão da necessidade futura de melhor definir as pré-condições da operação e se preocupar com a questão de ser ou não implementável. Lembramos que se espera que às classes implementáveis estejam associadas operações implementáveis.

SubDiretriz Oc6.2: Relacionada com ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é um objetivo, um objetivo-soft ou um recurso.

*Na operação que representa a tarefa meio, é adicionada uma **pós-condição** onde se tem uma **conjunção (AND)** das condições relacionadas com a ocorrência de quaisquer dos fins*

Portanto, se tivermos a operação **Op** com meio e como fim um objetivo bem definido, **Atrb_{Obj}**, um objetivo-soft, **Atrb_{ObjSoft}** e um recurso a ser disponibilizado, **Atrb_{re}**, então

Operação Op

pre: (pre-condOp)

post: Atrb_{Obj} and (Atrb_{ObjSoft} = 'valor') and Atrb_{Re}

onde Atrb_{obj} é um atributo booleano local que representa um objetivo bem definido, mapeado segundo Oc4; Atrb_{objSoft} é um atributo enumerado local que representa um objetivo soft, mapeado segundo Oc4; e Atrb_{re} é um atributo booleano público, da classe recurso, que indica a disponibilidade desse recurso.

Observando a Figura 5.4, não identificamos nenhuma ligação meio-fim que se adequa a esta situação aponta pela subdiretriz 13.2.

Considerações Oc6.3: Relacionadas com a expressão dos relacionamentos meio-fim dos modelos i*, quando o meio e fim são, simultaneamente, objetivos bem definidos ou objetivos-soft.

Apenas relembremos que todos os elementos devem ser mapeados segundo as diretrizes propostas, relacionadas com os objetivos e objetivos-soft. Como não estamos tratando com operações, mas sim com atributos, faremos uso de invariantes aplicados sobre os atributos da classe. No caso, a classe corresponderá ao ator cujo comportamento interno está sendo analisado e é expresso pelo Modelo de Razões Estratégicas (SR). Enfatizamos que, quando se utiliza um invariante, deve-se ter em mente que todas as instâncias da classe deverão satisfazer a este invariante.

SubDiretriz Oc6.3: Relacionada com a expressão dos relacionamentos meio-fim dos modelos i*, quando o meio e fim são, simultaneamente, objetivos bem definidos ou objetivos-soft.

*É estabelecido um invariante de classe do tipo **meio implies fim**, onde meio poderá ser uma disjunção (**OR.**) dos atributos que representam os objetivos bem definidos ou objetivos-soft.*

Portanto, se na classe **CL** está sendo observado um relacionamento meio-fim em que como fim tem-se um objetivo bem definido **Ob** e, como meio, **n** outros objetivos bem definidos (**Ob₁, Ob₂, ..., Ob_n**), então OCL

Classe CL

(Atrb_{Obj1} or Atrb_{Obj2} or or Atrb_{Objn}) implies Atrb_{Obj}

onde os atributos são resultantes da aplicação da diretriz Oc4. Da mesma, se ao invés de objetivos bem definidos tivéssemos objetivos soft, teríamos

Classe CL

((Atrb_{ObjSoft1}=‘valor₁’) or (Atrb_{ObjSoft2}=‘valor₂’) or or (Atrb_{ObjSoftn}=‘valor_n’)) implies (Atrb_{ObjSoft}=‘valor’)

Introduzimos um invariante, porque estamos tratando com atributos de classe.

Assim, na Figura 5.21 podemos observar a aplicação dessa subdiretriz Oc6.3, com relação a um relacionamento meio-fim, Figura 5.4, onde identificamos com fim objetivo-soft “Preços{Bons}” e como meios os objetivos soft: “Volume_Compras[Alto]” e “Limite_ Preço[Baixo]”, todos relativos à classe Loja.

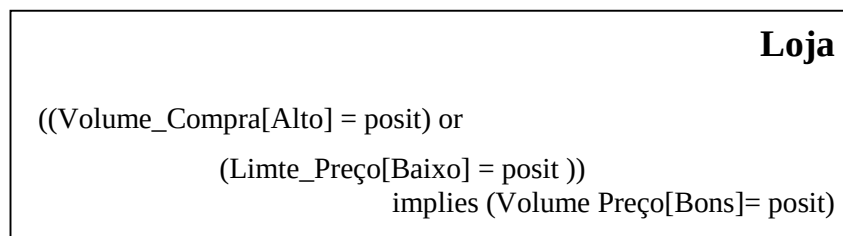


Figura 5.21 – Ligação Meio-Fim Representada como Invariante de Classe em OCL

Com estas diretrizes procuramos cobrir todos os elementos estruturais dos dois modelos que compõem a técnica i^* . A aplicação de OCL será evidenciada apenas no tocante inserção de restrições durante a navegação entre as classes ou quanto aos invariantes de classe e as pré e pós condições, que podem ser inclusas nas operações das classes.

5.3 – RESULTADOS ALCANÇADOS

No mundo real, durante o processo da Engenharia de Requisitos, o engenheiro pode interagir, navegando entre os modelos (SD) e (SR) apresentados pela técnica i^* e as especificações precisas em Structured MAL ou OCL, sendo possível tratar os impactos de cada uma das descrições. Precisamos ressaltar que os dois níveis de descrições — a organizacional e a especificação precisa — têm nos permitido adotar conceitos diferentes de agentes a cada nível. Nos requisitos iniciais (“Early Requirements”), a nível dos relacionamentos organizacionais, é necessário uma notação de agente mais flexível, para que seja possível expressar a liberdade que têm os agentes para violar restrições e compromentimentos. Quando os trabalhos na área da Engenharia de Requisitos avançam (“Late Requirements”) e mais detalhes da solicitação já existem, é possível então fazer uma especificação mais precisa e usar uma visão prescritiva mais apropriada.

Nas próximas seções, estarão sendo avaliados os pontos positivos e negativos encontrados na proposição das diretrizes para a integração dos requisitos organizacionais à especificação precisa dos requisitos funcionais dos sistemas que se deseja especificar.

5.3.1 – LIMITAÇÕES DOS RESULTADOS DO MAPEAMENTO DA TÉCNICA i^* PARA A LÓGICA MODAL DE AÇÕES ESTRUTURADA

Lembramos que a nossa proposta de trabalho trata apenas de suportar o mapeamento de i^* para Structured MAL e não o reverso, ou seja, de Structured MAL para i^* . O processo de especificação de requisitos é incremental, com refinamentos que adicionam detalhes aos requisitos iniciais (“Early Requirements”). Esses refinamentos, nos quais possivelmente mais restrições são impostas, dificultarão o mapeamento no sentido

reverso, ou seja, Structured MAL $\rightarrow$ i^* , pois precisam ser mais abstraídos para a obtenção da descrição em i^* .

Além disso, não estamos propondo um mapeamento completo entre todos elementos dos modelos estratégicos da técnica i^* e Structured MAL. Propomos meios para que aspectos relevantes, apontados pelos modelos estratégicos de i^* , possam complementar a especificação funcional do sistema, descrita em Structured MAL. Um dos aspectos que precisa ser melhor investigado é a possibilidade de um ator em i^* ser estruturado (especializado) em Agentes, Posições e Papéis. Nesta dissertação, o mapeamento da estruturação não foi abordado, em função dessa estruturação não se encontrar ainda bem definida na técnica i^* , sendo objeto de pesquisa atual.

Caso os atores em i^* sejam especializados em Agentes, Posições e Papéis, a semelhança do mecanismo de herança da orientação a objeto, em Structured MAL [RYAN91], ter-se-á também o mecanismo de Herança. Em Structured MAL, herança sintaticamente é expresso pela cláusula **includes**. Observa-se nesse ponto, que a mesma cláusula é usada para expressar Agregação. A diferença, em Structured MAL, entre os mecanismos de Agregação e de Herança está no fato de que neste último podemos modificar os métodos importados do agente pai, bem como acrescentarmos atributos e outros métodos, enquanto que este procedimento não é possível no primeiro mecanismo.

A não existência em Structured MAL de uma forma sintática de expressão distinta para os mecanismos de Herança e Agregação, também se constitui em mais uma dificuldade ao mapeamento no sentido reverso (Structured MAL $\rightarrow$ i^*).

As diretrizes propostas são satisfatórias ao mapeamento no sentido $i^* \rightarrow$ Structured MAL [ALE98a, ALE98b, ALE99a]. Evidente que, quando necessário, uma especificação mais detalhada deve ser produzida, isso considerando-se que apenas os aspectos organizacionais estratégicos são considerados nos modelos apresentados pelas técnicas de modelagem organizacional. No caso particular da técnica i^* , os próprios modelos recebem o nome de estratégicos, um para representar o comportamento externo entre os agentes — Modelo de Dependências Estratégicas (SD) — e o outro para representar o comportamento intencional (motivações, porquês, razões) presentes nos agentes — Modelo de Razões Estratégicas (SR).

5.3.2 – LIMITAÇÕES DOS RESULTADOS DO MAPEAMENTO DA TÉCNICA i* PARA A LINGUAGEM DE RESTRIÇÕES DE OBJETOS (OCL)

O objetivo do uso de OCL com i* será o de complementar os modelos estratégicos da técnica i* com descrições em uma linguagem precisa, no tocante aos aspectos considerados como de maior relevância ao sistema que se deseja especificar. Isso significa que também não estamos tentando descrever todos os aspectos e restrições em OCL, mesmo porque isto nos parece impossível no momento, mas apenas fornecer meios para formalizar aspectos em i*, que servirão, possivelmente, para complementar especificações da parte funcional do sistema. Pretendemos tão somente fornecer meios para tornar mais precisos aqueles aspectos dos modelos de i* que servirão para complementar especificações da parte funcional do sistema, possivelmente modelados segundo pUML. Em [HAM98a] fala-se que OCL não existe isoladamente, depende de uma descrição orientada a objeto, por exemplo definida nos diagramas de uma linguagem como UML. Daí a motivação de termos relacionado o modelo i* com o modelo UML. Como UML é muito ampla e nem todos os seus aspectos estão definidos precisamente, optamos por adotar um subconjunto de UML preciso, o chamado pUML. Desta forma, salientamos que utilizamos apenas um núcleo do UML e partes de OCL relacionadas - invariantes, pré e pós-condições. A semântica de OCL ainda não foi padronizada pela OMG e Grupos de pesquisadores internacionais. Desde a criação do UML, estes pesquisadores tentam incorporar precisão a essa linguagem UML e à linguagem de restrições de objetos OCL. Não realizamos o mapeamento no sentido reverso, ou seja, de pUML/OCL $\rightarrow$ i*, pelas mesmas razões apontadas.

A incorporação de precisão aos modelos da UML é uma área de pesquisa ainda em desenvolvimento, que vem sendo trabalhada pelo grupo de pUML [pG99]. Assim, os resultados obtidos com as diretrizes de mapeamento ainda são incipientes, sendo possível, com o uso das diretrizes propostas, chegar-se a um modelo de classes em pUML que,

como todo modelo, é passível de refinamentos, onde então podemos aplicar a precisão da linguagem de restrição de objetos OCL.

CONSIDERAÇÕES

Neste Capítulo foram propostas diretrizes que visam o mapeamento dos requisitos organizacionais para as especificações precisas, que, normalmente, tratam dos requisitos funcionais. Para a modelagem organizacional utilizamos a técnica *i** para modelar os requisitos organizacionais iniciais, através do Modelo de Dependências Estratégicas (SD) e do Modelo de Razões Estratégicas (SR). Para a descrição dos requisitos de forma mais precisa, utilizamos as linguagens Structured MAL (Structured Modal Action Logic) e o OCL (Object Constraints Language).

Em linhas gerais, no processo de determinação houve necessidade de oscilarmos nossas investigações neste trabalho entre as duas áreas foco: a de modelagem dos requisitos organizacionais e a de especificação precisa. Com isso, entendemos que nosso conhecimento na área tornou-se mais sólido. Inúmeras são as dificuldades a considerar: desde o entendimento da técnica de modelagem organizacional *i**, com suas aplicações e correlações com TELOS, até a obtenção de um nível mínimo de discernimento para que pudéssemos discutir sobre as linguagens Structured MAL e pUML/OCL. Nesse percurso, tivemos que adentrar pelo caminho das semânticas, bem como preocupar-nos, também, com o problema ainda em aberto, do UML preciso (pUML), para que pudéssemos associar descrições em OCL.

Não estamos propondo um mapeamento completo entre todos os elementos dos modelos estratégicos da técnica *i** e Structured MAL ou pUML/OCL, mas apenas objetivamos oferecer diretrizes para que aspectos organizacionais, relevantes à complementação da especificação funcional do sistema, possam ser considerados em Structured MAL ou pelas restrições em OCL.

Objetiva-se, com esse mapeamento, através da abordagem precisa, a possibilidade de:

- (i) obter melhor precisão sobre o contexto do problema;

- (ii) obter provas de propriedades relacionadas com os aspectos levantados através do entendimento do contexto; e
- (iii) animar as especificações obtidas,

com o objetivo maior de se ter um documento de requisitos bem mais preciso e correto, com relação às necessidades do cliente.

Os benefícios do mapeamento de i* [YU95a] para Structured MAL [KEN93], fundamentalmente são a inclusão de maior precisão aos aspectos organizacionais, permitem-nos:

- (i) melhorar a verificação e validação dos modelos de requisitos, através dos mecanismos de provas encontrados particularmente em Structured MAL;
- (ii) a possibilidade de animação da especificação, através do Structured MAL Animator [ATK91], que ajudará na etapa de validação dos requisitos; e
- (iii) a possibilidade de inclusão dos aspectos temporais e de sequenciamento encontrados em Structured MAL, para incorporar ao modelo i*; por exemplo, o seqüenciamento nas ligações de decomposição de tarefas encontradas no Modelo de Razões Estratégicas (SR).

Contudo, nesta tese não tratamos explicitamente deste potencial, sugerindo apreciação em trabalhos futuros.

Com relação ao mapeamento de i* para OCL, este nos permite fazer uso das possíveis ferramentas de animação que venham a existir (a IBM, através de Steve Cook [IBMOCL] investiga possibilidades de desenvolver ferramenta nesse sentido), para verificar e validar os requisitos elicitados. Por outro lado, através de uma semântica bem definida em OCL, haverá possibilidades de provas de propriedades, fato que nos permitirá definir, adequadamente, as pré e pós-condições das operações a serem realizadas. As operações em OCL têm relação direta com a ocorrência das tarefas na modelagem i* e, de certa forma, as restrições podem determinar uma seqüencialidade na ocorrência das tarefas (caso das ligações de decomposição de tarefa). Destacamos ainda a originalidade do mapeamento por propiciar a consideração de requisitos organizacionais pela modelagem pUML/OCL.

Devemos salientar ainda que os poderes de expressão de Structured MAL e de OCL são distintos, apenas em um núcleo comum podemos encontrar consistência entre as duas linguagens. Por exemplo, quando do mapeamento dos elementos atores, tarefas, recursos, objetivos e objetivos-soft representados na técnica i*, nas duas linguagens, estes foram mapeados segundo princípios comuns:

- (i) atores em i*, mapeados como agentes (objetos) em Structured MAL e como classes (conjunto de objetos) em pUML/OCL;
- (ii) tarefas em i*, mapeadas como ações em Structured MAL e como operações em pUML/OC;
- (iii) recursos em i*, mapeados como agentes (objetos) em Structured MAL e como classes (conjunto de objetos) em pUML/OCL;
- (iv) objetivos e objetivos-soft em i*, mapeados como atributos de um agente em Structured MAL e como atributos de uma classe em pUML/OCL.

Em cada mapeamento, informações adicionais, segundo a linguagem, podem enriquecer e precisar melhor cada um dos elementos. Todavia o básico é similar determinando um morfismo parcial entre essas linguagens precisas, no tocante aos aspectos básicos. Quanto a esse núcleo espera-se não ter problemas de contradições entre o que foi especificado em Structured MAL e em pUML/OCL. A investigação sobre a existência de algum morfismo é deixada para futuros direcionamentos (vide Seção 7.4).

Para melhor evidenciarmos a aplicabilidade da integração proposta, no próximo Capítulo apresentamos estudos de caso para cada uma das integrações, ou seja, a técnica i* com a linguagem MAL e a técnica i* com a linguagem OCL, totalizando 3(três) casos.

CAPÍTULO VI

ESTUDO DE CASOS

Neste Capítulo apresentaremos três estudos de caso, para consolidar os estudos realizados no Capítulo 5. Na Seção 6.1 apresentamos dois estudos de caso para exemplificação do mapeamento entre a técnica i^* e a linguagem formal MAL, enquanto na Seção 6.2 apresentamos um outro estudo de caso para exemplificação do mapeamento entre a técnica i^* e a linguagem precisa pUML/OCL.

6.1 – EXEMPLOS DE DESCRIÇÃO DO MAPEAMENTO ENTRE A TÉCNICA i^* E A LINGUAGEM MAL ESTRUTURADO

Nosso objetivo é oferecer meios para que se possa exemplificar a utilização das diretrizes propostas no Capítulo 5 anterior. Especificamente nesta Seção, através de dois estudos de casos, estaremos nos preocupando com as diretrizes relacionadas com a integração com a linguagem de especificação MAL estruturada (Modal Actions Logic).

6.1.1 – *ENVASADORA DE ÁGUA MINERAL*

Tomamos como primeiro estudo de caso uma pequena indústria no ramo de envase de água mineral, localizada na divisa de duas cidades da Região Metropolitana do Recife. A empresa possui cerca de 65 funcionários que estão em constante treinamento e aperfeiçoamento das técnicas. Sua estrutura organizacional, segundo as técnicas de modelagem organizacional, sob o ponto de vista da Administração, pode ser visualizada na Figura 6.1: trata-se do organograma da empresa, diagrama simples que descreve as posições dentro da organização em termos das funções de negócio e dos níveis de gerenciamento existentes.

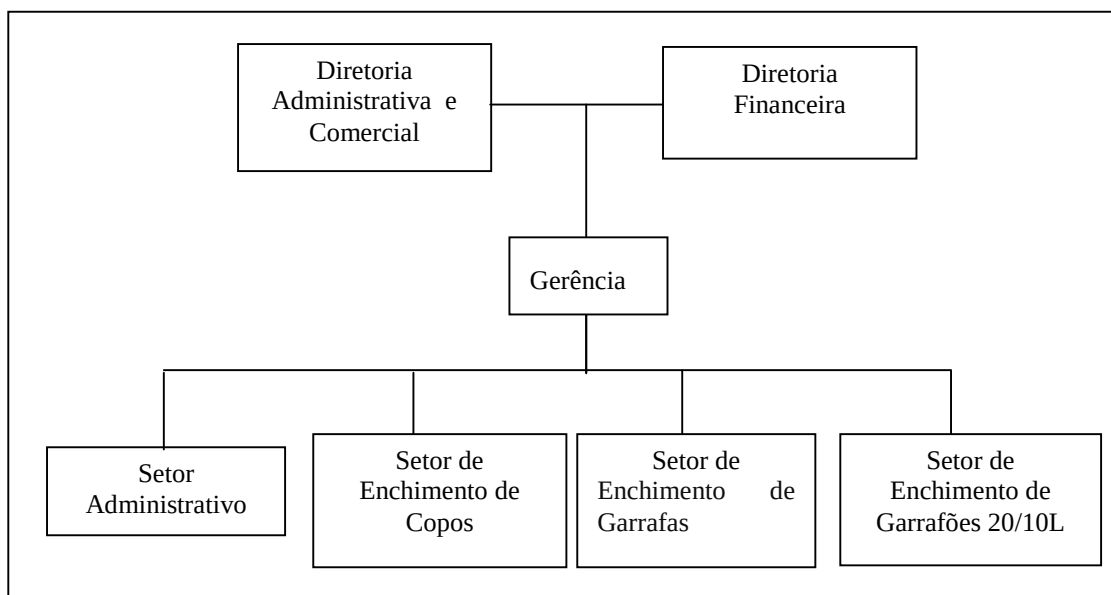


Figura 6.1 – Organograma Setorial da Envasadora de Água Mineral

Atualmente, a empresa possui três linhas de produção:

- (i) garrafões retornáveis de 20 e 10 litros e potes descartáveis de 5 litros;
- (ii) garrafas descartáveis de 300, 500 e 1500 mililitros; e
- (iii) copos descartáveis de 200 e 300 mililitros.

A empresa produz diariamente:

- (i) na primeira linha de produção, cerca de 2.000 garrafões de 10 litros, 7.000 garrafões de 20 litros e 1.000 potes de 5 litros;
- (ii) na segunda linha de produção, cerca de 8.000 garrafas de 300ml, 8.000 garrafas de 500ml e 8.000 garrafas de 1.500 ml; e
- (iii) na terceira linha de produção, cerca de 350 caixas de copos de 200ml e 350 caixas de copos de 300ml.

Neste trabalho, interessou-nos modelar o setor de produção, no tocante ao envase de garrafões de 20 e 10 litros.

No processo de envase, previamente é feito o controle do produto fornecido pelo cliente: no caso em questão, os garrafões de 20 e 10 litros.

Os garrafões para envase, oriundos do cliente, são colocados pelo pessoal do próprio cliente na plataforma, formando uma única pilha, de forma a facilitar a contagem

simultânea por parte do Encarregado da Produção ou do funcionário da produção que o Encarregado determinar e por parte do representante do cliente.

Procedida a contagem e anotada a quantidade, cada garrafão é inspecionado quanto a existência de trincas, tinta, resíduos estranhos e odor. Como resultado da inspeção, tem-se que:

- (i) os garrafões não conformes, ou seja, os que possuam trincas, tinta, resíduos estranhos ou odor são devolvidos, no ato, ao cliente; e
- (ii) os conformes são endereçados ao setor responsável pela limpeza externa dos garrafões.

Nesta fase, os garrafões são novamente inspecionados quanto à existência de não conformidades (odor e resíduos) e como resultado, tem-se que os garrafões não conformes são devolvido, no ato, ao cliente ou ao seu representante, enquanto que os garrafões conformes, em havendo tampa, lacre e/ou rótulo, estes são retirados.

Na sequência, os vasilhames são submetidos à lavagem e em seguida ao enxágüe onde é feita nova inspeção. Caso ainda sejam detectadas não conformidades – garrafões com resíduos estranhos e odor, estes garrafões são devolvidos ao cliente ou seu representante, enquanto que os garrafões conformes seguem para o próximo procedimento, o envase.

Ao final do envase, os garrafões são tampados, lacrados e rotulados e, em seguida, entregues ao cliente. O Encarregado de Produção expede e entrega ao representante do cliente, o relatório de envase desse cliente, que contém, entre outros, dados relativos aos garrafões conformes e não conformes da operação.

Com base em documentação e em visitas e entrevistas técnicas no local de trabalho, utilizando a técnica i*, realizamos várias modelagens da organização, que ao longo do tempo foram sendo gradativamente refinadas. Assim, na Figura 6.2, podemos observar o Modelo de Dependências Estratégicas, no qual observamos os atores envolvidos e o relacionamento externo entre esses atores.

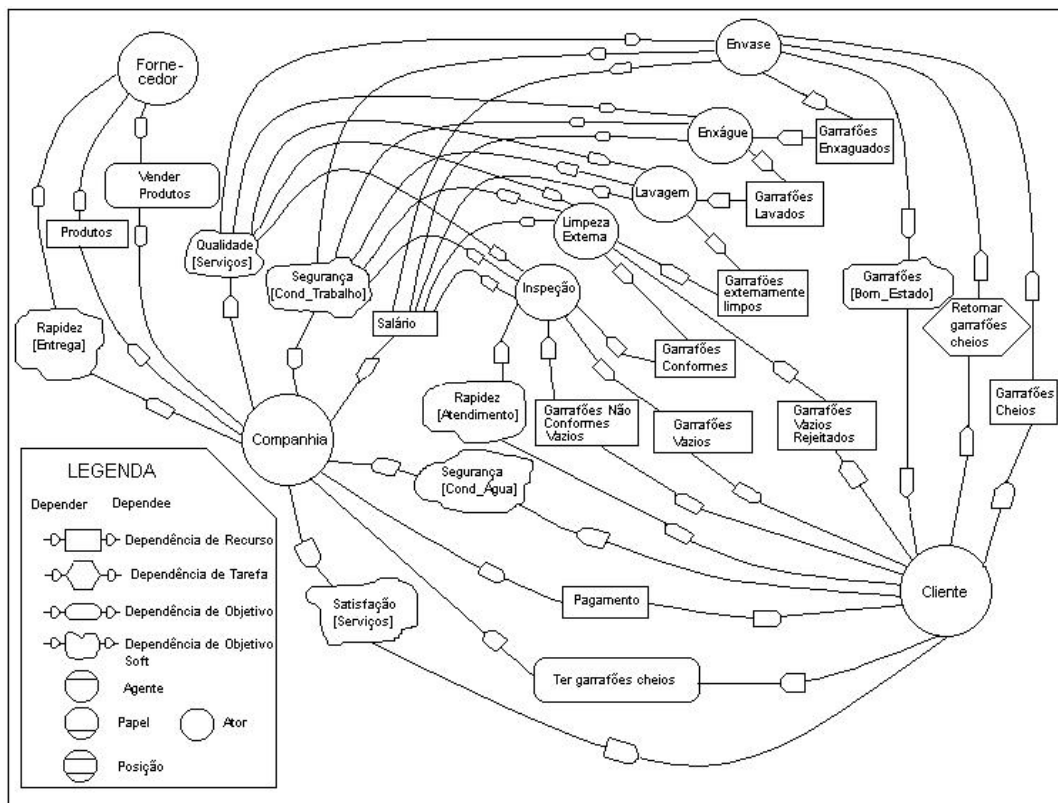


Figura 6.2 – Modelo de Dependências Estratégicas da Envasadora de Água Mineral

Da observação do Modelo de Dependências Estratégicas (SD) da Figura 6.2, identificamos a existência dos seguintes atores: *Cliente*, *Companhia*, *Fornecedor*, *Inspeção*, *Limpeza_Externa*, *Lavagem*, *Enxágue* e *Envase*. Não tratamos, nesta tese, da “especialização” dos atores em agentes, posições e papéis, conforme relatado no Capítulo 5.

O *Cliente* espera ter seus garrações cheios com água mineral: este fato é expresso através da dependência de objetivo “*Ter garrações cheios*”. Por meio de uma dependência de objetivo soft, esse mesmo *Cliente* deseja que a água, entregue pela *Companhia*, possua condições seguras, “*Segurança[Cond_Água]*”, isto é, que seja uma água apropriada para consumo. Ainda sob o ponto de vista do *Cliente*, este espera ser atendido rapidamente pela *Inspeção*, desejo este expresso pela dependência de objetivo soft, “*Rápido[Atendimento]*”, e ter seus garrações cheios de água mineral, fato esse expresso pela dependência de recurso, “*Garrações Cheios*”, com o *Envase*. O *Cliente* ainda depende que o *Envase*, ao final do processo, passe-lhe os garrações cheios, fato

esse expresso através da dependência de tarefa, “*Retornar garrações cheios*”; e que, durante o processo de envase, o Cliente tenha os “*Garrações Não Conformes Vazios*” a ele devolvidos pelos funcionários da *Inspeção* e os “*Garrações Vazios Rejeitados*” devolvidos pelos funcionários da *Limpeza Externa*. Estes dois relacionamentos são expressos pelas dependências de recursos.

Por seu lado, a *Companhia* espera a satisfação de seus *Clientes* quanto aos serviços oferecidos, expresso no modelo da Figura 6.2 pela dependência de objetivo soft “*Satisfação[Serviços]*”. Preocupando-se ainda com os serviços realizados pelos seus *Empregados*, expressa essa preocupação pela dependência de objetivo soft, “*Qualidade[Serviços]*”. Todavia, para manter seus serviços e, conseqüentemente, seus custos, a *Companhia* espera que o *Cliente* pague por seus serviços, sendo esse fato representado pela dependência de recurso “*Pagamento*”.

Os atores *Inspeção*, *Limpeza Externa*, *Lavagem*, *Enxágue* e *Envase*, por suas vezes, dependem da *Companhia* para manter as condições de trabalho seguras, “*Segurança[Cond_Trabalho]*”, ao tempo que a *Companhia* depende deste para manter a qualidade dos serviços, “*Qualidade[Serviços]*”. Por seu turno, esses atores dependem da *Companhia* para receberem seus salários, “*Salário*”. Por outro lado, para que o ator *Inspeção* realize seu trabalho, depende de que o *Cliente* entregue os garrações vazios e em bom estado, respectivamente, “*Garrações Vazios*” e “*Garrações[Bom_Estado]*”.

Com base no Modelo de Dependências Estratégicas da Figura 6.2, podemos proceder ao processo de mapeamento com a linguagem de especificação precisa MAL Estruturado. Como exemplo, concentremo-nos apenas em uma parte do modelo, mesmo porque não é nossa intenção promover o mapeamento total do modelo para MAL. Assim sendo, vamos considerar os agentes “*Inspeção*” e “*Limpeza Externa*”, porque estes terão seu comportamento detalhado pelo Modelo de Razões Estratégicas (SR), vide Figura 6.4, mais adiante.

No processo de aplicação das diretrizes apontadas no Capítulo anterior, apenas relembremos as diretrizes principais, e, caso haja alguma subdiretriz, só faremos a indicação.

Inicialmente, vamos aplicar a diretriz M1, que se aplica aos atores encontrados na empresa envasadora de água mineral.

Diretriz M1: Relacionada com o mapeamento dos atores do modelo i*.

*Os **Atores** no Modelo de Dependências Estratégicas (SD) podem ser entendidos como **Agentes** em MAL estruturado.*

De acordo com o Modelo de Dependências Estratégicas da Figura 6.2, no tocante ao envase de garrações retornáveis de 20/10L, a envasadora de água mineral é composta por nove atores, mapeados individualmente para MAL Estruturado segundo a diretriz M1. Ainda com base na Figura 6.2, a empresa “Envasadora de Água Mineral” pode ser representada como um ator composto, mapeado em MAL Estruturado segundo a aplicação da subdiretriz M1.1. Na Figura 6.3, vê-se o agente “Envasadora de Água Mineral” e a estrutura de seus agentes componentes, em MAL Estruturado.

Neste ponto, antes de procedermos à aplicação das demais diretrizes, faz-se necessário apresentarmos o Modelo de Razões Estratégicas (SR), dado que os mesmos elementos estruturais – tarefas, recursos, objetivos e objetivos-soft – também estarão presentes neste modelo. Assim, na Figura 6.4, tem-se esse modelo para a Envasadora de Água Mineral. Lembramos que estaremos focando no processo de envasamento, avaliando o comportamento interno dos agentes envolvidos, segundo o Modelo de Razões Estratégicas (SR), visto na Seção 3.3.2.

Com relação ao ator Inspeção, Figura 6.4, vemos que este é o responsável por realizar a primeira inspeção visual dos garrações que chegam dos clientes à empresa, fato este representado pela tarefa “*Fazer Inspeção Visual*”. Para a realização desta tarefa faz-se necessário ter “Salário”, ter “Segurança[Cond_Trabalho]”, “*Receber Garrações Vazios*” e “*Observar Não Conformidades*”, objetivando-se com isso “*Detectar Garrações Não Conformes*”. Esta é uma situação típica de uma decomposição de tarefa, conforme pode ser visualizado na Figura 6.4.

