Pós-Graduação em Ciência da Computação

" Uma Proposta para Melhorar o Rastreamento de Requisitos de Software "

Por

Marco Antonio Toranzo Cespedes

Tese de Doutorado

Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

www.cin.ufpe.br/~posgraduacao

RECIFE, Dezembro/2002

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARCO ANTONIO TORANZO CESPEDES

"Uma Proposta para Melhorar o Rastreamento de Requisitos de Software"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: JAELSON FREIRE BRELAZ de CASTRO

RECIFE, Dezembro/2002

AGRADECIMENTOS

A Deus, por conhecer as pessoas certas, no lugar certo e na hora certa. Obrigado, por compartilhar e desfrutar com minhas amizades e seres queridos esta nova conquista.

À minha esposa Janaína pelo seu apoio, paciência e carinho incondicional, sobretudo nos momentos mais críticos do desenvolvimento da tese.

Aos meus filhos Evelyn, Marquinho e Frederico, pelos momentos de alegria e inspiração fornecidos no decorrer do trabalho. Sem dúvida foram e serão sempre muito importantes.

À minha mãe Fidelia e irmãos Juan, Mário, Yenny e Ivonne pelo grande apoio.

A meu orientador pela amizade, solidaridade, confiança, motivação, apoio e orientação no desenvolvimento deste trabalho. Muito obrigado mesmo.

Aos meus professores Alexandre Vasconcelos, Augusto Sampaio, Paulo Borba e Roberto Souto Maior pelo ensino fornecido.

Aos membros da Banca Examinadora da Proposta de Doutorado, composta por Itana Gimenes, Alexandre Vasconcelos e Hermano Perrelli, que forneceram comentários e sugestões para melhorar a tese.

Ao professor Francisco Pinheiro pelo ensino do rastreamento de requisitos e pelos comentários realizados durante minha estadia no Departamento de Ciência da Computação da Universidade de Brasília. Obrigado ao Professor Júlio César Leite, Coordenador Nacional do Projeto WEST, pelo financiamento da estadia.

Aos colegas Fernanda Alencar, Victor Araya Santander, Márcio Bueno, Gilberto Cysneiro Filho, Juan Zambrano, e especialmente para Rosa Pinto e Andréia Castor pela leitura, comentários e correções do português da tese.

Aos alunos das turmas de 2001 e 2002 do Curso de informática da Universidade Estadual do Oeste do Paraná, , especialmente a Elton Mello pela contribuição no estudo de caso da Biblioteca.

À empresa MGInformática, da cidade de Cascavel, pela colaboração no estudo de caso do sistema de contabilidade.

RESUMO

Mudança é inevitável no desenvolvimento de sistemas. Mudanças podem ser pequenas ou grandes, simples ou complexas, importantes ou triviais. Experimentos, das últimas três décadas, mostram que a realização de mudanças de software, sem a análise nos efeitos delas, pode contribuir para uma estimativa equivocada do esforço, atrasos no cronograma, projeto de software degradado, produtos de software não-confiáveis e a desativação prematura do sistema. O problema do *bug* do ano 2000 (Y2K) é um bom exemplo da percepção errônea dos impactos de mudança sobre muitos software.

Entre as principais preocupações no desenvolvimento de software, detectamos a necessidade de se entender e rastrear requisitos adequadamente. Esse trabalho é realizado em conjunto pelos engenheiros de requisitos, desenvolvedores, e clientes e/ou usuários que requerem o software. A engenharia de requisitos desenvolveu algumas técnicas, modelos de rastreamento de requisitos, e ferramentas para ajudar os engenheiros de requisitos.

Nesta tese abordaremos em detalhe a questão do rastreamento de requisitos. Em particular, fornecemos uma proposta para melhorar o rastreamento de requisitos que está composta de uma classificação de informações que serão rastreadas, um meta-modelo, um modelo intermediário de rastreamento, e um processo para desenvolver um modelo conceitual de rastreamento.

A validação da tese foi realizada sobre cinco projetos. Neste tese apresentaremos os estudos de casos realizados sobre um sistema de vídeo locadora e o sistema de biblioteca da Universidade Estadual do Oeste do Paraná.

Palavras-chave: Requisitos, Rastreamento de Requisitos, Relacionamentos, Engenharia de Requisitos.

ABSTRACT

Change is inevitable in the development of systems. Changes can be small or large, simple or complex, important or trivial. Experiments over the last three decades show that performing software changes without the analysis of their effects can lead to poor effort estimates, delays in release schedules, degraded software design, unreliable software products, and premature retirement of the software system. The year 2000 (Y2K) problem is a good example of poor insight into the impacts of change on many software.

Among the main concerns in the software development, we have detected the need to understand and to trace requirements appropriately. This work, is accomplished together by requirements engineers, developers, and users and/or customers that request the software. Requirements Engineering has developed some techniques, requirements traceability models, and tools to assist requirements engineers.

In this thesis we focus on requirements traceability issue. In particular, we provide a proposal to improve requirements traceability which consist of the classification of the information to be traced, a meta-model, a traceability intermediate model, and a process to develop the traceability conceptual model.

The validation of the thesis was accomplished on five projects. In this thesis we will present the studies of cases accomplished on a system of video rental and the system of library of the Universidade Estadual of Paraná.

Key-words: Requirement, Requirements Traceability, Relationships, Requirements Engineering.

Índice de Conteúdo

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ACP	Área-chave de Processo
CMM	*Capability Maturity Model*
DoD	Department of Defense
ISO	International Standard Organization
SEI	Software Engineering Institute
UML	Unified Modeling Language
RR	Rastreamento de Requisitos

Capítulo 1

Introdução

Este capítulo apresenta alguns conceitos básicos do rastreamento de requisitos, o contexto, a motivação e as contribuições da tese de doutorado que também foram obtidas através do desenvolvimento e rastreamento de cinco projetos de software.

1.1 Introdução

O interesse e a atenção crescente da comunidade científica pelo desenvolvimento de software teve suas raízes na chamada Crise do Software [Naur, 1969]. A crise provocou várias mudanças na forma de como as pessoas desenvolviam software. Uma dessas mudanças foi um maior interesse dos pesquisadores acerca de como elicitar, coletar, analisar e especificar formalmente os requisitos de um sistema [Hsia, 1993]. Mais de três décadas depois, ainda continuamos com grandes dificuldades para produzir um documento de requisitos e mantê-lo consistente com outros artefatos produzidos no desenvolvimento de software.

A engenharia de requisitos é uma nova área, nasceu como uma disciplina para tratar dos problemas relacionados com requisitos, incluindo a questão do rastreamento de requisitos cujo objetivo é gerenciar os relacionamentos estabelecidos entre os requisitos e os artefatos de um projeto de software para manter consistentes as informações relacionadas. Esta tese aborda o tema do rastreamento de requisitos e apresenta uma proposta que introduz várias contribuições que visam melhorar a atividade do rastreamento de requisitos.

1.2 Conceitos Básicos

Um dos primeiros passos para se obter uma melhor compreensão do rastreamento de requisitos é começar o seu estudo conhecendo alguns dos seus conceitos básicos. Para isso, identificamos e extraímos da literatura algumas definições, que incluem: *Requisito, Documento de Requisitos, Rastreamento de Requisitos, Pré-rastreamento, Pós-rastreamento, Rastreamento Bidirecional, Rastreamento Vertical, Rastreamento Horizontal, Análise de Impacto, Modelo de Rastreamento e Modelo Intermediário de Rastreamento*. A seguir iniciaremos a apresentação desses conceitos.

O padrão IEEE (Institute of Electrical and Electronics Engineers [IEEE-610.2, 1991]) fornece as seguintes definições de *Requisito*:

a) uma condição ou propriedade necessitada por um usuário para resolver um problema ou alcançar um objetivo;

b) uma condição ou propriedade que deve estar contida em um sistema ou um componente do sistema para satisfazer um contrato, padrão, especificação ou outro documento requerido formalmente;

c) uma representação documentada de uma condição ou capacidade como nos item a) ou b).

As definições de Requisito da IEEE são mais interessantes porque requisito é definido a partir de diferentes perspectivas: do sistema final ou do componente de um sistema; do ponto de vista do usuário; e do ponto de vista da certificação, tais como, processo, representação, documentação, e/ou um padrão requerido no processo de desenvolvimento de um sistema. É importante salientar que existem muitas definições e classificações do termo Requisito. Particularmente, esta tese trabalhará sobre os requisitos funcionais, uma vez que o rastreamento dos requisitos não-funcionais está fora do escopo do nosso trabalho. Os requisitos funcionais especificam ações que um sistema deverá realizar, sem considerar restrições físicas.

O segundo termo a definir é o *Documento de Requisitos*. Um documento de requisitos é uma formalização das funcionalidades que um sistema deverá satisfazer [Dorfman, 1997]. Na literatura existem várias propostas para nomear e estruturar um documento de requisitos [Wiegers, 1999; Dorfman, 1997; Pressman, 2001]. A partir do documento de requisitos, outros artefatos intermediários (documentos, manuais, e programas) serão produzidos, devendo estar consistentes no final do desenvolvimento do sistema. A linha de pesquisa da engenharia de requisitos responsável pela consistência das informações relacionadas é o rastreamento de requisitos.

O termo *Rastreamento de Requisitos* possui várias definições. A definição da [IEEE-830, 1984] expressa que uma especificação de requisitos de software é rastreável se:

a) A origem de cada um dos seus requisitos é clara, e;
b) Se especificação de requisitos facilita a referência de cada um dos requisitos no desenvolvimento futuro.

O item b), da definição, estabelece a necessidade de usar relacionamentos entre requisitos e outros artefatos do processo de desenvolvimento. Porém, a definição não captura a parte dinâmica dos requisitos no processo de desenvolvimento [Pinheiro, 1996a].

O *Rastreamento de Requisitos* tradicional foi dividido em *Pré-rastreamento e Pós-rastreamento* [Gotel, 1996b]. O *Pré-rastreamento de Requisitos* refere-se à habilidade para descrever e seguir aqueles aspectos da vida do requisito antes da sua inclusão no documento de requisitos [Gotel, 1996b]. O *Pós-rastreamento de Requisitos* refere-se à habilidade para descrever e seguir aqueles aspectos da vida do requisito que resulta após inclusão no documento de requisitos [Gotel, 1996b].

O termo *Rastreamento Bidirecional* expressa que o rastreamento deve ter dois sentidos: para frente e para trás [Ramesh, 1993]. Por exemplo, o rastreamento para frente é fornecido se um requisito referencia o diagrama onde foi modelado. Seguindo o

mesmo exemplo, o rastreamento para trás é possível se o diagrama pode referenciar os requisitos modelados nele.

O rastreamento de requisitos pode estabelecer relacionamentos entre elementos de diferentes ou da mesma fase de desenvolvimento. Isso introduz as noções de *Rastreamento Vertical e Rastreamento Horizontal* [Ramesh, 1993]. O *Rastreamento Vertical* é um relacionamento entre elementos de diferentes fases do processo de desenvolvimento. Por exemplo, um relacionamento entre um requisito e um programa. O *Rastreamento Horizontal* é um relacionamento entre elementos de uma mesma fase do processo de desenvolvimento. Por exemplo, um relacionamento entre dois requisitos.

Mudanças nos requisitos são inevitáveis no processo de desenvolvimento. A experiência das três últimas décadas indica que fazer mudanças sem visibilidade nos efeitos colaterais pode contribuir para uma estimativa do esforço pobre, degradação do projeto de software e uma desativação prematura do sistema [Lehman, 1994]. Para evitar alguns desses problemas é recomendável fazer uma análise de impacto. O termo *Análise de Impacto*, que é baseado nos relacionamentos existentes entre os elementos do software, identifica os elementos afetados pela introdução da proposta de uma mudança. As abordagens sistemáticas para a análise de impacto não fazem parte do treinamento formal da engenharia de software [Shaw, 1996]. Informações adicionais sobre a análise de impacto podem ser encontradas em [Arnold, 1993], [Turver, 1994] e [Madhavji, 1991].

Neste trabalho definimos o termo *Modelo de Rastreamento* como um diagrama de classes conceitual que representa coleções de informações rastreadas no processo de desenvolvimento. Por exemplo, a classe *R_Requisito* pode representar todos os requisitos do sistema, a classe *R_Diagrama* pode representar todos os tipos de diagramas que modelam os requisitos do sistema. Um dos objetivos deste trabalho é contribuir com a elaboração de um modelo de rastreamento de um software.

Um outro termo definido neste trabalho é o *Modelo Intermediário* de rastreamento como um modelo de rastreamento que não é um modelo básico nem um modelo completo porque não possui todas as possíveis classes e relacionamentos existentes nos diferentes modelos de rastreamento de software. Desenvolver um modelo intermediário completo é impossível porque não possível prever todos as classes e relacionamentos de todos os projetos [Gotel, 1996b].

Um dos objetivos deste trabalho é contribuir com a elaboração de um modelo de rastreamento de software, a partir do zero, mas usando como apoio uma classificação de informações a serem rastreadas, um modelo intermediário e processo composto de diretrizes para guiar a elaboração de um modelo de rastreamento.

Com a definição do termo modelo intermediário de rastreamento concluímos a apresentação dos conceitos básicos. A seguir apresentaremos o contexto da nossa pesquisa.

1.3 Contexto

O desenvolvimento de software cada vez mais complexo, passível de certificação e com menor custo possível, tem se tornado um desafio constante para a comunidade de engenharia de software. Observamos que o passo primário e fundamental para o sucesso de qualquer processo de desenvolvimento de software é a definição e análise dos requisitos. Mesmo que se tenha um sistema bem projetado e codificado, se ele foi mal especificado, o sistema não satisfará as reais necessidades do negócio do cliente.

A partir do documento de requisitos, outros artefatos intermediários serão produzidos. Ao final do desenvolvimento do sistema, esses artefatos deverão estar de acordo com as funcionalidades oferecidas no produto final. Assim, é importante que no processo de desenvolvimento sejam gerenciados os relacionamentos (uni ou bidirecionais) entre os requisitos e os artefatos para facilitar a manutenção e a validação de um sistema. Em resumo, é necessário aplicar rastreamento de requisito no processo de desenvolvimento para manter um sistema de qualidade.

O rastreamento de requisitos é reconhecido como um importante pré-requisito para o desenvolvimento de sistemas de qualidade [Pohl, 1994a; Pohl, 1996a; Pohl, 1996b] porque ajuda a manter consistentes um conjunto de informações relacionadas do processo de desenvolvimento. Porém, como afirmado em [Palmer, 1997], o rastreamento é geralmente mal entendido, freqüentemente mal aplicado, e raramente realizado corretamente por uma série de fatores, tais como: existência de diferentes definições de rastreamento de requisitos, diferentes experência de problemas resolvidos usando rastreamento. O fato de uma organização possuir uma ferramenta para a gerência e/ou rastreamento de requisitos não garante que a mesma possua um processo para desenvolver um modelo de rastreamento e/ou um processo para rastrear requisitos.

Esta tese trata o rastreamento de requisitos, especificamente, apresenta e aplica uma proposta para melhorar a atividade de rastreamento de requisitos. A proposta é resultado da realização de uma combinação de contribuições teóricas e práticas.

As contribuições teóricas consistem em: a elaboração de um meta-modelo; elaboração um modelo intermediário para o rastreamento de requisitos; proposta de um conjunto de tipos de relacionamentos para rastrear; uma classificacação das informações que compõem um modelo de rastreamento; e um processo (conjunto de diretrizes) para elaborar um modelo de rastreamento.

As contribuições práticas consistem na aplicação e validação das contribuições teóricas para elaborar um modelo de rastreamento para cada um dos cinco software (controle de locadora, biblioteca, contabilidade, custo e produção de leite, e condomínio) desenvolvidos e usados como estudos de caso. Nos estudos de casos participaram 20 pessoas que foram distribuídas em grupos de quatro de pessoas para desenvolver os diferentes projetos. Neste trabalho apresentaremos somente os resultados do rastreamento dos sistemas de controle de locadora e biblioteca.

As motivações e justificativas da proposta do trabalho são identificadas e explicadas na seção seguinte.

1.4 Motivação

Muitas contribuições importantes e relacionadas com qualidade de software, como o DoD-2167A (Department of Defense [DoD, 1984]), ISO 9000-3 [ISO-9003] e o CMM (Capability Maturity Model for Software [Paulk, 1993]), recomendam que o rastreamento de requisitos seja praticado. Porém, elas não fornecem um modelo compreensivo e explícito que inclua as informações que deveriam ser capturadas e formar parte de um modelo de rastreamento [Ramesh, 1998a; Toranzo, 1998a; Toranzo 1998b].

Na última década foram propostos vários trabalhos visando solucionar muitos dos problemas que afetam o pré e pós-rastreamento de requisitos. Por exemplo, Gotel [Gotel, 1996a] centraliza sua atenção no processo de produção e refinamento de requisitos; em [Pohl, 1996a] é proposto um ambiente centrado em processo; em [Pinheiro, 1996a] é proposto uma abordagem formal (matemática) para o rastreamento; e em [Ramesh, 2001] são propostos modelos de referência para o rastreamento de requisitos. No Capítulo 2 explicaremos os trabalhos realizados por esses pesquisadores.

Após revisar os trabalhos desses e de outros pesquisadores, podemos dizer que as motivações da tese estão fundamentadas nos seguintes aspectos:

1. Os trabalhos partem da suposição que os itens a serem rastreados já foram definidos. Diferentes trabalhos argumentam que os requisitos de software são a matéria-prima para o desenvolvimento de software. Independentemente da tecnologia usada para implementar um sistema, se os requisitos estão errados ou incompletos, o uso da tecnologia não resolverá os erros sobre os mesmos. De forma análoga, argumentamos que a matéria-prima para o rastreamento são os elementos que serão rastreados, e que uma ferramenta usada para rastrear requisitos não resolverá o problema de omissão dos elementos. Podemos dizer que existe uma distância entre O que *devemos ou podemos* rastrear? e *Como rastrear*? Esta tese está preocupada com a questão *O que devemos ou podemos rastrear*? Sem dúvida, a questão *Como*

rastrear? é muito bem respondida pelos trabalhos de [Pinheiro, 1996a], [Pohl, 1996a] e [Ramesh, 2001]. Logo, fazendo alguns ajustes, tanto em nosso trabalho, como na abordagem preocupada como a questão como rastrear ?, é possível elaborar uma proposta mais rica e completa para o rastreamento de requisitos;

2. Muitos trabalhos formulam perguntas isoladas para identificar alguns dos elementos a serem rastreados. Alguns trabalhos ([Gotel, 1996b], [Pinheiro, 1996a], [Pohl, 1996a] e [Ramesh, 2001]) não fornecem um processo sistemático para identificar os elementos a serem rastreados. Nosso trabalho visa a construção de um modelo conceitual de rastreamento através da proposta de um simples processo composto de um conjunto de diretrizes. Dessa forma, e indiretamente, esperamos contribuir com a melhoria dos trabalhos preocupados com a questão *Como rastrear*?
3. Em geral, os trabalhos não usam tipos de relacionamento bem definidos e exemplificados. Dos vários trabalhos revisados ([Gotel, 1996b], [Pinheiro, 1996a], [Pohl, 1996a] e [Ramesh, 2001]), alguns trabalhos (por exemplo, [Morris, 1994] e [Ramesh,2001]) mostram uma preocupação com a modelagem dos elementos rastreados. Porém, somente o trabalho de [Ramesh, 2001] apresenta uma discussão, mas não exemplifica, o uso dos tipos de relacionamento para melhorar o significado dos mesmos. Nosso trabalho propõe e define um conjunto de tipos de relacionamento com o objetivo de melhorar o significado dos modelos de rastreamento. Em geral, os trabalhos delegam ao usuário a responsabilidade de atribuir o significado aos relacionamentos. O diagrama de classe da UML (*Unified Modeling Language* [Booch, 1999]) é um exemplo de como os tipos de relacionamento (agregação, composição e generalização) podem ajudar a melhorar o significado dos diagramas. Nesse exemplo, uma pessoa que conheça a definição desses tipos de relacionamento não precisará conhecer o nome atribuído pelo usuário aos mesmos porque é irrelevante neste caso, mas em outros casos é uma informação complementar para realçar o significado do relacionamento;
4. Existem diferentes meta-modelos. Um meta-modelo de rastreamento é um linguagem que identifica as meta-classes (classe e relacionamentos) que podem ser instanciadas e relacionadas para elaborar um modelo de rastreamento. Muitos trabalhos fornecem pouca ou nenhuma atenção à construção de meta-modelo. Existem trabalhos que usam notações não padronizadas para construir seus meta-modelos (por exemplo, [Ramesh, 2001] e [Pohl, 1996a]) que dificultam o entendimento do mesmo na hora de estabelecer relacionamentos entre os elementos que fazem parte de um modelo de rastreamento. Contrário aos meta-modelos propostos na literatura, o nosso meta-modelo foi elaborado usando uma notação padronizada (MOF (*Meta Object Facility* [OMG, 2000])) e identifica e define cada um dos relacionamentos que podem ser usados na elaboração de um modelo de rastreamento;

5. Existem poucos trabalhos que apresentam modelos de rastreamento genéricos. Em geral, os trabalhos não fornecem modelos de rastreamento pre-definidos para facilitar a identificação dos elementos que podem fazer parte do modelo de rastreamento de um software. Existem vários exemplos industriais e acadêmicos que ilustram a aceitação de modelos para guiar ou orientar algumas atividades do desenvolvimento de software. Por exemplo, a área de arquitetura de software tem proposto vários modelos arquiteturais genéricos que refletem a arquitetura dos sistemas existentes. Um outro exemplo é a proposta do modelo para ambientes CASE [ECMA, 1991]. Esta tese apresenta a proposta de um modelo intermediário de rastreamento. Recentemente, em [Ramesh, 2001], foram propostos modelos de referência (modelos de rastreamento genérico) para o rastreamento de requisitos que visam ajudar aos usuários no desenvolvimento dos seus modelos de rastreamento. Isso confirmou e motivou, ainda mais, o desenvolvimento do nosso modelo intermediário de rastreamento;
6. Em geral, os trabalhos não possuem um processo bem definido para desenvolver um modelo de rastreamento. Nesta tese, revisamos vários trabalhos que propõem soluções para o rastreamento de requisitos, porém, não fornecem um processo bem definido para aplicar ou instanciar sua própria proposta de trabalho. Por exemplo, os trabalhos de Ramesh [Ramesh, 1995a; Ramesh, 1997; Ramesh, 2001] não apresentam um processo para desenvolver um modelo de rastreamento ou criar um modelo de rastreamento usando seus modelos propostos. É importante salientar que na literatura relacionada com o rastreamento de requisitos, incluindo o livro da Gerência de Requisitos [Leffingwell, 2000], não foi encontrado um processo bem definido para a construção de um modelo de rastreamento. Portanto, nossa tese apresenta um processo na Seção 4.4 para desenvolver um modelo de rastreamento;
7. Em geral, a notação usada por muitos trabalhos para a representação matricial dos relacionamentos é pobre e ambígua. Considere a matriz da Figura 1, que expressa como alguns objetivos de um sistema (primeira linha) são satisfeitos por alguns requisitos (primeira coluna), porém, surgem as seguintes perguntas:
   a) Qual é o grau de dependência dos requisitos com os objetivos?
   b) Todos os requisitos associados com o objetivo organizacional OBS-2 (segunda coluna) são obrigatórios ou opcionais ?
   c) Qual é significado do relacionamento representado na matriz da Figura 1?

Um dos objetivos desta tese é recomendar uma notação alternativa para expressar a representação matricial dos tipos de relacionamento apresentados na Seção 3.3.

	[OBS-2] Implementar um cartão ponto no sistema	[OBO-8] Ter um cadastro de todos os funcionários da loja
[REQ-50] O sistema deverá permitir ao funcionário registrar a hora de entrada...	X	
[REQ-51] O sistema deverá gerar um relatório das horas extras trabalhadas...	X	
[REQ-6] O sistema deverá permitir ao supervisor a inclusão de funcionários ...		X
[REQ-7] O sistema deverá permitir ao supervisor a exclusão de funcionários ...		X
[REQ-8] O sistema deverá permitir ao supervisor consultar as ...		X

Figura 1: Exemplo de notação ambígua

A seguir apresentaremos os objetivos e a abordagem da tese.

