



Universidade Federal de Pernambuco

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Andréa Pinto Castor

Rastreamento de Requisitos no processo de desenvolvimento de software orientado a agentes

*Dissertação apresentada à Coordenação
da Pós-Graduação em Ciência da
Computação do Centro de Informática,
como parte dos requisitos para obtenção
do título de Mestre em Ciência da
Computação.*

ORIENTADOR: Jaelson Freire Brelaz de Castro

Recife, agosto de 2004.

Agradecimentos

Agradeço ao meu marido Eduardo, por ter me apoiado incondicionalmente, pelo seu incentivo e por me fazer acreditar que eu podia dar mais esse passo na minha vida. Agradeço pela sua dedicação a nossa família, por ser um pai tão presente e por tantas vezes ter sido também mãe, suprimindo minha ausência perante nossos filhos.

Agradeço ao meu professor e orientador Jaelson por sempre acreditar no meu trabalho apesar de todas as adversidades, pelo seu total apoio e pelo conhecimento adquirido com esse trabalho.

Agradeço especialmente a Rosa Cândida, minha amiga, pelas suas revisões, sugestões, paciência e disponibilidade. Obrigada amiga, sem o seu apoio não teria finalizado esse trabalho.

Agradeço a todos os amigos que me incentivaram com palavras de ânimo ou que, de alguma forma, contribuíram com a execução deste trabalho, principalmente a Carla Silva, Claudianne Rabelo e Cibele Ataíde.

Resumo

O desenvolvimento orientado a agentes é recente, no entanto, este novo paradigma de desenvolvimento de software tem sido utilizado cada vez mais em aplicações industriais, tais como telecomunicações e comércio eletrônico. Entre as principais preocupações para a consolidação desse novo paradigma, destacamos a necessidade de metodologias, técnicas, notações e ferramentas adequadas para suportar o desenvolvimento de sistemas orientados a agentes. Neste sentido, o projeto Tropos está desenvolvendo uma abordagem de desenvolvimento orientado a agentes centrada em requisitos que visa suportar todas as fases do desenvolvimento de software orientado a agentes.

A metodologia Tropos define cinco fases de desenvolvimento de software: Requisitos Iniciais, Requisitos Finais, Projeto Arquitetural, Projeto Detalhado e Implementação. No entanto, Tropos não define atividades relacionadas ao *rastreamento dos requisitos* necessárias para identificar, por exemplo, a origem destes no sistema, quem os identificou e porquê eles foram incluídos no documento de requisitos. A atual versão de Tropos também não identifica como os artefatos gerados em cada uma das fases da metodologia estão relacionados com os requisitos.

Uma extensão da metodologia Tropos para tratar de forma adequada a questão do rastreamento de requisitos proporcionará muitos benefícios, pois não apenas permitirá um melhor atendimento aos requisitos do cliente, como também apoiará a avaliação do impacto de mudanças solicitadas (mesmo antes das suas implementações no sistema). Como resultado haverá uma melhoria na qualidade do software em desenvolvimento em decorrência da melhoria do processo de desenvolvimento o que levará a redução de tempo e custo da manutenção do software.

Diante do exposto, propomos nessa dissertação, a utilização de um processo para elaboração de um modelo de rastreamento de requisitos a fim de incluir as atividades de rastreamento de requisitos na metodologia Tropos. A proposta será validada através da elaboração de um estudo de caso que utiliza a metodologia Tropos para o desenvolvimento de um sistema de software multi-agentes para uma aplicação de comércio eletrônico.

Palavras-chave: Engenharia de Requisitos, Orientação a agentes, Rastreamento de Requisitos.

Abstract

The agent-oriented development paradigm is recent, however, this new software development paradigm has been successfully used in industrial applications, such as telecommunications and e-commerce. Among the several concerns required for the consolidation of this new paradigm, we highlight the need for suitable methodologies, techniques, notations and tools to support the agent-oriented software development. In this sense, the Tropos Project is developing an approach for agent-oriented development centered into requirements aiming to support all phases of agent-oriented software development.

The Tropos methodology defines five software development phases: Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation. However, Tropos doesn't define activities related to the requirements traceability. These activities are necessary to identify, e.g. the requirements sources, who has identified its and why it was included in the requirements document. The actual Tropos version does not identify how the generated artifacts, in each one of the methodology phases, are related to the requirements.

The Tropos methodology extent to deal adequately with the requirements traceability will bring many benefits, because it will either help attend the client requirements or give support to the change impact evaluations requested (even before implement it on the system to be). The results will be the quality software improvement as a result of the software process improvement and the narrowing of the software manutenability time and cost.

Considering this, we propose in this dissertation the use a process elaboration for a requirements traceability model to include the requirements traceability activities on the Tropos methodology. This proposal will be validated through a case study elaboration that uses the Tropos methodology to develop a multi-agent software system to a e-commerce application.

Key words: Requirements Engineering, Object-Oriented, Requirements Traceability.

Conteúdo

Capítulo 1 – Introdução	1
1.1 Motivação	2
1.2 Estágio Atual da Engenharia de Software Orientada a Agentes	3
1.3 Escopo da Dissertação	4
1.4 Objetivos e Abordagem	5
1.5 Contribuição	5
1.6 Estrutura da Dissertação	6
Capítulo 2 – Engenharia de Requisitos	8
2.1 Visão Geral da Engenharia de Requisitos	9
2.2 Motivação para o estudo da Engenharia de Requisitos	11
2.3 Classificação de Requisitos	13
2.4 Processo de Engenharia de Requisitos	14
2.5 Considerações Finais	25
Capítulo 3 – Rastreamento de Requisitos	26
3.1 Importância do Rastreamento de Requisitos	28
3.2 Tipos de Rastreamento	31
3.3 Obstáculos Encontrados no Rastreamento de Requisitos	33
3.4 Revisão de Trabalhos de Pré-Especificação de Requisitos	34
3.5 Revisão do Trabalho de Pós-Especificação de Requisitos	38
3.6 Revisão do Trabalho de Toranzo	46
3.7 Considerações Finais	49
Capítulo 4 – Engenharia de Software Orientada a Agentes	51
4.1 Introdução	52
4.2 Características de agentes	53
4.3 Tipos de Sistemas Baseados em Agentes	54

4.4	Importância dos Sistemas Multi-Agentes na Engenharia de Software	55
4.5	Áreas de Aplicação.....	57
4.6	Desafios e Obstáculos do Desenvolvimento Orientado a Agentes	59
4.7	Tropos – Uma Metodologia Para o Desenvolvimento de Software Orientado a Agentes	63
4.8	Considerações Finais	68
Capítulo 5 – Incorporando Rastreamento de Requisitos na metodologia Tropos		70
5.1	Introdução.....	71
5.2	Modelo Intermediário de Rastreamento de Requisitos	73
5.3	Diretrizes para Instanciação do Modelo de Referência.....	79
5.4	Requisitos Iniciais	82
5.5	Requisitos Finais	86
5.6	Projeto Arquitetural.....	88
5.7	Projeto Detalhado	91
5.8	Implementação	92
5.9	Aplicação das diretrizes 13 a 16.....	92
5.10	Matrizes de Relacionamento Sugeridas Para a Metodologia Tropos.....	96
5.11	Considerações Finais.....	99
Capítulo 6 – Estudo de Caso		101
6.1	Introdução.....	102
6.2	O Sistema de Comércio Eletrônico <i>Medi@</i>	102
6.3	Requisitos Iniciais: Aplicando diretrizes 1 a 12.....	103
6.4	Requisitos Finais: Aplicando diretrizes 1 a 12.....	106
6.5	Projeto Arquitetural: Aplicando diretrizes 1 a 12	112
6.6	Considerações Finais.....	116
Capítulo 7 – Conclusões e Trabalhos Futuros.....		117
7.1	Contribuições do Trabalho	118
7.2	Trabalhos Futuros.....	119
Referências Bibliográficas		121

Índice de Figuras

Figura 2-1 Atividades do Processo da Engenharia de Requisitos no Modelo Espiral [KOT97]	16
Figura 3-1 Rastreamento para frente e para trás.....	31
Figura 3-2 Rastreamento Pré Especificação e Pós Especificação de Requisitos	32
Figura 3-3 Diagrama simplificado dos dois tipos básicos de rastreamento de requisitos: PRE-RS e POS-RS	36
Figura 3-4 Meta modelo de rastreamento de requisitos proposto por Ramesh.....	39
Figura 3-5 Sub-modelo de Gerenciamento de Requisitos (Ramesh).....	42
Figura 3-6 Sub-modelo de Raciocínio (Ramesh).....	43
Figura 3-7 Sub-modelo de Projeto (Ramesh).....	44
Figura 3-8 Sub-modelo de Verificação de Conformidade	45
Figura 3-9 Meta-modelo proposto por Toranzo	47
Figura-3-10 Modelo de rastreamento intermediário proposto por Toranzo.....	48
Figura 4-1 Divisão do Processo de Desenvolvimento de Software em Etapas	65
Figura 5-1 Classificação das informações do rastreamento - Segmentos de informação	71
Figura 5-2 Sub-modelo de Gerência de Requisitos.....	75
Figura 5-3 Sub-modelo de Projeto	76
Figura 5-4 Modelo de Raciocínio.....	78
Figura 5-5 Visão geral do processo para elaborar um modelo de rastreamento	80
Figura 5-6 Modelo i* para gerenciamento de conferência.....	83
Figura 5-7 Rastreamento vertical	93
Figura 5-8 Rastreamento horizontal	95
Figura 6-1 Modelo i* de Dependência Estratégica da Loja de Mídia.....	103
Figura 6-2 Modelo i* de Razão Estratégica da Loja de Mídia.....	104
Figura 6-3 Modelo i* de Razão Estratégica do sistema Medi@	108
Figura 6-4 Grafo de Interdependência de Metas-soft.....	112

Índice de Tabelas

Tabela 2-1- Entradas e Saídas do Processo de Engenharia de Requisitos	15
Tabela 5-1 Tipos de Relacionamentos utilizados no modelo intermediário de rastreamento de requisitos	74
Tabela 5-2 Diretrizes Propostas por Toranzo para Instanciar o Modelo de Referência.....	81
Tabela 5-3 Abreviaturas para as classes de informação.....	82
Tabela 5-4 Requisitos Iniciais: Aplicação das diretrizes para identificação das classes dos segmentos de informação	85
Tabela 5-5 Requisitos Iniciais: Resultado da aplicação das diretrizes 9 a 12	86
Tabela 5-6 Requisitos Finais: Aplicação das diretrizes para identificação das classes dos segmentos de informação	87
Tabela 5-7 Requisitos Finais: Resultado da aplicação das diretrizes 9 a 12	88
Tabela 5-8 Catálogo de correlação	90
Tabela 5-9 Mapeamento para o modelo de raciocínio	91
Tabela 5-10 Projeto Arquitetural: Aplicação das diretrizes para identificação das classes dos segmentos de informação	91
Tabela 5-11 Projeto Arquitetural: Resultado da aplicação das diretrizes 9 a 12.....	91
Tabela 5-12 Projeto Detalhado: Aplicação das diretrizes para identificação das classes dos segmentos de informação	92
Tabela 5-13 Implementação: Aplicação das diretrizes para identificação das classes dos segmentos de informação	92
Tabela 5-14 Matriz: Informação Externa x Informação Organizacional	97
Tabela 5-15 Matriz: Informação Organizacional x Requisito.....	97
Tabela 5-16 Matriz: Informação Organizacional x Objetivos do Sistema.....	97
Tabela 5-17 Matriz: Posição x Argumentos.....	98
Tabela 5-18 Assunto x Posição	98
Tabela 5-19 Matriz: Requisito x Diagrama.....	98
Tabela 5-20 Matriz: Informação Organizacional x Programa.....	99
Tabela 5-21 Matriz: Requisito x Programa	99
Tabela 6-1 Classes do modelo de rastreamento I.....	106
Tabela 6-2 Informações Organizacionais - Classes do modelo de rastreamento II	106
Tabela 6-3 Requisitos - Classes do modelo de rastreamento III	109
Tabela 6-4 Classes do modelo de rastreamento IV	110
Tabela 6-5 Matriz de rastreabilidade: Informação Organizacional x Requisito funcional	111
Tabela 6-6 Requisitos não-funcionais - Classes do Modelo de Rastreamento V	113
Tabela 6-7 Posições - Classes do Modelo de Rastreamento VI.....	114
Tabela 6-8 Mapeamento dos links do framework para a matriz de rastreabilidade.....	114
Tabela 6-9 Matriz de rastreabilidade: Argumentos (Requisitos Não-Funcionais) x Posições.....	116

Capítulo 1 – Introdução

Este capítulo apresenta as principais motivações para realização desta dissertação. Em seguida é definido o escopo da dissertação, os objetivos e a abordagem utilizada, bem como as principais contribuições. Finalmente a estrutura do trabalho é apresentada.

1.1 Motivação

O crescimento da Engenharia de Software nas últimas duas décadas vem sendo realizado através do desenvolvimento de conceitos cada vez mais poderosos que incluem abstração de procedimentos e tipos de dados, bem como, mais recentemente, objetos e componentes. A noção de agentes representa um avanço similar em abstração: eles vêm sendo utilizados pelos desenvolvedores de software como uma ferramenta para compreender, modelar, e desenvolver software que opera em um ambiente dinâmico, aberto e freqüentemente imprevisível [WOO01a].

O crescimento recente no interesse em agentes de software tem levado ao desenvolvimento de novas metodologias baseadas nos conceitos de agente. Metodologias - como Gaia [WOO00], MaSE [WOD00], AUML [ODE00b], e Tropos [CAS02a] entre outras – têm se tornado os pontos chave de pesquisa na área emergente de Engenharia de Software orientada a agentes.

A Engenharia de Requisitos vem sendo reconhecida como uma área de extrema relevância para o processo da Engenharia de Software. A falta de atenção com os requisitos do sistema leva a sistemas que não atendem às expectativas dos *stakeholders*. A captura de informações necessárias para compreender a evolução, validação e verificação dos requisitos pode facilitar o gerenciamento de requisitos. No entanto, poucas metodologias orientadas a agentes são orientadas a requisitos [MYL00].

Os principais objetivos do processo de Engenharia de Requisitos são a compreensão dos objetivos dos *stakeholders*, refinamento desses objetivos em requisitos, tratamento dos conflitos entre os requisitos visando uma especificação concisa e clara desses requisitos de maneira a atender às expectativas dos *stakeholders* e servir como base para o projeto e implementação do sistema. No entanto, uma tarefa importante nesse processo é o acompanhamento bi-direcional dos relacionamentos entre os requisitos e o processo de desenvolvimento de artefatos a fim de facilitar a manutenção e verificação do sistema [RAM98a].

Requisitos devem ser levantados pelos *stakeholders* levando-se em consideração o escopo do sistema, complexidade e restrições. Um requisito é rastreável se podemos descobrir sua origem, porque ele existe, com quais requisitos ele se relaciona, e como o requisito se relaciona com o projeto do sistema, implementação, testes e documentação do usuário [PIN03].

Durante a fase de manutenção de um sistema, o rastreamento permite aos desenvolvedores e mantenedores preverem as alterações que deverão ocorrer no sistema quando de uma solicitação de mudança antes desta ser implementada. Rastreamento é útil ao relacionar a fase de projeto a

justificativas, decisões importantes, como também ao contexto no qual as soluções de projeto foram dadas. Graças ao rastreamento podemos melhor definir custos e cronogramas de mudanças, até mesmo se a equipe original não estiver mais disponível.

Um modelo de referência faz-se necessário para facilitar a construção de modelos de rastreamento de requisitos no processo de desenvolvimento de software, tornando o rastreamento de requisitos mais efetivo e conseqüentemente agregando maior valor ao projeto [TOR02a].

Nessa dissertação propomos a utilização do processo de elaboração de um modelo de rastreamento de requisitos proposto por Toranzo [TOR02a] para a inclusão das atividades de rastreamento de requisitos na metodologia Tropos [CAS02a]. A proposta será validada através da elaboração de um estudo de caso que utiliza a metodologia Tropos para o desenvolvimento de um sistema de software multi-agentes para uma aplicação de comércio eletrônico.

1.2 Estágio Atual da Engenharia de Software Orientada a Agentes

Visando demonstrar a eficácia da abordagem orientada a agente, o ideal seria demonstrar quantitativamente que a adoção desse paradigma ajuda a melhorar, de acordo com algumas métricas de software, o processo de desenvolvimento de software. Entretanto, devido à imaturidade da área poucos desses dados estão disponíveis [CAS02b]. Mesmo assim, há argumentos qualitativos que nos fazem acreditar que uma abordagem orientada a agentes trará benefícios para a engenharia de certos sistemas de software complexos. Esses argumentos surgiram de uma década de experiência no uso da tecnologia de agente para construir aplicações do mundo-real em larga-escala e em uma grande variedade de domínios industriais e comerciais [AGE03]. Note que não estamos sugerindo que o desenvolvimento baseado em agentes é uma solução mágica (*Silver Bullet*), pois não há nenhuma evidência que venha a sugerir que eles representarão uma melhoria em ordem de magnitude na produtividade da engenharia de software. Entretanto, acredita-se que para certas classes de aplicação, uma abordagem orientada a agentes pode melhorar significativamente o processo de desenvolvimento de software [JEN03a].

No momento, existem algumas barreiras para que haja uma maior aceitação da tecnologia de agente. É necessário estabelecer melhor a relação da tecnologia de agente com a tecnologia antecedente (tecnologia orientada a objetos), apresentando essa nova tecnologia como uma extensão incremental de métodos conhecidos e confiáveis [ODE00b]. É também necessário compreender as situações em que soluções baseadas em agentes são apropriadas [JEN03a]. Metodologias sistemáticas que possibilitem aos projetistas especificar e estruturar claramente suas aplicações, de acordo com o paradigma orientado a agentes, ainda estão em evolução, como Gaia [WOO00],

MaSE [WOD00], AUML [ODE00b], e Tropos [CAS02a]. Ferramentas para suportar o desenvolvimento de sistemas orientados a agentes encontram-se em desenvolvimento e poucas estão disponíveis no mercado ([FIP], [FIP02], [BEL03], [MOR04], [JAC]) . De fato, diversas metodologias e ferramentas de software estão começando a ser propostas [WOO02]. Para ajudar a difundir sistemas orientados a agentes, entre os desafios atuais, nesta dissertação abordamos a necessidade de inclusão de atividades de rastreamento de requisitos no contexto de metodologias orientadas a agentes.

Até agora, técnicas de desenvolvimento de software têm sido direcionadas à fase de implementação, no sentido de que o paradigma de programação dita as técnicas utilizadas no projeto e na análise de requisitos [ODE00b]. No paradigma orientado a agentes, as metodologias para implementação estão sendo desenvolvidas em paralelo. Surgem as primeiras metodologias voltadas especificamente para o desenvolvimento orientado a agentes ([WOO00], [ODE00b]). Poucas dessas metodologias são orientadas a requisitos, isto é, são desenvolvidas levando em consideração os conceitos usados durante as fases iniciais do processo de desenvolvimento de software, como Elicitação de Requisitos, Análise e Negociação de Requisitos, Documentação de Requisitos, Verificação e Validação de Requisitos e Gerenciamento de Requisitos. Nessa dissertação propomos a expansão da metodologia de desenvolvimento de software orientada a agentes e centrada em requisitos, Tropos [CAS02a], através da inclusão de atividades de rastreamento de requisitos.

1.3 Escopo da Dissertação

Essa dissertação tem o foco na introdução das atividades de rastreamento de requisitos em metodologias orientadas a agentes. Para isso utilizamos o modelo de referência para rastreamento de requisitos proposto por Toranzo [TOR02a].

Estenderemos a metodologia Tropos, que é baseada em conceitos utilizados para modelagem de requisitos. De fato Tropos suporta cinco fases de desenvolvimento: Requisitos Iniciais (*Early Requirements*), Requisitos Finais (*Late Requirements*), Projeto Arquitetural, Projeto Detalhado e Implementação [CAS02a].

Sabemos que o acompanhamento bi-direcional dos relacionamentos entre os requisitos e o processo de desenvolvimento facilita a manutenção e verificação do sistema, motivo pelo qual propomos a extensão da metodologia Tropos através da inclusão das atividades de rastreamento de requisitos.

1.4 Objetivos e Abordagem

No nosso trabalho propomos a inclusão de atividades de rastreamento de requisitos na metodologia de desenvolvimento orientada a agentes Tropos. Estas atividades poderão ser realizadas em qualquer domínio de aplicação alterando-se apenas o nível de complexidade do modelo de rastreamento levando-se em consideração características do sistema a ser desenvolvido como complexidade, qualidade do produto exigida pelo cliente, prazo e custo.

Esta pesquisa é motivada pelos seguintes fatores:

- A Engenharia de Software Orientada a Agentes, por ser ainda recente, não apresenta um conjunto consolidado de notações, ferramentas e metodologias de desenvolvimento. Portanto, temos uma excelente oportunidade para contribuir para o estado da arte.
- O rastreamento de requisitos é um conjunto de atividades extremamente importante no desenvolvimento de software como já mencionado. Portanto se faz necessário estender Tropos para que a metodologia possa incluir apoio ao rastreamento de requisitos.

Para validar nossa proposta, será realizado um estudo de caso. Para isso, utilizaremos a metodologia Tropos acrescida das atividades de rastreamento de requisitos no desenvolvimento de um sistema de software orientado a agentes.

1.5 Contribuição

Esta dissertação fornece contribuições para a área de Engenharia de Software Orientada a Agentes, particularmente através da extensão da metodologia Tropos para a inclusão das atividades de rastreamento de requisitos. Portanto, entre os benefícios da nossa proposta, podemos citar:

- Apoio às atividades de gerenciamento de mudanças através das matrizes de rastreabilidade;
- Melhor adequação do sistema aos requisitos do cliente;
- Contribuição para atendimento de algumas das áreas chave presentes nos níveis 2 e 3 do CMM (*Capability Maturity Model*);

- Apoio à atividade de avaliação de impacto de mudanças através das matrizes de rastreabilidade.

1.6 Estrutura da Dissertação

Além deste capítulo introdutório, a dissertação está organizada da seguinte forma:

- O Capítulo 2 apresenta uma revisão geral sobre a Engenharia de Requisitos destacando sua classificação e importância no processo de desenvolvimento de software. Apresentamos uma descrição dos processos que compõem a Engenharia de Requisitos, suas principais características e as atividades de elicitação, análise e negociação, documentação, verificação e validação e gerência de requisitos.
- O Capítulo 3 apresenta uma revisão geral de Rastreamento de Requisitos. Inicialmente o Rastreamento de Requisitos é apresentado no contexto da Engenharia de Requisitos para então apresentarmos os tipos de rastreamento, sua importância e os obstáculos enfrentados para sua utilização no desenvolvimento de software. Finalizamos o capítulo com uma revisão de pesquisas acadêmicas relevantes na área incluindo a pesquisa utilizada nessa dissertação.
- O Capítulo 4 aborda a Engenharia de Software Orientada a Agentes. Inicialmente são apresentadas as motivações para o uso desse novo paradigma, seguido das características e propriedades que caracterizam um agente. São abordadas as motivações e a importância dos sistemas baseados em agentes para a Engenharia de Software. Além disso, são apresentadas as principais áreas de aplicação dos softwares orientados a agentes, bem como os desafios e obstáculos enfrentados atualmente no desenvolvimento desses softwares. O capítulo é concluído com a apresentação da metodologia orientada a agentes utilizada nessa dissertação, Tropos.
- O Capítulo 5 detalha a proposta desta dissertação; a expansão da metodologia de desenvolvimento de software Tropos com a inclusão das atividades de Rastreamento de Requisitos nas suas fases de desenvolvimento.

- O Capítulo 6 fornece um estudo de caso para validar nossa proposta de extensão da metodologia Tropos com a inclusão de atividades relacionadas ao Rastreamento de requisitos. Para isso foi utilizado um sistema multi-agentes de comércio eletrônico desenvolvido conforme metodologia Tropos.
- O Capítulo 7 apresenta as conclusões do trabalho, as contribuições trazidas por este, trabalhos relacionados e melhorias para trabalhos futuros.

Capítulo 2 – Engenharia de Requisitos

Este capítulo contém uma revisão geral sobre a Engenharia de Requisitos destacando sua classificação e importância no processo de desenvolvimento de software. Em seguida são descritos os processos que compõem a Engenharia de Requisitos e suas principais características. O capítulo também revisa as atividades de elicitação, análise e negociação, documentação, verificação e validação e gerência de requisitos.

A Engenharia de Requisitos vem sendo reconhecida como uma área de extrema relevância para o processo da Engenharia de Software. Pesquisadores como Bell e Thayer [BEL76] estabelecem que a maior parte dos problemas, geralmente os mais dispendiosos e de maior impacto negativo no desenvolvimento de software, ocorrem nas etapas iniciais do desenvolvimento. Os problemas de requisitos inconsistentes, incompletos, acabam provocando a criação de produtos de software de baixa qualidade.

As falhas originárias de levantamento de requisitos mal realizado são, geralmente, um dos fatores que levam a atrasos no cronograma de projetos e estouro em seus orçamentos fixados. Nesse contexto, a Engenharia de Requisitos surgiu com o objetivo principal de tratar dos problemas relacionados com requisitos.

Este Capítulo contém uma descrição dos elementos mais importantes da Engenharia de Requisitos. Inicialmente, a Seção 2.1 apresenta uma visão geral, apontando as definições para os termos Engenharia de Requisitos e Requisitos. Em seguida, a Seção 2.2 apresenta algumas das razões para o estudo da Engenharia de Requisitos. A Seção 2.3 trata da classificação dos requisitos enquanto a Seção 2.4 descreve as principais atividades do processo de Engenharia de Requisitos. Por fim a seção 2.5 conclui o capítulo com algumas considerações.

2.1 Visão Geral da Engenharia de Requisitos

Estudos do *Software Engineering Institute* (SEI), nos Estados Unidos, e da comunidade européia concluíram que as sérias deficiências no processo de especificação e na gerência de requisitos são os principais aspectos responsáveis pela atual “Crise de Software” (cf. [STA94], [ESI96]).

A qualidade do software final depende sobremaneira de um processo eficiente de levantamento de requisitos [BEL76], pois em geral muitos requisitos são inadequados, inconsistentes, incompletos e ambíguos, exigindo, assim, uma intervenção racional dos profissionais que trabalham no levantamento e revisões dos mesmos. A construção de software em tempo hábil - com melhor aproveitamento de recursos financeiros e cumprimento dos cronogramas - depende de uma boa definição de requisitos.

Requisitos inadequados desempenham o papel mais expressivo na falha do projeto de software [ALF79]. O desenvolvimento da especificação de requisitos em muitos casos parece

trivial, mas, provavelmente, é a fase do processo que provoca mais falhas que qualquer outra [SCH75].

Com o objetivo de obter uma melhor compreensão da área de Engenharia de Requisitos, apresentam-se alguns dos seus conceitos básicos que contribuirão para um melhor entendimento do assunto em questão.

Um *sistema* pode ser considerado uma coleção de hardware, software, dados, pessoas e procedimentos organizados para alcançar objetivos comuns. No contexto da Engenharia de Software, um sistema é um conjunto de programas que serve para resolver problemas [THA97]. A Engenharia de Software se dedica ao estabelecimento e uso de sólidos princípios de engenharia para que se possa obter, economicamente, um software que seja confiável e que funcione eficientemente [PRE05].

Um outro termo que merece explicação é *sistema de informação*. Um sistema de informação é o conjunto interdependente das pessoas, das estruturas da organização, das tecnologias de informação, dos procedimentos e métodos que deveriam permitir à organização dispor das informações necessárias ao seu funcionamento atual e à sua evolução [LES89].

Neste trabalho, *requisito* é definido como sendo uma propriedade que um sistema deve possuir. Segundo Toranzo [TOR02a], essa propriedade pode incluir elementos provenientes de um (ou mais) dos seguintes segmentos de informação¹:

- Externo: representa regulamentações, leis e restrições que pertencem ao contexto sócio-político e / ou econômico na qual está inserida a organização. Por exemplo, a lei CPMF (Contribuição sobre movimentação financeira) é um elemento externo aos bancos brasileiros. As informações do segmento externo são geralmente identificadas nas fases iniciais de desenvolvimento do sistema como requisitos não-funcionais (seção 2.3);
- Organizacional: representa requisitos organizacionais, isto é, estratégias, restrições e benefícios da organização na qual o sistema será inserido, que obrigatoriamente devem ser levados em consideração a fim de garantir que o sistema atenda plenamente às expectativas do cliente (seção 2.3);

¹ *Segmento de informação*: divisão das informações disponíveis para o desenvolvimento do sistema identificando assim, informações comuns, isto é, informações que dizem respeito a um determinado contexto levando em consideração o sistema a ser desenvolvido e a organização onde ele será inserido.

- Gerencial: Está relacionado à atividade de planejamento da gerência do projeto, mais especificamente, aos relacionamentos entre as tarefas e os requisitos para um melhor acompanhamento e controle dos requisitos do sistema;
- Desenvolvimento: representa os elementos / artefatos produzidos nas diferentes atividades do desenvolvimento de software. Alguns dos artefatos encontrados neste segmento são: documentos de requisitos, diagramas e programas.

Vale ressaltar que a dimensão de requisitos organizacionais é recente, pois o termo requisito era direcionado às necessidades de como desenvolver sistema de software e sistema de informação, onde pouca ênfase era dada aos aspectos mais gerais que fazem parte do domínio da aplicação (requisitos organizacionais)¹.

Neste trabalho utilizamos a seguinte definição de Engenharia de Requisitos: processo iterativo, incremental, cognitivo, social, comunicativo e criativo; cujos objetivos são conhecer, entender, estruturar, representar, comunicar e transcrever as informações relevantes de um sistema, extraídas a partir de diferentes segmentos de informação: externo, organizacional, gerencial, e o desenvolvimento.

2.2 Motivação para o estudo da Engenharia de Requisitos

O estabelecimento dos requisitos é considerado parte essencial para o alcance dos objetivos e metas estabelecidos por todos os atores envolvidos na construção do software. Devido às grandes dificuldades causadas pelas modificações nos requisitos em fases avançadas do processo de desenvolvimento de software, a tarefa de decidir, precisamente, o que construir é a parte mais difícil da construção de um sistema de software [BRO87]. Assim, quando os requisitos de um sistema são mal entendidos, possivelmente o produto final não será livre de falhas e a versão do sistema entregue ao cliente não corresponderá ao produto solicitado inicialmente.

Num mundo cada vez mais globalizado, marcado pelas constantes mudanças, onde os consumidores tornam-se cada vez mais exigentes e onde organizações precisam dar respostas com maior agilidade e eficiência ao enfrentamento dos grandes desafios na gestão dos negócios, torna-se claro que o estudo da Engenharia de Requisitos é um desafio a ser perseguido para melhorar a qualidade dos produtos requeridos pela sociedade.

¹ Outras conceituações de requisitos podem ser encontradas em ([LEI94], [JAC95], [SOM97], [KOT98], [IEE91]) e de Engenharia de Requisitos em ([LAM00], [IEE84], [POH96], [LEI87]).

O problema das falhas no levantamento de requisitos pode ser constatado pela pesquisa realizada pelo *Standish Group*, “*Software CHAOS*” [STA94]. Um estudo realizado entre oito mil projetos desenvolvidos por 350 companhias dos Estados Unidos revelou que um terço dos projetos nunca foram completados. Somente a metade dos projetos obteve sucesso, parcialmente. Como resultado, foram entregues sistemas fora dos cronogramas previamente estabelecidos, com gastos superiores aos orçamentos fixados, e, o que é pior, apresentando funcionalidades parciais.

De acordo com Pressman [PRE05], o custo de correção de erros na fase de projeto é cerca de três a seis vezes mais alto do que na fase de definição de requisitos. O custo de correção de erros na fase de operação do sistema é relativamente cerca de 40 a 100 vezes mais caro do que o custo da correção de erros na fase de definição dos requisitos.

Devido aos problemas acima expostos, várias organizações estão aperfeiçoando os métodos para adquirir, analisar e gerenciar os seus requisitos. Todos os projetos, desde os menores até os maiores, podem se beneficiar da atenção dada aos requisitos [PET98].

Contudo, é preciso ressaltar que, para muitos projetos, os requisitos são fornecidos verbalmente e não são completamente documentados. Pode-se concluir que não existe um único caminho para os clientes, desenvolvedores e testadores comparar anotações e garantir que todos trabalhem com os mesmos objetivos [PET98].

Segundo Kotonya [KOT98], erros nos requisitos originam: a) atraso na entrega do sistema e aumento do custo do desenvolvimento do sistema; b) insatisfação do cliente e dos usuários finais com o sistema; c) pouca confiabilidade no uso do sistema, devido a erros regulares e falhas que levam à interrupção da operação normal do sistema; d) aumento do custo de manutenção e evolução do sistema.

Os benefícios resultantes da aplicabilidade dos pressupostos da Engenharia de Requisitos podem ser vistos de diversas maneiras para os negócios de uma organização. Na visão de Leite [LEI00], destacam-se: a) concordância entre desenvolvedores, clientes e usuários sobre o trabalho a ser feito e quais os critérios de aceitação do sistema; b) uma base precisa para a estimativa dos recursos (custo, pessoal, prazos, ferramentas e equipamentos); c) melhoria na usabilidade, manutenibilidade e outras qualidades do sistema; d) atingir os objetivos com o mínimo de desperdício.

A Seção seguinte apresenta a classificação dos requisitos em funcionais, não-funcionais e organizacionais.

2.3 Classificação de Requisitos

Numa visão tradicional, os requisitos são classificados como requisitos funcionais e não-funcionais, conforme apresentado por Sommerville [SOM03]:

- Requisitos funcionais: são as declarações das funções que o sistema deve oferecer, como ele se comporta com entradas particulares e como o sistema deve se comportar em situações específicas. O termo “função” é usado no sentido genérico da operação que pode ser realizada pelo sistema, seja por meio de comandos dos usuários, seja pela ocorrência de eventos internos ou externos ao sistema. Em alguns casos, os requisitos funcionais podem, explicitamente, definir o que o sistema não deve fazer.
- Requisitos não-funcionais (RNFs): são as restrições nas funções oferecidas pelo sistema. Incluem restrições de tempo; restrições no processo de desenvolvimento; padrões; e de qualidades globais de um software, como custo, performance, confiabilidade, manutenibilidade, portabilidade, usabilidade, desempenho, dentre outras.

De maneira geral, a diferença entre requisitos funcionais e não-funcionais está no fato de os primeiros descreverem “o quê” o sistema deve fazer, enquanto que os outros fixam restrições sobre “como” os requisitos funcionais serão implementados.

Levando-se em consideração as rápidas e significativas transformações que o mundo vem experimentando nos últimos vinte anos - principalmente com a aceleração no processo de globalização econômica, com os avanços no campo da tecnologia e com as novas concepções sobre estratégias de gestão das organizações - diversos estudiosos afirmam que a classificação tradicional não mais apresenta respostas à compreensão de todas as facetas que compõem o ambiente das atuais organizações. A preocupação com a observação de aspectos relativos ao ambiente organizacional já vem se manifestando desde o início dos anos 80 ([LEI83], [DOB94a], [DOB94b]). Uma melhor compreensão da organização requer analisar não apenas requisitos funcionais ([YU97a], [YU97b], [MYL01]), mas também os aspectos organizacionais ([YU97c], [CYS99], [MYL00], [MYL01]) e sociais [SAN98].

Por essa razão, a classificação mais recente de requisitos agrega a conceituação de requisitos organizacionais ([LOU95], [KOT98], [DAV93]):

- Requisitos organizacionais: têm relação com as metas da empresa, suas políticas estratégicas adotadas, os relacionamentos entre os seus atores junto com seus respectivos objetivos.

A inserção dos requisitos organizacionais amplia a visão tradicional que aborda apenas aspectos funcionais e não-funcionais. Portanto os requisitos não se preocupam apenas com "como" o sistema deve fazer, mas sim com "o que" o sistema deve fazer, associado com "o porque" fazer, compreendendo a intencionalidade dos fatos organizacionais [[YU94], [LOU95], [MYL95], [YU95a], [YU97b]].

No enfoque dos requisitos organizacionais, estudam-se os objetivos do próprio sistema e o relacionamento desses com os objetivos da organização, bem como outras propriedades dos sistemas, como as relacionadas com o desempenho, segurança e restrições. Também são abordados problemas e aspectos gerenciais, organizacionais, econômicos, técnicos, sociais e externos.

Em literatura recente, há distinção entre requisitos iniciais (*Early Requirements*) e requisitos finais (*Late Requirements*) [YU97a], conforme conceituado a seguir:

- Requisitos iniciais: caracterizados por atividades que tratam de questões organizacionais e não-funcionais, com ênfase na compreensão das motivações e das razões que estão associadas aos requisitos do sistema.
- Requisitos finais: caracterizados por atividades que tratam de questões como completude, consistência e verificação dos requisitos, dando atenção ao refinamento, eliminação de ambigüidades e completude da descrição da alternativa escolhida.

Na Seção seguinte, são descritas as fases do processo de Engenharia de Requisitos.

2.4 Processo de Engenharia de Requisitos

De acordo com Kotonya [KOT98], um processo é um conjunto organizado de atividades que transforma entradas em saídas, considerando uma atividade inicial do processo de desenvolvimento de software. Para a *International Organization for Standardization* [ISO88] processo é definido como "um grupo de atividades inter-relacionadas que se caracterizam por uma série de entradas específicas que agregam valor e fornecem uma série de saídas específicas para clientes externos e internos".

O processo de Engenharia de Requisitos é um processo de projeto (design) com entradas e saídas, conforme demonstra as informações descritas na Tabela 2-1 a seguir [KOT98].

ENTRADA	DESCRIÇÃO
Informações dos sistemas existentes	Refere-se a informações gerais sobre o sistema que será substituído ou criado e de outros sistemas com os quais o sistema deverá interagir.
Necessidades dos stakeholders¹	Refere-se a uma descrição das necessidades dos <i>stakeholders</i> para apoiar seu trabalho.
Padrões corporativos	Refere-se aos padrões e normas adotadas pela empresa para o desenvolvimento de sistemas, incluindo métodos para o desenvolvimento de sistemas, práticas para garantia de qualidade, etc.
Normas e regulamentos	Normas e regulamentos externos que se apliquem ao sistema.
Informações do domínio	Informações gerais sobre o domínio do sistema.
SAÍDA	DESCRIÇÃO
Requisitos definidos	Descrição dos requisitos levantados, avaliados e aprovados pelos <i>stakeholders</i> .
Especificação do sistema	Uma especificação mais detalhada do sistema a ser desenvolvido.
Modelos do sistema	Um conjunto de modelos que descrevem o sistema a partir de diferentes perspectivas.