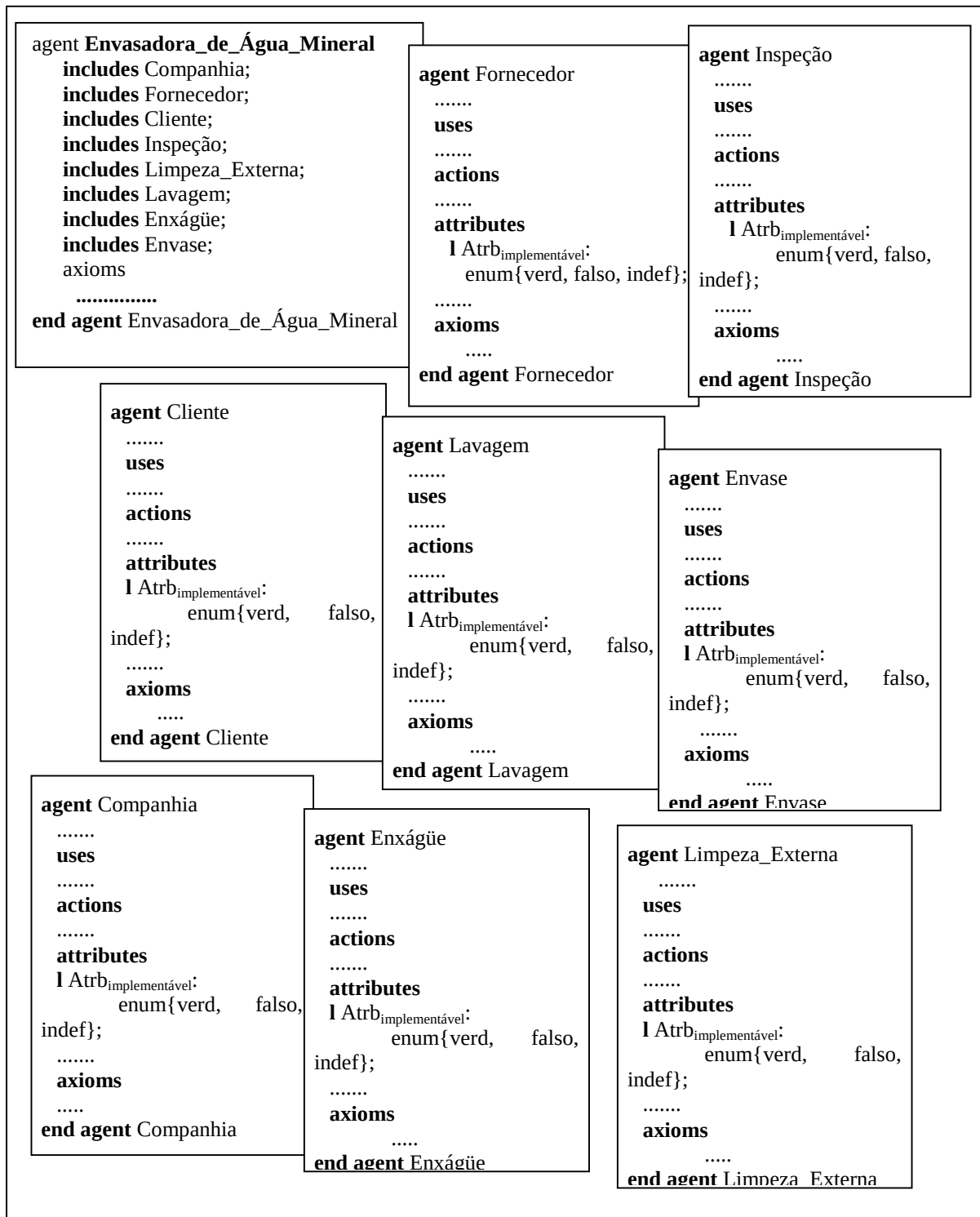


Figura 6.3 – Definição em MAL Estruturado da Envasadora de Água Mineral com seus Agentes Componentes

superficial. A aceitação dos garrafões, que passaram na inspeção, é representada pela tarefa “*Aceitar Garrafões*”, que consistirá na realização de outras duas tarefas:

- (i) contar os garrafões aceitos, “*Contar Garrafões*”; e
- (ii) passar os garrafões contados para a segunda fase do envase, “*Passa Garrafões*”.

Assim o agente “*Limpeza Externa*” dependerá do recurso “*Garrafões Conformes*”, fornecido pelo agente “*Inspeção*”.

Nesse ponto, passamos à avaliação do ator “*Limpeza Externa*”. No comportamento desse ator, segundo a Figura 6.4, temos que a tarefa de “*Fazer Limpeza Externa*” consistirá em receber garrafão, ter “*Salário*”, ter “*Segurança[Cond_Trabalho]*”, “*Receber Garrafões Conformes*”, e realizar uma nova inspeção, “*Nova Inspeção Visual*”. Esta última tarefa poderá resultar em:

- (i) rejeição dos garrafões, expresso pela tarefa “*Rejeitar Garrafões*”, que retornará os garrafões vazios rejeitados ao cliente; ou
- (ii)** aceitação dos garrafões, expresso pela tarefa “*Aceitar Garrafões*”.

Essa tarefa, então, consistirá:

- (i) na retirada do rótulo ainda existente nos garrafões, expressa pela tarefa “*Tirar Rótulo*”;
- (ii) na retirada de lacre, caso ainda exista, expressa pela tarefa “*Tirar Lacre*”;
- (iii) na retirada da tampa ainda existente nos garrafões, expressa pela tarefa “*Tirar Tampa*”; e
- (iv)** na passagem dos garrafões limpos externamente, expressa pela tarefa “*Passar Garrafões Limpos*”, para a lavagem interna, a qual será realizada por um outro agente.

Assim, neste ator *Limpeza Externa*, também foram observados a decomposição de tarefa e o relacionamento meio-fim, principais novos aspectos apresentados pelo Modelo de Razões Estratégicas (SR) da técnica *i**, não encontrados no Modelo de Dependências Estratégicas (SD), Figura 6.2.

Uma vez entendido o comportamento interno de alguns agentes da linha de produção de envase de garrafões de 20/10 litros, passaremos a fazer a aplicação das diretrizes de mapeamento propostas no Capítulo anterior.

Diretriz M2: Encontra-se relacionada com o componente **tarefas**, nos modelos estratégicos da técnica i*.

*Todas as **tarefas** dos modelos estratégicos i* serão mapeadas como **ações** de um agente em MAL estruturado.*

Todas as tarefas encontradas no modelo estratégico SD, Figura 6.2, serão representadas como **ações** do agente que tem a responsabilidade por executá-la, aplicando-se a subdiretriz M2.1, enquanto no modelo estratégico SR, Figura 6.4, serão representadas como **ações** do agente que esteja sendo detalhado, aplicando-se a subdiretriz M2.2. As ações podem ser de uso apenas de um agente, nesse caso rotuladas como **l** (local) ou compartilhadas com outros agentes e rotuladas com **s** (shared).

Considerando-se os agentes “Inspeção” e “Limpeza_Externa”, na Figura 6.5, tem-se um trecho de especificação em MAL estruturado, fruto da aplicação da diretriz em questão., ou seja, todas as tarefas internas são mapeadas como ações locais.

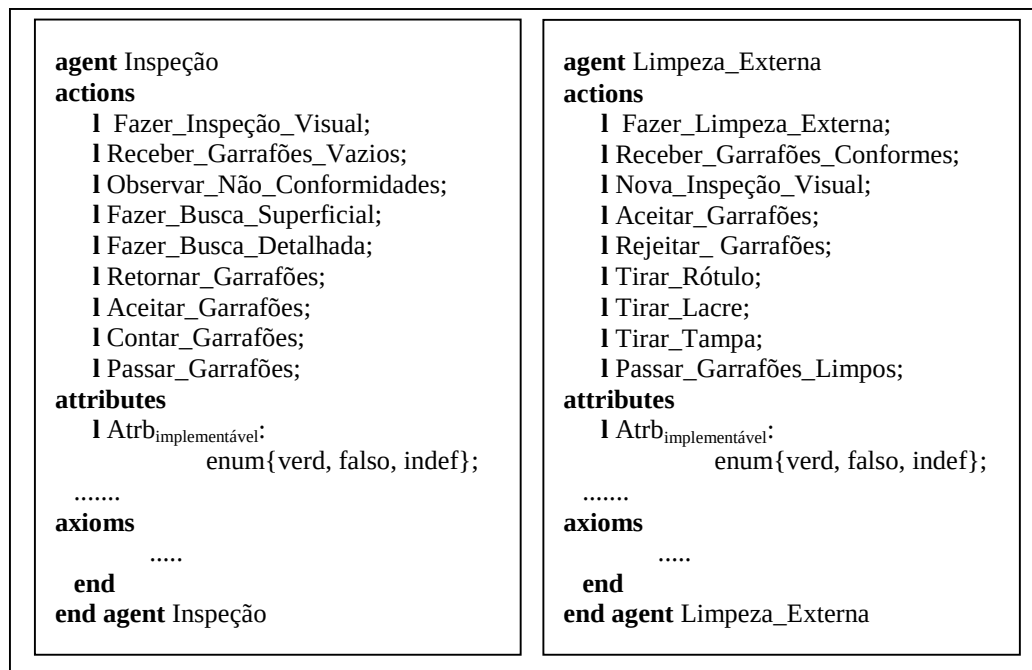


Figura 6.5 - Especificando Tarefas em i*, da Envasadora de Água Mineral, como Ações em MAL Estruturado

Observe que só existem ações relacionadas com o comportamento interno dos agentes considerados; do ponto de vista dos relacionamentos externos, não existe qualquer dependência de tarefa: daí todas as ações terem sido rotuladas como locais (I).

Continuando com esses mesmos agentes, passaremos à aplicação da diretriz M3.

Diretriz M3: Encontra-se relacionada com componente **recurso**, nos modelos estratégicos da técnica i*.

*Os **recursos** dos modelos estratégicos serão traduzidos como **agentes** em MAL estruturado, com um **atributo compartilhado** ($atrb_{rec}$) do tipo **booleano**, que indique a disponibilidade desse recurso;*

Observando-se os modelos estratégicos, Figuras 6.2 e 6.4, identificamos os recursos:

- (i) *Produtos*, que representam as tampas, rótulos, lacres, etc. que são comprados de um fornecedor;
- (ii) *Salário*, pagamento feito pelo serviço aos empregados da empresa;
- (iii) *Pagamento*, valor pago pelo Cliente, referente ao envase dos garrações;
- (iv) *Garrações Vazios*, são os garrações trazidos pelo Cliente para envase;
- (v) *GarraçõesNãoConformes*, são garrações que possuem algum odor, rachadura, resíduo, etc. e que não passaram na inspeção;
- (vi) *GarraçõesConformes*, são os garrações que passaram na primeira inspeção;
- (vii) *GarraçõesVaziosRejeitados*, são os garrações que não passaram numa segunda inspeção, observando-se ainda coloração estranha ou odor;
- (viii) *GarraçõesExternamenteLimpos*, são os garrações que passaram nas duas inspeções e que foram, por isso, externamente limpos;
- (ix) *GarraçõesLavados*, representam os garrações, que foram lavados internamente com água clorada;
- (x) *GarraçõesEnxaguados*, são os garrações que estão totalmente limpos e em condições para o envase; e

(xi) *GarraçõesCheios*, que são aqueles garrações envasados com água mineral e pelos quais o Cliente faz o pagamento.

Desses recursos, apenas representamos na Figura 6.6, os que fazem parte do processo de envase, dado que o procedimento é o mesmo para todos os demais. Observemos que, a cada um dos agentes que mapeam um recurso, corresponde um atributo compartilhado do tipo booleano, que assinalará a disponibilidade ou não desse recurso. Adotamos o nome do atributo o mais próximo do nome do recurso, apenas para facilitar a compreensão, por exemplo para “GarraçõesVaziosRejeitados” denominamos seu atributo de “GarVaziosRej”. A presença de um recurso, nos modelos estratégicos do i^* , apenas assinala a necessidade da existência dos mesmos, para a realização de alguma tarefa ou para se alcançar algum objetivo.

Tendo mapeado os atores, as tarefas e os recursos, resta-nos mapear os elementos *objetivos bem definidos* e os *objetivos-soft*, ambos tratados pela diretriz M4 e suas subdiretrizes.

Diretriz M4: Encontra-se relacionada com os componentes **objetivos bem definidos** e **objetivos soft**, nos modelos estratégicos da técnica i^* .

Os **objetivos bem definidos** e os **objetivos soft** dos modelos estratégicos serão mapeados como **atributos do tipo booleano e do tipo enumerado**, respectivamente, em um agente MAL estruturado;

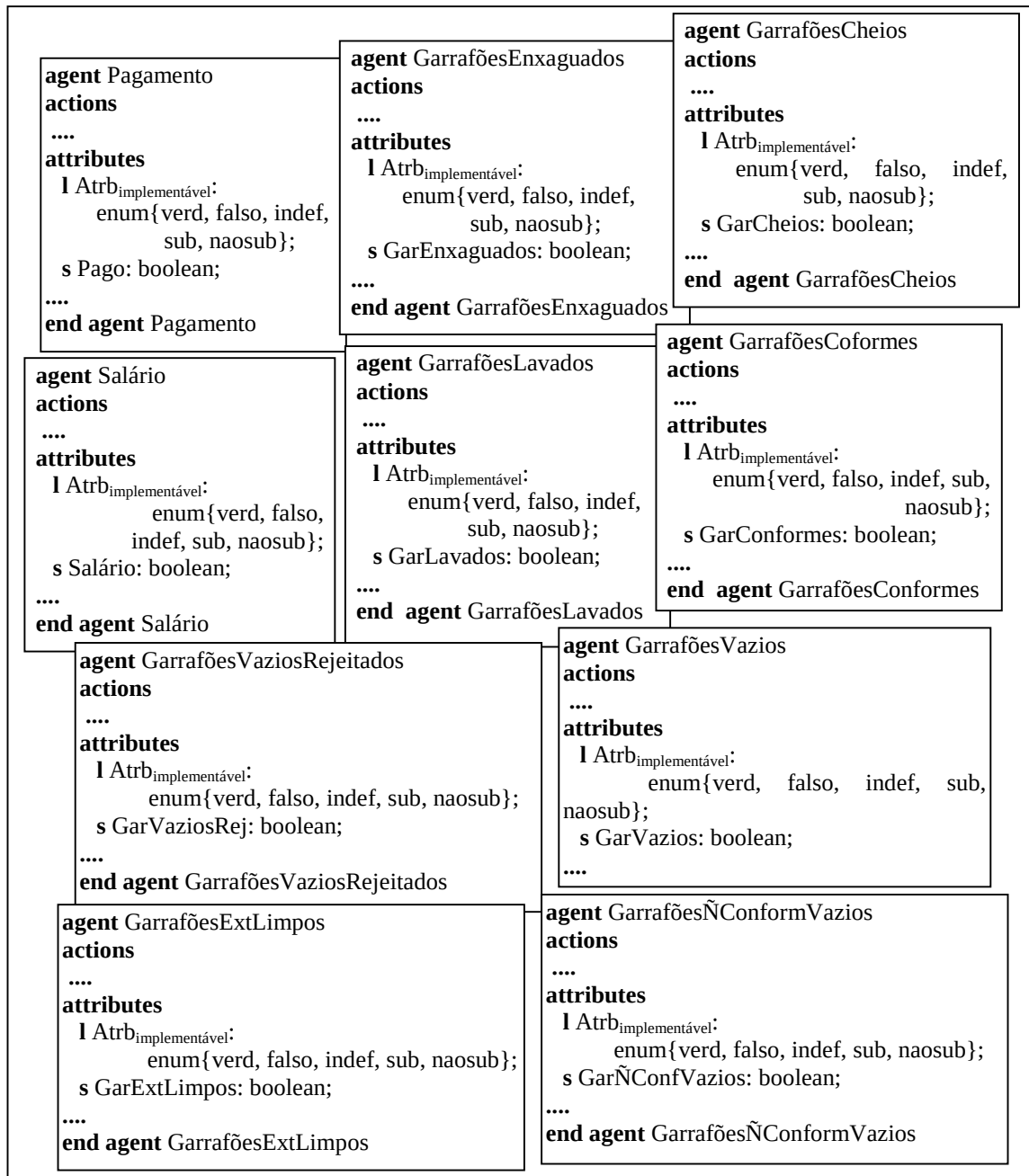


Figura 6.6 - Especificando Recursos em i*, da Envasadora de Água Mineral, como Agentes em MAL Estruturado

Observando o Modelo de Dependências Estratégicas da Figura 6.4, vemos que entre os atores Cliente e Inspeção existe um objetivo-soft, “*Rapidez[Atendimento]*” que, segundo a diretriz, é mapeado como um atributo enumerado no agente Cliente, cujo valor poderá ser verdadeiro, falso ou indefinido (vide figura 6.7, ator Cliente).

Ainda observando o relacionamento entre o ator *Cliente* e os demais atores, do modelo estratégico SD, Figura 6.2, temos:

- (i) uma dependência de objetivo-soft com e o ator *Envase*,
“*Garrações[Bom_Estado]*”, tratando-se de um objetivo do ator *Envase* e, portanto, mapeado como atributo enumerado deste;
- (ii) uma dependência de objetivo-soft com o ator *Companhia*,
“*Segurança[Cond_Água]*”, tratando-se de um objetivo do ator *Cliente*, evidenciando sua preocupação quanto a garantia de qualidade da água mineral;
- (iii) uma dependência de objetivo-soft com o ator *Companhia*,
“*Satisfação[Serviços]*”, no caso, um objetivo da *Companhia*; e
- (iv) um objetivo bem definido, “*TerGarraçõesCheios*”, que corresponde a um desejo bem claro, e, possivelmente alcançável, por parte do *Cliente*.

Com relação aos atores “*Inspeção*”, “*Limpeza Externa*”, “*Lavagem*”, “*Enxágue*” e “*Envase*”, todos possuem um objetivo-soft com relação a *Companhia*, “*Segurança[Cond_Trabalho]*”, que será mapeado como um atributo público do tipo enumerado em cada uma das especificações destes atores, em MAL Estruturado. Por outro lado, a “*Companhia*”, espera que cada um desses atores realizem suas atividades com qualidade, “*Qualidade[Serviço]*”, um objetivo-soft, que segundo M4, deve ser mapeado com um atributo público do tipo enumerado, no corpo da especificação do agente “*Companhia*”.

Conforme o Modelo de Razões Estratégicas, Figura 6.4, verificamos que apenas o ator *Inspeção*, possui um sub-objetivo, “*DetetarGarraçõesNãoConformes*”, que conforme a subdiretriz M4.2, será mapeado como um atributo local, do tipo booleano, na especificação do agente *Inspeção*. O mapeamento desses objetivos, segundo a diretriz M4 e suas subdiretrizes ,pode ser visto na Figura 6.7.

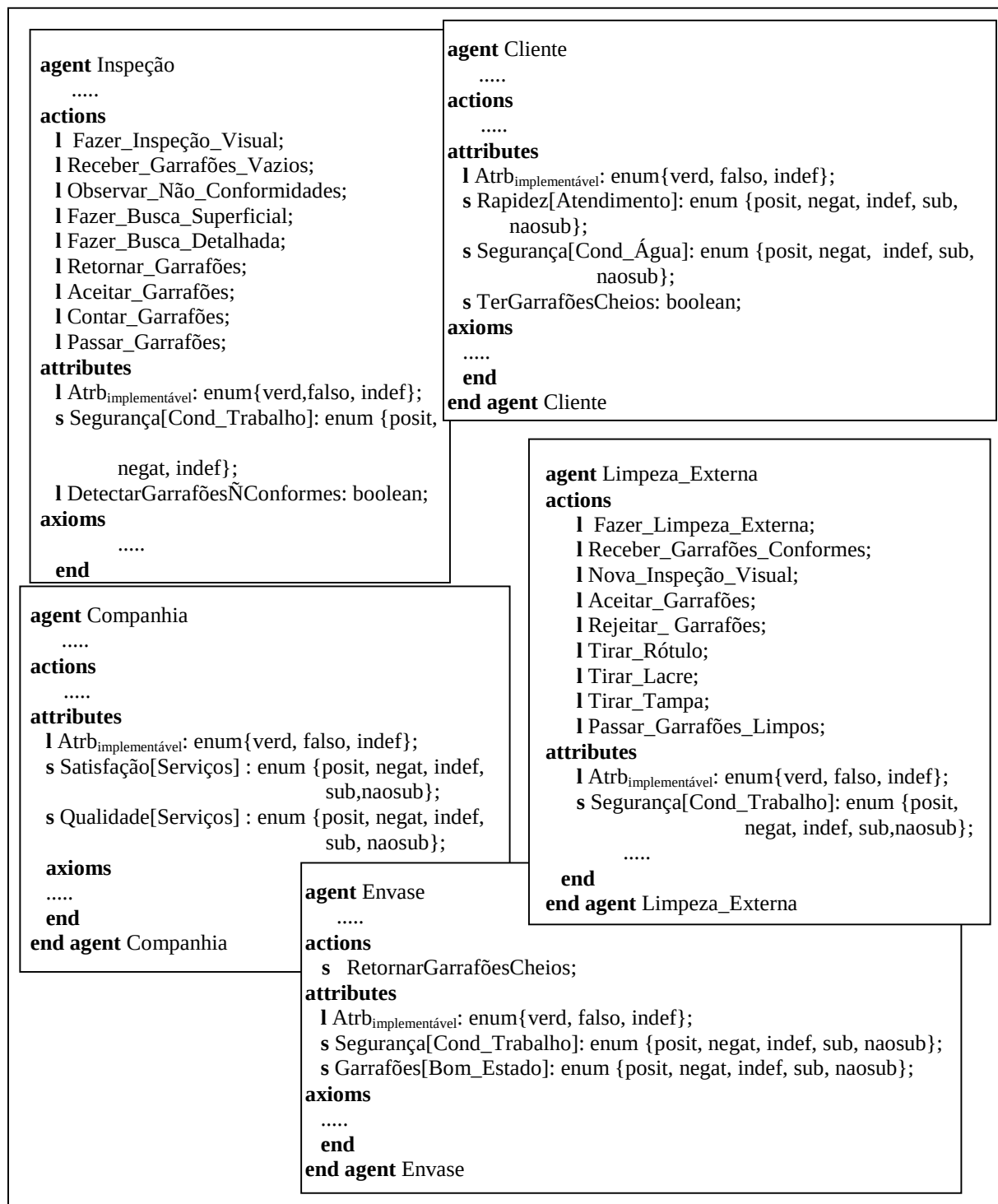


Figura 6.7 - Especificando Objetivos e Objetivos-Soft em i*, da Envasadora de Água Mineral, como Atributos Booleanos e Enumerados em MAL Estruturado

Neste ponto, passaremos a considerar os relacionamentos externos entre os agentes, identificados no Modelo SD da técnica i*, Figura 6.2, para procedermos à aplicação da diretriz M5.

Diretriz M5: Trata dos relacionamentos externos entre os atores do i*;

Os **relacionamentos** externos no Modelo de Dependências Estratégicas (SD) podem ser tratados pelos **axiomas**, em MAL estruturado.

s axiomas, em uma especificação MAL estruturado, estabelecem o comportamento dos agentes (objetos), considerando todos os elementos componentes das dependências (tarefas, recursos, objetivos e objetivos-soft). Falta estabelecer as restrições de ocorrência das ações (tarefas). Observemos, na Figura 6.2, o relacionamento entre os agentes “Cliente” e “Envase”: identificamos um relacionamento de dependência de tarefa, de objetivo-soft e de recurso. Procuraremos colocar restrições associadas à ação correspondente, através dos axiomas, Figura 6.8.

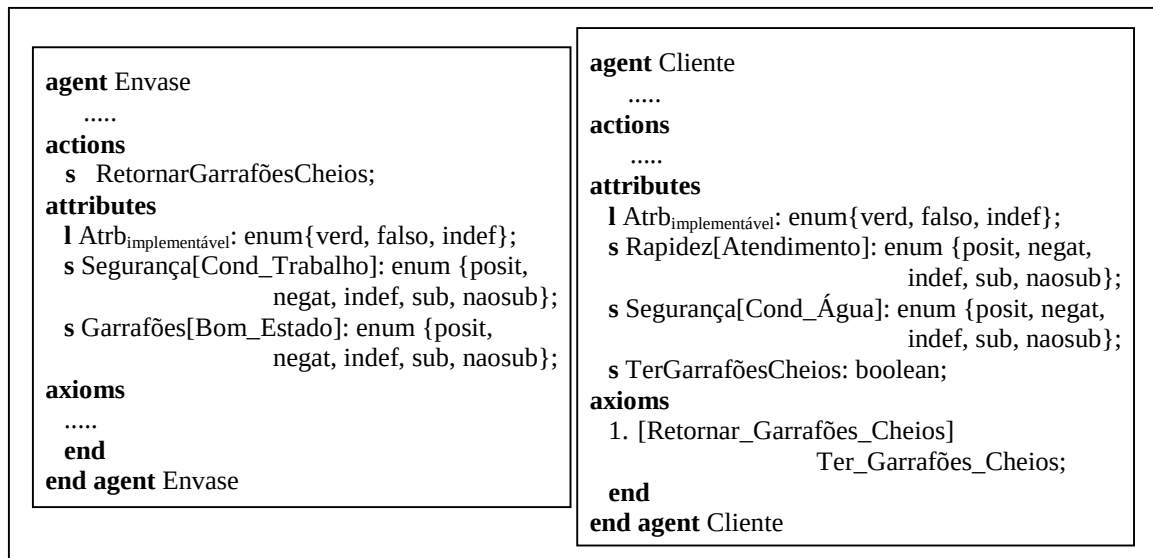


Figura 6.8 – Um Relacionamento Externo Representado em MAL através de Axiomas

Como o Cliente é o ator dependente de envase, para que a tarefa “RetornarGarraçõesCheios” seja satisfeita (vide Figura 6.2), o axioma 1 da Figura 6.10 retrata essa situação, quando especifica que, após a realização da ação de “Retornar_Garrações_Cheios”, o Cliente terá o seu objetivo, “Ter_Garrações_Cheios”,

satisfeito (representa uma pós-condição). A tarefa “*Retornar_Garrações_Cheios*” é de responsabilidade do agente Envase e está relacionada com o comportamento interno deste agente. Assim, após detalhar-se mais esse comportamento interno, poder-se-á acrescentar, no agente Envase, outros axiomas que definam, mais precisamente, esta ação de sua responsabilidade.

No tocante ao Modelo de Razões Estratégicas (SR) da Figura 6.4, são identificados os mesmos elementos componentes das dependências estratégicas (tarefa, recuso, objetivo e objetivo-soft) e dois outros relacionamentos adicionais, do tipo decomposição de tarefa e do tipo meio-fim. As duas últimas diretrizes M6 e M7, apontadas no Capítulo 5, tratam desses aspectos. Essas diretrizes serão aplicadas aos agentes “*Inspeção*” e “*Limpeza Externa*”, cujos comportamentos internos são explicitado na Figura 6.4.

Diretriz M6: Trata das ligações do tipo decomposição de tarefa;

Uma ligação do tipo decomposição de tarefa é representada, em MAL estruturado, através de um axioma de ação contendo o conectivo lógico AND e o operador deôntico obl aplicado às ações que possam existir, de acordo com a decomposição da tarefa em sub-objetivo, sub-tarefa, recursoPara ou objetivo-softPara.

No agente *Inspeção*, da Figura 6.4, encontramos duas tarefas decompostas:

- (i) “*Fazer_Inspeção_Visual*”, decomposta no objetivo bem definido, “*DetetarGarraçõesNãoConformes*”, no objetivo-soft “*Segurança [Cond_Trabalho]*”, no recurso “*Salário*” e nas tarefas “*Receber_Garrações_Vazios*” e “*ObservarNãoConformidades*”; e
- (ii) *Aceitar_Garrações*, decomposta em duas tarefas: “*Contar_Garrações*” e “*Passar_Garrações*”.

Desta forma, para a primeira decomposição, tem-se o axioma:

$$[Fazer_Inspeção_Visual] (DetectarGarraçõesNãoConformes \wedge Salário \wedge (Segurança[Cond_Trabalho]= posit) \wedge obl(Receber_Garrações_Vazios)) \wedge obl(ObservarNãoConformidades))$$

que na Figura 6.9, corresponderá ao axioma 1. Da mesma forma, para a segunda decomposição, temos:

$$[Aceitar_Garra\tilde{f}ões] (obl(Contar_Garra\tilde{f}ões) \wedge obl(Passar_Garra\tilde{f}ões))$$

Este último axioma corresponde, no trecho de especificação em MAL Estruturado da Figura 6.9, ao axioma 2.

```

agent Inspeção
actions
  l Fazer_Inspeção_Visual;
  l Receber_Garrações_Vazios;
  l Observar_Não_Conformidades;
  l Fazer_Busca_Superficial;
  l Fazer_Busca_Detalhada;
  l Retornar_Garrações;
  l Aceitar_Garrações;
  l Contar_Garrações;
  l Passar_Garrações;
end
attributes
  l Atribimplementável: enum{verd, falso, indef};
  l Detectar_Garrações_Não_Conformes: boolean;
  s Segurança[Cond_Trabalho]: enum {posit, negat, indef, sub, naosub};
axioms
  1. [Fazer_Inspeção_Visual] (Detectar_Garrações_Não_Conformes  $\wedge$ 
                               Salário  $\wedge$  (Segurança[Cond_Trabalho] = posit)  $\wedge$ 
                               obl(Receber_Garrações_Vazios)  $\wedge$ 
                               obl(Observar_Não_Conformidades));
  2. [Aceitar_Garrações] (obl (Contar_Garrações)  $\wedge$ 
                           obl (Passar_Garrações));
end agent Inspeção

```

Figura 6.9 - Axiomas para as Ligações de Decomposição de Tarefa do Agente Inspeção, da Envasadora de Água Mineral

No comportamento interno do ator “Limpeza_Externa”, identificamos duas decomposições de tarefas:

- (i) “Fazer_Limpeza_Externa”, decomposta em duas tarefas: “Receber_Garrações_Conformes” e “Nova_Inspeção_Visual”; em um objetivo-soft “Segurança[Cond_Trabalho]” e em um recurso “Salário”; e

- (ii) “Aceitar_Garrações”, decomposta em quatro tarefas: “Tirar_Rótulo”, “Tirar_Lacre”, “Tirar_Tampa” e “Passar_Garrações_Limpos”.