1.5 Objetivos e Abordagem

Considerando os itens da nossa motivação e da necessidade de aperfeiçoar o rastreamento de requisitos, o objetivo geral deste trabalho foi elaborar uma proposta para melhorar o rastreamento de requisitos. Os seus sub-objetivos são:

1. Propor uma estrutura para classificar as informações rastreadas. Independente do pré e pós-rastreamento, desejamos propor uma classificação das informações rastreadas fornecendo ao usuário uma fonte de informação para elicitar, analisar e validar as informações que serão rastreadas. Para isso analisamos algumas técnicas de análise de sistemas orientados a objetos ([Rumbaugh, 1996], [Bailin, 1997], [Booch, 1994] e [Harmon, 1997]), técnicas de análise estruturada de sistemas ([Gane, 1983] e [Yourdon, 1994]), trabalhos da gerência e rastreamento de requisitos ([Leffingwell, 2000] e [Palmer, 1997]). Além disso, foram aplicados a experiência e conhecimento obtidos como monitor dos cursos de extensão e especialização do Centro de Informática da UFPE, como também, a orientação fornecida a programadores afim de desenvolver sistemas para as empresas comerciais da cidade de Cascavel, Paraná;
2. Definir tipos de relacionamentos. Estudamos vários modelos de rastreamento propostos na literatura para chegar a um consenso sobre um conjunto de tipos de relacionamentos para o rastreamento. Para isso, desenvolvemos cinco estudos de caso de rastreamento: Sistema de Controle de Locadora, Sistema de Biblioteca, Sistema de Contabilidade, Sistema de Custo e Produção de Leite e Sistema de

Condomínio. Somente os estudos de caso sobre a locadora e biblioteca serão apresentados nesta tese. Entre os trabalhos revisados, optamos por incluir e redefinir alguns relacionamentos de dependência (satisfação e recurso) da proposta i * [Yu, 1995] e de Pohl [Pohl, 1996a]. Os relacionamentos são: generalização, agregação, recurso, satisfação, responsabilidade, representação e aloca. Esses relacionamentos serão apresentados e discutidos na Seção 3.3. Apesar dos cinco estudos de caso realizados e dos modelos de rastreamento elaborados, usando os tipos de relacionamentos, recomendamos ainda a realização de outros estudos de casos visando a melhoria e aplicação dos tipos de relacionamento. É necessário ter muitos estudos de caso de outras áreas para afirmar que os relacionamentos propostos nesta tese são uma solução definitiva e fechada para o rastreamento de requisitos;

3. Definir um meta-modelo. Para construir um modelo de rastreamento é necessário organizar os tipos de relacionamento em um meta-modelo para conhecer como as classes podem se relacionar. Decidimos não estender a UML porque a mesma foi originalmente projetada para modelar sistemas e porque teríamos que colocar uma série de restrições sobre as notações do diagrama de classe da UML para evitar que sejam usadas na construção de um modelo de rastreamento. Portanto, a partir do *MOF* (*Meta Object Facility* [OMG, 2000]) desenvolvemos nosso meta-modelo. O meta-modelo será apresentado e discutido na Seção 3.3;
4. Propor um modelo intermediário de rastreamento. Um modelo intermediário de requisitos pode agregar mais valor ao processo para desenvolver um modelo de rastreamento. A partir de uma combinação de fatores, tais como, os estudos de caso; os resultados desses estudos; os modelos de Pohl [Pohl, 1996a]; os trabalhos de Ramesh ([Ramesh, 1993; Ramesh, 1995a; Ramesh, 2001]); os trabalhos de [Morris, 1994] e Leffingwell [Leffingwell, 2000], para citar alguns, desenvolvemos e refinamos um modelo intermediário de rastreamento. É importante salientar que o modelo intermediário é independente do domínio de aplicação e do processo de desenvolvimento de software porque ele contem os elementos (requisitos, diagrama, programa, teste, objetivos do sistema e objetivos organizacionais) geralmente encontrados nos diferentes paradigmas. Na Seção 4.2 apresentaremos e discutiremos o modelo intermediário;
5. Propor um processo composto de diretrizes para desenvolver um modelo de rastreamento. Depois de apresentar os diferentes elementos que podem contribuir para melhorar o rastreamento de requisitos, existe a necessidade de se ter um processo para integrá-los para desenvolver um modelo de rastreamento. Para isso, revisamos vários trabalhos ([Bailin, 1997], [Harmon, 1997], [Gane, 1983], [Leffingwell, 2000] e [Palmer, 1997]) visando extrair o melhor de cada um deles (passos e recomendações). O resultado final foi um conjunto de diretrizes para guiar a elaboração e refinamento de um modelo de rastreamento. É importante salientar

que o processo e as diretrizes propostas são independentes do domínio de aplicação e do processo de software usado para desenvolver o software que será rastreado porque o processo de rastreamento identifica e manipula os elementos/artefatos (requisitos, diagrama, programa, teste, objetivos do sistema e objetivos organizacionais) geralmente encontrados nos diferentes processos de software. Na Seção 4.6 apresentaremos e ilustraremos as diretrizes do processo para desenvolver um modelo de rastreamento;

6. Propor uma representação matricial. Não basta a construção de um modelo de rastreamento, é necessário uma notação que permita refletir o significado de um relacionamento. Em geral, as ferramentas de gerência de requisitos ([RequisitePro, 2001], [QSS, 2000], [Marconi, 1996]) produzem matrizes como a apresentada na Figura 1. Nossos estudos de casos indicaram que essas matrizes tradicionais têm sentido quando estão acompanhadas de algum texto que explique o seu conteúdo, mas sem o texto, é difícil identificar seu verdadeiro significado. Para evitar ou minimizar essa ambigüidade, estudamos, testamos, comparamos e analisamos diferentes alternativas para representar o significado dos relacionamentos. A notação apresentada neste trabalho é resultado de cinco estudos de casos.
7. Validar nossa proposta sobre cinco estudos de casos. As contribuições práticas da tese são os rastreamento de requisitos de cinco estudos de casos (controle de locadora, biblioteca, contabilidade, custo e produção de leite, e condomínio). Nesses estudos de casos participaram 20 pessoas que foram distribuídas em grupos para desenvolver os diferentes projetos. Esta tese somente apresenta os estudos de casos da vídeo locadora (Capítulo 4) e da biblioteca (Capítulo 5). Para validar os modelos de rastreamento nos estudos de casos, foram formulados alguns cenários que incluiram a questão da análise de impacto. Além disso, fornecemos detalhes de como as informações foram organizadas nas matrizes para facilitar o cruzamento das informações. A análise de impacto de cada um dos projetos foram realizadas pelos dos próprios desenvolvedores e por outros desenvolvedores que não participaram na construção do sistema, mas que tinham conhecimentos teóricos da análise de impacto.

A seguir apresentaremos as contribuições da tese.

1.6 Contribuições da Tese

As contribuições da tese para a área de rastreamento de requisitos de software são:

1. Estruturar as informações a serem rastreadas foram estruturadas em níveis de informações: externo, organizacional, gerencial e de desenvolvimento. A

informação do nível externo representa os conceitos relacionados com o contexto político, econômico e o uso de normas externas às organizações. Por exemplo, a lei da CPMF (Contribuição Provisória sobre Movimentação Financeira) é um elemento externo aos bancos brasileiros. O nível de informação organizacional representa muito dos conceitos e informações que originaram o desenvolvimento e crescimento de uma organização e que afetam os seus sistemas de informação. O objetivo da introdução do nível organizacional é enfatizar que os requisitos dos sistemas devem satisfazer a proposta e modificação das regras e objetivos organizacionais. No nível de informação gerencial, este trabalho está interessado na atividade do planejamento da gerência do projeto, mais especificamente, no relacionamento entre as tarefas e os requisitos para um melhor acompanhamento e controle dos requisitos do sistema. O nível de informação de desenvolvimento representa os elementos/artefatos produzidos nas diferentes atividades do desenvolvimento de software. Alguns dos artefatos encontrados neste nível são: documentos de requisitos, diagramas e programas. Um dos objetivos do nosso trabalho é explicitar os tipos de informações que devem ser consideradas na construção de um modelo de rastreameto;

2. Definir um conjunto de tipos de relacionamento para melhorar a semântica de um modelo de rastreamento. A proposta dos tipos de relacionamento visa facilitar o entendimento do significado dos relacionamentos entre as classes do modelo de rastreamento. Em lugar de delegar a responsabilidade ao usuário para definir o significado dos relacionamentos em um modelo de rastreamento, através da nomeação dos mesmos, esta tese define o significado e atribui uma notação para cada um dos tipos de relacionamento do meta-modelo;
3. Desenvolver um meta-modelo que visa unificar os meta-modelos existentes. O meta-modelo deve ser considerado um ponto de partida para a unificação dos diferentes meta-modelos expressos nas diferentes notações não padronizadas. Nosso meta-modelo identifica, define e restringe os relacionamentos que podem ser usados na construção de um modelo de rastreamento. Algumas pesquisas existentes optaram por estender o diagrama de classe da UML com os estereótipos para introduzir novos relacionamentos, isso é bom, porém, deveremos restringir os tipos de relacionamento (ou associação) que podem ser usados em um diagrama de classe que representará um modelo de rastreamento, isto é, pode que não seja possível usar o relacionamento de *composição* porque seu significado não tem sentido no contexto do rastreamento. Além disso, é importante salientar que a UML foi projeta para modelar sistema e não para rastrear requisitos;
4. Propor um modelo intermediário de rastreamento de requisitos. Esta tese estende o modelo apresentado em [Ramesh, 2001] através da inclusão de: níveis de informação de rastreamento; novos conceitos (por exemplo tarefa), e tipos de relacionamento. A aplicação do modelo intermediário nos estudos de casos facilitou

a construção dos modelos de rastreamento dos projetos. O maior benefício do modelo intermediário é evitar que um usuário comece a construir seu modelo de rastreamento a partir do zero. O nosso trabalho de elaboração de um modelo-intermediário foi iniciado em [Toranzo, 1998a] e refinada do decorrer deste anos. Está fora do escopo deste trabalho o tratamento de requisitos não-funcionais porque acreditamos que seu o rastreamento e implicações são assuntos para um outro trabalho de doutorado;

5. Propor um processo composto de um conjunto de diretrizes para desenvolver um modelo de rastreamento. Este trabalho fornece um processo composto de um conjunto de diretrizes que visam identificar as classes e relacionamentos que podem compor o modelo de rastreamento de um projeto. Consideramos isso importante porque todas as pesquisas estudadas ([Jarke, 1998], [Pohl, 1996a], [Pinheiro, 1996a], [Ramesh, 2001] e [Gotel, 1996b]), não apresentam um processo bem definido para complementar e verificar melhor suas próprias propostas de trabalhos. O processo proposto foi gradualmente melhorado no decorrer da sua aplicação no desenvolvimento dos modelos de rastreamento dos diferentes projetos. A experiência obtida com os estudos de casos mostrou que quanto mais tarde se descobre a omissão de um item de rastreamento, mais trabalho e tempo serão necessários para manter as matrizes consistentes. Por exemplo, se o elemento *Subsistema* é introduzido em um modelo de rastreamento de um software que possui um estado avançado de implementaão, então todos ou quase todos os requisitos deverão ser relacionados (alocados) para cada um dos subsistemas identificados desse projeto. É importante dizer que o processo gera um modelo conceitual do rastreamento de requisitos de um software e não se compromete com nenhuma ferramenta ou abordagem de rastreamento. Cabe ao usuário fazer os ajustes necessários no modelo conforme a ferramenta ou abordagem particular;
6. Desenvolver estudos de casos para validar a proposta. Antes de iniciar os estudos de caso, os participantes no rastreamento do requisitos tiveram um treinamento teórico e prático sobre o rastreamento de requisitos. Conheceram as contribuições citadas anteriormente na seguinte ordem: níveis de informação, modelo intermediário e processo. Os trabalhos práticos (rastrear requisitos) dos participantes foram acompanhados e supervisionados para identificar erros e conflitos visando melhorar o processo de rastreamento do nosso trabalho.

1.7 Estruturação da Tese

O resto desta tese está estruturado da seguinte forma.

O Capítulo 2 apresenta uma revisão e comparação de várias pesquisas importantes na área de rastreamento de requisitos. Para cada uma das pesquisas são apresentados alguns dos seus prós e contras. Além disso, é apresentada uma revisão de algumas ferramentas para a gerência de requisitos.

O Capítulo 3 apresenta e exemplifica a proposta dos níveis de informação para classificar as informações a serem rastreadas. Também é apresentado, instanciado e comparado o nosso meta-modelo.

O Capítulo 4 apresenta e exemplifica um processo para desenvolver um modelo de rastreamento sobre um sistema de controle de locadora. Uma proposta para registrar o raciocínio de alguns problemas é apresentada e exemplificada.

O Capítulo 5 apresenta um outro estudo de caso de rastreamento, nesta oportunidade, sobre um sistema de biblioteca que visa integrar as diferentes bibliotecas das diversas sedes da UNIOESTE - Universidade Estadual do Oeste do Paraná. O estudo de caso apresenta uma parte do sistema que está sendo implantado na UNIOESTE.

Finalmente, o Capítulo 6 apresenta as conclusões da pesquisa e os futuros trabalhos.

Capítulo 2

Rastreamento de Requisitos

Este capítulo apresenta uma revisão do rastreamento de requisitos. Inicialmente, o rastreamento de requisitos é apresentado no contexto da gerência de requisitos. Em seguida, a importância e os obstáculos do rastreamento de requisitos são apresentados. Concluímos o capítulo com a revisão de importantes pesquisas acadêmicas e ferramentas industriais relacionadas com o rastreamento.

2.1 Introdução

A gerência de requisitos visa estabelecer e manter um acordo com o cliente em relação aos requisitos a serem observados no projeto de software [Caputo, 1998]. Algumas das atividades da gerência de requisitos incluem: rastreamento de requisitos, controle de mudança, controle de versão e rastreamento do estado dos requisitos. A seguir forneceremos uma explicação dessas atividades.

O rastreamento de requisitos é a habilidade para descrever e seguir a vida de um requisito tanto para frente como para trás, isto é, desde sua origem, através do seu desenvolvimento, e uso nos futuros refinamentos e iterações em quaisquer das etapas do desenvolvimento [Gotel, 1996a]. Uma atividade importante para o sucesso da gerência de requisitos é o rastreamento de requisitos que estabelece relacionamentos entre os requisitos e os elementos produzidos no processo de software para gerenciar as propostas de mudanças e garantir o sucesso dos sistemas [Carvalho, 2001].

O controle de mudança supervisiona as mudanças sobre os requisitos. A evolução e a criação de novos requisitos são legítimas e inevitáveis nos projetos. Os processos de negócio, as oportunidades de *marketing*, os produtos competitivos e a tecnologia de software podem mudar durante o processo de software, podendo afetar alguns dos seus requisitos e, conseqüentemente, ter um grande impacto sobre o sistema. O impacto de uma mudança sobre o desenvolvimento pode ser obtido através de uma análise de impacto que identifica e determina, entre outros, os requisitos, pessoas e elementos afetados pela introdução de uma mudança. Um artefato representa uma peça de informação (modelos, arquivos executáveis e arquivos fontes) que é usado ou produzido por um processo de desenvolvimento de software [Krutchten, 2000]. Uma das atividades-chave da análise de impacto é a inspeção dos relacionamentos estabelecidos entre os requisitos e os artefatos. Algumas das atividades do controle de mudanças são: registrar, avaliar, decidir, implementar e verificar uma proposta de mudança.

Uma outra atividade essencial da gerência de requisitos é o controle de versão. Toda versão de um documento de requisitos deve conter uma identificação única e todos os integrantes da equipe de desenvolvimento devem poder acessar esse documento. É recomendável que a introdução das mudanças sejam claramente documentadas e comunicadas a todas as partes afetadas e que o controle de versão esteja sob a responsabilidade e coordenação de uma ou duas pessoas da equipe de desenvolvimento para evitar confusão, conflitos e problemas de comunicação entre os desenvolvedores.

Uma outra atividade da gerência de requisitos é o rastreamento do estado dos requisitos. O termo estado refere-se ao conteúdo ou valor das propriedades dos

requisitos (por exemplo, prioridade, autor, data de criação, e acesso). A inspeção do estado das propriedades dos requisitos no processo de desenvolvimento é um aspecto importante para melhorar a monitoração dos projetos. Por exemplo, inspecionar os valores contidos nas propriedades dos requisitos, por exemplo, prioridade e implementação, contribui na identificação dos requisitos de alta prioridade que foram implementados [Caputo, 1998].

O resto do capítulo está estruturado como segue. A Seção 2.2 apresenta a importância do rastreamento de requisitos. Os obstáculos encontrados na área de rastreamento de requisitos são apresentados na Seção 2.3. Uma revisão das pesquisas mais importantes na área de rastreamento de requisitos é apresentada e discutida na Seção 2.4. Em seguida, a Seção 2.5 identifica e explica os critérios comparativos usados para comparar as pesquisas apresentadas na Seção 2.4. Seguidamente, a Seção 2.6 fornece uma visão geral de algumas ferramentas industriais para a gerência de requisitos. Finalmente, a Seção 2.7 apresenta as considerações finais do capítulo.

2.2 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:

1. Qualidade de software;
2. Melhoria contínua de um processo;
3. Análise de impacto e implementação de uma proposta de mudança;
4. Manutenção de software;
5. Melhoria do acompanhamento do progresso de um projeto.

A seguir explicaremos cada um dos fatores.

1) Qualidade de software. A área de qualidade de software tem-se caracterizado pelo estudo e proposta de normas (ISO [ISO-9003]) e modelos de qualidade (CMM [Paulk, 1993]), que, se aplicadas corretamente, poderiam contribuir para obter um software com maior qualidade. O rastreamento de requisitos é uma das atividades exigidas no nível 2 do CMM (*Capability Maturity Model for* Software). Uma das contribuições do rastreamento de requisitos para a gerência de requisitos e a qualidade de software é contribuir com a manutenção e integridade de um sistema de informação composto dos relacionamentos estabelecidos entre vários elementos do processo de software. Esse sistema de informação permite que os requisitos do

sistema estejam consistem com suas implementações e com os documentos nos quais são referenciados.

2) Melhoria contínua de um processo. Se considerarmos o conjunto de relacionamentos estabelecidos entre diferentes elementos do processo de software, então poderia ser possível responder algumas perguntas que contribuiriam para a melhoria de um processo de software. Por exemplo, consideremos 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:

   a) Por que um projeto foi bem sucedido ou mal sucedido?

      A inspeção dos relacionamentos entre tarefa e requisitos pode indicar como os requisitos dos clientes foram distribuídos no decorrer de um projeto.

   b) Quais foram os artefatos produzidos por cada programador?

      Uma alocação de recursos humanos para uma tarefa 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.

   c) 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 sistema.

   d) Quais foram os requisitos alocados nos diferentes subsistemas?

      Se foram estabelecidos relacionamentos entre requisitos e subsistemas, então uma inspeção desses relacionamentos pode identificar os subsistemas que implementam os requisitos do projeto. Isto é útil quando desejamos reusar um subsistema no desenvolvimento de um outro sistema.

Essas e outras perguntas podem ser formuladas em um determinado momento em que os profissionais que participaram de um projeto não trabalham mais na organização. Além disso, se o rastreamento não foi realizado, então muito tempo e dinheiro serão gastos na manutenção porque o software será observado como uma caixa preta em lugar de um conjunto de relacionamentos que identificam os programas nos quais requisitos foram implementados. Tradicionalmente, determinar os efeitos colaterais das mudanças no software tem sido realizado através de uma inspeção do código fonte e revisão da documentação. Isso pode funcionar de forma eficiente para pequenos sistemas, mas não para os grandes.

3) 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. Isso 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 realística;
4) Manutenção de software. Se um erro é detectado na execução de um sistema, então o erro deveria ser identificado, fixado, e as razões da sua causa, elicitadas. Se o rastreamento de requisitos estabeleceu relacionamentos entre requisitos e programas, então existe uma alta probabilidade de identificar e corrigir o erro porque identificaremos o programa que implementa o requisito que apresenta problemas. Como afirmado por [Palmer, 1997], um dos benefícios do rastreamento de requisitos é reduzir o tempo e custo da manutenção de um sistema;
5) 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 obter informações mais realísticas do verdadeiro 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 porque fornece um conjunto de relacionamentos entre requisitos e artefatos que facilitam o trabalho das outras atividades. Por exemplo, a manutenção de um sistema com centenas de requisitos poderia ser realizada 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 de 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 de desenvolvimento, mesmo que o custo do investimento no rastreamento de requisitos não tenha um retorno imediato. O recomendável será sempre rastrear durante o processo de desenvolvimento, e não depois, ou apenas quando existir um problema ou proposta de uma mudança.

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

2.3 Obstáculos Encontrados no Rastreamento de Requisitos

Nesta seção identificamos e discutimos alguns obstáculos que dificultam o progresso do rastreamento de requisitos. Esses obstáculos estão situados dentro de um espectro que varia da falta de reconhecimento e comprometimento organizacional, até chegar à pouca pesquisa sobre rastreamento de requisitos que incluam aspectos organizacionais. A seguir apresentamos os obstáculos.

1. Falta de reconhecimento e comprometimento organizacional. Um dos primeiros passos para melhorar um processo de desenvolvimento 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. O maior e mais importante passo na introdução da prática do rastreamento em uma organização é 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;
2. 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 formar parte de um modelo de rastreamento. Geralmente, as informações a serem rastreadas são classificadas e tratadas pelas diferentes pesquisas em informações de pré-rastreamento e pós-rastreamento de requisitos, porém, surgem as seguintes perguntas:
   a) A forma de classificar a informação é suficiente para identificar os elementos que devem fazer parte de um modelo de rastreamento?
   b) Como podemos classificar as informações dentro do pré e/ou pós-rastreamento?
   c) Existe uma outra forma de classificar as informações a serem rastreadas que sirvam para o pré e pós-rastreamento?

   Não identificamos nas pesquisas estudadas alguma intenção em responder as duas últimas perguntas. Um dos objetivos do nosso trabalho foi responder as duas últimas perguntas através da proposta de uma classificação das informações a serem rastreadas;

3. Falta de um processo bem definido para a elaboração de um modelo de rastreamento de requisitos. Nas diferentes pesquisas estudadas ([Pohl, 1996a], [Ramesh, 2001] e [Gotel, 1993]) não foi encontrado um processo bem definido para elaborar um modelo de rastreamento, em lugar disso, foi encontrado a formulação de algumas perguntas não estruturadas que são insuficientes para guiar a elaboração de um modelo de rastreamento. No Capítulo 4 apresentaremos e exemplificaremos um processo composto de um conjunto de diretrizes que guiam a elaboração de um modelo de rastreamento;
4. Rastreamento é geralmente mal entendido, freqüentemente mal aplicado, e raramente realizado corretamente [Palmer, 1997]. Muitos dos obstáculos apresentados são decorrentes de um fator ou uma combinação de fatores, tais como: a interpretação do rastreamento conforme às experiências pessoais, a falta de acordo entre as pessoas acerca das informações a serem rastreadas, e a complexidade e tipo de projeto;
5. Falta de uma cultura organizacional no uso das boas práticas de engenharia de requisitos, especialmente do rastreamento de requisitos [Sommerville, 2000; Wiegers, 2000; Ramesh, 2001]. 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;
6. 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. Uma exceção a este item são os trabalhos de [Ramesh, 1998; Ramesh, 2001] , [Toranzo, 1999; Toranzo, 2000] e [Jardim, 1999]. Caso exista um modelo de rastreamento com essas informações, a produtividade e o entendimento das pessoas encarregadas do rastreamento podem melhorar, porque evitaria que o desenvolvimento dos modelos de rastreamento fossem iniciados do zero.

Dessa forma, concluímos a apresentação dos obstáculos relacionados com o rastreamento de requisitos. Na próxima seção apresentaremos uma revisão das pesquisas do rastreamento de requisitos que foram estudadas neste trabalho.

2.4 Revisão Crítica das Importantes Pesquisas do Rastreamento de Requisitos

O objetivo desta seção é identificar e apresentar algumas das mais importantes pesquisas sobre o rastreamento de requisitos. Os critérios da seleção das pesquisas

foram fundamentados na contribuição, importância e o enfoque da pesquisa. A revisão inclui os seguintes pesquisadores: Orlena Gotel e Antonhy Filkenstein, Matthias Jarke, Balasubramaniam Ramesh, Klaus Pohl, e Francisco Pinheiro. A seguir apresentaremos uma revisão da pesquisa de Orlena Gotel e Anthony Finkelstein.

2.4.1 Revisão dos Trabalhos de Orlena Gotel e Anthony Finkelstein

Até o início da década de noventa, a aplicação do rastreamento de requisitos foi associar a documentação de projeto (*design*) ao documento de requisitos. Naquela época, [Finkelstein, 1991] identificou um importante problema que indicava a existência de pouca pesquisa sobre o uso do rastreamento para relacionar um documento de requisitos com suas fontes de informações. A aplicação desse tipo de rastreamento de requisitos é importante e independente de qualquer paradigma de desenvolvimento.

Em [Gotel, 1994c], foram apresentados os resultados de um trabalho empírico relacionados 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:

1. 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 básicos: pré-rastreamento de requisitos e pós-rastreamento de requisitos. É importante frisar que ambos os tipos de rastreamentos são importantes, mas também é necessário conhecer e entender que a principal diferença entre as abordagens são as informações gerenciadas e os problemas abordados por cada uma delas [Gotel, 1993; Gotel, 1994b; Gotel, 1994c]. A Figura 2 apresenta a divisão do rastreamento proposta por Gotel e Finkelstein;

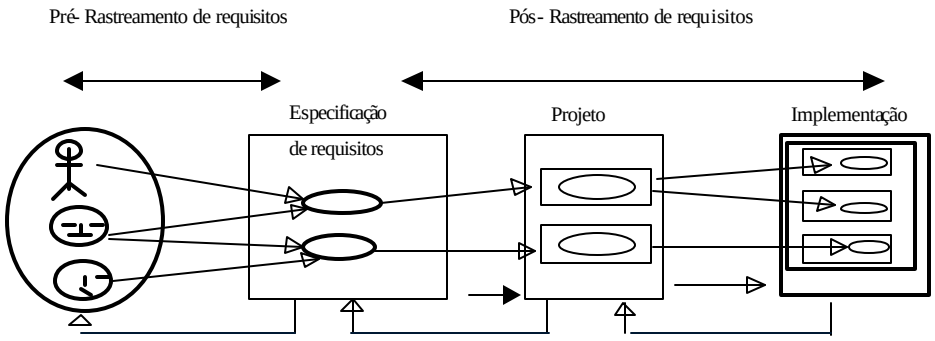


Figura 2: Dois tipos básicos de rastreamento de requisitos

2. Um maior aprofundamento do pré-rastreamento de requisitos porque havia pouca ou nenhuma pesquisa sobre o tema. O pré-rastreamento de requisitos depende da habilidade de rastrear requisitos para suas declarações 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 [Gotel 1994c; Gotel, 1996a];
3. Maior suporte computacional para 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 geralmente poucas porque o pré-rastreamento de requisitos é considerado opcional pelos fabricantes das ferramentas [Gotel, 1996a].

Gotel identificou e abordou 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 a autora 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.