Tabela 2-1- Entradas e Saídas do Processo de Engenharia de Requisitos

¹ *Stakeholders*: são pessoas ou organizações que serão afetadas pelo sistema e que possuem influência, direta ou indireta, sobre os requisitos do sistema – usuários finais, gerentes, engenheiros responsáveis pelo desenvolvimento e manutenção do sistema, clientes da organização que usará o sistema para fornecer algum serviço, etc.

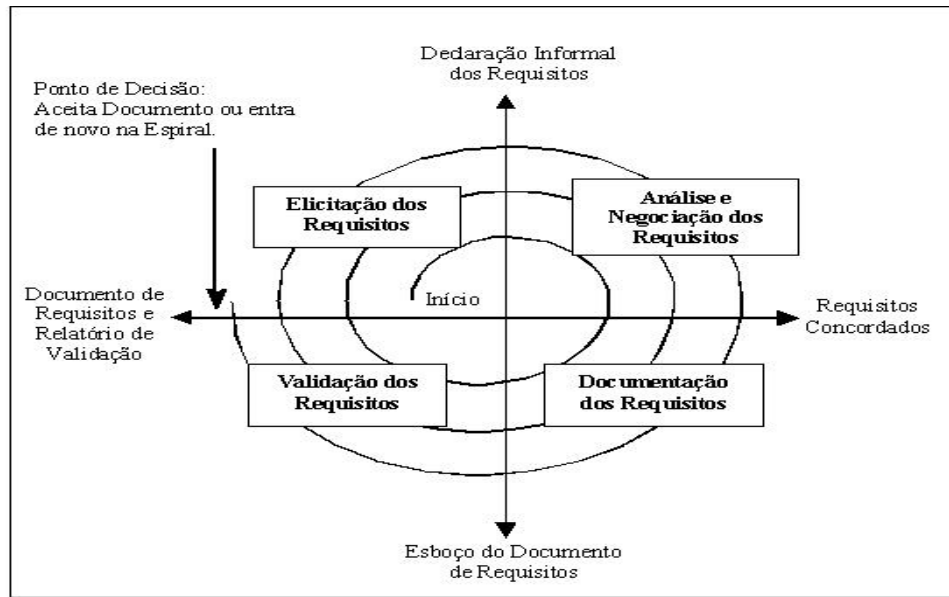


Figura 2-1 Atividades do Processo da Engenharia de Requisitos no Modelo Espiral [KOT97]

Com um processo bem definido, pode-se reutilizar conhecimentos aplicados anteriormente, bem como aproveitar soluções aplicadas com sucesso em problemas similares e adaptá-las de acordo com a realidade dos novos problemas.

Uma descrição completa do processo deve incluir: a) quais atividades são executadas; b) a estruturação e o cronograma delas; c) quem é o responsável por cada uma das atividades; d) suas entradas e saídas; e) e as ferramentas usadas para suportar a Engenharia de Requisitos.

A aplicabilidade dos modelos de processo de Engenharia de Requisitos varia de uma organização para outra. Fatores como maturidade técnica, envolvimento multidisciplinar, cultura organizacional e domínio da aplicação acabam influenciando no grau de estruturação do processo a ser implementado na organização.

A Figura 2-1 é a esquematização de um modelo em espiral, no qual o início do processo ocorre com as atividades de Elicitação de Requisitos, seguido pelas atividades de Análise e Negociação de Requisitos, Documentação dos Requisitos e Validação dos Requisitos. O processo se repete tantas vezes quantas forem necessárias, até o Ponto de Decisão de Aceitação do Documento Final. Na prática, o modelo espiral da Figura 2-1 consiste de processos iterativos, inter-relacionados e com retro-alimentação que podem cobrir todo o ciclo de vida do desenvolvimento de sistemas de software [NUS00].

O aperfeiçoamento do processo de Engenharia de Requisitos requer que as organizações estabeleçam, claramente, quais os objetivos a serem perseguidos para se alcançar o aperfeiçoamento

pretendido. Para Sommerville e Kotonya ([SOM97], [KOT98]), um bom planejamento do processo de aperfeiçoamento, ora em questão, objetiva promover a melhoria da qualidade; redução do tempo dedicado às reuniões realizadas para corrigir erros; e redução de gastos financeiros com o projeto. Os autores afirmam que o delineamento das ações para aperfeiçoamento do processo deve ser precedido de respostas para as seguintes perguntas:

- Quais são os problemas com os processos atuais?
- Quais os objetivos do aperfeiçoamento?
- Como introduzir aperfeiçoamento no processo para atingir os objetivos?
- Como poderá ser controlado e gerenciado o aperfeiçoamento?

Em geral, mesmo reconhecendo a importância de possuir um processo de Engenharia de Requisitos, alguns engenheiros de requisitos, por insegurança, preferem continuar utilizando os seus tradicionais esquemas de trabalho, que muitas vezes garantem uma aparente comodidade, autonomia e certa estabilidade. Para enfrentar esse problema, as organizações precisam adotar medidas capazes de alterar certas amarras burocráticas e culturais, pois não basta a existência de um processo, é imprescindível que os desenvolvedores e outras pessoas conheçam-no, aceitem-no, apliquem-no e o melhorem ([DID03], [PAI03]).

A seguir, serão apresentadas considerações sobre a primeira fase do processo de Engenharia de Requisitos, a elicitação.

2.4.1 Elicitação de Requisitos

A elicitação é a primeira atividade no ciclo de vida da Engenharia de Requisitos, tendo como finalidade geral obter conhecimento relevante para o problema a ser resolvido. A elicitação de requisitos é um processo de descoberta dos requisitos para um sistema, através da comunicação com clientes, usuários finais e com os demais interessados no desenvolvimento do sistema. Essa atividade requer uma criteriosa análise da organização, do domínio de aplicação e dos processos organizacionais, aliada ao processo de descoberta das necessidades dos usuários ([LEI90], [KOT98]).

A fase de elicitação de requisitos visa ao melhor conhecimento da dinâmica organizacional, identificando necessidades, a partir de consulta aos representantes de cada grupo de usuários, de documentos do domínio, de conhecimento do domínio e de pesquisa de mercado. Portanto, a tarefa da elicitação de requisitos é identificar os fatos que compõem os requisitos do sistema, de forma a fornecer o mais correto entendimento do que é esperado do sistema de software. Outro papel do processo de elicitação é identificar as necessidades das diferentes classes de usuários [SHA99], por exemplo, usuários experientes, ocasionais e iniciantes.

Problemas da Elicitação de Requisitos

Em função da dinâmica organizacional, os requisitos de um sistema podem mudar para se adequar às novas exigências da organização ou do ambiente social, político e econômico.

Para Toranzo [TOR02a] os principais problemas encontrados durante a fase de Elicitação podem ser classificados em quatro categorias, a saber:

- **Problemas de Escopo:** representa os problemas de elicitação de requisitos vinculados ao desconhecimento de informações do sistema relacionadas com o ambiente organizacional (contextos sociais, econômicos e políticos, nos quais estão inseridos a organização e o sistema desejado). É necessário descobrir as necessidades, restrições, objetivos, metas e expectativas, bem como as diferentes representações do domínio do problema. Também é necessário identificar as necessidades / restrições de software e hardware impostas ao sistema;
- **Problemas de Entendimento:** estão associados às barreiras de comunicação e às limitações cognitivas e de conhecimento;
- **Problemas Técnicos:** nesta categoria, estão incluídas algumas das dificuldades que afetam o sucesso do processo de elicitação. Algumas delas são: a) mudanças tecnológicas de software e hardware; b) mudança de requisitos ao passar do tempo; c) existência de muitas fontes de informação para os requisitos; d) dificuldade de reuso de conhecimento pelos analistas.
- **Problemas de Comportamento Humano:** são decorrentes da interação entre as pessoas e podem ser de várias modalidades, envolvendo aspectos tanto individuais como coletivos.

Para resolver esses problemas, várias técnicas de elicitação de requisitos foram propostas. Na próxima Subseção, algumas delas são identificadas.

Técnicas de Elicitação de Requisitos

As técnicas de elicitação têm sido desenvolvidas e empregadas pelos engenheiros de software para entender o ambiente, as necessidades dos usuários, e evitar as dificuldades que surgem durante a elicitação de requisitos. A escolha da técnica apropriada para elicitar requisitos depende do tempo e dos recursos disponíveis, assim como do tipo de informação necessária [NUS00].

A elicitação é considerada como um processo contínuo de Engenharia de Requisitos. Pode ser vista como a matéria-prima para outras etapas da Engenharia de Requisitos, como, por exemplo, a especificação. Nesse sentido, elicitação acontece em paralelo à especificação e às atividades de validação [LOU95].

Para uma eficiente Engenharia de Requisitos é necessário identificar, criar, compartilhar e tornar acessível o conhecimento. Diversas técnicas podem ser utilizadas no processo de criação do conhecimento, a exemplo da aplicação de questionários, realização de entrevistas, observação e análise de documentos.

A seguir, são apresentadas algumas classes de técnicas de elicitação de requisitos:

- Técnicas tradicionais: são aplicadas em várias áreas do conhecimento. Cita-se como por exemplo, a aplicação de questionários, a realização de entrevistas, a observação, e a análise de documentos ([WIE95], [MAM99]).
- Técnicas de elicitação de grupo: visam a compreender melhor o pensamento e comportamento dos grupos, com o intuito de captar, de maneira mais detalhada, as necessidades dos usuários. Exemplos: as técnicas de *brainstorming*, as sessões *Joint Application Design* (JAD) e *Rapid Application Development* (RAD) [DAM97].
- Prototipação: um protótipo de um sistema é uma versão inicial do sistema que está em fase de desenvolvimento. Em sistemas de hardware, são desenvolvidos, freqüentemente, protótipos para testar e experimentar os projetos de sistema. Em sistemas de software, os protótipos também podem ajudar na elicitação e validação dos requisitos de sistema

[SOM97]. Geralmente, a prototipação é utilizada para elicitar requisitos quando há um alto grau de incerteza ou quando é necessário um rápido feedback dos usuários.

- Técnicas cognitivas: originalmente desenvolvidas para aquisição de conhecimento para sistemas baseados em conhecimento, como, por exemplo: análise de protocolo, *laddering*, *card sorting* e *repertory grids* [SHA95].
- Técnicas contextuais: surgiram como uma alternativa para as técnicas tradicionais e cognitivas e incluem técnicas tais como a etnografia e análise social [VIL99]. Compartilham a visão de que o trabalho é uma atividade social que envolve grupos de pessoas que cooperam para o desenvolvimento de diferentes tarefas. A natureza da cooperação é freqüentemente complexa, e varia, dependendo das pessoas envolvidas, o ambiente físico e a organização na qual o trabalho acontece.

O processo de elicitação é essencial para o sucesso da construção de um software. A elicitação é uma atividade que requer capacidade humana para extrair as informações que retratarão, com fidelidade, os interesses dos diversos atores que exercem (ou sofrerão) influência direta ou indireta no desenvolvimento. Portanto, reconhecendo-se a sua importância, a elicitação de requisitos não deve ser realizada às pressas, empiricamente, sem o uso correto das técnicas. Requer uma comunicação eficiente e um compromisso responsável do grupo para trabalhar com destreza para promover o efetivo atendimento das expectativas dos usuários, das organizações e da sociedade de um modo geral.

A seguir, serão apresentadas considerações sobre a fase Análise e Negociação de Requisitos do processo de Engenharia de Requisitos.

2.4.2 Análise e Negociação de Requisitos

Análise e negociação de requisitos são atividades que visam a descobrir problemas com os requisitos de sistema e estabelecer um acordo de mudanças que satisfaça a todos os *stakeholders* do sistema. É importante salientar que a análise é normalmente intercalada com a atividade de elicitação de requisitos. Porém, uma atividade detalhada de análise geralmente acontece depois que o esboço inicial do documento de requisitos é produzido.

O processo de análise e negociação de requisitos é caro e demorado porque necessita de pessoas qualificadas e experientes para dedicar tempo, por exemplo, pode requerer uma leitura

cuidadosa dos documentos e a identificação das implicações das declarações contidas nestes documentos [SOM03].

A análise promove a identificação de problemas com requisitos conflitantes, ambíguos, redundantes, dentre outros. Os usuários devem discutir, priorizar e negociar esses problemas até que se obtenha um acordo com possíveis modificações e simplificações dos requisitos.

A negociação é um processo que envolve e considera todas as visões de todos os usuários relacionados com requisitos conflitantes não resolvidos. Ela é influenciada por aspectos organizacionais e políticos e pela personalidade dos participantes.

A seguir são apresentadas algumas técnicas que podem ser utilizadas nas atividades de análise:

- **Lista de Checagem da Análise:** é uma lista de itens que o analista pode usar para avaliar cada requisito. No final da checagem, pode-se disponibilizar uma lista de problemas associados com os requisitos, os quais podem ser solucionados através de negociação ou, se necessário, com nova elicitação;
- **Matriz de Interação:** é utilizada para descobrir as interações entre requisitos, apontando possíveis conflitos entre estes. A atividade de negociação pode solucionar esses conflitos;
- **Prototipação:** os protótipos desenvolvidos na etapa de elicitação de requisitos podem ser aperfeiçoados na etapa de análise e negociação, possibilitando um melhor entendimento dos requisitos. Protótipos facilitam o envolvimento dos diferentes *stakeholders* na elicitação, análise e negociação, e validação de requisitos.

É fundamental que os engenheiros de requisitos conheçam bem o potencial e as limitações das técnicas disponíveis, podendo escolher com mais segurança a ferramenta que melhor sirva ao objetivo pretendido.

A seguir, serão apresentadas considerações sobre a fase Documentação de Requisitos do processo de Engenharia de Requisitos.

2.4.3 Documentação de Requisitos

Em conformidade com a Figura 2-1, após as atividades de Elicitação, Análise e Negociação de Requisitos, as próximas atividades são as de Documentação e de Validação de Requisitos. O documento de requisitos é o produto final do processo de desenvolvimento de requisitos que reúne necessidades e propósitos demandados pelos *stakeholders*, servindo como um contrato entre usuários e desenvolvedores. É recomendável que o documento de requisitos seja compreendido por todos os envolvidos no processo de Engenharia de Requisitos a fim de que todos tenham conhecimento e comprometimento com o escopo do sistema a ser desenvolvido.

Na literatura, são encontradas várias conceituações para o documento de requisitos, a exemplo de:

- É uma declaração formal de requisitos de clientes, usuários finais e desenvolvedores de software [SOM97];
- É o meio utilizado para descrever: as restrições quanto às características do produto e quanto ao processo de desenvolvimento; a interface com outras aplicações; as informações sobre o domínio da aplicação; e informações de suporte ao conhecimento do problema, como, por exemplo, modelos, termos especializados do negócio, recuperação e gerenciamento de informações em mudança [KOT98].

Na metodologia Tropos os requisitos são documentados através dos diagramas de razão estratégica e dependência estratégica. A metodologia não sugere especificação explícita dos requisitos.

De acordo com a norma IEEE [IEE96], um documento de requisitos visa a possuir declarações não ambíguas e ser completo, verificável, consistente, modificável, rastreável e usável durante todas as fases do ciclo de vida do requisito.

Entre os principais objetivos de um documento de requisitos podemos incluir:

- Servir como instrumento de acordo (contrato) entre cliente e fornecedor sobre o que o software deverá fazer;
- Servir de base de especificação de software e projeto;

- Promover o suporte à verificação, validação do software e gerenciamento do projeto;
- Promover a redução do tempo total e esforço dedicado ao processo de criação do software, evitando o reprojeto, a recodificação e a necessidade de refazer os testes;
- Permitir o rastreamento e gerenciamento dos requisitos, ao longo da evolução do sistema.

A seguir, serão apresentadas considerações sobre a fase Verificação e Validação de Requisitos do processo de Engenharia de Requisitos.

2.4.4 Verificação e Validação de Requisitos

A verificação é um processo para assegurar que a solução do projeto satisfaça aos requisitos, enquanto que a validação é um processo para confirmar que as reais necessidades dos clientes estão contempladas no sistema construído [YOU02]. A qualidade dos requisitos pode ser melhorada e os custos e riscos podem ser controlados executando-se o planejamento prévio da verificação e da validação no processo do desenvolvimento de software.

Considerada a fase final da Engenharia de Requisitos, a validação não é considerada uma atividade separada das etapas de análise e especificação de requisitos. A validação de requisitos é definida como o processo que certifica que o modelo de requisitos é consistente com as reais necessidades dos clientes e intenções de usuários. Essa visão de validação é a mais geral que se encontra na literatura porque trata validação como uma atividade contínua, ocorrendo, geralmente, em paralelo às fases de elicitação e especificação. A necessidade de validação aparece no momento em que surge uma nova informação que deve ser agregada e assimilada pelo modelo atual de requisitos (a relevância, validade, consistência, etc., da nova informação devem ser examinadas) e quando partes diferentes de informação são integradas em um todo coerente. Validação, então, não é só aplicada no modelo final de requisitos, mas em todos os modelos intermediários produzidos, inclusive a informação recente recebida pelo processo de elicitação. [LOU95].

A validação de requisitos visa confirmar que o documento de requisitos e os diagramas complementares refletem todas as reais necessidades dos clientes e usuários finais, fornecendo os documentos que servirão de base para as fases seguintes do processo de desenvolvimento do sistema.

Em geral, a validação de requisitos é uma atividade que deve criar os meios necessários para que ocorra um consenso entre diferentes usuários, geralmente com objetivos conflitantes. Validação, portanto, não é tarefa fácil. Requer, dos membros do grupo de validação, tempo, disposição, dedicação, abertura de bons canais de comunicação e atenção para checagem de requisitos, principalmente quando o sistema é complexo. A validação é uma atividade que pode requerer várias sessões de trabalho e na qual as pessoas não somente precisam concordar com os requisitos, mas também visualizar as implicações futuras das suas decisões. As atividades de elicitação e validação são de responsabilidade do analista, cliente, e usuários finais, sobretudo dos clientes e usuários finais que são supostamente os especialistas do domínio. Uma revisão sobre a validação de requisitos pode ser encontrada em Kotonya [KOT98].

Enquanto o projeto e a implementação possuem o documento de requisitos para verificar seus resultados, a validação de requisitos não possui nenhuma outra representação que possa ser usada como base. Não é possível demonstrar formalmente que a especificação está correta [KOT98]. Contudo, existem várias técnicas que podem ser aplicadas para suportar a validação de requisitos, a exemplo da prototipação, do parafraseamento da linguagem natural, da validação de modelos [KOT98], dos testes de requisitos [CYS01] e dos sistemas especialistas [LOU95].

Na próxima seção serão apresentadas considerações sobre o Gerenciamento de Requisitos.

2.4.5 Gerenciamento de Requisitos

O Gerenciamento de Requisitos é uma atividade que não está presente na Figura 2-1, pois trata-se de uma atividade paralela às demais, isto é, ocorre em todos os momentos do desenvolvimento do software. Nessa atividade são tratados os aspectos de rastreamento de requisitos.

Tendo como objetivos principais o controle e gerenciamento de mudanças nos requisitos, as atividades de gerenciamento de requisitos ganham cada vez mais notoriedade no processo de desenvolvimento de software. A gerência de requisitos visa o acompanhamento de um requisito ao longo de todo o processo de desenvolvimento. Ela cuida dos requisitos acordados, das relações entre requisitos, das dependências entre os documentos de requisitos e outros documentos produzidos durante o processo de desenvolvimento do software.

Em geral, as mudanças de requisitos de um sistema podem ocorrer devido a erros ou mau entendimento dos mesmos, ou ainda problemas de implementação, que podem gerar efeitos colaterais e se propagar negativamente para outras partes do software. Novos requisitos surgem quanto os *stakeholders* desenvolvem um melhor entendimento do sistema. Porém, comumente,

mudanças de requisitos são o resultado de mudanças nas circunstâncias externas à organização. As estratégias da organização podem mudar em razão do estabelecimento de novos objetivos e metas, exigidas por mudanças econômicas, legais, políticas, culturais, dentre outras.

2.5 Considerações Finais

Este Capítulo apresentou uma breve retrospectiva da Engenharia de Requisitos com ênfase nos seus conceitos principais, motivações para estudo, a classificação de requisitos, o processo de Engenharia de Requisitos e principais técnicas usadas. O processo de requisitos foi subdividido em cinco atividades, a saber: Elicitação; Análise e Negociação; Documentação; Validação e Gerenciamento do processo de requisitos.

A Engenharia de Requisitos, apesar de ser uma área de estudo muito recente, vem ganhando uma notoriedade cada vez maior, em razão da enorme contribuição que vem dando para redução de custos e melhoria da qualidade do processo de desenvolvimento de software.

As pesquisas recentes em relação às técnicas de Modelagem Organizacional vêm demonstrando a importância dos aspectos inerentes à dinâmica organizacional e salientando os impactos oriundos das interações das organizações com os ambientes externos. As mudanças nos aspectos sociais, econômicos, legais, organizacionais, humanos, dentre outros, estão alargando a visão tradicional sobre requisitos. Os requisitos na visão moderna não se preocupam apenas com "o que" e "como" o sistema deve fazer, mas também com "o porque", compreendendo a intencionalidade dos fatos organizacionais.

O sucesso de um processo de requisitos depende da implantação de uma boa gerência de requisitos, composta por profissionais qualificados, possuidores de uma visão geral da organização e comprometidos com os trabalhos de Engenharia de Requisitos.

No próximo capítulo descreveremos a atividade de Rastreamento de Requisitos, abordando a necessidade de sua inclusão no processo de desenvolvimento de software.

Capítulo 3 – Rastreamento de Requisitos

Este capítulo apresenta uma revisão geral de Rastreamento de Requisitos. Inicialmente o Rastreamento de Requisitos é apresentado no contexto da Engenharia de Requisitos. As seções seguintes apresentam os tipos de rastreamento, sua importância e os obstáculos enfrentados para sua utilização no desenvolvimento do software. O capítulo é finalizado com uma revisão de pesquisas acadêmicas relevantes na área.

Capítulo 3 RASTREAMENTO DE REQUISITOS

A gerência de requisitos enfatiza estabelecer e manter um acordo com o cliente em relação aos requisitos a serem observados no projeto de software [CAP98]. Algumas das atividades da gerência de requisitos são: rastreamento de requisitos, controle de mudança, controle de versão e inspeção do estado dos requisitos.

O rastreamento de requisitos é uma atividade importante para o sucesso da gerência de requisitos, pois estabelece relacionamentos entre os requisitos e os elementos produzidos no processo de desenvolvimento de software a fim de gerenciar as propostas de mudanças e garantir o sucesso dos sistemas [CAR01]. Na literatura encontramos várias definições para o rastreamento de requisitos, por exemplo, o padrão IEEE 830-1984 [IEE84] estabelece que:

Uma especificação de requisitos de software é rastreável se (i) a origem de cada requisito é clara e se (ii) ela facilita a referência de cada requisito num desenvolvimento futuro ou na melhoria da documentação.

Já Gotel e Finkelstein expressam este conceito de uma maneira mais complexa [GOT95]:

Rastreamento de Requisitos é a habilidade de descrever e seguir a vida de um requisito a partir da sua origem (backward), como também através do seu desenvolvimento e especificação, até a sua distribuição e uso (forward); através de períodos de refinamento e iterações em qualquer uma dessas fases.

Para Pinheiro [PIN03] há dois aspectos importantes do rastreamento de requisitos que devem ser utilizados para estender essa definição. O primeiro aspecto é a habilidade de capturar os rastros que queremos seguir e o outro é a idéia de que rastros devem ser vistos como ocorrências naturais, isto é, devem ser naturalmente produzidos. Ele propõe a seguinte definição para o rastreamento de requisitos:

Rastreamento de Requisitos refere-se à habilidade de definir, capturar, e seguir os rastros deixados pelos requisitos em outros elementos do ambiente de desenvolvimento de software e os rastros deixados por esses elementos nos requisitos.

Para ele o ambiente de desenvolvimento de software envolve não apenas os aspectos técnicos, mas também os aspectos sociais. Os elementos não são apenas os artefatos técnicos, como especificações, diagramas e código, mas também as pessoas, políticas, decisões e coisas intangíveis, como objetivos e conceitos.

Na próxima seção descrevemos a importância do rastreamento de requisitos no processo de desenvolvimento de software.

3.1 Importância do Rastreamento de Requisitos

Durante nossa pesquisa identificamos vários fatores que justificam o estudo e importância do rastreamento de requisitos. A seguir identificamos alguns dos fatores: Qualidade de software; Melhoria contínua de um processo; Análise de impacto e implementação de uma proposta de mudança; Manutenção de software e Melhoria do acompanhamento do progresso de um projeto.

A seguir explicaremos cada um dos fatores:

- Qualidade de software: A área de qualidade de software tem-se caracterizado pelo estudo e proposta de normas [ISO97] e modelos de qualidade (CMM [PAU93]), que, se aplicadas corretamente, poderiam contribuir para obter um software com maior qualidade. De fato, o rastreamento de requisitos é uma das atividades exigidas no nível dois do *Capability Maturity Model for Software* (CMM).
- Melhoria contínua de um processo: Entende-se por melhoria contínua de um processo o *feedback* contínuo da utilização de um processo durante o desenvolvimento do software e conseqüente melhoria do processo através desse *feedback*. Considerando-se o conjunto de relacionamentos estabelecidos entre diferentes artefatos do processo de software é possível responder algumas perguntas que contribuiriam para a melhoria de um processo de software. Por exemplo, considerando que foram estabelecidos relacionamentos entre as tarefas do cronograma do projeto, documento de requisitos e documentação do projeto, então algumas das seguintes perguntas poderiam ser respondidas:
 - ◇ Por que um projeto foi bem sucedido ou mal sucedido? A inspeção dos relacionamentos entre requisitos e tarefas pode indicar como os requisitos dos clientes foram distribuídos no decorrer de um projeto.

- ◇ Quais foram os artefatos produzidos por cada programador? Uma alocação de uma tarefa para um ou mais recursos humanos pode ser enriquecida com o estabelecimento de relacionamentos entre pessoas e artefatos. Desta forma, identificaremos as pessoas responsáveis por cada um dos artefatos. Essa informação é importante para selecionar a pessoa mais apropriada para dar manutenção a um sistema.
- ◇ Quem foi o responsável por um determinado subsistema? De forma análoga ao item anterior, relacionamentos entre pessoas e artefatos ajudam a selecionar as pessoas mais apropriadas para dar manutenção a um subsistema.
- ◇ Quais foram os requisitos alocados nos diferentes subsistemas? Se forem estabelecidos relacionamentos entre requisitos e subsistemas, então uma inspeção desses relacionamentos pode identificar os subsistemas que implementam os requisitos do projeto. Isto será útil quando desejarmos reusar um subsistema no desenvolvimento de um outro sistema.

Estas e outras perguntas podem ser formuladas no momento em que os profissionais que participaram de um projeto já não mais trabalham na organização. Portanto, se o rastreamento não for realizado, maior tempo e recursos serão gastos na manutenção. Tradicionalmente, determinar os efeitos colaterais das mudanças no software tem sido realizado através de uma inspeção do código fonte e revisão da documentação. Contudo tal estratégia, embora possa funcionar de forma eficiente para pequenos sistemas, não é adequada para sistemas grandes e complexos.

- Análise de impacto e implementação de uma proposta de mudança: É sabido que no decorrer do processo de desenvolvimento, os clientes do sistema compreendem melhor alguns dos benefícios do sistema desejado. Isto os leva a formular ou reformular requisitos através das propostas de mudança. O uso do rastreamento em um projeto pode contribuir para conhecer e argumentar a decisão de aceitar ou rejeitar uma proposta de mudança. Por exemplo, uma proposta de mudança pode propor a inclusão de novos campos de informação para o gerenciamento dos clientes, mas a análise de impacto pode determinar que diagramas, programas, relatórios, interface, tabelas e código SQL

(*Structure Query Language*) precisam ser modificados. Com base na análise de impacto, o gerente poderá fazer uma estimativa de tempo e custo mais confiável e realista;

- Manutenção de software: Se um erro numa determinada funcionalidade (requisito) é detectado durante sua execução no sistema, este erro deveria ser identificado, resolvido, e as razões das suas causas, elucidadas. Se o rastreamento de requisitos estabeleceu relacionamentos entre requisitos e programas, podemos identificar e corrigir o erro mais eficientemente, pois saberemos de acordo com os relacionamentos qual programa implementa o requisito que apresenta problemas. Como afirmado por [PAL97], um dos benefícios do rastreamento de requisitos é reduzir o tempo e custo da manutenção e um sistema;
- Melhoria do acompanhamento do progresso de um projeto: Se o profissional do rastreamento de requisitos acompanha e registra os relacionamentos das tarefas com os artefatos e os requisitos do sistema, então existe uma grande probabilidade de se obter informações mais realistas do progresso do sistema.

Diante dos fatores que justificam o estudo e importância do rastreamento de requisitos, é possível afirmar que o rastreamento é um fator crucial para várias atividades do processo de desenvolvimento de software, pois fornece um conjunto de relacionamentos entre requisitos e artefatos que facilitam o trabalho de outras atividades. A manutenção de um sistema poderia realizar-se sem empregar rastreamento, porém, o custo para identificar os artefatos afetados pela introdução de uma mudança pode ser alto.

Uma organização que deseja empregar boas práticas de engenharia de software deve considerar a possibilidade de incluir a atividade do rastreamento de requisitos dentro das suas atividades de desenvolvimento. Para isso, é necessário que o rastreamento seja visto como uma atividade tão importante como as outras atividades do desenvolvimento, mesmo que o custo do investimento no rastreamento de requisitos seja alto e talvez não tenha um retorno tão imediato. O recomendável será sempre rastrear durante o processo de desenvolvimento, e não depois, ou apenas quando existir um problema ou proposta de mudança.

Na próxima seção apresentamos os tipos de rastreamento existentes.

3.2 Tipos de Rastreamento

Existem diversas maneiras de realizar um rastreamento de requisitos. Levando-se em consideração a direção do rastreamento, um requisito pode ser rastreado “para frente” ou “para trás”; levando-se em consideração a evolução do requisito, um requisito pode ser rastreado para aspectos que ocorrem “antes” ou “depois” da sua inclusão na especificação de requisitos; e levando-se em consideração os tipos dos objetos envolvidos, tem-se um rastreamento “interno” ou “externo”.

3.2.1 Rastreamento Para Frente e Para Trás

De acordo com [WIE95] as definições de rastreamento para frente e para trás são as seguintes:

Rastreamento para frente é a habilidade de rastrear um requisito para componentes de design ou implementação.

Rastreamento para trás é a habilidade de rastrear um requisito para sua fonte, isto é, para uma pessoa, instituição, lei, argumento, etc.

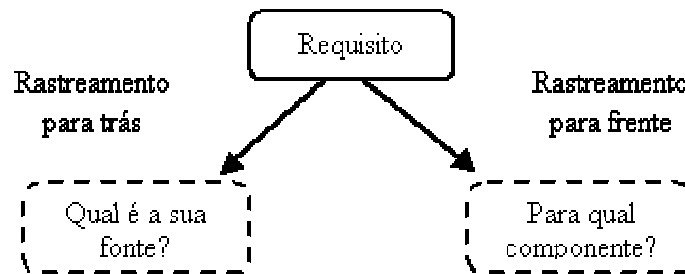


Figura 3-1 Rastreamento para frente e para trás

A Figura 3-1 apresenta um exemplo de utilização para os dois tipos de rastreamento, “Para Frente” e “Para Trás”. O rastreamento deve ser realizado para frente quando, por exemplo, um requisito é alterado e queremos investigar o impacto da mudança. Por exemplo, poderíamos localizar todos os procedimentos de teste relacionados para decidir quais testes devem ser adicionados, modificados ou removidos a fim de que os procedimentos de teste estejam de acordo com o requisito alterado.

O rastreamento deve ser realizado para trás quando, por exemplo, ocorre uma mudança e queremos compreendê-la, investigando a informação utilizada para elicitar o requisito alterado. Podemos rastrear quais as pessoas interessadas na alteração do requisito ou a partir de quais

documentos o requisito foi extraído ou ainda, com quais departamentos da organização o requisito está relacionado.

3.2.2 Rastreamento Pré-Especificação e Pós-Especificação de Requisitos

Uma especificação de requisitos é o resultado de um processo de elicitação. O rastreamento de um requisito pode ser realizado de duas formas: localizando informação relacionada ao processo de elicitação, antes da conclusão da especificação de requisitos, ou localizando informação relacionada ao seu uso, depois do requisito ter sido elicitado e incluído na especificação de requisitos. A Figura 3-2 ilustra esses dois conceitos [GOT94b].

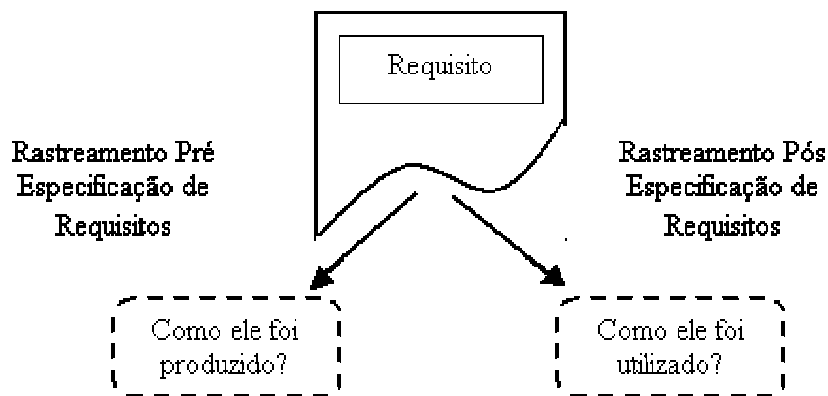


Figura 3-2 Rastreamento Pré Especificação e Pós Especificação de Requisitos

O rastreamento pré-especificação de requisitos é útil, por exemplo, quando ocorre uma mudança num requisito e queremos localizar as fontes dos requisitos ou as pessoas responsáveis a fim de validar a mudança. O rastreamento pós-especificação de requisitos é útil, por exemplo, para localizar o módulo de design no qual o requisito foi alocado ou os procedimentos de teste criados para verificar o requisito.

3.2.3 Extra-Rastreamento e Intra-Rastreamento de Requisitos

Durante a elaboração de uma especificação, os requisitos são comumente refinados, derivados e alguns são descartados. A habilidade de rastrear os relacionamentos entre os requisitos é chamada de intra-rastreamento de requisitos. Os relacionamentos entre os requisitos e outros artefatos são capturados pelo extra-rastreamento de requisitos [PIN03].

O intra-rastreamento de requisitos é importante para a análise e para o tratamento de mudanças e evolução dos requisitos.

A seguir identificaremos e explicaremos alguns dos obstáculos encontrados no rastreamento de requisitos.

3.3 Obstáculos Encontrados no Rastreamento de Requisitos

Nesta seção identificamos e discutimos alguns obstáculos que dificultam o progresso do rastreamento de requisitos. Esses obstáculos estão num escopo que varia desde a falta de uma definição comum de rastreamento de requisitos, até a pouca pesquisa sobre o assunto em relação a aspectos organizacionais. A seguir apresentamos os obstáculos:

- Falta de uma definição padronizada: Existem diferentes definições de rastreamento de requisitos, o que causa um mau entendimento e conseqüentemente má aplicação do rastreamento de requisitos no processo de desenvolvimento de software. Informações detalhadas sobre o esquema de classificação das definições de rastreamento de requisitos podem ser encontradas em [GOT95];
- Falta de reconhecimento e comprometimento organizacionais: Um dos primeiros passos para melhorar um processo de desenvolvimento de software é reconhecer e aceitar que a ausência do rastreamento de requisitos faz com que o processo de manutenção tenha problemas e efeitos colaterais indesejáveis na implementação de uma proposta de mudança. Os passos mais importantes na introdução da prática do rastreamento em uma organização são a participação e o comprometimento organizacional nos investimentos em tecnologia, treinamentos e na institucionalização da prática do rastreamento de requisitos. Isso será possível desde que a organização reconheça e compreenda o verdadeiro potencial do rastreamento de requisitos para o processo de software da organização;
- Falta de orientações acerca das informações que devem fazer parte de um modelo de rastreamento: Apesar da área de rastreamento de requisitos ter mais de duas décadas, ainda persiste a falta de um acordo dos elementos que devem fazer parte de um modelo de rastreamento. Geralmente, as informações a serem rastreadas são classificadas e tratadas nas pesquisas como informações de pré-especificação e pós-especificação de requisitos, porém, surgem as seguintes perguntas:

- ◇ A forma de classificar a informação que deve ser rastreada é suficiente para identificar os elementos que devem fazer parte de um modelo de rastreamento?
 - ◇ Como podemos classificar as informações dentro do pré e / ou pós-especificação?
 - ◇ Existe uma outra forma de classificar as informações a serem rastreadas além do pré e pós-especificação?
- Falta de uma cultura organizacional no uso das boas práticas de engenharia de requisitos, especialmente do rastreamento de requisitos ([SOM03]; [WIE95]; [RAM01]). Geralmente, muitos gerentes de projetos e desenvolvedores consideram o rastreamento de requisitos como uma atividade que sobrecarrega os seus trabalhos e não fornece nenhum benefício direto e imediato à equipe de desenvolvimento;
 - Carência de modelos de rastreamento de requisitos para orientar a elaboração de um projeto. Muitas das pesquisas relacionadas com o pré-rastreamento e o pós-rastreamento de requisitos não fornecem modelos de rastreamento para guiar a elaboração de um modelo para o pré ou pós-rastreamento. Algumas exceções a este item são os trabalhos de ([RAM98a]; [RAM01]; [TOR02a]; [JAR99]).

Dessa forma, concluímos a apresentação dos obstáculos relacionados com o rastreamento de requisitos. Nas próximas seções apresentaremos uma revisão das pesquisas mais relevantes na área de rastreamento de requisitos.

3.4 Revisão de Trabalhos de Pré-Especificação de Requisitos

Até o início da década de noventa, a aplicação do rastreamento de requisitos era associar a documentação de projeto (design) com o documento de requisitos. Naquela época, [FIN91] identificou um novo tipo de relacionamento para o rastreamento de requisitos: relacionar um documento de requisitos com suas fontes de informação. A aplicação desse tipo de rastreamento de requisitos é importante e independente de qualquer paradigma de desenvolvimento.