.Para a primeira decomposição, tem-se o seguinte axioma:

$$[Fazer_Limpeza_Externa](Salário \wedge (Segurança[Cond_Trabalho] = posit) \wedge \\ obl(Receber_Garrações_Conformes \wedge obl(Nova_Inspeção_Visual)))$$

Assim, esta situação será expressa no axioma 1, da Figura 6.10. Da mesma forma, para a segunda decomposição, temos:

$$[Aceitar_Garrações](obl(Tirar_Rótulo) \wedge obl(Tirar_Lacre) \wedge obl(Tirar_Tampa) \wedge \\ obl(Passar_Garrações_Limpos))$$

Este último axioma podem ser vistos no trecho de especificação em MAL Estruturado, para o agente Limpeza_Externa, Figura 6.10, como o axioma 2..

```
agent Limpeza_Externa
actions
actions
  l Fazer_Limpeza_Externa;
  l Receber_Garrações_Conformes;
  l Nova_Inspeção_Visual;
  l Aceitar_Garrações;
  l Rejeitar_Garrações;
  l Tirar_Rótulo;
  l Tirar_Lacre;
  l Tirar_Tampa;
  l Passar_Garrações_Limpos;
attributes
  l Atribimplementável: enum{verd, falso, indef};
  s Segurança[Cond_Trabalho]: enum {posit, negat, indef, sub, naosub};
  ....
axioms
1. [Fazer_Limpeza_Externa] (Salário ∧
                               (Segurança[Cond_Trabalho] = posit) ∧
                               obl(Receber_Garrações_Conformes) ∧
                               obl(Nova_Inspeção_Visual));
2. [Aceitar_Garrações] (obl (Tirar_Rótulo) ∧ obl (Tirar_Lacre) ∧
                        obl(Tirar_Tampa) ∧ obl(Passar_Garrações_Limpos) );
end agent Limpeza_Externa
```

Figura 6.10 - Axiomas para as Ligações de Decomposição de Tarefa do Agente Limpeza_Externa, da Envasadora de Água Mineral

Consideremos o outro tipo de ligação encontrada nos Modelo de Razões Estratégicas: o relacionamento meio-fim, onde, em geral, se tem como meio, tarefas possibilitam várias alternativas para se alcançar um fim.

Diretriz M7: Trata das ligações do tipo meio-fim;

*Uma ligação do tipo **meio-fim** é representada, em MAL estruturado, através de axiomas.*

SubDiretriz M7.1: Trata das ligações do tipo meio-fim, onde o meio é uma tarefa e o fim pode ser tarefa, objetivo bem definido, objetivo-soft ou recurso.

*Uma ligação do tipo **meio-fim**, onde o meio é uma tarefa, é representada, em MAL estruturado, através de um **axioma de ação** contendo o conectivo lógico **OR** e o operador deôntico **obl** aplicado às **ações** que possam existir, em função de ser o fim uma tarefa, um objetivo bem definido, um objetivo-soft ou um recurso.*

Observado a Figura 6.4, com respeito aos agentes “Inspeção” e “Limpeza-Externa”, tem-se um relacionamento meio-fim onde o meio e o fim são tarefas, pelo que devemos aplicar a subdiretriz M7.1.No agente “Inspeção”, tem-se:

- (i) a tarefa “*Observar Não Conformidades*”, como fim e dois meios de alcançá-la, representados pelas tarefas “*Fazer Busca Detalhada*” e “*Fazer Busca Superficial*”;

Assim, pela aplicação de M7.1, temos o axioma

[Observar_Não_Conformidades] (**obl**(Fazer_Busca_Detalhada)∨
obl(Fazer_Busca_Superficial));

que no corpo da especificação MAL Estruturado do agente “Inspeção”, corresponde ao axioma 3.

- (ii) a tarefa “*Fazer Busca Detalhada*” tem com alternativas duas outras tarefas “*Retornar Garrações*” ou “*Aceitar Garrações*”;

Assim, pela aplicação de M7.1, temos o axioma

[Fazer Busca Detalhada] (**obl**(Fazer_Busca_Detalhada) $\vee$ **obl**(Fazer Busca Superficial));

que no corpo da especificação MAL Estruturado do agente “Inspeção”,
corresponde ao axioma 4.

- (iii) a tarefa “Fazer Busca Superficial” tem com alternativa duas outras tarefas “Aceitar Garrações” ou “Retornar Garrações”

Assim, pela aplicação de M7.1, temos o axioma

[Fazer Busca Superficial] (**obl**(Fazer_Busca_Detalhada) $\vee$

obl(Fazer Busca Superficial));

que no corpo da especificação MAL Estruturado do agente “Inspeção”,
corresponde ao axioma 5.

- (iv) a tarefa “Fazer Inspeção”, pelo relacionamento externo com outros atores possui como fim dois objetivo-soft: “Qualidade[Serviço]” e “Rapidez[Atendimento]”. Assim, pela aplicação de M7.1, temos o axioma

[Fazer Inspeção](((Qualidade[Serviço] = posit) $\wedge$

(Rapidez[Atendimento] = posit));

que no corpo da especificação MAL Estruturado do agente “Inspeção”,
corresponde ao axioma 6.

Esses quatro axiomas serão incorporados à especificação do Agente Inspeção, em MAL estruturado, Figura 6.11.

Procedendo-se de forma semelhante para a agente *Limpeza_Externa*, identificamos no Modelo de Razões Estratégicas, Figura 6.4, três ligações meio-fim onde:

```

agent Inspeção
actions
  l Fazer_Inspeção_Visual;
  l Receber_Garrações_Vazios;
  l Observar_Não_Conformidades;
  l Fazer_Busca_Superficial;
  l Fazer_Busca_Detalhada;
  l Retornar_Garrações;
  l Aceitar_Garrações;
  l Contar_Garrações;
  l Passar_Garrações;
end
attributes
  l Atribimplementável: enum{verd, falso, indef};
  l Detectar_Garrações_Não_Conformes: boolean;
  s Segurança[Cond_Trabalho]: enum {posit, negat, indef, sub, naosub};
axioms
  1. [Fazer_Inspeção_Visual] (Detectar_Garrações_Não_Conformes ∧
                                obl(Receber_Garrações_Vazios) ∧
                                obl(Observar_Não_Conformidades));
  2. [Aceitar_Garrações] (obl (Contar_Garrações) ∧
                           obl (Passar_Garrações));
  3. [Observar_Não_Conformidades] (obl(Fazer_Busca_Detalhada) ∨
                                    obl(Fazer_Busca_Superficial));
  4. [Fazer_Busca_Detalhada] ((obl(Retornar_Garrações) ∨
                                obl(Aceitar_Garrações));
  5. [Fazer_Busca_Superficial] ((obl(Retornar_Garrações) ∨
                                obl(Aceitar_Garrações));
  6. [Fazer_Inspeção]((Qualidade[Serviço] = posit) ∧
                     (Rapidez[Atendimento] = posit));
end agent Inspeção

```

Figura 6.11 - Axiomas para as Ligações Meio-Fim
do Agente Inspeção, da Envasadora de Água Mineral

- (i) as tarefas “Rejeitar_Garrações” e “Aceitar_Garrações”, representam as possíveis alternativas da realização da tarefa “Nova_Inspeção_Visual”. Esta situação será representada pelos axiomas:

[Nova_Inspeção_Visual] (**obl** (Rejeitar_Garrações)
∨ obl (Aceitar_Garrações))

que no corpo da especificação MAL Estruturado do agente “Limpeza Externa”, corresponde ao axioma 3.

- (ii) a tarefa “Fazer Limpeza Externa” que pelo relacionamento externo com outros atores possui como fim um objetivo-soft “Qualidade[Serviço]”. Assim, pela aplicação de M7.1, temos o axioma

[Fazer Limpeza Externa] (Qualidade[Serviço] = posit)

que no corpo da especificação MAL Estruturado do agente “Limpeza Externa”, corresponde ao axioma 4.

- (iii) a tarefa “Rejeitar Garrações” sendo o meio para se alcançar o fim “Garrações Vazios Rejeitados”, um recurso esperado pelo Cliente. Assim, pela aplicação de M7.1, temos o axioma

[Rejeitar Garrações] GarVaziosRej

que no corpo da especificação MAL Estruturado do agente “Limpeza Externa”, corresponde ao axioma 5.

Estes axiomas serão incorporados à especificação do agente Limpeza_Externa, Figura 6.12.


```

agent Limpeza_Externa
actions
  l Fazer_Limpeza_Externa;
  l Receber_Garrações_Conformes;
  l Nova_Inspecção_Visual;
  l Aceitar_Garrações;
  l Rejeitar_Garrações;
  l Tirar_Rótulo;
  l Tirar_Lacre;
  l Tirar_Tampa;
  l Passar_Garrações_Limpos;
attributes
  l Atrbimplementável: enum{verd, falso, indef};
  s Segurança[Cond_Trabalho]: enum {posit, negat, indef, sub, naosub};
  ....
axioms
  1. [Fazer_Limpeza_Externa] obl(Receber_Garrações_Conformes) ∧
      obl(Nova_Inspecção_Visual));
  2. [Aceitar_Garrações] (obl (Tirar_Rótulo) ∧ obl (Tirar_Lacre) ∧
      obl(Tirar_Tampa) ∧ obl(Passar_Garrações_Limpos) );
  3. [Nova_Inspecção_Visual] (obl (Rejeitar_Garrações)
      ∨ obl (Aceitar_Garrações));
  4. [Fazer Limpeza Externa] (Qualidade[Serviço] = posit);
  5. [Rejeitar Garrações] GarVaziosRej;
end agent Limpeza_Externa

```

Figura 6.12 - Axiomas para as Ligações Meio-Fim do Agente Limpeza_Externa, da Envasadora de Água Mineral

Essas pequenas frações de especificação em MAL estruturado são suficientes para mostrar a aplicabilidade das diretrizes apontadas no Capítulo 5 pois, para todos os demais agentes envolvidos, as sete diretrizes devem ser aplicadas iterativamente, até se cobrir todos os aspectos que sejam de maior interesse. Dessa forma, desde que seja possível representar, em MAL estruturado, elementos expressos pelos modelos organizacionais, é possível integrar esses elementos às especificações de sistemas já realizadas nessa linguagem, de forma que os requisitos organizacionais assim formalizados, possam complementar aquelas especificações existentes. Em alguns trabalhos anteriores [ALE97a, ALE98a, ALE98b, ALE99a], abordamos esse mesmo estudo de caso.

6.1.2 – SISTEMA BANCÁRIO DE TRANSFERÊNCIA POR TELEFONE COM SEGURANÇA POR CRIPTOGRAFIA

Trata-se do segundo exemplo onde apresentaremos a utilização da ferramenta i*. Consideramos um exemplo tradicional, [YU97a, DUB98], de um sistema bancário, cuja transação de transferência de fundos entre contas é, feita com a ida do cliente ao banco, onde ele trata diretamente com o caixa. Dentre outros obstáculos, nesse tipo de sistema, encontramos a perda de tempo enfrentando-se filas enormes.

A proposta é modelar um sistema que possibilite transações via telefone e/ou via home-banking. Adicionalmente, sugerimos a utilização de um módulo para a segurança das informações que circulam; para isto, é proposto o uso de técnicas de criptografia [ALE97a]. Isto posto, partindo da determinação dos atores envolvidos no sistema, inicialmente, determinamos o Modelo SD da Figura 6.13.

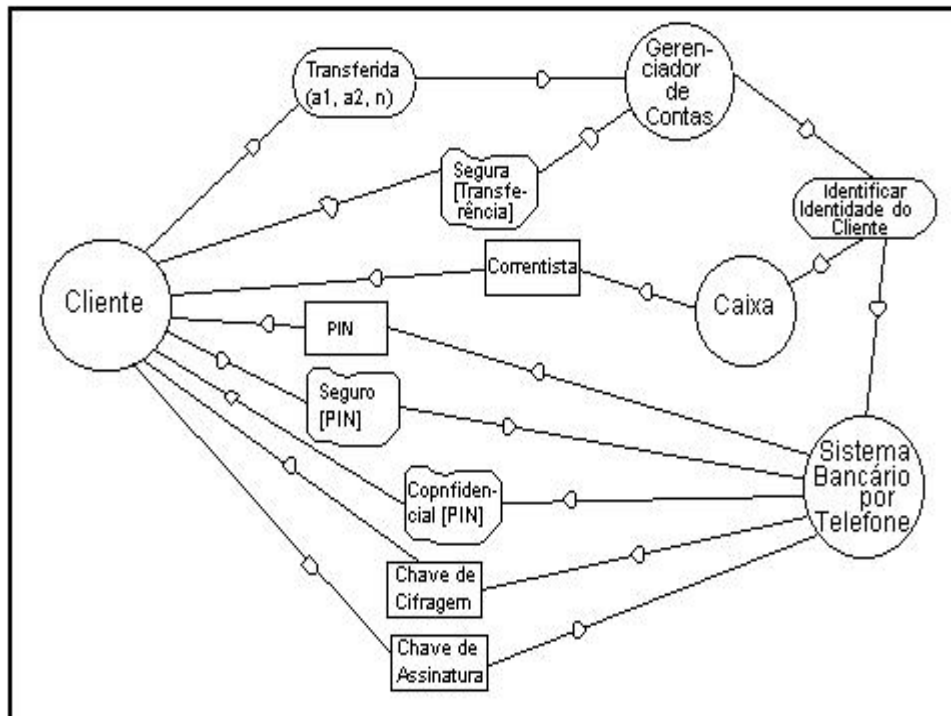


Figura 6.13 - Modelo de Dependências Estratégicas de um Sistema Bancário de Transferência por Telefone

Avaliando o Modelo de Dependências Estratégicas (SD), Figura 6.13, vemos que no sistema convencional, faz-se necessária a presença física do *Correntista* para que seja feita, pelo *Caixa*, sua identificação. Só após essa identificação, como forma de segurança da transação de transferência de fundos, o início da operação pode ser processado. Como melhoria ao sistema, tem-se a possibilidade de transferência via telefone. Neste caso, faz-se necessário a existência de um número de identificação pessoal (PIN- Personal Identification Number) que, a princípio, o banco espera que seja mantido confidencial (sob sigilo) pelo cliente. Ao mesmo tempo, o cliente espera que a segurança do PIN seja mantida pelo sistema. Surge, então, como alternativa para garantir o sigilo e a autenticidade do PIN, a utilização de técnicas de criptografia. Neste caso, o banco deve fornecer ao Cliente uma chave de cifragem, “*Chave de Cifragem*”, através da qual ele transmite seu número de identificação pessoal criptografado, bem como uma chave com a qual o *Banco* possa atestar a autenticidade das informações que estão sendo fornecidas, “*Chave de Assinatura*”.

No Modelo de Dependência Estratégica (SD) do sistema bancário da Figura 6.13, originariamente não existe, como ator, um *Centro Autorizador*, responsável pela validação e fornecimento das chaves de cifragem, bem como o habilitador do acesso ao usuário autorizado. Todavia, a medida em que a compreensão sobre o sistema vai se consolidando, os requisitos iniciais vão evoluindo, fazendo com que os modelos precisem ser refinados cada vez mais. Pensando nesse aspecto, refinamos o Modelo de Dependências Estratégicas da Figura 6.13, para o modelo da Figura 6.14, onde neste existe um novo agente, o “*Centro Autorizador*”.

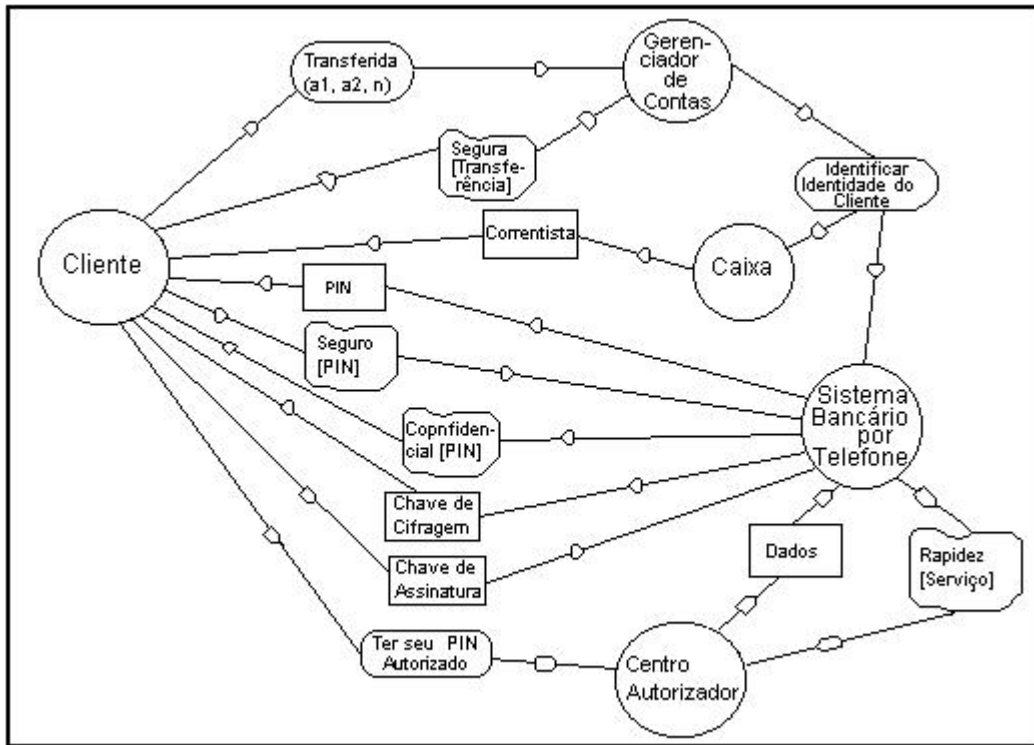


Figura 6.14 - Modelo de Dependências Estratégicas de um Sistema Bancário de Transferência por Telefone com um Centro Autorizador

Com base no Modelo de Dependências Estratégicas da Figura 6.14, aplicaremos as diretrizes, propostas no Capítulo 5. A primeira etapa a ser cumprida é quanto aos atores envolvidos. No caso específico, temos cinco atores; logo, aplicaremos a diretriz M1.

Diretriz M1: Relacionada com o mapeamento dos atores do modelo i^* .

*Os **Atores** no Modelo de Dependências Estratégicas (SD) podem ser entendidos como **Agentes** em MAL estruturado.*

Com base nesse diretriz, e na subdiretriz M1.1, para tratar da agregação de atores, dado que o sistema bancário é composto por cinco atores, o trecho de especificação em MAL estruturado apresenta-se conforme o explicado na Figura 6.15.

SubDiretriz M1.1: Relacionada com o mapeamento da composição dos vários atores.

A **Composição de Atores (ActComp)** que representam um sistema em i^* , é mapeada em MAL como morfismo de **Agregação de Agentes**, expresso no agente composto.

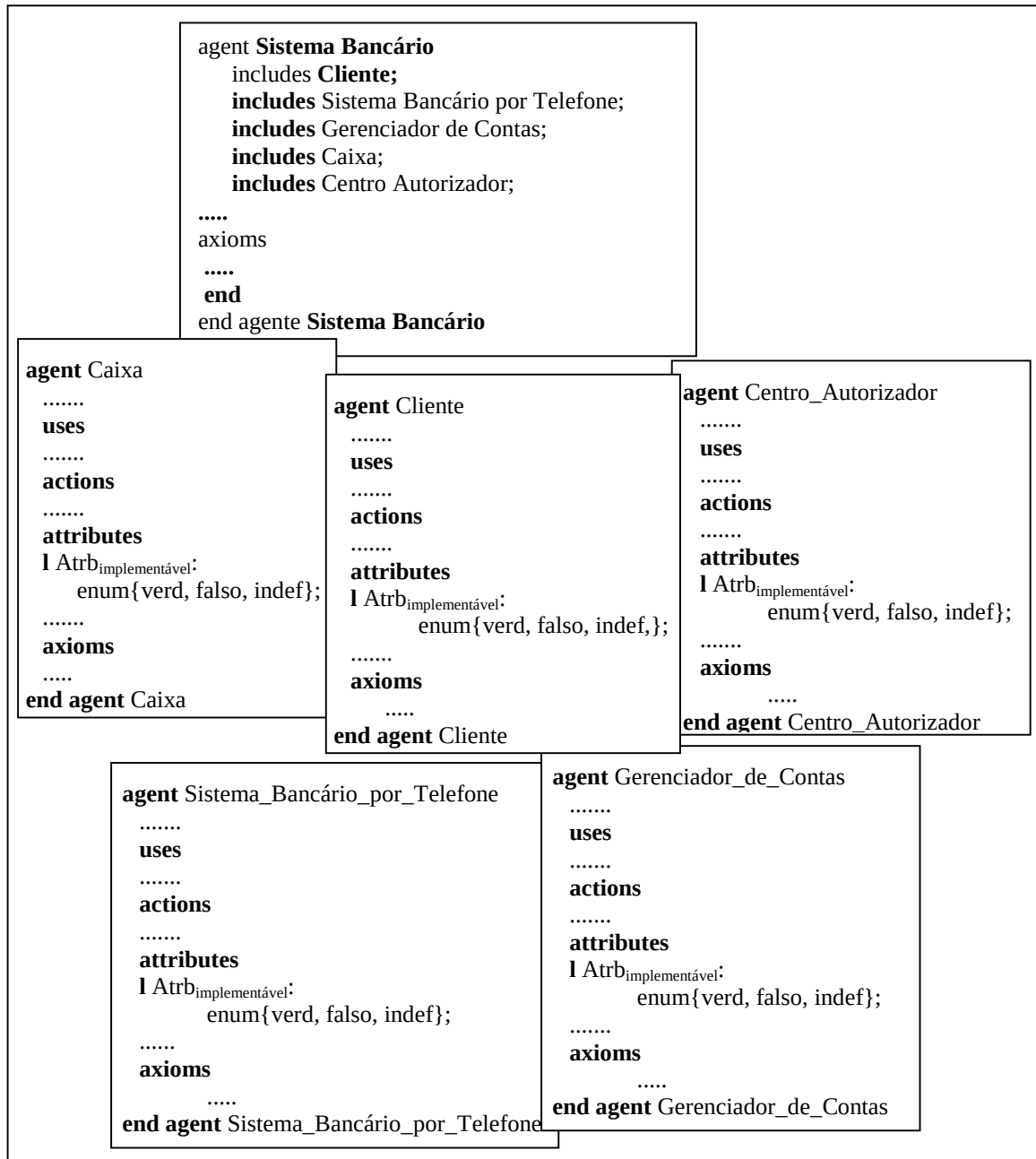


Figura 6.15 – Definição do Agente Sistema Bancário

Dado que o sistema bancário é composto por cinco atores, o trecho de especificação em MAL estruturado apresenta-se conforme o explicitado na Figura 6.15 – os cinco agentes e um agente “Sistema_Bancário”, representando a composição.

Deve-se então passar-se a observar os elementos das dependências entre os atores, aplicando-se as demais diretrizes. Como esses elementos também estão presentes no Modelo de Razões Estratégicas (SR), consideremos a Figura 6.16, onde é explorado o comportamento interno dos atores. Devemos ressaltar que apenas os aspectos considerados estratégicos são destacados

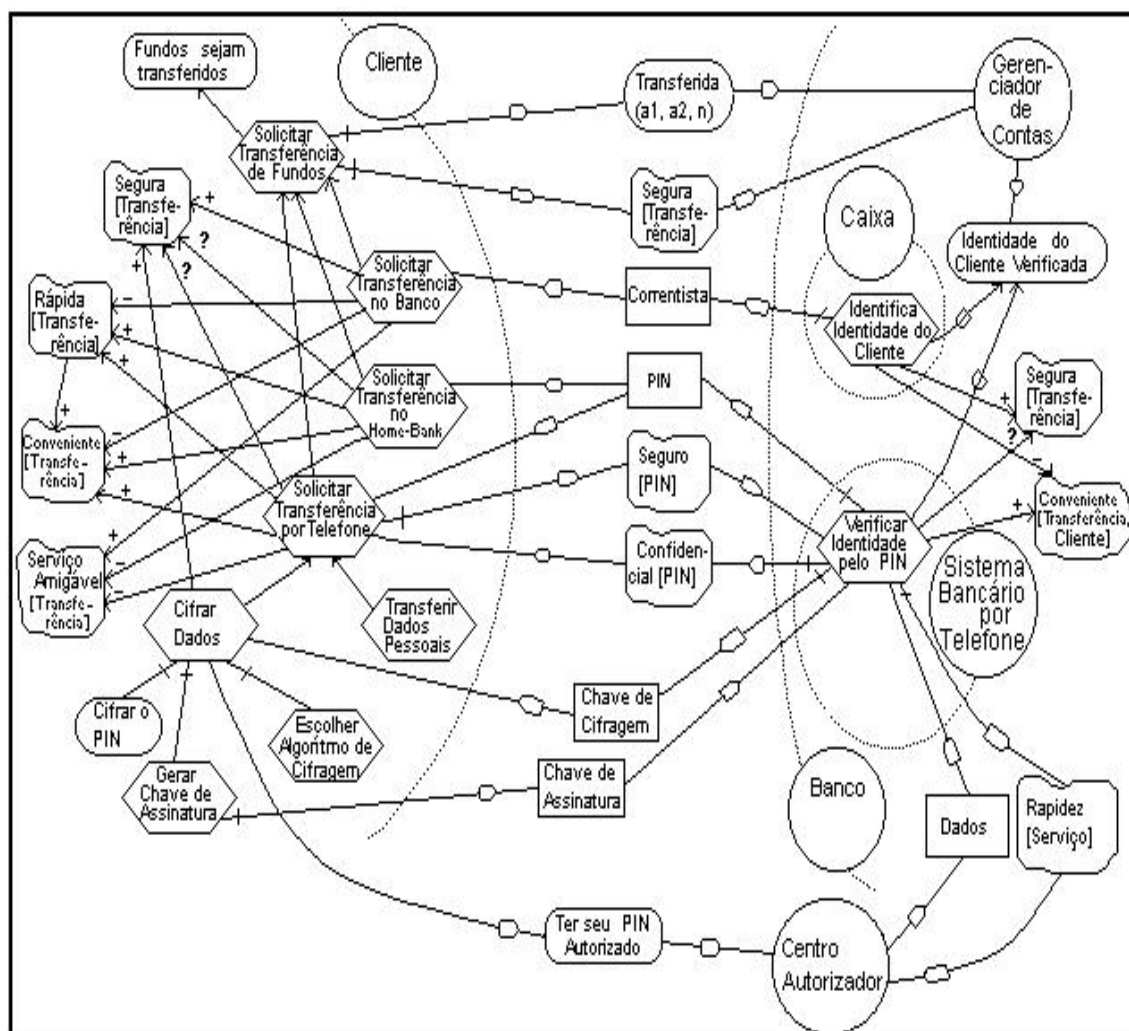


Figura 6.16 - Modelo de Razões Estratégicas para o Sistema Bancário de Transferência por Telefone

No Modelo de Razões Estratégicas (SR), Figura 6.16, podemos perceber melhor as vantagens e desvantagens existentes com a inserção das novas alternativas. A avaliação das contribuições pode ser graficamente apontada pelas setas com sinais positivo (“+”), negativo (“-“) ou desconhecido (“?”). Por exemplo, do ponto de vista do usuário, o *banking-por-telefone* é visto como uma forma mais rápida e mais conveniente, embora menos amigável. No sistema via telefone sem criptografia, de maneira geral, não se teria garantia de segurança, como pode ser visto na Figura 6.16, pela existência de uma interrogação (?). Contudo, com a inclusão das técnicas de criptografia, essa segurança é evidenciada. Um dos objetivos do banco é que seus serviços se tornem mais lucrativos e com isso o mercado seja ampliado, havendo um maior atrativo em se trabalhar com esse banco. Todos esses desejos podem ser traduzidos por objetivos soft (*soft-goals*).

Mais uma vez, podemos destacar que, para desenvolver sistemas que verdadeiramente traduzam as necessidades reais de uma organização, faz-se necessário se ter o conhecimento aprofundado de como o sistema é encaixado no ambiente organizacional. Passemos, então, à aplicação das diretrizes propostas no Capítulo 5, quanto aos elementos envolvidos na dependência (tarefas, recursos, objetivos e objetivos-soft).

Diretriz M2: Encontra-se relacionada com componente **tarefa**, nos modelos estratégicos da técnica *i**.

*Todas as **tarefas** dos modelos estratégicos *i** serão traduzidas como **ações** de um agente em MAL estruturado.*

A depender do modelo que se esteja trabalhando, precisaremos das subdiretrizes M2.1 e M2.2. No caso em estudo, no Modelo SD, Figura 6.14, não encontramos nenhuma tarefa como elemento do relacionamento de dependência, pelo que não aplicamos M2.1. No Modelo SR, Figura 6.16, observamos que algumas tarefas fazem parte do comportamento interno dos seguintes atores: “Cliente”, “Caixa” e “Sistema_Bancário_por_Telefone”.

Consideremos, inicialmente, o ator “Cliente”. Aplicando-se a subdiretriz M2.2, todas as tarefas identificadas (Solicitar_Transferência_de_Fundos, Solicitar_Transferência_no_Banco, Solicitar_Transferência_no_Home-Bank, Solicitar_Transferência_por_Telefone; Transferir_Dados_Pessoais, Cifrar_Dados, Gerar_Chave_de_Assinatura e Escolher_Algoritmo_de_Cifragem) mapeadas como ações locais ao *agente Cliente*, em MAL estruturado, Figura 6.17. Da mesma forma, aplicando-se essa mesma subdiretriz aos outros agentes *Caixa* e *Sistema Bancário por Telefone*, teremos todas as tarefas mapeadas como ações locais a esses agentes, em MAL Estruturado. Desta forma, associada ao *Agente Caixa*, haverá a ação local “Identifica_Identidade_do_Cliente”, enquanto para o *Agente Sistema* será incluída a ação local “Verificar_Identidade_pelo_PIN”. Trecho da especificação desses agentes, também é visualizado na Figura 6.17.