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

O termo *contribuição* (artefato) 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 *mails* informais, diagramas e documentos de requisitos. Os encontros, conversações e reuniões informais podem ser considerados contribuições se os mesmos foram registrados em um meio físico. O trabalho de Gotel não impõe nenhuma restrição sobre o 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 (agentes) 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 inclui as seguintes atividades:

1. Formato de contribuição;
2. Relações baseadas em artefatos;
3. Qualificação do formato de contribuição;
4. Papéis da contribuição social e papéis das relações;
5. 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, a atividade do formato de contribuição 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 documentalista), nos quais os contribuidores podem contribuir na produção das contribuições. O contribuidor principal é responsável por motivar a produção das contribuições. O contribuidor autor é responsável pela organização do conteúdo e estrutura da contribuição. O contribuidor documentalista é responsável pelo registro das informações nas contribuições.

A segunda atividade, chamada relações baseadas em artefatos, tem o objetivo de identificar as relações entre as mesmas contribuições.

A terceira atividade, qualificação do formato de contribuição, consiste em definir um conjunto de atributos e detalhar o grau e estado da contribuição. Por exemplo, uma relação de contribuição entre um documentalista (contribuidor documentalista) e a documentação de projeto (uma contribuição) pode ser qualificada com atributos, tais como, estado(aprovado, reprovado), versão, data e restrição para entender melhor a participação da pessoa sobre a elaboração do documento de projeto.

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: principal, 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 principal 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 é chamada de comprometimento do contribuidor com as contribuições, tem o objetivo de identificar o comprometimento dos contribuidores com as contribuições. Por exemplo, pode ajudar a entender e identificar a responsabilidade e a obrigação do contribuidor sobre a contribuição.

Finalmente, o trabalho de Gotel contribuiu para compreender melhor a 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) podem assumir diferentes papéis enquanto contribuem na elaboração de uma contribuição (artefato). Algumas observações sobre o trabalho de Gotel:

1. Identificação das fontes de informação. Apesar do trabalho de Gotel ter uma atividade chamada formato que identifica os contribuidores (agentes), não ficou claro nem explícito quais tipos de informações deveriam ser consideradas e analisadas para tentar obter conjuntos de contribuidores e contribuições que sejam mais completos. Por exemplo, se for necessário, os aspectos políticos e econômicos, externos a uma organização, devem ser considerados no rastreamento, porque são eles que podem afetar o sistema da empresa (folha de pagamento);
2. Definição de um meta-modelo. O trabalho de Gotel define e exemplifica os conceitos usados para rastrear requisitos, porém os mesmos não são organizados em um meta-modelo que permita conhecer outras formas de relacionar contribuidores e contribuições;
3. Definição e uso de tipos de relacionamento. Gotel propõe relações de contribuição para representar qualquer relacionamento físico (bidirecional) que exista entre um contribuidor e uma contribuição. Porém, em [Gotel, 1996a] as matrizes de rastreamento não refletem o significado dos relacionamentos porque seguem a tendência em usar o símbolo "X" para indicar que dois elementos estão associados, sem refletir maiores informações da associação;
4. Definição e uso de tipos de dependência. O trabalho de Gotel pode ser melhorado com a introdução e uso de tipos de dependência definidos em Yu [Yu, 1994; Yu, 1995a; Yu, 1995b; Yu, 1997a] afim de incrementar o significado dos relacionamentos usados na construção de um modelo de rastreamento;
5. Falta de diretrizes para construir um modelo de rastreamento. Apesar das atividades fornecidas por Gotel para rastrear requisitos, as mesmas não fornecem informações mais detalhadas para a construção de um modelo de rastreamento para um projeto de software;

A seguir apresentaremos uma revisão do trabalho do pesquisador Matthias Jarke.

2.4.2 Revisão do Trabalho de Matthias Jarke

A pesquisa e motivação de Matthias Jarke no rastreamento de requisitos originou-se no projeto *Nature* (*Novel Approach to Theories Underlying Requirements Engineering* [Jarke, 1993]). O principal objetivo desse projeto foi fornecer um conjunto de teorias relacionadas com a representação do conhecimento, análise do domínio e aspectos do processo da engenharia de requisitos. Algumas das contribuições do projeto *Nature* foram:

1. Fornecer o *framework* das três dimensões (especificação, representação e acordo) para a engenharia de requisitos [Pohl, 1993; Pohl, 1994b]. O *framework* foi usado nos trabalhos de rastreamento de requisitos nos projetos *Nature* e de Pohl [Pohl, 1996a];
2. Fornecer modelos que serviram de base para entendimento, reuso e integração de modelos e métodos. O objetivo do projeto *Nature* foi que os modelos e métodos pudessem ser adaptados pelas organizações conforme às suas necessidades;
3. Fornecer uma classificação e identificação dos relacionamentos para o rastreamento [Pohl, 1996a]. Os relacionamentos foram classificados em: condição, conteúdo, documentação, evolução e abstração. Os relacionamentos de condição expressam restrições para um objeto. Os relacionamentos de conteúdo representam uma relação entre o conteúdo dos objetos. Os relacionamentos de documentação são usados para representar associações entre documentos. Os relacionamentos de evolução são usados para representar que um requisito foi substituído por outro. Os relacionamentos de abstração são usados para representar abstrações (generalização e refinamento) entre os objetos;

Em [Jarke, 1996], foram apresentados relacionamentos de rastreamentos entre ferramentas, atividades do modelo de processo e produtos da engenharia de requisitos. A Figura 3 apresenta a classificação dos relacionamentos propostos pelo autor.

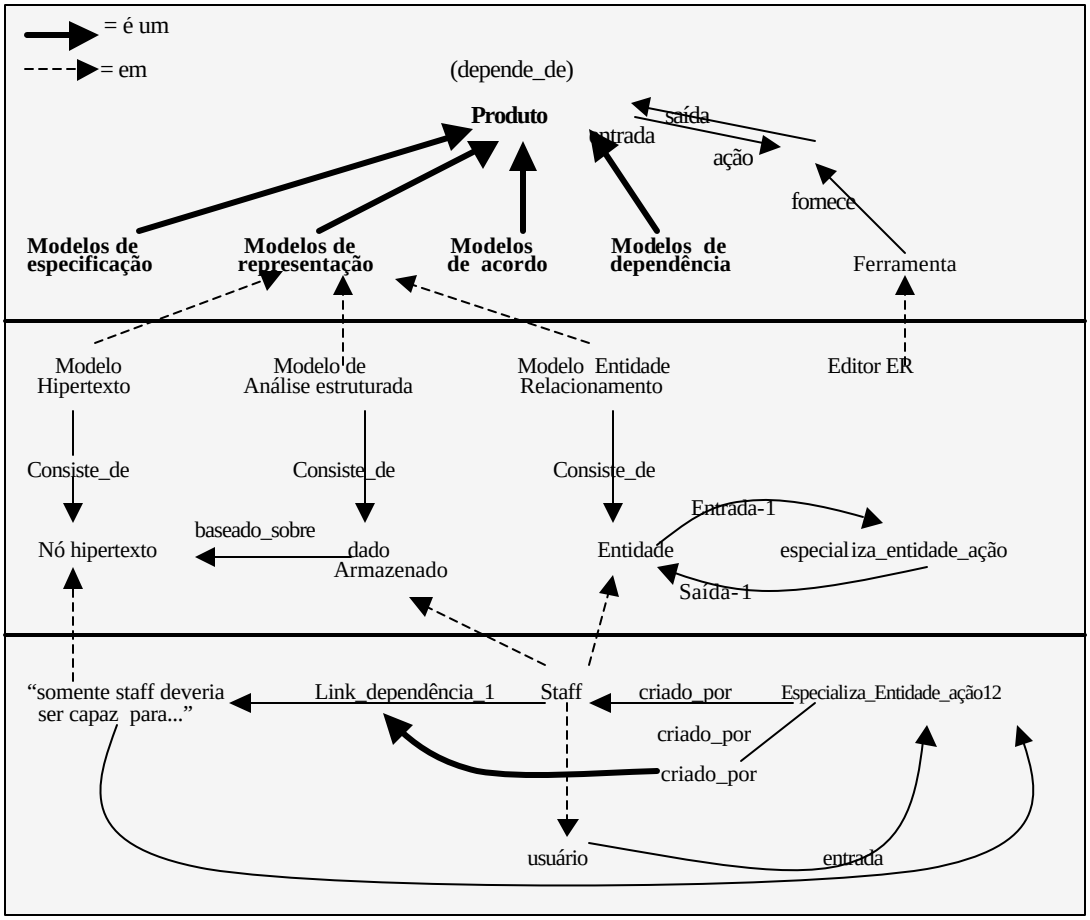


Figura 3: Relacionamentos de rastreamento entre ferramentas, atividades de um modelo de processo e produtos da engenharia de requisitos

O primeiro nível (de cima para baixo), identifica os modelos de especificação, representação e acordo do *framework* das três dimensões (especificação, representação e acordo) da engenharia de requisitos [Pohl, 1996a]. O modelo de dependência expressa a necessidade de manter relacionamentos e a consistência entre os objetos. Por exemplo, manter a consistência entre um requisito (dimensão *especificação*) e o diagrama onde foi representado (dimensão *representação*).

O segundo nível identifica alguns modelos (hipertexto, análise estruturada ou modelo entidade-relacionamento) para representar informação.

O terceiro nível representa as instâncias dos elementos que compõem os modelos do nível dois. Por exemplo, as instâncias (tuplas) de uma entidade do modelo entidade-relacionamento e suas ligações com outras instâncias de entidades. Informações detalhadas dos três níveis podem ser encontradas em [Jarke, 1996] e [Pohl, 1996a].

O Meta-modelo proposto por Jarke está fundamentado nos trabalhos de contribuições estruturadas [Gotel, 1994c], no *framework* das três dimensões da engenharia de requisitos [Pohl, 1996a] e nos modelos de rastreamento apresentados em [Ramesh, 1998a].

A Figura 4 apresenta o meta-modelo de Jarke que visa capturar os aspectos estático e dinâmico do rastreamento de requisitos. Os *stakeholders* e a fonte participam nos aspectos estático e dinâmico. O meta-metamolo proposto por Jarke é composto de quatro relacionamentos (*satisfaz, depende_de, evolui_para* e *raciocínio*), os *stakeholders*, fonte e um relacionamento entre *stakeholders* e fonte. O aspecto estático é capturado pelos relacionamentos *depende_de* e *satifaz*, enquanto o aspecto dinâmico é capturado pelos relacionamentos *desenvolvido_para* e *raciocínio*.

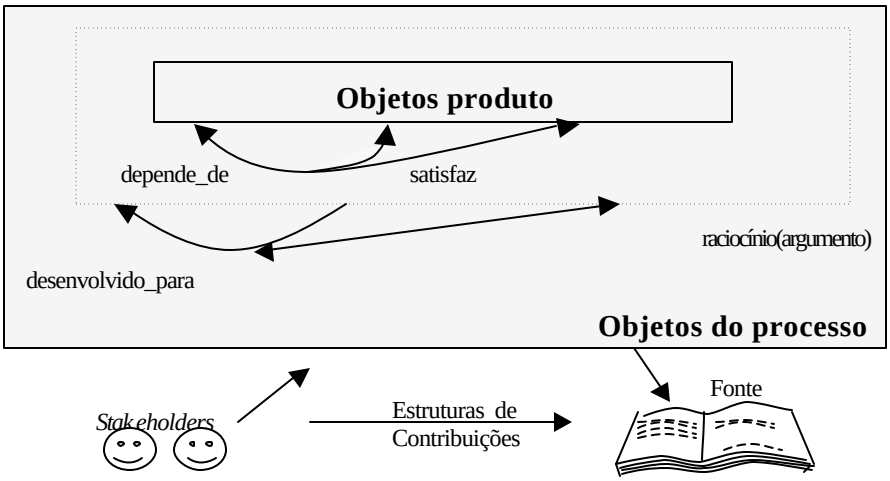


Figura 4 : Meta-modelo de rastreamento de Matthias Jarke

Na Figura 4, os objetos de produtos representam os objetos que fazem parte de um modelo de rastreamento. O relacionamento *satisfaz* estabelece que os objetos de produto são satisfeitos (realizados) por outros objetos de produto. O relacionamento *depende_de* ajuda a gerenciar as dependências (por exemplo, agregação) entre os objetos. Os *stakeholders* capturam a participação das pessoas sobre os objetos de produtos definidos no modelo de rastreamento e sobre os documentos disponíveis (fonte) através do relacionamento "estrutura de contribuições". A fonte expressa que os objetos (de produto ou processo) devem ser documentados.

Os relacionamentos *desenvolvido_para* e *raciocínio* capturam o aspecto dinâmico do rastreamento de requisitos. O relacionamento *desenvolvido_para* rastreia as ações que geram ou modificam objetos do produto. O relacionamento *raciocínio* associa as justificativas (razões) aos passos evolutivos dos objetos do produto.

As conclusões dos trabalhos de Matthias Jarke são as seguintes:

a) O projeto *Nature* implementou técnicas (análise estruturada e entidade-relacionamento) no ambiente chamado PRO-ART [Pohl, 1994a], que foi posteriormente melhorado por Pohl ([Pohl,1996a; Pohl, 1996b]). Entretanto, o problema da falta de tipos de relacionamento continuaram no trabalho de Pohl;
b) Identificamos que o aspecto externo (leis, circulares e normas) não foi considerado de forma explícita. Esse aspecto é importante para alguns sistemas (por exemplo, folha de pagamento, sistema de conta corrente, e declaração de imposto de renda) de algumas organizações (bancos e financeiras);
c) Os meta-modelos são uma forma importante para melhorar o entendimento e as atividades da gerência e rastreamento de requisitos porque permitem conhecer e identificar as associações que podem ser empregadas para construir um modelo de rastreamento;

A seguir apresentaremos algumas observações do trabalho de Matthias Jarke.

1. Os diferentes relacionamentos propostos no meta-modelo de Jarke não capturam nem representam o grau associativo entre os objetos relacionados. O grau associativo é um valor quantitativo ou qualitativo que expressa a dependência (em um sentido da associação) de um objeto com relação a um outro objeto. Por exemplo, a conformidade dos objetivos de um sistema de biblioteca, que estabelece a incorporação de normas internacionais de catalogação, deve ser satisfeita (realizada) através da identificação, especificação e implementação dos requisitos.
2. Muitos dos relacionamentos definidos no projeto *Nature* sobrepõem-se operacionalmente, por exemplo, os relacionamentos de conteúdo e de documentação expressam uma dependência de recurso (de informação ou física). Esses relacionamentos poderiam ser unificados através de algum outro tipo de relacionamento que expresse o mesmo.
3. Falta um relacionamento para expressar que os requisitos são atribuídos (alocados) aos subsistemas. Um relacionamento bem específico e usado nas atividades de rastreamento é o relacionamento *aloca* que expressa que os requisitos do sistema são atribuídos a diferentes subsistemas. Considerando as definições e classificação dos relacionamentos propostos por Jarke, é certo afirmar que não é possível representar naturalmente o relacionamento alocado.

Com este último item concluímos a revisão da pesquisa de Jarke. A seguir apresentaremos uma revisão da pesquisa do rastreamento de requisitos de Ramesh.

2.4.3 Revisão do Trabalho de B. Ramesh

Uma importante preocupação no desenvolvimento dos sistemas complexos, de tempo real, e de grande porte, é garantir que o desempenho do sistema esteja em conformidade com os requisitos especificados [Ramesh, 1993]. É evidente e natural que durante o ciclo de vida de um sistema muitas decisões e análises de custo-benefício afetarão vários artefatos e requisitos do sistema. Portanto, é importante manter um modelo de rastreamento de requisitos.

Baseado em um estudo de caso da migração de um sistema para uma outra linguagem de programação, Ramesh propôs um modelo de rastreamento (vide Figura 5) que pode ser usado para examinar o significado das informações capturadas [Ramesh, 1995a]. A seguir apresentaremos esse modelo.

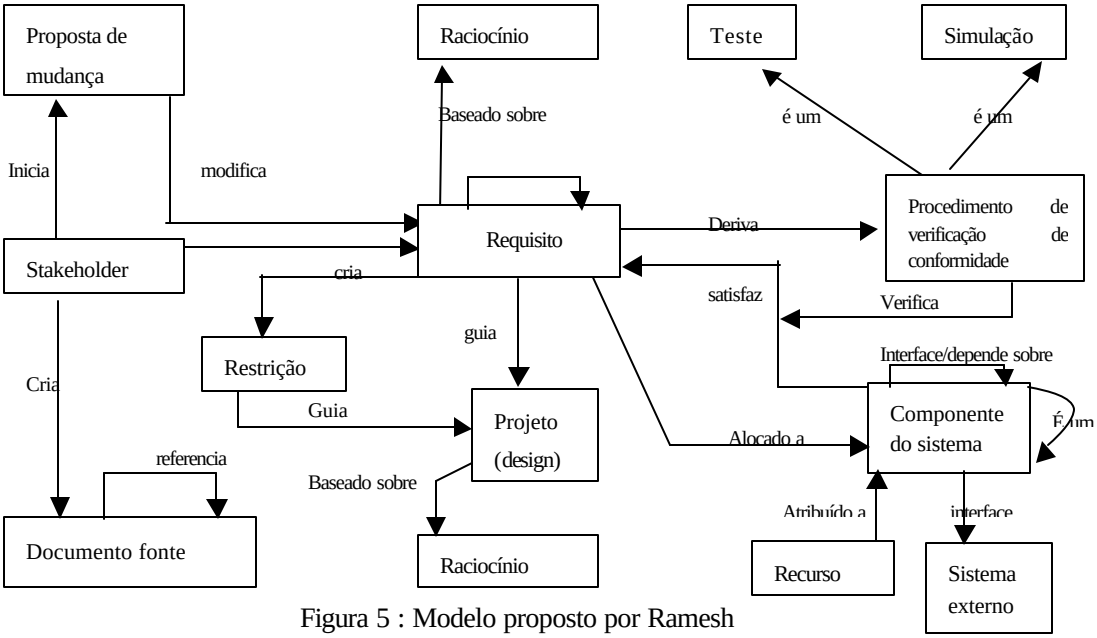


Figura 5 : Modelo proposto por Ramesh

O modelo da Figura 5 expressa o raciocínio associado com os requisitos e com o projeto para capturar as justificativas associadas com a geração e evolução dos mesmos. As propostas de mudanças são associadas com os requisitos para registrar e formalizar suas mudanças. O relacionamento *cria* (de requisito para restrição) expressa as condições impostas sobre os requisitos. O relacionamento *satisfaz* (do componente do sistema para requisito) identifica os componentes que implementam os requisitos. O relacionamento *atribuído a* (de recurso para componente do sistema) representa as pessoas, software e *hardware* que foram necessários para a implementação dos componentes.

Particularmente, criticamos a falta e uso de tipos de relacionamento no modelo da Figura 5, como os tipos existentes na UML (Unified Modelling Language [Booch, 1999]), tais como, agregação, composição e generalização cujos significados são bem definidos, aceitos e independentes do nome atribuído pelos usuários. Informações mais detalhadas do modelo da Figura 5 podem ser encontradas em [Ramesh, 1995b].

Em [Ramesh 1998b], foram identificados alguns fatores dos contextos institucionais (ambiental, organizacional e desenvolvimento de sistemas) que influenciam nas classificações dos usuários e das práticas do rastreamento de requisitos (veja a Figura 6).

O contexto ambiental representa a tecnologia disponível para todos os usuários interessados no rastreamento de requisistos. O contexto organizacional representa as estratégias corporativas para realizar o rastreamento. O contexto desenvolvimento de sistemas representa o policiamento dos procedimentos do rastreamento aplicados no desenvolvimento do sistema.

Os usuários do rastreamento foram classificados em *low-end* e *high-end*, segundo a visão e uso que os mesmos têm sobre os contextos institucionais. No que diz respeito à visão, os usuários *low-end* observam o rastreamento como obrigatório, enquanto que os usuários *high-end* observam o rastreamento como uma importante característica de um processo de engenharia. No que diz respeito ao uso, os usuários *low-end* usam o rastreamento para representar as dependências entre requisitos e a alocação dos requisitos aos componentes de um sistema, enquanto os usuários *high-end* empregam metodologias e procedimentos padronizados do processo de desenvolvimento.

Na Figura 6, os três contextos institucionais (ambiental, organizacional e desenvolvimento de sistema) influenciam três condutas estratégicas para a adoção e uso do rastreamento, denominadas:

1. Condições para adoção e uso do rastreamento;
2. Adoção e uso do rastreamento;
3. Conseqüência da adoção e uso do rastreamento.

Para cada uma das condutas estratégicas existe um conjunto de ações (veja nos retângulos ovalados na Figura 6), que influenciam um outro conjunto de ações (indicada pelas setas horizontais).

Em resumo, em [Ramesh 1998b], foram identificados três contextos institucionais e três condutas estratégicas que explicam a diferença das práticas dos tipos de usuário (*low-end* e *high-end*) do rastreamento de requisitos.

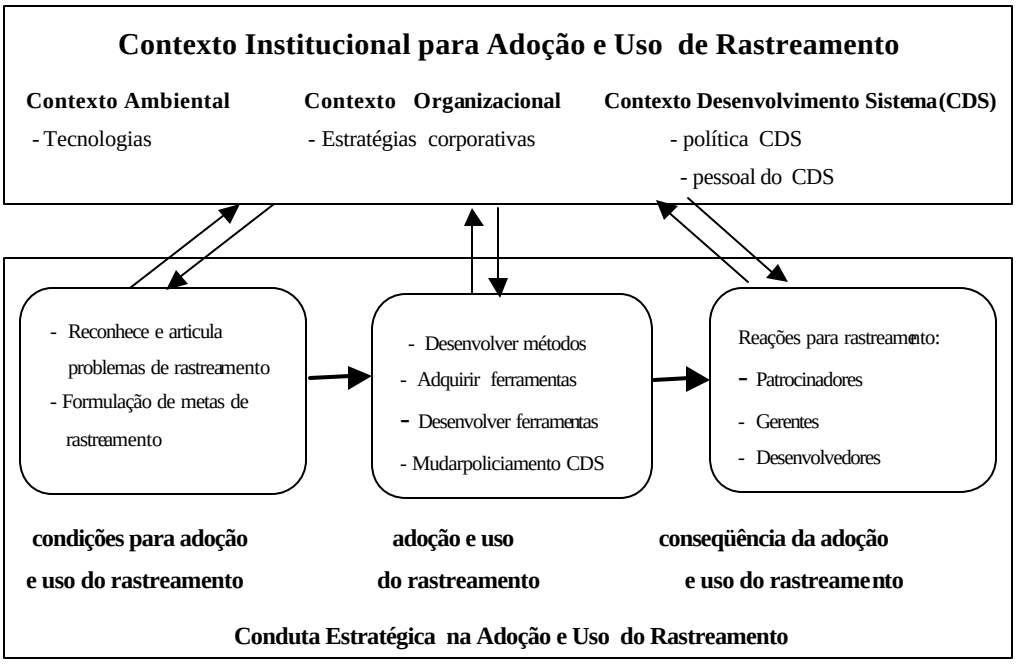


Figura 6: Fatores que afetam a prática do rastreamento

Em [Ramesh, 2001], foram propostos e discutidos modelos de referências para o rastreamento de requisitos. Alguns dos objetivos dos modelos de referência são permitir ao usuário extrair e adaptar os elementos desses modelos na construção de um modelo de rastreamento para um projeto particular. Além dos modelos, o trabalho do autor inclui:

1. Um estudo piloto visando produzir um meta-modelo inicial;
2. A análise de ferramentas para identificar benefícios, falhas e suporte para o rastreamento;
3. Um estudo principal que realizou 30 grupos, que envolveu 26 organizações, e teve a participação de profissionais experientes de diferentes áreas- chave (*Engenharia de* software, *Gerência de requisitos*, etc) do desenvolvimento de software;
4. Classificação dos usuários de rastreamento de requisitos em *low-end* e *high-end*, que é uma extensão do trabalho publicado em [Ramesh, 1998b];
5. Uma discussão sobre a necessidade de melhorar o significado dos relacionamentos empregados nos modelos de rastreamento. Entre os diferentes esforços para melhorar a prática do rastreamento de requisitos foram identificadas as necessidades de construir e melhorar o entendimento dos modelos de rastreamentos e melhorar a interpretação das matrizes de rastreamentos, isto é, qual é o significado atribuído ao

símbolo “x” usado nas mesmas. Este último problema é uma conseqüência da falta de tipos de relacionamento para o rastreamento e da sua representação matricial. Como veremos a seguir, o autor propõe um meta-modelo composto de quatro classes de relacionamentos que podem ser especializados para construir um modelo de rastreamento para um projeto.

Em [Ramesh, 2001], foi proposto e usado um meta-modelo de rastreamento de requisito para guiar a construção e proposta de vários modelos de rastreamento de requisitos. A Figura 7 apresenta o meta-modelo de Ramesh.

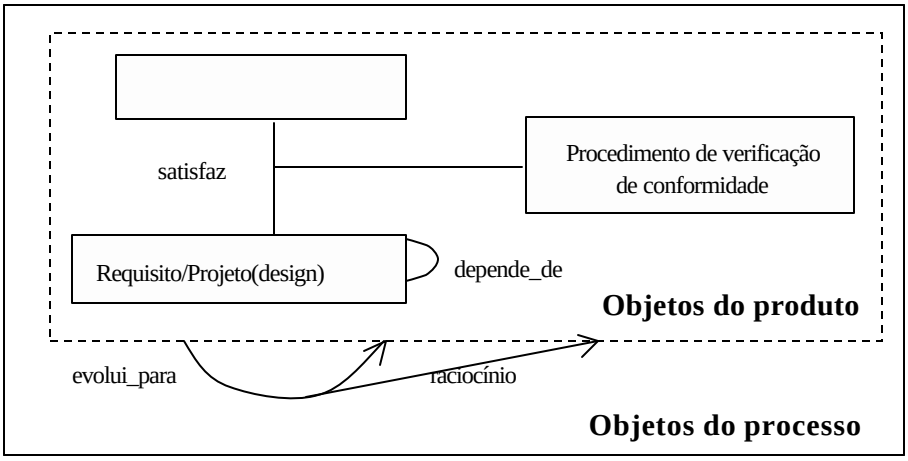


Figura 7: Meta-modelo de rastreamento de Ramesh

Na Figura 7 é expresso que o meta-modelo de rastreamento de Ramesh é composto de quatro relacionamentos: *satisfaz, depende_de*, *evolui_para* e *raciocínio*. Na mesma figura, é modelado que a checagem da satisfação dos requisitos é realizada através de um procedimento de verificação de conformidade.