Em [GOT94b] foram apresentados os resultados de um trabalho empírico relacionado com a identificação e o entendimento dos problemas e das práticas associadas com o rastreamento de requisitos. O rastreamento de requisitos tradicional consiste em estabelecer relacionamentos

bidirecionais entre requisitos e os diferentes artefatos produzidos por um processo de desenvolvimento. Porém, Gotel e Finkelstein recomendaram:

- Divisão do rastreamento tradicional: Considerando como ponto de referência o documento de requisitos, os autores propõem que o rastreamento tradicional poderia ser dividido em dois tipos: Pré-Especificação de Requisitos (Pre-RS) de requisitos e Pós-Especificação de Requisitos (Pos-RS). É importante ressaltar que os dois tipos de rastreamento (Pre-RS e Pos-RS) são importantes, sendo que as principais diferenças entre as abordagens são as informações gerenciadas e os problemas abordados em cada uma delas [GOT93]. A Figura 3-3 apresenta de maneira simplificada a divisão do rastreamento de requisitos proposta por Gotel e Finkelstein, identificando apenas o rastreamento de requisitos horizontal através de uma progressão linear de documentos subsequentes. O rastreamento de requisitos vertical entre versões de documentos não é apresentado;
- Uma maior pesquisa no que diz respeito a Pré-Especificação de Requisitos: A Pré-Especificação de Requisitos depende da habilidade de rastrear requisitos para suas fontes originais (o processo de produção e refinamento de requisitos), no qual as declarações de diversas fontes são, eventualmente, integradas e incluídas no documento de requisitos [GOT96];
- Maior suporte computacional para o Pré-Rastreamento de requisitos. Atualmente existem várias ferramentas disponíveis para o Pós-Rastreamento de requisitos. As funcionalidades das ferramentas disponíveis para o Pré-Rastreamento são poucas porque ele é considerado opcional pelos fabricantes das ferramentas [GOT95].

Gotel e Finkelstein abordaram os seguintes problemas: a identificação das pessoas que são fontes de informação dos requisitos; as propriedades relacionadas com os requisitos; e as atividades relacionadas com a produção e refinamento dos requisitos. Para Gotel é crucial rastrear os indivíduos e grupos de pessoas que contribuíram no processo de Engenharia de Requisitos porque um dos problemas do rastreamento de requisitos nos projetos é a inabilidade de acessar as informações relacionadas com a produção e refinamento de requisitos.

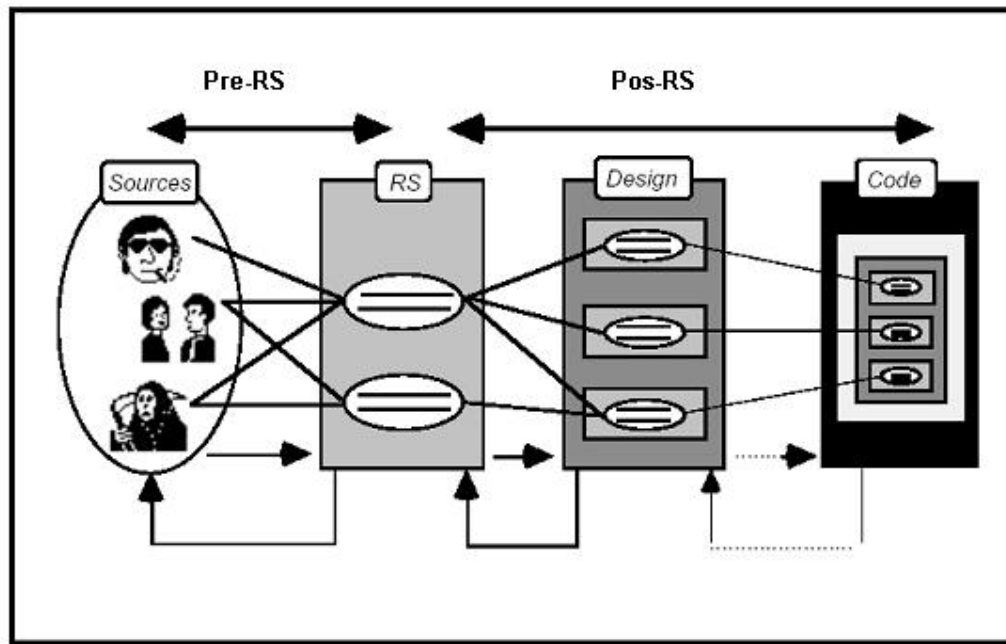


Figura 3-3 Diagrama simplificado dos dois tipos básicos de rastreamento de requisitos: PRE-RS e POS-RS

O trabalho de Gotel e Finkelstein está fundamentado nos seguintes termos: contribuição (artefatos), contribuidor (agente), estrutura social, relação de contribuição, estrutura da contribuição, e rastreamento baseado em contribuição (artefato).

O termo *contribuição* é usado para referir-se a qualquer ocorrência comunicativa no processo de Engenharia de Requisitos que tenha uma existência física. Alguns exemplos de contribuição são e-mails informais, diagramas e documento de requisitos. Os encontros, conversações e reuniões informais podem ser considerados contribuições se os mesmos foram registrados em um meio físico. O trabalho não impõe restrição em relação ao conteúdo, representação e granularidade das informações registradas sobre as contribuições.

O termo *contribuidor* (agente) refere-se a todos os participantes humanos no processo de Engenharia de Requisitos. Os participantes são organizados em uma estrutura social. Uma estrutura social é um sistema de contribuidores no processo de Engenharia de Requisitos que participam na produção de contribuições.

O termo *relação de contribuição* é usado para representar qualquer relacionamento físico (bidirecional) que exista entre um contribuidor e uma contribuição que expressa a forma como o contribuidor participa na produção da contribuição.

O termo *estrutura de contribuição* é o conjunto de todas as relações de contribuição e inter-relações estabelecidas entre os contribuidores e as contribuições.

O termo *rastreamento baseado em contribuição* refere-se a todas as relações definidas entre as contribuições.

O processo do rastreamento de requisitos proposto por Gotel e Finkelstein inclui as seguintes atividades:

- Formato de contribuição;
- Relações entre artefatos;
- Qualificação do formato de contribuição;
- Papéis da contribuição social e papéis das relações;
- Comprometimento do contribuidor com as contribuições.

A primeira atividade, chamada *formato de contribuição*, consiste na identificação e análise dos contribuidores envolvidos na produção das contribuições e nos receptores das mesmas. Através da identificação dos contribuidores, esta atividade reconhece que as contribuições são produzidas e usadas dentro de um ambiente social. Para identificar a natureza das relações de contribuição, Gotel identifica três papéis (principal, autor e documentador), nos quais os contribuidores podem contribuir na produção das contribuições. O contribuidor *chefe* é responsável por motivar a produção das contribuições. O contribuidor *autor* é responsável pela organização do conteúdo e estrutura da contribuição. O contribuidor *documentador* é responsável pelo registro das informações nas contribuições.

A segunda atividade, chamada *relações entre artefatos*, tem o objetivo de identificar as relações entre as contribuições. Gotel sugere três categorias de relacionamento: temporal – que descreve a estrutura histórica de desenvolvimento, e fornece meios para rastrear a história de um requisito, desenvolvedor – que descreve a estrutura lógica do desenvolvimento, e fornece meios para rastreamento de requisitos através do desenvolvimento do software, auxiliar – que descreve os caminhos adicionais (dentro e através dos artefatos) nos quais a informação está relacionada, e fornece formas e estruturas suplementares de rastreamento.

A terceira atividade, *qualificação do formato de contribuição*, consiste em definir um conjunto de atributos e detalhar o estado da contribuição. Por exemplo, uma relação de contribuição

entre o contribuidor chefe e uma contribuição poderia ser qualificada com um dos seguintes atributos: aprovado, reprovado ou aprovado com restrição.

A quarta atividade, *papéis da contribuição social e papéis das relações*, usa e distingue os conjuntos de papéis da contribuição básica e da contribuição derivada. Os papéis da contribuição básica podem ser: *chefe*, *autor* e *documentador*. Os papéis de contribuição derivada são papéis sociais que os contribuidores ocupam enquanto participam na produção de uma contribuição no processo de Engenharia de Requisitos. Alguns papéis das contribuições derivadas são: *autor nominal* e *autor representativo*. O autor *nominal* expressa que o contribuidor é o chefe e o documentador da contribuição. O autor *representativo* representa o contribuidor que é o autor e o documentador da contribuição.

A quinta e última atividade tem o objetivo de identificar o *comprometimento do contribuidor com as contribuições*. Por exemplo, pode ajudar a entender e identificar a responsabilidade e a obrigação do contribuidor sobre a contribuição.

Em resumo, o trabalho de Gotel contribuiu para a melhor compreensão da importância do Pré-Rastreamento de requisitos na identificação das fontes de informação de um requisito na fase de produção e refinamento, na qual um conjunto de contribuidores (agentes) pode assumir diferentes papéis enquanto contribuem na elaboração de uma contribuição (artefato).

Na próxima seção apresentamos os sub-modelos de rastreamento de requisitos propostos por Ramesh.

3.5 Revisão do Trabalho de Pós-Especificação de Requisitos

De acordo com [RAM01], o objetivo do rastreamento de requisitos é assegurar a compatibilidade entre os requisitos dos *stakeholders* e os vários artefatos gerados no processo de desenvolvimento de um sistema e para que o rastreamento de requisitos seja útil deve-se organizá-lo de acordo com um modelo pré-definido.

No seu trabalho foram propostos e discutidos modelos de referência para o rastreamento de requisitos. Um dos objetivos de um modelo de referência é permitir a extração e adaptação dos elementos desses modelos para construção de um modelo de rastreamento para um projeto em domínio particular. Além dos modelos, o trabalho do autor inclui:

- Um estudo piloto visando produzir um meta-modelo;

- Análise de ferramentas de rastreamento de requisitos para identificar benefícios, falhas e suporte para o rastreamento;
- Um estudo principal sobre as atividades de rastreamento de requisitos realizado com 30 grupos envolvendo 26 organizações, com a participação de profissionais experientes de diferentes áreas-chave (Engenharia de software, Gerência de requisitos, etc) do desenvolvimento de software;
- Classificação dos usuários de rastreamento de requisitos em *low-end*¹ e *high-end*², uma extensão do trabalho publicado em [RAM98b];
- Uma discussão sobre a necessidade de melhorar o significado dos relacionamentos empregados nos modelos de rastreamento, como também a interpretação das matrizes de rastreamento, isto é, qual é o significado atribuído ao símbolo “x” usado nas mesmas. Este último problema é uma consequência da falta de tipos de relacionamento para o rastreamento e da sua representação matricial.

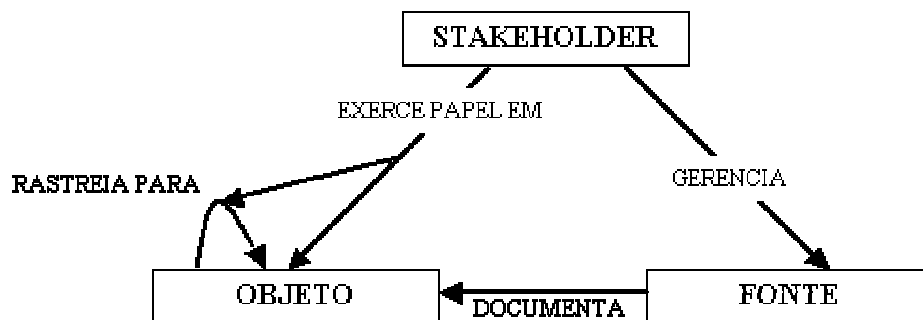


Figura 3-4 Meta modelo de rastreamento de requisitos proposto por Ramesh

A Figura 3-4 apresenta o meta-modelo de rastreamento de requisitos proposto por Ramesh. O meta-modelo pode ser utilizado para representar as seguintes dimensões de informações rastreáveis:

- Qual informação está representada?

¹ *Usuários low-end*: entendem o rastreamento de requisitos como uma obrigatoriedade imposta pelo *sponsor* do projeto ou por padrões.

² *Usuários high-end*: entendem o rastreamento de requisitos como uma grande oportunidade para a satisfação do cliente e geração de conhecimento através do ciclo de vida do sistema.

- ⇒ No modelo, **OBJETOS** representam as entradas e saídas do processo de desenvolvimento de software. Exemplos de **OBJETOS** são: REQUISITOS, PROJETO, COMPONENTES DO SISTEMA, DECISÕES, ALTERNATIVAS, etc. O rastreamento através dos objetos é representado pelo relacionamento *RASTREIA PARA*.
- Quais são os stakeholders que exercem diferentes papéis na criação, manutenção e uso dos diferentes objetos e relacionamentos entre eles?
 - ⇒ No modelo, **STAKEHOLDERS** representam os agentes envolvidos no desenvolvimento do software e nas atividades de manutenção. Gerentes de projeto, analistas de sistemas e designers são exemplos de **STAKEHOLDERS** que exercem diferentes *PAPÉIS* no estabelecimento e uso dos vários **OBJETOS** e *links* de rastreamento.
- Onde está representada a informação em relação às fontes que as documentam?
 - ⇒ Todos os **OBJETOS** são documentados por **FONTES**, que podem ser físicas, como documentos ou algo intangível, como referências para pessoas ou políticas e procedimentos não documentados. Exemplos de **FONTES** são: DOCUMENTO DE ESPECIFICAÇÃO DE REQUISITOS, ATA DE REUNIÃO, DOCUMENTO DE DESIGN, TELEFONEMAS, etc. **STAKEHOLDERS** gerenciam as **FONTES**, isto é, criam, mantêm e as usam.
- Como a informação está representada e como ela se relaciona com outros componentes de rastreamento?
 - ⇒ Como as **FONTES** podem ser físicas ou intangíveis, elas podem ser documentadas em níveis diferentes de formalidade. Algumas fontes, como uma especificação de requisitos, são representadas através de documentos, enquanto informações de projeto podem ser representadas, por exemplo, em gráficos.
- Porque um determinado objeto foi criado, modificado ou evoluiu?
 - ⇒ O raciocínio por detrás da criação, modificação e evolução dos **OBJETOS** pode ser representado como uma especialização da meta classe **OBJETO**.

- Quando a informação foi capturada, modifica e evoluída?
- ⇒ Informações temporais relevantes sobre qualquer uma das entidades ou *links* do modelo podem ser representadas como seus atributos. Por exemplo, a frequência ou duração com a qual um requisito ou projeto foi criado, revisado ou modificado pode ser representada dessa maneira.

Ramesh identificou que os usuários *low-end* criam *links* de rastreamento para modelar dependências entre requisitos, alocação dos requisitos aos componentes do sistema, verificação de conformidade e controle de mudança. Por sua vez os usuários *high-end* enxergam o rastreamento de requisitos como uma oportunidade de satisfação do cliente do sistema e como criação de conhecimento através do ciclo de vida de desenvolvimento do sistema. Eles percebem como a importância do rastreamento cresce à medida que cresce a complexidade do sistema, por conseguinte gerenciam esquemas de rastreamento muito mais ricos. Para melhor compreensão, o modelo para usuários *high-end* foi dividido em quatro partes: Gerenciamento de Requisitos, Alocação de Projeto, Verificação de Conformidade e Gerenciamento de Raciocínio.

A Figura 3-5 apresenta o sub-modelo de Gerenciamento de Requisitos. A seguir fazemos uma breve descrição desse sub-modelo. Sistemas são construídos primariamente para satisfazer **NECESSIDADES_ORGANIZACIONAIS**. Essas necessidades podem ser **NECESSIDADES ESTRATÉGICAS** ou **NECESSIDADES OPERACIONAIS** como denotado pelo relacionamento *É-UM* (representado com uma seta na cor cinza). Essas duas necessidades (operacional e estratégica) são freqüentemente detalhadas em **CENÁRIOS** descrevendo o uso esperado do sistema.

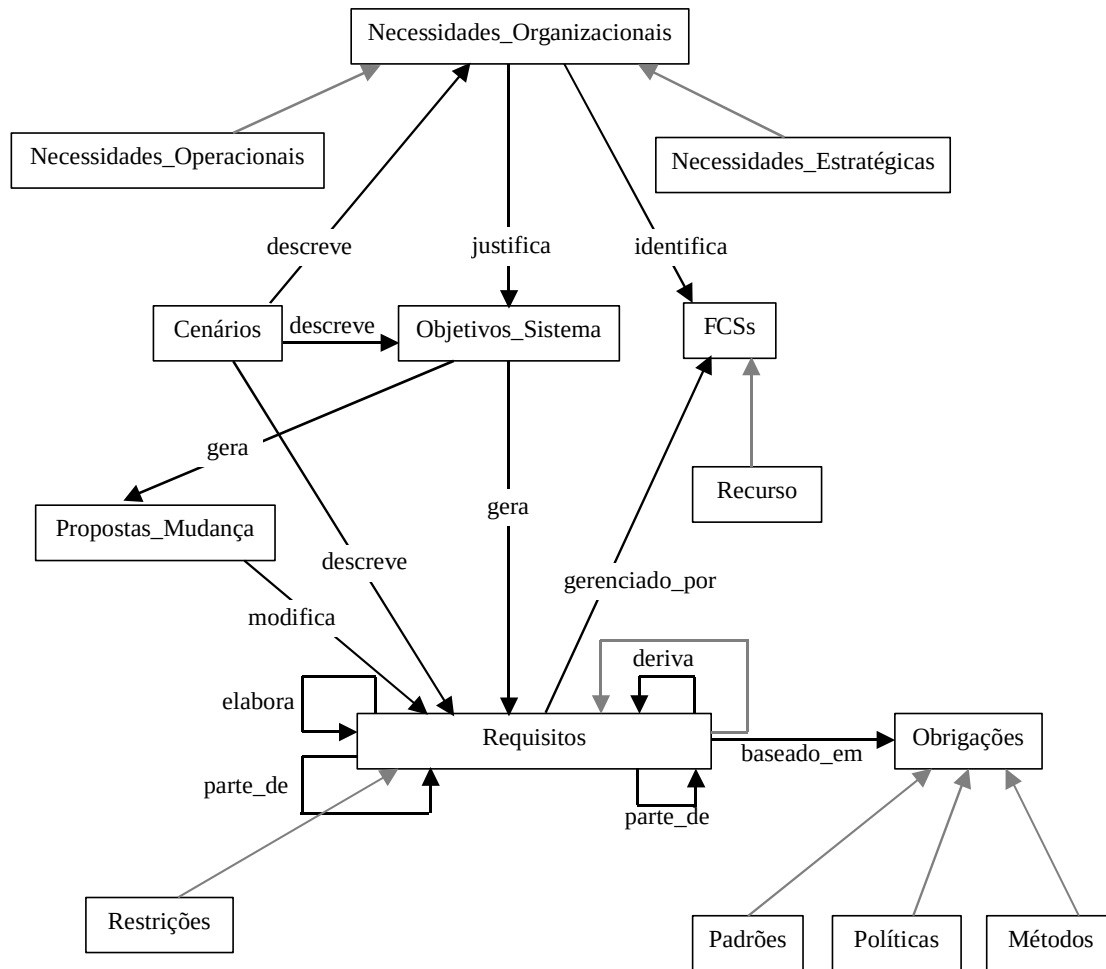


Figura 3-5 Sub-modelo de Gerenciamento de Requisitos (Ramesh)

Os objetivos que o sistema tenta alcançar, **OBJETIVOS_SISTEMA**, devem ser *JUSTIFICADOS* pelas **NECESSIDADES_ORGANIZACIONAIS**. *Stakeholders* como o cliente, gerente do projeto, patrocinador do projeto, etc., especificam os **OBJETIVOS_SISTEMA**. Os requisitos para o sistema são gerados a partir destes **OBJETIVOS_SISTEMA**. Os *stakeholders* envolvidos na identificação das **NECESSIDADES_ORGANIZACIONAIS** também identificam Fatores Críticos de Sucesso (**FCSs**). **RECURSOS** como custo e tempo são exemplos de **FCSs**. **REQUISITOS** do sistema são gerenciados por estes **CSFs**. **REQUISITOS** podem ser baseados em **PADRÕES**, **POLÍTICAS** e **MÉTODOS**. Manter um *link* entre os requisitos e suas fontes ajudará a compreender e interpretar de uma maneira mais adequada esses requisitos.

O sub-modelo de raciocínio é apresentado na Figura 3-6. A especificação, elaboração, decomposição, derivação e modificação de objetos como requisitos e projetos geram **INTERESSES_OU_CONFLITOS** (freqüentemente devido a diferentes interpretações, suposições, interesses, pontos de vista e objetivos dos *stakeholders*). Informações sobre como **DECISÕES** são tomadas, para resolver esses interesses, devem ser mantidas para certificar que os requisitos do

cliente foram compreendidos e satisfeitos. Estas **DECISÕES** podem afetar **REQUISITOS**, **COMPONENTES** e **PROJETOS**.

As **ALTERNATIVAS** que direcionam a resolução dos interesses são avaliadas e selecionadas de acordo com os **ARGUMENTOS** que as suportam ou se opõem a elas. As **DECISÕES** sobre a **SELEÇÃO** de uma ou mais **ALTERNATIVAS** são freqüentemente **INFLUENCIADAS** pelos **FCSs**.

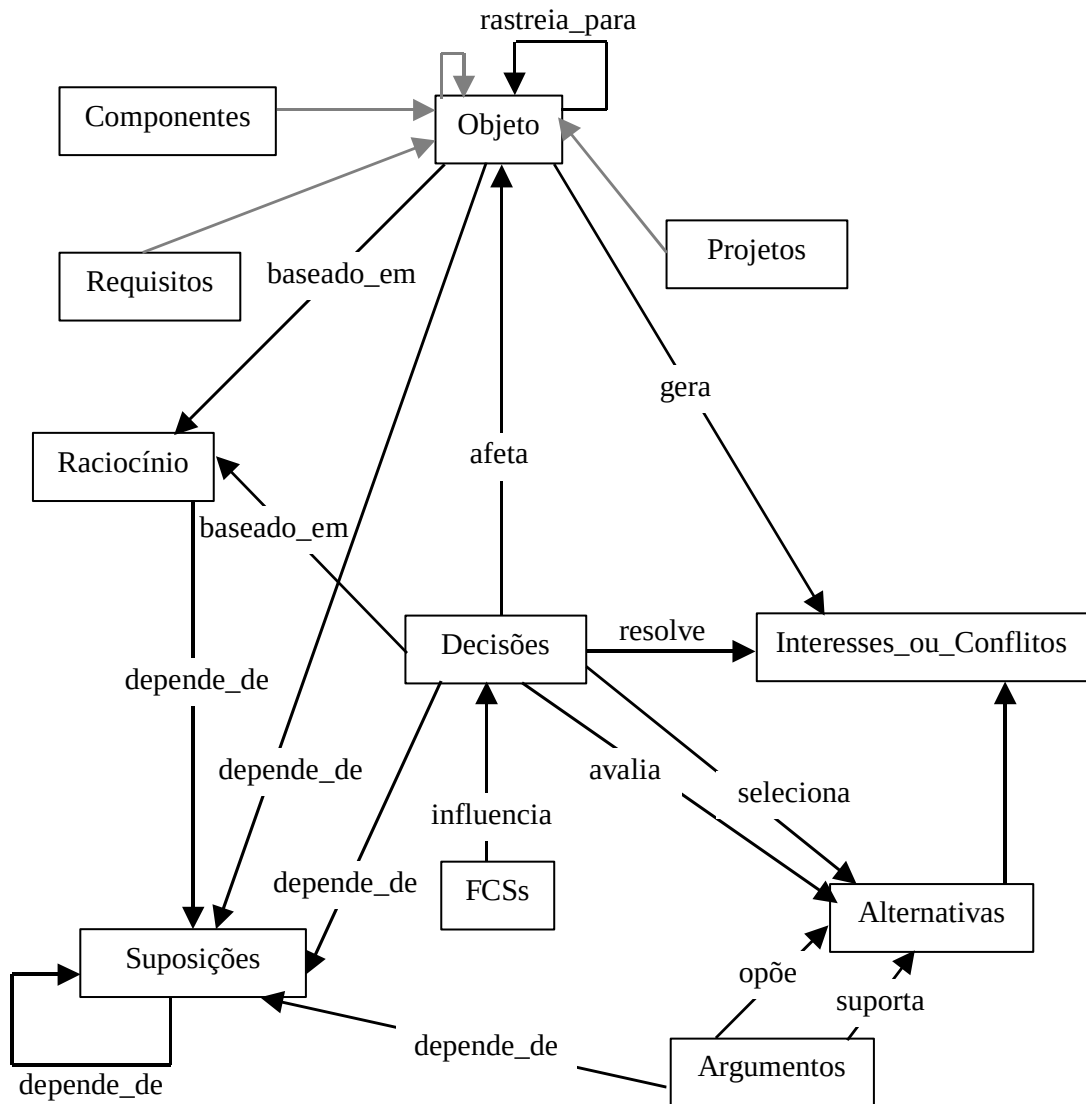


Figura 3-6 Sub-modelo de Raciocínio (Ramesh)

A Figura 3-7 apresenta o sub-modelo de Projeto. A seguir fazemos uma breve descrição desse sub-modelo. Para Ramesh o termo projeto refere-se a qualquer atividade que gera artefatos, incluindo implementação. **REQUISITOS** direcionam o **PROJETO**, que por sua vez é **BASEADO EM**

OBRIGAÇÕES como **PADRÕES**, **POLÍTICAS** e **MÉTODOS** que devem ser seguidos no desenvolvimento do sistema.

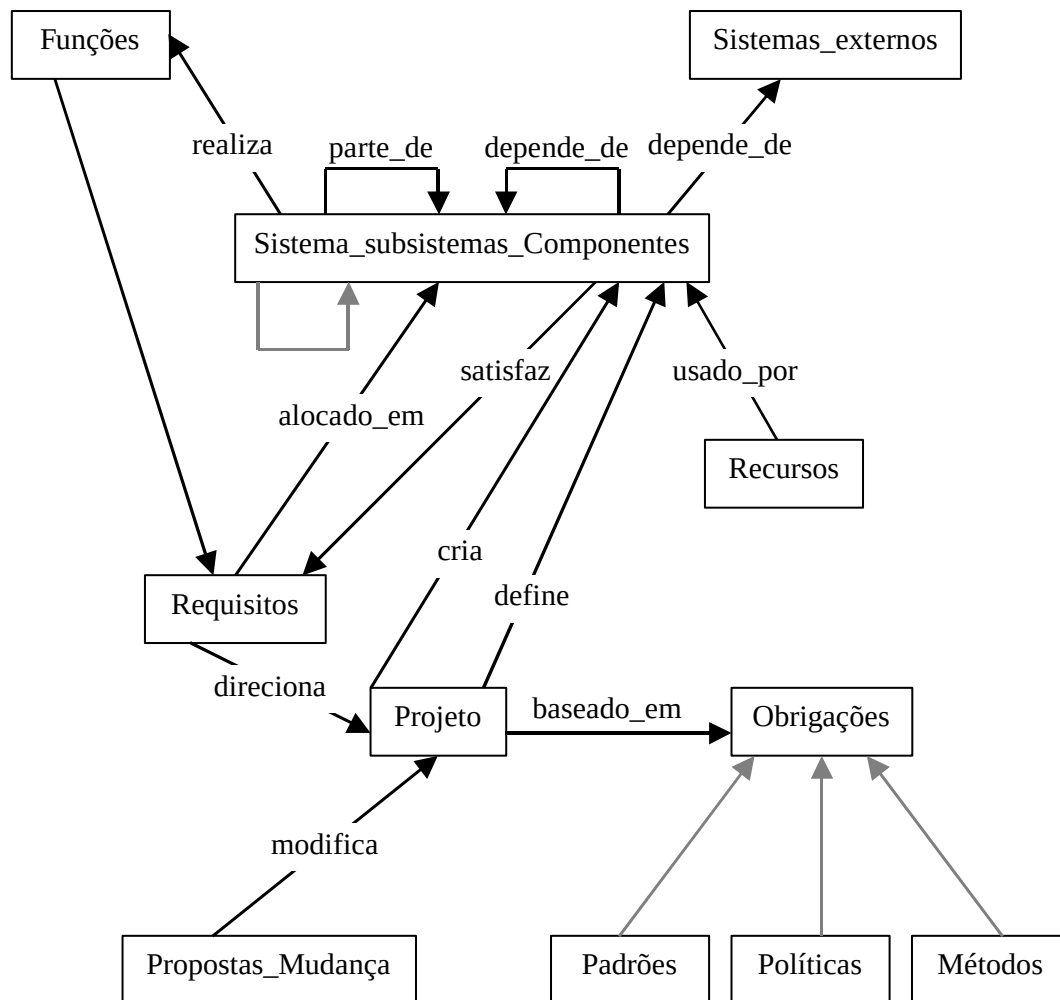


Figura 3-7 Sub-modelo de Projeto (Ramesh)

SISTEMA_SUBSISTEMAS_COMPONENTES são os blocos do sistema que são *DEFINIDOS* ou criados pelo PROJETO do sistema. REQUISITOS são *ALOCADOS EM* COMPONENTES. COMPONENTES DEPENDEM DE outros COMPONENTES, pois a funcionalidade ou performance de um pode depender de outros e RECURSOS são USADOS POR COMPONENTES.

No estudo realizado, Ramesh [RAM98a] identificou a necessidade de armazenar as **FUNÇÕES REALIZADAS** pelos **COMPONENTES**. Essas **FUNÇÕES** são rastreadas para os **REQUISITOS** funcionais explicitamente identificados no documento de requisitos. Os **COMPONENTES DEPENDEM DE SISTEMAS_EXTERNOS** com os quais farão interface.

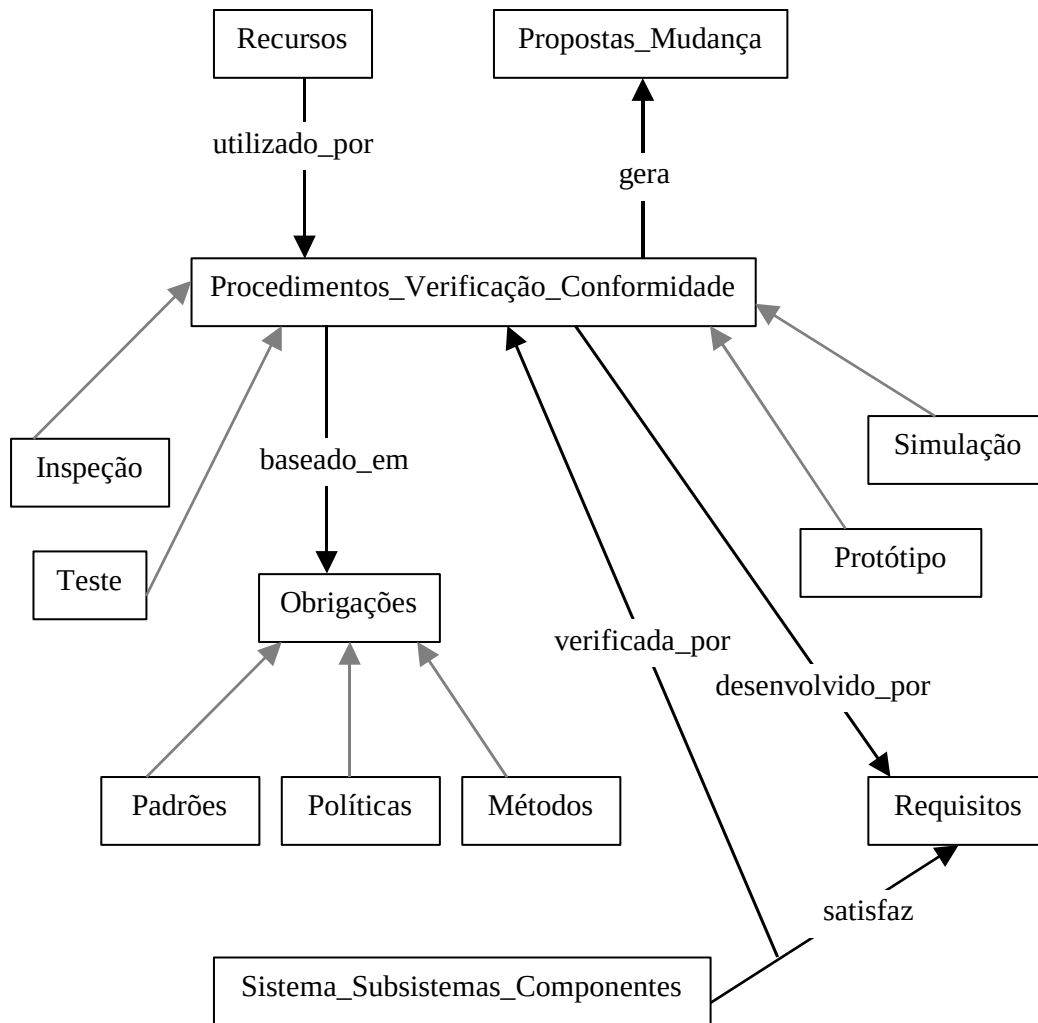


Figura 3-8 Sub-modelo de Verificação de Conformidade

A Figura 3-8 apresenta o sub-modelo de verificação e conformidade. Procedimentos de verificação e conformidade são desenvolvidos para garantir que cada um dos requisitos foi adequadamente desenvolvido no sistema. Exemplos típicos de **PROCEDIMENTOS_VERIFICAÇÃO_CONFORMIDADE** (PVCs) são os **PROTÓTIPOS**, **TESTES**, **INSPEÇÃO** (ões) e **SIMULAÇÃO** (ões). Para o desenvolvimento dos **PVCs** são utilizados **RECURSOS** (recursos humanos, financeiros, etc.). **OBRIGAÇÕES** são a base para os **PVCs** determinando como eles serão realizados.

De fato, os **PVCs** produzem resultados que *VERIFICAM* como os **SISTEMA_SUBSISTEMAS_COMPONENTES** *SATISFAZEM* os **REQUISITOS**, como também ajudam a *GERAR* **PROPOSTAS_MUDANÇA** para os **REQUISITOS**.

A seguir apresentaremos uma revisão da pesquisa do rastreamento de requisitos por Toranzo ([TOR99], [TOR02a], [TOR02b]).

3.6 Revisão do Trabalho de Toranzo

O trabalho de Toranzo apresenta um *framework* de uso geral que suporta o rastreamento de requisitos ([TOR99], [TOR02a], [TOR02b]). No seu trabalho, ele define o rastreamento de requisitos como a habilidade de seguir a vida de um requisito para frente e para trás, em um contexto de quatro camadas paralelas e inter-relacionadas (seção 2.1)

Toranzo propõe um processo para o desenvolvimento de um modelo de rastreamento composto por três fases:

- Coleta de informações: engloba as atividades de identificação de informações das quatro camadas citadas anteriormente;
- Estruturação e construção do modelo de rastreamento: relacionada com a organização das classes e relacionamentos;
- Definição e preenchimento das matrizes de rastreamento.

O meta-modelo proposto por Toranzo está fundamentado no MOF (Meta Object Facility [OMG00]), utilizado para descrever e definir meta-dado (dado acerca de dado). Para isso, MOF usa uma arquitetura de meta-dado em quatro camadas: meta-meta-modelo, meta-modelo, modelo e informação. O meta-modelo é apresentado na Figura 3-9. A meta-classe base do meta-modelo é *Elemento* que é especializado nas meta-classes *ElementoGeneralizável* e *Relacionamento*.

Quando da instanciação do meta-modelo, o *ElementoGeneralizável* pode classificar as instâncias da meta-classe *Classe* de duas maneiras: em subclasse, se a especialização for usada, ou como superclasse, se generalização for usada. Dois *ElementoGeneralizáveis* se relacionam através da *Generalização* e *Associação*, ambas são subclasses de *Relacionamento*.

Os relacionamentos utilizados para rastrear informações são: *Satisfação*, *Recurso*, *Responsabilidade*, *Representação*, *Alocado*, *Agregação* e *Generalização*. Algumas associações possuem adornos, por exemplo, uma expressão “[D/O]” na qual a letra “D” captura o grau de dependência de uma classe em relação a outra (A=alta, M=média e B=baixa) e a letra “O” expressa o grau de obrigação (A=obrigatório, O=opcional), isto é, se todas ou algumas das instâncias da qual depende uma classe devem ser satisfeitas ou alcançadas. Por exemplo, um relacionamento de dependência pode expressar que a satisfação de um requisito organizacional tem um alto grau de

dependência com vários outros requisitos e que todos ou alguns requisitos devem ser satisfeitos para que a satisfação do objetivo organizacional seja alcançada

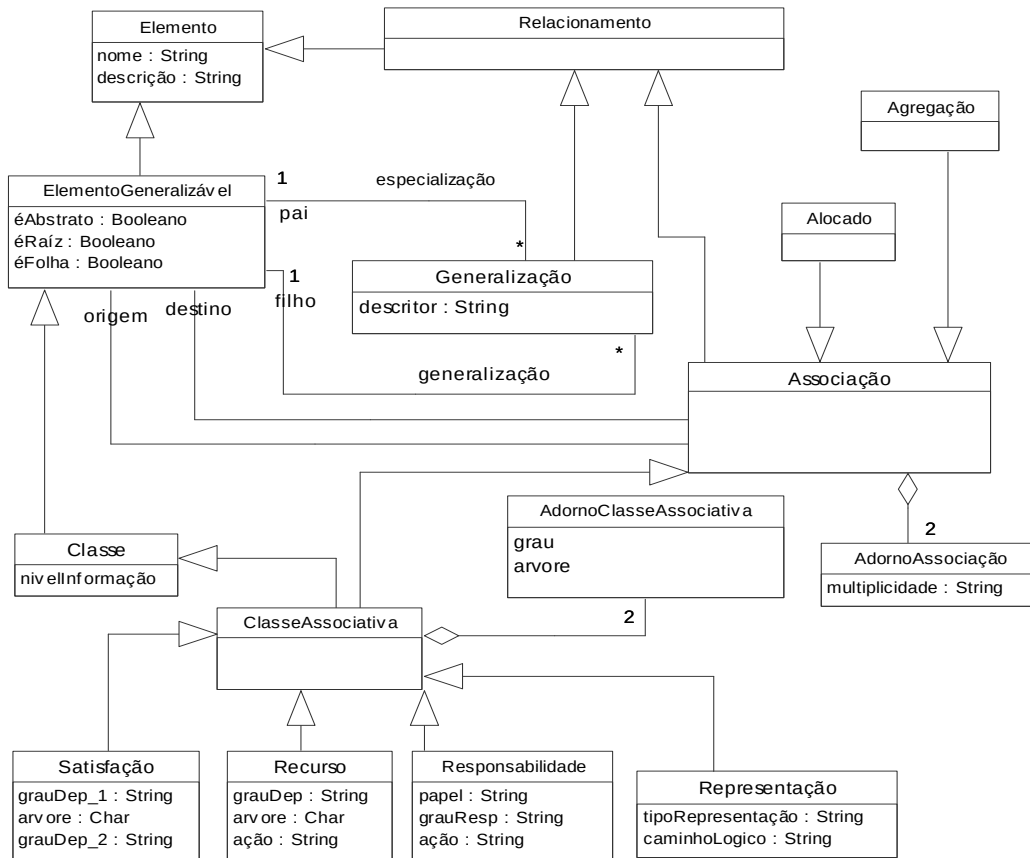


Figura 3-9 Meta-modelo proposto por Toranzo

O modelo de rastreamento intermediário¹ proposto por Toranzo [TOR02a] é apresentado na Figura 3-10. Esse modelo é resultado de uma combinação de fatores, tais como: boas práticas, estudo de caso, abstração, aplicação do meta-modelo proposto e dos segmentos de informação.