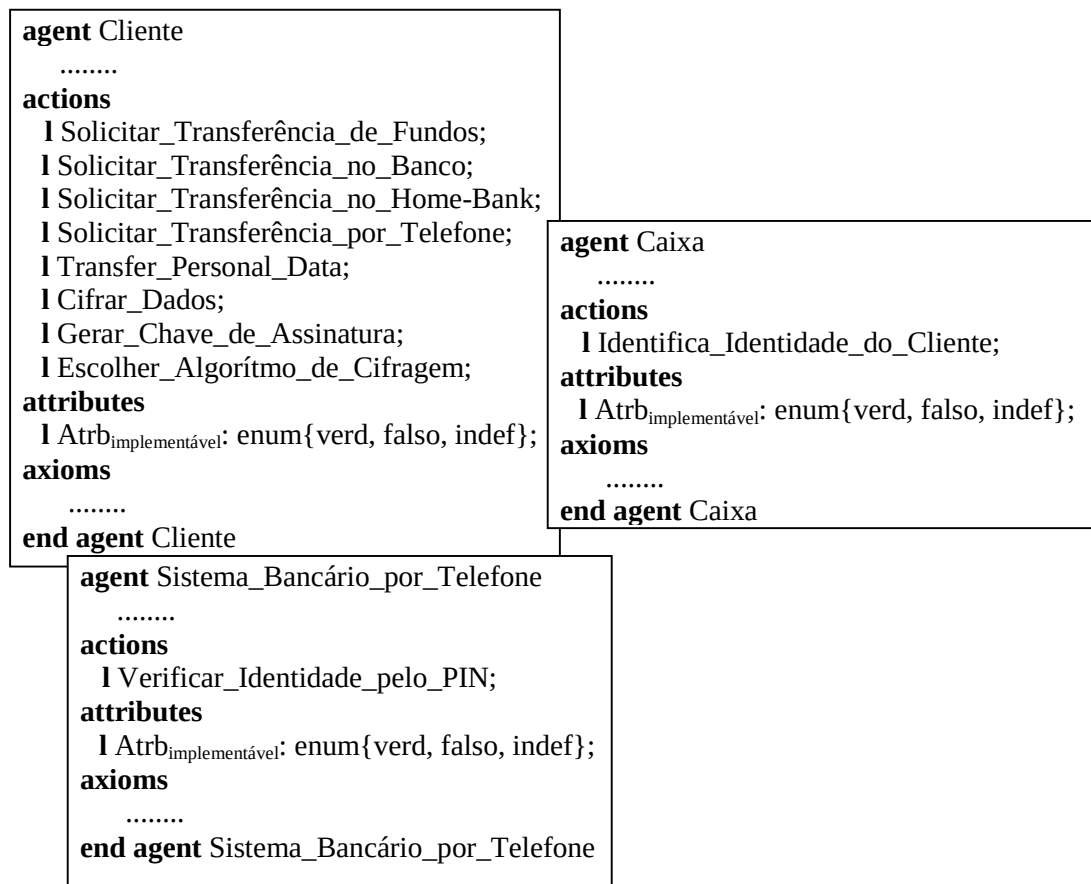


Figura 6.17 - Especificando Tarefas, em i*, dos atores Cliente, Caixa e Sistema Bancário por Telefone como Ações em MAL Estruturado

O próximo elemento a ser considerado serão os recursos. Para tanto, observando o Modelo de Dependências Estratégicas (SD), Figura 6.14, identificamos vários desses elementos envolvidos nas relações de dependência, enquanto que no Modelo de Razões Estratégicas (SR), Figura 6.16, apenas identificamos esse tipo de elemento no comportamento interno do ator “Sistema_Bancário_por_Telefone”.

Diretriz M3: Encontra-se relacionada com componente **recurso**, nos modelos estratégicos da técnica i*.

*Os **recursos** dos modelos estratégicos serão traduzidos como **agentes** em MAL estruturado, com um **atributo compartilhado** ($atrb_{rec}$) do tipo **booleano**, que indique a disponibilidade desse recurso;*

Observando os Modelos Estratégicos da Figura 6.16, identificamos os recursos:

- (i) *Correntista*, que representa a pessoa do Cliente que se dirige ao banco para fazer a transação pessoalmente;
- (i) *PIN*, que representa o número de identificação pessoal do cliente;
- (ii) *Chave_de_Cifragem*, que será a chave utilizada para cifrar as informações do cliente;
- (iii) *Chave_de_Assinatura*, a chave que permitirá ao cliente, eletronicamente assinar os documentos; e
- (iv) *Dados*, corresponde às informações do cliente, armazenadas no banco.

A estes recursos corresponderão agentes em MAL Estruturado, Figura 6.18.

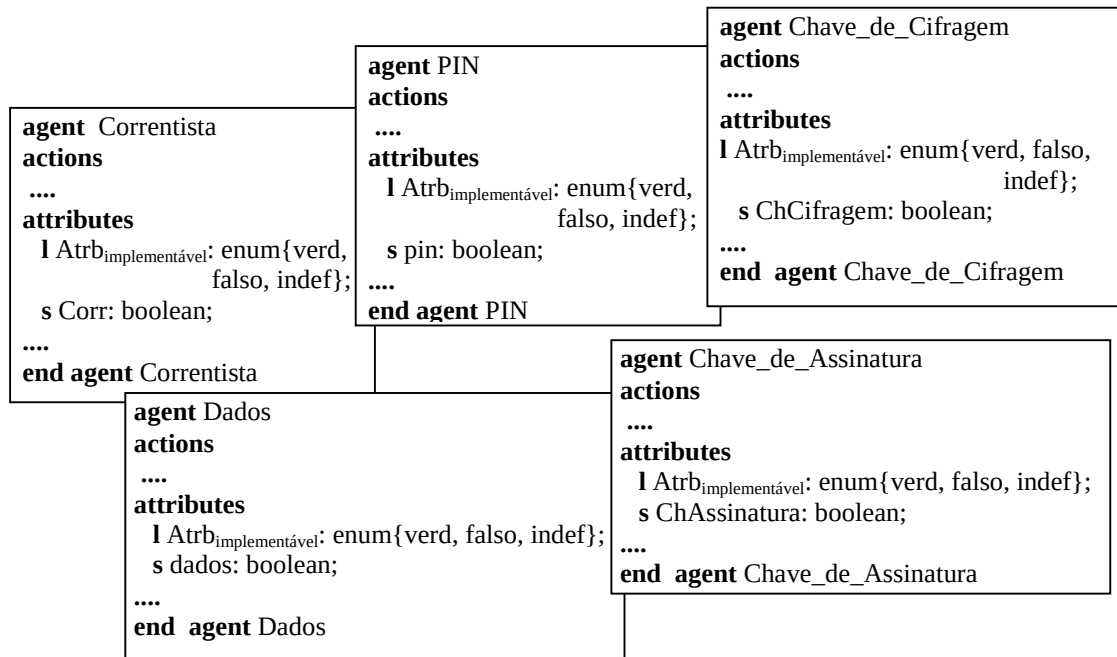


Figura 6.18 – Recursos do Sistema Bancário Mapeados em Agentes em MAL Estruturado

Observemos, que a cada agente, correspondente a um recurso nos modelos estratégicos de i*, está associado um atributo compartilhado, que indica a disponibilidade desse recurso.

Quanto ainda aos elementos presentes na relação de dependências, identificamos na Figura 6.14 objetivos bem definidos pelo ator “Cliente”, mas também objetivos-soft. Desta forma, aplicaremos a diretriz M4.

Diretriz M4: Encontra-se relacionada com os componentes **objetivos bem definidos** e **objetivos soft**, nos modelos estratégicos da técnica i*.

*Os **objetivos bem definidos** e os **objetivos soft** dos modelos estratégicos serão traduzidos como **atributos do tipo booleano e do tipo enumerado**, respectivamente, em um agente MAL estruturado;*

Em se tratando dos objetivos bem definidos e/ou dos objetivos soft, relacionados com aspectos não bem delimitados, é proposto associar um atributo do tipo booleano e do tipo enumerado, respectivamente, a esses elementos. Como considerado anteriormente, esses componentes, devem ser vistos segundo o Modelo SD ou Modelo

SR. Desta forma aplicamos as duas subdiretrizes, M4.1 e M4.2, respectivamente, para os Modelos SD (Figura 6.14) e SR (Figura 6.16).

Com relação ao ator *Cliente*, segundo a Figura 6.14, identificamos:

- (i) dois objetivos bem definidos:
 - (a) “Transferida (a_1 , a_2 , n)”, onde o ator objetiva ter uma quantia n transferida de uma conta a_1 para uma conta a_2 , objetivo esse que deve ser realizado pelo ator “Gerenciador_de_Contas”; e
 - (b) “Ter_seu_PIN_Autorizado”, onde objetiva que o “Centro_Autorizador” libere seu PIN; e
- (ii) dois objetivo-soft:
 - (a) “Seguro[PIN]”, onde espera que o seu número de identificação pessoal (PIN) seja mantido seguro, pelo “Sistema_Bancário_por_Telefone”; e
 - (b) “Segura[Transferência]”, onde o cliente espera segurança no processo de transferência de fundos entre contas.

Com relação ao Sistema_Bancário_por_Telefone, tem-se apenas o objetivo-soft “Confidencial[PIN]”, onde o ator espera que o *Cliente* mantenha seu número de identificação pessoal consigo, não o revelando a ninguém. A esses objetivos será aplicada a subdiretriz M4.1, onde na Figura 6.19 teremos o mapeamento desses, em MAL estruturado, como atributos booleanos e enumerados compartilhados.

Por outro lado, passando a observar o comportamento interno dos atores, Figura 6.16, apenas ator “*Cliente*” possui alguns objetivos como parte do seu comportamento. Identificamos:

- (i) um sub-objetivo bem definido, “Cifrar_PIN”; e
- (ii) três objetivos-soft:
 - (a) “Rápida [Transferência]”, onde o Cliente espera rapidez no processo de transferência de fundos entre as contas;
 - (b) “Conveniente [Transferência]”, onde espera que o processo de transferência seja conveniente; e
 - (c) “Serviço_Amigável [Transferência]”.

A estes tipos de objetivos aplicamos a subdiretriz M4.2, de forma que na Figura 6.19 podemos observar o trecho de especificação correspondente a esses objetivos, em MAL estruturado, em que aparecem como atributos locais.

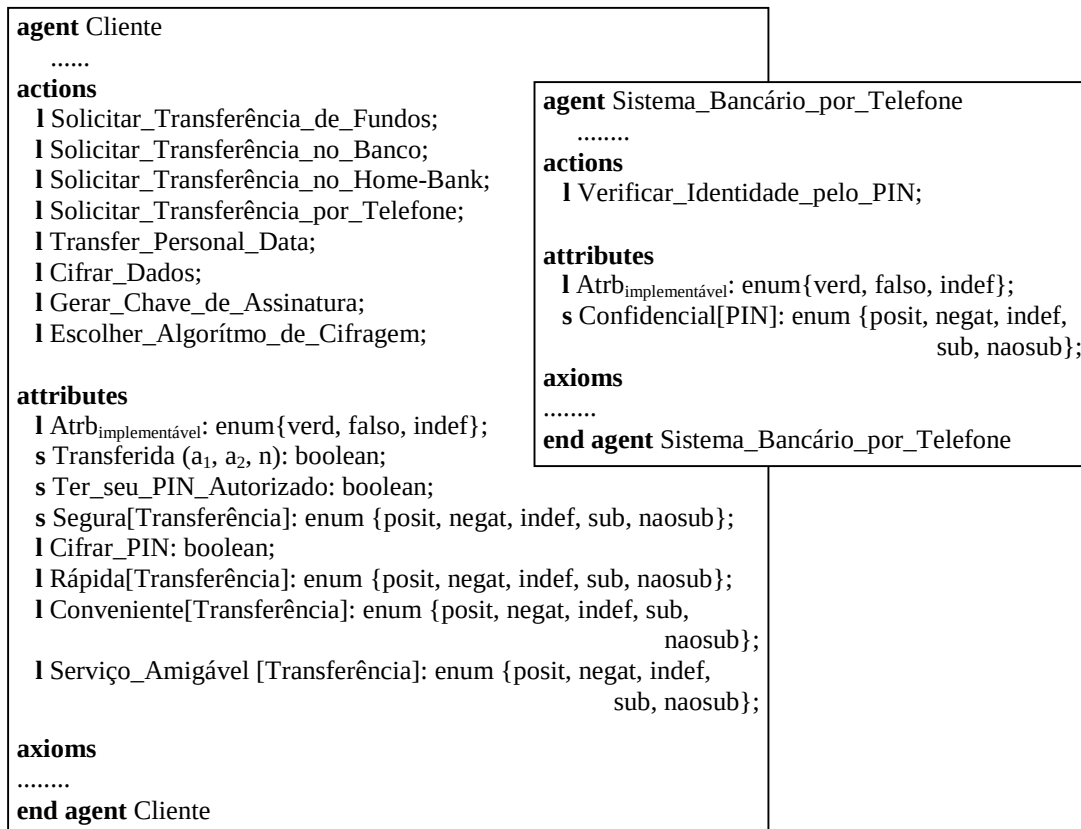


Figura 6.19 - Especificando Objetivos e Objetivos-Soft em i*, para o Sistema Bancário, como Atributos Booleanos, em MAL Estruturado

Quanto ao Modelo de Dependências Estratégicas (SD), existem os relacionamentos externos entre os agentes; assim sendo, faremos uso da segunda diretriz, M5.

Diretriz M5: Trata dos relacionamentos entre os atores do i*;

Os *relacionamentos* nos Modelos Estratégicos (SD e SR) da técnica i* podem ser tratados pelos *axiomas*, em MAL estruturado.

Na verdade, os axiomas em uma especificação MAL estruturado descrevem o comportamento dos agentes (objetos). Consideremos pois os atores “Gerenciador de Contas” e o “Cliente”. Entre estes atores, por exemplo, existe a dependência de objetivo soft, “Segura[Transferência]”, e a dependência de objetivo bem definido, “Transferida(a1,a2,n)”, significando que o “Cliente” deseja transferir de uma conta a1 para uma conta a2, a quantia n. Assim, na Figura 6.20, podemos observar um trecho da especificação em MAL estruturado, do ator “Cliente”, que é o ator dependente nesses relacionamentos citados. Assim, como é o agente dependente, no corpo da especificação do agente do qual depende, no caso o agente “Gerenciador_de_Contas”, constarão os axiomas que precisão a dependência. Observe que na especificação do agente “Gerenciador_de_Contas”, acrescentamos duas ações que não estão no Modelo de Razões Estratégicas, Figura 6.16, porque seu comportamento interno não foi detalhado.

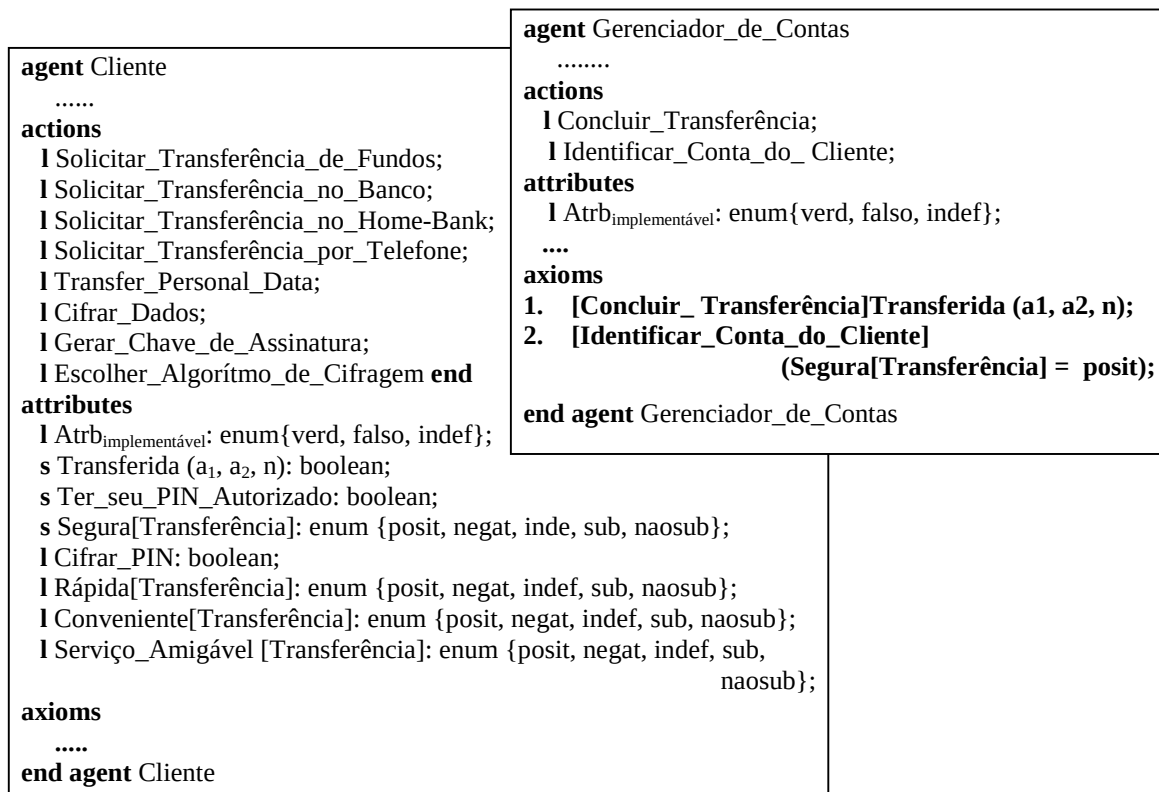


Figura 6.20 – Relacionamento entre Agentes em i*, de um Sistema Bancário, Expresso por Axiomas em MAL Estruturado

Observemos que o axioma 1 envolve uma ação local do agente Gerenciador_de_Contas, “Concluir_Transferência”, que ao seu término tornará satisfeito um objetivo do Cliente, “Transferida(a_1, a_2, n)”. Da mesma forma, o axioma 2 envolve uma ação local do Gerenciador_de_Contas, “Identificar_Conta_do_Cliente”, que visa satisfazer um objetivo-soft do Cliente, “Segura[Transferência]”. Os demais relacionamentos existentes entre o agente “Cliente” e os demais agentes, também seriam explicitados através dos axiomas, aumentando o número de axiomas da Figura 6.20.

Neste ponto, com base, no Modelo de Razões Estratégicas da Figura 6.16, passaremos a aplicar as diretrizes de número M6 e M7, lembrando que todas as vezes em que haja qualquer dos elementos tarefa, recurso, objetivo ou objetivo-soft, faz-se necessário a utilização das diretrizes anteriores (M1 a M4).

Diretriz M6: Trata das ligações do tipo decomposição de tarefa;

*Uma ligação do tipo **decomposição de tarefa** é representada, em MAL estruturado, através de um **axioma de ação** contendo o conectivo lógico **AND** e o operador deôntico **obl** aplicado às **ações** que possam existir, de acordo com a decomposição da tarefa em sub-objetivo, sub-tarefa, recursoPara ou objetivo-softPara.*

No agente Cliente, encontramos:

- (i) a tarefa “Cifrar_Dados” sendo decomposta em:
 - (a) um objetivo bem definido, “Cifrar_o_PIN”; e
 - (b) uma tarefa, “Gerar_Chave_de_Assinatura”; e
 - (c) uma tarefa, “Escolher_Algoritmo_de_Cifragem”;

Aplicando-se a diretriz M6, a essa decomposição, tem-se o seguinte axioma:

$$[Cifrar_Dados] (Cifrar_PIN) \wedge obl(Gerar_Chave_de_Assinatura) \wedge obl(Escolher_Algoritmo_de_Cifragem));$$

- (ii) a tarefa “Solicitar_Transferência_por_Telefone”, decomposta em um objetivo-soft, “Seguro[PIN]”;

Aplicando-se a diretriz M6, a essa decomposição, tem-se o seguinte axioma:

$$[Solicitar_Transferência_por_Telefone] (Seguro[PIN]);$$

(iii) A tarefa “Solicitar_Transferência_de_Fundos” sendo decomposta em:

- (a) um objetivo bem definido, “Transferida(a_1, a_2, n)”; e
- (b) um objetivo-soft, “Segura[Transferência].

Aplicando-se a diretriz M6, a essa decomposição, tem-se o seguinte axioma:

$$[\text{Solicitar_Transferência_de_Fundos}] (\text{Transferida}(a_1, a_2, n) \wedge (\text{Segura}[\text{Transferência}] = \text{posit}));$$

Na Figura 6.21, mais adiante, podemos ver esses axiomas incorporados ao corpo da especificação em MAL Estruturado do agente Cliente.

No agente Sistema_Bancário_por_Telefone, encontramos:

- (i) a tarefa “Verificar_Identidade_pelo_PIN”, sendo decomposta:
 - (a) no recurso, “PIN”;
 - (b) no recurso, “Chave_de_Cifragem”;
 - (c) no objetivos-soft, “Confidencial[PIN]”; e
 - (d) no objetivos-soft, Rapidez[Serviço]”.

Aplicando-se a diretriz M6, a essa decomposição, tem-se o seguinte axioma:

$$[\text{Verificar_Identidade_pelo_PIN}] (\text{ChCifragem} \wedge \text{pin} \wedge (\text{Confidencial}[\text{PIN}] = \text{posit}) \wedge (\text{Rapidez}[\text{Serviço}] = \text{posit}));$$

onde observamos que os recursos são considerados, no axioma, através de seus atributos (pin e ChCifragem).

Na Figura 6.21, mais adiante, podemos ver esse axioma incorporado ao corpo da especificação em MAL Estruturado do agente Sistema_Bancário_por_Telefone.

No agente Caixa, encontramos a tarefa “Identifica_Identidade_do_Cliente”, decomposta no recurso “Cliente”. Desta forma, teremos o seguinte axioma:

$$[\text{Identifica_Identidade_do_Cliente}] \text{Corr};$$

onde Corr é o atributo booleano do agente recurso Correntista.

Na Figura 6.21, mais adiante, podemos ver esse axioma incorporado ao corpo da especificação em MAL Estruturado do agente Caixa.

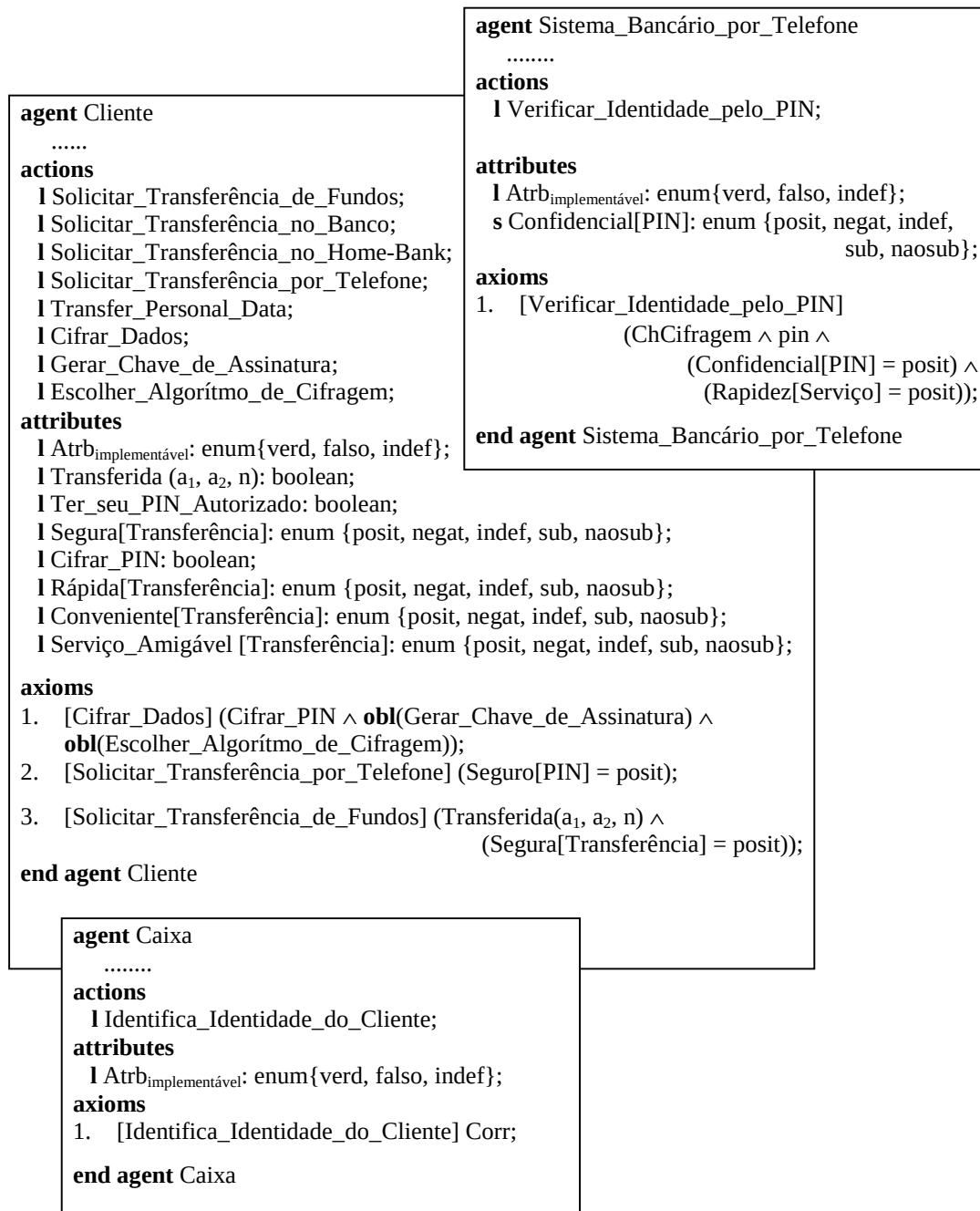


Figura 6.21 - As Ligações de Decomposição de Tarefa em i*, para o Sistema Bancário, Especificadas em MAL Estruturado

Salientamos que, à proporção que seja necessário, em evoluindo os requisitos, novos elementos podem ser acrescentados à especificação.

Diretriz M7: Trata das ligações do tipo meio-fim;

*Uma ligação do tipo **meio-fim** é representada, em MAL estruturado, através de axiomas.*

Observando a Figura 6.16, com respeito aos agentes “*Cliente*”, “*Caixa*” e “*Sistema_Bancário_por_Telefone*”, tem-se vários relacionamentos meio-fim, onde precisaremos classificá-los nos diversos tipos, conforme sejam *o meio* e *o fim*. Como o procedimento é similar, apenas avaliaremos o ator “*Cliente*”, onde identificamos:

- (i) um relacionamento meio-fim, onde a tarefa “*Solicitar_Transferência_de_Fundos*” é o *fim*, tendo como *meios*, três outras tarefas:
 - (a) “*Solicitar_Transferência_no_Banco*”, onde a solicitação é feita na própria agência bancária;
 - (b) “*Solicitar_Transferência_no_Home-Bank*”, onde a solicitação é feita via o Home-Bank; e
 - (c) “*Solicitar_Transferência_por_Telefone*”, onde a solicitação é feita via telefone;

Nesta situação, aplicaremos a sub-diretriz M7.1, item iv.

SubDiretriz M7.1: Trata das ligações do tipo meio-fim, onde o meio é uma tarefa e o fim pode ser tarefa, objetivo bem definido, objetivo-soft ou recurso.

*Uma ligação do tipo **meio-fim**, onde o meio é uma tarefa, é representada, em MAL estruturado, através de um **axioma de ação** contendo o conectivo lógico **OR** e o operador deôntico **obl** aplicado às **ações** que possam existir, em função de poder ser o fim uma tarefa, um objetivo bem definido, um objetivo-soft ou um recurso.*

M7.1. iv - uma tarefa como um fim, mapeada para MAL estruturado, segundo M2, em uma ação, **Ac**, e como meio tenhamos *n* tarefas, mapeadas para MAL estruturado, segundo M2 ($Ac_1, Ac_2, \dots, Ac_n$), então teremos, axiom $[Ac] (obl (Ac_1) \vee obl (Ac_2) \vee \dots obl (Ac_n))$

Com base nesse item da subdiretriz M7.1, teremos o seguinte axioma:

$$[\text{Solicitar_Transfer\^encia_de_Fundos}](\text{obl}(\text{Solicitar_Transfer\^encia_no_Banco}) \vee \\ \text{obl}(\text{Solicitar_Transfer\^encia_no_Home-Bank}) \vee \\ \text{obl}(\text{Solicitar_Transfer\^encia_por_Telefone}));$$

Este axioma, no corpo da especificao em MAL Estruturado do agente Cliente, Figura6.22,  o de nmero 4.

- (ii) um relacionamento meio-fim, onde a tarefa “Solicitar_Transferncia_por_Telefone”  o fim, tendo como meio duas outras tarefas:
 - (a) “Cifrar_Dados”; utilizar a cifraem dos dados, situao expressa pela tarefa”; e
 - (b) “Transfer_Personal_Data”, onde so utilizados dados pessoais do Cliente, tais como, data de aniversrio, filiao, etc., sem cifraem;

Nesta situao, aplicaremos tambm a sub-diretriz M7.1, item iv, de forma que tenhamos o seguinte axioma

$$[\text{Solicitar_Transfer\^encia_por_Telefone}](\text{obl}(\text{Cifrar_Dados}) \vee \\ \text{obl}(\text{Transfer_Personal_Data}));$$

Este axioma, no corpo da especificao em MAL Estruturado do agente Cliente, Figura6.22,  o de nmero 5.

- (iii) um relacionamento meio-fim, onde a tarefa “Solicitar_Transferncia_de_Fundos”  o meio, tendo como fim um objetivo bem definido, “Fundos_sejam_transferidos”;

Nesta situao, aplicaremos a sub-diretriz M7.1, item i.

M7.1.i - um objetivo bem definido como um fim, mapeado para MAL estruturado, segundo D.4, Atrb_{ob} , e como meio tenhamos n tarefas, mapeadas para MAL estruturado, segundo D.2 ($\text{Ac}_1, \text{Ac}_2, \dots, \text{Ac}_n$), ento teremos

axiom $[\text{Ac}_1] \text{Atrb}_{ob} \vee [\text{Ac}_2] \text{Atrb}_{ob} \vee \dots [\text{Ac}_n] \text{Atrb}_{ob}$

Com base nesse item (i) da subdiretriz M7.1, teremos o seguinte axioma:

[Solicitar_Transferência_de_Fundos](Fundos_sejam_transferidos);

Este axioma, no corpo da especificação em MAL Estruturado do agente Cliente, Figura 6.22, é o de número 6.

- (iv) um relacionamento meio-fim, onde a tarefa “Solicitar_Transferência_no_Banco”, é o meio para se alcançar:
 - (a) o objetivos soft “Segura[Transferência]”, contribuindo positivamente, significando que a transferência no próprio banco é mais segura; essa situação será indicada atribuindo-se ao objetivo soft o valor true.
 - (b) o objetivo soft “Rápida [Transferência]”, contribuindo negativamente, significando que a transferência no próprio banco é menos rápida, pelo fato de ter-se que se deslocar até o estabelecimento; essa situação será indicada atribuindo-se ao objetivo soft o valor false;
 - (c) o objetivo soft “Conveniente[Transferência]”, contribuindo negativamente, significando que a transferência no próprio banco é menos conveniente pelo fato de ter-se que se deslocar até o estabelecimento, enfrentar filas, etc.; essa situação será indicada atribuindo-se ao objetivo soft o valor false;
 - (d) o objetivo soft “Serviço_Amigável[Transferência]”, contribuindo positivamente, significando que a transferência no próprio banco é mais amigável, pelo fato de proporcionar o contato direto e pessoal entre as pessoas; essa situação será indicada atribuindo-se ao objetivo soft o valor true;
 - (e) o recurso “Correntista”, significando a necessidade do cliente está presente no banco, para poder efetuar a transferência, traduzida pelo atributo Corr, do agente Correntista;
 - (f) a tarefa “Solicitar_Transfereência_de_Fundos”, significando ser a tarefa “Solicitar_Transferência_no_Banco”, uma alternativa para a primeira.

Nesta situação, aplicaremos a sub-diretriz M7.1, item vi.