O relacionamento *satisfaz* estabelece que os requisitos são satisfeitos pelos os objetos de projeto (*design*). O relacionamento *evolui_para* rastreia as ações que geram ou modificam objetos. O relacionamento *raciocínio* associa as justificativas (razões) aos passos evolutivos de um objeto. O relacionamentos *depende_de* ajuda a gerenciar as dependências (agregação é um exemplo) entre os objetos. Pela discussão e exemplos apresentados por [Ramesh, 2001] acerca dos quatro relacionamentos do meta-modelo, podemos inferir que os relacionamentos estabelecidos e ilustrados nos modelos de rastreamento são uma especialização dos relacionamentos do meta-modelo. Os modelos propostos por Ramesh abordam: a gerência de requisitos, o raciocínio, a alocação de projeto (*design*) e a verificação de conformidade.

Na Figura 8, o modelo de gerência de requisitos expressa que as necessidades organizacionais (*Necessidade organizacional*) podem ser operacionais ou estratégicas. As necessidades organizacionais justificam os objetivos de um sistema (*objetivo do sistema*). Os objetivos do sistema geram requisitos e propostas de mudança. O modelo inclui cenários para descrever as necessidades organizacionais, objetivos e requisitos de um sistema. As necessidades organizacionais contribuem na identificação dos fatores de sucesso críticos que guiam os requisitos de um sistema.

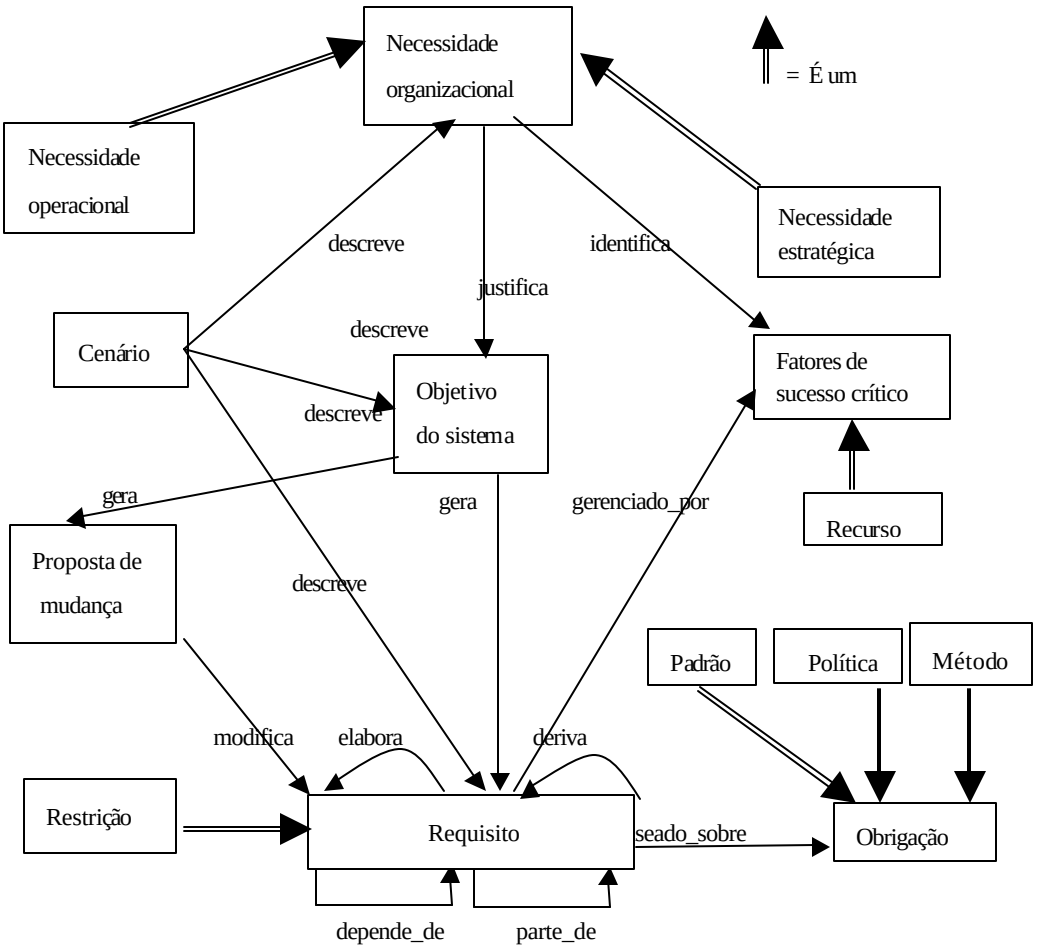


Figura 8: Modelo para a gerência de requisitos

O modelo de gerência de requisitos também expressa que os requisitos podem relacionar-se entre si de quatro formas: *elabora*, *deriva*, *parte_de* e *depende_de*.

O relacionamento *elabora* expressa que os requisitos são realizados a partir de outros requisitos mais elaborados. O relacionamento *deriva* representa a derivação de requisitos. O relacionamento *parte_de*, representa uma composição de elementos. O relacionamento *depende_de*, expressa que um requisito depende de outros.

O relacionamento *baseado_sobre*, entre requisito e obrigação, expressa que os requisitos podem estar relacionados com padrões, políticas e métodos recomendados para um projeto.

No que diz respeito ao modelo da gerência de requisitos, temos algumas observações a fazer:

1. O modelo não representa explicitamente o aspecto ambiental de uma organização;
2. O modelo não expressa que as necessidades organizacionais podem estar compostas ou derivadas de outras necessidades organizacionais;
3. Foi excluído o aspecto social e de responsabilidade, isto é, o modelo não inclui a modelagem das pessoas envolvidas (usuários, analistas, projetistas, etc) nem as responsabilidades das mesmas sobre os artefatos e elementos produzidos. Como indicado por Gotel, um dos importantes problemas do rastreamento de requisitos é a identificação das fontes de informações das diferentes contribuições realizadas no processo de produção e refinamento de requisitos;
4. Falta incluir o conceito de tarefa da gerência de projeto. Idealmente, os artefatos produzidos por um processo de software são desenvolvidos conforme um plano de projeto. A inclusão do conceito tarefa no modelo e seu relacionamento com os requisitos pode ajudar a registrar e entender melhor a ordem da implementação dos requisitos do sistema;
5. Falta de tipos de relacionamento para melhorar o significado dos relacionamentos estabelecidos no modelo. O significado atribuído aos diferentes relacionamentos é nominal. Por ser nominal, o significado de um mesmo relacionamento pode variar de um para outro modelo de rastreamento. Uma solução alternativa para o problema é definir e introduzir tipos de relacionamento. Por exemplo, o significado do tipo de relacionamento agregação da UML é independente do nome atribuído pelo usuário ao mesmo relacionamento.

Continuando com a apresentação dos modelos de rastreamento de Ramesh, a Figura 9 apresenta o modelo de projeto.

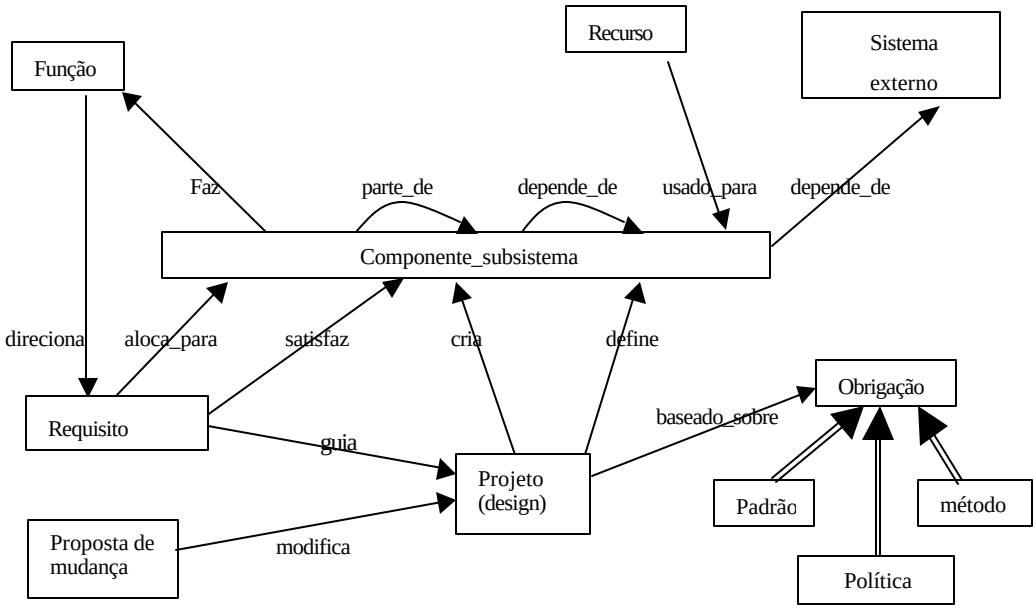


Figura 9: Modelo de projeto

Na Figura 9, os requisitos são alocados (*aloca_para*) para os componentes dos subsistemas e relacionados (*guia*) com o projeto. O projeto representa qualquer atividade de projeto (relacionamentos *cria* e *define*) que produz componentes e que é guiada por padrões, políticas e métodos.

Os *componente_subsistema* podem estar compostos (*parte_de*) de outros componentes, depender de outros componentes (*depende_de*) e de outros sistemas extemos (*depende_de*). O relacionamento entre *componente_subsistema* e *Função* expressa que os componentes implementam (*faz*) funções que são rastreadas para os requisitos funcionais do documento de requisitos. O relacionamento entre *componente_susbsistema* e *recurso* expressa os recursos (pessoal, orçamento, etc) que são usados na implementação dos componentes.

Algumas observações sobre o modelo de projeto são:

1. O conceito de tarefa da gerência de projeto não é representado. Portanto, o modelo não poderia responder às seguintes perguntas:
   a) Quais são os artefatos gerados por uma tarefa de projeto?
   b) Quais foram os requisitos trabalhados em cada tarefa?

Os registros dessas informações são uma boa fonte de informação para entender de diferentes pontos de vista (gerentes, analistas, e projetistas), o êxito ou o fracasso de um projeto;

2. Falta de tipos de relacionamento para melhorar o significado dos relacionamentos estabelecidos no modelo. A fundamentação desse item é igual à apresentada nas observações realizadas sobre o modelo de gerência de requisitos;
3. Foi excluído o aspecto social e de responsabilidade, isto é, o modelo não inclui as pessoas envolvidas (usuários, analista, projetista, etc) nem as responsabilidades das mesmas sobre os artefatos produzidos.

Um outro modelo proposto por Ramesh foi o modelo de raciocínio para registrar os elementos que formam parte de uma discussão e da tomada de decisão. Para isso, ele baseou-se no modelo IBIS (*Issue Based Information System* [Conklin, 1988]) que ajuda a estruturar o conhecimento do acordo produzido dentro de um projeto. A Figura 10 apresenta o modelo IBIS.

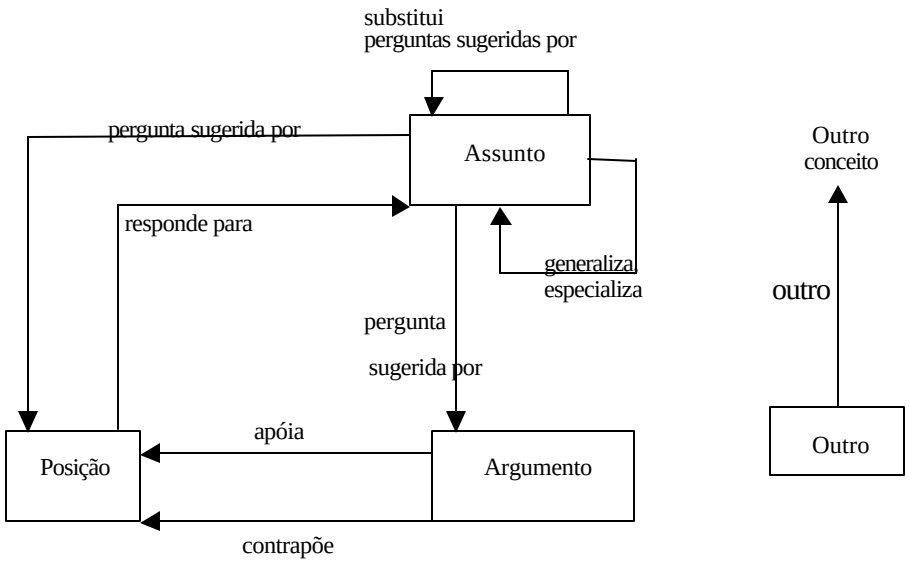


Figura 10: O modelo IBIS (Issue Based Information System)

O modelo consiste de quatro objetos e nove relacionamentos. Os objetos são: *Assunto, Posição, Argumento e Outro*. Os relacionamentos do modelo IBIS são: *generaliza, especializa, apóia, contrapõe, responde para, pergunta sugerida por, substitui, pergunta* e *sugerido por*.

Em relação aos objetos podemos dizer que o conceito assunto é a raiz da estrutura hierárquica. Um *Assunto* pode ser um problema ou uma preocupação que pode requerer

uma discussão e resolução. Cada *Assunto* pode ter várias *Posições* que representam alternativas para sua solução.

Uma *Posição* pode ter vários *Argumentos* de apoio e/ou contra. O conceito *Outro* é um mecanismo de escape para introduzir um novo conceito e associá-lo com os outros conceitos.

Os relacionamentos *generaliza* e *especializa* ajudam a organizar e estruturar um assunto. Um *Assunto* pode relacionar-se com uma *Posição* através do relacionamento *pergunta sugerida por*. Uma *Posição* pode relacionar-se com um *Assunto* através do relacionamento *responde para*. Um *Argumento* pode se relacionar com uma *Posição* através dos relacionamentos *apóia e contrapõe*.

O modelo IBIS [Conklin, 1988], foi usado por Ramesh ([Ramesh, 2001]) para propor seu próprio modelo de raciocínio (veja a Figura 11).

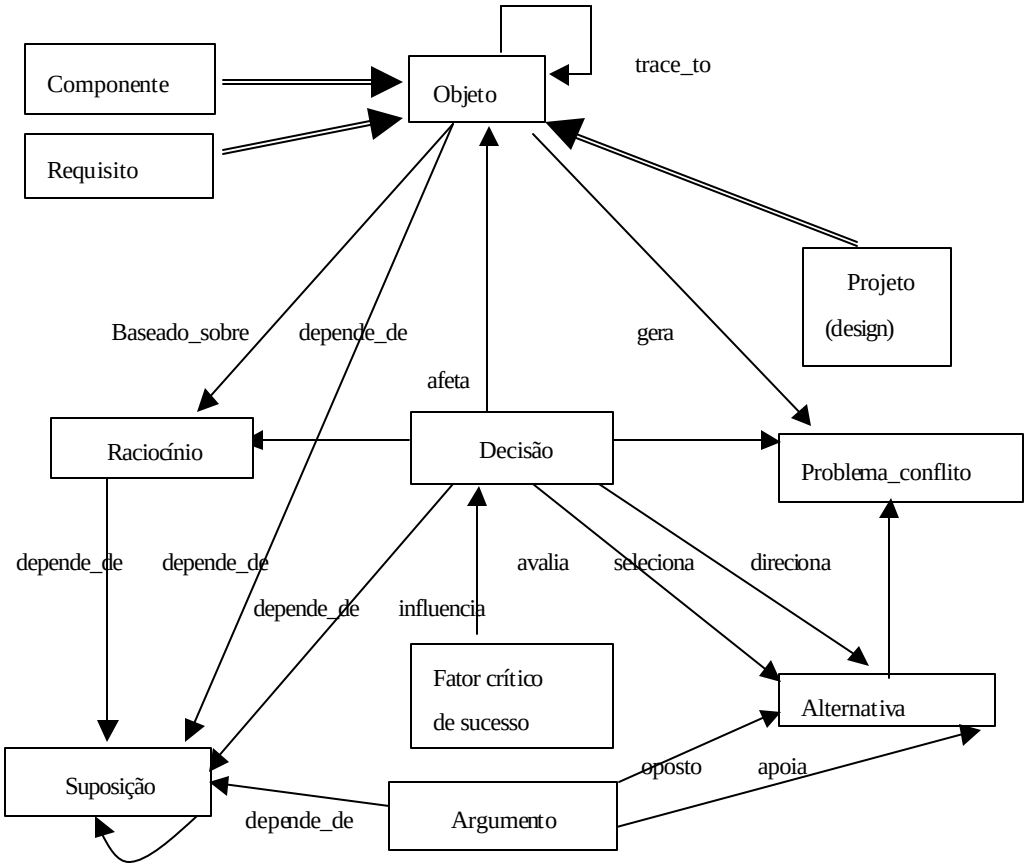


Figura 11: Modelo de raciocínio

A Figura 11 apresenta o modelo de raciocínio de Ramesh, o qual expressa que os objetos geram problemas ou conflitos (classe *Problema_conflito*) devido às diferentes interpretações, supostos, interesses, experiências e objetivos dos *stakeholders* [Ramesh, 2001].

Várias alternativas (classe *Alternativa*) são consideradas na resolução de um problema ou conflito. Os argumentos a favor e contra para cada alternativa podem ser propostos. Esses argumentos podem estar baseados em suposições (classe *Suposição*). A decisão de selecionar uma ou mais alternativas é freqüentemente influenciada pelos fatores críticos de sucesso.

Algumas observações sobre o modelo de raciocínio são:

1. O modelo não expressa que um problema ou conflito pode ter uma restrição associada. Isso significa que o escopo para solucionar um problema pode não estar delimitado. Por exemplo, o problema de determinar qual SGBD (Sistema Gerenciador de Banco de Dados) usar em um sistema da organização pode ser associado à restrição que esse SGBD deve ser um dos existentes na organização do cliente;
2. O modelo não expressa que as decisões obtidas para um problema podem ter alguma restrição imposta. A decisão da escolha de um SGBD pode ter uma restrição que diz respeito ao número de transações por segundo;
3. O modelo não considera o fator humano como produtor e/ou responsável pela tomada de decisões e pelos argumentos que apóiam e/ou contrapõem as alternativas.

De forma semelhante aos modelos de gerência de requisitos e de projeto, o modelo de raciocínio carece de tipos de relacionamento. Maiores informações dos modelos podem ser encontradas em [Ramesh, 2001].

Finalmente, as conclusões das contribuições dos trabalhos de Ramesh podem ser resumidas como seguem:

1. Identifica e discuti os problemas associados com o desenvolvimento de um modelo de rastreamento de requisitos. O modelo pode ser melhorado se levarmos em consideração, de forma explícita, a existência das informações relacionadas com os aspectos organizacionais e os aspectos externos à organização;
2. Propõe um modelo que representa o rastreamento de informação da fase de projeto (*design*) de um sistema. O modelo não está completo, ele pode ser melhorado com a adição de novos objetos e tipos de relacionamento;
3. Propõe uma classificação dos tipos de usuários (*low* e *high*) do rastreamento de requisitos baseada nas práticas dos mesmos;

4. Mostra através de estudos de casos reais, a importância do rastreamento de requisitos para as grandes organizações;
5. Apresenta a proposta de modelos de rastreamento. Apesar dessa contribuição ser interessante, os modelos contem alguns problemas que foram identificados e discutidos junto com a apresentação dos mesmos;
6. A abordagem de Ramesh ([Ramesh, 2001]) não fornece diretrizes de como instanciar os seus modelos de rastreamento para desenvolver um modelo de rastreamento de requisitos para um projeto.

Na subseção seguinte apresentaremos uma revisão do trabalho de pesquisa de Klaus Pohl.

2.4.4 Revisão do Trabalho de Klaus Pohl

Segundo Pohl [Pohl, 1996a], independentemente da abordagem empregada para desenvolver software, uma organização deveria considerar e observar o rastreamento como uma atividade essencial e um importante pré-requisito para a produção de software com qualidade porque o rastreamento fornecerá as informações dos relacionamentos existentes entre os requisitos e alguns artefatos do sistema que são importantes para reduzir o custo e o tempo da manutenção de um sistema.

O trabalho de Klaus Pohl focaliza sua pesquisa no pré-rastreamento de requisitos. Apesar de muitas pesquisas terem incluído o pré-rastreamento de requisitos ([Gotel, 1996a] e [Ramesh, 1993; Ramesh, 1998; Ramesh, 2001]), falta uma abordagem concreta para capturar as informações de rastreamento de um software [Pohl, 1996a]. Pohl formula três perguntas que toda abordagem deveria responder:

1. Qual tipo de informação tem que ser registrada?
2. Como capturar a informação?
3. Como estruturar a informação?

O pré-rastreamento de requisitos de Pohl é aplicado sobre o *framework* das três dimensões [Pohl, 1993; Pohl, 1994b] para descrever o processo de engenharia de requisitos sobre as dimensões de representação, especificação e acordo. Essa proposta considera alguns tipos básicos de informações para serem registradas no pré-rastreamento de requisitos, tais como:

1. Informação relacionada com a dimensão representação que inclui os formatos de representação empregados durante o processo e suas relações;
2. Informação relacionada com a dimensão especificação que inclui o conteúdo da especificação empregado durante o processo e suas relações;

3. Informação relacionada com a dimensão acordo do framework das três dimensões [Pohl, 1993; Pohl, 1994b] que inclui problemas acontecidos, as partes interessadas, os argumentos, as soluções alternativas e as decisões tomadas durante o processo;
4. Relações (dependências) entre os tipos de informações registradas;
5. Relacionamento da informação registrada com os passos do processo e com as fontes de informação envolvidas nesses passos.

No que diz respeito à captura da informação, é importante salientar que sua abordagem é baseada sobre duas premissas: o rastreamento de requisitos deve ser realizado durante a execução de um processo e que o rastreamento de requisitos deve ser automatizado para evitar sobrecarga de trabalho para os desenvolvedores [Pohl, 1994a; Ramesh, 1993; Ramesh, 2001].

A seguir apresentamos o meta-modelo proposto por Pohl.

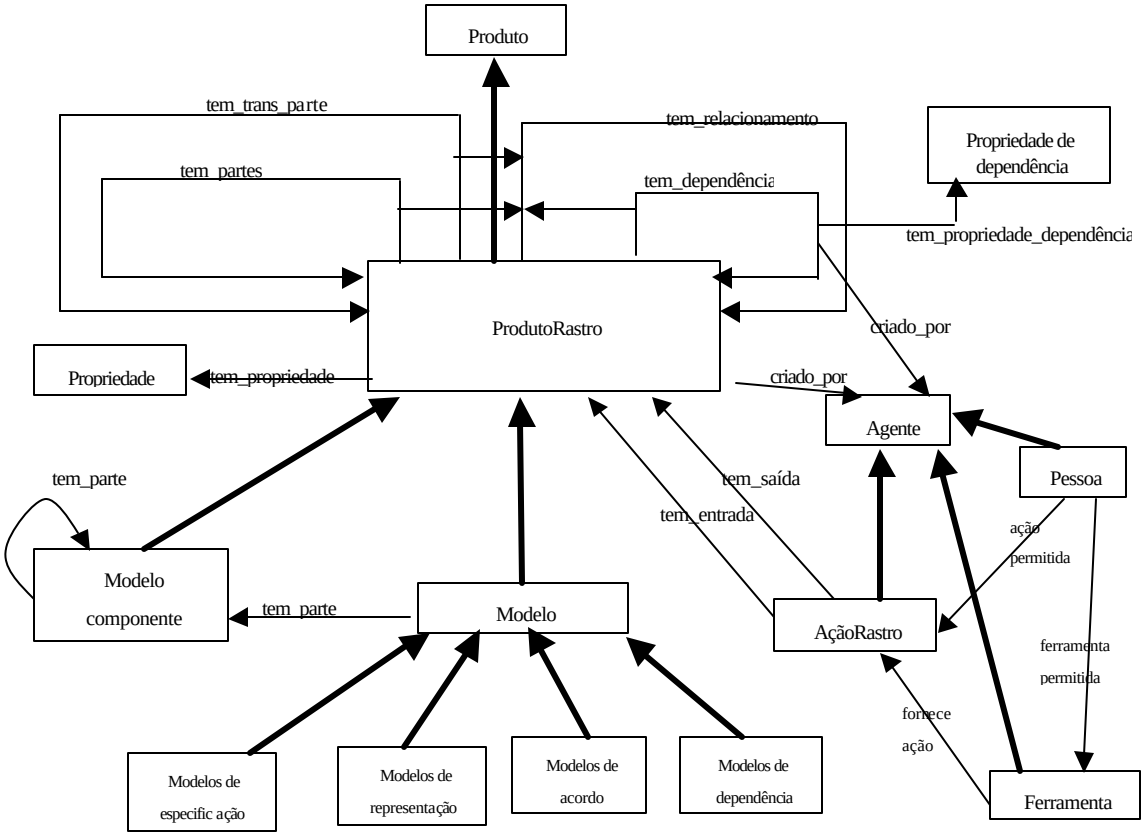


Figura 12: Meta-modelo de Klaus Pohl

Na Figura 12, o conceito *ProdutoRastro* (no centro da figura) é definido como uma especialização da classe *Produto*. As setas com maior largura representam o

conceito "é um" . O *ProdutoRastro* tem algumas propriedades (*tem_propriedade, tem_partes, tem_relacionamentos*, etc) e pode estar composto de outros *ProdutoRastro* (veja o relacionamento *tem_partes* sobre *ProdutoRastro*). Na mesma figura, *ProdutoRastro* é especializado em *Modelo*, que por sua vez, é especializado nos modelos de especificação, representação, acordo e dependência. Um modelo pode estar composto de outros componentes, por exemplo, um modelo de entidade-relacionamento está composto de entidades e relacionamentos.