A raiz da estrutura hierárquica é a classe *Informação* e algumas das suas subclasses são: *InformaçãoExterna*, *InformaçãoOrganizacional*, *ObjetivoSistema*, *Tarefa*, *Requisito*, *Diagrama*, *Programa* e *Subsistema*.

A classe *InformaçãoExterna* é uma abstração de todos os possíveis tipos de informações do segmento de informação externo. A classe *InformaçãoOrganizacional* é uma abstração das informações do segmento de informação organizacional. As classes *Tarefa* e *ObjetivoSistema*

¹ Modelo de rastreamento intermediário: Modelo que possui todas as possíveis classes e relacionamentos existentes no meta-modelo.

representam algumas das informações do segmento de informação gerencial. As classes *Requisito*, *Diagrama*, *Programa* e *Subsistema* representam algumas das informações do segmento de informação de desenvolvimento.

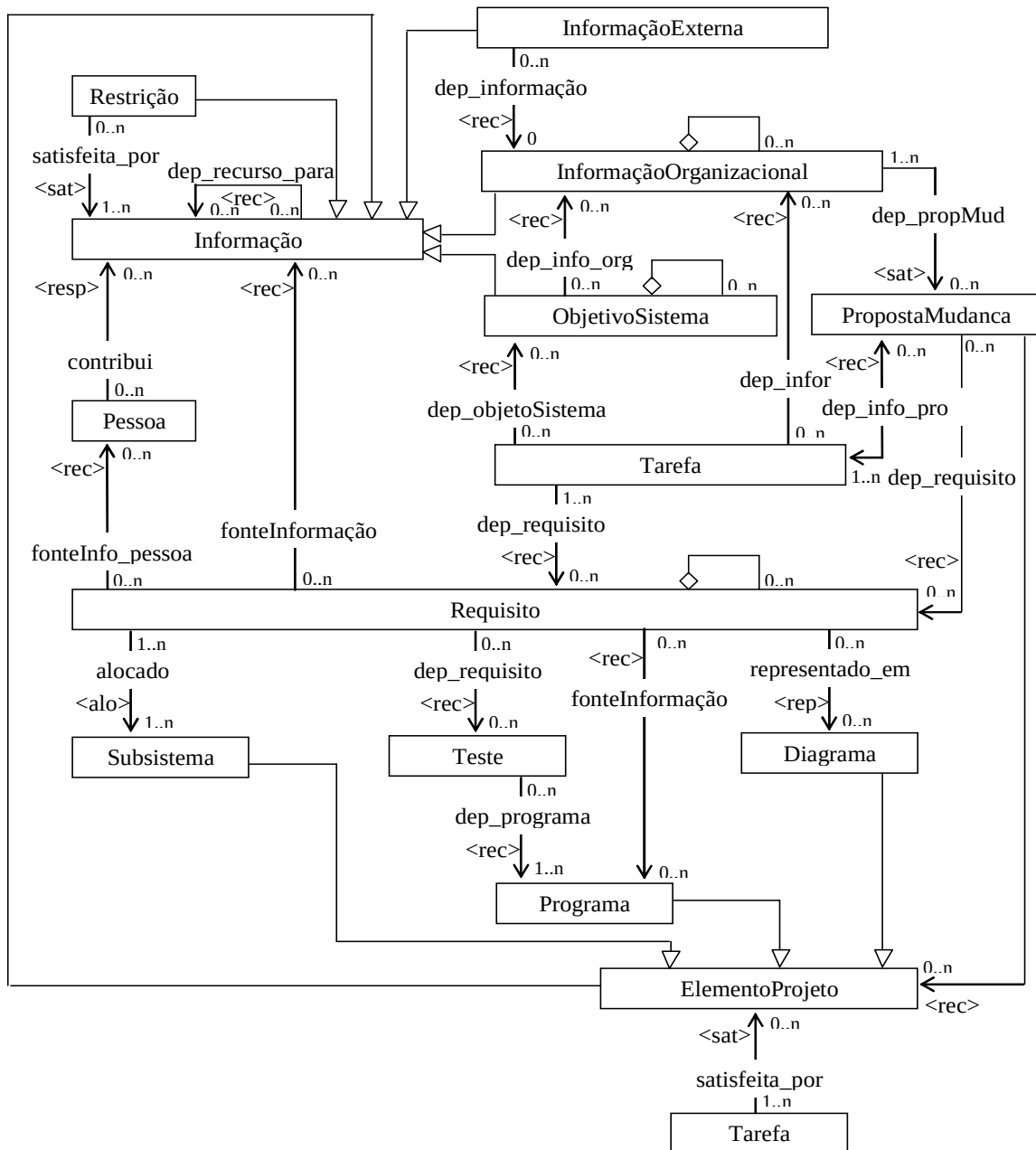


Figura-3-10 Modelo de rastreamento intermediário proposto por Toranzo

O relacionamento recursivo *dep_recurso_para* sobre a classe *Informação* é herdado pelas suas subclasses. Conseqüentemente, em um modelo de rastreamento de um projeto é possível expressar diretamente uma relação de dependência de recurso entre as subclasses da classe

Informação. Por exemplo, os objetivos, estratégias e/ou regras de uma organização podem ser recursos de informação para criar os objetivos de um sistema.

O relacionamento *dep_informação* (de *InformaçãoOrganizacional* para *InformaçãoExterna*) é uma abstração das possíveis dependências de recurso das informações do segmento de informação organizacional com as do segmento de informação externo.

O relacionamento *satisfeita_por* (de *Restrição* para *Informação*) expressa que as instâncias da classe *Restrição* devem ser satisfeitas por algumas instâncias das subclasses de *Informação*. O importante disso é que devemos considerar as restrições impostas ao projeto de software.

O relacionamento *contribui* (de *Pessoa* para *Informação*) expressa que pessoas contribuem ou são responsáveis pelas instâncias das subclasses de *Informação*. Por exemplo, pessoas são responsáveis pelo desenvolvimento de diagramas, programas, requisitos, etc.

Nossa proposta de extensão da metodologia de desenvolvimento orientada a agentes TROPOS, está baseada no processo de Toranzo [TOR02a], um *framework* elaborado para ser utilizado em qualquer paradigma de desenvolvimento de software, aborda diversos tipos de informações: organizacional, externo, informações gerenciais e de desenvolvimento e define um processo para o desenvolvimento de um modelo de rastreamento levando em consideração as particularidades do projeto, modelo esse que pode ser baseado no modelo intermediário de rastreamento proposto no *framework*.

3.7 Considerações Finais

Neste capítulo apresentamos a atividade de Rastreamento de Requisitos, conceituando-a e destacando sua importância no processo de desenvolvimento de software. Descrevemos os tipos de rastreamento: “para frente” e “para trás”, pré-especificação de requisitos e pós-especificação de requisitos, extra-rastreamento e intra-rastreamento de requisitos.

Alguns obstáculos dificultam o progresso do Rastreamento de Requisitos como: falta de uma definição padronizada, falta de reconhecimento e comprometimento organizacional, falta de orientações acerca das informações que devem fazer parte de um modelo de rastreamento, má compreensão da atividade e dificuldade de aplicá-la corretamente, falta de uma cultura organizacional no uso das boas práticas de Engenharia de Requisitos, bem como a carência de modelos de rastreamento de requisitos para orientar a elaboração de um projeto.

O trabalho de Gotel e Finkelstein recomenda a divisão do rastreamento de requisitos tradicional em dois tipos: Pré-Especificação de Requisitos e Pós-Especificação de Requisitos, com

ênfase na pesquisa e maior suporte computacional a Pré-Especificação de Requisitos que é a habilidade de rastrear os requisitos para suas fontes originais.

Ramesh propõe sub-modelos de referência para o rastreamento de requisitos: sub-modelo de Gerenciamento de Requisitos, sub-modelo de Raciocínio e sub-modelo de Projeto e divide os usuários de rastreamento de requisitos em *low end* e *high end*.

Na nossa proposta de dissertação vamos utilizar o processo para desenvolvimento de um modelo de rastreamento proposto por Toranzo para extensão da metodologia orientada a agentes Tropos. Esse processo é composto por três fases: coleta de informações, estruturação e construção do modelo de rastreamento e definição e preenchimento das matrizes de rastreamento. O trabalho de Toranzo enumera um conjunto de diretrizes para o desenvolvimento desse processo, levando em consideração sua proposta de segmentos de informação, modelo de referência para o rastreamento de requisitos e relacionamentos entre os elementos desse modelo. Esse processo será detalhado no capítulo 7.

No próximo capítulo descreveremos a Engenharia de Software Orientada a Agentes e algumas metodologias de desenvolvimento de software orientadas a agentes. A metodologia Tropos utilizada nessa dissertação será abordada no próximo capítulo.

Capítulo 4 – Engenharia de Software Orientada a Agentes

Este capítulo aborda a Engenharia de Software Orientada a Agentes. Inicialmente são apresentadas as motivações para o uso desse novo paradigma, seguido dos aspectos e propriedades que caracterizam um agente. São abordadas as motivações e a importância do paradigma orientado a agentes para a Engenharia de Software. Além disso, são descritas as principais áreas de aplicação dos softwares orientados a agentes, bem como os desafios e obstáculos enfrentados atualmente no desenvolvimento desses softwares. O capítulo é concluído com a apresentação da metodologia orientada a agentes utilizada nessa dissertação, Tropos.

4.1 Introdução

Projetar e construir software industrial de alta qualidade é difícil. De fato, assume-se que a tarefa de desenvolver tais projetos está entre as mais complexas tarefas de construção realizadas por humanos. Diante disso, vários paradigmas de implementação foram propostos na literatura (ex., programação estruturada, programação declarativa, programação orientada a objetos e baseada em componentes). Esta evolução dos paradigmas visa tornar a engenharia de software mais gerenciável bem como permite aumentar a complexidade das aplicações que podem ser construídas.

As empresas de desenvolvimento de sistemas de grande porte vêm apresentando grandes problemas e desejam produzir software de qualidade de acordo com [SIL03]. Na visão do usuário, software de qualidade é aquele que além de satisfazer as suas necessidades, é feito no custo e prazo acordados. Já na visão do projetista de software, um bom software é aquele que apresenta as seguintes características:

- Maior flexibilidade: Possibilita satisfazer novos requisitos de negócio (funcionalidades) de forma fácil e rápida.
- Melhor adaptabilidade: Possibilita configurar uma aplicação para vários usuários, usando várias alternativas para oferecer os serviços da aplicação com o mínimo de impacto no seu núcleo.
- Melhor manutenibilidade: Possibilita alterar partes de uma aplicação, de modo que as outras partes sofram um impacto mínimo.
- Melhor reusabilidade: Possibilita construir aplicações únicas e dinâmicas rapidamente.
- Melhor aproveitamento do legado: Possibilita reusar a funcionalidade de sistemas legados em novas aplicações.
- Melhor interoperabilidade: Possibilita que aplicações que executam em plataformas diferentes troquem informações.

- Melhor escalabilidade: Possibilita distribuir e configurar a execução da aplicação de modo a gerenciar grandes volumes de transação.
- Maior robustez: Possibilita implementar soluções de software com menos defeitos.

Nos últimos anos os estudiosos vêm pesquisando o paradigma de agentes, de forma a melhorar significativamente o processo de desenvolvimento de software. Técnicas orientadas a agentes representam um novo meio de analisar, projetar e construir sistemas de software complexos. O uso da abordagem orientada a agente é baseada nos seguintes argumentos [JEN03a]: (i) o aparato conceitual de sistemas orientados a agentes é adequado para construir soluções de software para sistemas complexos e (ii) as abordagens orientadas a agentes representam um verdadeiro avanço sobre o atual estado da arte na engenharia de sistemas complexos.

De fato, técnicas orientadas a agentes são adequadas para desenvolver sistemas complexos de software porque:

- A orientação a agentes permite decompor de maneira efetiva o escopo do problema em um sistema complexo;
- As abstrações chave presentes na modelagem orientada a agentes são um meio natural de modelar sistemas complexos;
- O paradigma orientado a agente identifica e gerencia relacionamentos organizacionais sendo então apropriado para lidar com as dependências e interações que existem em um sistema complexo.

Na próxima seção apresentaremos os principais conceitos relacionados ao paradigma orientado a agentes.

4.2 Características de agentes

Não há uma definição padrão para agente. Russel e Norvig [RUS03] definem agente como sendo uma entidade que percebe seu ambiente através de sensores e atua no ambiente através de atuadores. Nesta dissertação direcionamos a conceituação do que seriam agentes de software para a seguinte definição [WOO01a]:

“Um agente é um sistema computacional encapsulado que está situado em algum ambiente e é capaz de ação flexível autônoma neste ambiente, a fim de alcançar os seus objetivos de projeto”.

Agentes possuem as seguintes propriedades entre outras ([WOO95a], [MUL99], [STU03]):

- Autonomia – habilidade do software de agir independentemente sem intervenção direta humana ou de outros agentes [YU02].
- Proatividade – agentes não agem simplesmente em resposta ao seu ambiente, eles são capazes de exibir proatividade, comportamento dirigido a objetivo e tomar a iniciativa quando necessário [JEN00].
- Sociabilidade – habilidade de participar de muitos relacionamentos, interagindo com outros agentes ao mesmo tempo ou em tempos diferentes [YU02].

Na próxima seção apresentamos os tipos de sistemas baseados em agentes encontrados hoje na engenharia de software orientada a agentes.

4.3 Tipos de Sistemas Baseados em Agentes

Um sistema baseado em agentes pode conter um ou mais agentes. Existem casos em que um único agente é apropriado. Um bom exemplo é a classe de sistemas conhecida como assistentes especialistas, na qual um agente age como um especialista que auxilia um usuário de computador na execução de alguma tarefa. Entretanto, quando adotamos uma visão orientada a agentes do mundo, torna-se claro que um único agente é insuficiente [JEN00]. O caso multi-agentes – onde o sistema é projetado e implementado com vários agentes interagindo entre si – é mais geral e mais interessante do ponto de vista da engenharia de software.

A pesquisa em sistemas multi-agentes se preocupa com o comportamento de uma coleção de agentes autônomos, possivelmente pré-existentes, visando resolver um dado problema.

Os sistemas multi-agentes são apropriados para representar problemas que têm vários métodos de resolução, múltiplas perspectivas e / ou múltiplas entidades de resolução. Tais sistemas têm as vantagens tradicionais da resolução de problemas concorrentes e distribuídos, mas incluem as vantagens dos padrões de interações sofisticados. Por exemplo, agentes podem se organizar para realizar tarefas não previstas em tempo de projeto.

Um sistema multi-agentes pode ser definido como uma rede fracamente acoplada de solucionadores de problema que trabalham juntos para resolver problemas que estão além de suas capacidades ou conhecimentos individuais. Estes solucionadores de problemas – agentes - são autônomos e podem ser heterogêneos por natureza. Um sistema multi-agentes inclui as seguintes características [JEN00]:

- Cada agente tem informações, recursos ou capacidades insuficientes para solucionar um dado problema. Dessa forma, cada agente tem um ponto de vista limitado;
- A informação é descentralizada;
- A computação e a interação são assíncronas.

Na maioria dos casos, agentes agem para atingir objetivos em benefício de indivíduos ou de empresas. Assim, quando agentes interagem, há tipicamente algum contexto organizacional em destaque. Este contexto ajuda a definir a natureza dos relacionamentos entre os agentes.

Para suportar sistemas multi-agentes, um ambiente apropriado precisa ser estabelecido, devendo [ODE00a]:

- Fornecer uma infra-estrutura de comunicação específica e protocolos de interação;
- Ser tipicamente aberto e não possuir um projeto centralizado ou uma função de controle *top-down*;
- Conter agentes que sejam autônomos e adaptativos.

Na próxima seção destacaremos a importância e a motivação dos sistemas multi-agentes.

4.4 Importância dos Sistemas Multi-Agentes na Engenharia de Software

Uma questão óbvia a perguntar é por que sistemas multi-agentes são vistos como um novo direcionamento importante na engenharia de software. Agentes são importantes / úteis porque [ODE99]:

- Fornecem uma boa maneira de pensar sobre o fluxo de controle em um sistema altamente distribuído;
- Oferecem um mecanismo que permite um comportamento emergente ao invés de uma arquitetura estática;
- Codificam melhores práticas de como organizar entidades colaborativas concorrentes.

De fato, existem várias razões que justificam o aumento de interesse na pesquisa de sistemas multi-agentes ([ODE00a], [WOO01a]):

- Distribuição de dado e controle: Para muitos sistemas de software o controle global é distribuído em vários nós de computação freqüentemente distribuídos geograficamente. A fim de fazer com que esses sistemas trabalhem efetivamente, esses nós devem ser capazes de interagir autonomamente uns com os outros.
- Sistemas abertos: Muitos sistemas são abertos no sentido de que não é possível saber, em tempo de projeto, quais componentes farão parte exatamente do sistema, nem como esses componentes interagem entre si. Para operarem efetivamente, estes sistemas devem ser capazes de tomar decisão autônoma flexível.
- Dividir funcionalidade entre muitos agentes permite que a aplicação seja modular, flexível, modificável e extensível.
- Conhecimento especializado não é freqüentemente disponível em um único agente (onisciência). O conhecimento que é distribuído em várias fontes (agentes) pode ser integrado para uma maior completude quando for necessária flexibilidade.

Observa-se que sistemas orientados a agentes estão sendo cada vez mais usados na indústria como, por exemplo, em aplicações voltadas para telecomunicações e comércio eletrônico. Na próxima seção abordaremos as principais áreas onde os sistemas multi agentes estão presentes atualmente.

4.5 Áreas de Aplicação

A tecnologia de agentes está ultrapassando rapidamente os limites das universidades e os laboratórios de pesquisa e está começando a ser usada para resolver problemas do mundo-real em uma escala de aplicações industriais e comerciais. Aplicações em uso existem hoje e novos sistemas estão sendo desenvolvidos cada vez mais. Nesta seção nós identificamos as principais áreas onde as abordagens baseadas em agente estão sendo usadas e fornecemos exemplos de sistemas nestas áreas [AGE03]:

- **Aplicações industriais** – Agentes estão sendo aplicados em uma diversidade de sistemas industriais, entre eles:
 - ◇ Fabricação: Sistemas nessa área incluem projeto de configuração de produtos de fabricação, projeto colaborativo, cronograma e controle de operações de fabricação, controle de fabricação de um robô e determinação de seqüências de produção para uma fábrica [JEN03b];
 - ◇ Telecomunicações: Controle de rede, transmissão e chaveamento, gerência de serviço e gerência de rede;
 - ◇ Controle de Tráfego Aéreo: Agentes são usados para representar tanto os equipamentos de aviação quanto vários sistemas de controle de tráfego aéreo em operação.
 - ◇ Sistemas de transporte: O domínio de gerência de tráfego e transporte é adequado para uma abordagem baseada em agentes devido a sua natureza geograficamente distribuída.

O melhor exemplo até a data atual de sistemas desenvolvidos para aplicações industriais utilizaram-se da plataforma de software ARCHON para a construção de sistemas multi-agentes que vêm sendo aplicados em muitos domínios, incluindo transporte de eletricidade e controle de acelerador de partículas [LUC03].

- **Aplicações Comerciais** – Enquanto aplicações industriais tendem a serem altamente complexas, indicando sistemas que operam em áreas de nicho comparativamente

pequeno, as aplicações comerciais, especialmente aquelas preocupadas com gerência da informação, tendem a serem utilizadas por uma fatia maior do mercado.

- ◇ Gerência de informação: Mecanismos de busca são realizados por agentes, que agem autonomamente para buscar na *web* em benefício de algum usuário. De fato esta é provavelmente uma das áreas mais ativas para aplicações de agente. Como exemplos, um assistente pessoal que aprende sobre os interesses do usuário e com base neles compila um jornal pessoal, um agente assistente para automatizar várias tarefas de usuário em um *desktop* de computador, um agente de procura de *home page*, um assistente de navegação na *web* e um agente especialista de localização.
- ◇ Comércio eletrônico: Algumas tomadas de decisão comerciais podem ser colocadas nas mãos de agentes. Um aumento da quantidade de comércio está sendo empreendido por agentes. Entre estas aplicações comerciais encontra-se um agente que descobre os *Compact Discs* mais baratos, um assistente pessoal de compras capaz de buscar lojas on-line para disponibilizar o produto e informação sobre o preço, um mercado virtual para comércio eletrônico e vários catálogos interativos baseados em agentes.
- ◇ Gerência de processo de negócio: Organizações têm procurado desenvolver sistemas de Tecnologia da Informação para dar assistência a vários aspectos da gerência de processos de negócio. Outras aplicações nesta área incluem um sistema de gerência de cadeias fornecedoras, um sistema para gerência de fluxos de trabalho heterogêneos e um sistema baseado em agentes móveis para gerência de fluxo de trabalho interorganizacional.
- **Aplicações de entretenimento** – Agentes têm um papel óbvio nos jogos de computador, cinema interativo, e aplicações de realidade virtual: tais sistemas tendem a ser cheios de caracteres animados autônomos, que podem naturalmente ser implementados como agentes.
- **Aplicações médicas** – Recentes aplicações incluem a área de monitoração de pacientes e plano de saúde [JEN01].

Diante do uso crescente dessa nova tecnologia, alguns obstáculos e desafios foram encontrados no desenvolvimento de software orientado a agentes, os quais serão apresentados a seguir.

4.6 Desafios e Obstáculos do Desenvolvimento Orientado a Agentes

Depois de destacar os benefícios potenciais da engenharia de software orientada a agentes, apresentamos nesta seção desafios inerentes à construção de software usando esta nova tecnologia. Estes podem ser tanto específicos da tecnologia de agentes, como armadilhas que surgem no processo de desenvolvimento de sistemas multi-agentes (ou aquelas comuns a qualquer nova tecnologia). Primeiramente consideraremos os problemas diretamente atribuídos ao processo de desenvolvimento de software. Projetistas têm encontrado meios de contornar estes problemas, já que sistemas de agente robustos e confiáveis têm sido construídos.

O fato de agentes terem de agir para satisfazer os seus objetivos enquanto mantêm uma interação contínua com seu ambiente torna difícil projetar software capaz de manter um equilíbrio entre comportamento pró-ativo e reativo. Para atingir este equilíbrio, é necessário que a tomada de decisão seja sensível ao contexto, resultando num grau significativo de imprevisibilidade sobre quais objetivos o agente perseguirá, em quais circunstâncias e quais os métodos que serão usados para atingir os objetivos escolhidos [JEN00].

Como agentes são autônomos, os padrões e os efeitos de suas interações são imprevisíveis. Em resumo, tanto a natureza (uma requisição simples versus uma negociação prolongada) como a consequência de uma interação pode não estar determinada desde o início do projeto do sistema multi-agentes [JEN00].

Outro aspecto da imprevisibilidade no projeto de sistemas orientados a agentes refere-se à noção de comportamento emergente. Foi reconhecido que a interação de muitos agentes individuais resulta no fenômeno comportamental que não pode ser entendido de forma geral exclusivamente em termos do comportamento dos agentes individuais.

Além desses problemas específicos da tecnologia de agentes, identificamos vários desafios que ocorrem repetidamente no desenvolvimento de sistemas multi-agentes, além de outros que são comuns a qualquer nova tecnologia. Eles são classificados geralmente em cinco grupos [TVE01]: Política, Conceitual, Análise e Projeto, Nível de Agente e Nível de Sociedade. A seguir descreveremos esses grupos em detalhes:

- Política:

- ◇ Apresentar com entusiasmo excessivo as soluções de agente ou falhar em entender onde agentes podem ser aplicados adequadamente: a tecnologia de agentes representa uma inovação e uma importante maneira de conceituar e implementar software, mas é importante entender suas limitações.
- ◇ Não tomar agentes como sendo uma religião ou dogma: embora haja muitas aplicações usando agentes, eles não são uma solução universal. Há muitas aplicações para as quais os paradigmas convencionais de desenvolvimento de software (tal como programação orientada a objetos) são mais apropriados.
- Conceitual:
 - ◇ Esquecer que está desenvolvendo software: a ausência de técnicas testadas e confiáveis para dar assistência no desenvolvimento de software multi-agentes encoraja os projetistas a esquecerem que estão na verdade desenvolvendo software. Nesse caso, a falha do projeto não é por causa dos problemas específicos de agente, mas porque a boa prática da engenharia de software foi ignorada.
 - ◇ Esquecer que está desenvolvendo software com múltiplas linhas de controle (*multithreaded*): por natureza, sistemas multi-agentes tendem a ter múltiplas linhas de controle (tanto num agente quanto numa sociedade de agentes). Assim, na construção de software multi-agentes, é vital não ignorar as lições aprendidas da comunidade de sistemas concorrentes e sistemas distribuídos, pois os problemas inerentes aos sistemas com múltiplas linhas de controle não desaparecem apenas porque você adotou uma abordagem baseada em agentes.
- Análise e Projeto
 - ◇ Não saber porque necessita de uma solução orientada a agentes: pode não haver entendimento de como agentes podem ser usados para melhorar seus produtos nem como eles podem capacitá-los a gerar novas linhas de produto, etc.
 - ◇ Querer construir soluções genéricas para problemas exclusivos: tipicamente se manifesta no planejamento de uma arquitetura que supostamente possibilita que uma

classe completa de sistemas seja construída, quando o que realmente é solicitado é uma solução feita sob encomenda direcionada para um único problema.

- ◇ Acreditar que usar agentes é a única solução (*Silver Bullet*): a tecnologia de agentes é um paradigma de software novo, emergente e ainda não testado na sua essência. Portanto, ela não pode ser considerada uma solução mágica. Há bons argumentos que indicam que a tecnologia de agentes levará a melhorias no desenvolvimento de software distribuído complexo, mas estes argumentos ainda não foram quantitativamente comprovados [TVE01].
 - ◇ Criar sua própria arquitetura de agentes: quando se inicia um projeto multi-agentes, tende-se a imaginar que nenhuma arquitetura de agentes existente atende aos requisitos do seu problema. Portanto, há uma tentação para projetar uma arquitetura a partir do zero. Isto é freqüentemente um erro, pois deve-se estudar as várias arquiteturas descritas na literatura antes de construir uma própria.
- Nível de Agente
 - ◇ Os agentes usam excessiva Inteligência Artificial: ao se focar demais em aspectos de “inteligência”, o framework de construção de um agente fica muito sobrecarregado com as técnicas (e.g., processamento de linguagem natural, planejadores, provadores de teorema, sistemas de manutenção de razão, etc.). Para ser útil sugere-se construir agentes com um mínimo de técnicas da Inteligência Artificial. De fato, alguns pesquisadores chegam a afirmar que, dependendo da aplicação, somente 1% de inteligência artificial é necessário [JEN03a].
 - Nível de Sociedade:
 - ◇ O projeto não explora concorrência: se há a necessidade de uma única linha de controle num sistema, então a utilidade de uma solução baseada em agentes deve ser seriamente questionada.
 - ◇ Ver agentes em qualquer lugar: quando se adota este ponto de vista, a tendência é utilizar agentes para tudo. Porém, a sobrecarga gerada para gerenciar agentes e a

comunicação entre agentes pesará mais do que os benefícios de uma solução baseada em agentes.

- ◇ Ter pouquíssimos agentes: alguns projetistas criam um sistema que completamente falha em explorar o poder oferecido pelo paradigma de agente e desenvolvem uma solução com um número muito pequeno de agentes fazendo todo o trabalho. Tais soluções tendem a não atender o teste de coerência da engenharia de software, que requer que um módulo de software tenha uma única função coerente.

A seguir expomos os obstáculos chave que devem ser superados para que a engenharia de software orientada a agentes se torne popular ([WOO01b], [CAS02b]):

- Organização da relação entre agentes e outros paradigmas: não está claro como o desenvolvimento de sistemas multi-agentes irá coexistir com outros paradigmas de software, tal como o desenvolvimento orientado a objetos.
- Metodologias orientadas a agentes: embora várias metodologias preliminares de análise e projeto orientados a agentes estejam sendo propostas, há comparativamente pouco consenso entre elas e nenhum acordo sobre os tipos de conceitos que a metodologia deveria suportar ([LUC03], [SIL04]).
- Engenharia para sistemas abertos: precisa-se de um melhor entendimento de como fazer engenharia em sistemas abertos. Em tais sistemas, acredita-se que é essencial ser capaz de reagir a eventos imprevistos, explorar oportunidades onde estes eventos surgem, e conseguir dinamicamente acordos com componentes do sistema cuja presença não poderia ser prevista em tempo de projeto.
- Engenharia de escalabilidade (*scalability*): precisa-se de um melhor entendimento de como fazer engenharia de forma segura e previsível em sistemas que consistem de um grande número de agentes interagindo dinamicamente uns com os outros a fim de atingir suas metas.
- Rastreamento de Requisitos: a inclusão dessa atividade no processo de desenvolvimento orientado a agentes proporcionará muitos benefícios, pois não apenas permitirá um

melhor atendimento aos requisitos do cliente, como também apoiará a avaliação do impacto de mudanças solicitadas (mesmo antes das suas implementações no sistema). Como resultado, ter-se-á uma melhoria na qualidade do software em desenvolvimento em decorrência da melhoria do processo de desenvolvimento o que levará a redução de tempo e custo da manutenção do software.

Nesta seção nossa intenção não foi sugerir que o desenvolvimento baseado em agentes é mais complexo e inclinado a erro do que as abordagens tradicionais de engenharia de software. Ao invés disso, nós reconhecemos que há certas armadilhas que são comuns às soluções de agente – assim como há certas armadilhas comuns às soluções baseadas em objetos.

Na próxima seção apresentamos a metodologia orientada a agentes Tropos utilizada na nossa proposta. Essa metodologia é uma das que abrange o maior número de fases no processo de desenvolvimento de software orientado a agentes. Através da utilização do framework NFR para comparações de metodologias de desenvolvimento de software orientada a agentes Silva [SIL04] demonstrou que a metodologia Tropos é bastante adequada ao desenvolvimento de sistemas multi-agentes.

4.7 Tropos – Uma Metodologia Para o Desenvolvimento de Software Orientado a Agentes

Sistemas de software freqüentemente falham no suporte às organizações das quais eles são parte integrante. Isto acontece devido à ausência de um entendimento apropriado da organização pelos projetistas do sistema de software, bem como à freqüência das mudanças organizacionais que não podem ser acomodadas pelos sistemas de software existentes. Neste contexto, a engenharia de requisitos é reconhecida como a fase mais crítica no desenvolvimento de sistemas pois, nessa fase, devem-se balancear as considerações técnicas em relação às questões sociais e organizacionais. O projeto Tropos ([CAS02a], [BRE04]), vem desenvolvendo uma metodologia inspirada em conceitos organizacionais, que reduz as falhas do software em relação aos aspectos organizacionais da empresa. O nome Tropos deriva do grego tropé, que significa “facilmente alterado” ou “facilmente adaptado”.

A metodologia Tropos é baseada nos conceitos utilizados para modelar requisitos iniciais (vide seção 2.4) e complementa propostas para plataformas de programação orientada a agentes. O processo inicia com um modelo do ambiente no qual o sistema que está sendo desenvolvido irá operar. O modelo é descrito em termos de atores, suas metas e interdependências. Através de refinamentos incrementais, este modelo é estendido para incluir tanto o sistema a ser desenvolvido

quanto os seus subsistemas, que também são representados como atores a quem foram delegadas metas para atingir, planos para executar e recursos para fornecer.

A proposta Tropos é fundamentada em um pequeno conjunto de conceitos e fornece ferramentas e técnicas para construir modelos baseados nos conceitos oferecidos pelo i* [YU97a] (que simboliza “intencionalmente distribuído”), um framework de modelagem que inclui os conceitos de ator (atores podem ser agentes, posições ou papéis) e suas interdependências intencionais, incluindo dependências de meta, meta-soft, tarefa e recurso¹. Este framework inclui o modelo de dependência estratégica e o modelo de razão estratégica que são usados para capturar as intenções dos *stakeholders* (usuários, proprietários, gerentes, etc.), as responsabilidades do novo sistema em relação a estes *stakeholders*, a arquitetura do novo sistema e os detalhes do seu projeto. Estes modelos podem ser usados como parte da documentação de um sistema de software durante a operação e manutenção.

Contudo, Tropos ainda não contempla explicitamente as atividades de desenvolvimento de software tradicional: testes, distribuição, planejamento e gerenciamento e gerência de configuração. O processo de desenvolvimento de software tradicional [SOM03] (apresentado na Figura 4-1).

¹ Dependência de meta: utilizada para representar delegação de responsabilidade a fim de satisfazer um objetivo.
Dependência meta-soft: similar a dependência de meta, mas a sua satisfação não pode ser definida precisamente.
Dependência de tarefa: utilizada em situações onde o ator dependente é requisitado a realizar uma atividade.
Dependência de recurso: requisita que um ator dependente forneça um recurso para o ator do qual ele depende.

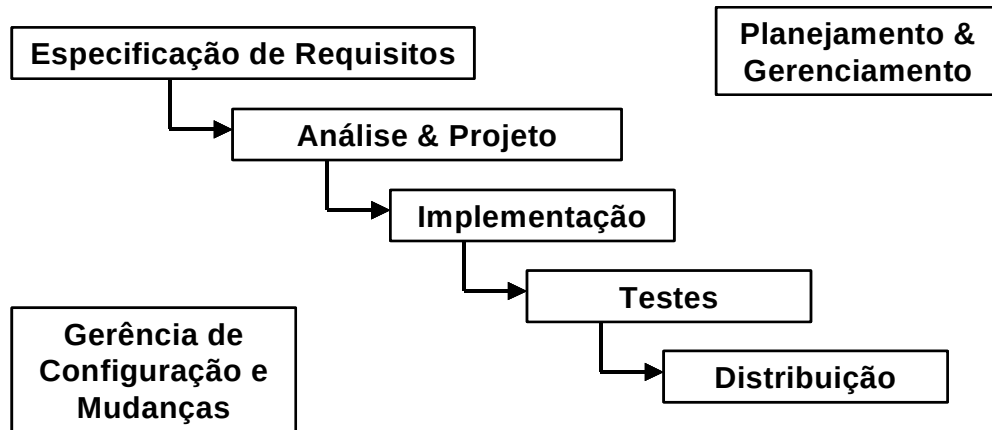


Figura 4-1 Divisão do Processo de Desenvolvimento de Software em Etapas

As características chaves da metodologia incluem o uso de conceitos de nível de conhecimento [NEW93], tais como agente, meta, plano e outros, durante todas as fases do desenvolvimento de software cobertas por Tropos. Também é atribuído um papel crucial à Engenharia de Requisitos quando o ambiente e o sistema a ser construído são analisados. As fases cobertas pela metodologia são as seguintes [CAS02a]:

- Requisitos Iniciais (*Early Requirements*): o entendimento de um problema dá-se estudando uma configuração organizacional existente. Durante esta fase, os engenheiros de requisitos modelam os stakeholders como atores e suas intenções como metas. Cada meta é analisada do ponto de vista de seu ator resultando em um conjunto de dependências entre pares de atores. As saídas desta fase são dois modelos:
 - ◇ O *modelo de dependência estratégica* que captura os atores relevantes, suas metas respectivas e suas interdependências;
 - ◇ O *modelo de razão estratégica* que determina através de uma análise meio-fim como as metas podem ser cumpridas através das contribuições de outros atores.
- Requisitos Finais (*Late Requirements*): inclui o sistema a ser desenvolvido como um outro ator no modelo de dependência estratégica. O ator representando o sistema relaciona-se com os atores sociais através das dependências. Nesta fase considera-se este ator que representa o sistema e faz-se uma análise meio-fim para produzir um novo modelo de razão estratégica. Suas metas são analisadas e podem levar a uma revisão e inclusão de novas dependências com um subconjunto de atores sociais (os usuários). Se

necessário, decompõe-se o sistema em vários sub-atores, revisando assim, os modelos de razão e dependência estratégica.

- Projeto Arquitetural: define a arquitetura global do sistema em termos de subsistemas, interconectados através de fluxos de controle de dados. A arquitetura de um sistema constitui um modelo da estrutura relativamente pequeno e intelectualmente gerenciável, que descreve como os componentes do sistema trabalham juntos. Subsistemas são representados como atores e interconexões de dado/controle são representados como dependências de ator (que representa o sistema). Esta fase consiste de três passos ([SIL04], [BAS04]):
 - ◇ Selecionar o estilo arquitetural usando como critério as qualidades desejadas que foram identificadas na fase anterior e o *framework* NFR [CHU00] para conduzir esta análise de qualidade. Incluir novos atores, de acordo com a escolha de um estilo arquitetural específico de agente [BAS03];
 - ◇ Incluir novos atores e dependências, assim como decompor os atores e as dependências existentes em sub-atores e sub-dependências. Revisar os modelos de razão e dependência estratégica. As capacidades do ator são identificadas com a análise das dependências indo e vindo do ator, bem como das metas e planos que o ator irá executar a fim de cumprir requisitos funcionais e não-funcionais;
 - ◇ Definir como as metas associadas a cada ator são cumpridas por agentes em relação aos padrões sociais. Eles são usados para solucionar uma meta específica que foi definida no nível arquitetural através da identificação de estilos organizacionais e atributos de qualidade relevantes (*softgoals*). Uma análise detalhada de cada padrão social permite definir um conjunto de capacidades associadas com os agentes envolvidos no padrão. Uma capacidade estabelece que um ator é capaz de agir para satisfazer uma determinada meta. Em particular, para cada capacidade o ator tem um conjunto de planos que podem aplicar em diferentes situações. Um plano descreve a sequência de ações a executar e as condições sob as quais o plano é aplicável. Capacidades são coletadas num catálogo e associadas ao padrão. Isto permite definir os papéis e capacidade do ator que são adequados para um domínio particular [BAS04].