M7.1.v - vários fins tais com, uma tarefa mapeada como uma ação Ac_1 , um objetivo bem definido mapeado como um atributo $Atrb_{obj}$, um recurso mapeado como um atributo de uma classe $Atrb_{rec}$, e um objetivo-soft mapeado como $Atrb_{objSoft}$, segundo as diretrizes de M2 a M4 e como meio uma tarefa, segundo M2, mapeada em uma ação, Ac , então teremos

axiom $[Ac] (obl(Ac_1) \wedge Atrb_{obj} \wedge Atrb_{rec} \wedge Atrb_{objSoft})$

Com base nesse item (v) da subdiretriz M7.1, teremos o seguinte axioma:

$[Solicitar_Transferência_no_Banco] ((Segura[Transferência] = posit) \wedge$
 $(Rápida [Transferência] = negat) \wedge (Conveniente[Transferência] = negat) \wedge$
 $(Serviço_Amigável[Transferência] = posit) \wedge Corr \wedge$
 $obl(Solicitar_Transferência_de_Fundos))$

Este axioma, no corpo da especificação em MAL Estruturado do agente Cliente, Figura6.22, é o de número 7.

- (v) um relacionamento meio-fim, onde a tarefa “Solicitar_Transferência_no_Home-Bank”, é o meio para se alcançar:
 - (a) o objetivos soft “Segura[Transferência]”, contribuindo de forma indefinida, significando que a transferência no Home-Bank não assegura esse objetivo-soft; essa situação será indicada atribuindo-se ao objetivo soft o valor indef.
 - (b) o objetivo soft “Rápida [Transferência]”, contribuindo positivamente, significando que a transferência no Home-Bank é mais rápida, pelo fato de não haver deslocamento até o estabelecimento; essa situação será indicada atribuindo-se ao objetivo soft o valor true;
 - (c) o objetivo soft “Conveniente[Transferência]”, contribuindo positivamente, significando que a transferência no Home-Bank é mais conveniente pelo fato de não ter-se que se deslocar até o

estabelecimento, enfrentar filas, etc.; essa situação será indicada atribuindo-se ao objetivo soft o valor true;

- (d) o objetivo soft “Serviço_Amigável[Transferência]”, contribuindo negativamente, significando que a transferência no Home-Bank é menos amigável, pelo fato de não existir o contato direto e pessoal entre as partes; essa situação será indicada atribuindo-se ao objetivo soft o valor false;
- (e) o recurso “PIN”, significando que o número de identificação pessoal deve ser fornecido no momento de realizar a tarefa. Esse número é representado pelo atributo “pin” do agente PIN;
- (f) a tarefa “Solicitar_Transferência_de_Fundos”, significando ser a tarefa “Solicitar_Transferência_no_Home-Bank”, uma alternativa para a primeira.

Com base nesse item (v) da subdiretriz M7.1, teremos o seguinte axioma:

$$\begin{aligned}
 &[\text{Solicitar_Transferência_no_Home-Bank}] ((\text{Segura}[\text{Transferência}] = \text{indef}) \wedge \\
 &\quad (\text{Rápida} [\text{Transferência}] = \text{posit}) \wedge (\text{Conveniente}[\text{Transferência}] = \text{posit}) \wedge \\
 &\quad (\text{Serviço_Amigável}[\text{Transferência}] = \text{negat}) \wedge \text{pin} \wedge \\
 &\quad \text{obl}(\text{Solicitar_Transfereência_de_Fundos}))
 \end{aligned}$$

Este axioma, no corpo da especificação em MAL Estruturado do agente Cliente, Figura6.22, é o de número 8.

- (vi) um relacionamento meio-fim, onde a tarefa “Solicitar_Transferência_por_Telefone”, é o meio para se alcançar:
 - (a) o objetivos soft “Segura[Transferência]”, contribuindo de forma indefinida, significando que a transferência no Home-Bank não assegura esse objetivo-soft; essa situação será indicada atribuindo-se ao objetivo soft o valor indef.
 - (b) o objetivo soft “Rápida [Transferência]”, contribuindo positivamente, significando que a transferência no Home-Bank é mais rápida, pelo fato de não haver deslocamento até o

estabelecimento; essa situação será indicada atribuindo-se ao objetivo soft o valor true;

- (c) o objetivo soft “Conveniente[Transferência]”, contribuindo positivamente, significando que a transferência no Home-Bank é mais conveniente pelo fato de não ter-se que se deslocar até o estabelecimento, enfrentar filas, etc.; essa situação será indicada atribuindo-se ao objetivo soft o valor true;
- (d) o objetivo soft “Serviço_Amigável[Transferência]”, contribuindo negativamente, significando que a transferência no Home-Bank é menos amigável, pelo fato de não existir o contato direto e pessoal entre as partes; essa situação será indicada atribuindo-se ao objetivo soft o valor false;
- (e) o objetivo soft “Confidencial[PIN]”, significando que a necessidade de manter secreto o número de identificação pessoal; essa situação será indicada atribuindo-se ao objetivo soft o valor true;
- (f) o recurso “PIN”, significando que o número de identificação pessoal deve ser fornecido no momento de realizar a tarefa. Esse número é representado pelo atributo “pin”, do recurso mapeado como agente PIN;
- (g) a tarefa “Solicitar_Transferência_de_Fundos”, significando ser a tarefa “Solicitar_Transferência_por_Telefone”, uma alternativa para a primeira.

Com base nesse item (v) da subdiretriz M7.1, teremos o seguinte axioma:

$$\begin{aligned}
 &[\text{Solicitar_Transferência_por_Telefone}] ((\text{Segura}[\text{Transferência}] = \text{indef}) \wedge \\
 &\quad (\text{Rápida}[\text{Transferência}] = \text{posit}) \wedge (\text{Conveniente}[\text{Transferência}] = \text{posit}) \wedge \\
 &\quad (\text{Serviço_Amigável}[\text{Transferência}] = \text{negat}) \wedge (\text{Confidencial}[\text{PIN}] = \text{posit}) \wedge \text{pin} \wedge \\
 &\quad \text{obl}(\text{Solicitar_Transferência_de_Fundos}))
 \end{aligned}$$

Este axioma, no corpo da especificação em MAL Estruturado do agente Cliente, Figura 6.22, é o de número 9.

- (vii) um relacionamento meio-fim, onde a tarefa “Cifrar_Dados”, é o meio para se alcançar:
 - (e) o objetivos soft “Segura[Transferência]”, contribuindo de forma positiva, significando que a cifragem dos dados pode garantir a segurança; essa situação será indicada atribuindo-se ao objetivo soft o valor true.
 - (h) o recurso “Chave_de_Cifragem”, significando que o cliente deve fornecer ao banco uma chave de cifragem. Essa chave é representado pelo atributo “ChCifragem” do recurso mapeado como agente Chave_de_Cifragem;
 - (f) a tarefa “Solicitar_Transferência_por_Telefone”, significando ser a tarefa “Cifrar_Dados”, uma alternativa para a primeira.

Com base nesse item (v) da subdiretriz M7.1, teremos o seguinte axioma:

$$[Cifra_Dados]((Segura[Transferência] = posit) \wedge ChCifragem \wedge \\ obl(Solicitar_Transfereência_de_Fundos))$$

Este axioma, no corpo da especificação em MAL Estruturado do agente Cliente, Figura 6.22, é o de número 10.

Na Figura 6.22, veremos todos esses axiomas incorporados à especificação do agente Cliente, em MAL Estruturado.

```

agent Cliente
.....
actions
  1 Solicitar_Transferência_de_Fundos;
  1 Solicitar_Transferência_no_Banco;
  1 Solicitar_Transferência_no_Home-Bank;
  1 Solicitar_Transferência_por_Telefone;
  1 Transfer_Personal_Data;
  1 Cifrar_Dados;
  1 Gerar_Chave_de_Assinatura;
  1 Escolher_Algoritmo_de_Cifragem;
attributes
  1 Atrbimplementável: enum{verd, falso, indef};
  1 Transferida (a1, a2, n): boolean;
  1 Ter_seu_PIN_Autorizado: boolean;
  1 Segura[Transferência]: enum {posit, negat, indef, sub, naosub};
  1 Cifrar_PIN: boolean;
  1 Rápida[Transferência]: enum {posit, negat, indef, sub, naosub};
  1 Conveniente[Transferência]: enum {posit, negat, indef, sub, naosub};
  1 Serviço_Amigável [Transferência]: enum {posit, negat, indef, sub, naosub};
axioms
1. [Cifrar_Dados] (Cifrar_PIN) ∧ obl(Gerar_Chave_de_Assinatura) ∧
                                     obl(Escolher_Algoritmo_de_Cifragem));
2. [Solicitar_Transferência_por_Telefone] (Seguro[PIN] = posit);
3. [Solicitar_Transferência_de_Fundos] (Transferida(a1, a2, n) ∧
                                     (Segura[Transferência] = posit));
4. [Solicitar_Transferência_de_Fundos](obl(Solicitar_Transferência_no_Banco) ∨
                                     obl(Solicitar_Transferência_no_Home-Bank) ∨
                                     obl(Solicitar_Transferência_por_Telefone));
5. [Solicitar_Transferência_por_Telefone](obl(Cifrar_Dados) ∨
                                     obl(Transfer_Personal_Data));
6. [Solicitar_Transferência_de_Fundos](Fundos_sejam_transferidos);
7. [Solicitar_Transferência_no_Banco] ((Segura[Transferência] = posit) ∧
                                     (Rápida [Transferência] = negat) ∧ (Conveniente[Transferência] = negat) ∧
                                     (Serviço_Amigável[Transferência] = posit) ∧ Corr ∧ obl(Solicitar_Transfereência_de_Fundos));
8. [Solicitar_Transferência_no_Home-Bank] ((Segura[Transferência] = indef) ∧
                                     (Rápida [Transferência] = posit) ∧ (Conveniente[Transferência] = posit) ∧
                                     (Serviço_Amigável[Transferência] = negat) ∧ pin ∧ obl(Solicitar_Transfereência_de_Fundos));
9. [Solicitar_Transferência_por_Telefone] ((Segura[Transferência] = indef) ∧
                                     (Rápida [Transferência] = posit) ∧ (Conveniente[Transferência] = posit) ∧
                                     (Serviço_Amigável[Transferência] = negat) ∧ (Confidencial[PIN] = posit) ∧ pin ∧
                                     obl(Solicitar_Transfereência_de_Fundos));
10. [Cifra_Dados]((Segura[Transferência] = posit) ∧ ChCifragem ∧
                                     obl(Solicitar_Transfereência_de_Fundos));

end agent Cliente

```

Figura 6.22 - As Ligações de Meio-Fim para o Agente Cliente, do Sistema Bancário, Especificadas em MAL Estruturado

Nesse estudo de caso foi apresentada uma situação, cujo objetivo foi o de oferecer alternativas ao sistema de transferência de fundos via telefone, que pudessem garantir a segurança das informações. Salientamos que não se desceu a níveis mais profundos, tais como, a escolha do algoritmo de criptografia a ser utilizado, etc.. A análise foi realizada a nível organizacional. Pode-se, entretanto, descer a níveis onde várias alternativas de implementação de rotinas de segurança sejam sugeridas, ficando esta questão a nível de requisitos não funcionais, aos quais graficamente com i* podemos nos referir como objetivos soft (*softgoals*).

As diretrizes apontadas no Capítulo 5 foram adequadas ao estudo de caso. Necessário se faz voltar a lembrar que não se trata de uma especificação completa, mas sim de oferecer meios para que requisitos do tipo organizacionais, capturados pelas técnicas de modelagem organizacional, possam ser integrados aos outros fragmentos que, por ventura, o especificador já tenha descrito precisamente.

6.2 – EXEMPLO DE DESCRIÇÃO DO MAPEAMENTO ENTRE A TÉCNICA I* E A LINGUAGEM OCL

Nosso objetivo é oferecer meios para que se possa exemplificar a utilização das diretrizes propostas no Capítulo 5 anterior. Especificamente nesta Seção, através de um estudo de caso, estaremos nos preocupando com as diretrizes relacionadas com a integração com a especificação pUML/OCL (Precise UML/Object Constraints Language).

6.2.1 – LOJA DE VENDA DE CD'S POR ENCOMENDA

Este estudo de caso subsidiou nossos exemplos para apresentação das técnicas de modelagem de requisitos no Capítulo 3. Retomando o modelo, iremos agora tratá-lo sob o ponto de vista de aplicação das diretrizes propostas no Capítulo 5, quanto à linguagem de especificação formal OCL.

Voltamos a enfatizar que o caso trata da definição de requisitos para um software que faça o controle das vendas de CDs por encomenda, de uma empresa real, que

trabalha no sistema de franquia, tem sete funcionários, vende por mês uma média de mil CDs e anuncia dispor de qualquer estilo musical. A clientela especial, caracteriza-se por fazer compras acima de dez unidades por trimestre, sendo esses clientes os mais representativos no faturamento da empresa. Além do estilo seletivo e de serem compulsivos compradores, estes clientes compram por catálogo.

Com vista a atrair estes clientes, a empresa sentiu necessidade de melhorar o serviço de vendas por encomenda, contratando a implantação de um sistema informatizado para controlar esse tipo de venda.

Com base nas definições preliminares, a primeira equipe de analistas manteve contato com a empresa de vendas de CDs e propôs, inicialmente, o desenvolvimento do modelo organizacional da empresa, utilizando a técnica de Bubenko [BUB, BUB95a, BUB95b, SOU98].

Nosso objetivo é promover o mapeamento dos requisitos do tipo organizacional, expressos através da técnica de modelagem i^* , para que esses sejam transformados em descrições precisas. No caso específico dessa Seção, trabalharemos com o mapeamento de i^* para o diagrama de classes de pUML, sob o qual aplicaremos a linguagem de restrições de objetos (OCL), segundo as diretrizes que foram propostas no Capítulo 5.

Para evitar nos reportarmos constantemente aos Capítulos anteriores, reproduziremos os modelos estratégicos segundo a técnica i^* , para a empresa em questão, assim como, nas situações necessárias, os trechos das diretrizes apontadas na Seção 5.2.2. Desta forma, na Figura 6.23, tem-se o Modelo de Dependências Estratégicas - SD para esta loja de Venda de CD's, que expressa o comportamento externo envolvendo os atores identificados pelo modelo.

No caso específico da Figura 6.23, temos que a organização de venda de CD's é composta por seis atores - Loja, Cliente, Fornecedor, Atendente, Caixa e Apoio. Começaremos, assim, pela aplicação da diretriz Oc1, com sua subdiretriz Oc1.1, obtendo as classes correspondentes, denominadas com os mesmos nomes utilizados nos

modelos estratégicos de i*, vide Figura 6.24. Essas classes representam o início do processo de mapeamento e, desta forma, tem-se um esboço inicial de um Diagrama de Classes em pUML.

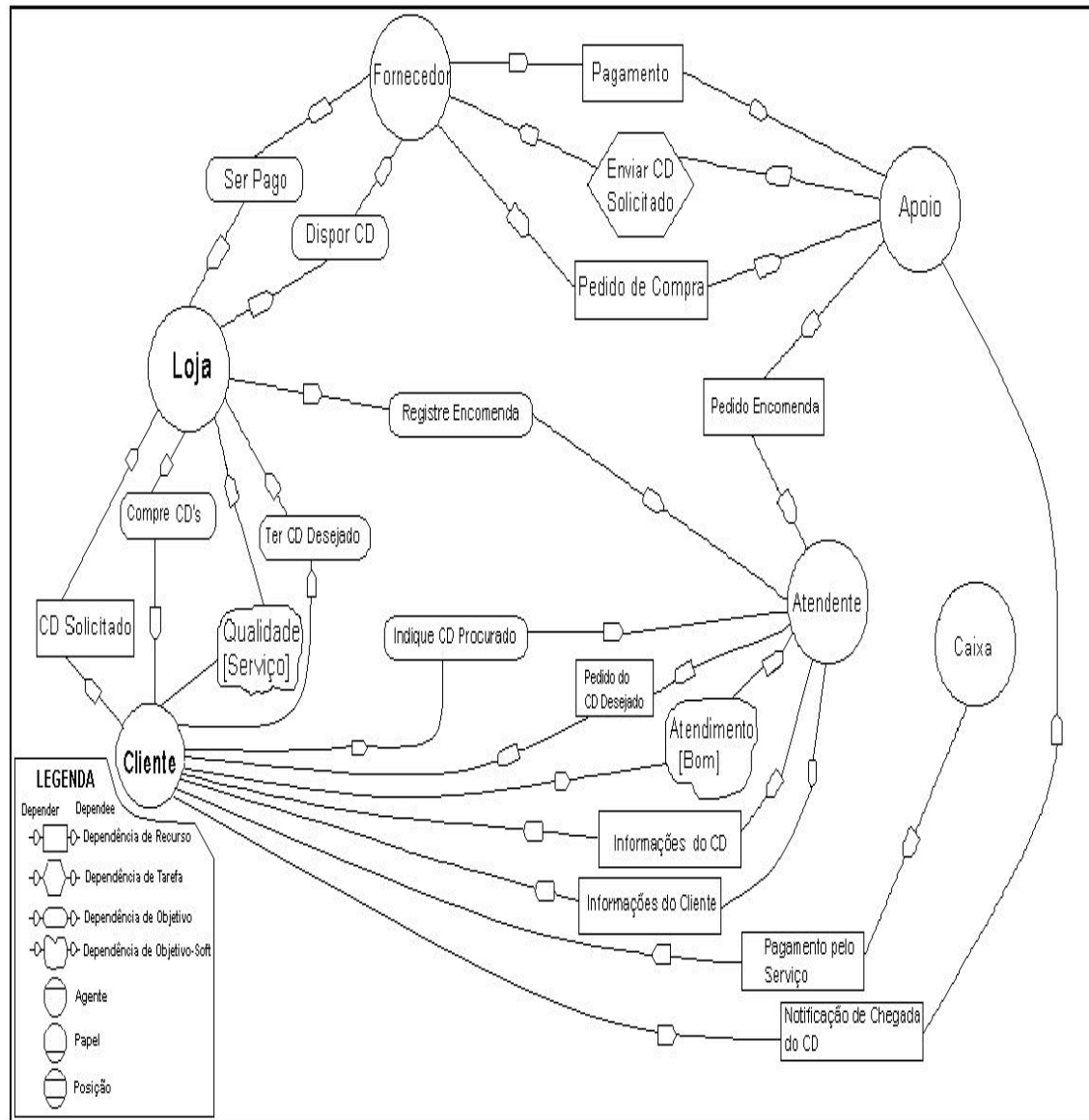


Figura 6.23 - Modelo de Dependências Estratégicas para a Loja de Venda de CD's

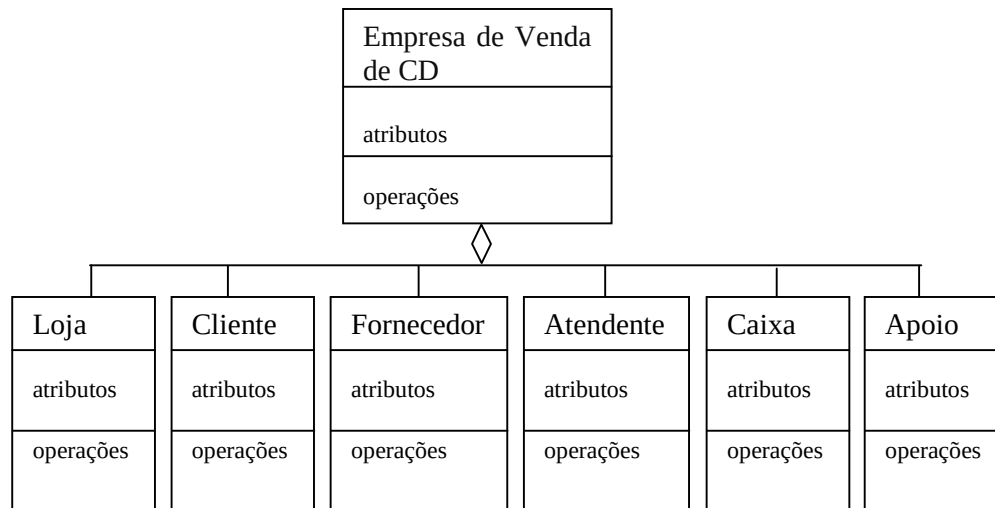


Figura 6.24 – Esboço Inicial do Diagrama de Classes, em UML, para a Loja de Venda de CD's

Na Figura 6.25, reproduzimos o Modelo de Razões Estratégicas - SR para a Loja de Venda de CD's. Neste modelo, apenas os atores “Loja”, “Atendente” e “Apoio”, aparecem detalhados. Outro fato observado é o de que os mesmos elementos que compõem as relações de dependência (tarefas, recursos, objetivos e objetivos-soft), também estão presentes nesse modelo. Continuemos, assim, a aplicação das diretrizes propostas na Seção 5.2.2.

Diretriz Oc2: Relacionada com o mapeamento das tarefas dos modelos i^* .

As **Tarefas** nos modelos estratégicos da técnica i^* são mapeadas como **operações** de uma classe em pUML.

$F_{\text{Tarefa}}: i^* \rightarrow \text{pUML}$

Tarefa $\rightarrow$ Operação

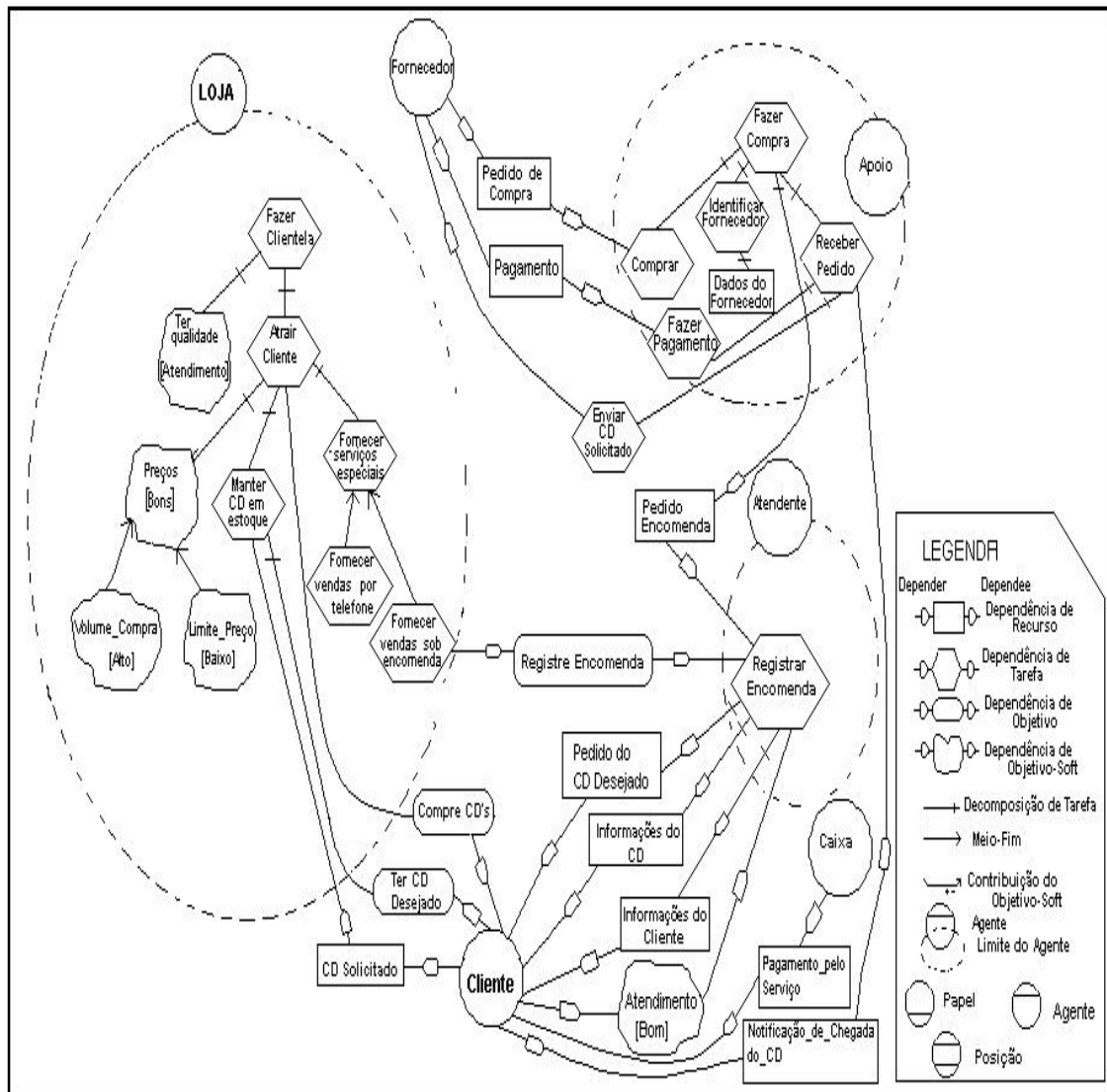


Figura 6.25 - Modelo de Razões Estratégicas para a Loja de Venda de CD's

SubDiretriz Oc2.1: Relacionada com o mapeamento das tarefas no Modelo de Dependências Estratégicas (SD).

Uma **Tarefa**, no Modelo de Dependências Estratégicas, será entendida como uma **operação pública** da classe relativa ao ator que seja responsável por realizá-la.

SubDiretriz Oc2.2: Relacionada com o mapeamento das tarefas no Modelo de Razões Estratégicas (SR).

Uma **Tarefa**, no Modelo de Razões Estratégicas, é mapeada como uma **operação local** da classe relativa ao ator que se esteja detalhando o comportamento.

Com relação a classe (agente) “Apoio”, verificamos a tarefa “*Enviar_CD*”, como elemento da dependência com a classe “*Fornecedor*”. Essa tarefa deve ser de responsabilidade dessa última classe, e portanto, descrita como uma *operação pública* no corpo da classe *Fornecedor*. Observemos que os atores “*Cliente*” e “*Caixa*”, não possuem tarefas associados, em nenhum dos modelos estratégicos (SD ou SR). Assim, considerando-se todas as tarefas, envolvidas nos dois modelos estratégicos, pela aplicação das subdiretrizes, chegamos ao especificado na Figura 6.26.

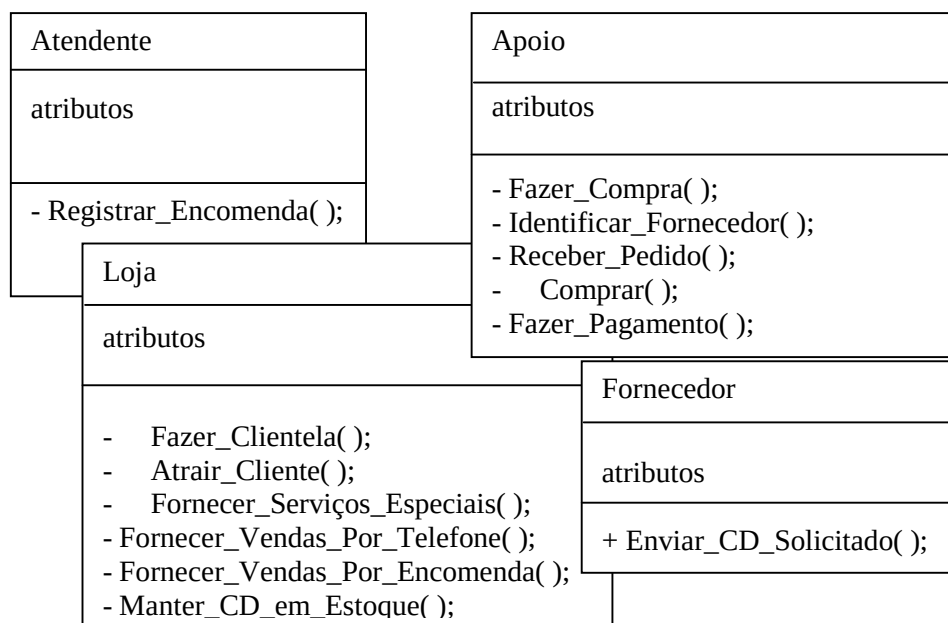


Figura 6.26 – Tarefas em i*, mapeadas em Operações sobre as Classes

Podemos observar que a maioria das operações têm a indicação de localidade “-“, dado que foram obtidas do comportamento interno dos agentes, no Modelo de Razões Estratégicas(SR), Figura 6.25. A operação da classe *Fornecedor* tem indicação de pública “+”, por ser elemento de dependência externa entre atores do Modelo de Dependências Estratégicas (SD), Figura 6.23.

Diretriz Oc3: Relacionada com o mapeamento dos recursos dos modelos i*.

Os **Recursos** dos modelos estratégicos da técnica i* são mapeados como **Classes** em pUML, com um **atributo público (*atrb_{rec}*) do tipo booleano**, que indique a disponibilidade desse recurso.

Conforme descrito na Seção 5.2.1, este mapeamento em uma classe pUML dá-se pelo fato de um recurso ser uma entidade física ou uma informação concreta. Procedendo-se na observação dos modelos estratégicos, identificamos todos os recursos (Pedido_Encomenda, Pedido_de_Compra, CD_Solicitado, Pagamento, Pedido_do_CD_Desejado, Dados_do_Fornecedor, Informações_do_Cliente, Pagamento_pelo_Serviço, Informações_do_CD e Notificação_de_Chegada_do_CD) e os mapeamos conforme a diretriz Oc3. Na Figura 6.27 podemos visualizar o resultado desse mapeamento.