Baseado no modelo apresentado na Figura 12, Pohl propôs e implementou um ambiente de engenharia de requisitos centrado em processo, chamado de PRO-ART [Pohl, 1994a; Pohl, 1996b].

Finalmente, as nossas conclusões sobre a pesquisa de Pohl são:

1. O trabalho de Pohl está centrado sobre pré-rastreamento de requisitos;
2. Na pesquisa de Pohl não identificamos trabalhos explícitos relacionados com aspectos externos à organização e com a própria organização (objetivos organizacionais, regras, etc);
3. Ao igual que os trabalhos revisados anteriormente, existe uma carência de tipos de relacionamento usados para rastrear informação;
4. Apesar das diretrizes fornecidas por Pohl para rastrear requisitos, consideramos que elas são insuficientes para produzir um modelo de rastreamento para um projeto porque as informações obtidas não são analisadas, nem organizadas.

A próxima subseção apresentará o trabalho do pesquisador Francisco Pinheiro.

2.4.5 Revisão do Trabalho de Francisco Pinheiro

Segundo Pinheiro ([Pinheiro, 1996a]), o rastreamento de requisitos refere-se à habilidade para definir, capturar, e seguir os rastros deixados pelos requisitos sobre os elementos do ambiente de desenvolvimento de software e vice-versa. Um rastro pode ser definido como uma marca de um elemento sobre um outro elemento. Quando isso acontece, então podemos observar aqueles elementos como relacionados de alguma forma.

O autor apresenta uma solução de rastreamento de requisitos no contexto de desenvolvimento de software orientado a objetos. As principais contribuições de Pinheiro para a área de rastreamento de requisitos incluem:

1. A definição de um modelo de rastreamento de software. O termo "modelo de rastreamento de software" é empregado para indicar que o modelo deve rastrear artefatos de software e requisitos;

2. Um ambiente chamado TOOR (*Traceability of Object-Oriented Requirements* [Pinheiro, 1996a; Pinheiro, 1996b; Pinheiro, 1996c]) para automatizar o rastreamento. TOOR implementa uma hierarquia de classes para representar o modelo IBIS [Conklin, 1988], isto é, classes para representar assuntos, posições, argumentos e relações. Os trabalhos de Pinheiro ([Pinheiro, 1996a; Pinheiro, 1996b; Pinheiro, 1999]) não estenderam o modelo IBIS [Conklin, 1988].

Finalmente, o trabalho de Francisco Pinheiro nos ensinou, entre outras coisas, algumas características que deveriam ter um modelo de rastreamento. Aprendemos que um modelo abstrato é importante para rastrear informação e que o rastro entre os elementos pode ser expresso através de axiomas. Além disso, Pinheiro apresenta algumas diretrizes para orientar a construção de um modelo de rastreamento. Entretanto, na abordagem não identificamos alguma preocupação ou trabalho explícito sobre o rastreamento dos aspectos externos às organizações (político, econômico, padrão) e das próprias organizações (objetivos, regras, etc).

Na próxima seção apresentaremos e explicaremos os critérios usados para comparar as pesquisas estudas.

2.5 Quadro Comparativo das Importantes Pesquisas do Rastreamento de Requisitos

Os objetivos desta seção são de reunir e comparar todas as pesquisas apresentadas e fornecer uma visão geral desta tese para facilitar o seu entendimento nos capítulos seguintes. Na explicação dos critérios de comparação aplicados sobre as pesquisas estudadas, forneceremos uma visão da nossa pesquisa. Após apresentação dos critérios, os mesmos são reunidos e apresentados na Tabela 1.

1. Pesquisa pré-rastreamento de requisitos. Indica que o estudo está voltado para o rastreamento de requisitos no processo de produção e refinamento de requisitos. No quadro comparativo, a abordagem de Ramesh é a única abordagem que não trata o pré-rastreamento de requisitos porque não apresentou trabalhos sobre esse assunto. Na pesquisa observamos o pré-rastreamento de requisitos porque é uma atividade que contribui para o entendimento da origem dos requisitos;
2. Pesquisa pós-rastreamento de requisitos. Indica que o trabalho está voltado para o rastreamento de requisitos relacionados com os artefatos produzidos a partir da inclusão dos requisitos no documento de requisitos. As pesquisas que satisfazem estes critérios são: Ramesh, Jarke e Pinheiro. A nossa pesquisa propõe um

meta-modelo e um processo de rastreamento que incluem a questão do pós-rastreamento;

3. Apresenta um meta-modelo para o rastreamento de requisitos. Indica se a abordagem apresenta um meta-modelo para estruturar os diferentes tipos de relacionamento usados para associar os elementos de um modelo de rastreamento. Alguns dos relacionamentos propostos por Gotel não estão organizados em um meta-modelo. Por isso, no quadro comparativo é indicado que Gotel não possui um meta-modelo. Um dos objetivos da nossa pesquisa é propor um meta-modelo de rastreamento visando melhorar a construção e entendimento dos modelos de rastreamento de um projeto;
4. Propõe um modelo de raciocínio. Um modelo de raciocínio contém elementos que ajudam a expressar as razões ou justificativas (prós, contras, suposições, alternativas) associadas às tomadas de decisões de um projeto. As duas únicas abordagens que não possuem um modelo de raciocínio são as de Matthias Jarke e Orlena Gotel. A partir do meta-modelo propusemos um modelo de raciocínio baseado no modelo IBIS (*Issue Based Information System*) [Conklin, 1988]) visando melhorar a captura, representação e registro do raciocínio;
5. Classifica as informações rastreadas de um projeto. Desejamos identificar os trabalhos que classifiquem as informações a serem rastreadas para ajudar o usuário na identificação e elaboração de um modelo de rastreamento. A justificativa do critério é que, independente do pré e pós-rastreamento, uma classificação pode ajudar um usuário a identificar e compreender os tipos de informações (padrão, objetivos, programas, etc) que podem compor um modelo de rastreamento. Infelizmente, nenhuma das pesquisas apresentadas satisfazem esse critério. A nossa pesquisa propõe uma classificação das informações a serem rastreadas visando melhorar a atividade do rastreamento;
6. Usa uma abordagem matemática para o rastreamento de requisitos. Entendemos por uma abordagem matemática como aquela que faz uso de elementos matemáticos (relação, função, axioma e conjunto, por exemplo) para epecificar e rastrear informação. A abordagem de Pinheiro é a única que usa elementos matemáticos. A definição e aplicação de uma notação matemática na elaboração e representação de um modelo de rastreamento estão fora do escopo da nossa pesquisa;
7. Propõe tipos de relacionamento. Esse critério determina se uma pesquisa definiu um conjunto de tipos de relacionamento para facilitar a construção e compreensão dos modelos de rastreamento. Apesar de Ramesh, Jarke e Pohl reconhecerem a importância de melhorar o significado dos relacionamentos usados no rastreamento, os mesmos apresentam pouca discussão relacionada com tipos de relacionamento. A nossa pesquisa parte do princípio que não é fácil definir um conjunto mínimo e

completo de tipos de relacionamento para o rastreamento de requisitos, mesmo assim, definimos e exemplificamos um conjunto de tipos de relacionamento para melhorar a construção e compreensão dos modelos de rastreamento;

8. Atribui significado aos relacionamentos no modelo de rastreamento. Desejamos identificar as pesquisas que definem o significado dos tipos de relacionamentos usados nos modelos de rastreamento. Geralmente, o significado atribuído aos relacionamentos no modelo de rastreamento é compreendido somente pelos seus autores. Por exemplo, as pesquisas de Ramesh e Pohl tratam essa questão de forma insatisfatória porque não usam nem definem o significado dos relacionamentos de um modelo de rastreamento. O trabalho de Gotel propõe e define relacionamentos. No trabalho de Pinheiro o significado atribuído aos relacionamentos é refletido pelo nome definido pelo usuário. Pohl e Gotel, foram considerados como satisfatórios neste critério. Um dos objetivos da nossa pesquisa é atribuir significado a um conjunto de tipos de relacionamento usados para a elaboração dos modelos de pré e pós-rastreamento;
9. Propõe modelo intermediário de rastreamento. Desejamos identificar os trabalhos que propõem modelos de rastreamentos que orientam seus usuários na identificação dos possíveis elementos e relacionamentos que formarão parte de um modelo de rastreamento de um projeto. Somente a pesquisa de Ramesh apresentou modelos mais elaborados para o rastreamento de requisitos que ainda podem ser melhorados. Um dos objetivos da nossa pesquisa é propor um modelo intermediário para evitar que o desenvolvimento dos modelos de rastreamento comece do zero;
10. Fornece um processo para construir um modelo de rastreamento. Independente do pré e pós-rastreamento, desejamos identificar as pesquisas que fornecem um processo para construir um modelo de rastreamento que seja uma forma de instanciar a proposta de rastreamento recomenda por uma pesquisa. Infelizmente, nenhuma das pesquisas estudadas fornece um processo definido para construir um modelo de rastreamento. Por exemplo, Ramesh propôs modelos de rastreamento, mas não apresentou nenhum processo definido para construir um modelo de rastreamento. Nossa pesquisa propõe e exemplifica um processo para construir um modelo de rastreamento;
11. Trata explicitamente aspectos externos (político, econômico e normas) de uma organização. Desejamos identificar as pesquisas que alertam e/ou recomendam aos usuários considerar os aspectos externos à organização que possam afetar alguns dos seus sistemas de informação. Por exemplo, alguns sistemas bancários brasileiros (conta corrente, crédito eletrônico, etc) tiveram que ser modificados afim de satisfazer a lei da CPFM (Contribuição sobre movimentação financeira). Existem outras leis, tais como, retenção de imposto de renda e desconto da previdência social que afetam os sistemas de folha de pagamento. Todas as pesquisas estudadas não

possuem uma discussão explícita e detalhada para lidar com os aspectos externos à organização na elaboração de um modelo de rastreamento;

12. Trata explicitamente aspectos organizacionais. Com esse critério desejamos identificar as abordagens que fornecem recomendações e/ou diretrizes para identificar as informações internas da organização que afetam alguns dos seus sistemas. Um passo importante no desenvolvimento de um sistema qualquer é entender e satisfazer às necessidades organizacionais (objetivos, restrições, regras, etc) da organização que hospedará o sistema. Certamente, as necessidades organizacionais influenciam vários aspectos de um sistema (requisitos, *hardware* e software, por exemplo). Dessa forma, é estabelecida uma dependência que deve ser registrada porque a mudança nos objetivos organizacionais pode afetar alguns dos sistemas da organização. A abordagem de Ramesh foi a única que apresentou interesse por aspectos organizacionais, mesmo assim, consideramos insuficiente seu trabalho. A nossa pesquisa trata este item no processo de elaboração de um modelo de rastreamento;
13. Trata explicitamente a gerência de projeto. Estamos interessados nos trabalhos que identificam, registram e relacionam as tarefas (da gerência de projeto) com os requisitos porque ajuda a identificar os requisitos que foram implementados no decorrer do projeto, e conseqüentemente, a entender o sucesso ou fracasso de um projeto. Infelizmente, todas as abordagens estudadas não apresentaram uma discussão relacionando os aspectos da gerência de projeto e o rastreamento de requisitos.

A seguir apresentaremos um quadro comparativo que reúne os critérios e as pesquisas estudadas.

Critérios de comparação		Identificação dos Pesquisadores				
		Ramesh	Gotel e Filkenstein	Jarke	Pohl	Pinheiro
1	Pesquisa pré-rastreamento de requisitos	Não	Sim	Sim	Sim	Sim
2	Pesquisa pós-rastreamento de requisitos	Sim	Não	Sim	Não	Sim
3	Apresenta um meta-modelo para o rastreamento de requisitos	Sim	Não	Sim	Sim	Sim
4	Propõe um modelo de raciocínio	Sim	Não	Não	Sim	Sim
5	Classifica as informações rastreadas de um projeto	Não	Não	Não	Não	Não
6	Usa uma abordagem matemática para o rastreamento de requisitos	Não	Não	Não	Não	Sim
7	Propõe de tipos de relacionamento	Sim	Não	Sim	Sim	Não
8	Atribui significado aos relacionamentos no modelo	Sim	Sim	Sim	Sim	Não
9	Propõe modelo intermediários de rastreamento	Sim	Não	Não	Não	Não
10	Fornece um processo para construir um modelo de rastreamento	Não	Não	Não	Não	Não
11	Trata explicitamente aspectos externos (político, econômico e normas) de uma organização	Não	Não	Não	Não	Não
12	Trata explicitamente aspectos organizacionais	Sim	Não	Não	Não	Não
13	Trata explicitamente a gerência de projeto.	Não	Não	Não	Não	Não
Total de respostas “Sim”		7	2	5	5	5

Tabela 1: Quadro comparativo das abordagens para o rastreamento de requisitos

A última linha do quadro comparativo totaliza o número de respostas “Sim” obtidos pelos pesquisadores para cada um dos critérios propostos. Considerando os aspectos quantitativo (número de “Sim”) e qualitativo (relevância para a área de rastreamento) para o nosso projeto, foi decidido compreender melhor os trabalhos de Ramesh. Do ponto de vista qualitativo ele fez uma contribuição em propor modelos de referência para o rastreamento de requisitos.

Com a apresentação do quadro comparativo concluímos a revisão das pesquisas acadêmicas. A seguir apresentamos uma visão industrial de algumas ferramentas para a gerência de requisitos que incluem a questão do pós-rastreamento de requisitos.

2.6 Revisão de Ferramentas para a Gerência de Requisitos

Os objetivos desta seção são de identificar as dificuldades de gerenciar manualmente um documento de requisitos e fornecer uma visão geral das facilidades de algumas ferramentas industriais para gerenciar requisitos.

Tradicionalmente, o documento de requisito é baseado na linguagem natural. A ênfase desse documento é conter requisitos claros, consistentes e não ambíguos. Porém, existem propriedades sobre os requisitos que não são registradas no documento de requisitos, por exemplo, a prioridade do requisito, o estado do requisito (aprovado, reprovado, implementado, etc). Essas propriedades ajudam acompanhar e controlar o estado de desenvolvimento de um sistema, por exemplo, uma consulta (requisitos prioritários que foram implementados), podem levar a examinar o conteúdo da propriedade prioridade dos requisitos.

Geralmente, as funcionalidades das ferramentas de gerência de requisitos são: armazenar requisitos; identificar de forma única todos os requisitos; facilitar a seleção manual ou automática dos requisitos de um texto; definir e manipular as propriedades dos requisitos; e gerar filtros (visões) e relatórios sobre os requisitos.

Esta seção não apresenta uma revisão e nem uma comparação detalhada das ferramentas estudadas para a gerência de requisitos porque elas estão constantemente evoluindo. Informações mais detalhadas e comparativas das várias ferramentas de gerência de requisitos estão disponíveis no Conselho Internacional de Engenharia de software (www.incose.org/tools/tooltax.html). A seguir fornecemos uma visão geral das ferramentas RequisitePro, Caliber, QSSrequireit, DOORS e RTM. Começaremos nossa revisão com RequisitePro.

No RequisitePro (versão 2001A.04.00), o primeiro passo a ser realizado para gerenciar um projeto é definir o mesmo. Essa ferramenta usa e estende a funcionalidade do Word, da Microsoft, para: selecionar e marcar um texto como requisito; gerar de forma automática o identificador único do requisito; registrar palavras-chave para os requisitos; e gerenciar as propriedades dos requisitos. O RequisitePro oferece documentos *templates* (de especificação de requisitos e teste) que podem ser reusados na definição de outros projetos. Uma vez ingressados os requisitos, o RequisitePro oferece facilidades para definir visões e propriedades sobre os requisitos. Além disso, as mudanças sobre as propriedades dos requisitos são registradas de forma automática.

Algumas desvantagens do RequisitePro são: não fornecer nem deixar os usuários definirem tipos de relacionamento para associar os elementos de um projeto; não possuir interfaces para ambientes de programação (*Visual C++, Delphi, JBuilder*, por exemplo); a representação matricial dos relacionamentos estabelecidos no projeto é realizada usando um único símbolo; não fornecer um processo definido para elaborar um modelo de rastreamento; e não representar nem manipular de forma natural um modelo de raciocínio.

O Caliber (versão 3.0) considera o papel do administrador para definir projeto, fazer *backup* e controlar o acesso dos usuários. Inicialmente, o usuário deve definir um projeto para definir requisitos, visões, matrizes de rastreamento, etc. O suporte do Caliber para o rastreamento de requisito permite ao usuário visualizar relacionamentos entre requisitos e outras informações, tais como: outros requisitos, conjunto de teste, caso de uso e classes. Caliber possui interface para vários aplicativos (Excel, arquivos gráficos, HTML, áudio e vídeo), fornece funcionalidades para apoiar e registrar informações relacionadas com problemas com os requisitos.

O QSSrequireit (versão 2.0) é uma ferramenta que usa e estende a funcionalidade do Word da Microsoft. A funcionalidade que fornece é restrita porque não existe um controle de acesso dos usuários, e conseqüentemente, um controle de acesso sobre diferentes documentos. Algumas das facilidades da ferramenta são: analisar e extrair requisitos desde um texto; e definir propriedades sobre os requisitos. O QSSrequireit não possui um modelo de raciocínio e não permite definir tipos de relacionamento.

O DOORS (*Dynamic Object-Oriented Requirements System*), versão 5.0, é um pacote de ferramentas (DOORS, DOORnet, e DOORSrequireIT) que visa gerenciar requisitos. O DOORnet (VERSÃO 4.0) é uma ferramenta que permite ao usuário usar o navegador Netscape e Internet Explorer para acessar os dados armazenados no DOORS. A ferramenta DOORS é quem gerencia o banco de dados. O DOORSrequireIT (versão 2.2) é uma ferramenta com funcionalidades semelhantes as do QSSrequireit. O DOORS fornece *templates* para organizar e gerenciar as informações. Essa ferramenta é um produto sofisticado para gerenciar requisitos de grandes projetos. Cada projeto possui uma lista de usuários autorizados. Finalmente, o DOORS inclui uma linguagem *script* para sua personalização ou desenvolvimento de extensões, além disso, possui interfaces, entre outros, para o Microsoft Project e Rational Rose. Igualmente às ferramentas anteriores, não permite definir tipos de relacionamento e usa um único símbolo nas matrizes de rastreamento para representar as associações entre os elementos rastreados.

A ferramenta Requirements Tool Manager *Workshop* é uma completa re-engenharia da ferramenta *Requirements Tool Manager*. A Marconi Systems Technology desenvolveu originalmente o RTM em 1990. Em 1997, a empressa Integrated Chipware começou a gerenciar o produto e tem, desde então, continuamente atualizado o RTM.

O RTM Workshop se integra com o Word e com o Adobe FrameMaker. Além disso, o RTM possui uma interface tipo Windows Explorer para manipular requisitos. Ele possui a facilidade de importar requisitos de diferentes tipos de documentos (por exemplo, Excel e o MS Project) e pode exportar requisitos em vários formatos. A RTM Workshop possui os mesmos problemas que a ferramenta DOOR.

Certamente, todas as ferramentas têm prós e contras que devem ser considerados e avaliados na seleção de uma ferramenta para a gerência de requisitos. A seguir apresentaremos as considerações finais.

2.7 Considerações Finais

Nossa pesquisa aponta que diferentes praticantes do rastreamento de requisitos têm seu próprio entendimento das principais causas e problemas do rastreamento e sua própria visão de como o rastreamento de requisitos pode ser melhorado. Além disso, deve-se acrescentar a falta de compromisso de todas as partes envolvidas no rastramento e os problemas da imaturidade tecnológica na integração (dado e controle) das diferentes ferramentas de rastreamento de requisitos.

Já existe muito suporte computacional para o rastreamento de requisitos, principalmente para o pós-rastreamento. Contudo, o suporte fornecido para o pós-rastreamento não pode ser aplicado completamente sobre o pré-rastreamento, e vice-versa, porque manipulam informações diferentes e abordam problemas diferentes. Por exemplo, diante de uma proposta de uma mudança de requisitos, o pós-rastreamento de requisitos pode contribuir com a habilidade de identificar os artefatos e as partes afetadas do produto final, enquanto que o pré-rastreamento de requisitos pode identificar as fontes de informação dos requisitos.

Todos os trabalhos de pesquisas estudados são interessantes, mas não identificamos uma preocupação forte e constante no rastreamento de requisitos pelos seguintes aspectos:

1. Tratar explicitamente com aspectos externos a uma organização;
2. Tratar explicitamente com aspectos internos das organizações;
3. Tratar explicitamente com aspectos de gerência de projeto;
4. Definir um conjunto de tipos de relacionamento para contribuir para melhorar o significado dos relacionamentos entre os artefatos;
5. Propor um modelo intermediário de rastreamento de requisitos usando um conjunto de tipos de relacionamentos de rastreamento para incrementar o desenvolvimento e o entendimento de um modelo de rastreamento de um projeto.

Muitos desses aspectos são tratados de forma insuficiente pelas pesquisas estudadas. Eles são contemplados e incluídos na nossa abordagem de trabalho que será explicada no próximo capítulo.

Capítulo 3

Proposta de Níveis de Informação e de um Meta-modelo para o Rastreamento de Informação

---

O objetivo do capítulo é propor níveis de informações para classificar as elementos/artefatos que possam formar parte de um modelo de rastreamento. Também propomos um meta-modelo que inclui tipos de relacionamento definidos. Além disso, comparamos nosso meta-modelo com outros meta-modelos.

3.1 Introdução

Um consenso está emergindo na comunidade de desenvolvimento de software que aponta a necessidade de complementar os requisitos de software com modelos organizacionais que descrevam o contexto no qual o sistema desejado operará [Erikson, 2000].

A motivação inicial desse consenso é que os softwares falham no suporte fornecido às organizações. Algumas das principais razões das falhas são a falta de um apropriado entendimento da organização por parte dos engenheiros de requisitos e engenheiros de software, e ao fato de que as mudanças organizacionais não podem ser acomodadas facilmente no software [Alencar, 2000].

Logo, a captura de requisitos é uma fase crítica do processo de desenvolvimento de software porque trata de problemas organizacionais, gerenciais, econômicos e sociais [Castro, 2001b]. Também, é geralmente reconhecido que uma cuidadosa atenção aos problemas organizacionais é crucial para o sucesso dos sistemas de informação [Yu, 1997c].

Portanto, considerando esses problemas da perspectiva do rastreamento de requisitos, surgem as seguintes perguntas: Existe uma classificação das informações que podem fazer parte de um modelo de rastreamento? Quais seriam os benefícios dessa classificação? Por que não dedicamos uma maior atenção ao rastreamento de requisitos do ponto de vista organizacional e externo (aspectos político, econômico da organização e/ou padrões internacionais)? Existem diversos trabalhos que sugerem a necessidade de considerar o aspecto organizacional dentro do rastreamento de requisitos, por exemplo, [Morris, 1994], [Jarke, 1993; Jarke, 1996], [Yu, 1995b] e [Ramesh, 1998b; Ramesh, 2001].

A estrutura do capítulo é como segue. A Seção 3.2 apresenta uma proposta para classificar as informações a serem rastreadas. A Seção 3.3 apresenta e exemplifica um meta-modelo para o rastreamento de requisitos. A Seção 3.4 compara nosso meta-modelo com outros meta-modelos. Finalmente, as conclusões do capítulo são apresentadas na Seção 3.5.

3.2 Proposta de Níveis de Informação para o Rastreamento de Requisitos

As pesquisas apresentadas no capítulo anterior, em geral, formulam algumas perguntas básicas para identificar os elementos que serão rastreados, mas não fornecem

uma classificação dos elementos que justifique ou esclareça a formulação das perguntas. Por exemplo, o trabalho de Pohl apresenta algumas perguntas que são insuficientes para instanciar sua própria proposta. Sabemos que os objetivos das perguntas de Pohl ([Pohl, 1996a]) são identificar as informações a serem rastreadas, mas surgem outras perguntas, tais como: Que tipo de informação identificar? Podemos classificar as informações para entender melhor a atividade de rastreamento? As tarefas da gerência de projeto e o aspecto organizacional são consideradas e relacionadas?

O pré-rastreamento proposto por Gotel estabelece uma clara necessidade de rastrear os requisitos para as atividades que produziram e refinaram os mesmos, mas surgem as mesmas perguntas sem resposta às formuladas no trabalho de Pohl.

Diante do problema identificado propomos uma classificação dos elementos de rastreamento composta de quatro níveis de informação: externo, organizacional, gerencial e desenvolvimento. É importante salientar que está fora do escopo deste trabalho a identificação e a descoberta de sobreposição de informações nos diferentes níveis de informação porque isso varia de organização para organização. Independente do pré e pós-rastremento, o nosso objetivo é que os níveis de informação contribuam para os usuários serem mais precisos na formulação das perguntas para a identificação das informações candidatas para o modelo de um projeto. Isto é, preparar perguntas para identificar de cada um dos níveis os elementos que formarão o modelo de rastreamento de um software.

A Figura 13 apresenta e exemplifica os níveis de informações da classificação.