- Projeto Detalhado: visa especificar o nível micro de agente, detalhando capacidades e planos usando o diagrama de atividades, como também protocolos de comunicação e coordenação. Nesta etapa pode ocorrer, por exemplo:
 - ◊ O desenvolvimento de diagramas de atividade para representar as capacidades de cada agente;
 - ◊ O desenvolvimento de diagramas de seqüência e colaboração para capturar as interações entre os agentes;
 - ◊ A elaboração dos diagramas de atividade baseados em estado para representar cada plano identificado nos diagramas de atividade.
- Implementação: transforma a especificação detalhada de um projeto em um esqueleto para a implementação. Isto é feito através de um mapeamento entre os conceitos Tropos e os elementos da plataforma de implementação de agente escolhido, tal como JACK [BUS01]. O código é adicionado ao esqueleto usando a linguagem de programação suportada pela plataforma de programação.

Tropos é a favor de uma metodologia de desenvolvimento de software que seja fundamentada em conceitos intencionais, tais como ator, meta, dependência (de meta, tarefa, recurso, meta-soft), etc.

Tropos propôs um *framework* de modelagem que visualiza o software de quatro perspectivas complementares ([CAS02a], [BRE04]):

- Social: quais são os atores relevantes, o que eles querem? Quais são suas obrigações? Quais são suas capacidades?
- Intencional: quais são as metas relevantes e como elas se relacionam? Como elas estão sendo reconhecidas e quem solicita as dependências?
- Comunicativa: como os atores dialogam e como eles podem interagir uns com os outros?

- Orientada a processo: quais são os processos de negócio / computador relevantes? Quem é responsável pelo que?

Contudo, Tropos não define atividades de Rastreamento de Requisitos no seu processo de desenvolvimento, apesar de ser uma metodologia centrada em requisitos. Nessa dissertação avaliaremos como o modelo de rastreamento de requisitos proposto por Toranzo [TOR02a] pode ser aplicado na metodologia Tropos para apoiar o desenvolvimento orientado a agentes. Em particular examinaremos como a metodologia Tropos poderia ser estendida para suportar o rastreamento de requisitos.

A seguir apresentamos as considerações finais acerca da engenharia de software orientada a agentes.

4.8 Considerações Finais

As empresas que desenvolvem software de grande porte estão apresentando grandes problemas com os atuais paradigmas de desenvolvimento. Isto é constatado, por exemplo, pela dificuldade de expressar o conceito de objeto, para tipos específicos de aplicações, como por exemplo, aplicações que exigem comportamento autônomo. Além disso, a orientação a objetos demanda um esforço considerável dos engenheiros de software para produzir software de qualidade. Nesse contexto, a orientação a agentes surge como uma nova abordagem para construir sistemas complexos e está começando a ser usada em aplicações industriais e comerciais, deixando de existir apenas nas universidades e laboratórios de pesquisa.

Neste capítulo vimos que existem várias diferenças entre o paradigma de objetos e o paradigma de agentes. No entanto, uma forma bem mais importante de diferenciar agentes de objetos não é baseada em nenhum aspecto em particular, mas no fato de que agentes possuem características adicionais em relação a objetos estando em um nível mais alto (ou diferente) de abstração. Não estamos afirmando que a orientação a agentes é uma solução universal para a construção de software e muito menos que ela irá melhorar em ordem de magnitude o processo de desenvolvimento de software. Entretanto, o uso desta tecnologia na indústria tem demonstrado que as técnicas orientadas a agentes levam a melhorias no desenvolvimento de software distribuído complexo.

Infelizmente, os benefícios prometidos pelo paradigma de agentes ainda não podem ser completamente obtidos porque não existe até o momento um consenso sobre a padronização de notações e processos de desenvolvimento para construção de software orientado a agentes.

Atualmente, muitos deles são construídos de forma *ad-hoc* e outros utilizam metodologias ainda imaturas.

Existem várias propostas de metodologias voltadas para o desenvolvimento de software orientado a agentes. Porém, poucas são centradas na Engenharia de Requisitos. Por esse motivo, abordamos neste capítulo a metodologia Tropos, que possui uma abordagem centrada em requisitos e é orientada a metas.

A cobertura completa do processo de desenvolvimento de software é considerada essencial para a engenharia de software orientada a agentes. Entretanto, não é apenas com o argumento de tratar a fase de requisitos iniciais que uma metodologia orientada a agentes pode ser considerada melhor em relação a outras metodologias. Em particular, metodologias orientadas a agentes são inerentemente intencionais, baseadas em conceitos como agente, meta, plano, etc. Isto mostra claramente a necessidade de se focar no ambiente (organizacional) onde o sistema em desenvolvimento irá operar. Entender tal ambiente pode reduzir consideravelmente a incompatibilidade entre o sistema e seu ambiente operacional. Neste contexto, fica clara a relevância do Projeto Tropos no desenvolvimento de software orientado a agentes.

Contudo, Tropos não suporta o Rastreamento de Requisitos apesar de ser uma metodologia com uma abordagem centrada em requisitos. No próximo capítulo apresentamos nossa proposta de extensão da metodologia Tropos com a incorporação de atividades relacionadas ao rastreamento de requisitos.

Capítulo 5 – Incorporando Rastreamento de Requisitos na metodologia Tropos

Este capítulo detalha a proposta desta dissertação, a expansão da metodologia de desenvolvimento de software orientada a agentes Tropos com a inclusão das atividades de Rastreamento de Requisitos nas suas fases de desenvolvimento.

Capítulo 5 INCORPORANDO RASTREAMENTO DE REQUISITOS NA METODOLOGIA TROPOS

5.1 Introdução

Nesse capítulo detalhamos nossa proposta de trabalho cujo objetivo principal é estender a metodologia Tropos apresentada na seção 4.7, para incorporar as atividades de rastreamento de requisitos. Para isso, vamos nos basear nas diretrizes sugeridas por Toranzo [TOR02a] para o processo de desenvolvimento de um modelo de rastreamento de requisitos (apresentado na seção 3.6).

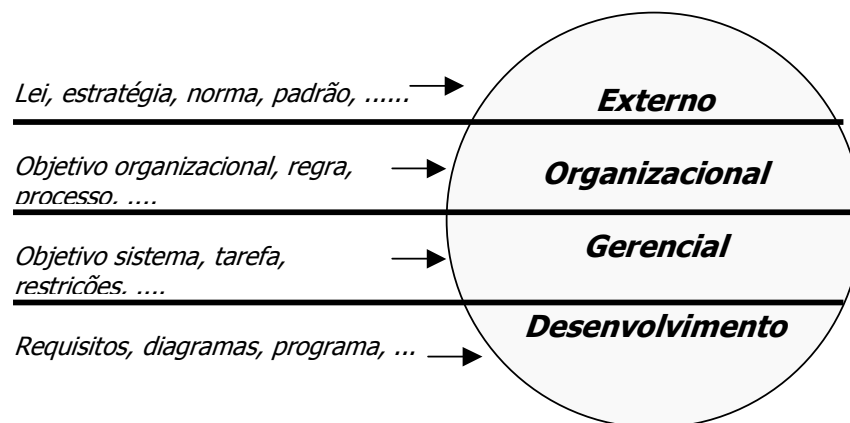


Figura 5-1 Classificação das informações do rastreamento - Segmentos de informação

As diretrizes propostas por Toranzo trabalham com os seguintes segmentos de informação: Externo, Organizacional, Gerencial e Desenvolvimento, apresentados na Figura 5-1 e detalhados a seguir:

- Externo: representa regulamentações, leis e restrições que pertencem ao contexto sócio-político e / ou econômico na qual está inserida a organização. Por exemplo, a lei CPMF (Contribuição sobre movimentação financeira) é um elemento externo aos bancos brasileiros. As informações do segmento externo são geralmente identificadas nas fases iniciais de desenvolvimento do sistema como requisitos não-funcionais;
- Organizacional: representa requisitos organizacionais, isto é, estratégias, restrições e benefícios da organização na qual o sistema será inserido, que obrigatoriamente devem

ser levados em consideração a fim de garantir que o sistema atenda plenamente às expectativas do cliente;

- Gerencial: está relacionado à atividade de planejamento da gerência do projeto, mais especificamente, aos relacionamentos entre as tarefas e os requisitos para um melhor acompanhamento e controle dos requisitos do sistema;
- Desenvolvimento: representa os elementos / artefatos produzidos nas diferentes atividades do desenvolvimento de software. Alguns dos artefatos encontrados neste segmento são: documentos de requisitos, diagramas e programas.

Para que as atividades de rastreamento de requisitos sejam implementadas em um projeto, faz-se necessário documentar os artefatos produzidos durante o ciclo de vida de desenvolvimento do software, bem como selecionar as informações que deverão ser rastreadas de acordo com as restrições do projeto (por exemplo, tamanho, complexidade, custo, cronograma, etc).

Nas próximas seções apresentamos como as informações associadas aos artefatos gerados em cada uma das fases da metodologia Tropos podem ser armazenadas no modelo de rastreamento de requisitos. Para alcançar esse objetivo utilizamos as diretrizes propostas por Toranzo para o processo de desenvolvimento de um modelo de rastreamento de requisitos.

5.2 Modelo Intermediário¹ de Rastreamento de Requisitos

O modelo intermediário de rastreamento de requisitos proposto por Toranzo (seção 3.6) [TOR02a] foi concebido através de uma combinação de fatores, tais como: boas práticas, estudo de caso, abstração, aplicação dos segmentos de informação e extensão dos modelos de Ramesh [RAM01] devido a sua contribuição e experiência na elaboração de modelos de rastreamento. Os principais objetivos do modelo intermediário de rastreamento são:

- Servir como um ponto de partida para identificação, discussão e construção de um modelo de rastreamento;
- Ressaltar as informações identificadas nos modelos de rastreamento de diferentes projetos de software;
- Contribuir na derivação dos modelos de rastreamento dos projetos de software.

O modelo intermediário de rastreamento de requisitos é dividido em dois sub-modelos: sub-modelo para a gerência de requisitos e sub-modelo de projeto de acordo com a proposta de Toranzo. A subdivisão do modelo intermediário foi realizada para facilitar sua apresentação e compreensão. Além desses sub-modelos, Toranzo propõe um modelo de raciocínio para identificação e estruturação dos problemas e das decisões tomadas (raciocínio) no decorrer do desenvolvimento de um software.

O modelo intermediário de rastreamento de requisitos proposto por Toranzo utiliza os relacionamentos apresentados na Tabela 5-1.

Tipo de associação	Descrição
Generalização	Um relacionamento entre um elemento geral e um elemento mais específico. O elemento mais específico possui todas as informações do elemento mais geral e contém informações adicionais.
Aloca	Uma associação unidirecional entre dois elementos, representa um relacionamento entre requisitos funcionais e um subsistema.
Agregação	Uma associação que modela um relacionamento todo-parte entre um elemento, denominado <i>todo</i> , e um ou mais elementos que são denominados <i>partes</i> .

¹ Modelo Intermediário: modelo que não é completo, pois não possui todas as possíveis classes e relacionamentos existentes nos diferentes modelos de rastreamento de software [TOR02a]. Desenvolver um modelo intermediário completo é impossível porque não é possível prever todas as classes e relacionamentos para todos os domínios de aplicação [GOT96].

Tipo de associação	Descrição
Satisfação	Uma associação entre dois elementos denominados elemento origem e elemento destino. A associação é desenhada do elemento origem para o elemento destino. A associação satisfação é uma relação de dependência que especifica que o elemento origem tem uma dependência de realização com o elemento destino. O termo satisfação é usado para expressar que algo deverá ser realizado no elemento destino para satisfazer o elemento origem.
Recurso	Uma associação que especifica que um elemento possui uma dependência de informação ou dependência física com outro elemento.
Responsabilidade	Uma associação que visa capturar a participação, responsabilidade e ação das pessoas sobre artefatos ou elementos do processo de software.
Representação	Uma associação que é usada para mapear um elemento em um outro elemento.

Tabela 5-1 Tipos de Relacionamentos utilizados no modelo intermediário de rastreamento de requisitos

Nas próximas seções apresentamos os sub-modelos resultantes do modelo intermediário e o modelo de raciocínio.

5.2.1 Sub-Modelo para a Gerência de Requisitos

Independentemente da natureza do projeto de software, o surgimento de novos requisitos e as mudanças dos requisitos existentes do sistema são inevitáveis [KOT98]. Acrescentando-se a complexidade, o tamanho do sistema e a enorme quantidade de requisitos a gerenciar, fica claro que várias dificuldades irão surgir, tais como: consulta a informações relacionadas; recuperação de informações relacionadas para avaliar uma proposta de mudança; acompanhamento do projeto em relação ao estado dos seus requisitos (por exemplo: aprovado, rejeitado, ou implementado). Para evitar esses problemas há a necessidade de gerenciar requisitos.

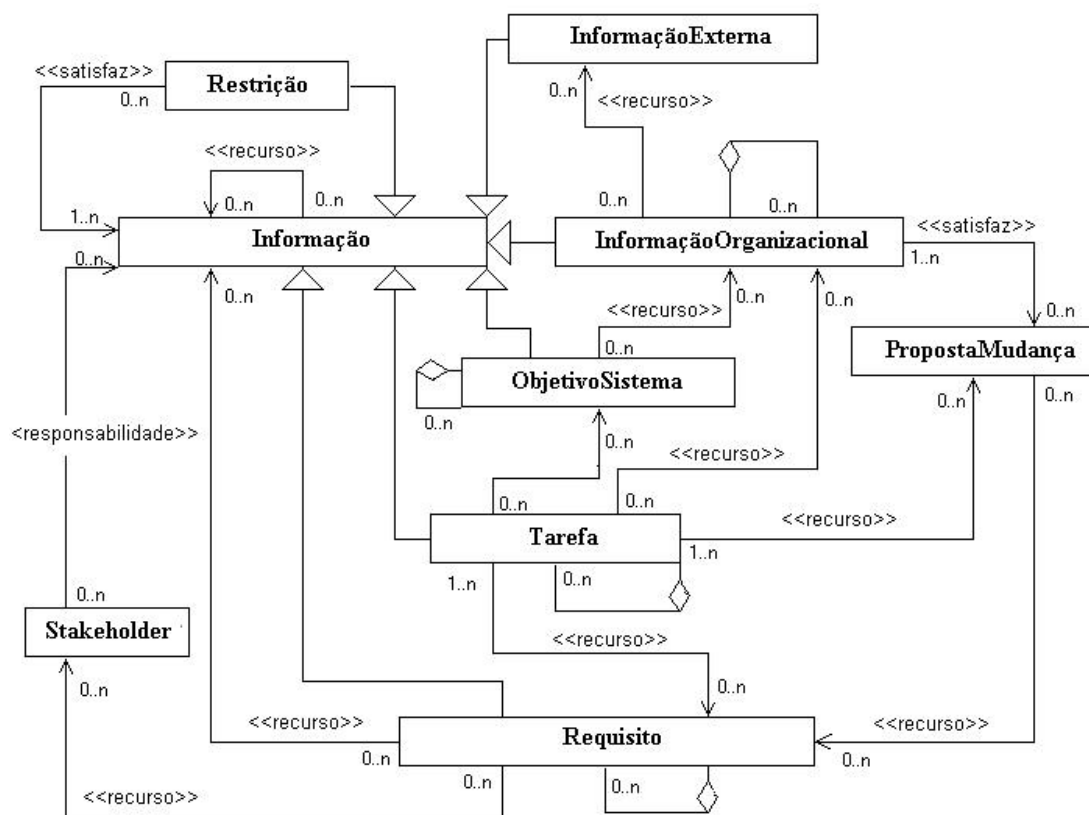


Figura 5-2 Sub-modelo de Gerência de Requisitos

A Figura 5-2 apresenta o sub-modelo para a gerência de requisitos, subconjunto do modelo intermediário. A seguir, explicaremos brevemente os relacionamentos desse sub-modelo.

O relacionamento entre as classes *InformaçãoOrganizacional* e *PropostaMudança* expressa que a satisfação de uma informação organizacional depende da satisfação de muitas (ou nenhuma) propostas de mudança. Por exemplo, uma administradora de cartões de crédito que deseja incentivar os seus clientes a usarem os seus cartões, pode optar por sortear viagens entre os mesmos. Cada vez que o cliente pagar suas compras (digamos, com valores superiores a R\$ 50,00) com um dos cartões da administradora, o sistema garantirá a geração de um cupom para participação no sorteio das viagens. Para viabilizar essa estratégia de marketing, a administradora propõe uma mudança no sistema para identificar as compras com valores superiores a R\$ 50,00. Portanto, o objetivo de incentivar o uso dos cartões da administradora depende da efetivação da mudança proposta.

O auto-relacionamento de agregação sobre a classe *Tarefa* expressa que uma tarefa pode estar composta de outras tarefas. As dependências do tipo recurso da classe *Tarefa* com as classes *InformaçãoOrganizacional*, *ObjetivoSistema*, *PropostaMudança* e *Requisitos*, identificam as informações utilizadas pelo gerente de projeto na elaboração do cronograma, ou seja, os recursos

(*InformaçãoOrganizacional*, *ObjetivoSistema*, *PropostaMudança* e *Requisito*) que servirão para a criação das atividades de acordo com o cronograma de desenvolvimento desejado.

Para a identificação dos *stakeholders* que levantaram os requisitos, o modelo sugere o relacionamento entre as classes *Requisito* e *Stakeholder*.

O relacionamento entre as classes *Requisito* e *Informação* indica quais informações (segmentos externo, organizacional, gerencial e de desenvolvimento – vide seção 5.1) foram utilizadas para a elaboração dos requisitos.

A subseção seguinte apresenta o sub-modelo de rastreamento para o projeto de software.

5.2.2 Sub-modelo para o Projeto do Software

Um sub-modelo de rastreamento para o projeto de software identifica os elementos encontrados na fase de análise e projeto de um sistema. A Figura 5-3 apresenta o sub-modelo de rastreamento para o projeto de software, subdivisão do modelo intermediário.

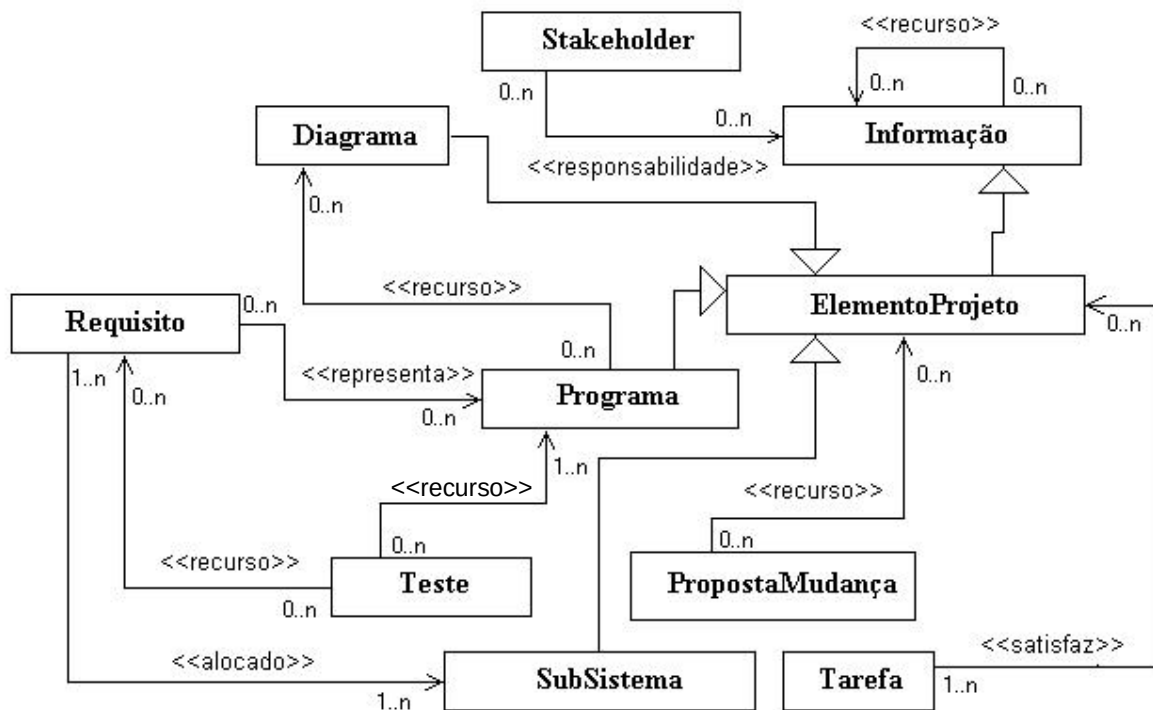


Figura 5-3 Sub-modelo de Projeto

O relacionamento entre as classes *Tarefa* e *ElementoProjeto* expressa que a satisfação de uma tarefa depende de elementos do projeto. Por exemplo, para a integração dos subsistemas de um projeto é necessário que os subsistemas já tenham sido implementados e testados.

Para o armazenamento de quais elementos serão impactados por uma proposta de mudança, o modelo sugere o relacionamento entre as classes *PropostaMudança* e *ElementoProjeto*.

Os relacionamentos entre as classes *Requisito* e *Programa*, e entre as classes *Programa* e *Diagrama* identificam, respectivamente, o programa que implementa o requisito e o diagrama que serviu de subsídio para a implementação do programa.

Para a identificação de quais requisitos foram implementados em um subsistema deve existir um relacionamento entre as classes *Requisito* e *Subsistema*.

Os relacionamentos entre as classes *Teste* e *Requisito*; e entre as classes *Teste* e *Programa* expressam respectivamente, os requisitos utilizados no planejamento dos testes e os programas utilizados na execução dos testes.

O relacionamento entre as classes *Programa* e *Diagrama* identifica que um diagrama pode ser um recurso de informação para um programa. Por exemplo, um diagrama de entidade-relacionamento é um recurso de informação para escrever as sentenças SQL (*Structure Query Language*) de um programa.

No decorrer do desenvolvimento do projeto, muitas decisões são tomadas em função de novos problemas e necessidades. Geralmente essas decisões ficam registradas na mente dos participantes e responsáveis. Toranzo [TOR02a] sugere esse modelo para identificação e estruturação dos problemas e das decisões tomadas (raciocínio) no decorrer do desenvolvimento de um software. Na próxima seção apresentamos esse modelo.

5.2.3 Modelo de Raciocínio

A Figura 5-4 apresenta o modelo de raciocínio. A classe *Assunto* representa um problema ou uma preocupação que requer uma discussão e resolução no projeto. Independentemente do assunto, devem ser considerados os objetivos do sistema, as restrições do projeto, e a existência de suposições. Para o armazenamento dessas informações, devem ser utilizados os relacionamentos do tipo recurso da classe *Assunto* com as classes *ObjetivoSistema*, *Restrição* e *Suposição*.

O relacionamento entre as classes *Assunto* e *ObjetivoSistema* identifica os objetivos do sistema que devem ser levados em consideração para a discussão de um assunto. Por exemplo, os desenvolvedores podem propor a modificação de um relatório visando sua melhoria, porém, um objetivo do sistema estabelece que todos os relatórios devem seguir os padrões definidos pelo cliente. Esse exemplo ilustra que é necessário que um assunto proposto leve sempre em consideração os objetivos do sistema.

O relacionamento entre as classes *Assunto* e *Restrição* identifica quais restrições devem ser levadas em consideração na resolução de um problema (*Assunto*). Um exemplo de assunto é a escolha de um estilo de documentação, porém, pode existir uma restrição do sistema que estabeleça que o estilo de documentação a ser utilizado no projeto deve ser previamente definido e aprovado pelo cliente.

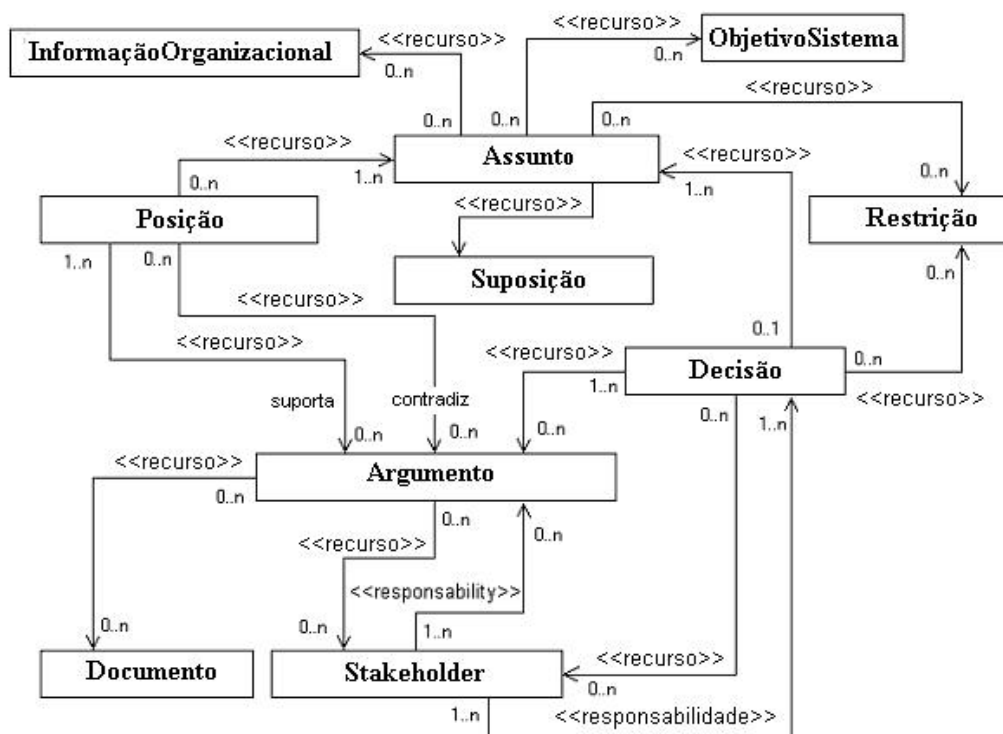


Figura 5-4 Modelo de Raciocínio

O relacionamento entre as classes *Assunto* e *Suposição* expressa que um assunto pode estar associado a suposições. Por exemplo, para integrar várias ferramentas CASE (*Assunto*), pode-se ter como suposição que um Sistema Gerenciador de Banco de Dados Orientado a Objetos (SGBDOO) permite uma melhor integração dos dados.

No modelo de raciocínio, uma posição representa uma solução alternativa para um assunto. O relacionamento entre as classes *Posição* e *Assunto* identifica quais as posições em relação ao assunto. Por exemplo, considerando o problema da integração de ferramentas CASE, uma posição pode recomendar a compra de um SGBDOO enquanto que uma outra alternativa recomenda a compra de um Sistema Gerenciador de Banco de Dados Relacional.

Os relacionamentos *suporta* e *contradiz* entre as classes *Posição* e *Argumento* expressam quais argumentos estão a favor ou contra a posição. Seguindo o exemplo anterior, a posição que recomenda a compra de um SGBDOO pode argumentar que o projeto deseja explorar a tecnologia

orientada a objetos, enquanto a posição contrária pode argumentar que a tecnologia relacional está madura e consolidada no mercado industrial.

Os argumentos podem ter como recursos de informação *Documento* e *Stakeholder*. Isso é apresentado pelos relacionamentos entre as classes *Argumento*, *Documento* e *Stakeholder*. Por exemplo, um argumento que recomenda a compra de um Sistema Gerenciador de Banco de dados (SGBD) pode ter como recurso de informação uma pessoa especialista na área de integração de ferramentas e ambientes CASE, bem como uma revisão favorável (recurso) publicada em uma revista especializada.

O relacionamento *responsável* (de *Stakeholder* para *Argumento*) expressa que uma pessoa pode ser responsável por muitos (ou nenhum) argumentos. Por exemplo, um engenheiro de *software* pode ser responsável por um argumento que recomenda a compra do SGBD para um projeto.

A classe *Decisão* representa uma coleção de decisões. O relacionamento entre as classes *Decisão* e *Argumento* especifica que um argumento pode ter como base uma decisão prévia do projeto. O relacionamento entre as classes *Decisão* e *Restrição* especifica que uma decisão tem como recurso de informação as restrições do projeto. A idéia é que uma decisão tomada não pode invalidar algumas restrições impostas a um projeto. O relacionamento entre as classes *Decisão* e *Stakeholder* especifica que uma decisão tem como recurso de informação as pessoas envolvidas para futuros esclarecimentos. O relacionamento entre as classes *Stakeholder* e *Decisão* especifica que as pessoas são responsáveis pela tomada de decisões.

Nas próximas seções, apresentamos como o modelo de rastreamento intermediário proposto por Toranzo foi adequado à metodologia Tropos.

5.3 Diretrizes para Instanciação do Modelo de Referência

O processo para o desenvolvimento de um modelo de rastreamento proposto por Toranzo [TOR02a] consiste de três fases:

- *Coleta de informações*: engloba as atividades de identificação das informações dos quatro segmentos,
- *Estruturação e construção do modelo de rastreamento*: está relacionada com a organização das classes,

- *Definição e preenchimento das matrizes de rastreamento*: consiste na definição, preenchimento e validação das matrizes de rastreamento.

A Figura 5-5 apresenta um diagrama de atividades da UML (*Unified Modeling Language*) [BOO99] que identifica as atividades do processo para desenvolver um modelo de rastreamento.

A primeira fase do processo, chamada *coleta de informação*, é composta de quatro atividades concorrentes: identificar classes no segmento de informação externo; identificar classes no segmento de informação organizacional; identificar classes no segmento de informação gerencial; e identificar classes no segmento de informação de desenvolvimento. O objetivo dessas atividades é identificar classes candidatas dos quatro diferentes segmentos de informação que podem compor o modelo de rastreamento de um sistema.

A segunda fase do processo, chamada *estruturação e construção do modelo de rastreamento*, é composta das atividades: organizar e estruturar as classes candidatas e estabelecer relacionamentos. Essas atividades são responsáveis pela organização e relacionamentos das classes candidatas.

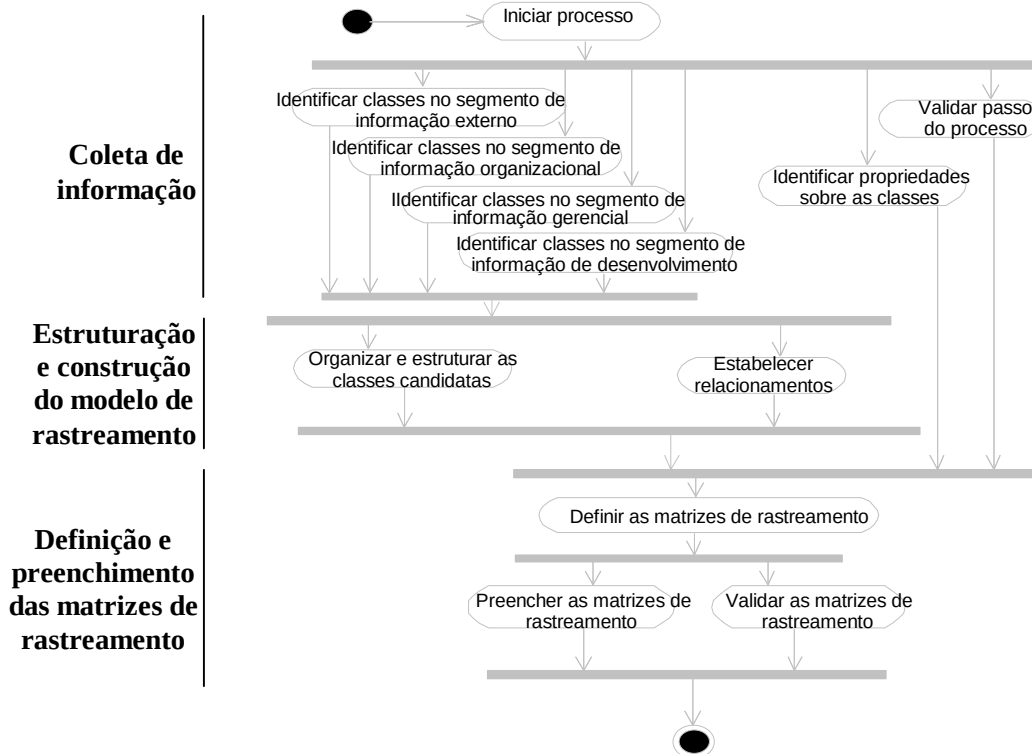


Figura 5-5 Visão geral do processo para elaborar um modelo de rastreamento

A terceira fase, chamada *definição e preenchimento das matrizes de rastreamento*, é formada pelas seguintes atividades executadas em paralelo: preencher as matrizes de rastreamento (seção 5.10) e validar as matrizes de rastreamento. Elas possuem o objetivo de armazenamento e captura das informações das dependências entre as instâncias das classes.

As atividades *validar passo do processo e identificar propriedades sobre as classes* são concorrentes às atividades da primeira e segunda fase do processo de rastreamento.

Para a aplicação desse processo, Toranzo [TOR02a] sugeriu um conjunto de diretrizes apresentadas na Tabela 5-2.

Diretriz	Descrição
Diretriz 1	Identificar classes no segmento de informação externo (restrições, objetivos e leis) que podem afetar o sistema.
Diretriz 2	Identificar classes no segmento de informação organizacional (objetivos, estratégias ou regras de negócio) que serão rastreadas.
Diretriz 3	Identificar classes no segmento de informação gerencial no modelo de rastreamento.
Diretriz 4	Identificar classes no segmento de informação de desenvolvimento: identificar subsistemas.
Diretriz 5	Identificar classes no segmento de informação de desenvolvimento: identificar requisitos.
Diretriz 6	Identificar classes no segmento de informação de desenvolvimento: identificar diagramas.
Diretriz 7	Identificar classes no segmento de informação de desenvolvimento: identificar programas.
Diretriz 8	Identificar classes no segmento de informação de desenvolvimento: identificar teste.
Diretriz 9	Remover as classes de informação irrelevantes.
Diretriz 10	Integrar as classes com o mesmo significado.
Diretriz 11	Integrar novas classes.
Diretriz 12	Organizar as classes.
Diretriz 13	Estabelecer relacionamentos.
Diretriz 14	Recomendar atributos sobre as classes.
Diretriz 15	Definir uma matriz para cada relacionamento do modelo.
Diretriz 16	Validar o modelo de rastreamento.

Tabela 5-2 Diretrizes Propostas por Toranzo para Instanciar o Modelo de Referência

As diretrizes devem ser aplicadas sequencialmente para cada fase do desenvolvimento de software. Como nossa proposta utiliza a metodologia Tropos, as fases dessa metodologia são: Requisitos Iniciais, Requisitos Finais, Projeto Arquitetural, Projeto Detalhado e Implementação.

Nessa dissertação utilizaremos as convenções /abreviaturas apresentadas na Tabela 5-3 para as classes do modelo de rastreamento de Toranzo [TOR02a]:

Classe	Abreviatura
INFORMAÇÃO EXTERNA	[IEXT]
INFORMAÇÃO ORGANIZACIONAL	[IORG]
OBJETIVO DO SISTEMA	[OBJ]
OBJETIVO ORGANIZACIONAL	[OBO]
REQUISITO FUNCIONAL	[RF]
REQUISITO NÃO-FUNCIONAL	[RNF]
PROGRAMA	[PROG]
DIAGRAMA	[DIAG]

Classe	Abreviatura
TAREFA	[TAR]
RESTRIÇÃO DO SISTEMA	[SRES]
SUBSISTEMA	[SUB]
POSIÇÃO	[POS]
ASSUNTO	[ASS]
DECISÃO	[DEC]
ARGUMENTO	[ARG]
RESTRIÇÃO	[RES]
POSIÇÃO	[POS]

Tabela 5-3 Abreviaturas para as classes de informação

Para facilitar o entendimento do trabalho, nas próximas seções apresentaremos inicialmente as diretrizes 1 a 12, referentes à identificação das classes em cada um dos segmentos de informação, aplicadas em cada uma das fases de desenvolvimento da metodologia Tropos, verificando a aderência de cada uma delas às fases, bem como identificando oportunidades de melhoria.

As diretrizes 13 a 16 são aplicadas nas classes que serão incluídas no modelo de rastreamento cobrindo, portanto, todas as fases da metodologia Tropos. Por essa razão, os resultados da aplicação das diretrizes 13 a 16 são apresentadas numa seção a parte (seção 5.9).

Nas próximas seções, os resultados da aplicação das diretrizes são apresentados de maneira consolidada por fase da metodologia Tropos em tabelas com a seguinte estrutura: Nome do Artefato gerado na fase, Segmentos de Informação identificados e Classes do Modelo de Rastreamento identificadas. Começaremos pela fase de Requisitos Iniciais.

5.4 Requisitos Iniciais

Vimos na seção 4.7 que Tropos adota os conceitos oferecidos pelo framework *i** para suportar a modelagem e a análise de suas fases iniciais. Isto significa que tanto o ambiente do sistema quanto o sistema em si são vistos como organizações de atores, cada um tendo metas a serem cumpridas e dependendo da ajuda de outros atores no cumprimento destas metas.

Durante a fase de requisitos iniciais, o engenheiro de requisitos deve capturar e analisar as intenções dos *stakeholders*. Essas intenções são modeladas como objetivos utilizando o *framework i**. O *framework i** inclui o Modelo de Dependência Estratégica (*Strategic Dependency-SD*) para descrever a rede de relacionamentos entre atores, como também o Modelo de Razão Estratégica (*Strategic Rationale-SR*) para descrever e dar suporte às decisões de cada ator.

Um modelo de dependência estratégica é um grafo, onde cada nó representa um ator e um relacionamento entre dois atores indica que um ator depende do outro para satisfazer um objetivo.

O *framework i** representa “atores” como círculos. *Dependums* – metas, metas-soft, tarefas e recursos – são respectivamente representados como elipses, nuvens, hexágonos e retângulos. As

dependências (*dependum*) têm a forma *depender* → *dependum* → *dependee*. Para ilustrar esses conceitos, vamos utilizar um modelo para gerenciamento de conferência apresentado na Figura 5-6 que possui um processo com muitas fases, envolvendo muitas pessoas ou grupos de pessoas.