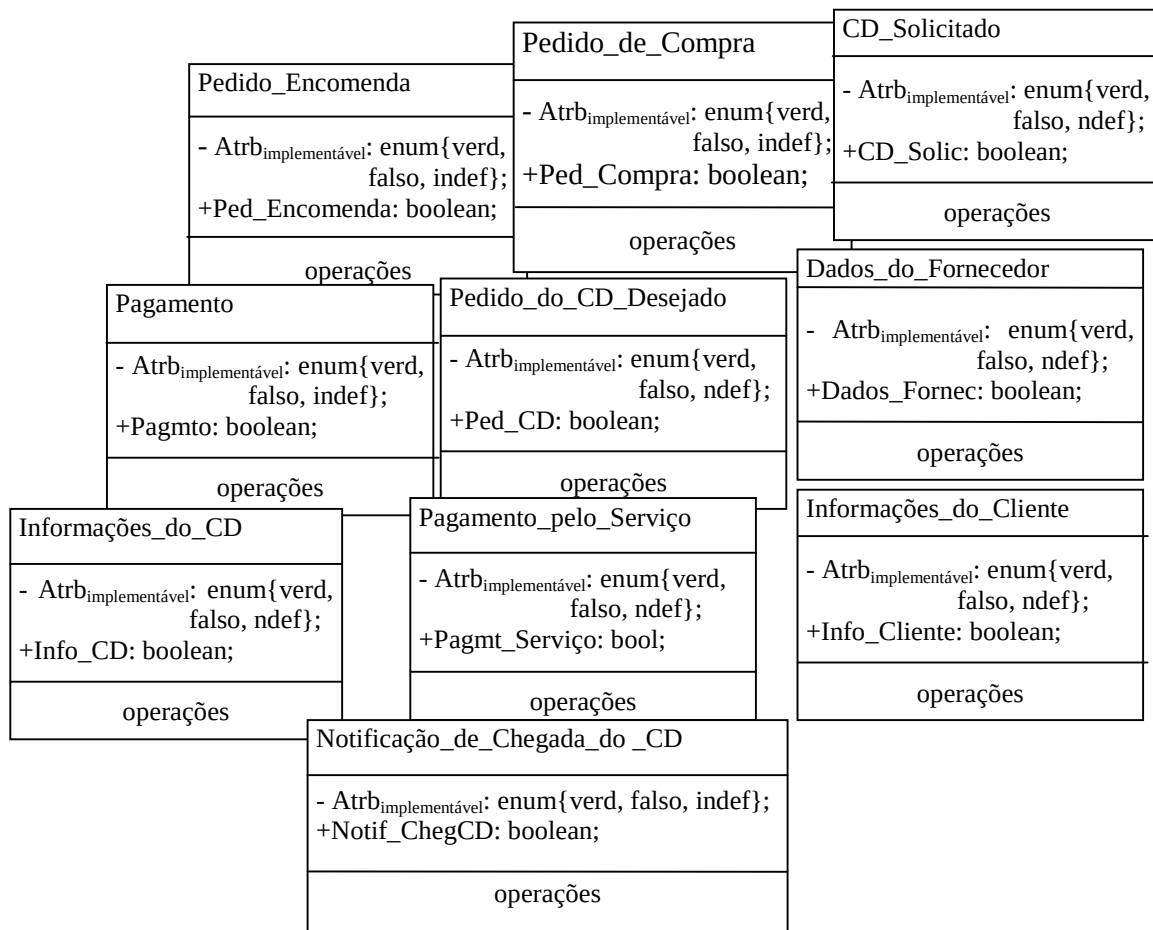


Figura 6. 27 – Recursos em i*, mapeados como Classes em UML

Diretriz Oc4: Relacionada com o mapeamento dos objetivos bem definidos e objetivos-soft dos modelos i*.

Os **objetivos bem definidos** e os **objetivos soft** dos modelos estratégicos serão mapeados como **atributos do tipo booleano e do tipo enumerado**, respectivamente, de uma classe pUML;

de forma que, os objetivos bem definidos sejam mapeados por

$$f_{\text{objetivos}}: i^* \rightarrow \text{pUML}$$

$$\text{Objetivos} \rightarrow \text{Atributo: boolean}$$

assim como, os objetivos-soft sejam mapeados por

$$f_{\text{objetivo-Soft}}: i^* \rightarrow \text{pUML}$$

Objetivos-Soft → Atributo: enum {posit, negat, indef, sub, naosub}

Da mesma forma, dependendo do modelo tratado, utilizamos as subdiretrizes Oc4.1, para o modelo SD, ou Oc4.2, para o modelo SR.

SubDiretriz Oc4.1: Relacionada com o mapeamento dos objetivos bem definidos e objetivos- soft no modelo de dependências estratégicas.

Objetivos e Objetivos-Soft, no Modelo de Dependências Estratégicas (SD) da técnica *i**, podem ser entendidos como **atributos booleanos público e enumerados públicos**, respectivamente, da Classe, em pUML, que corresponde ao ator a quem pertençam os objetivos.

SubDiretriz Oc4.2: Relacionada com o mapeamento dos objetivos bem definidos e objetivos- soft no Modelo de Razões Estratégicas (SR).

Os **objetivos bem definidos** e os **objetivos soft** internos a um ator que se esteja detalhando, em *i**, são mapeados como **atributos do tipo booleano local e do tipo enumerado**, respectivamente, da classe correspondente em pUML que se esteja observando o comportamento interno.

Observando-se o Modelo de Dependências Estratégicas da Figura 6.23, identificamos, para os atores “Loja”, “Atendente” e “Apoio”, alguns com objetivos *bem definidos*. Em relação ao ator “Loja”, identificamos um objetivo bem definido com relação ao “Cliente”, “Compre_CD’s” que será mapeado como um atributo booleano público da classe “Loja”. Esse atributo terá valor verdadeiro tão logo esse objetivo seja satisfeito.

Ainda com relação a “Loja”, existe:

- (i) um objetivo bem definido com relação a “Atendente”, “Registre_Encomenda”, que também será mapeado como um atributo público, do tipo booleano, da classe “Loja”; e

- (ii) um outro objetivo bem definido, numa dependência como o agente “Fornecedor”, “*Dispor_CD*”, que também será mapeado como um atributo público, do tipo booleano, da classe “Loja”.

Os demais atores mencionados, não apresentam nenhum dos tipos de objetivos.

Passando-se a observar o Modelo de Razões Estratégicas, Figura 6.25, observa-se que apenas o agente “Loja” possui objetivos-soft internos (“*Volume_Compra[Alto]*”, “*Limite_Preço[Baixo]*”, “*Preços[Bons]*” e “*Ter_Qualidade [Atendimento]*”). Aplicando-se a subdiretriz Oc4.2, mapeamos estes objetivos-soft como atributos locais da classe Loja, do tipo enumerado (true, false, indef).

Os demais atores (Atendente e Apoio), não possuem nenhum objetivo bem definido ou soft. Desta forma, na Figura 6.28, podemos identificar esses mapeamentos, para a classe Loja.

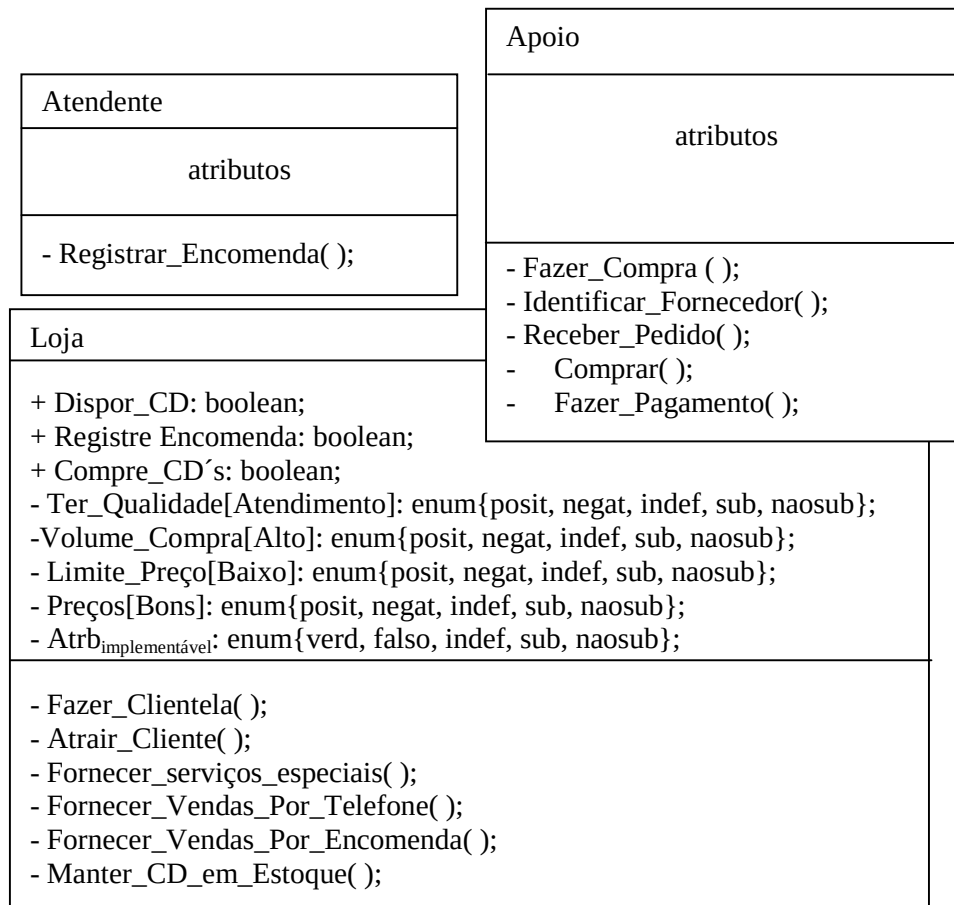


Figura 6.28 – Objetivos e Objetivos-Soft em i*, mapeados como Atributos do Tipo Booleano e Enumerado

Uma vez mapeados todos os elementos dos modelos estratégicos, podemos passar à observação das restrições que podem ser expressas em OCL. Assim, a partir da representação da classe Loja, Figura 6.28, poderemos começar a verificar restrições quanto às pré e pós-condições, relacionadas às operações da classe.

Como vimos anteriormente, Seção 3.2.2, surgem duas novas ligações no Modelo de Razões Estratégicas (SR) da técnica i*:

- (i) as ligações de decomposição de tarefas e
- (ii) as ligações meio-fim.

Uma vez mapeados os elementos para pUML, estaremos tratando do comportamento com relação as classes que correspondem aos atores, cujo comportamento é observado no Modelo de Razões Estratégicas - SR. Assim, para a determinação desse comportamento interno serão aplicadas as restrições em OCL (invariantes, pré e pós-condições), utilizando-se, em geral, do mecanismo de navegação entre as classes. Neste sentido, estabelecemos as diretrizes Oc5 e Oc6, respectivamente, para tratarem da decomposição de tarefas e do relacionamento meio-fim.

Diretriz Oc5: Relacionada com o mapeamento dos relacionamentos de decomposição de tarefa modelos i*.

*Uma ligação do tipo **decomposição de tarefa** é representada, em OCL, através de **pré-** e **pós-condição** na ação correspondente à tarefa que está sendo decomposta. Caso exista como elemento da composição uma sub-tarefa, suas pré-condições farão parte da pré condição da tarefa que está sendo decomposta. Na possibilidade de conjunção dos elementos são utilizados os conectivos lógico **AND**.*

Assim, se a operação (tarefa) Op é decomposta em uma sub-operação (sub-tarefa) Op₁, uma sub-objetivo Ob₁, um recurso Re₁ e um objetivo-soft ObSoft₁, então, segundo a diretriz, tem-se em OCL,

Operação Op

pre: (pre-condOp₁)

post: (pos-condOp₁) and(Atrb_{Re1}) and(Atrb_{ob1}= 'valor') and(Atrb_{obSoft1})

Ou seja, os elementos em que a tarefa é decomposta representam tudo o que é necessário para a execução dessa tarefa. Desta forma, esses elementos precisam que as pré-condições para a ocorrência da operação sejam satisfeitas e que todas as pos-condições sejam verdadeiras. O conectivo AND será utilizado para capturar este aspecto. No Modelo de Razões Estratégicas da Figura 6.25, encontramos algumas decomposições de tarefa. Considerando o ator *Loja*, identificamos:

- (i) a tarefa “Fazer_Clientela”, sendo decomposta na tarefa “Atrair_Cliente” e no objetivo-soft “Ter_Qualidade[Atendimento]”. Assim, aplicando a diretriz Oc5, temos,

Fazer_Clientela

pre: pre-condAtrair_Cliente

post: (pos-condAtrair_Cliente) and (Ter_Qualidade[Atendimento] = posit)

- (ii) a tarefa “Manter_CD_em_Estoque” será decomposta no objetivo “Compre_CD’s”. Assim, aplicando-se Oc5, teremos

Manter_CD_em_Estoque

pre: -- não se tem definida pelos modelo.

post: Compre_CD’s

- (iii) a tarefa “Fornecer_Vendas_sob_encomenda” será decomposta no objetivo “Registre_Encomenda”. Assim, aplicando-se Oc5, teremos

Fornecer_Vendas_sob_encomenda

pre: -- não se tem definida pelos modelo.

post: Registre_Encomenda

- (iv) a tarefa “Atrair_Cliente”, sendo decomposta:
 - (a) na tarefa “Manter_CD_em_Estoque”;
 - (b) na tarefa “Fornecer_Serviços_Especiais”;

- (c) no objetivo-soft “Preços[Bons]”; e
- (d) no objetivo “Compre_CD’s”.

Pela aplicação da diretriz Oc5, temos,

Atrair_Cliente

pre: (pre-Manter_CD_em_Estoque) and
 (pre-condFornecer_Serviços_Especiais)
 post: (pre-Manter_CD_em_Estoque) and
 (pre-condFornecer_Serviços_Especiais) and
 Compre_CD's and (Preços[Bons] = posit);

Na Figura 6.29a, podemos visualizar as restrições para a classe Loja, onde todas as cláusulas apontadas acima, aparecem na sintaxe OCL.

Loja: : Fazer_Clientela pre: pre-condAtrair_Cliente -- temos apenas a pré-condição da operação é que ela é decomposta. post: post: (pos-condAtrair_Cliente) and (Ter_Qualidade[Atendimento] = posit) -- aplicação da diretriz Oc5, decomposição de tarefa.
Loja: : Atrair_Cliente pre: (pre-Manter_CD_em_Estoque) and (pre-condFornecer_Serviços_Especiais) -- temos uma conjunção das a pré-condições das sub-operações. post : (pre-Manter_CD_em_Estoque) and Compre_CD's and (pre-condFornecer_Serviços_Especiais) and (Preços[Bons] = posit); -- aplicação da diretriz Oc5, decomposição de tarefa.
Loja: : Manter_CD_em_Estoque pre: -- não se tem definida pelos modelo. post: Compre_CD's -- aplicação da diretriz Oc5, decomposição de tarefa. <u>Loja: :Fornecer_Vendas_sob_encomenda</u>

Figura 6.29a – Relacionamento de Decomposição de Tarefa em i*, Expresso por Restrições em OCL, para a Classe Loja.

Considerando o ator *Atendente*, também identificamos uma decomposição de tarefa, com relação:

- (i) a tarefa “Registrar_Encomenda”, que é decomposta:
 - (a) no recurso “Pedido_do_CD_Desejado”;
 - (b) no recurso “Informações_do_Cliente”; e
 - (c) no recurso “Informações_do CD”.

Neste caso, os atributos das classes que representam os recursos necessários aparecerão como pós-condições. Deste modo, tem-se,

Registrar_Encomenda

post: Ped_CD and Info_Cliente and Info_CD;

Na Figura 6.29b, podemos visualizar as restrições para a classe *Atendente*, onde todas as clausulas apontadas acima, também aparecem na sintaxe OCL.

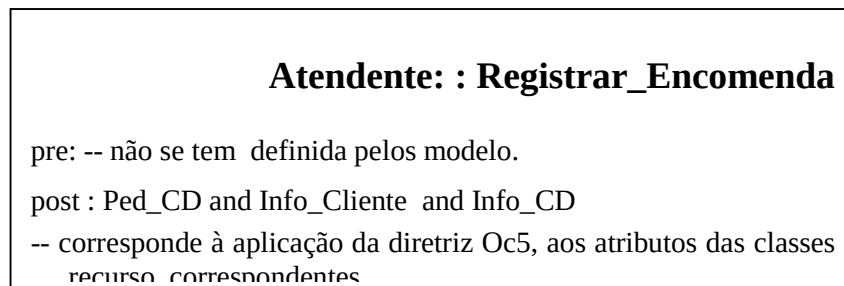


**Figura 6.29b – Relacionamento de Decomposição de Tarefa em i*,
Expresso por Restrições em OCL, para a Classe Atendente**

Com relação ao ator *Apoio*, também identificamos várias decomposições de tarefa, onde temos,

- (i) com relação a tarefa “Fazer_Compra”, decomposta em:
 - (a) uma sub-tarefa “Comprar”;
 - (b) uma sub-tarefa “Identificar_Fornecedor”;
 - (c) uma sub-tarefa “Receber_Pedido”; e
 - (d) em um recurso “Pedido_Encomenda”, considerado através de seu atributo booleano Ped_Encomenda.

Assim aplicando-se a diretriz Oc5 temos

Fazer_Compra

pre: (pre-condComprar) and (pre-condIdentificar_Fornecedor) and
(pre-condReceber_Pedido)

post: (pos-condComprar) and (pos-condIdentificar_Fornecedor) and
(pos-condReceber_Pedido) and Ped_Encomenda;

(ii) com relação a tarefa "Receber_Pedido", decomposta:

- (a) na sub-tarefa "Fazer_Pagamento"; e
- (b) na sub-tarefa "Enviar_CD_Solicitado".

Devemos observar que esta última tarefa é de responsabilidade do ator Fornecedor, mas é pública (Figura 6.26), por fazer parte de um relacionamento externo. Assim sendo, aplicando a diretriz Oc5

Receber_Pedido

pre: (pre-condFazer_Pagamento) and (pre-condEnviar_CD_Solicitado)

post: (pos-condFazer_Pagamento) and (pos-condEnviar_CD_Solicitado)

(iii) com relação a tarefa "Identificar_Fornecedor", decomposta em apenas um recurso, "Dados_do_Fornecedor". Lembrando que esse recurso será refletido através de seu atributo booleano Dados_Fornec (Figura 6.27). Assim sendo, aplicando a diretriz Oc5, temos

Identificar_Fornecedor

post: Dados_Fornec;

Na Figura 6.29c, podemos visualizar para a classe Apoio as restrições apontadas, na sintaxe OCL.

Apoio: : Fazer_Compra

pre: (pre-condComprar) and (pre-condIdentificar_Fornecedor) and
(pre-condReceber_Pedido)

post: (pos-condComprar) and (pos-condIdentificar_Fornecedor) and

**Figura 6.29c – Relacionamento de Decomposição de Tarefa em i*,
Expresso por Restrições em OCL, para a Classe Apoio**

Um segundo relacionamento importante é o de meio-fim. Conforme visto no Capítulo 3, em geral um meio é sempre uma tarefa e o fim pode ser um dos elementos: tarefa, objetivo bem definido, objetivo-soft ou um recurso. Porém, existem situações nas quais tanto o meio quanto o fim são objetivos bem definidos ou então, objetivos-soft. Estas situações são cobertas pela diretriz Oc6 com suas subdiretrizes Oc6.1, Oc6.2 e Oc6.3, abaixo citadas. Voltamos a lembrar que o elemento fim é aquele indicado pelo sentido da seta. Assim, consideraremos apenas as situações encontradas no Modelo Estratégico da Figura 6.25.

Diretriz Oc6: Trata das ligações do tipo meio-fim no modelo SR;

*Uma ligação do tipo **meio-fim** é representada, em OCL, através de **pré** e **pós-condições** nas operações correspondente ou **invariantes** de classe.*

SubDiretriz Oc6.1: Trata das ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é uma tarefa.

Uma ligação do tipo **meio-fim**, onde o **meio** é uma tarefa, e o **fim** é uma tarefa é representada, em OCL, através da inclusão de uma **pré-condição** na operação correspondente à **tarefa fim**, que pode ser uma conjunção (**AND**) das pré-condições das subtarefas, e da inclusão de uma **pós-condição** na operação correspondente à **tarefa fim**, em associação com o conectivo lógico **OR**.

Tendo-se uma operação (tarefa) **Op** como fim e como meio **n** tarefas, mapeadas para pUML, segundo D.9 ($Op_1, Op_2, \dots, Op_n$), então teremos

Operação Op

pre: (*pre-condOp₁*) and (*pre-condOp₂*) and ... and (*pre-condOp_n*)

post: (*pos-condOp₁*) or (*pos-condOp₂*) or ... or (*pos-condOp_n*)

SubDiretriz Oc6.2: Trata das ligações do tipo meio-fim, onde o meio é uma tarefa e o fim é um objetivo, um objetivo-soft ou um recurso.

Na operação que representa a tarefa meio, é adicionada uma **pós-condição** onde se tem uma **conjunção (AND)** das condições relacionadas com a ocorrência de quaisquer dos fins

Portanto, se tivermos a operação **Op** com meio e como fim um objetivo bem definido, **Atrb_{obj}**, um objetivo-soft, **Atrb_{objSoft}** que pode assumir um determinado valor (posit, negat ou indef) e um recurso a ser disponibilizado, **Atrb_{re}**, então,

Operação Op

post: **Atrb_{obj}** and (**Atrb_{objSoft}** = “valor”) and **Atrb_{re}**

Diretriz Oc63: Relacionada com a expressão dos relacionamentos meio-fim dos modelos i*, quando o meio e fim são, simultaneamente, objetivos bem definidos ou objetivos-soft.

É estabelecido um invariante de classe do tipo **meio implies fim**, onde meio poderá ser uma disjunção (**OR.**) dos atributos que representam os objetivos bem definidos ou objetivos-soft.

Portanto, se na classe **CL** está sendo observado um relacionamento meio-fim em que como fim tem-se um objetivo bem definido **Ob**, e como meio, **n** outros objetivos bem definidos (**Ob₁**, **Ob₂**, ..., **Ob_n**), então OCL,

Classe CL

(AtrbOb₁ or AtrbOb₂ or or AtrbOb_n) implies AtrbOb

A partir da Figura 6.25, analisemos as ligações meio-fim existentes, para os atores detalhados.

O ator Loja possui um relacionamento meio-fim do tipo:

- (i) meio e fim são tarefas; onde temos “Fornecer_Serviços_Especiais” como a tarefa fim e como meio temos:
 - (a) a tarefa “Fornecer_Vendas_por_Telefone”; e
 - (b) a tarefa “Fornecer_Vendas_sob_Encomenda”.

Desta forma, faremos uso da subdiretriz 13.1, donde

Fornecer_Serviços_Especiais

pre: (pre-condFornecer_Vendas_por_Telefone) and

(pre-condFornecer_Vendas_sob_Encomenda)

post: (pos-condFornecer_Vendas_por_Telefone) or

(pos-condFornecer_Vendas_sob_Encomenda)

Em nenhum outro ator do Modelo de Razões Estratégicas , Figura 6.25, identificamos um relacionamento meio-fim, onde meio e fim são todos tarefas.

Ainda no ator Loja, identificamos um relacionamento meio-fim do tipo:

- (ii) a tarefa “Manter_CD_Em_Estoque” é meio enquanto o recurso “CD_Solicitado” é o fim. Neste, caso em específico, aplicaremos a subdiretriz Oc6.2, onde tomamos o atributo da classe recurso “CD_Solic”, para indicar a sua disponibilidade ou não. Assim, será acrescida uma pós-condição à tarefa meio,

Manter_CD_Em_Estoque

post: CD_Solic

(iii) *objetivo-soft* em *objetivo-soft*, onde como fim temos “Preços[Bons] e como fim:

(a) o objetivo-soft “Volume_Compra[Alto]”; e

(b) o objetivo-soft “Limite_Preço[Baixo]”.

Nesta situação aplicaremos a subdiretriz Oc6.3, assim teremos um invariante na classe

Loja

(Volume_Compra[Alto] = posit) or (Limite_Preço[Baixo] = posit)

implies Preços[Bons] = posit;

Assim, à Figura 6.29a, acrescentamos essa última restrição, conforme pode ser visto na Figura 6.30a.

Loja: : Fornecer_serviços_especiais

pre: (pre-condFornecer_vendas_por_telefone) and
(pre-condFornecer_vendas_sob_encomenda)

post: (pos-condFornecer_vendas_por_telefone) or
(pos-condFornecer_vendas_sob_encomenda)

-- representam a aplicação da diretriz Oc6.1, ligação meio-fim.

Loja: : Manter_CD_em_Estoque

pre: -- não se tem definida pelos modelo.

**Figura 6.30a - Relacionamento Meio-Fim em i*,
Expresso por Restrições em OCL, para a Classe Loja.**

Observando os demais atores, cujo o comportamento é detalhado no Modelo de Razões Estratégicas (SR), Figura 6.25, tem-se no ator Atendente:

- (ii) a tarefa “Registrar_Encomenda” como meio de dois fins:
 - (a) “Registre_Encomenda”, relacionado com um objetivo bem definido da Loja; e
 - (a) “Pedido_Encomenda”, relacionado com um recurso que deve ser passado, representado aqui por seu atributo “Ped_Encomenda”.

Nesta situação, aplicaremos a diretriz Oc6.2, donde teremos

Registrar_Encomenda

Post: (Registre_Encomenda) and (Ped_Encomenda)

.

Desta forma, na Figura 6.30b pode ser visto a inclusão dessa pós-condição para a operação Registrar_Encomenda, da Figura 6.29b.

Apoio: : Registrar_Encomenda

pre: -- não se tem definida pelos modelo.

post : Ped_CD and Info_Cliente and Info_CD

-- corresponde à aplicação da diretriz Oc5, aos atributos das classes recurso, correspondentes.

post: (Registe_Encomenda) and (Ped_Encomenda)

**Figura 6.30b - Relacionamento Meio-Fim em i*,
Expresso por Restrições em OCL, para a Classe Atendente.**

Observe na Figura 6.25 que, para o ator Apoio, tem-se

- (i) a tarefa “Comprar” é um meio e para o recurso “Pedido_de_Compra” um fim, onde “Ped_Compra” é o atributo da classe “Pedido_de_Compra”;

Aplicando-se a subdiretriz Oc6.2, tem-se apenas uma pós-condição à operação

Comprar
post: Ped_Compra

- (ii) a tarefa “Fazer_Pagamento” como meio e o recurso “Pagamento” um como fim, onde “Pagmto” é o atributo da classe “Pagamento”;

Aplicando-se a subdiretriz Oc6.2, tem-se apenas uma pós-condição à operação

Fazer_Pagamento
post: Pagmto

- (iii) a tarefa “Receber_Pedido” como meio e o recurso “Notificação_de_Chegada_do_CD” como um fim, onde “Notif_ChegCD” é o atributo da classe “Notificação_de_Chegada_do_CD”.

Aplicando-se a subdiretriz Oc6.2, tem-se apenas uma pós-condição à operação que será acrescentada a especificação da operação na Figura 6.29c.

Receber_Pedido
post: Notif_ChegCD

Assim, na Figura 6.30c, tem-se as restrições em OCL para a classe Apoio.

Apoio::Comprar pre: -- não se tem definida pelos modelo. post: Ped_Compra --corresponde à aplicação da subdiretriz Oc6.2, ligação meio-fim.
Apoio:: Fazer_Pagamento pre: -- não se tem definida pelos modelo. post: Pagmto --corresponde à aplicação da subdiretriz Oc6.2, ligação meio-fim.
Apoio: : Receber_Pedido pre: (pre-condFazer_Pagamento) and (pre-condEnviar_CD_Solicitado) post: (pos-condFazer_Pagamento) and (pos-condEnviar_CD_Solicitado) -- corresponde à aplicação da diretriz Oc5, decomposição de tarefa. post: Notif_ChegCD --corresponde à aplicação da subdiretriz Oc6.2, ligação

Figura 6.30c - Relacionamento Meio-Fim em i*, Expresso por Restrições em OCL, para a Classe Apoio

Em algumas situações foi utilizada a navegação entre classes, princípio bastante forte de OCL. Na Figura 6.31, podemos visualizar as restrições em OCL encontradas para a classe Loja, segundo as Figuras 6.29a e 6.30a, o mesmo podendo ser feito para as classes Atendente e Apoio.

Loja: : Fazer_Clientela pre: pre-condAtrair_Cliente -- temos apenas a pré-condição da operação é que ela é decomposta. post: post: (pos-condAtrair_Cliente) and (Ter_Qualidade[Atendimento] = posit) -- aplicação da diretriz Oc5, decomposição de tarefa.
Loja: : Atrair_Cliente pre: (pre-Manter_CD_em_Estoque) and (pre-condFornecer_Serviços_Especiais) -- temos uma conjunção das a pré-condições das sub-operações. post : (pre-Manter_CD_em_Estoque) and Compre_CD's and (pre-condFornecer_Serviços_Especiais) and (Preços[Bons] = posit); -- aplicação da diretriz Oc5, decomposição de tarefa.

Figura 6.31 - Restrições em OCL sobre Classe Loja

Podemos observar que, algumas vezes, é útil acrescentarmos atributos e até mesmo operações extras, que podem ser usadas nas expressões OCL, para melhor definir (refinar) a especificação do que se deseja.

CONSIDERAÇÕES

Neste Capítulo apresentamos 3 (três) casos que nos orientaram quanto a aplicação das diretrizes de mapeamento, levantadas no Capítulo 5, para situações distintas, das quais duas foram aplicadas ao mapeamento para a linguagem de

especificação formal MAL Estruturado e a uma ao mapeamento para a linguagem pUML/OCL.

Observamos, quando da modelagem organizacional dos estudos de casos realizados, que os analistas devem começar focando sua atenção nos possíveis atores envolvidos no sistema e no ambiente no qual estará o sistema embutido.

Lembramos que, em todos os estudos de casos, foi necessário executar as tarefas de elicitação dos requisitos, muito embora nosso trabalho esteja inserido no contexto da etapa de modelagem, segundo o processo da Engenharia de Requisitos. Descrito no Capítulo 2, versões preliminares dos três estudos de casos, da Seção 6.1.1, 6.1.2 e 6.2.1, foram publicados em [ALE95, ALE97a, ALE97b, ALE97c, ALE98a, ALE98b, ALE99a, ALE99b].

De forma em geral, para todos os casos, na prática tivemos algumas limitações com relação a técnica i^* , no tocante à especialização dos atores em Agentes, Posições e Papéis. Nas empresas, normalmente, encontramos esse tipo de especialização dos atores, quanto a posição que ocupam na empresa e que atividades eles realizam dentro dessa posição. Todavia, essa limitação deve-se ao fato de que a estruturação dos atores na técnica i^* ainda não se encontra bem definida e é objeto de estudos futuros.

Outro fato é que como UML é muito amplo e ainda não de todo preciso, tomamos apenas o núcleo de UML, para o qual já existe uma precisão, o chamado UML preciso ou pUML (Precise UML). Em decorrência tomamos a parte de OCL que cobre esse núcleo, limitado-nos às restrições de invariantes, pré e pós-condições.

Também lembremos que todo o mapeamento proposto foi apenas unidirecional, no sentido de acrescentar informações do mundo organizacional às especificações precisas existentes, ou seja, caminhamos da abstração para a precisão. O sentido reverso, como já mencionado, é mais complexo e será objeto de trabalhos futuros.

Com especificações precisas, nos estudos de casos, fomos capazes de identificar erros cruciais na modelagem, que antes não havíamos percebido. Por exemplo, na modelagem de algumas ligações do tipo meio-fim, por questão de interpretação, havíamos invertido o elemento fim, fato só percebido no momento da inclusão das restrições. Quando da colocação dos axiomas e restrições, percebemos a necessidade de reavaliação. Assim, com esse mapeamento tem-se um potencial para a prova de

propriedades e animação das especificações, de forma a não incorrermos em erros grosseiros ou gravíssimos. Todavia, a prova de propriedades ou animação não é no momento o objeto de nosso trabalho. Concentramo-nos nas diretrizes para o mapeamento.

Como ferramenta de apoio existe o OME, em desenvolvimento pelo grupo de Toronto, para a modelagem com o i^* . Todavia, esta ferramenta não foi utilizada neste trabalho, por não ser uma versão estável que constantemente está sofrendo alterações; hoje se encontra na versão 3. Obviamente, esse fato representou uma grande dificuldade, no momento da elaboração dos diagramas. Principalmente, quando a complexidade dos mesmos aumentou. Uma outra ferramenta que existe em desenvolvimento, é o Multiview, que permite gerar especificações precisas na linguagem MAL Estruturado, sustentado pelo método VSCS. É nossa intenção, no futuro tentar integrar essas ferramentas.