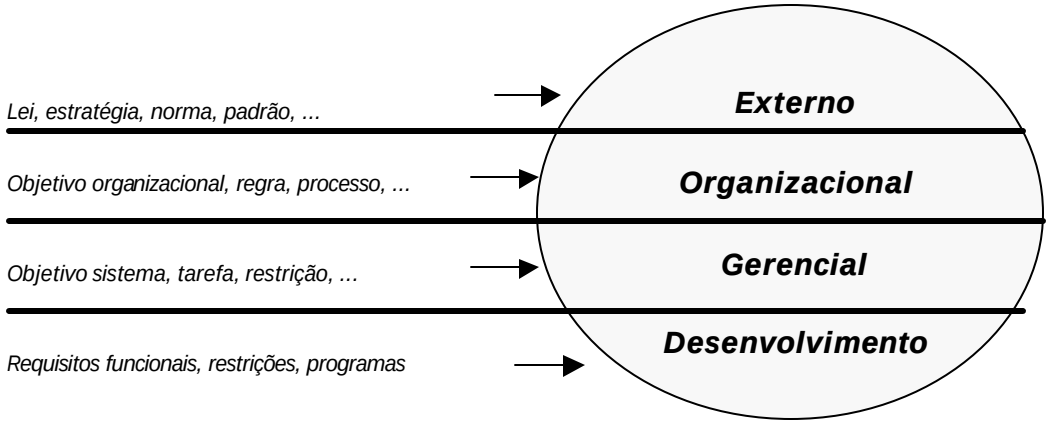


Figura 13: Classificação da informação do rastreamento

Antes de iniciar a explicação de cada um dos níveis de informação nas subseções seguintes, podemos afirmar que nosso trabalho é a única contribuição que expressa e propõe de forma explícita uma classificação das informações do rastreamento de requisitos. A nosso entender, a classificação constitui-se em um pequeno passo para

facilitar a identificação das informações do rastreamento de requisitos de um projeto. É importante salientar que os níveis de informações podem conter informações comuns. A seguir explicaremos o nível de informação externo.

3.2.1 Nível Informação Externo

A informação do nível externo representa os conceitos relacionados com o contexto político e econômico no qual uma organização está inserida. Existem alguns elementos, por exemplo leis e normas, que são externos a uma organização e que podem afetar alguns dos sistemas de informação de uma organização.

Uma lei é uma regra de direito ditada pela autoridade e tornada obrigatória para manter, numa comunidade, a ordem e o desenvolvimento. Por exemplo, a Lei da Contribuição Provisória sobre Movimentação ou Transmissão de Valores e de Créditos e Direitos de Natureza Financeira (CPMF [Receita, 2001]), considera como movimentação de valores e de créditos e direitos de natureza financeira qualquer operação liquidada ou lançamento realizado pelas instituições financeiras, que representem circulação escritural ou física de moeda. A mesma lei considera vários tipos de geradores de contribuição, tais como, o lançamento a débito, por instituição financeira, em contas correntes de depósito, em contas correntes de empréstimo, e de depósito judicial. Porém, a CPMF não incide, por exemplo, a) no lançamento nas contas da União, dos Estados, do Distrito Federal, dos Municípios, etc; b) no lançamento para pagamento da própria contribuição; c) nos saques efetuados diretamente nas contas vinculadas do Fundo de Garantia do Tempo de Serviço - FGTS e do Fundo de Participação PIS/PASEP e d) no saque do valor do benefício do seguro-desemprego.

A implementação da CPMF obrigou todos os bancos a criarem e/ou modificarem alguns objetivos de negócio sobre alguns dos sistemas bancários (por exemplo, conta corrente, empréstimos e fundo de garantia) para implementar e satisfazer a lei. Dessa forma, alguns sistemas foram modificados para calcular, recolher, reter, e registrar dados relacionados com a CPMF. A lei da CPMF estabelece que as instituições responsáveis pela retenção e recolhimento das contribuições fornecerão à Secretaria da Receita Federal as informações necessárias para a identificação dos contribuintes e os valores globais das respectivas operações.

Assim como a CPMF, existem outras leis externas às organizações que podem afetar alguns dos seus sistemas de informação. Essas leis extrapolam o escopo dos processos de produção de requisitos e de desenvolvimento de sistema porque os analistas não têm acesso às pessoas que propõem e discutem as leis. Logo, os responsáveis pelo processo de produção de requisitos necessitam da asessoria de outros

profissionais para interpretar e identificar as implicações das leis para os sistemas de informação da organização.

Note-se que no nível de informação externo as informações são expressas geralmente em termos de objetivos e regras políticas e econômicas que precisam ser satisfeitas.

Do ponto de vista do rastreamento de requisitos, a inclusão explícita do nível externo ajuda a tornar mais claro o que Gotel, Pohl e Ramesh podem denominar fontes de informação. As informações representadas pelo nível externo são uma fonte de informação, porém, Gotel, Pohl, Pinheiro e Ramesh não fazem um relacionamento e esclarecimento explícito do conceito de fonte de informação com as informações do nível externo. Os relacionamentos propostos nesta tese para elaborar um modelo de rastreamento são: generalização, agregação, satisfação, recurso, responsabilidade, representação e aloca. Cada um deles serão abordados em detalhes na Seção 3.3.

A seguir explicaremos o nível de informação organizacional.

3.2.2 Nível Informação Organizacional

Existem muitos trabalhos, por exemplo, [Erikson, 2000], [Mylopoulos, 2000a] e [Yu, 1997c], que visam mostrar a importância de aumentar o conhecimento dos requisitos organizacionais de um sistema. Por exemplo, a manutenção de um sistema, que deve satisfazer novos requisitos, torna necessário um melhor entendimento dos objetivos organizacionais da organização para decidir que mudanças deverão ser realizadas. O Projeto Tropos [Mylopoulos, 2000b; Mylopoulos, 2001] usa os modelos de dependência estratégica e dependência de razão [Yu, 1995b] para obter um bom entendimento e raciocínio do ambiente organizacional.

Em geral, é possível afirmar que alguns dos objetivos dos modelos organizacionais são:

1) Entender a estrutura e a dinâmica da organização que hospedará o sistema;
2) Entender os problemas da organização e identificar as suas soluções;
3) Obter uma única e melhor compreensão da organização;
4) Contribuir para derivar requisitos do sistema para apoiar a organização.

Portanto, uma cuidadosa atenção aos aspectos organizacionais é crucial para o sucesso dos sistemas de informação [Yu, 1997c].

Considerando os problemas e os objetivos dos modelos organizacionais sob a perspectiva do rastreamento de requisitos surge a seguinte pergunta: Por que as diferentes pesquisas sobre o rastreamento de requisitos não tornaram explícita a necessidade de rastrear informações organizacionais para entender as justificativas,

mudanças de requisitos e os requisitos de software? Existem diversos trabalhos que recomendam a necessidade de considerar o aspecto organizacional dentro do rastreamento de requisitos, por exemplo, [Morris, 1994], [Jarke, 1993], [Jarke, 1996], [Ramesh, 1998b; Ramesh, 2001], [Pinto, 2002] e [Toranzo, 2000].

Alguns dos termos do nível organizacional são *recurso*, *processo*, *objetivo* organizacional e *regra*, os quais explicaremos a seguir.

Um *recurso* representa os elementos usados dentro de um negócio, tais como pessoas, materiais, informações e produtos. Por exemplo, o desenvolvimento de um sistema depende dos recursos humanos de uma empressa prestadora de serviços.

Um *processo* representa um conjunto de atividades realizadas dentro de um negócio. Alguns exemplos são os processos dos diferentes departamentos de uma organização para alcançar os objetivos organizacionais.

O *objetivo organizacional* representa o resultado esperado da organização sobre um sistema. Por exemplo, um objetivo organizacional sobre um sistema de biblioteca é que esse sistema inclua normas internacionais para a catalogação de livros.

Uma *regra* é uma declaração que define ou restringe alguns aspectos do negócio, e representa conhecimento do negócio. As regras podem indicar como os processos do negócio devem funcionar e/ou como os recursos podem ser estruturados e relacionados com outros processos. Uma regra sobre um processo de empréstimo eletrônico pode estabelecer que o limite do crédito do cliente é de 60% sobre o salário liquido cadastrado.

O objetivo da introdução do nível de informação organizacional no rastreamento de requisitos é tornar explícito o fato de que podem existir alguns conceitos organizacionais importantes (regras, recurso, processo, etc) que tem fortes implicações sobre os requisitos de software. Por exemplo, a regra que fixa o limite de crédito dos clientes do sistema de empréstimo afetará alguns requisitos desse sistema que determinam o crédito dos clientes. Uma outra regra sobre o sistema de empréstimo estabelece que a mudança de limite de crédito do cliente será depois da aprovação da comissão de crédito da agência da financeira.

O nível de informação organizacional representa muitos dos conceitos que originam o desenvolvimento e crescimento de uma organização. Conseqüentemente, a compra e uso dos sistemas para uma organização têm suas origens aqui porque qualquer sistema utilizado na organização deve agregar valor a seus serviços oferecidos.

A seguir explicaremos o nível de informação gerencial.

3.2.3 Nível de Informação Gerencial

Desenvolver software de qualidade assegurada, com elevada produtividade, dentro do prazo estabelecido e sem necessitar de mais recursos do que os alocados, tem sido o grande desafio da Engenharia de Software [Fiorini, 1998].

A gerência de projeto engloba as tarefas subordinadas a uma ou mais pessoas com os objetivos de planejar e controlar as atividades de outras pessoas para alcançar as metas de um projeto que não poderiam ser conseguidas facilmente por pessoas trabalhando de forma isolada [Thayer, 1997b]. As funções de gerência de projeto podem ser classificadas como planejamento, organização, seleção de pessoal, direção, supervisão e acompanhamento do projeto de software.

Durante o planejamento, são organizadas as tarefas para o desenvolvimento dos artefatos de software, as estimativas do tamanho dos artefatos, estimativa dos recursos necessários para produção, elaboração do cronograma e a identificação e avaliação dos riscos inerentes ao projeto [Fiorini, 1998]. Em geral, os gerentes de projetos estão preocupados com prazos, custos e elaboração dos relatórios de acompanhamentos dos projetos.

O nosso trabalho está interessado na atividade de planejamento da gerência de projeto, mais especificamente, no relacionamento entre as tarefas e os requisitos do sistema para permitir um melhor acompanhamento e controle dos requisitos do sistema. Existem alguns argumentos que fundamentam o relacionamento entre tarefa e requisito, e os outros níveis de informação. Eles são:

1. O CMM (*Capability Maturity Model* [Paulk, 1993]) é composto de cinco níveis de maturidade: inicial, repetitivo, definido, gerenciado e otimizado. Com exceção do nível inicial, cada nível é composto de Áreas-Chave de Processo (ACPs). O CMM não fornece diretrizes ou informações de como as áreas-chave dos níveis de maturidade poderiam relacionar-se melhor visando processos mais integrados. O nosso objetivo é que os relacionamentos entre as tarefas (planejamento e acompanhamento de tarefas) e os requisitos (gerência de requisitos) contribuam à repetição do sucesso no desenvolvimento de um software. Através das ligações (instâncias dos relacionamentos) das tarefas com os requisitos, podemos identificar efetivamente os requisitos trabalhados nas tarefas. A experiência com o rastreamento de requisitos nos ensinou que por mais que o nome de uma tarefa seja auto-explicativa, não significa que todos os requisitos relacionados com o nome da tarefa foram tratados; também, nomes genéricos podem englobar requisitos que envolvem diferentes informações, por exemplo, uma tarefa nomeada “relatórios do sistema” pode estar relacionada com requisitos que manipulam diferentes informações.

2. Todas as pesquisas sobre o rastreamento de requisitos concentram-se na produção e no refinamento de requisitos e no desenvolvimento de software, sem preocupar-se como o rastreamento poderia ajudar as outras atividades do desenvolvimento de um sistema, por exemplo, a gerência de projeto.
3. A gerência de projeto de software deveria estar mais relacionada com os requisitos do software [Wiegers, 1999]. Os planos de projetos deveriam estar fundamentados na satisfação dos requisitos dos sistemas e nas mudanças dos requisitos que afetarão esses planos.
4. Para a gerência de projeto, o rastreamento de requisitos é essencial porque fornece um meio para demonstrar ao cliente que todos os requisitos acordados foram satisfeitos e que o trabalho foi concluído [Ramesh, 1995a; Ramesh, 1995b].
5. O rastreamento poderia contribuir para que alguns elementos dos níveis de informação externo e organizacional possam relacionar-se com as tarefas do nível de informação gerencial. Por exemplo, um relacionamento entre as tarefas e os objetivos organizacionais de um sistema podem contribuir na identificação das tarefas que ajudam a alcançar esses objetivos no decorrer de um projeto.

Até agora abordamos os níveis externo, organizacional e gerencial. O quarto nível de informação a considerar no rastreamento é chamado de desenvolvimento, o qual está relacionado com os artefatos do desenvolvimento de software, tais como, documentos de requisitos, diagramas, e programas. A subseção seguinte apresenta e explica com mais detalhe o nível de desenvolvimento.

3.2.4 Nível de Informação de Desenvolvimento

O nível de informação de desenvolvimento representa os elementos/artefatos do processo de desenvolvimento de software. Alguns dos elementos produzidos neste nível são: documento de requisitos, diagrama e programa.

Nesse nível estão as atividades de pré e pós-rastreamento de requisitos porque aqui estão geralmente os elementos rastreados (documento de requisitos, diagrama e programa) pela maioria das pesquisas de rastreamento, por exemplo, [Spence, 2000], [Leffingwell, 2000], [Ramesh, 1993] e [Wiegers, 1999].

Nas subseções seguintes apresentaremos os benefícios dos níveis de informações propostos.

3.2.5 Benefícios dos Níveis de Informação

Os quatro níveis de informações fazem parte da contribuição deste trabalho para o rastreamento. Alguns dos benefícios dos níveis de informação são:

1. Considerar de forma explícita no rastreamento de requisitos os aspectos externo, organizacional, gerencial e do desenvolvimento. A introdução dos níveis de informação apresenta uma outra forma de perceber, identificar, estruturar e formular perguntas para a identificação das informações a serem rastreadas. Muitas ferramentas, por exemplo, RequisitePro e RTM, fornecem facilidades aos usuários para definir essas informações, deixando ao usuário a responsabilidade de identificá-las e organizá-las. Portanto, o fato de se ter uma ferramenta para rastrear requisitos não significa que se tenha um processo para o mesmo propósito;
2. Considerar os níveis de informação como um ponto de partida e/ou complementar para identificar, analisar e validar as informações que farão parte de um modelo de rastreamento de software. Todo projeto que inclua o rastreamento de requisitos deveria questionar-se acerca da necessidade de rastrear informações dos diferentes níveis porque pode existir alguma informação que possa mudar (uma lei ou uma norma) e afetar diferentes partes de um sistema;
3. Fornecer ao usuário um meio de abstrair-se das informações a serem rastreadas. Os níveis podem ser aplicados para entender as perguntas formuladas por Pohl e outros pesquisadores para identificar tais informações. Os níveis de informação são abstratos e contribuem na identificação das informações que farão parte de um modelo de rastreamento;
4. Melhorar a prática e entendimento do rastreamento de requisitos para o desenvolvimento e manutenção de software. A proposta dos níveis de informação é uma forma de melhorar a prática do rastreamento porque os usuários precisam adaptar esses níveis para a realidade de cada organização e projeto. Sabemos que os níveis por si mesmos são insuficientes para rastrear, então propomos um meta-modelo de rastreamento que as organizações podem adaptar em seus projetos. O meta-modelo será apresentado na Seção 3.3. Fundamentado nos níveis de informação e seguindo o meta-modelo, sugerimos a inclusão de um modelo intermediário de rastreamento para que os usuários identifiquem e adaptem as informações e relacionamentos propostos no modelo;
5. Fornecer uma classificação que seja independente do pré e pós-rastreamento de requisitos. Se os usuários optam pelo pré-rastreamento de requisitos, as informações recompiladas podem ser compreendidas e classificadas em função dos níveis de informação. Por exemplo, se o sistema a ser desenvolvido deve interagir com outros sistemas existentes, o pré-rastreamento deverá incluir informações dos outros

sistemas. Portanto, o pré-rastreamento deverá lidar com informações (requisitos, decisões de projeto fixadas) que possam ser classificadas em algum dos níveis de informação;

6. Fornecer um ponto de partida alternativo para as técnicas de rastreamento. As pesquisas abordadas no capítulo anterior, em geral, não fornecem um processo para construir um modelo de rastreamento. Também, a elicitação das informações a serem rastreadas são realizadas através de algumas perguntas que, do ponto de vista dos praticantes, podem ser um pouco confusas porque não é claro o tipo de informação que se pretende identificar. Essa confusão pode ser eliminada ou reduzida se as perguntas formuladas fossem focalizadas na identificação e unificação das informações dos níveis de informação. Dessa forma, a identificação e a análise das informações rastreadas podem contribuir com a inclusão de novas informações no modelo de rastreamento. A pessoa responsável pelo rastreamento das informações deverá identificar, com ajuda do analista, tais informações para cada nível de informação.

Com os benefícios dos níveis de informação concluímos a apresentação dos mesmos. Na seção seguinte apresentaremos um meta-modelo para o rastreamento de requisitos.

3.3 Um Meta-Modelo de Rastreamento

Esta seção apresenta e explica o nosso meta-modelo para o rastreamento de requisitos. De forma similar ao meta-modelo da UML, o nosso também está fundamentado no MOF (*Meta Object Facility* [OMG, 2000]), que é usado para descrever e definir meta-dado (dado que define dado). A especificação do MOF define um conjunto de interfaces para CORBA IDL (*Interface definition Language*) que podem ser usadas para definir e manipular um conjunto de meta-modelos interoperavéis. Os modelos interoperavéis incluem o meta-modelo da UML, o meta-metamodelo do MOF, assim como também futuras tecnologias adotadas pela OMG (Objetc Management Group) e que serão especificadas usando os meta-modelos [OMG, 2000].

Particularmente, o objetivo do nosso meta-modelo é ajudar na construção de um diagrama de classe, como da UML, para expressar um modelo de rastreamento de requisitos de um software. Para isso, usamos os relacionamentos de agregação e generalização da UML e outros tipos propostos neste trabalho. É importante salientar que o MOF fornece a oportunidade de construir meta-modelos a partir do zero. Desta forma, a construção de um modelo de rastreamento está restrita a um número definidos de relacionamentos. Adatar a UML para o rastreamento de requisitos implica colocar

uma série de restrições sobre os tipos de associações que podem ser usadas no diagrama de classe que expresse um modelo de rastreamento.

O MOF usa uma arquitetura de meta-dado em camada que está fundamentada na arquitetura de meta-modelagem das quatro camadas: meta-meta-modelo, meta-modelo, modelo e informação. A Figura 14 apresenta nosso meta-modelo na arquitetura de meta-dado do MOF.

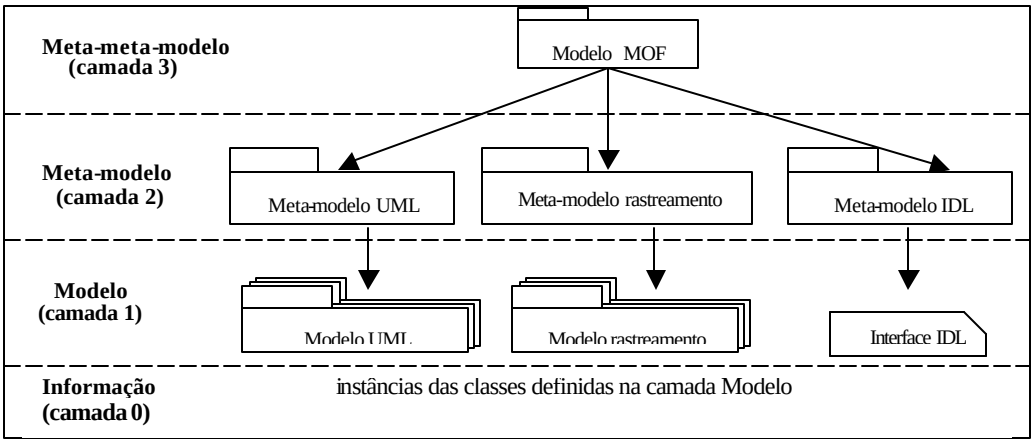


Figura 14: Arquitetura de meta-dado do Meta Object Facility (MOF)

A camada meta-meta-modelo descreve estruturalmente e semânticamente o meta-meta-dado. Em outras palavras, é uma linguagem para a definição de diferentes meta-dados.

A camada meta-modelo descreve estruturalmente e semânticamente o meta-dado. Um meta-modelo é uma linguagem que permite definir diferentes tipos de dados. O meta-modelo da UML define o meta-dado e regras que devem ser respeitadas nas diferentes construções dos diagramas (classe, colaboração, atividades, etc). A partir dos recursos do MOF, elaboramos o meta-modelo de rastreamento para que os usuários construam modelos de rastreamento mais ricos em termos de relacionamentos. Na camada meta-modelo está o meta-modelo IDL (Interface definition Language), que é uma linguagem padronizada e independente de arquitetura, que permite especificar as interfaces dos objetos distribuídos de uma forma que todos possam requisitar seus serviços. Maiores informações da IDL são encontradas em [OMG, 2000].

A camada modelo que descreve as classes e relacionamentos. Nesta camada são instanciados os meta-modelos para definir os elementos (por exemplo, classe e tipo de relacionamento) contidos nos diagramas.

A camada informação descreve as instâncias. Essa informação é tipicamente chamada de "dado". A camada informação representa as instâncias das classes definidas na camada modelo. Se na camada modelo foi definida alguma classe, por exemplo, *Cliente*, então a camada informação conterá todas as instâncias dessa classe.

Neste documento usaremos o termo ligações para referirmos às instâncias das associações modeladas na camada modelo do MOF.

A Figura 15 apresenta nosso meta-modelo para o rastreamento de requisitos. Nas classes foram colocados somente os atributos relevantes para evitar sugerir atributos inadequados para o rastreamento de software.

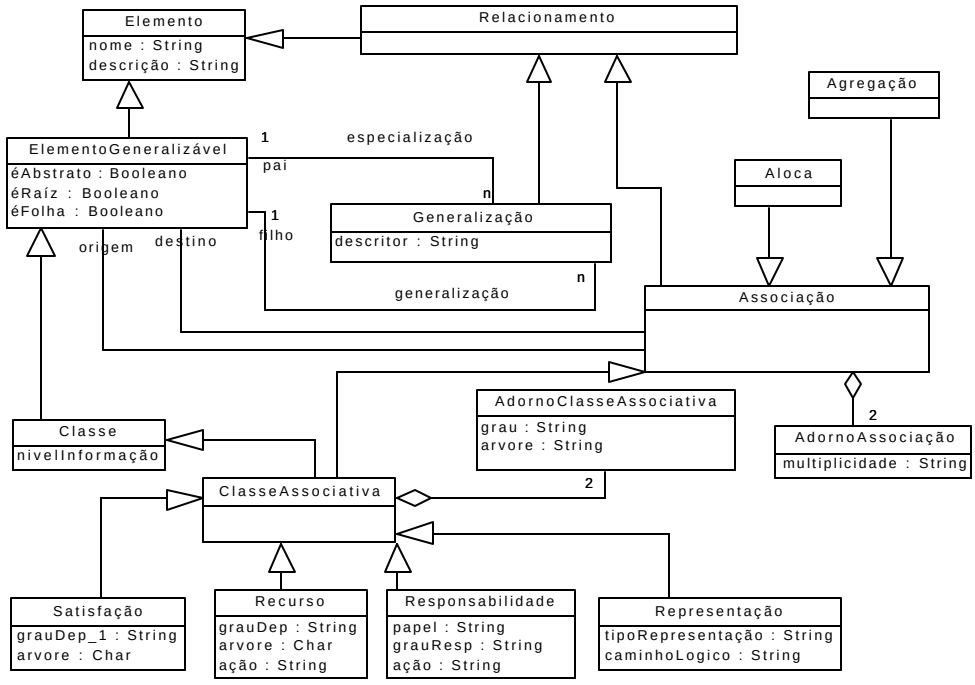


Figura 15: Meta-modelo para o rastreamento

Na Figura 15 a meta-classe base do meta-modelo é *Elemento*, formada por dois meta-atributos (nome e descrição), ambos do tipo *String*. As subclasses de *Elemento* são as meta-classes *ElementoGeneralizável* e *Relacionamento*.

O *ElementoGeneralizável* classifica e caracteriza as instâncias da meta-classe *Classe* em subclasse, se a especialização for usada, ou como superclasse, se generalização for usada. A meta-classe *ElementoGeneralizável* possui três propriedades: éAbstrato, éRaiz e éFolha.

A propriedade éAbstrato especifica se *ElementoGeneralizável* espera ter instâncias. A propriedade éRaiz especifica se *ElementoGeneralizável* pode ter superclasses. A propriedade éFolha especifica se *ElementoGeneralizável* pode ser superclasse de outro elemento *ElementoGeneralizável*.

Uma subclasse de *ElementoGeneralizável* é a meta-classe *Classe*, cujas instâncias são classes (na camada modelo), que representam um conjunto de objetos que compartilham as mesmas propriedades e comportamento.

Duas instâncias de *ElementoGeneralizável* se relacionam entre si através da *Generalização* e *Associação*, ambas são subclasses de *Relacionamento* .

A *Generalização* é um relacionamento entre uma classe específica e uma classe genérica. A classe mais especifíca herda as propriedades da classe genérica e contem propriedades adicionais.

A *Associação* é um relacionamento binário e bidirecional entre dois *ElementoGeneralizável* porque o rastreamento é realizado sempre entre um par de elementos. No meta-modelo, a associação entre dois *ElementoGeneralizável* é especificada por dois relacionamentos entre as meta-classes *ElementoGeneralizável* e *Associação*. Nesses dois relacionamentos, os papéis, origem e destino, identificam e diferenciam as classes na associação em classe origem e classe destino. A Figura 16 apresenta uma visão geral da associação.