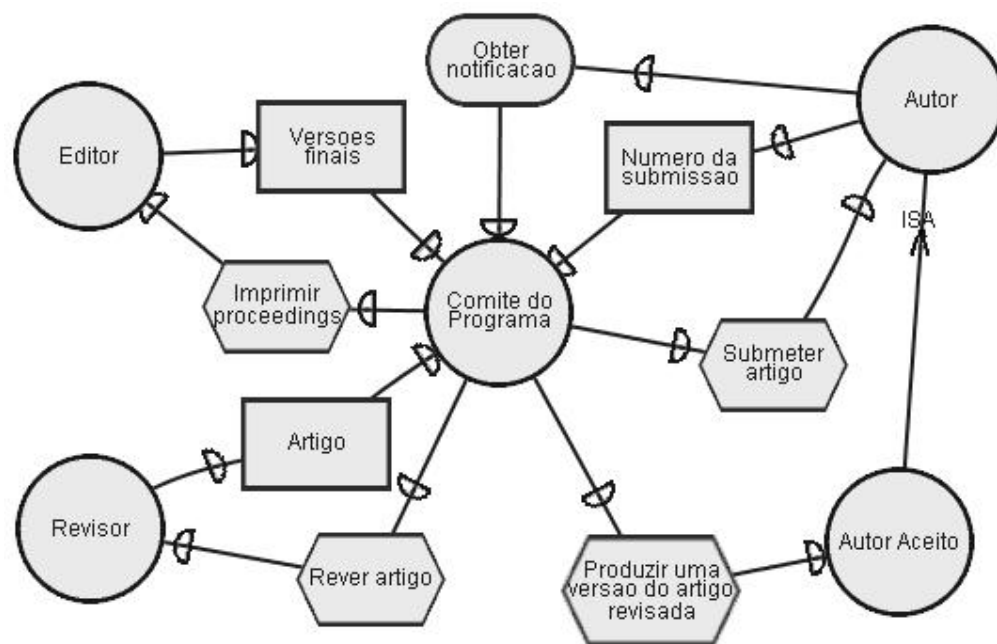


Figura 5-6 Modelo i* para gerenciamento de conferência

Durante a fase de submissão, autores submetem seus artigos, e são informados de que seus artigos foram recebidos através de um número de submissão. Na fase de revisão, o Comitê do Programa deve realizar a revisão dos artigos: contatando, se necessário, revisores adicionais e delegando-os a revisão de alguns artigos. Eventualmente, revisões são utilizadas para decidir sobre a aceitação ou rejeição da submissão. Na fase final, autores precisam ser notificados em relação à decisão e, em caso de aceitação, devem ser contatados para a produção de uma versão revisada dos artigos. O Editor deve coletar as versões finais dos artigos e imprimi-las. A seguir explicamos os conceitos encontrados em Tropos:

- **Ator:** Um ator é uma entidade ativa que executa ações para atingir metas exercitando sua base de conhecimento. Por exemplo, na Figura 5-6 encontramos o ator *Revisor*. O ator *Revisor* é aquele que atua na revisão dos artigos delegados pelo comitê do programa.
- **Dependência:** descreve um relacionamento intencional entre dois atores, i.e., um “acordo” (chamado *dependum*) entre dois atores: o *depender* e o *dependee*, onde um ator (*depender*) depende de um outro ator (*dependee*) para alcançar algo (*dependum*). Segue abaixo a descrição de cada uma das dependências representadas na metodologia Tropos.

- ◇ Meta ou Objetivo: Uma meta é uma condição ou situação no mundo que o ator gostaria de atingir. Como a meta vai ser atingida não está especificada. Na Figura 5-6, encontramos como exemplo a meta *Obter notificação* caracterizando a dependência entre os atores *Autor* e *Comitê do Programa*, isto é, o ator *Autor* depende do comitê do programa para receber o número da submissão que atesta que seu artigo foi devidamente recebido pelo comitê.
- ◇ Meta-soft ou Objetivo-soft: Uma meta-soft é uma condição ou situação no mundo que o ator gostaria de atingir, mas diferente do conceito anterior de meta, não há um critério claramente definido de verificação se a condição desejada foi atingida. Portanto, a sua satisfação depende do julgamento subjetivo e interpretação dos *stakeholders*.
- ◇ Recurso: Um recurso é uma entidade (física ou de informação), com a qual a principal preocupação é se ela está disponível. A dependência de recurso *Artigo* entre os atores *Revisor* e *Comitê do Programa*, ilustrada na Figura 5-6, significa que o *Revisor* depende do recebimento do *Artigo* (recurso) do comitê do programa para dar início a sua revisão.
- ◇ Tarefa: Uma tarefa especifica uma forma particular de se fazer algo. Tarefas podem ser vistas como as soluções que irão satisfazer as metas e metas-soft (operacionalizações). Estas soluções provêm operações, processos, representações de dados, estruturação, restrições e agentes para atender às necessidades estabelecidas nas metas e metas-soft. Dependências de tarefa são usadas em situações onde o *dependee* é requisitado a executar uma dada atividade. Por exemplo, a dependência de tarefa *Produzir uma versão revisada do artigo* entre os atores *Comitê do Programa* e *Autor Aceito* (vide Figura 5-6) ilustra que para o *Comitê do Programa* ter o artigo revisado precisa que o *Autor Aceito* faça a revisão. Nessa dissertação a dependência tarefa será mapeada em requisito funcional.
- *Depender*: O *dependor* é o ator dependente. Por exemplo, na Figura 5-6 é apresentado o ator *Autor* como sendo o *dependor* do recurso *Número da submissão*, isto é, ele depende do *Comitê do Programa* para receber o número da submissão do artigo.

- *Dependee*: O *dependee* é o ator de quem se depende. Portanto, o *dependee* da dependência *Obter notificação*, ilustrada na Figura 5-6, é o ator *Comitê do Programa*, isto é, ele é responsável por enviar uma notificação ao autor se o artigo do autor for aceito ou não.

O Modelo de Dependência Estratégica e o Modelo de Razão Estratégica são artefatos da fase de requisitos iniciais da metodologia Tropos. Esses artefatos armazenam informações do sistema a ser desenvolvido que devem fazer parte de um modelo de rastreamento: atores e dependências. Esses conceitos são tratados na proposta de Toranzo: atores são *stakeholders* e dependências são os relacionamentos entre eles.

Sugerimos que o artefato Modelo de Dependência Estratégica faça parte de um documento inicial do sistema que deve apresentar uma visão geral do sistema a ser desenvolvido dos pontos de vista organizacional e externo. Nesse documento, o engenheiro de requisitos deverá detalhar de forma escrita o papel de cada ator e as dependências entre eles. Os atores e dependências devem receber uma numeração única a fim de serem identificados em qualquer fase do projeto.

Na próxima seção aplicamos as diretrizes propostas por Toranzo (vide seção 5.3) na fase Requisitos Iniciais de Tropos.

5.4.1 Aplicando Diretrizes 1 a 12 na Fase de Requisitos Iniciais

De uma maneira geral, Toranzo propõe nas diretrizes 1 a 8 que as informações de cada um dos segmentos (externo, organizacional, gerencial e desenvolvimento) sejam identificadas. Para a fase de requisitos iniciais, esta identificação foi realizada para cada um dos conceitos do Modelo de Dependência Estratégica. O resultado da aplicação das diretrizes é apresentado na Tabela 5-4.

Modelo Dependência Estratégica	Segmentos de Informação → Classes
Dependência de Meta / Objetivo	Externo → InformaçãoExterna
Dependência de Meta / Objetivo (soft)	Organizacional → InformaçãoOrganizacional
	Gerencial → ObjetivoSistema
Dependência de Recurso	Externo → InformaçãoExterna
	Organizacional → InformaçãoOrganizacional
Dependência de Tarefa	Gerencial → Requisito

Tabela 5-4 Requisitos Iniciais: Aplicação das diretrizes para identificação das classes dos segmentos de informação

A fase de requisitos iniciais é caracterizada por atividades que tratam de questões externas, organizacionais e não funcionais [YU97a]. Devido a essa característica, as dependências de meta e meta-soft identificadas no Modelo de Dependência Estratégica serão obrigatoriamente alocadas nos segmentos de informação externo, organizacional ou gerencial através da classe Objetivo do

Sistema, pois nessa fase (Requisitos Iniciais) tem-se uma visão macro do sistema a ser desenvolvido que deve ser mapeada em objetivos do sistema.

Já a dependência de recurso pode ser mapeada para os segmentos de informação externo ou organizacional, visto que na fase Requisitos Iniciais não é possível identificar dependência de recurso em relação ao sistema (nível gerencial e de desenvolvimento). A dependência de tarefa pode ser mapeada apenas para o segmento de informação gerencial na classe Requisito, mas nessa fase os requisitos fornecem uma visão geral dos requisitos sistema, sendo decompostos nas fases posteriores.

Os resultados da aplicação das diretrizes 9 a 12 estão apresentados na Tabela 5-5. A não utilização das diretrizes na metodologia Tropos não significa que elas não devem ser utilizadas quando da instanciação da metodologia no desenvolvimento de um sistema, isto é, dependendo da complexidade e / ou do domínio da aplicação, as diretrizes poderão ser utilizadas. O mesmo raciocínio vale para cada uma das fases subseqüentes da metodologia.

Diretriz	Resultado da aplicação
Diretriz 9 - Remover as classes de informação irrelevantes.	Não foram identificadas classes irrelevantes.
Diretriz 10 - Integrar as classes com o mesmo significado.	Não foi necessário integrar as classes, visto que nessa fase foram identificadas apenas três classes de acordo com a Tabela 5-4.
Diretriz 11 - Integrar novas classes.	Não foi necessário.
Diretriz 12 - Organizar as classes.	O relacionamento generalização não foi utilizado como sugerido na proposta de Toranzo.

Tabela 5-5 Requisitos Iniciais: Resultado da aplicação das diretrizes 9 a 12

Na próxima seção apresentamos a fase Requisitos Finais da metodologia Tropos.

5.5 Requisitos Finais

A análise na fase de Requisitos Finais tem como saída um documento de especificação dos requisitos que descreve todos os requisitos funcionais e não funcionais do sistema. Na metodologia Tropos, o sistema de software é representado por um ou mais atores que participam de um Modelo de Dependência Estratégica juntamente com os atores do ambiente operacional do sistema.

Embora o Modelo de Dependência Estratégica utilizado na fase de requisitos iniciais forneça dicas sobre como o sistema é estruturado, ele não suporta suficientemente o processo de sugestão, análise e avaliação de soluções alternativas. Este é o papel do modelo de razão estratégica que é um grafo com quatro tipos de nós: objetivo, tarefa, recurso e objetivo *soft*, e dois tipos de

links: meio-fim¹ e *links* de decomposição de tarefa² [YU95b]. O Modelo de Razão Estratégica determina pela análise meio-fim¹ como as metas podem ser cumpridas através das contribuições de outros atores.

Sugerimos que o artefato Modelo de Razão Estratégica faça parte do documento de especificação de requisitos do projeto, pois é na fase de requisitos finais que os requisitos do sistema são identificados. Nesse documento o engenheiro de requisitos deverá detalhar o papel de cada ator, as dependências entre eles e os objetivos, tarefas, recursos e objetivos *soft* do sistema. Os atores, dependências, objetivos, tarefas, recursos e objetivos *soft* devem receber uma numeração única permitindo que eles sejam identificados em qualquer fase do projeto.

Na próxima seção aplicamos as diretrizes de Toranzo na fase Requisitos Finais.

5.5.1 Aplicando Diretrizes 1 a 12 na Fase de Requisitos Finais

O resultado da aplicação das diretrizes 1 a 8 que identificam as informações de cada um dos segmentos (externo, organizacional, gerencial e desenvolvimento) são apresentados na Tabela 5-6.

Modelo Razão Estratégica	Segmentos de Informação → Classes
Dependência de Meta / Objetivo	Organizacional → Informação Organizacional
Dependência de Meta / Objetivo (<i>soft</i>)	Gerencial → Requisito (não-funcional)
Dependência de Recurso	Gerencial → Requisito
Dependência de Tarefa	

Tabela 5-6 Requisitos Finais: Aplicação das diretrizes para identificação das classes dos segmentos de informação

No Modelo de Razão Estratégica são apresentados os objetivos, recursos, tarefas e as dependências do sistema levando-se em consideração duas visões do sistema: visão de atores externos e visão interna do sistema. A partir da representação dessas duas visões do sistema, foram identificadas classes em apenas dois segmentos de informação: organizacional e gerencial, sendo o último mapeado diretamente nos requisitos funcionais do sistema. As dependências de objetivo podem, dependendo do domínio da aplicação, ser mapeadas para requisitos não-funcionais.

Deve-se atentar à diferença entre o conceito de dependência de tarefa encontrado em Tropos [CAS02a] (Modelos de Dependência Estratégica e Razão Estratégica) e a classe TAREFA presente na proposta de Toranzo [TOR02a]. O primeiro diz respeito às tarefas que o sistema ou qualquer ator externo deve realizar para satisfazer algum objetivo sendo, portanto mapeadas para requisitos do

¹ *Link* meio-fim: o Modelo de Raciocínio Estratégico utiliza vários tipos de *links* meio-fim. O fim pode ser um objetivo, tarefa, recurso, ou *softgoal*, enquanto que o meio é usualmente uma tarefa. Na notação gráfica, a seta aponta do meio para o fim.

² *Links* de decomposição de tarefa: uma tarefa é modelada em termos da sua decomposição em subcomponentes. Estes componentes podem ser objetivos, tarefas, recursos, e / ou *softgoals*.

sistema. Já a classe TAREFA do modelo de Toranzo refere-se às tarefas executadas pela equipe de desenvolvimento de software visando à implementação dos requisitos do sistema, sendo assim relacionadas ao segmento de informação de desenvolvimento.

Os resultados da aplicação das diretrizes 9 a 12 estão apresentados na Tabela 5-7.

Diretriz	Resultado da aplicação
Diretriz 9 - Remover as classes de informação irrelevantes.	Não foram identificadas classes irrelevantes.
Diretriz 10 - Integrar as classes com o mesmo significado.	Não foi necessária a integração das classes.
Diretriz 11 - Integrar novas classes.	Não foi necessário.
Diretriz 12 - Organizar as classes.	O relacionamento generalização não foi utilizado como sugerido na proposta de Toranzo.

Tabela 5-7 Requisitos Finais: Resultado da aplicação das diretrizes 9 a 12

Na próxima seção apresentamos a fase Projeto Arquitetural de Tropos.

5.6 Projeto Arquitetural

A arquitetura de um sistema consiste de um modelo da estrutura do sistema que descreve como os componentes trabalham em conjunto. Tropos define estilos arquiteturais organizacionais que são representados inicialmente através da notação i^* [YU94]. Entre os estilos organizacionais identificados, existem [KOL02]:

- **União Estratégica:** envolve acordos entre dois ou mais parceiros principais para obter benefícios derivados da operação em uma larga escala, com investimentos limitados e custos de manutenção mais baixos;
- **Estrutura em 5:** consiste dos componentes típicos (estratégicos e logísticos) geralmente encontrados em várias organizações;
- **Integração Vertical:** consiste de atores comprometidos em atingir metas ou realizar tarefas relacionadas em estágios diferentes de um processo de produção;
- **Apropriação:** envolve a incorporação de agentes externos na estrutura ou no comportamento tomador de decisão ou conselheiro de uma organização iniciante;

¹ Análise meio-fim: oferece suporte ao processo de análise de vulnerabilidades e oportunidades, como também de sugestão, exploração e avaliação de decisões alternativas para atingir uma determinada meta.

- **Comprimento de Braço:** implica em acordos entre atores independentes e competitivos, porém parceiros;
- **Tomada de Controle:** envolve a delegação total de autoridade e gerenciamento de dois ou mais parceiros para um único ator, semelhante ao estilo União Estratégica;
- **Pirâmide:** estrutura de autoridade hierárquica exercida com limites organizacionais;
- **Contratação Hierárquica:** identifica mecanismos de coordenação que combinam características do acordo Comprimento de Braço com aspectos de autoridade Pirâmide;
- **Oferta:** envolve mecanismos de competitividade e os atores se comportam como se estivessem tomando parte em um leilão;
- **Estrutura Plana:** não possui estrutura fixa e assume que nenhum ator tenha controle sobre outro.

Cada um dos estilos organizacionais mencionados acima possui uma representação na metodologia Tropos utilizando-se o *framework* i* [KOL02].

A fase projeto arquitetural é composta de duas atividades, a primeira é a seleção do estilo arquitetural adequado usando como um dos critérios os requisitos não-funcionais (meta-soft) identificados na fase anterior (Requisitos Finais). A segunda atividade consiste em incluir novos atores e dependências nos modelos de dependência estratégica e razão estratégica.

Para apoiar o processo de seleção dos estilos organizacionais, Tropos utiliza o *framework* NFR [CHU00]. A Tabela 5-8 resume o catálogo de correlação entre os estilos organizacionais e os requisitos não-funcionais ou atributos de qualidade ([KOL01a], [KOL01b]). As seguintes notações usadas pelo *framework* NFR (*help*, *make*, *hurt*, *break*) modelam, respectivamente, as contribuições: parcial / positiva, suficiente / positiva, parcial / negativa e suficiente / negativa.

Catálogo de correlação	Previsibilidade	Segurança	Adaptabilidade	Coordenação	Cooperação	Disponibilidade	Integridade	Modularidade	Agregabilidade
<i>Estrutura Plana</i>	<i>Break</i>	<i>Break</i>	<i>Hurt</i>			<i>Help</i>	<i>Help</i>	<i>Make</i>	<i>Hurt</i>
<i>Estrutura em 5</i>	<i>Help</i>	<i>Help</i>		<i>Help</i>	<i>Hurt</i>	<i>Help</i>	<i>Make</i>	<i>Make</i>	<i>Make</i>
<i>Pirâmide</i>	<i>Make</i>	<i>Make</i>	<i>Help</i>	<i>Make</i>	<i>Hurt</i>	<i>Help</i>	<i>Break</i>	<i>Hurt</i>	
<i>União Estratégica</i>	<i>Help</i>	<i>Help</i>	<i>Make</i>	<i>Help</i>	<i>Hurt</i>	<i>Make</i>		<i>Help</i>	<i>Make</i>
<i>Oferta</i>	<i>Break</i>	<i>Break</i>	<i>Make</i>	<i>Hurt</i>	<i>Make</i>	<i>Hurt</i>	<i>Break</i>	<i>Make</i>	
<i>Tomada de Controle</i>	<i>Make</i>	<i>Make</i>	<i>Hurt</i>	<i>Make</i>	<i>Break</i>	<i>Help</i>		<i>Help</i>	<i>Help</i>
<i>Comprimento de Braço</i>	<i>Hurt</i>	<i>Break</i>	<i>Help</i>	<i>Hurt</i>	<i>Make</i>	<i>Break</i>	<i>Make</i>	<i>Help</i>	
<i>Contratação Hierárquica</i>			<i>Help</i>	<i>Help</i>	<i>Help</i>	<i>Help</i>		<i>Help</i>	<i>Help</i>
<i>Integração Vertical</i>	<i>Help</i>	<i>Help</i>	<i>Hurt</i>	<i>Help</i>	<i>Hurt</i>	<i>Help</i>	<i>Break</i>	<i>Break</i>	<i>Break</i>
<i>Apropriação</i>	<i>Hurt</i>	<i>Hurt</i>	<i>Make</i>	<i>Make</i>	<i>Help</i>	<i>Break</i>	<i>Hurt</i>	<i>Break</i>	

Tabela 5-8 Catálogo de correlação

Um documento descrevendo a arquitetura selecionada para o sistema deve ser gerado na fase projeto arquitetural.

Na próxima seção aplicamos as diretrizes propostas por Toranzo na fase Projeto Arquitetural de Tropos.

5.6.1 Aplicando Diretrizes 1 a 12 na Fase de Projeto Arquitetural

A primeira atividade da fase Projeto Arquitetural é a seleção do estilo organizacional que será utilizado no sistema. Para essa etapa sugerimos a utilização do Modelo de Raciocínio proposto por Toranzo, para armazenamento de decisões tomadas no decorrer do desenvolvimento do sistema (vide seção 5.2.3).

Recomendamos que as informações que foram utilizadas para a tomada de decisão da escolha do estilo organizacional sejam armazenadas para que os argumentos, posições, decisão, e *stakeholders* envolvidos possam ser futuramente consultados. As informações identificadas são do segmento organizacional e podem ser mapeadas diretamente para as classes do modelo de Toranzo de acordo com a Tabela 5-9.

Mapeamento	Classe do modelo Toranzo
Estilos organizacionais	POSIÇÃO
Seleção do estilo organizacional	ASSUNTO
Estilo organizacional selecionado	DECISÃO
Requisitos não-funcionais do sistema / Notação NFR	ARGUMENTO
Catálogo de correlação	RESTRIÇÃO

Tabela 5-9 Mapeamento para o modelo de raciocínio

Para a segunda atividade da fase projeto arquitetural que consiste em incluir novos atores e dependências nos modelos de dependência estratégica e razão estratégica, valem os mesmos resultados identificados na seção 5.5.1 representados na Tabela 5-10.

Modelo Razão Estratégica	Segmentos de Informação → Classes
Dependência de Meta / Objetivo	Organizacional → Informação Organizacional
Dependência de Meta / Objetivo (soft)	Gerencial → Requisito (não-funcional)
Dependência de Recurso	Gerencial → Requisito
Dependência de Tarefa	

Tabela 5-10 Projeto Arquitetural: Aplicação das diretrizes para identificação das classes dos segmentos de informação

Os resultados da aplicação das diretrizes 9 a 12 estão apresentados na Tabela 5-11.

Diretriz	Resultado da aplicação
Diretriz 9 - Remover as classes de informação irrelevantes.	Não foram identificadas classes irrelevantes.
Diretriz 10 - Integrar as classes com o mesmo significado.	Não foi necessário integrar as classes.
Diretriz 11 - Integrar novas classes.	Não foi necessário
Diretriz 12 - Organizar as classes.	O relacionamento generalização não foi utilizado como sugerido na proposta de Toranzo.

Tabela 5-11 Projeto Arquitetural: Resultado da aplicação das diretrizes 9 a 12

Na próxima seção apresentamos a fase Projeto Detalhado da metodologia Tropos.

5.7 Projeto Detalhado

O objetivo da fase Projeto Detalhado é introduzir detalhes adicionais para cada componente da arquitetura do sistema, isto é, a comunicação entre os atores e seus comportamentos. Para suportar essa fase, Tropos propõe a utilização de extensões da UML, como a AUML – *Agent Unified Modelling Language* ([BAU01], [ODE00b]) proposta pela FIPA – *Foundation for Physical Intelligent Agents* e a OMG – *Agent World Group*.

5.7.1 Aplicando Diretrizes na Fase de Projeto Detalhado

O resultado da aplicação das diretrizes 1 a 8 na fase projeto detalhado traz informações apenas do segmento de desenvolvimento (Tabela 5-12), pois nessa fase a única atividade é detalhar os estilos organizacionais com base na comunicação entre os atores, atividade que diz respeito ao funcionamento do sistema a ser desenvolvido.

Além desse resultado, as informações geradas durante esta fase como possíveis diagramas de comunicação, diagramas de sequência e colaboração, bem como diagramas de atividade e

quaisquer outros artefatos de projeto podem ser armazenados de acordo com o sub-modelo de projeto proposto por Toranzo.

Segmentos de Informação → Classes
Desenvolvimento → Diagrama

Tabela 5-12 Projeto Detalhado: Aplicação das diretrizes para identificação das classes dos segmentos de informação

As diretrizes de 9 a 12 não foram aplicadas porque apenas uma classe foi sugerida para inclusão no modelo de rastreamento de requisitos, a classe diagramas.

Na próxima seção apresentamos a fase Implementação da metodologia Tropos.

5.8 Implementação

A fase de implementação consiste na codificação dos componentes identificados na fase de projeto arquitetural e detalhados na fase de projeto detalhado. Nesta fase, Tropos propõe a utilização de linguagens de comunicação entre agentes, tais como FIPA-ACL [LAB99] ou KQML [FIN91], mecanismos de transporte de mensagens e outros conceitos e ferramentas. As informações geradas nessa fase fazem parte do segmento de desenvolvimento e de acordo com as diretrizes 1 a 8 devem ser armazenadas na classe Programa do sub-modelo de projeto proposto por Toranzo como apresentado na Tabela 5-13.

Segmentos de Informação → Classes
Desenvolvimento → Programa

Tabela 5-13 Implementação: Aplicação das diretrizes para identificação das classes dos segmentos de informação

As diretrizes de 9 a 12 não foram aplicadas porque apenas uma classe foi sugerida para inclusão no modelo de rastreamento de requisitos, a classe programa.

Na próxima seção apresentamos a aplicação das diretrizes 13 a 16 que devem ser utilizadas independentemente dos segmentos de informação.

5.9 Aplicação das diretrizes 13 a 16

Aplicando-se as diretrizes 13 a 16, foram identificadas classes em todos os segmentos de informação sugeridos por Toranzo (externo, organizacional, gerencial e desenvolvimento). Para a aplicação da diretriz 13, que estabelece os relacionamentos entre os elementos do modelo, utilizamos o rastreamento vertical e o horizontal. O Rastreamento Vertical representado na Figura

5-7 é o rastreamento entre fases prévias ou subseqüentes da vida de um requisito [GOT95]. As fases foram mapeadas nessa dissertação para os segmentos de informação para visualizarmos a vida do requisito levando em consideração a qual segmento de informação ele pertence, isto é, em qual segmento de informação o requisito foi identificado.

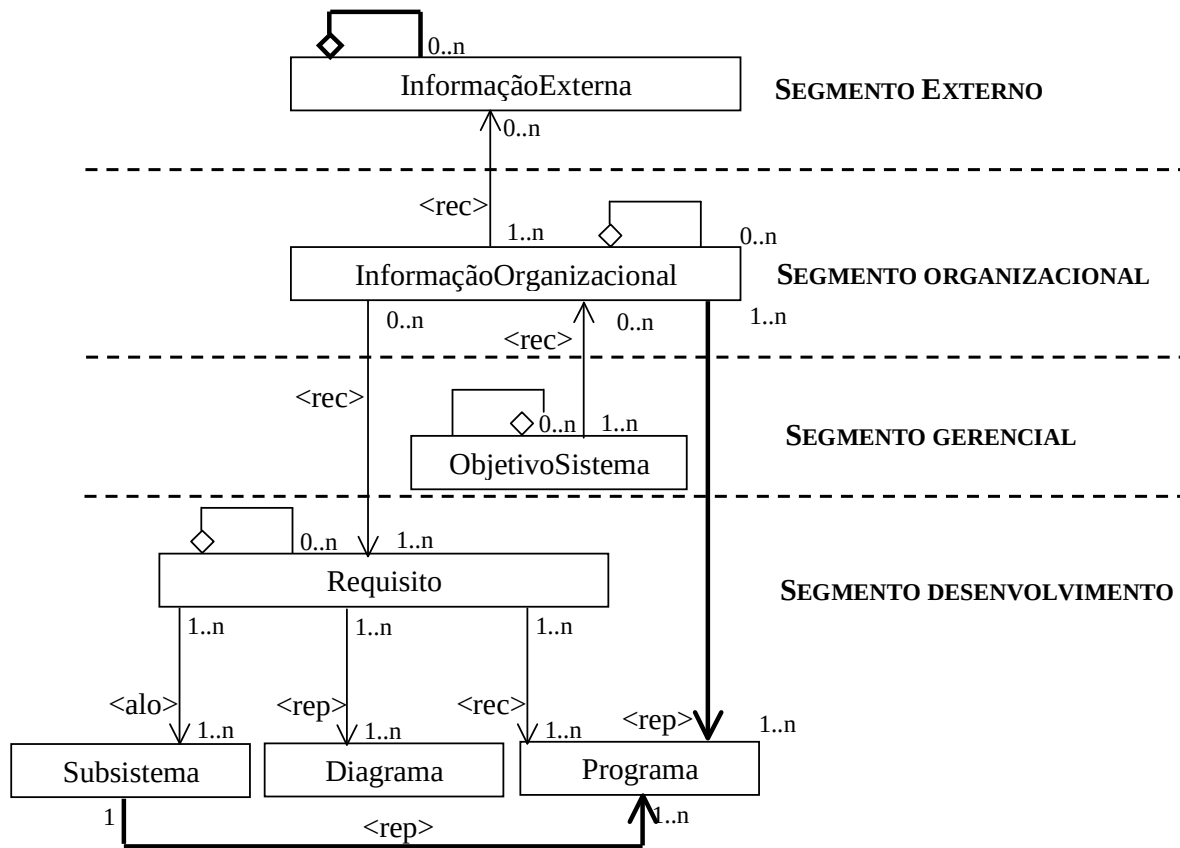


Figura 5-7 Rastreamento vertical

Os relacionamentos representados na Figura 5-7 por uma linha com espessura maior são aqueles que foram incluídos nessa dissertação.

Para a metodologia Tropos identificamos a necessidade de inclusão de um relacionamento entre as classes *InformaçãoOrganizacional* e *Programa* pois a metodologia trabalha com o conceito de estilos de arquitetura organizacionais nas fases Projeto Arquitetural e Projeto Detalhado que antecedem a fase Implementação, isto é, as informações organizacionais identificadas nas fases Projeto Arquitetural e Projeto Detalhado são implementadas pelos programas na fase Implementação. O tipo de relacionamento utilizado entre as classes *InformaçãoOrganizacional* e *Programa* é o de recurso. Acrescentamos um relacionamento de representação entre as classes *Subsistema* e *Programa*, pois subsistemas são implementados pelos programas (código fonte). Um auto-relacionamento de agregação foi criado para a classe *InformaçãoExterna*, pois uma informação externa pode gerar zero ou mais informações externas.

Visando minimizar a quantidade de informações a serem manipuladas e facilitar o entendimento do modelo de rastreamento, excluímos o atributo “nome” dos relacionamentos proposto no trabalho de Toranzo [TOR02a]. Para a representação no diagrama da Figura 5-7 permanecemos com os atributos: tipo do relacionamento e multiplicidade. A multiplicidade foi alterada em alguns relacionamentos, como por exemplo, no relacionamento entre as classes *InformaçãoExterna* e *InformaçãoOrganizacional*, pois para que uma informação externa seja relevante ela deve estar relacionada com uma ou mais informações organizacionais e não com zero ou mais como apresentado na proposta original de Toranzo. Todos os relacionamentos apresentados na Figura 5-7 que apresentam multiplicidade 1..n foram alterados em relação à proposta de Toranzo onde consta multiplicidade 0..n.

A metodologia Tropos é orientada a requisitos e possui uma natureza organizacional. Tarefa e Teste são atividades que ainda não são cobertas pela metodologia como apresentado na seção 4.7. Portanto, não foram identificadas nas classes dos segmentos de informação.

O rastreamento Horizontal representado na Figura 5-8 é o rastreamento entre versões ou variações de um requisito dentro de uma fase particular. As fases foram mapeadas para cada uma das fases da metodologia Tropos. No rastreamento Horizontal são apresentados os mesmos elementos já apresentados no rastreamento Vertical (exceto os elementos do Modelo de Raciocínio), mas com um enfoque diferente. Enquanto no rastreamento Vertical a preocupação é na origem do requisito, isto é, em qual segmento de informação ele foi identificado, o rastreamento Horizontal apresenta a vida do requisito dentro das fases da metodologia Tropos. A proposta de representação dos relacionamentos através dos rastreamentos Vertical e Horizontal é uma recomendação do nosso trabalho, isto é, não faz parte da proposta original de Toranzo [TOR02a].

As classes *InformaçãoExterna* e *InformaçãoOrganizacional* podem estar presentes nas fases Requisitos Iniciais, Requisitos Finais e Projeto Arquitetural da metodologia Tropos. Por esse motivo, a primeira camada da Figura 5-8 representa as três fases.

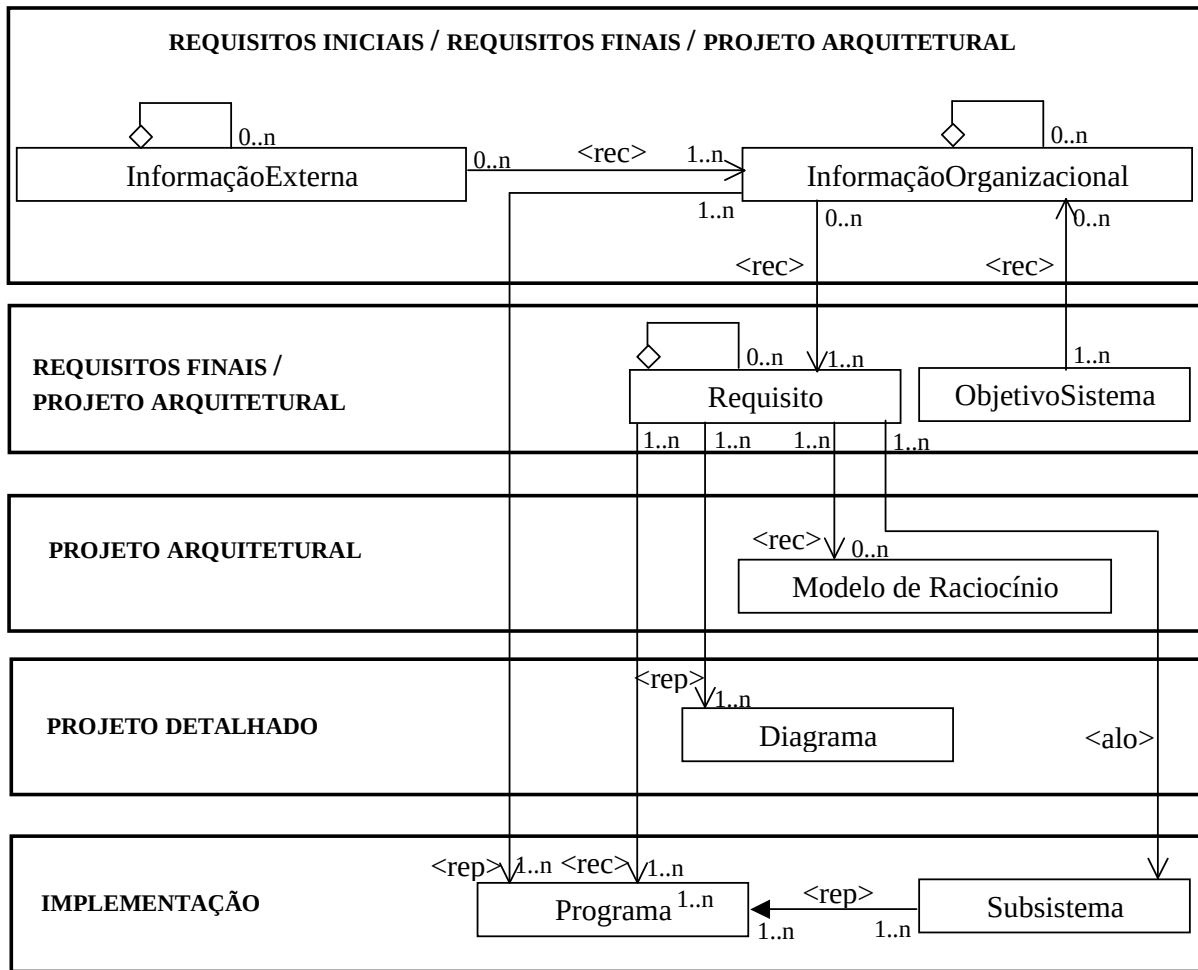


Figura 5-8 Rastreamento horizontal

Para facilitar o entendimento, o modelo de raciocínio foi incluído no rastreamento horizontal sem detalhamento das classes que estão representadas na Figura 3-6. Como esse Modelo de Raciocínio foi utilizado apenas na fase projeto arquitetural, ele é alocado nessa fase, mas nada impede que ele seja utilizado nas demais fases da metodologia Tropos. Incluímos o relacionamento entre o Modelo de Raciocínio e a classe Requisito, pois na seleção do estilo organizacional os requisitos não-funcionais (objetivos *soft*) são obrigatoriamente utilizados. O rastreamento horizontal será representado através de matrizes na próxima seção.

Conforme a diretriz 14 devemos identificar os atributos das classes do modelo que serão rastreados. Para projetos de software que forem apoiados por uma ferramenta para gerenciamento de requisitos (ex: Doors, Requisite Pro, Together, etc.), sugerimos que todos os atributos propostos por Toranzo [TOR02a] sejam utilizados. Para essa dissertação vamos utilizar apenas os atributos nome e descrição, supondo que uma ferramenta não será utilizada fazendo com que a utilização de todos os atributos cause uma sobrecarga para as atividades de rastreamento de requisitos.

De acordo com a diretriz 15 deve-se elaborar as matrizes de relacionamento entre as classes. As convenções para representar um relacionamento do modelo de rastreamento em uma matriz são as seguintes [TOR02b]:

- O tipo da associação representada na matriz é apresentado na parte superior esquerda da matriz através de um rótulo. Por exemplo, o rótulo “<rec>” representa uma associação do tipo recurso.
- Junto com o tipo da associação é apresentado o símbolo de uma seta (“→” ou “←”) para identificar as instâncias da classe destino na associação. Se for usado o símbolo “→”, então as instâncias declaradas na primeira fila da matriz são instâncias da classe destino. Caso contrário (“←”) as instâncias declaradas na primeira coluna da matriz são instâncias da classe origem.
- Na primeira coluna e primeira fila das matrizes de rastreamento usaremos expressões para identificar as instâncias de uma classe de acordo com as abreviaturas previamente apresentadas (Tabela 5-5, Tabela 5-7, Tabela 5-11) e a descrição das classes (opcional).
- Nas interseções entre linhas e colunas será apresentado o grau de dependência entre as classes, se o relacionamento existir. Este grau de dependência pode ser expresso em forma quantitativa ou qualitativa. Utilizaremos uma forma qualitativa para representação do grau de dependência: A (alta dependência), M (média dependência) e B (baixa dependência). Por exemplo, alguns requisitos de um sistema podem ter um alto grau de dependência com algumas leis (ex: CPMF) que, se mudarem, podem ter um forte impacto sobre os requisitos e o sistema que os implementa.

Na próxima seção apresentamos as matrizes, recomendadas no nosso trabalho, que devem ser elaboradas durante o processo de desenvolvimento de software levando-se em consideração cada uma das fases da metodologia Tropos.

5.10 Matrizes de Relacionamento Sugeridas Para a Metodologia Tropos

Para representação das matrizes de relacionamento não utilizamos dados reais ou fictícios. No estudo de caso apresentado no próximo capítulo algumas dessas tabelas serão utilizadas.

A utilização das matrizes recomendadas no nosso trabalho, para cada uma das fases da metodologia Tropos, irá depender do sistema a ser desenvolvido e das informações identificadas em cada uma das fases, isto é, pode acontecer de uma determinada matriz não ser utilizada na fase Requisitos Finais porque no processo de desenvolvimento de software não foram identificadas as classes que compõem essa matriz. As informações apresentadas nas matrizes a seguir são meramente explicativas. No próximo capítulo serão apresentadas matrizes contendo dados do estudo de caso.