No próximo Capítulo apresentaremos algumas colocações, destacando-se as contribuições deste trabalho que achamos relevantes, os trabalhos relacionados, bem como as metas quanto a trabalhos futuros.

CAPÍTULO VII

CONCLUSÕES

7.1 – SUMÁRIO DOS RESULTADOS

Neste trabalho começamos apresentando uma retrospectiva das principais prerrogativas, conceitos e etapas do processo da Engenharia de Requisitos, onde levantamos a necessidade de melhor compreendermos o domínio da aplicação, o problema específico a ser resolvido, as necessidades e limitações organizacionais e as facilidades específicas, necessárias às partes interessadas.

Dentre as atividades do processo da Engenharia de Requisitos, enquadramos este trabalho na atividade de modelagem, tratando especificamente dos requisitos categorizados como organizacionais, relacionados com as metas, políticas estratégicas adotadas e com os empregados da empresa e seus respectivos objetivos, abarcando aqueles requisitos que dizem respeito à estrutura da organização.

A Motivação deste trabalho se deve a termos identificado que existem, na área da Engenharia de Requisitos, alguns problemas em aberto, dentre os quais, especificamente, os que tratam de aspectos relacionados com organização e que, em geral, não são considerados.

Tradicionalmente, nos processos da Engenharia de Requisitos, encontram-se tipicamente preocupações apenas com os aspectos do ponto de vista técnico-funcional, referentes aos sistemas que se deseja especificar. Não são considerados aspectos relacionados com as organizações, aspectos esses importantes à compreensão das reais necessidades e intenções dessas organizações e alternativas de modelagem, fatos que, em geral, não são relevados, mesmo em técnicas que orientam o estudo da viabilidade dos sistemas que se deseja implementar. A compreensão dos requisitos organizacionais muito contribuirá para a uma especificação que satisfaça as reais necessidades do Cliente.

Nesse caminho, foi importante observar as principais técnicas para a modelagem dos requisitos (funcionais, não funcionais e organizacionais) existentes, sob o ponto de vista da Engenharia de Requisitos (Técnicas Computacionais).

Com base na literatura pesquisada, constatamos que a maioria das técnicas faz uso de modelos conceituais de natureza gráfica, que fornecem uma descrição dos *quês*, mas não dos *porquês* existentes no processo, sendo mal equipadas para ajudar a responder algumas perguntas que objetivam atingir estes *porquês*. Responder os *porquês* é mais importante, não apenas por ajudar, em primeira instância, no desenvolvimento com sucesso do sistema, mas também por facilitar o desenvolvimento de cooperação com outros sistemas, bem como a evolução desses sistemas.

Hoje, porém, na área da Engenharia de Requisitos, vê-se um grande número de trabalhos que direcionam para a necessidade de modelos conceituais capazes de lidar com agentes, objetivos e intenções [DAR93, LOU95, YU97c, MYL95, MYL99], de modo a retratar, de forma mais ampla, as necessidades do mundo real, bem como outros trabalhos que buscam tornar as definições cada vez mais precisas [DAR97, YU97a, DUB98, LEV98].

A definição de Engenharia de Requisitos tem evoluído passando hoje a ter maior destaque, sobretudo por se preocupar com a comunicação com atores organizacionais. Assim, a modelagem organizacional e a classificação dos requisitos em funcional, não funcional e organizacional, passam a ser primordiais.

Dentre as técnicas de modelagem dos requisitos organizacionais, sob o ponto de vista da Ciência da Computação, uma avaliação mais criteriosa foi realizada recentemente entre as técnicas *i** e Bubenko [ALE99b]. Nesta comparação, alguns pontos característicos foram levados em consideração e, a conclusão obtida foi a de que as técnicas *i** e Bubenko têm poder de expressão similar. Pelo que foi avaliado das demais técnicas de modelagem organizacional, em linhas gerais, para o objetivo desse trabalho, qualquer uma das técnicas teria sido adequada; evidentemente, que em função das peculiaridades de cada uma quanto à notação adotada, pequenas adaptações poderiam ser necessárias.

Destacamos também, que na busca por técnicas que se preocupam com aspectos relativos a estrutura dos agentes organizacionais, fomos levados à pesquisa sob a ótica

da Ciência da Administração. Constatamos que as técnicas utilizadas possuem notação gráfica muito simples, tais como diagramas de fluxo de dados, entidade-relacionamento, work flow e organogramas. Estas técnicas são pouco expressivas, sendo de pouca ajuda aos engenheiros de software, pois não permitem análises mais precisas, avaliação de alternativas para o processo de negócio, o estabelecimento de metas e estratégias, etc., a fim de que se tenha uma definição do contexto mais ampliada e uma visão mais próxima da realidade. A rigor, esses aspectos são considerados pela Administração, é bem verdade, mas de forma muito vaga, através de tabelas e informações textuais.

Identificamos também que em sistemas críticos, distribuídos e de segurança, faz-se necessário técnicas de especificação mais precisas. De forma geral, uma linguagem de especificação formal é composta por:

- (1) uma *sintaxe* que define a notação própria da especificação e que permite que os requisitos sejam interpretados de forma única;
- (2) uma *semântica* que ajuda a definir um universo de objetos que serão utilizados na descrição do sistema; e
- (3) um *conjunto de relações* que define as regras que indicam que objetos satisfazem a especificação.

As propriedades de uma especificação precisa – ausência de ambigüidade, consistência e completude – constituem-se nos objetivos de todos os métodos de especificação.

Vimos que técnicas precisas para a especificação de requisitos emergem lentamente e aos poucos são utilizadas pela indústria no desenvolvimento de software, especialmente quando se trata de sistemas críticos onde estão envolvidos aspectos de segurança [BOW94a, VAS95, PIN96, CAS96, REI97, REI98, LIM98]. Todavia, salientamos que os métodos precisos preocupam-se, fundamentalmente, com os requisitos funcionais do sistema que se deseja especificar.

Apesar de ser evidente a necessidade de utilização dos métodos precisos, sua ampla utilização entre os desenvolvedores ainda é baixa, devido a fatores tais como:

- (i) dificuldade de entendimento das notações;
- (ii) dificuldade na formalização de certos aspectos de requisitos; e
- (iii) o próprio conservadorismo do desenvolvimento de software.

É importante lembrar ainda que não existe um único método perfeitamente adaptado a uma aplicação, em particular, devido a limitações na expressividade, a habilidade de manipular grandes especificações, a disponibilidade de ferramentas de suporte e outras restrições.

A escolha de um ou outro método a ser adotado depende de quais propriedades serão modeladas e a praticidade ou não de criar extensões do método. Assim, a utilização de linguagens precisas, para a descrição de requisitos em sistemas complexos, pode recair em duas principais dificuldades [CAS96, CAS97, ALE99a]:

- (i) o escopo limitado das linguagens; e
- (ii) a aquisição preliminar dos requisitos relevantes do sistema, tais como, aspectos comportamentais, principalmente no tocante aos requisitos organizacionais, que não são considerados nas especificações.

Observando a importância que tem o Mundo da Modelagem Organizacional para a melhor compreensão dos requisitos e a importância do Mundo Preciso para sistemas críticos, identificamos um vazio (gap) entre esses dois mundos. Neste sentido é que propusemos diretrizes que possibilitam mapear os requisitos organizacionais, capturados pelas técnicas de modelagem específicas, conforme vimos na Seção 3.2, para especificações de requisitos funcionais, já existentes em linguagens precisas.

Nosso objetivo é o de contribuir para o aprofundamento do conhecimento sobre os “porquês” e mesmo o “como” o sistema deve ser desenvolvido para funcionar adequadamente, com o mínimo de falhas e custos, dentro dos prazos de projeto previstos.

Como meio, para estabelecermos o mapeamento, optamos por utilizar a técnica i* para modelar os requisitos organizacionais, através dos seus dois modelos:

- (i) o de dependências estratégicas (SD) e o de razões
- (i) dispor de uma rica ontologia, sendo graficamente mais concisa;
- (ii) tratar com atores, unidades semi autônomas, semelhante a idéia de orientação a objeto;
- (iii) permitir a compreensão mais aprofundada do processo através de uma visão intencional e estratégica; e

- (iv) possibilitar a análise das razões envolvidas nos processos de decisões.

Sob o lado da especificação precisa, utilizamos as linguagens MAL Estruturado (Modal Action Logic) e o pUML(preciseUML)/OCL (Object Constraints Language). Fundamentando nossa opção pela linguagem MAL Estruturado em:

- (i) ter-se divisão das especificações em partes menores através dos mecanismos de estruturação;
- (ii) possibilitar especificações como um conjunto de descrições de agentes (objeto) que realizam ações, semelhante aos conceitos de orientação a objeto;
- (iii) ser uma linguagem em uso no campo industrial;
- (iv) ter sido acoplada ao ambiente Multiview [CAS96, CAS97] desenvolvido no Departamento de Informática da Universidade Federal de Pernambuco; e
- (v) possibilitar ordenação na ocorrência das várias tarefas representadas nos modelos estratégicos da técnica organizacional selecionada.

Ao passo que a opção por OCL deveu-se a:

- (i) ser esta linguagem parte do padrão orientado a objeto, UML [JACO99], bastante em uso; e
- (ii) ter como proposta ser mais voltada aos engenheiros de desenvolvimento de sistemas.

A possibilidade de trabalharmos com a linguagem de especificação Z [WOO96] foi considerada para o presente trabalho. Todavia, observamos que esta linguagem não se aplicaria diretamente ao nosso presente propósito, em virtude da técnica de modelagem organizacional i^* possuir aspectos orientados a objetos. Assim, consideramos utilizar uma variação orientada a objeto de Z, denominada de MOOZ (Modular Object Oriented Z) [MEI90]. Ocorre que o requisito fundamental de ser uma linguagem que esteja em uso e a necessidade de uma base matemática consolidada, demoveu-nos de utilizá-la, no momento. A posteriori, (vide seção 7.4), examinaremos a possibilidade de estendermos o trabalho atual para lidarmos com a linguagem MOOZ.

Para melhor evidenciar a aplicabilidade das diretrizes de mapeamento proposta foram utilizados três estudos de casos, que refletem preocupações e situações distintas, das quais duas foram aplicadas para averiguar o mapeamento com relação à linguagem de especificação precisa MAL Estruturado e uma para averiguar o mapeamento com a linguagem pUML/OCL.

Os mapeamentos propostos tanto de i* para MAL Estruturado, quanto de i* para pUML/OCL, restringiram-se a uma direção, ou seja, do mundo abstrato (modelagem organizacional) para o mundo mais preciso e concreto. Isso porque no processo de refinamento são acrescentadas informações adicionais que no processo reverso, fatalmente serão perdidas. Deixamos como proposta de trabalho futuro, observar o mapeamento no sentido reverso.

No processo de mapeamento não foi considerado o princípio de especialização dos atores da técnica i*, em Agentes, Posições e Papéis, por não estarem bem definidos e ainda serem objeto de pesquisa, por seus autores em Toronto. Da mesma forma que tomamos apenas um núcleo de UML, por está mais bem definido e que é chamado de pUML. Para esse núcleo associamos a linguagem de restrições de Objetos (OCL) cuja semântica ainda não se encontra bem definida e aceita pela OMG. Um grupo de cientistas internacionais, que compõe o grupo do pUML (Precise UML), está com pesquisa em desenvolvimento neste sentido.

Ressaltamos que, em todos os estudos de casos, foi necessário proceder a tarefa de elicitação dos requisitos, muito embora o contexto de nosso trabalho não seja esse. Nossos estudos nos levam a concluir que nós, desenvolvedores de produtos tecnológicos, devemos ampliar nossa visão e considerar que só o emprego de modernas tecnologias de automação, informação e de comunicação, associadas a um trabalho organizacional de alto nível, pode produzir soluções importantes na conquista por produtos de alta qualidade, que conduzam à satisfação e conquista de clientes.

7.2 – TRABALHOS RELACIONADOS

Conforme mencionamos no capítulo 1, existem alguns trabalhos que tentam integrar as ferramentas formais com a técnica i* [YU95a], que captura os aspectos organizacionais.

Em [YU95b] os autores propõem mostrar como duas ferramentas orientadas a agente (ALBERT e i*) podem ser usadas em combinação durante a fase de requisitos do desenvolvimento, para sistemas de informação cooperativos. A linguagem ALBERT é usada para especificar os requisitos em termos de estados, ações, informação e percepção, enquanto a técnica i* é usada na compreensão do processo organizacional, em termos dos relacionamentos e razões estratégicas. Pela união da duas ferramentas, espera-se poder ganhar confiança nas especificações em ALBERT, ligando-se cada fragmento de especificação em ALBERT ao cumprimento de alguns objetivos expressos nos modelos i*. Assim sendo, segundo esses objetivos apontados pelo i*, são feitas alterações a nível de especificação funcional, em ALBERT, ao sistema em desenvolvimento, para que o objetivo seja considerado. Não é feito nenhum mapeamento entre os elementos dos modelos i* e os elementos em ALBERT, como fazemos em nossa proposta de mapeamento para MAL e para pUML/OCL, facilitando o trabalho do especificador, com informações mais precisas.

Em um segundo trabalho [DUB98] encontramos três linguagens formais sendo utilizadas para a modelagem de requisitos, o KAOS, ALBERT II e um Máquina de Estado, em conjunção com a técnica i*, aqui usada para propiciar a ligação entre esses modelos formais e fornecer um modelo de mais alto nível, capaz de informar quais aspectos organizacionais são capturados.

Neste trabalho [DUB98], são considerados três atividades distintas e conectadas: (i) a modelagem dos objetivos associados com a introdução de algum software na empresa; (ii) a modelagem dos requisitos do software; e (iii) a modelagem do comportamento funcional do software e dos protocolos utilizados. A linguagem KAOS [DAR93] é utilizada para se raciocinar sobre os objetivos do sistema. A linguagem ALBERT II [BOIS95, BOIS97] é utilizada para a especificação dos requisitos do sistema, enquanto que a Máquina de Estado para a modelagem do comportamento interno do sistema.

Como resultado do processo de especificação, tem-se três produtos distintos. É mostrado, então, como a técnica i*, a nível de requisitos, é utilizada na complementação de cada um dos produtos e como os modelos evoluem a partir dos produtos especificados. Neste trabalho, também não é proposto um mapeamento entre os elementos dos modelos, organizacional e precisos.

7.3 – CONTRIBUIÇÕES

Apesar da integração entre a modelagem organizacional e a modelagem precisa dos requisitos também terem sido objeto de estudo de vários pesquisadores, haja visto os trabalhos de [YU97a, WIE98, CIA98], o que estamos propondo são mapeamentos entre objetos do mundo organizacional e objetos do mundo preciso. Isso com relação ao mapeamento de i^* para Structured MAL, destacando-se que além de se poder tratar com componentes deônticos para forçar uma ordem na ocorrência das tarefas em i^* , podemos ainda fazer uso do aspecto temporal encontrado em Structured MAL. Deste modo, afirmamos que mapeamentos não foram tratados em nenhum dos trabalhos relacionados. Com relação ao mapeamento para pUML/OCL, segundo as pesquisas realizadas, nenhum dos processos que implementam UML, tratam muito bem dos aspectos organizacionais. Neste sentido, nosso trabalho é totalmente original, por permitir a representação desses aspectos em UML e propor meios para que se aplique as restrições em OCL.

Salientamos ainda que as linguagens adotadas nesta proposta de trabalho, Structured MAL e pUML/OCL, têm semânticas e poder de modelagem distintas das que foram consideradas pelos outros pesquisadores (seção 7.2) e que desconhecemos a existência de alguma proposta de mapeamento entre a técnica i^* e as linguagens Structured MAL e pUML/OCL (parte do padrão UML) até o momento.

Conforme salientamos, as diretrizes propostas no momento, neste trabalho, conduzem ao mapeamento em um direção, ou seja, da modelagem organizacional para a modelagem formal, fornecendo meios para que os requisitos ainda iniciais possam ser verificados. Com o mapeamento é possível se estabelecer provas de propriedades, bem como a animação das especificações desses requisitos, que não constituem objeto de nosso trabalho, mas que serão considerados no futuro. Com a incorporação de mais precisão aos aspectos organizacionais modelados, e com as possibilidades de prova de propriedades e animação levantadas acima, será possível dirimir que qualquer ambigüidade possa interferir num processo de decisão da organização. Tomemos o exemplo, da Envasadora de Água Mineral, é de fundamental importância para os Clientes que se tenha a garantia de todo o processo de tratamento dos garrações, que vai da Inspeção até o Enxágüe, a fim de que seja possível garantir a qualidade do produto

entregue. Assim através de definições mais precisas é possível se detectar possíveis falhas no processo e corrigi-las mais cedo.

Portanto, acreditamos que as diretrizes propostas nesse trabalho ajudam a reduzir a distância peculiar existe entre o mundo da especificação precisa, que trata de aspectos técnicos relativos à modelagem dos requisitos organizacionais e à modelagem da funcionalidade dos sistemas desejado, que aprofundam a compreensão sobre as necessidades e porquês envolvidos e que, diretamente, influenciam na funcionalidade e na eficácia dos sistemas que se deseja especificar. Desta forma, teremos associado às especificações formais, aspectos, relativos aos requisitos organizacionais, que antes não eram, explicitamente considerados.

7.4 – FUTUROS DIRECIONAMENTOS

Nesta seção, são levantadas algumas possibilidades de futuros trabalhos tendo por base as idéias inicialmente levantadas neste trabalho. Assim sendo:

1. Desenvolvimento de Ferramenta para o Mapeamento

Sempre que um método é usado ou proposto, mais cedo ou mais tarde, será necessária uma ferramenta para suportá-lo. Já existem ferramentas que suportem a técnica i^* , no caso a OME, que se encontra em uma terceira versão (OME3), e um ambiente MULTIVIEW, que permite gerar especificações precisas na linguagem MAL Estruturado, sustentado pelo método VSCS [CAS96, CAS97]. Tem-se, como sugestões de trabalhos futuros, a integração dessas ferramentas sobre um único ambiente e o desenvolvimento e integração, nesse ambiente único, de uma ferramenta capaz de auxiliar as diretrizes aqui propostas, de forma a permitir, automaticamente, a integração dos requisitos organizacionais capturados com a utilização da técnica i^* com as especificações funcionais formais obtidas na linguagem MAL Estruturado e geradas pelo MULTIVIEW. Desta forma possibilitar a animação das especificações de requisitos;

2. O Mapeamento Reverso

Estudar o caminho reverso do mapeamento, ou seja, estudar as possibilidades de se tratar mapeamentos tanto do MAL Estruturado para i^* , como de descrições em pUML/OCL para i^* . Desta forma, podemos obter os requisitos organizacionais a partir das especificações precisas.

3. Tratar a Estruturação dos Atores em i^*

Esclarecer os conceitos atribuídos às “especializações” de atores em – agente, posição e papel – dado que ainda são obscuros por ausência de uma semântica bem definida. Feito isso, propor um mapeamento desses conceitos nas linguagens formais MAL Estruturado e pUML/OCL.

4. Utilização de outras Linguagens Precisas

É sabido que a linguagem de especificação precisa Z continua sendo utilizada, mas como em i^* temos aspectos de orientação a objeto, poderemos considerar a possibilidade de utilização de MooZ como possível linguagem para descrição precisa.

5. Utilização de MAL Estruturado para Atribuição de Ordenação das Tarefas

Poderemos utilizar o poder de expressão da linguagem MAL Estruturado para descrever a seqüência de tarefas em i^* . Por exemplo, na Loja de venda de CD’ vimos algumas decomposições de tarefa em sub-tarefas, mas não encontramos qualquer definição sobre que sub-tarefa deve ser executada primeiro.

6. Avaliação da possibilidade de existência de algum morfismo entre Structured MAL e OCL.

Apesar das linguagens precisas aqui apontadas tem poder de expressão distintos, pode-se pensar em identificar algum morfismo entre estas, de forma a garantir que não haverá inconsistência naquilo que foi especificado.

Isto posto, podemos concluir que muito ainda se encontra em aberto. Com esse trabalho almejamos começar a levantar as idéias, mas certamente, muitos desafios ainda estão por serem vencidos. Acreditamos, todavia, que nosso caminho traçado,

originariamente, foi percorrido e alcançamos nossos maiores desafios, quais sejam o de ressaltar a importância dos requisitos organizacionais para a especificação dos requisitos dos sistemas e o de contribuir para que esses requisitos possam ser traduzidos para o mundo das especificações precisas, garantindo a satisfação das necessidades do cliente. Exposta a proposta desta nova forma de pensar e de modelarmos os requisitos de sistemas, resta-nos esperar seus desdobramentos futuros, que esperamos serem benéficos a todos.

REFERÊNCIAS BIBLIOGRÁFICAS

- [ABR81] ABRIAL, J-R: "A Course on System Specification". RAISE/DDC/EM/A PRG-Oxford University, Oxford, UK, 1981.
- [ALE95] Alencar, F. M. R. and Castro, J. B. C.: "Formalização de Requisitos - Melhoria na Qualidade dos Softwares". Anais do Workshop Qualidade em Software – SBES'95. pp. 56-60. Recife-PE. Out. 1995.
- [ALE97a] Alencar, F. M. R. and Castro, J. F. B.: "Uso de Modelos Intencionais em Sistemas de Comunicação". XV Simpósio Brasileiro de Telecomunicações - TELEMO'97. pp. 573-576. Recife-PE, Brasil. Set. 8-11, 1997.
- [ALE97b] Alencar, F. M. R. and Castro, J. F. B.: "Utilizando Lógica Modal de Ações para Formalização de Requisitos Organizacionais (in Portuguese)". In Proceedings of XXIII Conferência Latinoamericana de Informática, Valparaíso, Chile, Nov. 1997, pp. 515-524.
- [ALE97c] Alencar, F. M. R. and Castro, J. B. C.: "Modelando Intenções e Melhorando a Captura de Requisitos de Qualidade". Anais do Workshop Qualidade em Software - COOPE. Rio de Janeiro - RJ, pp. 1-4. Mai., 1997.
- [ALE98a] Alencar, F. M. R. e Castro, J. F.B.: "Formalização de Requisitos Organizacionais". Workshop Iberoamericana de Engenharia de Requisitos e Ambientes de Software - IDEAS'98. pp. 89-98. Torres-RS, Brasil. 01-03 de abril, 1998.
- [ALE98b] Alencar, F. M. R. e Castro, J. F.B.: "Guidelines from Organizational Requirements to Formal Specification". I Workshop em Engenharia de Requisitos – WER'98. pp. 9-17. Maringá, PR, Brasil. Outubro, 1998.
- [ALE99a] Alencar, F. M. R. e Castro, J. F.B.: "Integrating Early and Late-Phase Requirements: A Factory Case Study". XIII Simpósio Brasileiro de Engenharia de Software – SBES'99. To appear. Florianópolis, SC, Brasil. 13-15 de Outubro, 1999.
- [ALE99b] Alencar, F. M.R., Sousa, F.M. and Castro, J. F. B.: "Modelagem Organizacional: Análise Comparativa das Técnica i* e Bubenko". Workshop Iberoamericana de Engenharia de Requisitos e Ambientes de Software, IDEAS'99. San Jose, Costa Rica. Abril, 1999.

- [ATK91] Atkison, W. D., Booth, J. P. and Quirk, W. J.: "Validation Tools for Structured Mal". FOREST Research Project. NFR/WP5.2/HRL/R/002/A. Oct. 1991.
- [BAU85] Bauer, F. L. et al: " The Wide-Spectrum Language CIP-L". Springer LLLNLCS 183. 1985.
- [BEL77] Bell, T. E., Bixler, D. C. and Dyer, M. E.: "An extendable approach to computer-aided software requirements engineering ". IEEE Transactions on Software Engineering, SE-E, pp. 849-860. 1977.
- [BER94] Berry, Daniel: "The Requirement Iceberg and Various Icepicks Chipping at it". VIII Simpósio Brasileiro de Engenharia de Software, Curitiba - PR, 25-28 Out., 1994.
- [BOE89]Boehm, B. W. : "Software Risk Management". IEEE Computer Society Press. Washington, 1989.
- [BOIS95] Bois, Philippe Du: " The ALBERT II Language: On the Design and the Use of a Formal Specification Language for Requirements Analysis". Thesis. Facultés Universitaires Notre-Dame de la Paix Namur (Belgium), 1995.
- [BOIS97] BOIS, P. Du, Dubois, E. and Zeippen, J.M.: "On the Use of Formal RE Language: The Generalized Railroad Crossing Problem". In Proceedings of the 3rd IEEE International Symposium on Requirements Engineering. Annapolis, Maryland, USA. pp. 128-137. Jan 1997.
- [BOO95] Booch, G. and Rumbaugh, J.: "UML – Unified Method for Object-oriented Development". Conjunto de documentação, versão 0.8. Rational Software Corp. Santa Clara, California, 1995.
- [BOO99] Booch, G., Jacobson, I. and Rumbaugh, J.: "Unified Modeling Language User Guide". Rational Software Corporation. Addison-Wesley Object Technology Series. Jan., 1999.
- [BOW94a] Bowen, J. and Hinchey, G.: " Seven More Myths of Formal Methods: Dispelling Industrial Prejudices". In M. Naftalin, T. Denvir and M. Bertran (eds.), FME'94: Industrial Benefit of Formal Methods. Springer-Verlag, LNCS 873, PP. 105-117, 1994.
- [BOW94b] Bowen, J. and Hinchey, G.: " Formal Methods and Safety-Critical Standards". IEEE Computer, 27 (8), pp. 68-71. August 1994.

- [BOW95] Bowen, J. and Hinchey, G.: “Seven more Myths of Formal Methods”. IEEE Software, 12 (4): 34-41, Jul. 1995.
- [BRE98] Breitman, Karin K., Leite, Júlio C. S. P.: “A Framework for Scenario Evolution”. ICRE’98 Third International Conference on Requirements Engineering. Colorado Springs, Colorado, USA. 1 ed. USA: IEEE CSP, Los Alamitos, CA. Proceedings, pp. 214-221. Abril, 1998.
- [BUB83] Bubenko, Janis. A.: “On Concepts and Strategies for Requirements and Information Analysis”. In Information Modelling, pp. 125-169. Chartwell-Bratt, 1983.
- [BUB93b] Bubenko, J. A.: “Extending the Scope of Information Modeling”. In Proceedings of the 4th International Workshop on the Deductive Approach to Information Systems and Database. Lloret-Costa Brava, Catalonia. pp. 73-98. Sept.20-22, 1993.
- [BUB94] Bubenko, Janis A. and Kirikova, M.: “Enterprise Modelling: Improving the Quality of Requirements Specification”. IRIS-17 Information Systems Research Seminar in Scandinavia. Oulu, Finland, 1994.
- [BUB94a] Bubenko, J. A. and Kirikova, Marite.: ““Worlds” in Requirements Acquisition and Modelling”. Sweden. 1994.
- [BUB94b] Bubenko, J. A, Rolland, C., Loucopoulos, P. and de Antonellis, V.: “Facilitating ‘fuzzy to formal’ requirements modelling”. IEEE International Conference on Requirements Engineering. pp. 154-157. 1994.
- [BUB95a] Bubenko, J. A jr.: “Challenges in Requirements Engineering”. Second IEEE International Symposium on Requirements Engineering. IEEE Computer Society Press. York, England. pp. 160-162. March 27-29, 1995.
- [BUB95b] Bubenko, Janis A. and A. T. Berztiss: “A Software Process Model for Business Reengineering”. Sweden. 1995.
- [BUR80] Burstall, R.M. and Goguen, J. A.: “The Semantics of CLEAR, a Specification Language”. In Proceedings of the Advanced Course on Abstract Software Specifications - LNCS 86, Springer-Verlag, Copenhagen, pp. 292-332. 1980.

- [CAS91] Castro, J. F. B. and Finkelstein, A.: “VSCS: Na Object Oriented Method for Requirements Elicitation and Formalisation”. Imperial College, Department of Computing FOREST Research Project, Deliverable NFR/WP2.2/IC/R/002/A, 107 pag., Oct. 1991
- [CAS92] Castro, J. F. B. and Finkelstein, A.: “Requirements Elicitation and Formalisation”. In Proc. of XI Congress of the Brazilian Computing Society, XIX Seminar on Integrated Hardware and Software (SEMISH). pp. 214-228, Rio de Janeiro, Brasil, 1992.
- [CAS95]Castro, J. F.: “Introdução a Engenharia de Requisitos”. XV Congresso da Sociedade Brasileira de Computação, XIV Jornada de Atualização em Informática - JAI 95. Canela, RS. 1995.
- [CAS96]Castro, J. B., Toranzo, M. A. and Gautreau, C.: “Tool Support For Requirements Formalisation”. International Workshop on Multiple Perspective in Software Development. San Francisco, USA. pp. 202-206. Out., 1996.
- [CAS97]Castro, J. B. and Toranzo, M. A.: “Towards Software Quality: Multiview Case”. Third International Workshop on Requirements Engineering: Foundation for Software Quality – REFSQ’97. Barcelona, Espanha. pp. 107-118. June 16-17, 1997.
- [CHE89] Checkland, P. B.: “ Soft Systems Methodology”. In J. Rosehead (Ed.): Rational Analysis for a Problematic World, John Wiley & Sons: Chishestes. pp. 71-100, 1989.
- [CHEL80] Chellas, B. F.: “Modal Logic: na Introduction”. Cambridge University Press, 1fst. Ed., 1980.
- [CHU99] Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J.: “Non-Functional Requirements in Software Engineering”. 1999.
- [CIA98] Ciapessoni, E., Coen-Porisini, A., Crivelli, E., Mandrioli, D., Mirandola, P., Morzenti, A.: “From Formal Models to Formally-based Methods: an Industrial Experience”. April, 1998. To appear in ACM Transactions on Software Engineering and Methodologies.

- [COA91] Coad, P. and Yourdon, E.: "OOA - Object-Oriented Analysis". N. J. Prentice Hall. Englewood Cliffs, 1991.
- [COH86] Cohen, B., Harwood, W. T. and Jackson, M. I.: "The Specification of Complex Systems". Addison-Wesley Publishing Company, Inc.. 1986.
- [COO94] Cook, S. and Daniels, J.: "Designing Object Systems – Object Oriented Modelling with Syntropy". Prentice-Hall, 1994.
- [COR94] Cordeiro, V. A. O. , Sampaio, A. e Meira, S.: "From MooZ to Eiffel – A Rigorous Approach to System Development". In Formal Methods Europe: "Industrial Benefit of Formal Methods", pp. 306-325, 1994.
- [CYS97] Cysneiros, Luiz M. e Leite, J. C. S. P.: "Definindo Requisitos Não-Funcionais (in Portuguese)". XI Simpósio Brasileiro de Engenharia de Software, Fortaleza - CE - Brasil, pp. 49-64, 13-17 Out., 1997.
- [CYS98] Cysneiros, L. M. e Leite, J. C. S. P.: "Utilizando Requisitos Não Funcionais para Análise de Modelos Orientados a Dados". I Workshop em Engenharia de Requisitos – WER'98. Maringá, PR, Brasil. Outubro, 1998.
- [DAR93] Dardenne, A, van Lamsweerde, Fickas, S.: "Goal Directed Requirements Acquisition". Science of Computer Programming, vol. 20, pp. 3-50, Apr., 1993.
- [DARI97] Darimont, R. , Delor, E., Massonet, P. and Lamsweerde A. van: "GRAIL/KAOS: a Requirements Engineering Environment". International Conference on Software Engineering – ICSE'97. 1997.
- [DAV93] Davis, A M.: "Software Requirements: Objects, Functions, and States". P T R Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.
- [DEL90] Delisle, N. and Garfan, D.: "A Formal Specification of an Oscilloscope". IEEE Software, pp. 29-36. Set., 1990.
- [DeMA79] DeMarco: "Structured Analysis and Systems Specifications". Prentice Hall. Englewood Cliffs, 1979.
- [DOB89] Dobson, J. E. and McDermid, J. A.: "Security Models and Enterprise Models". In C. E. Landwehr (Ed.), Database Security: Status and Prospects II, pp. 1-39. Elsevier. 1989.