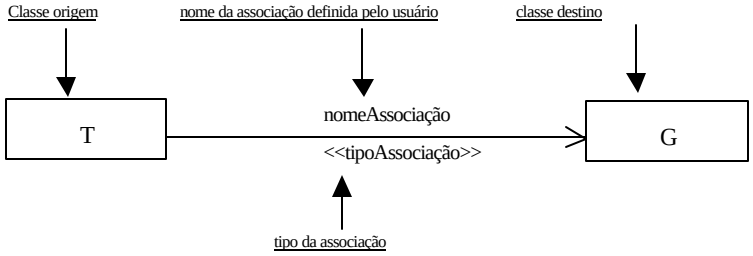


Figura 16: Alguns elementos básicos de uma associação

Na Figura 16 as palavras sublinhadas esclarecem o significado da notação e das expressões. A expressão *nomeAssociação* representa o nome da associação atribuída pelo usuário. A expressão *<<tipoAssociação>>* representa uma abreviação do nome do tipo da associação: satisfação (sat), recurso (rec), responsabilidade (resp), representação (rep) e aloca (alo). Independente do significado das classes *T* e *G*, a classe *T* é chamada classe origem da associação e a classe *G*, de classe destino. Uma associação é representada por uma seta desenhada da classe origem para a classe destino para facilitar sua interpretação na modelagem. É importante salientar que os termos de classe origem e classe destino serão usados para explicar todas os meta-relacionamentos do meta-modelo.

A meta-classe *ClasseAssociativa* é composta da meta-classe *AdornoClasseAssociativa* que representa os adornos aplicados sobre as instâncias das subclasses da meta-classe *ClasseAssociativa*.

O meta-modelo possui as associações do tipo *agregação* e *aloca*. As classes associativas são: *satisfação, recurso, responsabilidade* e *representação* porque elas podem possuir atributos próprios.

A Tabela 2 apresenta um dicionário dos tipos de relacionamento propostos e usados nesta tese para elaborar um modelo de rastreamento.

Tipo de associação	Descrição
Generalização	Um relacionamento entre um elemento geral e um elemento mais específico. O elemento mais específico é complemente consistente com o elemento geral e contem informações adicionais.
Aloca	Uma associação uni-direcional de uma classe, que representa requisitos funcionais, para uma outra classe que representa um subsistema.
Agregação	Uma associação que modela um relacionamento todo-parte entre uma classe, denominado *todo*, e uma ou mais classes que são denominadas *partes*.
Satisfação	Uma associação entre duas classes denominadas classe origem e classe destino. A associação é desenhada da classe origem para a classe destino. A associação satisfação é uma relação de dependência que especifica que a classe origem tem uma dependência de realização com a classe destino. O termo realização é usado para expressar que algo deverá ser realizado ou feito sobre as instâncias da classe destino para satisfazer as instâncias da classe origem com as quais estão conectadas.
Recurso	Uma associação que especifica que uma classe possui uma dependência de informação ou física com uma outra classe.
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 2: Dicionário dos tipos de relacionamentos para rastrear requisitos

Nas subseções seguintes apresentaremos cada um dos relacionamentos.

3.3.1 Adornos da Associação

O meta-modelo de rastreamento expressa que uma associação possui dois adornos (classe *AdornoAssociação*) cujo objetivo é especificar a multiplicidade.

Conforme a arquitetura do meta-dado do MOF (Figura 14), a camada modelo (camada 1) contém instâncias do meta-modelo de rastreamento. Portanto, todos os elementos da Figura 17 são instâncias das meta-classes. Na Figura 17, as palavras sublinhadas expressam o significado das notações.

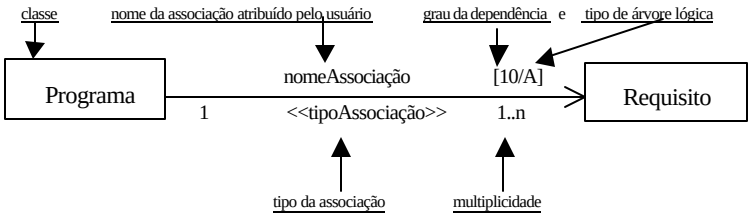


Figura 17: Um fragmento genérico de um modelo de rastreamento

A seguir explicaremos a multiplicidade. A multiplicidade especifica o número de instâncias de uma classe que podem estar ligadas com uma simples instância de uma outra classe. A notação e significado das multiplicidades usadas neste trabalho são: "*" (muitos), 1 (uma instância somente), 0..n (zero ou muitas instâncias), 1..n (uma ou muitas instâncias), 0..1 (zero ou uma instância), e literal..literal (faixa de instâncias). Por exemplo, a Figura 17 expressa que uma instância da classe *Programa* deve estar ligada com uma ou muitas instâncias da classe *Requisito*, e que uma instância da classe *Requisito* deve estar ligada com uma instância da classe *Programa*. Uma informação complementar à multiplicidade é o conceito chamado grau de dependência. Esse conceito expressa a dependência entre as instâncias das classes associadas.

O grau de dependência pode ser expresso em forma quantitativa ou qualitativa (A=alta, M=Médio e B=Baixo). Por exemplo, se consideramos uma escala de 1 até 10 para expressar o grau de dependência da classe *Programa* para a classe *Requisito* (Figura 17), então o valor 10 na expressão "[10/A]" expressa que uma instância da classe *Programa* tem um alto grau de dependência com todas as instâncias da classe *Requisito*, com as quais está ligada. Por exemplo, alguns requisitos de um sistema podem ter um alto grau de dependência com algumas leis que, se mudarem, podem ter um forte impacto sobre os requisitos e o sistema que os implementa.

Se na Figura 17 o grau de dependência não for especificado ([/A]) da classe *Programa* para *Requisito*, então é expresso que cada uma das instâncias da classe *Programa* pode ter diferentes graus de dependência com todas as instâncias da classe *Requisito*, com as quais está ligada. Nas expressões "[10/A]" ou "[/A]", a letra "A" representa um tipo de árvore lógico que será explicado a seguir.

Na Figura 17 foi expresso que uma instância da classe *Programa* tem um alto grau de dependência com as instâncias da classe *Requisito*, com as quais está ligada.

Essas ligações formam uma árvore cuja raiz é uma instância da classe *Programa* e as folhas são as instâncias da classe *Requisito*. Porém, não é possível classificar e identificar nessas ligações as instâncias da classe *Requisito*, que são obrigatórias ou opcionais para uma instância da classe *Programa* ser realizada. Uma solução alternativa para esse problema é chamada de árvore lógica.

Uma árvore lógica é uma estrutura hierárquica que contribui para identificar e classificar as folhas de uma árvore usando os operadores lógicos *AND* e *OR* para facilitar e incrementar o significado da árvore. As notações dos operadores lógicos *AND* e *OR* na árvore são as letras "A" e "O", respectivamente.

Na Figura 17, a letra "A" (operador lógico *AND*) na expressão "[10/A]" especifica que é obrigatório que todas as instâncias de *Requisito*, ligadas com uma instância de *Programa*, devam ser realizadas. A letra "O" (operador lógico *OR*) na expressão "[10/O]" especifica que pelo menos uma das instâncias de *Requisito*, ligadas com uma instância de *Programa*, deve ser realizada.

Se nenhum operador lógico é especificado ("[10/]"), então é expresso que todas as instâncias de *Requisito*, ligadas com uma instância de *Programa*, podem ser uma combinação de "A" e "O". A Figura 18 apresenta uma árvore lógica de uma associação com operador lógico *AND*. A Figura 19 apresenta uma árvore lógica de uma associação com os operadors lógicos *AND* e *OR*. A aplicação dos operadores lógica serão ilustrada no Seção 4.6.

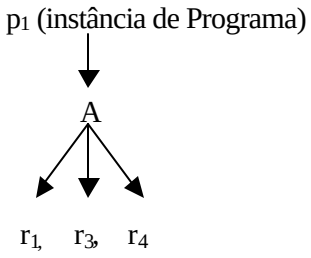


Figura 18: Árvore do tipo AND

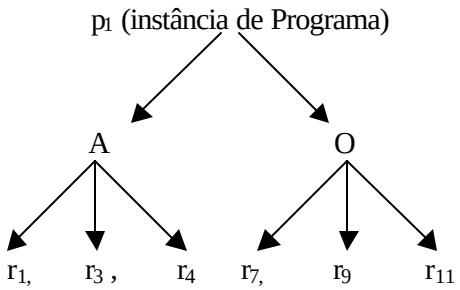


Figura 19: Árvore do tipo AND e OR

Cada uma das ligações entre as instâncias das classes *Programa* e *Requisito* são representadas nas matrizes de rastreamento por uma tupla. A estrutura da tupla varia entre as associações. A explicação da estrutura das tuplas de cada associação será explicada nas próximas subseções que apresentam os diferentes tipos de meta-associações do meta-modelo.

3.3.2 A Associação Satisfação

O primeiro tipo de meta-associação proposto neste trabalho chama-se *Satisfação*. Uma instância da meta-associação *Satisfação* é uma associação de dependência que especifica que a classe origem (com papel origem) tem uma dependência de satisfação com a classe destino (com o papel de destino). O termo satisfação é usado para expressar que algo deverá ser realizado sobre as instâncias da classe destino para satisfazer as instâncias da classe origem com as quais estão conectadas. Por exemplo, um objetivo organizacional pode expressar que "o sistema de biblioteca deverá incluir padrões internacionais de catalogação de livros", esse objetivo é satisfeito pelos vários requisitos implementados no sistema. Se posteriormente esse objetivo for excluído, os requisitos que o satisfazem deverão ser excluídos do sistema.

A Figura 20 apresenta um fragmento de um modelo de rastreamento de um sistema de agendamento de reuniões que possui três instâncias da meta-associação *Satisfação* (*planejado_em*, *considerado_em* e *satisfeito_em*), cuja representação gráfica é o rótulo "<<sat>>". Em lugar de desenhar diferentes setas para cada um dos tipos de relacionamento, decidimos rotular a maioria dos relacionamento. Um exemplo de rótulo é o usado para identificar os relacionamento do tipo satisfação.

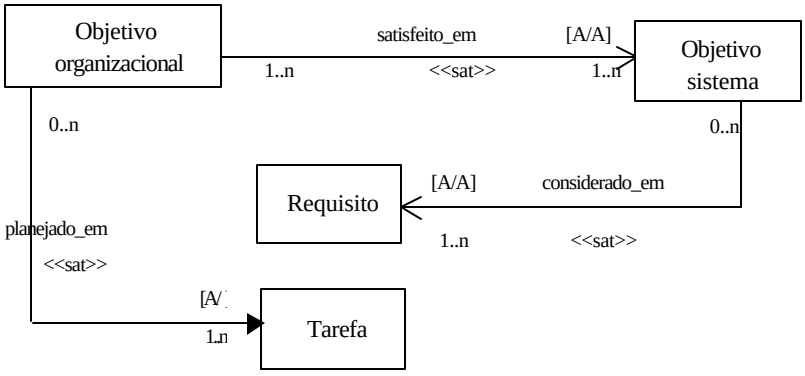


Figura 20: Representação da associação satisfação

Na Figura 20, a associação *satisfeito_em* expressa que a conformidade (ou satisfação) de uma instância da classe *Objetivo organizacional* (classe origem) tem um alto grau dependência com uma ou muitas instâncias da classe *Objetivo sistema* (classe destino) com as quais está conectada. Também é expresso que a satisfação de uma instância da classe *Objetivo organizacional* requer que sejam satisfeitas todas as instâncias da classe *Objetivo sistema* com as quais está ligada.

Quando trabalhamos com uma associação do tipo *Satisfação*, por exemplo, *satisfeito_em*, recomendamos omitir o grau de dependência e a árvore lógica em relação à classe origem (*Objetivo organizacional*) porque essas informações são irrelevantes do ponto de vista das instâncias da classe destino (*Objetivo sistema*) e porque sobrecarregam de forma desnecessária o trabalho de rastreamento.

Na Figura 20, a associação *considerando_em* expressa que a conformidade (ou satisfação) de uma instância da classe *Objetivo sistema* tem um alto grau de dependência com uma ou muitas instâncias da classe *Requisito* com as quais está ligada, e que devem ser satisfeitas todas as instâncias da classe *Requisito* ligadas com as instâncias da classe *Objetivo sistema*.

Um terceiro exemplo da instância da meta-associação *Satisfação* é a associação *planejado_em* que expressa que a conformidade (ou satisfação) de uma instância da classe *Objetivo organizacional* tem um alto grau de dependência com uma ou muitas instâncias da classe *Tarefa* com as quais está ligada. Também é expresso que a satisfação de uma instância da classe *Objetivo organizacional* é obrigatória para as instâncias da classe *Tarefa* com as quais está conectada.

A Figura 21 apresenta a representação matricial da associação *satisfeito_em* sobre um sistema de agendamento de reuniões. As convenções adotadas a respeito da representação matricial de uma associação são:

1. O nome e tipo da associação representada na matriz são colocados na parte superior esquerda da matriz. O tipo da associação é expresso por um rótulo, por exemplo, o rotulo "<<*sat*>>" representa uma associação do tipo satisfação;
2. Baixo o nome da associação colocaremos o símbolo seta ("→" ou "←") que indicará as instâncias da classe destino na associação. Se for usado o símbolo "→", então as instâncias declaradas na primeira linha da matriz são instâncias da classe destino. Caso contrário ("←"), as instâncias declaradas na primeira coluna da matriz são instâncias da classe destino;
3. Para a primeira coluna e primeira linha das matrizes de rastreamento recomendamos expressões para identificar as instâncias de uma classe. O usuário pode usar outras expressões que considerem mais apropriadas para seu projeto. Algumas expressões recomendadas são:

[OBO] = Objetivos organizacionais	[PRO] = Programa do sistema
[OBS] = Objetivos do sistema	[DGM] = Diagramas
[REQ] = Requisitos do sistema	[TAR] = Tarefa da gerência
[RES] = Restrições do sistema	[FUN] = Funcionário
[LEI] = Lei	[SUB] = Subsistema do sistema

A representação dos relacionamentos entre as instâncias das classes associadas podem ser representadas através de listas, matrizes ou árvores. Porém, neste trabalho decidimos usar listas e matrizes para representar os relacionamentos porque são mais simples e por serem mais usadas no rastreamento, especialmente as matrizes. Geralmente usaremos as matrizes quando existam poucas informações relacionadas entre si. Já as listas serão usadas geralmente para representar relacionamentos que não possuem atributos, por exemplo, os relacionamentos entre requisito e subsistema, e requisito e programa. A Figura 21 apresenta um exemplo da representação matricial do relacionamento *Satisfeito* da Figura 20. É importante dizer que as expressões usadas nas matrizes para identificar as instâncias das classes estão acompanhadas de textos para facilitar a compreensão das mesmas. Porém, também é possível colocar somente as expressões quando existam várias informações associadas.

Satisfeito_em: <<sat>> ←	[OBO-1] Agendar reunião	[OBO-5] Ter um banco de informações das reuniões
[OBS-3] Requer datas de participação dos funcionários	<A; A>	
[OBS-4] Agendar uma reunião para todos os funcionários poder participar em	<A; A>	
[OBS–5] Informar a data da reunião	<A; A>	
[OBS-6] Registrar a confirmação da participação dos funcionários	<A; A>	
[OBS-7] Enviar pauta da reunião	< A; A>	<A; A>
[OBS-8] Registrar as atividades e resultados das reuniões		<A; A>

Figura 21: Representação matricial da associação satisfação

A representação matricial da associação *Satisfação* é uma tupla composta de dois componentes (<*GrauDep1*; *Árvore*>). O primeiro componente, *GrauDep1*, especifica o grau da dependência de uma instância com respeito a outra instância. Por exemplo, na segunda coluna da matriz, o primeiro componente das tuplas expressa que a instância *[OBO-1]* tem um alto grau de dependência com as instâncias *[OBS-3]*, *[OBS-4]*, *[OBS-5]*, *[OBS-6]* e *[OBS-7]*.

O segundo componente, *Árvore*, identifica o tipo da árvore lógica. Por exemplo, na segunda coluna da matriz da Figura 21, a letra "A" no segundo componente das tuplas expressa que é obrigatório que todas as instâncias *[OBS-3]*, *[OBS-4]*, *[OBS-5]*, *[OBS-6]* e *[OBS-7]* sejam satisfeitas para poder satisfazer a instância *[OBO-1]*.

Se tivéssemos usado uma abordagem tradicional para preencher a matriz de rastreamento da Figura 21, essa matriz conteria somente o símbolo "X" ou outro símbolo qualquer. Consequentemente, teriamos uma matriz de rastreamento que forneceria informações incompletas e imprecisas porque não saberíamos o grau de dependência e de satisfação existentes entre as instâncias associadas. O único que saberiamos que ambas estão relacionadas. Portanto, podemos afirmar que a associação *Satisfação* e sua representação matricial contribuem para melhorar o trabalho do rastreamento de requisitos. A título de exemplo, veja a Figura 21.

A seguir apresentaremos uma outra associação chamada *recurso*.

3.3.3 A Associação Recurso

O segundo tipo de meta-associação chama-se *Recurso*. Uma meta-associação *Recurso* é uma associação de dependência que especifica que uma instância da meta-classe origem tem uma dependência de recurso (informação ou um elemento físico) em relação à instância da meta-classe destino. Entende-se por recurso um elemento que representa uma informação ou um elemento físico. Por exemplo, o plano de teste de um sistema usa como recurso os requisitos do sistema. Um segundo exemplo é a gerência de projeto que no planejamento das tarefas usa como recurso os requisitos do sistema. Um terceiro exemplo são os requisitos de um sistema de folha de pagamento que dependem das leis (recurso de informação) que regulam a retenção de imposto de renda. Isto é, se as leis mudam, os requisitos do sistema deverão mudar para satisfazer as leis.

A Figura 22 apresenta algumas instâncias da meta-associação *Recurso*. Essa figura apresenta um fragmento de um modelo de rastreamento de um sistema de conta corrente bancário cujos requisitos podem depender dos artigos da CPMF (classe *Artigo_Lei*).

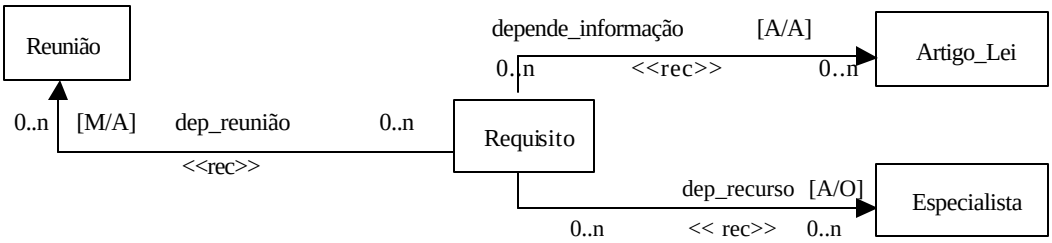


Figura 22: Representação da associação recurso

Na Figura 22, o rótulo "<<rec>>" é usado para expressar que as associações *dep_reunião*, *depende_informação* e *dep_recurso* são do tipo recurso. Se considerarmos a associação *depende_informação* com seus adornos, então uma instância da classe *Requisito* tem um alto grau de dependência de recurso com zero ou muitas instâncias da

classe *Artigo_Lei* e que uma instância da classe *Artigo_Lei* é fonte de recurso para zero ou muitas instâncias da classe *Requisito*. Nessa mesma associação, a árvore lógica (a letra "A") expressa que é obrigatório considerar e revisar todas as instâncias da classe *Artigo_Lei* para introduzir e compreender melhor alguma alteração sobre os requisitos.

Na associação *dep_recurso* é expresso que uma instância da classe *Requisito* tem uma dependência alta de recurso com zero ou muitas instâncias da classe *Especialista*. Também, a árvore lógica, expressa que o recurso informacional pode ser obtido de qualquer especialista.

Na associação *dep_reunião* foi especificado que uma instância da classe *Requisito* tem grau médio de dependência de recurso com uma ou muitas instâncias da classe *Reunião* com as quais está ligada. Isto significa que algumas mudanças sobre as informações registradas nas instâncias de *Reunião* têm um impacto médio sobre os requisitos.

A Figura 23 é uma matriz que sugere uma representação matricial da associação *depende_informação*, modelada na Figura 22. A primeira coluna da matriz apresenta alguns dos requisitos de um hipotético sistema de conta corrente, enquanto que a primeira linha apresenta os artigos de lei da CPMF que são recursos de informação na criação dos requisitos.

A representação matricial das instâncias da associação *Recurso* é uma tupla composta de três componentes (<*GrauDep*; *Árvore*; *ação*>).

O primeiro componente, *GrauDep*, expressa o grau de dependência em forma qualitativa (A=alto, M=Médio e B=Baixo) ou quantitativa (valores entre 1 e 10).

O segundo componente, *Árvore*, expressa o tipo da árvore.

O terceiro componente, *ação*, expressa a forma como a instância da classe origem usará (criar, modificar, etc) as instâncias da classe destino. Por exemplo, o componente *ação* do relacionamento recurso, da classe *Requisito* para a classe Objetivo, expressa que uma instância da classe *Requisito* usa as instâncias da classe *Objetivo* como fonte para sua criação/modificação. Para o uso desse componente é necessário que o projeto codifique a ação para facilitar seu uso e compreensão na matriz de rastreamento. Por exemplo, a expressão "Cri" (significa criar) é usado na matriz da Figura 23 para expressar que os requisitos do sistema foram criados considerando como recursos de informação os artigos das leis. Também, poderíamos usar a expressão "Ref" (significa refinar) para especificar que os recursos foram usados para refinar os requisitos. Se a expressão "Der" (derivar) for usada, então é especificado que o recurso foi usado para derivar os requisitos. Se a expressão "Jus" (justificar) for usada, então é especificado que o recurso foi usado para justificar ou associar um raciocínio aos requisitos. Um dos grandes benefícios da componente *ação* é reduzir a proposta de outras associações que teriam a mesma funcionalidade semelhante à associação recurso. As abreviações

sugeridas neste trabalho podem ser adotadas ou modificadas conforme às necessidades de cada projeto de software.

Depende_informação: <<rec>> →	[LEI-3.1] **Art. 3º** A contribuição não incide: I – no lançamento nas contas da União, dos Estados, do Distrito Federal, dos Municípios.....	[LEI-3.5] **Art. 4º** São contribuintes: I - os titulares das contas referidas nos incisos I e II do art. 2º, ainda que mo.....	[LEI-3.4] **Art. 7º** A alíquota da contribuição é de vinte centésimos por cento	[LEI-3.6] **Art. 8º** A alíquota fica reduzida a zero: I – nos lançamentos a débito em contas de depósito de poupança, de depósito judicial e de depósito em consignação de pagamento....
[REQ-1] Registrar o valor CPMF a debitar	<A; A; Cri>	<A; A; Cri>	< A; A; Cri>	< A; A; Cri>
[REQ-2] Debitar o valor da CPMF das contas correntes (CC)...	< A; A; Cri>	< A; A; Cri>	< A; A; Cri>	<A; A; Cri>

Figura 23: Representação matricial da associação recurso

É importante reconhecer que a especificação da *ação* na matriz de rastreamento só será possível se os diferentes analistas e desenvolvedores registrarem ou colaborem para obter essa informação. Portanto, a informação refletida na componente *ação* deve ser obtida durante o processo de produção de requisitos e implementação do sistema.

A seguir apresentaremos o relacionamento *generalização* que permite estruturar um conjunto de classes.

3.3.4 O Relacionamento Generalização

A generalização fornece um mecanismo para estruturar classes. Sua definição é a mesma fornecida pela UML [Booch, 1999]. O maior benefício da *generalização* é fatorar as propriedades comuns de um conjunto de classe hierárquica de um modelo de rastreamento. O nosso trabalho não atribuiu uma representação matricial para a generalização porque ela é somente um mecanismo de estruturação de classes. A Figura 24 apresenta a notação da generalização.

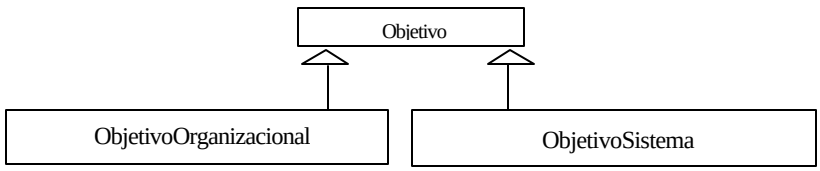


Figura 24: Representação do relacionamento generalização

A Figura 24 apresenta uma estrutura hierárquica na qual os atributos da classe *Objetivo* são herdados pelas subclasses. Uma forma de representar essa hierarquia em uma ferramenta de gerência de requisitos é definir uma classe *Objetivo* com uma propriedade adicional para diferenciar as instâncias dos diferentes objetivos (organizacional e sistema).

A seguir apresentaremos uma outra associação, chamada *agregação*, para capturar e expressar que os requisitos podem estar compostos de outros requisitos.

3.3.5 A Associação Agregação

Quando o projeto de um software é iniciado, freqüentemente os requisitos e objetivos são gerais. No decorrer das atividades da engenharia de requisitos esses requisitos são decompostos em outros requisitos mais específicos. Por exemplo, um dos primeiros requisitos de um sistema de recursos humanos pode ser: "o sistema deverá gerenciar as informações dos funcionários". Posteriormente esse requisito pode ser decomposto e refinado em outros requisitos, tais como: "o sistema deverá incluir funcionários com os seguintes campos de informação...", "o sistema deverá gerar um relatório dos funcionários com os seguintes campos...", e assim por diante.