Levando-se em consideração o rastreamento Horizontal apresentado na Figura 5-8, recomendamos que a seguinte matriz seja elaborada na fase Requisitos Iniciais da metodologia Tropos:

<rec> ←	[IEXT1] Descrição 1	[IEXT2] Descrição 2	[IEXT3] Descrição 3
[IORG1] Descrição 1	A	B	M
[IORG2] Descrição 2	A	A	M
[IORG3] Descrição 3	M	A	B

Tabela 5-14 Matriz: Informação Externa x Informação Organizacional

Na Tabela 5-14 representamos o relacionamento *recurso* entre as classes *InformaçãoExterna* e *InformaçãoOrganizacional*, isto é, as informações organizacionais que foram identificadas através das informações externas.

Para a fase Requisitos Finais recomendamos a elaboração tanto da matriz representada na Tabela 5-14 como das matrizes representadas nas Tabelas 5-14 e 5-15.

<rec> ←	[IORG1] Descrição 1	[IORG2] Descrição 2	[IORG3] Descrição 3
[REQ1] Descrição 1	A	B	M
[REQ2] Descrição 2	A	A	M
[REQ3] Descrição 3	M	A	B

Tabela 5-15 Matriz: Informação Organizacional x Requisito

Na Tabela 5-15 representamos o relacionamento *recurso* entre as classes *InformaçãoOrganizacional* e *Requisito*, isto é, quais requisitos do sistema foram originados das informações organizacionais identificadas no sistema.

<rec> ←	[IORG1] Descrição 1	[IORG2] Descrição 2	[IORG3] Descrição 3
[OBJ1] Descrição 1	A	B	M
[OBJ2] Descrição 2	A	A	M
[OBJ3] Descrição 3	M	A	B

Tabela 5-16 Matriz: Informação Organizacional x Objetivos do Sistema

A Tabela 5-16 representa o relacionamento *recurso* entre as classes *Informação Organizacional* e *Objetivo Sistema*, isto é, as informações organizacionais que deram origem aos objetivos do sistema.

Para a fase Projeto Arquitetural da metodologia Tropos recomendamos a elaboração das matrizes representadas nas Tabelas 5-14, 5-15 e 5-16. Para o Modelo de Raciocínio utilizado na fase Projeto Arquitetural, recomendamos a construção de matrizes para todos os relacionamentos representados nesse modelo (vide seção 5.2.3). As Tabelas 5-17 e 5-18 são exemplos desses relacionamentos.

<rec>	[POS1] Descrição 1	[POS2] Descrição 2	[POS3] Descrição 3
[ARG1] Descrição 1	<suporta, M>	<contradiz, B>	<contradiz, M>
[ARG2] Descrição 2	<contradiz, A>	<suporta, A>	<suporta, M>
[ARG3] Descrição 3	<contradiz, M>	<suporta, A>	<suporta, B>

Tabela 5-17 Matriz: Posição x Argumentos

A Tabela 5-17 representa o relacionamento *recurso* entre as classes *Posição* e *Argumento*. Esses relacionamentos são representados no Modelo de Raciocínio de Toranzo por dois *links* entre as classes (*Posição* e *Argumento*), um representando argumentos que contradizem a posição e outro representando argumentos que suportam a posição. Para esse caso específico utilizamos o atributo “nome” para o relacionamento como proposto originalmente por Toranzo [TOR02a], enriquecendo assim a matriz de rastreabilidade.

<rec>	[ASS1] Descrição 1
[POS1] Descrição 1	A
[POS2] Descrição 2	A
[POS3] Descrição 3	M

Tabela 5-18 Assunto x Posição

Na Tabela 5-18 representamos o relacionamento *recurso* entre as classes *Posição* e *Assunto*, isto é, para um determinado assunto quais foram as posições recomendadas pelos *stakeholders*.

Para a fase Projeto Detalhado da metodologia Tropos recomendamos a elaboração da matriz de relacionamento representada na Tabela 5-19.

<rec>	[REQ1] Descrição 1	[REQ2] Descrição 2	[REQ3] Descrição 3
[DIAG1] Descrição 1	A	B	M
[DIAG2] Descrição 2	A	A	M
[DIAG3] Descrição 3	M	A	B

Tabela 5-19 Matriz: Requisito x Diagrama

A Tabela 5-19 representa o relacionamento *representa* entre as classes *Requisito* e *Diagrama*. Através desse relacionamento pode-se identificar os diagramas que representam cada um dos requisitos identificados durante o desenvolvimento do sistema.

Para a fase Implementação da metodologia Tropos recomendamos a elaboração das matrizes representadas nas Tabelas 5-20 e 5-2.

← <rec>	[IORG1] Descrição 1	[IORG2] Descrição 2	[IORG3] Descrição 3
[PROG1] Descrição 1	A	B	M
[PROG2] Descrição 2	A	A	M
[PROG3] Descrição 3	M	A	B

Tabela 5-20 Matriz: Informação Organizacional x Programa

A Tabela 5-20 representa o relacionamento *representa* entre as classes *Informação Organizacional* e *Programa*, relacionamento proposto no nosso trabalho. Esse relacionamento identifica quais programas implementaram as informações organizacionais identificadas durante o desenvolvimento do sistema.

← <rec>	[REQ1] Descrição 1	[REQ2] Descrição 2	[REQ3] Descrição 3
[PROG1] Descrição 1	M	B	A
[PROG2] Descrição 2	B	M	M
[PROG3] Descrição 3	A	A	B

Tabela 5-21 Matriz: Requisito x Programa

Na Tabela 5-21 representa o relacionamento *recurso* entre as classes *Requisito* e *Programa*, identificando assim, em quais programas cada um dos requisitos identificados durante o desenvolvimento do sistema foram implementados.

Na próxima seção apresentamos as considerações finais desse capítulo.

5.11 Considerações Finais

O objetivo principal da nossa proposta era verificar a adequação do modelo de rastreamento de requisitos proposto por Toranzo à metodologia Tropos, pois um dos pontos fortes do modelo proposto por Toranzo é a sua aplicação a qualquer tipo de paradigma de desenvolvimento.

Partindo-se das diretrizes propostas no modelo, verificamos que o modelo de rastreamento pode ser incorporado à metodologia Tropos, principalmente pelo tratamento de informações dos segmentos externo e organizacional, conceitos fortemente utilizados nessa metodologia.

Foram poucas as alterações propostas no modelo de Toranzo, dentre elas identificamos a necessidade de um relacionamento entre as classes Informação Organizacional e Programa, pois as

classes do segmento organizacional identificadas na fase Projeto Arquitetural são diretamente utilizadas na fase de Implementação, como também um relacionamento entre as classes Subsistema e Programa, pois subsistemas no desenvolvimento de software são implementados via programas (código fonte).

A multiplicidade de alguns relacionamentos foi alterada com o objetivo de melhor representar os relacionamentos entre as classes, assim como foi criado um auto-relacionamento de agregação na classe *InformaçãoExterna*.

O Modelo de Raciocínio proposto por Toranzo foi inserido no rastreamento horizontal através de um relacionamento com a classe requisito, pois a fase Projeto Arquitetural da metodologia Tropos utiliza os requisitos não-funcionais (meta-soft) identificados nas fases anteriores para a seleção do estilo organizacional a ser utilizado no sistema. Essa seleção deve ser armazenada nas classes do sub-modelo de raciocínio, pois se trata de uma decisão de suma importância para as fases posteriores do desenvolvimento.

Sugerimos, quando da representação gráfica ou matricial dos relacionamentos, a exclusão do nome do relacionamento, pois acreditamos que a informação do tipo do relacionamento já é suficiente para o entendimento da matriz, bem como para diminuir a complexidade e reduzir a quantidade de informações manipuladas. Como exceção temos os relacionamentos entre as classes *Posição* e *Assunto* do Modelo de Raciocínio que foram representados numa única matriz de rastreamento para melhor compreensão, sendo necessário então, para identificar os relacionamentos, nomeá-los (contradiz e suporta). O adorno dos relacionamentos proposto por Toranzo foi alterado para redução da complexidade, ficando apenas a representação qualitativa: A (alta dependência), M (média dependência) e B (baixa dependência).

Algumas classes do modelo de rastreamento de requisitos proposto por Toranzo não foram utilizadas. A classe *Tarefa* não é necessária porque a metodologia Tropos ainda não trata das fases de planejamento e acompanhamento do projeto. Já a classe *Teste* não é utilizada porque a metodologia ainda não especifica como planejar e / ou executar testes.

Pelo exposto acima, concluímos que o modelo de rastreamento de requisitos proposto por Toranzo pode ser incorporado à metodologia Tropos, podendo ser incluído como uma atividade paralela às fases de desenvolvimento dessa metodologia. Alguns exemplos dos benefícios da inclusão do rastreamento de requisitos na metodologia são: avaliação do impacto das mudanças; consistência entre os requisitos do cliente e o sistema implementado; facilidade de obtenção da origem do requisito, isto é, quais *stakeholders* propuseram os requisitos; facilidade na fase de manutenção do sistema.

Capítulo 6 – Estudo de Caso

Este capítulo fornece um estudo de caso para validar nossa proposta de extensão da metodologia Tropos com a inclusão de atividades relacionadas ao Rastreamento de requisitos. Para isso foi utilizado um sistema multi-agentes de comércio eletrônico desenvolvido com a utilização da metodologia Tropos.

6.1 Introdução

Este capítulo apresenta um estudo de caso com o objetivo de validar a proposta e ajudar na sua utilização. Nas próximas seções aplicamos nossa proposta ao estudo de caso de um comércio eletrônico para uma loja de mídia. Os Modelos de Dependência Estratégica e de Razão Estratégica que utilizamos nas primeiras fases do desenvolvimento do sistema foram extraídos de [CAS02a].

6.2 O Sistema de Comércio Eletrônico *Medi@*

O exemplo escolhido para validação da proposta é o de uma Loja de Mídia, que vende diferentes tipos de itens de mídia tais como livros, jornais, revistas, CDs de áudio, fitas de vídeo e assim por diante. Os clientes da Loja de Mídia (locais ou remotos) podem definir seu pedido através de um catálogo periodicamente atualizado que descreve os itens de mídia disponíveis na Loja de Mídia. Ela é abastecida com os mais recentes lançamentos pelo Produtor de Mídia e com os itens já cadastrados pelo Fornecedor de Mídia. Para aumentar sua participação no mercado, a Loja de Mídia decidiu abrir uma frente de vendas na Internet. Com a nova configuração, um cliente pode pedir itens à Loja de Mídia pessoalmente, por telefone ou através da Internet. O sistema foi chamado *Medi@* e estará disponível na rede mundial de computadores usando as facilidades de comunicação providas pela Cia de Telecomunicações. O sistema usará serviços de finanças providos pela Cia Bancária, especializada em transações on-line. O propósito básico do novo sistema é permitir que um cliente possa examinar os itens disponíveis no catálogo do *Medi@* e fazer pedidos via Internet.

Não há restrições acerca do registro ou sobre procedimentos de identificação para usuários do *Medi@*. Clientes potenciais podem localizar itens na loja on-line navegando no catálogo ou buscando-os no banco de dados de itens sem estarem previamente registrados. O catálogo agrupa itens de mídia do mesmo tipo em hierarquias e gêneros (ex, CDs de áudio são classificados como pop, rock, jazz, opera, mundo, música clássica, trilha sonora, etc.). Portanto, estes clientes podem navegar só na categoria de seu interesse. Um motor de busca on-line permite que os clientes com itens particulares em mente possam pesquisar por título, autor/artista e campos de descrição através da busca por palavras-chaves ou texto completo. Se o item não está disponível no catálogo, o cliente tem a opção de pedir que a Loja de Mídia solicite-o, desde que o cliente tenha as referências da editora/gravadora (ex. ISBN, ISSN) e se identifique (em termos de nome e número do cartão de crédito). Os detalhes sobre itens de mídia incluem título, categoria de mídia (ex. livro) e gênero (ex. ficção científica), autor/artista, descrição concisa, referências e informações internacionais de

editora/gravadora, data, custo e às vezes figuras (quando houver disponível). De acordo com a proposta Tropos, nesta dissertação o sistema Medi@ será desenvolvido ao longo das fases de Requisitos Iniciais, Requisitos Finais e Projeto Arquitetural, conforme apresentaremos nas seções a seguir. A fase de Implementação não será contemplada nessa dissertação.

Na próxima seção apresentamos o resultado da aplicação das diretrizes para a elaboração do modelo de rastreamento de requisitos na fase Requisitos Iniciais da metodologia Tropos.

6.3 Requisitos Iniciais: Aplicando diretrizes 1 a 12

Analizando o ambiente organizacional descrito acima conseguimos elementos suficientes para produzir um primeiro modelo i* de dependências estratégicas (Figura 6.1) para a Loja de Mídia [CAS02a]. Os atores deste ambiente são *Cliente*, *Loja de Mídia*, *Fornecedor de Mídia* e *Produtor de Mídia*. O *Cliente* depende da *Loja de Mídia* para alcançar sua meta: *Comprar Itens de Mídia*. A *Loja de Mídia* depende dos *Clientes* para a satisfação das metas *Aumentar Participação no Mercado* e *Satisfazer Clientes*. Já que o *dependum* *Satisfazer Clientes* e *Aumentar Fatia de Mercado* não podem ser definidos precisamente, são representados como meta-softs. O *Cliente* depende da *Loja de Mídia* para *Consultar Catálogo* (dependência de tarefa). Para o suprimento de itens de mídia de forma contínua (*Suprimento contínuo*) a *Loja de Mídia* depende do *Fornecedor de Mídia* (dependência de recurso). Para que a dependência *Continuidade do Negócio* seja atingida, o *Fornecedor de Mídia* depende da *Loja de Mídia*. Finalmente, espera-se que o *Produtor de Mídia* supra o *Fornecedor de Mídia* com *Pacotes de Qualidade*, também uma meta-soft.

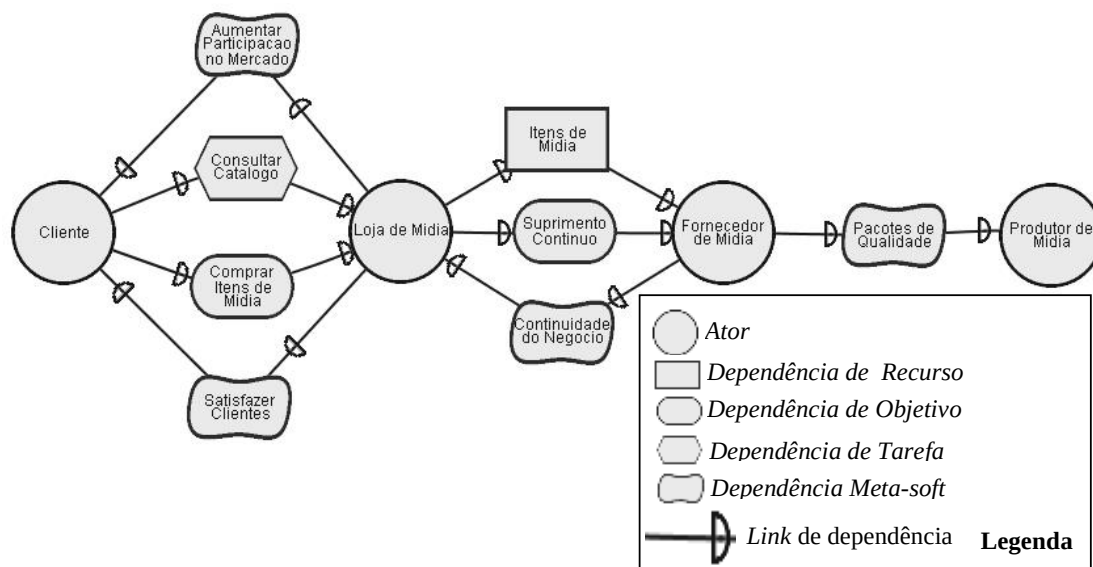


Figura 6-1 Modelo i* de Dependência Estratégica da Loja de Mídia

O Modelo de Razão Estratégica para a fase de Requisitos Iniciais está representado na Figura 6-2. O objetivo desse grafo é determinar através de uma análise meio-fim como cada uma das metas apresentadas na Figura 6-1 podem ser atendidas através das contribuições de outros atores. Assim, identificamos as tarefas e objetivos referentes à *Loja de Mídia* necessárias para o alcance dessas metas.

Para a satisfação das meta-softs *Aumentar Participação no Mercado* e *Satisfazer Clientes* é necessário que as seguintes metas sejam alcançadas: *Loja Funcionando* e *Melhorar Serviço*, respectivamente. Tarefas são seqüências ordenadas de passos que realizam alguma meta-soft. Tarefas podem ser decompostas em metas e / ou sub-tarefas, cujo cumprimento coletivo satisfaça a tarefa.

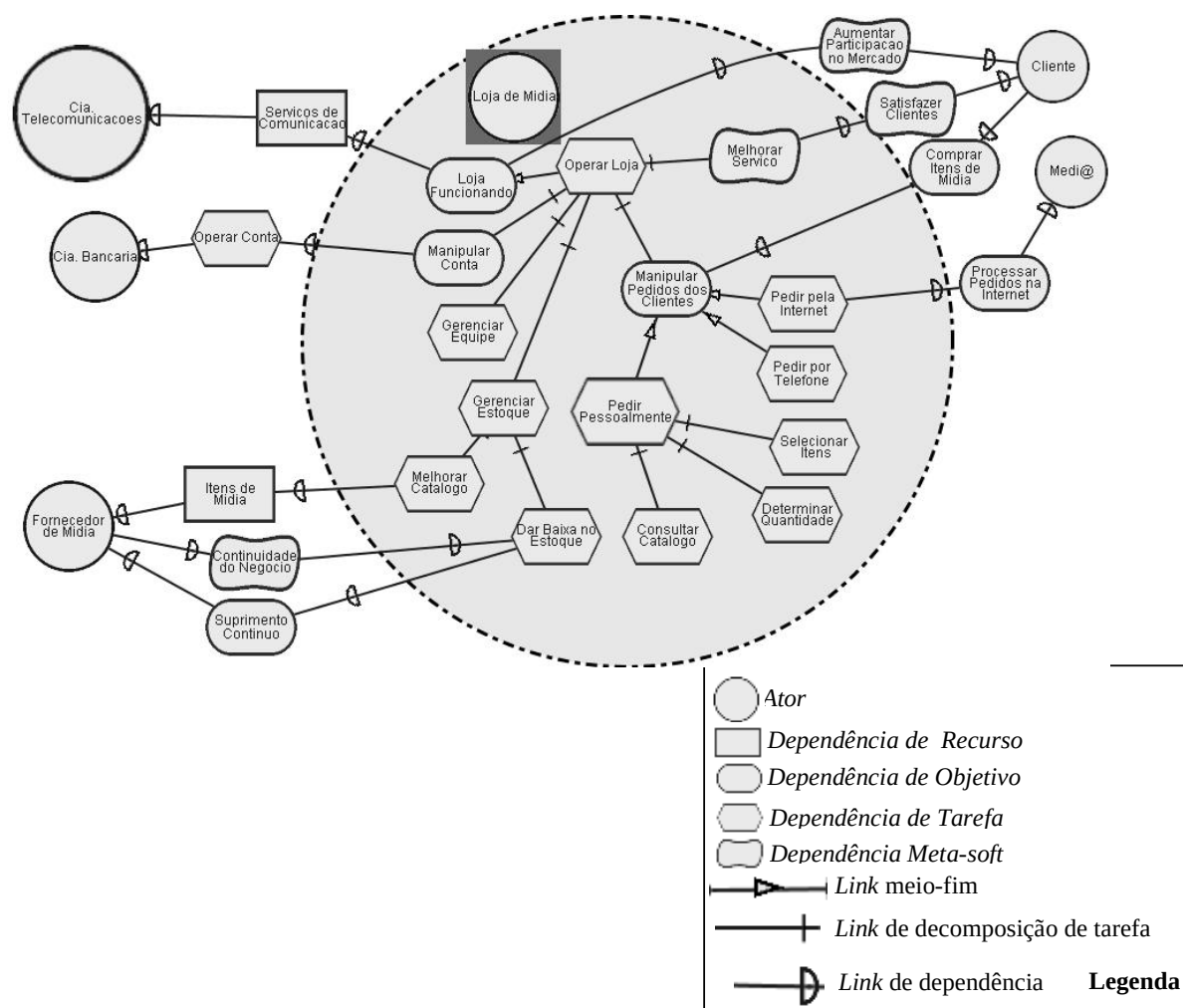


Figura 6-2 Modelo i* de Razão Estratégica da Loja de Mídia

Na Figura 6-2, a tarefa *Operar Loja* é decomposta tanto nas metas *Manipular Conta* e *Manipular Pedidos dos Clientes*, como nas tarefas *Gerenciar Equipe* e *Gerenciar Inventário*. Também há a meta-soft *Melhorar Serviço*. Sub-metas e sub-tarefas podem ser especificadas mais precisamente através de refinamentos. Por exemplo, a meta *Manipular Pedidos dos Clientes* é satisfeita através das tarefas *Pedir por Telefone* ou *Pedir pela Internet*, enquanto que a tarefa *Gerenciar Estoque* é realizada pelas tarefas *Dar Baixa no Estoque* e *Melhorar Catálogo*. Estas decomposições eventualmente permitem-nos identificar atores que podem realizar uma meta, executar uma tarefa ou fornecer algum recurso necessário para a *Loja de Mídia*. Tais dependências (vide Figura 6-2) são entre outras, as dependências de meta e recurso de *Fornecedor de Mídia* para fornecer, de maneira contínua, itens de mídia que são disponibilizados no catálogo e para venda dos produtos. As dependências de meta-soft são *Aumentar a Participação no Mercado* (implementando a loja virtual) e *Satisfazer os Clientes* (melhorando o serviço), enquanto a dependência de tarefa *Operar Conta* da *Cia Bancária* visa o controle de transações de negócio. Observe que na Figura 6-2 introduzimos um ator para lidar com a responsabilidade de tratar dos pedidos feitos pela Internet. Este ator representa o sistema a ser desenvolvido, chamado de *Medi@*, que será o foco da segunda fase de Tropos (Requisitos Finais), como veremos a seguir.

Utilizando o Modelo de Dependência Estratégica apresentado na Figura 6-1 mapeamos as classes que farão parte do modelo de rastreamento. Todas as classes identificadas no estudo de caso do sistema *Medi@* farão parte do modelo de rastreamento, pois trata-se de um estudo de caso pequeno em termos da quantidade de classes identificadas. Num estudo de caso real, alguns fatores deverão ser levados em consideração na escolha das classes que farão parte do modelo de rastreamento, tais como: custo, prazo, complexidade do sistema, domínio da aplicação e benefícios do rastreamento para o desenvolvimento do sistema.

Como mencionado em 5.4.1, a fase de Requisitos Iniciais trata de uma visão mais organizacional do sistema, identificando aspectos externos e objetivos macro do sistema. A Tabela 6-1 apresenta as classes por segmento de informação para facilitar o entendimento.

Informação Externa	
IEXT1	Pacotes de qualidade (meta-soft)
Informação Organizacional	
IORG1	Aumentar Participação no Mercado (meta-soft)
IORG2	Satisfazer Clientes (meta-soft)
IORG3	Suprimento contínuo (meta)

IORG4	Continuidade do negócio (meta-soft)
Objetivo do Sistema	
OBJ1	Comprar Itens de mídia (meta)
OBJ2	Receber itens de mídia (meta)
Requisito	
RF1	Consultar catálogo (tarefa)
RF2	Gerenciar Estoque (tarefa)
RF3	Pedir pela Internet (tarefa)

Tabela 6-1 Classes do modelo de rastreamento I

Como o Modelo de Razão Estratégica apresentado na Figura 6-2 detalha a informação organizacional Aumentar Participação no Mercado, identificamos mais classes do segmento organizacional como apresentado na Tabela 6-2.

Informação Organizacional	
IORG5	Loja funcionando
IORG6	Melhorar serviço
IORG7	Manipular conta
IORG8	Manipular pedidos dos clientes

Tabela 6-2 Informações Organizacionais - Classes do modelo de rastreamento II

Na próxima seção apresentamos o resultado da aplicação das diretrizes para elaboração do modelo de rastreamento na fase Requisitos Finais da metodologia Tropos.

6.4 Requisitos Finais: Aplicando diretrizes 1 a 12

Na fase Requisitos Iniciais, apresentamos a visão das necessidades da Loja de Mídia para que o sistema Medi@ pudesse ser desenvolvido. Na fase de Requisitos Finais o modelo de dependência estratégica é detalhado, sendo introduzido o sistema Medi@ como um ator (Figura 6-3).

Nesse modelo, identificamos as tarefas e metas dos atores identificados na fase anterior (Requisitos Iniciais) em relação ao sistema Medi@. Assim, o sistema Medi@ depende do ator Cia. Telecomunicações para que o recurso *Serviços de Internet* esteja disponível, bem como depende do ator Cia. Bancária para que a meta *Processar Transações de Dinheiro On-line* seja alcançada. O ator *Fornecedor de mídia* depende do ator Medi@ para satisfazer a meta *Encontrar as Novas Necessidades do Usuário*. O Cliente espera que o sistema Medi@ atenda as seguintes expectativas:

Disponibilidade e Segurança, como também deseja que os seguintes serviços (tarefas) estejam disponíveis: *Navegar no Catálogo*, *Buscar Palavra-Chave* e *Fazer Pedido*.

Um Modelo de Dependência Estratégica identifica os processos e os estrutura em termos de metas, tarefas e recursos. Também permite a avaliação e exploração de soluções alternativas. À medida que a análise de requisitos finais segue, é dado a Medi@ responsabilidades adicionais. O sistema também participará de várias dependências. Além disso, o sistema é decomposto em vários sub-atores que se encarregam destas responsabilidades. Esta decomposição e atribuição de responsabilidades são realizadas usando uma análise meio-fim (vide Figura 6-3). Esta figura foca no sistema em si, ao invés de focar em um *stakeholder* externo.

A Figura 6-3 apresenta uma tarefa raiz *Gerenciar Loja na Internet* que fornece apoio suficiente (*make*) [CHU00] à meta-soft *Aumentar Participação no Mercado*. Esta tarefa (*Gerenciar Loja na Internet*) é inicialmente refinada nas metas *Manipular Pedido na Internet* e *Manipular Busca de Item*. Também existem as metas-soft *Atrair Novos Clientes*, *Seguro* e *Disponível*, bem como a tarefa *Adaptação*. Para gerenciar pedidos na internet, a meta *Manipular Pedidos na Internet* é satisfeita através da tarefa *Carrinho de Compra* que é decomposta nas sub-tarefas *Selecionar Item*, *Adicionar Item*, *Confirmar Compra* e *Capturar Detalhes de Identificação*. Estas são as principais atividades de processo requisitadas para projetar um carrinho de compra operacional on-line [CON00].

A última tarefa (*Capturar Detalhes de Identificação*) é atingida através da sub-meta *Manipular pela Internet* que será satisfeita através das tarefas *Formulário de Pedido Seguro* e *Formulário de Pedido Padrão*. Para permitir o pedido de novos itens não listados no catálogo, *Selecionar Item* é refinado em duas sub-tarefas alternativas, uma dedicada a selecionar itens de catálogo (*Pegar Item Disponível*), e outra para pré-pedidos de produtos não disponíveis (*Fazer Pedido de Item não Disponível*). O apoio da meta-soft *Adaptação* é para satisfazer a meta-soft *Adaptável*. A meta-soft *Adaptação* é refinada em quatro sub-tarefas lidando com as atualizações de catálogo, evolução do sistema, atualizações de interface e monitoração do sistema. A meta *Manipular Busca de Item* deve ser alternativamente cumprida através das tarefas *Buscar no Banco de dados* ou *Consultar no Catálogo* levando em consideração a maneira de navegação dos clientes,

ou seja, uma busca por itens específicos em mente, usando funções de busca, ou simplesmente navegando no catálogo de produtos.

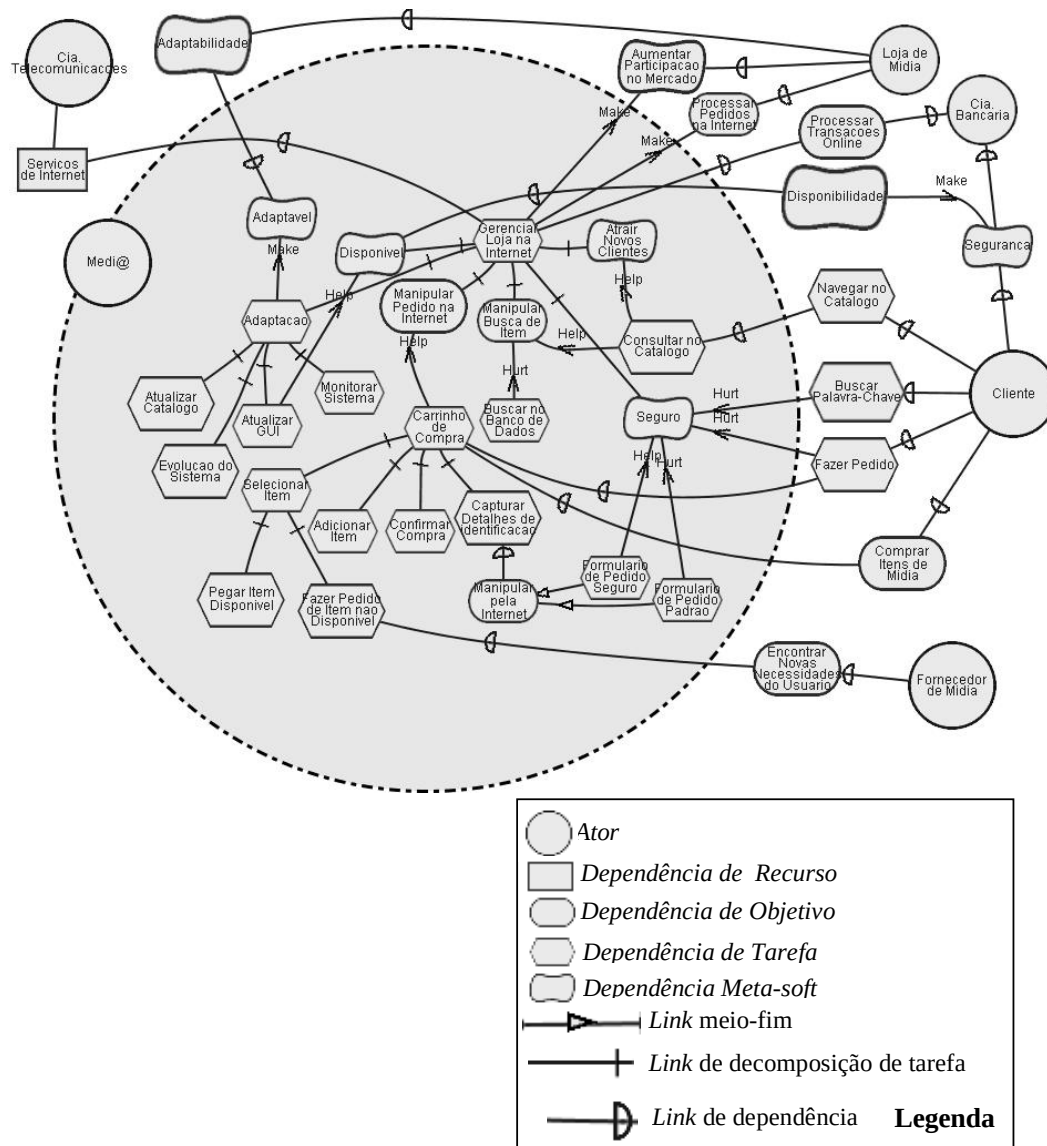


Figura 6-3 Modelo i* de Razão Estratégica do sistema Medi@

Em adição, como já apresentado, a Figura 6-3 introduz as contribuições de meta-soft para apoiar suficiente/parcial positivamente (respectivamente *make* e *help*) ou negativamente (respectivamente *break* e *hurt*) o modelo das metas-soft *Seguro*, *Disponível*, *Adaptabilidade* e *Aumentar Participação no Mercado*. O resultado desta análise meio-fim é um conjunto de atores (sistema e humano) que possuem algumas das dependências que foram postuladas.

No exemplo, foram deixadas em aberto, isto é, sem operacionalização ou refinamentos, três metas-soft (*Disponibilidade*, *Segurança*, *Adaptabilidade*) no modelo de requisitos finais, que serão utilizadas na próxima etapa da construção do software. Portanto, após a conclusão dos requisitos

finais, o desenvolvimento deverá partir para a fase de projeto da arquitetura do sistema, conforme veremos a seguir.

Através do Modelo de Dependência Estratégica apresentado na Figura 6-4, identificamos os requisitos funcionais e não-funcionais do sistema.

Requisitos não-funcionais	
RNF1	Disponibilidade
RNF2	Segurança
RNF3	Adaptabilidade
Requisitos funcionais	
RF4	Navegar no catálogo
RF5	Buscar palavra-chave
RF6	Fazer pedido
RF7	Processar pedidos na Internet
RF8	Processar transações de dinheiro on-line
RF9	Encontrar as novas necessidades do cliente

Tabela 6-3 Requisitos - Classes do modelo de rastreamento III

Requisitos não-funcionais	
RNF4	Evolução do sistema
Objetivos do sistema	
OBJ4	Atrair novos clientes
OBJ5	Gerenciar loja na Internet
Requisitos funcionais	
RF10	Manipular busca de item
RF11	Manipular pedido na Internet
RF12	Buscar no banco de dados
RF13	Carrinho de compra
RF14	Capturar detalhes de identificação
RF15	Começar compra
RF16	Adicionar item
RF17	Selecionar item
RF18	Pagar item disponível
RF19	Fazer pedido de item não disponível
RF20	Formulário de pedido seguro
RF21	Formulário de pedido padrão
RF22	Produzir estatísticas
RF23	Atualizar GUI
RF24	Atualizar catálogo
RF25	Monitorar sistema
RF26	Manipular via comunicação clássica
RF27	Pedido por telefone
RF28	Pedido por fax

Tabela 6-4 Classes do modelo de rastreamento IV

Para demonstrar a utilização da matriz de rastreabilidade, vamos primeiramente apresentar a matriz de relacionamento entre as informações organizacionais identificadas na seção 6.3 e os requisitos funcionais identificados nesta seção. No cruzamento das informações, são apresentados os graus de dependência: alto (A), médio (M) e baixo (B).

<rec>	[IORG1]	[IORG2]	[IORG3]	[IORG4]	[IORG5]	[IORG6]	[IORG7]	[IORG8]	[IORG9]	[IORG10]	[IORG11]	[IORG12]	[IORG13]	[IORG14]	[IORG15]	[IORG16]	[IORG17]
[RF4]	-	M	-	-	M	M	M	-	-	-	-	-	-	-	-	-	-
[RF5]	-	M	-	-	M	M	M	-	-	-	-	-	-	-	-	-	-
[RF6]	-	M	-	-	M	M	M	-	-	-	-	-	-	-	-	-	-
[RF7]	-	A	-	-	A	A	A	-	-	-	-	-	-	-	-	-	-
[RF8]	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	A
[RF9]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[RF10]	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[RF11]	B	-	-	-	-	-	-	-	-	-	-	-	-	A	-	-	-
[RF12]	B	B	-	-	B	B	B	-	-	-	-	-	-	-	-	-	-
[RF13]	B	B	-	-	B	B	B	-	A	-	-	-	-	A	-	-	-
[RF14]	B	B	-	-	B	B	B	-	M	-	-	-	-	A	-	-	-
[RF15]	B	B	-	-	B	B	B	-	M	-	-	-	-	A	-	-	-
[RF16]	B	B	-	-	B	B	B	-	M	-	-	-	-	A	-	-	-
[RF17]	B	B	-	-	B	B	B	-	M	-	-	-	-	A	-	-	-
[RF18]	B	B	-	-	B	B	B	-	M	-	-	-	-	A	-	-	-
[RF19]	B	-	-	-	-	-	-	-	M	-	-	-	-	A	-	-	-
[RF20]	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-	-	-
[RF21]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[RF12]	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[RF23]	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-
[RF24]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A	-
[RF25]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[RF26]	-	-	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-
[RF27]	-	-	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-
[RF28]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Tabela 6-5 Matriz de rastreabilidade: Informação Organizacional x Requisito funcional

Os principais requisitos do sistema são visualizados de forma explícita na matriz de rastreabilidade apresentada na Tabela 6-5 pela quantidade de relacionamentos com as informações organizacionais. Outra informação importante retirada da matriz é que existem informações organizacionais que não possuem relacionamento com os requisitos; indicando que essas informações organizacionais ainda precisam ser detalhadas em outros diagramas ou que não são de relevância para o desenvolvimento do sistema.

Na próxima seção apresentamos o resultado da aplicação das diretrizes para elaboração do modelo de rastreamento de requisitos, na fase Projeto Arquitetural da metodologia Tropos.

6.5 Projeto Arquitetural: Aplicando diretrizes 1 a 12

A primeira atividade durante o projeto arquitetural é a de selecionar o estilo arquitetural adequado usando como critério qualidades desejadas que foram identificadas nas fases anteriores (fase de Requisitos Iniciais e Requisitos Finais). Elas guiarão o processo de seleção do estilo arquitetural apropriado. Tropos utiliza o *framework* NFR [Chung00] para conduzir tal análise de qualidade.

Os atributos de qualidade (meta-soft) Segurança, Disponibilidade e Adaptabilidade foram identificados no modelo de requisitos finais (seção 6.4). Nesta fase de Projeto Arquitetural eles serão críticos, pois guiarão o processo de seleção do melhor estilo arquitetural (vide Figura 6-4). A análise envolve refinar estes atributos de qualidade (representadas como metas-soft) em sub-metas que são mais precisas e mais específicas. Após este refinamento é realizada uma avaliação dos estilos arquiteturais alternativos. Os estilos são representados como metas-soft operacionalizadas. As razões de projeto são representadas por metas-soft requeridas, desenhadas como nuvens tracejadas (*claim*). Estas podem, por exemplo, representar informação de conteúdo (tais como prioridades) a serem consideradas e apropriadamente refletidas no processo de tomada de decisão.