- [DOB90] Dobson, J. E. and McDermid, J. A.: “An Investigation into Modelling and Categorisation of Non-Functional Requirements”. Report No. YCS 141. University of York. 1990.
- [DOB92] Dobson, J. E.: “Information and Denial of Service”. In C. E. Landwehr and S. Jajodia (Eds.), Database Security: Status and Prospects V, pp. 21-46. Elsevier. 1992.
- [DOB94a] Dobson, J. E. and Strems, R.: “Organizational Requirements Definition for Information Technology”. IEEE International Conference on Requirements Engineering. Los Alamitos, California: IEEE Computer Society Press. pp. 158-165. April 1994.
- [DOB94b] Dobson, J. E., Blyth, A.J. C., Chudge, J. and Strems, R.: “The ORDIT approach to organisational requirements” . In M. Jirotko and J. Goguen (Eds.), Requirements Engineering, Academic Press Ltd. pp. 87-106. London, 1994.
- [DRU99] Drucker, Peter: “Os Novos Paradigmas da Administração”. Artigo no site da Informal Informática, <http://www.informal.com.br/artigos/a28091999001.html>. 1999.
- [DSOU99] D’Souza, D. F. and Will, A C.: “Objects, Components, and Frameworks with UML: The Catalysis Approach”. Addison Wesley Longman. 1999.
- [DUB95] Dubois, E., Wu, S. and Zeippen, J-M.: “Towards a Goal-oriented Security Framework for Elaborating Formal Cooperative System Requirements”. In Proc. of 2nd. International Workshop on Requirements Engineering: Foundation of Software Quality – CAISE’95. Jyväskylä, Finland. June 12-13th, 1995.
- [DUB98] Dubois, Eric, Yu, Eric and Petit, Michael: “From Early to Late Formal Requirements: a Process-Control Case Study”. Proc. 9th International Workshop on Software Specification and Design, IWSSD98. IEEE Computer Society, pp. 34-42. Ise-Shima, Japan. April, 1998.
- [EAS97] Eason, K., Harker, S. and Olphert, Wendy: “Working with Users to Generate Organisational Requirements: The ORDIT Methodology”. ICL Systems Journal. Vol. 11. Iss.2. Jan. 1997.
- [EDE94] Edelweiss, Nina e Oliveira, J. P. M.: “Modelagem de Aspectos Temporais de Sistemas de Informação”. IX Escola de Computação. Recife – PE. Julho, 24-31, 1994.

- [EHR85] Ehrig, H. and Amhr, B.: “Fundamentals of Algebraic Specifications 1”. EATCS Monographs on Computer Science, Springer. 1985.
- [EVA99] Evans, A. and Kent, S.: “Core Meta-Modelling Semantics of UML: The pUML Approach”. UML’99 – The Unified Modeling Language. In Proc. of the Second International Conference. Eds. Robert France and Bernhard Rumpe. Fort Collins, CO, USA. pp. 140-155. Out. 1999.
- [FIA92] Fiadeiro, J. L. and Maibaum, T.: “ Temporal Theories as Modularisation Units for Concurrent System Specification”. Formal Aspects of Computing, n. 4, pp. 239-272, 1992.
- [FIN92] Finkelstein, A., Kramer, J., Nuseibeh, B. and Goedicke, M.: “Viewpoints: A Framework for Integrating Multiple Perspectives in Systems Development”. International Journal of Software Engineering and Knowledge Engineering, vol 2, pp. 31-58. 1992.
- [FIN93] Finkelstein, A.: “Report of The Inquiry Into The London Ambulance Service”. International Workshop on Software Specification and Design Case Study. Feb., 1993. In the web <ftp://cs.ucl.ac.uk/acwf/info/lascase0.9.pdf>.
- [FOX97] Fox, M. S. and Gruninger, M.: “Enterprise Modelling”. Out. 1997.
- [FOX98] Fox, M. S., Barbuceanu, M., Beck, C., Davenport, A. and Gruninger, M.: “ Managing the Supply Chain An AI Perspective”. NSF Workshop on Supply Chain Management in Electronic Commerce. Oct., 1998. In the web site <http://www.ise.ufl.edu/Supplychain/done/Day2/MarkFox/tsld001.htm>.
- [FUR98] Furlan, J. D.: “Modelagem de Objetos através da UML: Análise e Desenho Orientados a Objeto”. Makron Books do Brasil. 1998.
- [GOG88] Goguen, J. et al.: “Introducing OBJ3”. SRI-CSL-88-9, SRI Intl, 1988.
- [GOL92]Goldsack, S. J., Booth, J., Ryan, M. D. and Quirk, W. J.: “The Structured MAL Specification Language”. FOREST Research Project, NFR/WP1.2/IC/002/A, Report R2 of activity WP1(Logic and Language). (Eds) S. J. Goldsack., Part 5, Feb. 1992.

- [GUS95] Gustas, R., Bubenko, J. and Wangler, B.: "Goal Driven Enterprise Modelling : Bridging Pragmatic and Semantic Descriptions of Information Systems". Swede. 1995.
- [GUT93] Guttag, J. V. and Horning, J. J.: "LARCH: Languages and Tools for Formal Specification", Springer-Verlag. 1993.
- [HAB99] Haberkorn, E.: "Teoria do ERP". Makron Books. Rio de Janeiro, 1999.
- [HAL90] Hall, A.: "Seven Myths of Formal Methods". IEEE Software, pp. 11-20. Set., 1990.
- [HAM98] Hamie, A., Civello, F., Howse, J., Kent, S. and Mitchell, R.: "Reflections on the Object Constraint Language". In Proceedings of UML'98 International Workshop. Eds. Pierre-Alain Muller and Jean Bézivin. pp. 137-145. Mullhouse, France. Jun., 1998.
- [HAM98a] Hamie, A., Howse, J. and Kent, S.: "Interpreting the Object Constraint Language". In Proc. of Asia Pacific Conference in Software Engineering, IEEE Press. Set. 1998.
- [HER99] Heroux, M.: "Enterprise Modeling Techniques". In the web site http://www.users.csbsju.edu/~mheroux/csci330/module_c/outlinee.htm. 1999.
- [HOA78] Hoare, C. A. R.: "Communicating Sequential Processes". Communications of the ACM, vol. 17, pp. 666-667. 1978.
- [HOF93] Hofman, Hubert F. : "Requirements Engineering: A Survey of Methods and Tools". Institut für Informatik der Universität Zürich, Nr. 93.05, März 93.
- [HOL93] Holanda, S. M.: "Sobre a Aplicação de Especificação Formal Orientada a Objetos à Semântica Formal de Linguagens Orientadas a Objetos". Dissertação de Mestrado. Departamento de Informática, Recife. Abr. 1993.
- [IBMOCL] IBM Application Development: Object Constraint Language (OCL). In web site http://www-4.ibm.com/software/ad/standards/ocl.html/ocl_info.htm.

- [IS88]International Standard – IS 8807. “Information Processing Systems - Open Systems Interconnection – LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour”. 1988.
- [JAC95a] Jackson, M.: “Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices”. 1ed. Addison-Wesley. Massachusetts, USA. 1995.
- [JAC95b] Jackson, M.: “Problems and Requirements”. Invited Speaker on the Second IEEE International Symposium on Requirements Engineering. IEEE Computer Society Press. York, England. pp. 2-8. March 27-29, 1995.
- [JAC98] Jackson, M.: “Formal Methods and Traditional Engineering”. Journal of Systems and Software special issue on Formal Methods Technology Transfer 40, 3. pp. 191-194. March, 1998.
- [JACO92]Jacobson, I., Christerson, M., Jonsson, P., and Gunnar, O. : “Object-oriented Software Engineering: A Use Case Driven Approach”. Addison-Wesley. 1992.
- [JACO99] Jacobson, I., Booch, G. and Rumbaugh, J.: “Unified Software Development Process”. Rational Software Corporation. Addison-Wesley Object Technology Series. Jan., 1999.
- [JAR92] Jarke, M., Mylopoulos, J., Schmidt, J.W., Vasiliou, Y.: “DAIDA: An Environment for Evolving Information Systems”. Vol. 10. Jan, 1992.
- [JIR94] Jirotko, M. and Goguen, J. A.: “Requirements Engineering: Social and Technical Issues”. Academic Press, San Diego, CA. 1994.
- [JIR98]Jirotko, M.: “Video-Supported Ethnography for Requirements Capture”. ICRE’98 Third International Conference on Requirements Engineering. Colorado Springs, Colorado, USA. 1 ed. USA: IEEE CSP, Los Alamitos, CA. Tutorial, Abril, 1998.
- [JON90] Jones, C. B.: “Systematic Software Development Using VDM”. Prentice Hall International, 2nd Edition. 1990.

- [JUR99] Jirisica, I., Mylopoulos, J. and Yu, E.: Using Ontologies for Knowledge Management: An Information Systems Perspective". Annual Conference of the American Society for Information Science. Washington, D.C., 1999.
- [KAI98] Kaindl, H., Krammer, S., Kacsich, R.: "A Case Study of Decomposing Functional Requirements Using Scenarios". ICRE'98 Third International Conference on Requirements Engineering. Colorado Springs, Colorado, USA. 1 ed. USA: IEEE CSP, Los Alamitos, CA. Proceedings, pp. 156-163. April, 1998.
- [KEL92] Kelly, J. C., Sherif, J. S. and Hops, J.: "An Analysis of Defect Densities Found During Software Inspections". Journal of Systems Software, vol. 17, pp. 111-117. 1992.
- [KEN91] Kent, S.: "A Deduction Calculus for Modal Action Logic with Action Combinators". FOREST Deliverable Report WP3.R2. 1991.
- [KEN92] Kent, S., Maibaum, T.S.E. and Quirk, W. J.: "Specifying Deontic Behaviour in Modal Action Logic". FOREST Deliverable Report WP1.R2. 1992.
- [KEN93] Kent, S., Maibaum, T. and Quirk, W.: "Formally Specifying Temporal Constraints and Error Recovery". Proceedings of IEEE International Symposium on Requirements Engineering - RE93, p.208-215, Jan. 1993.
- [KEN99] Kent, S., Gaito, S. and Ross, N.: "A meta-model semantics for structural constraints in UML: Behavioural Specifications for Business and Systems". Ed. Haim Kilov. Kluwer, 1999.
- [KIR94] Kirikova, M. and Bubenko, J. A. : "Software Requirements Acquisition through Enterprise Modelling". Stockholm University. Sweden. 1994.
- [KOT96] Kotonya, G. and Sommerville, I.: "Requirements Engineering with Viewpoints". Software Engineering, 1(11), pp. 5-18. 1996.
- [KOT98] Kotonya, G. and Sommerville, I.: "Requirements Engineering: Processes and Techniques". 1ed England: John Wiley & Sons Ltd. 1998.
- [KRU98] Kruchten, P.: "The Rational Unified Process: An Introduction". Rational Software Corporation. Addison-Wesley. 1998.

- [LAM95]Lamsweerde, A van, Darimont, R. and Massonet, P.: “Goal-Directed Elaboration of Requirements for a Meeting Scheduler Problems and Lessons Learnt”. Proc. 2nd IEEE Symposium on Requirements Engineering. pp. 194-203. York. March, 1995.
- [LAM98a] Lamsweerde, Axel van and Letier, E.: “Integrating Obstacles in Goal-Driven Requirements Engineering”. In Proc. of the 20th. International Conference on Software Engineering – ICSE’98. Kyoto. ACM-IEEE. April 1998.
- [LAM98b] Lamsweerde, A van, Darimont, R. and Letier, E.: “ Managing Conflicts in Goal-Driven Requirements Engineering”. IEEE Transactions on Software Engineering, Special Issue on Managing Inconsistency in Software Development. Nov. 1998.
- [LAW93]Chung, L.: “Dealing with Security Requirements During the Development of Information Systems”. In 5th. Int. Conference on Advanced Info. Systems Engineering – CaiSE’93. Paris, France. Jun 1993.
- [LAW95] Chung, L., Nixon, B.A. , Yu, E.: “Using Non-Functional Requirements to Systematically Support Change”. Second IEEE International Symposium on Requirements Engineering. IEEE Computer Society Press. York, England. pp. 132-139. March 27-29, 1995.
- [LEE91]Lee, J.: “Extending the Potts and Brauns model for recording design rationale”. In. Proc. 13th International Conference on Software Engineering. pp. 114-125, May, 1991.
- [LEI87] Leite, J. C. S. P.: “A Survey on Requirements Analysis”. Technical Report. Department of Information and Computer Science. California, 1987.
- [LEI95] Leite, J. C. S. P and Cerqueira, Paulo M.: “Recovering Business Rules from Structured Analysis Specification”. PUC–Rio. Brasil. 1995. IEEE.
- [LEI96]Leite, J. C.S.P., Gilvaz, Ana P.P.: “Requirements Elicitation Driven by Interviews: the use of viewpoint”. IWSSD’96 Eighth International Workshop on Software Specification Design. 1ed. USA: IEEE CSP, Los Alamitos, CA. Proceedings, pp. 87-94. Março, 1996.

- [LEI98]Leite, J.C.S.P., Leonardi, Maria C.: “Business Rules as Organizational Policies”. IWSSD’98 Ninth International Workshop on Software Specification Design. ISE-Shima, Japan. 1ed. USA: IEEE CSP, Los Alamitos, CA. Proceedings, pp. 68-76. Abril, 1998.
- [LEO98]Leonardi, M. C., Leite, J.C. S. P. e Rossi, G.: “Estrategias para la identificación de Reglas de Negocio (in Espanhol). XII Simpósio Brasileiro de Engenharia de Software, Maringá – PR - Brasil, pp. 53-67, 13-16 Out., 1998.
- [LEV98]Levson, Nancy G.: “Intent Specifications: An Approach to Building Human-Centered Specifications”. ICRE’98 Third International Conference on Requirements Engineering. Colorado Springs, Colorado, USA. 1ed. USA: IEEE CSP, Los Alamitos, CA. Proceedings, pp. 204-213. Abril, 1998.
- [LIM98] Lima, C. A. G., Reis, R. Q. and Nunes, D. J.: “Gerenciamento do Processo de Desenvolvimento Cooperativo de Software no Ambiente PROSOFT”. XII Simpósio Brasileiro de Engenharia de Software – SBES’98. UEM/Maringá-PR, Brasil. Out., 1998.
- [LOU95]Loucopoulos, P. e Karakostas, V.: "System Requirements Engineering". McGraw-Hill Book Company.1995.
- [LUT93] Lutz, R. R.: “Analyzing Software Requirements Erros in Safety-Critical, Embedded Systems”. In Proc. IEEE International Symposium on Requirements Engineering – RE’93. IEEE Computer Society Press. San Diego, California, USA. pp. 126-133. Jan. 4-6, 1993.
- [MAC96]Macaulay, Linda A.: “Requirements Engineering” 1ed. Great Britain: Springer-Verlag London Limited. 1996.
- [MAI87]Maibaum, T. S. E.: “ A Logic for the Formal Requirements Specification of Real-time Embedded Systems” . The FOREST Project (Alvey SE015), Deliverable R3. 1987.
- [MAL94] Malone, T. W. and Crowston, K.: “The Interdisciplinary Study of Coordination. Computing Surveys”. 26:87-119, Mar. 1994.
- [MAN86] Manes, E. G. and Arbib, M. A.: “Algebraic Approaches to Program Semantics”. Sprringer-Verlag, 1986.

- [McDE96] McDermid, J.A.: "Software Engineer's Reference Book". Butterworth-Heinemann (ed.). 1996.
- [MEI90] Meira, S.R. L. e Cavalcanti, A.L.C.: "Modular Object-Oriented Z Specifications". V Z User Meeting, Oxford, UK. Dec. 1990.
- [MEY88] Meyer, B.: "Object-Oriented Software Construction". Prentice Hall, 1988.
- [MIL98] Millard, N., Lynch, P., Tracey, K.: "Child's Play: using Techniques Developed to Elicit Requirements from Children with Adults". Colorado Springs, Colorado, USA. 1ed. USA: IEEE CSP, Los Alamitos, CA. Proceedings, pp. 66-73. Abril, 1998.
- [MILN80] Milner, R.: "A Calculus for Communicating Systems". Lecture Notes in Computer Science, pp. 92. Springer. 1980.
- [MIN93] Min, Lin Tse: "A Semântica Formal de MooZ". Dissertação de Mestrado. Universidade Federal de Pernambuco, Departamento de Informática, Recife (PE). Dez. 1993.
- [MUL79] Mullery, G. P.: "CORE: A Method for Controlled Requirements Expression". Proceedings of the Fourth International Conference on Software Engineering., IEEE Press. 1979.
- [MUN90] Mundy, P. and Wordsworth, J. B.: "The CICS application programming interface: Transient data and storage control". IBM Hursley Park Technical Report. TR 12.299. Oct., 1990.
- [MYL90] Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: "Telos: Representing Knowledge about Information Systems". ACM Trans. Information Systems, 8 (4), pp. 325-362. Outubro, 1990.
- [MYL92] Mylopoulos, J., Chung, L. and Nixon, B.: "Representing and Using Non-Functional Requirements: A Process Oriented Approach". IEEE Transaction on Software Engineering, vol. SE-18, nº. 6, pp. 483-497. June, 1992.
- [MYL95] Mylopoulos, J.: "Capturing Intentions in Requirements Engineering: A modeling Perspective". Invited Talk at the IFIP W2.9 meeting, Bramshill, UK, March, 1995.

- [MYL99] Mylopoulos, J., Chung, L. and Yu, E.: "From Object-Oriented to Goal-Oriented: Requirements Analysis". Communications of the ACM, vol. 42, n. 1. Jan. 1999.
- [NEL94] Nellborn, C. and Holm, P.: "Capturing Information Systems through Enterprise and Speech Act Modelling". Sweden. 1994.
- [NIC96] Nicolao, M. e Oliveira, J. P. M. de: "Caracterizando Sistemas de Workflow". Revista REAd, ISSN 1413-2311, Programa de Pós-graduação em Administração – UFRGS, n.3, Out. 1996.
- [NIX93] Nixon, B. A.: "Dealing with Performance Requirements During the Development of Information Systems". In Proc. International Symposium on Requirements Engineering, IEEE Computer Society Press. San Diego, CA. pp. 42-49. Jan. 4-6, 1993.
- [OME3] Yu, Eric.: OME framework, version 3, 1999. Web site <ftp://ftp.cs.toronto.edu/pub/eric/ome3>
- [ORD93]ORDIT Consortium: "ORDIT Process Manual". HUSAT Research Institute, Loughborough University, Loughborough, UK. 1993.
- [PAI99] Paige, R. F. and Ostroff, J. S.: "A Comparison of the Business Object Notation and the Unified Modeling Language". Technical Report CS-1999-03, 1999.
- [PET90] Petit, M. : "The Agent-oriented Language for Building and Eliciting Requirements for real-Time systems – ALBERT". PhD Thesis, University of Namur, 1990.
- [PET97]Petit, M. and Dubois, E.: "Defining an Ontology for the Formal Requirements Engineering of Manufacturing Systems". In Proc. of the International Conference on Enterprise Integration and Modelling Technology - ICEIMT'97. Turin, Italy. October, 1997.
- [pG99] Grupo do Precise UML, pUML: <http://www.cs.york.ac.uk/puml>, 1999.
- [PIN96] Pinheiro, A. C. and Goguen, J. A.: "An Object-Oriented Tool for Tracing Requirements". In Proc. of the 2nd IEEE International Conference on Requirements Engineering - ICRE'96. Colorado Springs, Colorado, USA. April, 1996.

- [PIN98] Pinheiro, A. C.: “ Da Importância dos Modelos de Desenvolvimento de Software para a Engenharia de Requisitos”. WER98 – Workshop on Requirements Engineering, event of XII SBES – Simpósio Brasileiro de Engenharia de Software. Maringá-PR, Brasil, pp. 237-250. Out. 1998.
- [PIN98b] Pinheiro, A. C.: “An Object-Oriented Library for Tracing Requirements”. WER99 – II Workshop on Requirements Engineering, event of XXVIII JAIIO Jornadas Argentinas de Informática e Investigación Operativa. Buenos Aires, Argentina. pp. 187-197. Set., 1999.
- [PNU77] Pnueli, A.: “ The Temporal Logic of Programs”. In Proc. 18th Ann. Symp. on Foundations of Computer Science, pp. 46-57. 1977.
- [POR99] Porter, Michael E.: “ Vantagem Competitiva: Criando e Sustentando um Desempenho Superior”. Trad. Elizabeth M. P. Braga. Editora Campus. Rio de Janeiro. Março, 1999.
- [PRE96] Pressman, Roger S.: "Software Engineering: A Practitioner's Approach". McGraw-Hill Book Company. 4th. Edição, 1996.
- [REI97] Reis, R. Q., Lima, C. A. G. and Nunes, D. J.: “Cooperative Software Development and the PROSOFT Environment”. XII International Symposium on Computer and Information Sciences. Bogazici University, Antalya, Turquia. 1997.
- [REI98] Reis, R. Q., Lima, C. A. G. and Nunes, D. J.: “A Synchronous Cooperative Architecture for the PROSOFT Software Engineering Environment”. IV Congresso Argentino de Ciencias de la Computacion – CACIC’98. Neuquén, Argentina. Out., 1998.
- [REM95] Remigijus Gustas.: “A Basis for Integration within Enterprise Modelling”. Stockholm University. Sweden. 1995.
- [RIC98] Richters, M. and Gogolla, M.: “On Formalizing the UML Object Constraint Language OCL”. In Tok Wang Ling, Sudha Ram, and Mong Li Lee, editors, Pro. 17th Int. Conf. Conceptual Modeling (ER’98), pp. 449-464. Springer, Berlin, LNCS 1507, 1998.

- [ROS77] Ross, D.: “Structured Analysis (AS): A Language for Communicating Ideas”. IEEE Transactions on Software Engineering. 3,1. pp. 16-34. Jan. 1977.
- [RUM91] Rumbaugh, J. et. al.: “Oriented Modeling Technique”. 1991.
- [RUM99] Rumbaugh, J., Jacobson, I. and Booch, G.: “Unified Modeling Language Reference Guide”. Rational Software Corporation. Addison-Wesley Object Technology Series. Jan., 1999.
- [RYAN90] Ryan, M.: “ Development of MAL in FOREST Research”. Imperial College, Department of Computing, FOREST Research Project, Deliverable NFR/WP1.1/IC/R/001/A. Oct. 1990.
- [RYAN91] Ryan, M., Fiadeiro, J. L. e Maibaum, T. S. E.: “Sharing Actions on Attributes in Modal Action Logic”. In Proc. of the International Conference on Theoretical Aspects of Computer Science (TACS’91). Ito T. and Meyer A.(eds.). Springer-Verlag, 1991.
- [RYAN93] Ryan, M.: “Defaults in Specifications” . In Proc. IEEE International Symposium on Requirements Engineering – RE’93. IEEE Computer Society Press. San Diego, California, USA. pp. 142-149. Jan. 4-6, 1993.
- [SAN85] Sannella, D.T. and Tarlecki, A.: “Program Specification and Development in Standard ML”. In Proc. of the 12th ACM Symposium on Principles of Programming Languages. New Orleans. pp. 67-77. 1985.
- [SAN93] Sannella, D. T.: “A survey of formal software development methods”. In Richard Thayer and Andrew McGettrick, editors, Software Engineering: A European Perspective, pages 281-297. IEEE Computer Society Press, 1993 (Originally published as report ECS-LFCS-88-56, Dept. of Computer Science, Univ. of Edinburgh, 1988).
- [SAP] SAP do Brasil. Web Site: <http://www.sap.com/brazil>.
- [SCH97] Schlabre, H. and Nunes, D. J.: “PROSOFT: An Object-based Distributed Software Development Environment”. Technical Report. 1997.
- [SHL92] Shlaer, S. and Mellor, S.: “Object Lifecycles: Modeling the World in States”. N. J. Prentice Hall. Englewood Cliffs, 1992.

- [SOM97] Sommerville, Ian e Sawyer, P.: "Requirements Engineering: A Good Practice Guide". John Wiley & Sons, 1997.
- [SOU98] Sousa, F. M. e Castro, J.F. B.: "Modelagem Organizacional : Um Estudo de Caso no Comércio". WER'98 – I Workshop de Engenharia de Requisitos. Maringá, Paraná. Brasil, Outubro 1998
- [SPI89] Spivey, J. M.: "The Z Notation". Prentice Hall. 1989.
- [STD830] IEEE Std. 830. "IEEE Guide to Software Requirements Specification". The Institute of Electrical and Electronics Engineers. New York , 1984.
- [STD830-98] IEEE Std. 830-98. "Recommended Practice for Software Requirements Specification". The Institute of Electrical and Electronics Engineers. New York , 1990.
- [TAU99] Taurion, C.: "Como Alinhar Tecnologia e Negócio". Opinião. Revista Coputerworld, Edição 305. 13 Set 1999.
- [TEI99] Teixeira, J.: "Repensando a Gestão e a Tecnologia". Artigo no site da Informal Informática, <http://www.informal.com.br/artigos/a20099004.html>. 1999.
- [TOR99] Toranzo, M. A.C. and Castro, J. F. B.: "A Comprehensive Traceability Model to Support the Design of Interactive Systems". Presented at the International Workshop on Interactive System Development and ObjectModels - WISDOM 99. 9 pages. Lisboa, June 1999.
- [UML99] OMG. UML Semantics. In OMG Unified Modeling Language Specification, Version 1.3, Cap. 2, Jun. 1999.
- [VAS95] Vasconcelos, A. M. L.: "ForMooZ: Um Ambiente Baseado em Hipertexto para Desenvolvimento de Especificações Formais". Caderno de Ferramentas dos Anais do IX Simpósio Brasileiro de Engenharia de Software. Recife-PE, Brasil. Out. 1995.
- [WAL95] Walden, K. and Nerson, J.M.: "Seamless Object-Oriented Software Architecture: Analysis and Design of Reliable Systems". Prentice Hall. 1995.
- [WAR85] Ward, P. and Mellor, S.: "Structured Development for Real-Time Systems". Prentice Hall. Englewood Cliffs, N.J., 1985.
- [WAR86] Ward, P. and Mellor, S.: "The Transformation Schema: an Extension of Data Flow Diagram to Represent Control and Timing". IEEE Transactions on Software Engineering 12, 2. pp. 198-210. Feb. 1986.

- [WARM99] Warmer, Jos B. and Kleppe, Anneke G.: "The Object Constraint Language: Precise Modeling with UML". Addison-Wesley. USA, 1999.
- [WEI98] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P.: "Scenario Usage in System Development: A Report on Current Practice". Colorado Springs, Colorado, USA. 1ed. USA: IEEE CSP, Los Alamitos, CA. Proceedings. Abril, 1998. "Scenarios in System Development: Current Practice". 1ed. USA: IEEE Software, pp. 26-29. Março/Abril 1998.
- [WHI94] White, I.: "Using the Booch method: A rational approach". Benjamin/Cummings Publishing Company. USA, 1994.
- [WIE98] Wieringa, Roel, Dubois, E. and Huyts, S.: "Integrating Semi-Formal and Formal Requirements". In Proceedings of the 7th Int. Conference on Advanced Info. Systems Engineering – CaiSE'97, Barcelona, Spain, pp. 19-32. 1997.
- [WIR86] Wirsing, M.: "Structured Algebraic Specifications: A Kernel Language". Theoretical Computer Science, 42, pp. 124-249. 1986.
- [WOO96] Woodcok, Jim and Davies, Jim: " Using Z Specification, Refinement, and Proof". Prentice Hall Europe, 1996.
- [YU93] Yu, Eric.: "Modelling Organizations for Information Systems Requirements Engineering". IEEE International Symposium on Requirements Engineering. IEEE Computer Society Press, San Diego, CA. pp. 34-41. 1993.
- [YU94] Yu, E. and Mylopoulos, J.: "Understanding 'Why' in Software Process Modelling, Analysis and Design". Sixteenth International Conference on Software Engineering. Sorrento. 1994.
- [YU95a] Yu, Eric S. K.: "Modelling Strategic Relationship for Process Reengineering". PhD thesis, Computer Science Department, University of Toronto, Toronto (Canada), 1995.
- [YU95b] Yu, E., Bois, P. and Mylopoulos, J.: "From Organization Models to System Requirements - A Cooperating Agents Approach". The 3rd International Conference on Cooperative Information Systems - CoopIS-95, Vienna (Austria), May, 2-9, 1995.
- [YU96] Yu, E., Mylopoulos, J. and Lesperance, Y.: " AI Models for Business Process Reengineering". IEEE Expert, pp. 16-23. August, 1996.

- [YU97a] E. S. K. Yu: "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering". Proceedings of the 3rd IEEE International Symposium on Requirements Engineering - RE97, p.226-235, Jan. 1997.
- [YU97b] Yu, Eric and Mylopoulos, J.: "Modelling Organizational Issues for Enterprise Integration". ICEIMT'97 - International Conference on Enterprise Integration and Modelling Technology. Turin, Italy. October, 1997.
- [YU97c] Yu, Eric.: "Why Agent-Oriented Requirements Engineering". Proceedings of 3rd International Workshop on Requirements Engineering: Foundations for Software Quality. Barcelona, Catalonia. June 16-17, 1997.
- [YU98] Yu, E. and Mylopoulos, J. "Why Goal-Oriented Requirements Engineering". Proceedings of the 4th International Workshop on Requirements Engineering? Foundations of Software Quality. pp. 15-22. Pisa, Italy. 8-9 June, 1998.
- [YU99] Yu, E.: "Strategic Modelling for Enterprise Integration". In Proc of the 14th World Congress of International Federation of Automatic Control – IFAC99. Beijing, China. July 1999.
- [ZAV97] Zave, Pamela, Jackson, Michael.: "Four dark corners of requirements engineering". 1ed. USA: ACM, Inc – Association for Computing Proceedings, vol. 6, n. 1, pp. 1-30. Janeiro, 1997.