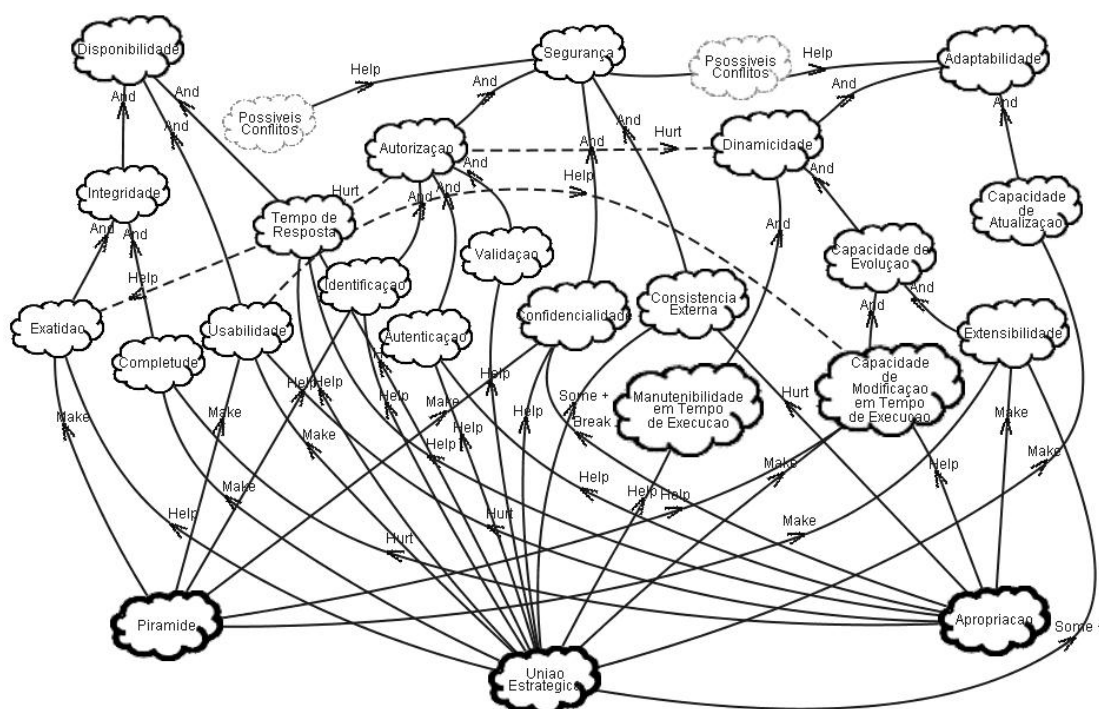


Figura 6-4 Grafo de Interdependência de Metas-soft

A análise apresentada na Figura 6-4 permite escolher o estilo arquitetural União Estratégica para o exemplo que está sendo apresentado. Em adição, atributos mais específicos foram identificados durante o processo de decomposição, tais como Integridade (*Exatidão*, *Completude*),

Usabilidade, Tempo de Resposta, Manutenibilidade em Tempo de Execução, Adaptabilidade, Dinamicidade e Capacidade de Atualização, Confidencialidade, Autorização (Identificação, Autenticação, Validação), Consistência. Eles precisam ser considerados na arquitetura do sistema. Mais detalhes sobre a seleção e o processo de decomposição de requisitos não funcionais podem ser encontrados em ([KOL01a], [KOL01b]).

O refinamento dos atributos de qualidade / requisitos funcionais identificou outros requisitos funcionais listados na Tabela 6-6.

Requisitos não-funcionais	
RNF5	Integridade
RNF6	Tempo de resposta
RNF7	Usabilidade
RNF8	Compleitude
RNF9	Autorização
RNF10	Consistência Externa
RNF11	Confidencialidade
RNF12	Identificação
RNF13	Validação
RNF14	Autenticação
RNF15	Dinamicidade
RNF16	Capacidade de Evolução
RNF17	Capacidade de Atualização
RNF18	Extensibilidade
RNF19	Manutenibilidade em Tempo de Execução
RNF20	Capacidade de Modificação em Tempo de Execução
RNF21	Exatidão

Tabela 6-6 Requisitos não-funcionais - Classes do Modelo de Rastreamento V

Os requisitos funcionais identificados na seção 6.4 e os requisitos não funcionais apresentados na Tabela 6-6 são utilizados na fase projeto Arquitetural para seleção do estilo organizacional que será utilizado no desenvolvimento do sistema. Como apresentado na seção 5.6.1, esses requisitos não funcionais serão mapeados para a classe Argumento do sub-modelo de raciocínio de Toranzo.

Os estilos arquiteturais apresentados na Figura 6-4 serão mapeados para a classe Posição do sub-modelo de raciocínio de acordo com a Tabela 6-7.

Posições	
POS1	Pirâmide
POS2	União Estratégica
POS3	Apropriação (Co-optation)

Tabela 6-7 Posições - Classes do Modelo de Rastreamento VI

O relacionamento entre os argumentos e as posições está representado na Tabela 6-9. Os *links* utilizados no framework NFR serão mapeados da seguinte maneira para a matriz de rastreabilidade:

<i>link</i>	Mapeamento
<i>Make</i>	<suporta, A>
<i>Help</i>	<suporta, M>
<i>Hurt</i>	<contradiz, M>
<i>Break</i>	<contradiz, A>

Tabela 6-8 Mapeamento dos links do framework para a matriz de rastreabilidade

O mapeamento foi realizado utilizando os relacionamentos *suporta* e *contradiz* entre as classes *Posição* e *Argumento* do sub-modelo de raciocínio (seção 5.2.3) e o grau de dependência entre as classes: A (alta dependência), M (média dependência) e B (baixa dependência). Para o significado dado ao relacionamento partimos do significado de cada um dos *links* do *framework*: *make* -> suficiente / positivo, *help* -> parcial / positivo, *hurt* -> parcial / negativo e *break* -> suficiente / negativo.

Para a elaboração da matriz a partir do *framework* NFR os seguintes passos devem ser seguidos:

1. Mapear requisitos não-funcionais folhas operacionalizados pelos estilos organizacionais. Os requisitos não-funcionais folhas da figura Figura 6-4 são: Exatidão, Completude, Usabilidade, Identificação, Autenticação, Validação, Confidencialidade, Consistência externa, Manutenibilidade em tempo de execução, Capacidade de modificação em tempo de execução e Extensibilidade. Assim, o relacionamento, por exemplo, entre o requisito não-funcional folha Exatidão e o estilo organizacional Pirâmide (*make*) é mapeado na matriz para <suporta, A>.

enquanto que o relacionamento desse requisito com o estilo organizacional União Estratégica (*help*) é mapeado para <suporta, M>.

2. Mapear o relacionamento entre cada um dos requisitos não-funcionais decompostos e os requisitos não-funcionais originados da decomposição usando a seguinte regra:
 - a. Verificar os *links* do requisito não-funcional originado da decomposição. Se o tipo dos *links* é igual (todos *make*, *help*, *hurt* ou *break*), o tipo do *link* entre esse requisito não-funcional e o requisito não-funcional decomposto será o igual a estes. Assim, o *link* entre os requisitos Autenticação e Autorização é mapeado para <suporta, M>, pois os *links* do requisito Autenticação são todos do tipo *help*.
 - b. Caso contrário, deve-se escolher o *link* que aparece em maior quantidade ou o *link* “mais fraco” caso a proporção seja igual. Assim, o *link* entre os requisitos Exatidão e Integridade é mapeado para <suporta, M>, pois os *links* do requisito Exatidão são *make* e *help*, isto é, a proporção é igual, logo o *link* “mais fraco” *help* é o mapeado.

A matriz abaixo apresenta todos os relacionamentos representados na Figura 6-4.

<rec> →	[POS1] Pirâmide	[POS2] União Estratégica	[POS3] Apropriação
[RNF1] Disponibilidade	<suporta, M>	<suporta, A>	<contradiz, M>
[RNF2] Segurança	<suporta, M>	<suporta, M>	<contradiz, A>
[RNF3] Adaptabilidade	<suporta, M>	<suporta, A>	<suporta, M>
[RNF4] Evolução do sistema	-	<suporta, A>	<suporta, M>
[RNF5] Integridade	<suporta, M>	<suporta, A>	<contradiz, M>
[RNF6] Tempo de resposta	-	<suporta, A>	<suporta, A>
[RNF7] Usabilidade	<suporta, A>	<suporta, A>	<contradiz, M>
[RNF8] Completude	-	<suporta, A>	<contradiz, M>
[RNF9] Autorização	<suporta, M>	<suporta, M>	<contradiz, A>
[RNF10] Consistência Externa	-	<suporta, M>	<contradiz, M>
[RNF11] Confidencialidade	<suporta, A>	<suporta, M>	<contradiz, A>
[RNF12] Identificação	<suporta, M>	<suporta, M>	-
[RNF13] Validação	-	<suporta, M>	-
[RNF14] Autenticação	-	<suporta, M>	<contradiz, A>
[RNF15] Dinamicidade	-	<suporta, M>	-
[RNF16] Capacidade de Evolução	-	<suporta, A>	<suporta, M>

[RNF17] Capacidade de Atualização	-	<suporta, A>	-
[RNF18] Extensibilidade	<suporta, A>	<suporta, A>	<suporta, A>
[RNF19] Manutenibilidade em Tempo de Execução	-	<suporta, M>	-
[RNF20] Capacidade de modificação em tempo de	<suporta, M>	<suporta, A>	<suporta, M>
[RNF21] Exatidão	<suporta, A>	<suporta, M>	-

Tabela 6-9 Matriz de rastreabilidade: Argumentos (Requisitos Não-Funcionais) x Posições

A Tabela 6-9 apresenta de maneira simples e clara as alternativas, argumentos e posições quando da seleção do estilo organizacional mais adequado ao sistema levando em consideração os requisitos não-funcionais do sistema. O armazenamento dessa informação evita que em fases posteriores do projeto os *stakeholders* não saibam dizer porque uma determinada arquitetura foi selecionada para o sistema, evitando retrabalho.

6.6 Considerações Finais

Apresentamos nesse capítulo a aplicação prática da nossa proposta de expansão da metodologia Tropos em relação às atividades de rastreamento de requisitos. Um estudo de caso referente a um sistema de comércio eletrônico foi tratado nesse capítulo. Como resultado, conseguimos verificar com mais clareza a utilização da nossa proposta.

Utilizamos as diretrizes propostas por Toranzo, para a elaboração de um modelo de rastreamento, identificando classes nos segmentos de informação (externo, organizacional, gerenciamento e desenvolvimento) para cada uma das fases da metodologia Tropos, exceto para a fase de Implementação que não é definida no estudo de caso.

Com a inclusão da atividade de Rastreamento de Requisitos na metodologia Tropos podemos avaliar melhor o impacto de uma mudança no sistema, por exemplo, se a Loja de Mídia decidisse alterar o sistema Media@, para que ele fosse um *site* de leilão. Com a utilização das matrizes de rastreabilidade podemos identificar quais requisitos, programas e diagramas devem ser alterados para implementar essa mudança.

No próximo capítulo, apresentamos nossas conclusões e apontamos os trabalhos futuros.

Capítulo 7 – Conclusões e Trabalhos Futuros

Este capítulo apresenta os resultados alcançados e as lições aprendidas com esta dissertação, como também aponta trabalhos futuros.

7.1 Contribuições do Trabalho

Para a execução desse trabalho foi realizada uma revisão geral da Engenharia de Requisitos destacando sua classificação e importância no processo de desenvolvimento de software, bem como uma descrição dos processos que envolvem a Engenharia de Requisitos e suas principais características. Também revisamos as atividades de elicitação, análise e negociação, documentação, verificação e validação e gerência de requisitos.

Apresentamos uma revisão geral de Rastreamento de Requisitos, inicialmente no contexto da Engenharia de Requisitos, destacando as pesquisas acadêmicas relevantes na área. Abordamos também a Engenharia de Software Orientada a Agentes, apresentando as motivações e a importância do uso desse novo paradigma. Destacamos as principais áreas de aplicação dos softwares orientados a agentes, bem como os desafios e obstáculos enfrentados atualmente no desenvolvimento desses softwares.

Apresentamos nesta dissertação uma proposta de expansão da metodologia de desenvolvimento de software orientada a agentes Tropos com a inclusão das atividades de Rastreamento de Requisitos nas suas fases de desenvolvimento.

Para isso, utilizamos o processo de desenvolvimento de um modelo de rastreamento de requisitos proposto por Toranzo [TOR02a] aplicando-o a cada uma das fases da metodologia Tropos. Esse processo é subdividido em três fases: Coleta de informação, Estruturação e construção do modelo de rastreamento e Validar passo do processo identificando propriedades sobre as classes.

Nesse trabalho identificamos que para os dois primeiros passos o processo adequou-se perfeitamente à metodologia Tropos, principalmente devido aos segmentos de informação externa e organizacional do modelo de rastreamento, conceitos fortemente utilizados na metodologia Tropos, e pelo fato da metodologia Tropos priorizar a fase de requisito. O terceiro passo do processo deverá ser utilizado quando da instanciación da metodologia no desenvolvimento de um sistema, haja vista esse passo depende do domínio da aplicação.

Foram utilizadas classes dos sub-modelos de Gerência de Requisitos e Projeto, bem como do Modelo de Raciocínio proposto por Toranzo.

Nossa proposta é original, pois até o presente momento não há outros trabalhos ou pesquisas de rastreamento de requisitos em metodologias orientadas a agentes. Algumas das contribuições da nossa proposta incluem:

- Validação do modelo intermediário para rastreamento de requisitos de Toranzo, bem como utilização do processo para elaboração do modelo de rastreamento na metodologia Tropos;
- Inclusão do Rastreamento de Requisitos numa metodologia de desenvolvimento de software orientada a agentes;
- Visualização da vida de um requisito tanto em relação aos segmentos de informação, como às fases da metodologia Tropos;
- Inclusão de relacionamentos nos sub-modelos propostos por Toranzo para adequação à metodologia Tropos;
- Validação da nossa proposta utilizando um estudo de caso.

7.2 Trabalhos Futuros

Para trabalhos futuros, temos como prioridade aplicar as diretrizes de Toranzo de uma maneira mais detalhada nas fases de Projeto Detalhado e Implementação da metodologia Tropos. Também pretendemos realizar mais estudos de caso com projetos reais utilizando a metodologia Tropos a fim de melhor adaptar a nossa proposta, identificando pontos de melhoria ao utilizá-la em sistemas de grande porte e construir indicadores da sua aplicabilidade em sistemas reais.

Para a aplicação completa do modelo de referência proposto por Toranzo, precisamos identificar outras oportunidades de utilização do Modelo de Raciocínio, que na nossa proposta ficou restrita à fase Projeto Arquitetural da metodologia Tropos.

Também pretendemos estudar ferramentas que oferecem suporte ao processo de rastreamento de requisitos a fim de verificar a adequação destas ferramentas a nossa proposta.

Para complementar nossa proposta, precisamos aplicar o processo de rastreamento de requisitos aos estudos mais recentes [SIL03] referentes ao detalhamento das fases da metodologia Tropos, bem como detalhar o mapeamento das informações identificadas na metodologia para os segmentos de informação (externo, organizacional, gerenciamento e desenvolvimento).

Dessa forma, esperamos oferecer um modelo de rastreamento completo que auxilie as pessoas envolvidas no desenvolvimento de um projeto em questões como: avaliação de impacto de mudança.

REFERÊNCIAS BIBLIOGRÁFICAS

-
- [AGE03] The Agentcities-Agentlink: **Agents for Commercial Applications**. Agent Technology Conference. Barcelona, Spain, 2003.
 - [ALE00] ALENCAR, F; CASTRO, J; CYSNEIROS; G and MYLOPOULOS, J. From Early Requirements Modeled by the i* Technique to Later Requirements Modeled in Precise UML, Em *Anais do III Workshop em Engenharia de Requisitos*, Rio de Janeiro, Brazil, 2000, pp. 92-109.
 - [ALF79] ALFORD, M. and LAWSON, J. **Software Requirements Engineering Methodology (Development)**. RADC-TR-79-168, U. S. Air Force Rome Air Development center, Griffiss, AFB, NY, June 1979 (DDc-AD-A073132).
 - [BAS03] BASTOS, L. R. D., CASTRO, J. F. B. **Integration between Organizational Requirements and Architecture**. WER03 - VI Workshop em Engenharia de Requisitos, 2003, Piracicaba, Brasil.
 - [BAS04] BASTOS, L. R. D., CASTRO, J. B. F. **Systematic Integration between Requirements and Architecture**. Proceedings of SELMAS 2004 - Software Engineering for Large-Scale Multi-Agent Systems, 2004.
 - [BAU01] BAUER, B., MULLER, J. and ODELL, J. **Agent UML: A formalism for specifying multiagent interaction**. In *Proc. of the 1st Int. Workshop on Agent-Oriented Software Engineering, AOSE'00*, pages 91–104, Limerick, Ireland, 2001.
 - [BEL76] BELL, T. E. and THAYER, T. A. **Software requirements: Are they really a problem?** Proceedings on 2nd International Conference on Software Engineering. San Francisco, 1976.
 - [BEL03] BELLIFEMINE, F., CAIRE, G., POGGI, A., RIMASSA, G., **JADE - A White Paper**, Special issue on JADE of the TILAB Journal "EXP - in search of innovation", September 2003.
 - [BOO99] BOOCH, G., RUMBAUGH, J., and JACOBSON I. Jacobson. **Unified Modeling Language: Users Guide**. Addison Wesley, 1999.
 - [BRE04] BRESCIANI, P., GIORGINI, P., GIUNCHIGLIA, F, MYLOPOULOS, J. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, A. Perini **Tropos: An Agent -Oriented Software Development Methodology**, Autonomous Agents and Multi –Agent Systems 8 (3): 203-236, May 2004.
 - [BRO87] BROOKS, F. **No Silver Bullet: Essence and Accidents of Software Engineering**. IEEE Computer, Vol. 20 No. 4, April 1987, pp. 10-19.
 - [BUS01] BUSETTA, P., RONNQUIST, R., HODGSON, A., LUCAS, A.: **Jack intelligent agents—components for intelligent agents in java**. User Manual 1, Agent Oriented Software, Melbourne, Australia, March 2001.
 - [CAP98] CAPUTO, K. **CMM Implementation Guide: Choreographing Software Process Improvement**. Addison-Wesley, 1998.
 - [CAR01] CARVALHO, A., E. **Uma Estratégia para Implantação de uma Gerência de Requisitos Visando a Melhoria dos Processos de Software**. Dissertação de Mestrado. Universidade Federal de Pernambuco, Centro de Informática. Brasil, 2001.
 - [CAS01] CASTRO, J; ALENCAR, F; CYSNEIROS, G and MYLOPOULOS, J, **Integrating Organizational and Object Oriented Modeling**. In 5th International Symposium on Requirements Engineering – ER'01, IEEE Computer Society Press, 2001
 - [CAS02a] CASTRO, J., KOLP, M., MYLOPOULOS, J.: **Towards Requirements-Driven Information Systems Engineering: The Tropos Project**. Information Systems Journal, Vol. 27:365-389, Elsevier, Amsterdam, The Netherlands, 2002.

- [CAS02b] CASTRO, J., PINTO, R., CASTOR A. and MYLOPOULOS J. **Requirements Traceability in Agent Oriented Development**. SELMAS 2002: 57-72
- [CHU00] CHUNG, L., NIXON, B., YU, E., MYLOPOULOS, J.: **Non-Functional Requirements in Software Engineering** . Kluwer Publishing, 2000.
- [CON00] CONALLEN, J. **Building Web Applications with UML**. Addison-Wesley, 2000.
- [CYS99] CYSNEIROS, L. and LEITE, J. **Integrating Non-Functional Requirements into Data Modeling**. 4th International Symposium on Requirements Engineering. Limerick, Ireland. Jun. 1999.
- [CYS01] CYSNEIROS, G. **Ferramenta para o Suporte do Mapeamento da Modelagem Organizacional em i* para UML**. Dissertação de Mestrado. Centro de Informática, Universidade Federal de Pernambuco, 2001.
- [DAM97] DAMIAN, A.; HONG, D. And LI, H. **Joint Application Development and Participatory Design**. 1997.
- [DAV93] DAVIS, A. M. **Software requirements: objects, functions, and states**. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.
- [DID03] DIDIER, A. **WRE-Process: Um processo de Engenharia de Requisitos Baseado no Rup**. Dissertação de Mestrado. Centro de Informática, Universidade Federal de Pernambuco, Agosto, 2003.
- [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 organizational requirements**. In M. Jirotko and J. Goguen (Eds.), Requirements Engineering, Academic Press Ltd. pp. 87-106. London, 1994.
- [ERI00] ERIKSON, H and PENKER, M. **Business Modeling with UML: Business Patterns at Work**. OMG Press. John Wileys & Sons 2000.
- [ESI96] EUROPEAN SOFTWARE INSTITUTE, **European user survey analysis**. Report USV_EUR 2.1 ESPITI Project, 1996.
- [FIN91] FINKELSTEIN, A. **Tracing Back from Requirements**. In Proc. Of the colloquium on Tools and Techniques for Maintaining Traceability During Design, (Savoy place, UK.), pp. 7/1-7/2, Dec. 1991. IEE Digest No. 1991/180.
- [FIN97] FININ, T., LABROU, Y. and MAYFIELD, J. **KQML as an agent communication language**. In J. Bradshaw, editor, *Software Agents*. MIT Press, 1997.
- [FIP] FIPA-OS (<http://fipa-os.sourceforge.net/>).
- [FIP02] Emorphia, **FIPA-OS Developers Guide**, Nortel Networks' Agent Technology Group, 2002, disponível em: <http://sourceforge.net/projects/fipa-os/>.
- [GOT93] GOTEL, O and FINKELSTEIN, A. **An Analyze of the Requirements Traceability Problem**. Technical Report TR-93-41, Department of Computing, Imperial College of Science, Technology and Medicine, London, U.K., August 1993.
- [GOT94a] GOTEL, O and FINKELSTEIN, A. **Contribution Structures**, Technical Report TR, Department of Computing, Imperial College of Science, Technology and Medicine, London, U.K., September 1994.
- [GOT94b] GOTEL, O and FINKELSTEIN, A. **An Analysis of the Requirements Traceability Problem**. Proceedings of the First International Conference Requirements Engineering (ICRE'94); Colorado Springs, IEEE CS Press, pp. 94 - 101, 1994.
- [GOT95] GOTEL, O. **Contribution Structures for Requirements Engineering**. Ph.D Thesis.

- Department of Computing, Imperial College of Science, Technology, and Medicine, London, U.K., 1996.
- [GOT96] GOTEL, O. and FINKELSTEIN, A. **Extended Requirements Traceability Results of an Industrial Case Study**. In Proceedings of the 2nd International Conference on Requirements Engineering, IEEE Computer Society Press, pp. 169-178. Colorado Springs, USA, 1996.
- [IBM01] IBM. **Patterns for e-business**. At <http://www.ibm.com/developerworks/patterns>, 2001.
- [IEE84] IEEE Std. 830. **IEEE Guide to Software Requirements Specification**. The Institute of Electrical and Electronics Engineers. New York, EUA.1984.
- [IEE91] IEEE Std. 610.2 **IEEE Standard Glossary of Software Engineering Terminology** The Institute of Electrical and Electronics Engineers. EUA, 1991
- [IEE96] IEEE Std. 12333-1996 **IEEE Guide for Developing System Requirements Specification**. The Institute of Electrical and Electronics Engineers. Piscataway, NJ. Guide, EUA, 1996.
- [ISO88] ISO/IEC **9126 Standard for information technology: software products evaluation: quality characteristics and guidelines for theirs use**. Geneva, 1988.
- [ISO97] ISO9000-3. **Quality Management and Quality Assurance Standards – Part 3: Guidelines for the Application of ISO to Development, Supply and Maintenance**. International Institute for Standardization, Geneva, Switzerland, 1997.
- [JAC] The Agent Oriented Software Group, "Jack Development Enviroment", disponível em: <http://www.agent-software.com>.
- [JAC95] JACKSON, M. **Software requirements and specifications: a lexicon of practice, principles and prejudices**. 1 ed. Addison-Wesley. Massachusetts, USA. 1995.
- [JEN00] JENNINGS, N. R.: **On Agent-Based Software Engine ering**. Artificial Intelligence, 117 (2) 277-296, 2000.
- [JEN01] JENNINGS, N. R.: **An agent-based approach for building complex software systems**. Comms. Of the ACM, 44 (4) 35-41, 2001.
- [JEN03a] JENNINGS, N. R. **Applying Agent Technology**. Proceedings of the 5th European Agent Systems Spring School. EASSS 2003. Barcelona, Espanha, Fevereiro 2003.
- [JEN03b] JENNINGS, N. R., BUSSMANN, S.: **Agent-based control systems**. IEEE Control Systems Magazine, 2003.
- [KOL01a] KOLP, M., CASTRO, J., MYLOPOULOS, J.: **A social organization perspective on software architectures**. In Proc. of the 1st Int. Workshop From Software Requirements to Architectures (5–12). STRAW’01, Toronto, Canada (2001).
- [KOL01b] KOLP, M., GIORGINI, P., MYLOPOULOS, J.: **A goal-based organizational perspective on multi-agents architectures**. In Proc. of the 8th Int. Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages. ATAL’01, Seattle, USA (2001).
- [KOL02] KOLP, M., GIORGINI, P., MYLOPOULOS, J.: **Information Systems Development through Social Structures**. 14th International Conference on Software Engineering and Knowledge Engineering (SEKE’02), Ishia, Italy, July (2002).
- [KOT98] KOTONYA, G. and SOMMERVILLE, I. **Requirements engineering – processes and techniques**. John Willy & Sons, 1998.
- [LAB99] LABROU, Y., FININ, T., and PENG, Y. **The current landscape of agent communication languages**. *Intelligent Systems*, 14(2):45–52, 1999.

- [LAM00] LAMSWEERDE, A. **Requirements engineering in the year 00: A research perspective.** 22nd Proceedings of International Conference on Software Engineering. Limerick, Ireland. Jun. 2000.
- [LEI83] LEITE, J.C.S.P. **O Enfoque Social na Análise de Sistemas.** Anais do XVI Congresso Nacional de Informática, São Paulo. 1983, pp. 272-279.
- [LEI87] LEITE, J.C.S.P. **A Survey on Requirements Analysis.** Technical Report. Department of Information and Computer Science. Califórnia, 1987
- [LEI90] LEITE, J.C.S.P. **Elicitação de Requisitos.** Anais do XXIII Congresso Nacional de Informática, Edição em Hipertexto, Rio de Janeiro, 1990.
- [LEI94] LEITE, J.C S.P. **Engenharia de Requisitos.** 1 ed. Rio de Janeiro: PUC-RIO (Notas de Aula), 1994, 63 p.
- [LEI00] LEITE, J.C.S.P. et al. **A Scenario Construction Process.** Requirements Engineering Journal, 2000.
- [LES89] LESCA, H. **Information et adaptation de l'entreprise.** Paris, Masson, 1989.
- [LI00] LI, Y., BENWELL, G., WHIGHAM, P.A., MULGAN, N. **An Agent-oriented software engineering paradigm and the design of a new generation of spatial information system.** The 12th Annual Colloquium of the Spatial Information Research Centre. P.A. Whigham ed. (165-179) Dunedin, New Zealand, 2000.
- [LOU95] LOUCOPOULOS, P. e KARAKOSTAS, V. **System requirements engineering.** London McGraw-Hill, 1995.
- [LUC03] LUCK, M., MCBURNEY, P., PREIST, C.: **Agent Technology: Enabling Next Generation Computing.** AgentLink, 2003.
- [MAM99] MAMANI M. LEITE, J.C.S.P. **Elicit@99 um Protótipo de Ferramenta para a Elicitação de Requisitos.** WER99 - Workshop en Requerimientos / Workshop de Engenharia de Requisitos - II. SADIO. Buenos Aires. 1999. pp. 45-55.
- [MOR04] MORAITIS, P., SPANOUDAKIS, N., **Combining Gaia and JADE for Multi-Agent Systems Development,** in Fourth International Symposium From Agent Theory to Agent Implementation (AT2AI'04), Vienna, Austria, 2004.
- [MUL96] MÜLLER, J.: **The Design of Intelligent Agents, A Layered Approach,** Springer, 1996
- [MUL99] MULLER, J. P. **The Right Agent (Architecture) to Do the Right Thing.** Muller, J.P., Singh, M.P, and Rao, A. S., eds., Intelligent Agents V, LNAI, 1555: 214-246. Springer- Verlag, 1999.
- [MYL95] MYLOPOULOS, J. **Capturing Intentions in Requirements Engineering: A modeling Perspective.** Invited Talk at the IFIP W2.9 meeting, Bramshill, UK, March, 1995.
- [MYL00] MYLOPOULOS, J.; CASTRO, J.and KOLP, M. **Tropos: Toward Agent-Oriented Information Systems Engineering.** Position Paper @ the Second International Bi-Conference Workshop on Agent-Oriented Information Systems AOIS2000. Stockholm, Sweden, June 5-6, 2000.
- [MYL01] MYLOPOULOS, J.; KOLP, M., and CASTRO, J. **UML for Agent Oriented Software Development: the Tropos Proposal** In Proceedings of the Fourth International Conference on the Unified Modeling Language (<<UML>> 2001),

Toronto, Canada, October 2001.

- [NEW93] NEWELL, A.: **Reflections on the Knowledge Level**. Artificial Intelligence, Vol. 59:31-38, 1993.
- [NUS00] NUSEIBEH, B. and EASTERBROOK, S. **Requirements engineering: a roadmap**. Proceedings of the 22nd International Conference on Software Engineering. Limerick, Ireland. Jun. 2000.
- [ODE99] ODELL, J.: **Objects and Agents: How do they differ?** , working paper v2.2, September 1999.
- [ODE00a] ODELL, J.: **Agent Technology**, OMG, green paper produced by the OMG Agent Working Group 2000.
- [ODE00b] ODELL, J., PARUNAK, H. V. D., BAUER, B.: **Extending UML for Agents**. Proc. of the Agent-Oriented Information Systems Workshop (AOIS'00) at the 17th National conference on Artificial Intelligence (AIII'00) (3-17), Gerd Wagner, Yves Lesperance, and Eric Yu eds., Austin, TX (2000).
- [OMG00] Object Management Group. Meta Object Facility Specification (Version 1.3). www.omg.org. Página visitada em 20/07/2004.
- [PAI03] PAIM, F. **Uma Metodologia para Definição de Requisitos em Sistemas Data Warehouse**. Dissertação de Mestrado. Centro de Informática, Universidade Federal de Pernambuco, Fevereiro, 2003.
- [PAL97] PALMER, J. **Traceability. Software Requirements Engineering**. In R. Thayer and M. Dorfman (Eds.), IEEE Computer Society Press, pp. 364-374, 1997.
- [PET98] PAULK M. C. **The Capability Maturity Model - Guidelines for Improve the Software Process**, Addison Wesley, 1995.
- [PIN03] PINHEIRO, F.A. C. **Requirements Traceability**, Chapter of the Book Perspectives On Software Requirements. Kluwer Academic Publishers. 2003.
- [POH96] POHL, K. **Process-Centered Requirements Engineering**. Advanced software Development Series, John Wiley and Sons Ltd., 1996.
- [PRE05] PRESSMAN R. **Software engineering: a practioner's approach**, McGraw-Hill, 1997.
- [RAM98a] RAMESH, B; JARKE, M. **Toward Reference Models For Requirements Traceability**. Technical Report, Georgia State University, 1998.
- [RAM98b] RAMESH, B. **Factors Influencing Requirements Traceability Prattice**, Communication ACM, vol. 41, pp. 37-44, Dec. 1998.
- [RAM01] RAMESH, B and JARKE, M. **Towards Reference Models For Requirements Traceability**. IEEE Transactions on software Eng., vol. 27, pp. 58-93, Jan. 2001.
- [RUS03] RUSSEL, S., Norvig, P. **Artificial Intelligence A Modern Approach**, Prentice Hall, 2003.
- [SAN98] SANTOS, I. and CARVALHO, J. **The Structural, Social, Political and Symbolic Dimensions of Computer-Based Systems Requirements**. In Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality- REFSQ'98, pp. 121-132, June 1998, Pisa, Italy.
- [SHA95] SHAW, M. and GAINES, B. **Requirements acquisition**. Software Engineering Journal, vol 11, 1995.
- [SHA99] SHARP, H. FINKELSTEIN, A. and GALAL, G. **Stakeholder Identification in the Requirements Engineering Process**. Workshop on Requirements Engineering

- Process. Florença, Itália, 1999.
- [SCH75] SCHWARTZ, J. **Construction of Software, Problems and Practicalities, in Practical Strategies for Developing Large Software Systems**, E. Horowitz, ed., Addison-Wesley, Reading, MA, 1975.
 - [SIL03] SILVIA, C. T. L. L. **Detalhando o projeto arquitetural no desenvolvimento de software orientado a agentes: O caso Tropos**. Dissertação de Mestrado. Centro de Informática, Universidade Federal de Pernambuco, Fevereiro, 2003.
 - [SIL04] SILVIA, C., TEDESCO, P., CASTRO, J. and PINTO, R. **Comparing Agent-Oriented Methodologies Using NFR Framework**. Proceedings of SELMAS 2004 - Software Engineering for Large-Scale Multi-Agent Systems.
 - [SOM97] SOMMERVILLE, I. and SAWYER, P. **Requirements engineering – a good practice guide**, John Wiley & Sons Ltd, 1997.
 - [SOM03] SOMMERVILLE, I. **Engenharia de Software**. 6th Edição, Addison Wesley, 2003.
 - [STA94] THE STANDISH GROUP, **The CHAOS Report**. 1994. Disponível em: <http://www.standishgroup.com/sample_research/chaos_1994_1.php>. Acesso em julho. 2004.
 - [STU03] STURM, A., SHEHORY, O. **A Framework for Evaluating Agent-Oriented Methodologies**, Proceedings of the AOIS-2003, 2003.
 - [TAH99] TAHARA, Y., OHSUGA, A., HONIDEN, S.: **Agent System Development Method Based on Agent Patterns**. The 21st International Conference on Software Engineering - ICSE 1999 (356-367), Los Angeles, CA, USA, 1999.
 - [THA97] THAYER R e DORFMAN, M. **Software Requirements Engineering**. IEEE Computer Society Press, 1997.
 - [TOR99] TORANZO, M and CASTRO, J. A. **Comprehensive Traceability Model to Support the Design of Interactive Systems**. in *WISDOM'99 - International Workshop on Interactive System Development and Object Models*, 1999, Lisboa. 1999. Also included in NUNES, N., et al. **Interactive System Design and Object Models**. International Workshop on Interactive System Development and Object Models, 1999, Lisboa. **ECOO'99 - Workshop Reader**. London: Springer Verlag - Lecture Notes in Computer Science, 1999. v.1743. (267- 287).
 - [TOR02a] TORANZO, M. **Uma Proposta para Melhoria do Rastreamento de Requisitos de Software**. Tese de Doutorado. Centro de Informática, Universidade Federal de Pernambuco, Dezembro, 2002.
 - [TOR02b] TORANZO, M, CASTRO, J, MELLO, E. **Uma Proposta para Melhorar o Rastreamento de Requisitos**.
 - [TVE01] TVEIT, A.: **A survey of Agent-Oriented Software Engineering**, First NTNU CSGSC, May 2001.
 - [VIL99] VILLER, S. And SOMMERVILLE, I. **Social analysis in the requirements engineering process: from ethnography to method**. 4th International Symposium on Requirements Engineering, Limerick, Ireland. Jun. 1999.
 - [WIE95] WIERINGA R. J. **An introduction to requirements traceability**. Technical Report IR-389, Faculty of Mathematics and Computer Science, University of Vrije, Amsterdam, September 1995.
 - [WOD00] WOOD, M. and DELOACH, S. A. **“An Overview of the Multiagent System**

- Engineering Methodology**”, in the *First International Workshop on Agent-Oriented Software Engineering (AOSE-2000)*, June, 10, 2000 – Limerick. Ireland.
- [WOO95a] WOOLDRIGE, M., JENNINGS, N. R.: **Intelligent agents: theory and practice** The Knowledge Engineering Review, vol.10 (2):115-152, 1995.
- [WOO95b] WOOLDRIDGE, M.: **Conceptualizing and Developing Agents**. Proceedings of The Agent Software Seminar, UNICOM Seminars, Londres, Abril, 1995.
- [WOO99] WOOLDRIGE, M., JENNINGS, N.R.: **Software Engineering with agents: Pitfalls and Pratfalls**. Internet Computing 3 (3) 20-27, 1999.
- [WOO00] WOOLDRIGE, M., JENNINGS, N. and KINNY D. “**The Gaia Methodology for Agent-Oriented Analysis and Design**,” *Journal of Autonomous Agents and Multi-Agent Systems*, 2000.
- [WOO01a] WOOLDRIGE, M., CIANCARINI, P. **Agent-Oriented Software Engineering: The State of the Art**. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*. Springer- Verlag Lecture Notes in AI Volume 1957, January 2001.
- [WOO01b] WOOLDRIDGE, M., WEISS, G., CIANCARINI, P. editors: **Agent-Oriented Software Engineering II**. Springer-Verlag Lecture Notes in Computer Science Volume 2222, February 2001.
- [WOO02] WOOLDRIGE, M.: **Introduction to Multiagent Systems**. John Wiley and Sons, New York 2002.
- [YOU02] YOUNG, R. Disponível em: < <http://www.ralphyoung.net/verification.html>>. Acesso em Julho. 2004
- [YU94] YU, E. and MYLOPOULOS, J. **Understanding ‘Why’ in Software Process Modeling, Analysis and Design**. Sixteenth International Conference on Software Engineering. Sorrento, 1994.
- [YU95a] YU, E et al. **From Organization Models to System Requirements – A cooperating Agents Approach**. In Proc. of the Third International Conference on Cooperative Information Systems – COOPIS’95, pp. 2-9, Vienna, Austria. May 1995.
- [YU95b] YU, E. **Modelling Strategic Relationships for Process Reengineering**. PhD thesis, University of Toronto, Department of Computer Science, 1995.
- [YU97a] YU, E. **Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering**. Proceedings of IEEE International Symposium on Requirements Engineering - RE97, pp.226-235, Jan. 1997.
- [YU97b] YU, E. **Why Agent-Oriented Requirements Engineering**. In Proceedings of 3rd International Workshop on Requirements Engineering: Foundations for Software Quality – REFSQ’97, pp. 5-12, June 1997, Barcelona, Spain.
- [YU97c] YU, E and MYLOPOULOS, J. **Modeling Organizational Issues for Enterprise Integration**. In Proc. of the International Conference on Enterprise Integration and Modelling Technology – ICEIMT’97, pp. 12-21, October 1997, Turin, Italy.
- [YU02] YU, E. **Agent-oriented modelling: software versus world**. In M. J. Wooldridge, G. Weiss, and P. Ciancarini, editors, *Agent-oriented software engineering*. Proceedings of the Second International Workshop (AOSE-2301), Lecture Notes in Artificial

Intelligence, Vol. 2424. Springer-Verlag, 2002.