A meta-associação *Agregação* especifica que uma meta-classe C_1 (chamada *todo*) pode ser composta de uma ou mais meta-classes C_2, $C_{3,...,}C_n$ (chamadas *parte*) ou estar relacionada com ela mesma. A associação de agregação acrescenta o fato de que a existência da instância C_1 com relação às instâncias C_2 $C_{3,...,}C_n$ são independentes. Como é possível apreciar, a definição de agregação é a mesma fornecida pela UML [Booch, 1999]. Por exemplo, se duas instâncias de uma classe estão relacionadas por agregação e uma delas é removida, não implica que a outra instância seja removida, e vice-versa. A Figura 25 ilustra alguns exemplos da meta-associação *Agregação*.

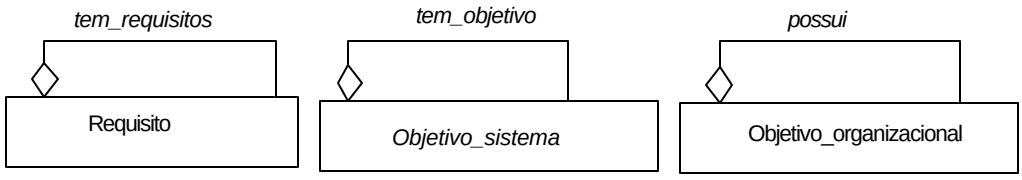


Figura 25: Representação da associação agregação

A Figura 25 especifica que as instâncias das classes *Objetivo_sistema*, *Objetivo_Organizacional* e *Requisito* estão relacionadas entre si. A Figura 26 recomenda uma representação para a agregação *tem_requisitos* da Figura 25.

tem_requisitos: agregação
+ [REQ-4] O sistema deverá gerenciar informações dos clientes
[REQ-4.1] O sistema deverá incluir clientes com os seguintes campos código, data e hora de inclusão, senha, supervisor(sim/não), nome, foto, endereço, complemento, bairro, cidade, estado, CEP, telefone, celular, E-mail, Sexo, data de nascimento, naturalidade, estado civil, grau de instrução, CPF, CGC, RG, órgão expedidor, referencias, outras locadoras, nome do pai, nome da mãe, local de trabalho, profissão, cargo, tempo de trabalho...
[REQ-4.2] O sistema deverá excluir clientes usando CPF e que não tenham pendências com a vídeo locadora.
[REQ-4.3] Consultar funcionário. O sistema deverá permitir pesquisar o funcionário por código, nome, ou parte do nome. Em seguida, o sistema apresentará todas as informações do funcionário. Nenhuma informação apresentada poderá ser modificada.
[REQ-6] O sistema deverá incluir produtos com nome, fornecedor,...

Figura 26: Representação matricial da associação agregação

Na Figura 26 sugerimos o uso do símbolo "+" para representar a agregação *tem_requisitos*. Por exemplo, o requisito "[REQ-4]" está composto dos requisitos *[REQ-4.1]*, *[REQ-4.2]* e *[REQ-4.3]*.

A seguir apresentaremos uma outra associação, chamada *aloca*, para registrar a alocação dos requisitos nos subsistemas de um software.

3.3.6 A Associação Aloca

Na literatura corrente sobre rastreamento ([Palmer, 1997], [Dorfman, 1997], [Thayer, 1997a] e [Ramesh, 2001]), muito autores usam a palavra "alocado" para expressar que os requisitos são atribuídos a um subsistema.

A associação *Aloca* significa que uma instância da classe de origem foi atribuída ou alocada a uma instância da classe destino, que representa um subsistema.

Na Figura 27 as classes formam parte de um fragmento de um modelo de rastreamento que expressa que os requisitos foram alocados em subsistemas. O tipo de uma associação *Aloca* é identificado pelo o rótulo "<<aloc>>".

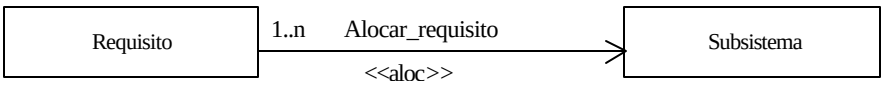


Figura 27: Representação da associação aloca

Na Figura 27 é expresso que a associação *Alocar_requisito* é do tipo aloca e que uma instância da classe *Requisito* está alocada em um subsistema e que um subsistema aloca uma ou muitas instâncias da classe *Requisito*.

A Figura 28 apresenta a representação matricial da associação *Aloca*.

Alocar_requisito:<<aloc>> →	[SUB-1] Subsistema cliente	[SUB-4] Subsistema de relatório	[SUB-6] Subsistema de controle de acesso
[REQ-1] O sistema deverá incluir clientes com os campos... [REQ-2] O sistema deverá emitir o relatório de clientes com os campos nome, depto, ...	→	→	
[REQ-4] O sistema deverá excluir cliente usando o CPF.	→		
[REQ-6] O sistema deverá emitir o relatório de produtos com os campos código, nome, estoque, venda, ...		→	
[REQ-11] O sistema deverá gerenciar todos seus usuários.			→
[REQ-12] O sistema deverá ter um controle de acesso (login e senha)			→

Figura 28: Representação matricial da associação aloca

Na Figura 28 é expresso que os requisitos [REQ-1] e [REQ-4] foram alocados no subsistema *Cliente* (*[SUB-1]*), enquanto que os requisitos [REQ-2] e [REQ-6] foram alocados no subsistema de *Relatório* ([SUB4]). Os requisitos [REQ-11] e [REQ-12] foram alocados no subsistema de *Controle de acesso* (*[SUB6]*). A única diferença da matriz da Figura 28 com relação às matrizes tradicionais é que na parte superior esquerda é indicado o nome do relacionamento e seu tipo.

A matriz da Figura 28 tem um grande benefício para os desenvolvedores que desejam reusar alguns desses subsistemas na construção de outros sistemas porque cada coluna da matriz torna possível identificar os requisitos alocados nos diferentes subsistemas. Uma pessoa que conheça a representação matricial do relacionamento aloca poderá interpretar e entender a matriz da Figura 28.

A seguir apresentaremos uma outra associação, chamada *responsabilidade*, para capturar a informação relacionada com a participação das pessoas sobre os artefatos de software.

3.3.7 A Associação Responsabilidade

As atividades de desenvolvimento de software são altamente comunicativas e sociais porque as pessoas trabalham e interagem exercendo diferentes papéis no processo de software, seja para idealizar/desenvolver diferentes elementos (por exemplo, requisitos) e artefatos de software.

A introdução de uma associação entre as pessoas e elementos/artefatos de software traz a noção de responsabilidade e a necessidade de capturar informações complementares à responsabilidade, visando obter e registrar informações detalhadas e reais da contribuição das diferentes pessoas em relação aos artefatos ou às atividades

realizadas no processo de software. Logo surge a pergunta: De que forma nosso meta-modelo pode contribuir para capturar a participação e responsabilidade das pessoas no processo de software?

Entre as atividades clássicas do modelo da gerência de projeto (planejamento, organização, seleção de pessoas, coordenação e controle), a atividade de organização cria posições e responsabilidades organizacionais de um projeto, isto é, identifica e captura uma visão estática da sociedade das pessoas participantes do projeto [Thayer, 1997]. Porém, a visão dinâmica dessa sociedade é igualmente importante. Entende-se por visão dinâmica a captura do papel da participação das pessoas (analista, programador, etc), o seu grau de responsabilidade e ação realizada sobre o artefato.

Consideremos o seguinte exemplo que ilustra a diferença entre as visões estática e dinâmica. A gerência de projeto pode afirmar que as pessoas alocadas para as diferentes tarefas foram responsáveis pela implementação de um projeto. Mas essa informação pode ser ainda mais detalhada se introduzirmos e especificarmos a responsabilidade dos participantes dos projetos em uma matriz de rastreamento. Por exemplo, poderíamos dizer que o funcionário Frederico foi altamente responsável pela implementação e teste do módulo de contas; mediamente responsável pela programação dos módulos de relatórios e controle de acesso; e medianamente (ou regularmente) responsável como escritor do manual de usuário do sistema de contabilidade.

A idéia de capturar e registrar o aspecto social está fundamentado no trabalho de pré-rastreamento [Gotel, 1996a]. Porém, a nossa pesquisa visa enriquecer a captura dos papéis sociais e estendê-la para pós-rastreamento de requisitos. Por isso propusemos no meta-modelo uma meta-associação chamada *Responsabilidade* que visa capturar a participação, responsabilidade e ação das pessoas sobre artefatos ou elementos do processo de software.

A Figura 29 é um fragmento de um modelo de rastreamento que ilustra a associação do tipo responsabilidade, cuja notação é o rótulo "<<resp>>". Em termos de gráficos, na associação de *Responsabilidade* somente capturamos o grau de responsabilidade que pode ser expresso em forma qualitativa ou quantitativa. O exemplo da Figura 29 omite o grau de responsabilidade sobre os diferentes artefatos porque a mesma varia entre as pessoas relacionadas com um mesmo artefato.

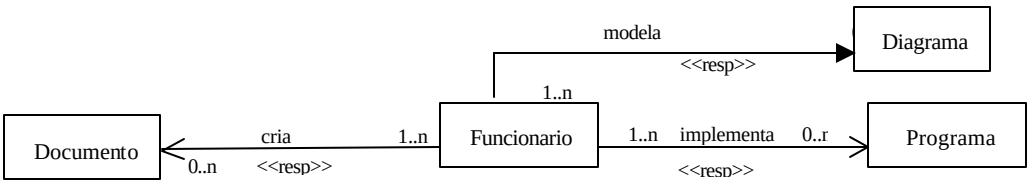


Figura 29: Representação da associação responsabilidade

Na Figura 29, as associações *cria*, *modela*, e *implementa* são do tipo *Responsabilidade*, que juntas, contribuem para modelar a responsabilidade dos funcionários sobre os programas, diagramas e documentos de um projeto.

A associação *cria* expressa que uma pessoa é responsável por zero ou muitos documentos e que um documento é de responsabilidade de uma ou muitas pessoas.

A associação *modela* expressa que uma pessoa é responsável pela modelagem de um ou muitos diagramas e que um diagrama é de responsabilidade de uma ou muitas pessoas.

A associação *implementa* expressa que uma pessoa (funcionário) é responsável pela implementação de zero ou muitos programas e que um programa é de responsabilidade de uma ou muitas pessoas. A Figura 30 introduz uma representação matricial para a associação *implementa*.

A representação matricial da associação *Responsabilidade* é uma tupla composta de três componentes (<*grauResp*; *papel*; *ação* >). A Figura 30 apresenta vários exemplos ilustrando os componentes da associação *Responsabilidade*.

Implementa:<<resp>> →	[PRO -1] modulo_cliente.cpp	[PRO -4] modulo_relatório.cpp	[PRO -6] modulo_acesso.cpp	[PRO -9] modulo_produto.cpp
[FUN-1] Frederico	<A; Ana; entrevista> <A; prg; documentar>		<M; prg; programar>	
[FUN -2] Marco		<A; test; integração>	<A; prg; programar>	
[FUN -5] Evelyn	<A; ana; validar>			<M;ana;documentar>
[FUN -9] Janaína			<A; prg; depurar>	<M; test; unitário>

Figura 30: Representação matricial da associação responsabilidade

O primeiro componente, *grauResp*, indica o grau de responsabilidade da pessoa sobre o artefato. O grau da associação de responsabilidade pode ser expresso em forma quantitativa ou qualitativa (A=alta; M=médio; B=baixo). Por exemplo, a funcionária Evelyn foi altamente responsável pelo programa *modulo_cliente.cpp*. As informações contidas nos outros componentes ajudam a entender melhor a responsabilidade dos funcionários.

O segundo componente, *papel*, especifica o papel usado pelo funcionário no trabalho sobre o artefato. Os papéis foram codificados da seguinte forma: “ana” significa analista; “prg” significa programador; e “test” significa testador. Por exemplo, a funcionária Janaína foi uma programadora altamente responsável pelo programa *modulo_ acesso.cpp*; como testadora foi medianamente responsável pelo teste unitário

sobre o programa *modulo_produto.cpp*. Os papéis devem ser adaptados às necessidades dos projetos.

O terceiro componente, *ação*, descreve o trabalho realizado pelo funcionário. Considerando os dois últimos exemplos, podemos dizer que Janaína, como programadora, foi altamente responsável pela depuração do programa *modulo_acesso.cpp*, enquanto que no papel de testadora foi medianamente responsável pelo teste unitário do programa *modulo_produto.cpp*.

Através da representação matricial da associação *responsabilidade* é possível representar os diferentes papéis, graus de responsabilidade e ações realizadas pelas pessoas sobre os artefatos. Dessa forma, através da informação registrada na matriz da Figura 30, podemos ter um melhor conhecimento, compreensão e identificação das habilidades dos participantes do projeto. Essa informação contribui para que o gerente de projeto conheça melhor seus desenvolvedores e a contribuição dos mesmos no projeto. Porém, é importante salientar que a captura das informações da responsabilidade no processo de software demanda mais tempo e dedicação do encarregado da gerência de requisitos.

A seguir apresentaremos uma outra associação, chamada representação, para rastrear os requisitos através dos diagramas e programas.

3.3.8 A Associação Representação

No início de um projeto a abstração é alta e a precisão é baixa. Na medida que progride o projeto, a abstração tende a diminuir e a precisão a aumentar. Dois fatores que contribuem para isso são: o melhor entendimento do analista sobre o problema e o uso das linguagens (gráfica ou de programação) para expressar o entendimento e satisfação dos requisitos.

A representação dos requisitos nas linguagens de modelagem ou de programação é natural no processo de desenvolvimento de software. Existem empresas e desenvolvedores que omitem a fase de projeto de software e começam a implementar um sistema assim que alguns requisitos do sistema são identificados e compreendidos. Essa situação reflete o fato de que o entendimento de requisitos são transformados e representados diretamente em uma linguagem de programação. Também os requisitos poderiam ser expressos em uma especificação formal, por exemplo Z [Woodcook, 1996], ou na linguagem de modelagem com os objetivos de representar e produzir um sistema.

Para capturar a representação dos requisitos em outras linguagens, propomos uma associação chamada *Representação*. Se os requisitos evoluem, então consultando as associações do tipo representação nas matrizes de rastreamento podemos identificar e atualizar as diferentes representações.

Todas as instâncias da meta-associação *Representação* são representadas por uma seta, rotulada com a expressão "<<rep>>". A Figura 31 é um fragmento de um modelo de rastreamento que ilustra somente a associação *Representação*.

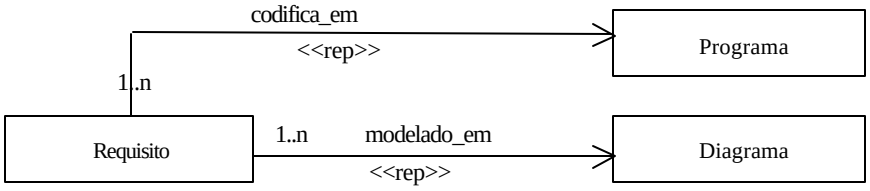


Figura 31: Representação da associação representação

Na Figura 31, a associação *modelado_em* expressa que um requisito é representado em um diagrama e que um diagrama pode conter a modelagem de um ou muitos requisitos. A associação *codifica_em* expressa que um requisito é implementado em um programa e que um programa pode conter a implementação de um ou vários requisitos. A Figura 32 sugere uma representação matricial da associação *modelado_em*.

Modelado_em: <<rep>> →	[DGM-3] diagrama_cliente	[DGM-6] diagrama_controle _acesso
[REQ-1] O sistema deverá incluir clientes com os campos nome, data nascimento, endereço, telefone,...	<rep;"C: CCliente;A: nome, nascimento, telefone; O:inclusaoCliente()" >	
[REQ-2] O sistema deverá imprimir relatórios com os campos nome, curso, ingresso,...	<rep;"C:CCliente;O:gerarRelatorio()">	
[REQ-3] O sistema deverá controla o acesso de pessoas		<rep; "C: CControle; O: acesso(login, senha)" >
[REQ-4] O sistema deverá incluir os usuários no sistema com os campos: nome, login, senha, grupo...		<rep;"C: Ccontrole; O: incluir()"> <rep;"C:Usuario;A: nome, senha, grupo,...">
[REQ-5] O sistema deverá gerenciar um sistema permissão para grupos de usuários		<rep;"C: CControle;O:permissao()">

Figura 32: Representação matricial da associação representação

Na Figura 32, a representação matricial da associação *Representação* consiste de uma tupla composta de dois componentes (<*TipoRepresentação*; *caminhoLógico*>)

O primeiro componente da tupla, *TipoRepresentação*, especifica o tipo de representação utilizada para expressar os requisitos em uma outra linguagem. Alguns dos valores que podem ser atribuídos a *TipoRepresentação* são os rótulos "rep" e

"imp". O rótulo "rep" expressa a modelagem dos requisitos em algum tipo de diagrama. O rótulo "imp" expressa a implementação dos requisitos em algum programa. Por exemplo, a matriz da Figura 32 mostra que os requisitos *[REQ-1], [REQ-2]* foram modelados no diagrama *diagrama_cliente*. Logo, se alguma mudança for realizada sobre *[REQ-1]* e *[REQ-2],* então o diagrama *[DGM-3]* deverá ser revisado e atualizado, se for necessário.

Além dos rótulos "rep" e "imp", podem ser definidos outros rótulos, por exemplo, "esp" para indicar que os requisitos foram representados em uma especificação formal.

O segundo componente da tupla, *caminhoLógico*, é uma seqüência de caracteres que captura a granularidade da associação. O objetivo do *caminhoLógico* é mapear um elemento (por exemplo, requisito) em um outro elemento (pro exemplo, programa). A estrutura do caminho que representa o caminho a percorrer para alcançar o elemento do outro extremo de uma associação. O caminho lógico deve ser definido de acordo com as necessidades de rastreamento de cada projeto. Por exemplo, o caminho lógico pode representar um mapeamento de um requisito em um programa (granularidade grossa) ou um mapeamento de um requisito em uma classe definida em um programa (granularidade fina).

Independentemente do paradigma de desenvolvimento, é necessário usar abreviações para facilitar o entendimento e estruturação dos caminhos lógicos. Na Figura 33 usamos abreviações para classe ("C"), atributo ("A"), operação ("O") e relacionamento ("R") para estruturar os caminhos lógicos. O ponto e vírgula (";") é usado no caminho lógico para separar diferentes tipos de informações, por exemplo, atributos e operações. A vírgula (",") é usada para separar elementos de uma natureza (somente atributos, operações, etc), por exemplo, nome, endereço e departamento são atributos ("A"). A seguir explicaremos alguns exemplos.

1. A expressão "C:nome_classe" expressa que *nome_classe* é o nome de uma classe. Por exemplo, "C:Funcionario" significa que *Funcionário* é uma classe, isto é, que um elemento foi modelado na classe *Funcionario*. Também podemos ter expressões que relacionam elementos de uma mesma natureza. Por exemplo, "C:Funcionário, Departamento, Centro" significa que a modelagem de um elemento foi mapeada nas classes *Funcionário*, *Departamento* e *Centro*.
2. A expressão "A: nome_Atributo" indica que *nome_Atributo* é o nome de um atributo da classe. Por exemplo, "C:Funcionario;A:Cpf,profissão" significa que *Cpf* e *profissão* são atributos da classe *Funcionario*.
3. A expressão "O:nome_operação" especifica que *nome_operação* é o nome de uma operação da classe. Por exemplo, "C:Funcionario;O:consultar()" expressa que *consultar( )* é o nome de uma operação classe *Funcionario*.

4. A expressão "C:Cliente;R:cliente_tem_dependente;C:Dependente" expressa que *cliente_tem_dependente* é um relacionamento.

Na Figura 32, é especificado que o requisito *[REQ-1]* foi modelado no diagrama *diagrama_cliente*, mais especificamente, nos atributos *nome*, *nascimento* e telefone, e na operação *inclusaoCliente()* da classe *CCliente*.

A Figura 33 apresenta alguns caminhos lógicos para mapear requisitos para elementos em um diagrama de classes.

Tipo referência	Caminho lógico	Exemplo
de requisito para uma classe.	"C: nome_classe"	"C: Funcionário"
de requisito para um atributo da classe	"C: nome_classe;A: nome_atributo1, nome_atributo2, nome_atributo3,..."	"C: Funcionario;A:Cpf,profissão,..."
de requisito para um método da classe	"C: nome_classe;O:nome_operação"	"C: Funcionário;O:consultar()"
de requisito para um relacionamento.	"C: classe;R: nome_relacionamento; C: classe"	"C: Funcionario;R: cliente_tem_dependente; C:Dependente"

Figura 33: Exemplos de caminho para mapear requisitos em um diagrama de classes

A Figura 33 é composta de três colunas. A primeira coluna fornece uma descrição do tipo de referência. A segunda mostra a estrutura do caminho lógico para o tipo de referência. A terceira apresenta um exemplo da estrutura lógica.

Com a apresentação da última associação concluímos a apresentação da proposta dos relacionamentos de rastreamento que podem ser empregados na construção de um modelo de rastreamento.

3.4 Comparando nosso Meta-modelo com outros Meta-modelos

Para fornecer uma maior compreensão do nosso meta-modelo, dos seus benefícios, e flexibilidade com respeito aos meta-modelos das pesquisas apresentadas no capítulo anterior; optamos por usar diferentes estratégias de comparação com outros meta-modelos porque muitos deles foram originalmente representados, usando diferentes notações gráficas. Inicialmente abordaremos o trabalho de Gotel.

O trabalho de Gotel não possui um meta-modelo expresso em forma gráfica. Portanto, decidimos elaborar um modelo de rastreamento cujas classes e relacionamentos são instâncias das meta-classes do nosso meta-modelo. A Figura 34

apresenta uma proposta de um modelo de pré-rastreamento de requisitos do trabalho de Gotel.

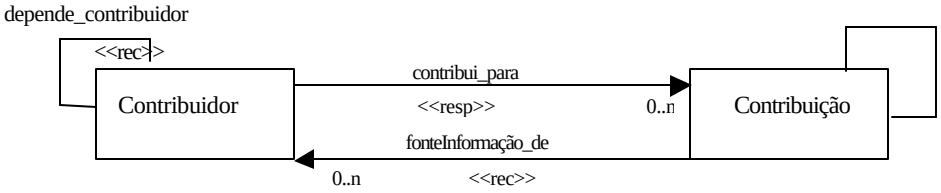


Figura 34: Proposta de um modelo de rastreamento para Gotel

Na Figura 34, os conceitos de contribuidor e contribuição de Gotel são representados pelas classes *Contribuidor* e *Contribuição*, respectivamente. Essas classes são instâncias da meta-classe *Classe*. A associação *contribui_para*, do tipo responsabilidade, expressa que um contribuidor é responsável por zero ou muitas contribuições. A associação *fonteInformação_de*, do tipo recurso, expressa que uma contribuição depende do recurso de zero ou muitos contribuidores. A associação *depende_contribuidor* expressa que um contribuidor depende de um recurso de um outro contribuidor. De forma análoga, a associação *depende_contribuição* expressa que uma contribuição depende de um recurso de uma outra contribuição.

Conforme apresentado no nosso meta-modelo na Seção 3.3, é possível elaborar uma matriz de rastreamento para cada relacionamento da Figura 34. É importante salientar que omitimos alguns adornos nas associações (grau de dependência e árvore lógica) porque não são representados no trabalho de Gotel. Com a elaboração do modelo de rastreamento da Figura 34, desejamos mostrar que nosso meta-modelo pode gerar um modelo de pré-rastreamento que, se usarmos nossos adornos, pode ser ainda mais enriquecido. Com isso concluímos a comparação do nosso meta-modelo com o modelo de rastreamento de Gotel. A seguir comparamos nosso meta-modelo com o meta-modelo de Jarke.

A estratégia de comparação com o trabalho de Jarke consiste em incluir nossos tipos de relacionamento no meta-modelo de Jarke. Lembramos que o meta-modelo de Jarke, que é semelhante ao de Ramesh, propõe quatro tipos de relacionamento: *depende_de*, *satisfaz*, *desenvolvido_para* e *raciocínio*. Alguns dos nossos tipos de relacionamento (recurso, responsabilidade e agregação) podem ser uma especialização do relacionamento *depende_de* que captura a dependência entre artefatos. O nosso relacionamento *satisfação* pode ser uma especialização do relacionamento *satisfaz* porque o relacionamento *satisfação* captura a noção de intenção ou desejo que alguma coisa deve ser realizada para satisfazer alguma condição ou conformidade. Também, o relacionamento de responsabilidade pode ser uma especialização do relacionamento *desenvolvido_para* para capturar a participação das pessoas. O relacionamento recurso

pode ser usado para capturar o raciocínio da evolução dos objetos porque o raciocínio será uma fonte de informação para entender alguma parte do software. Como conclusão da comparação podemos dizer que nossos tipos de relacionamento são semelhantes aos relacionamentos definidos por Jarke, exceto que nossos relacionamentos são melhor definidos, exemplificados e ilustrados em matrizes. A seguir fazemos a comparação com o meta-modelo de Ramesh.

A explicação da comparação, do nosso meta-modelo com o meta-modelo de Ramesh, é análoga à apresentada na comparação com o meta-modelo de Jarke. Porém, a comparação inclui uma instância do nosso meta-modelo para gerar o modelo de gerência de requisitos de Ramesh que foi apresentado e explicado na Subseção 2.4.3.

Todas as classes e relacionamentos da Figura 35 são instâncias das meta-classes do nosso meta-modelo de rastreamento. Por exemplo, os relacionamentos "É um" são instâncias da meta-classe *Generalização* do meta-modelo. Na Figura 35, foram preservados os nomes e notações originais de todos os elementos do modelo de gerência de requisitos de Ramesh. Algumas das modificações realizadas foram a rotulação dos relacionamentos e a mudança do sentido da seta de alguns relacionamentos.

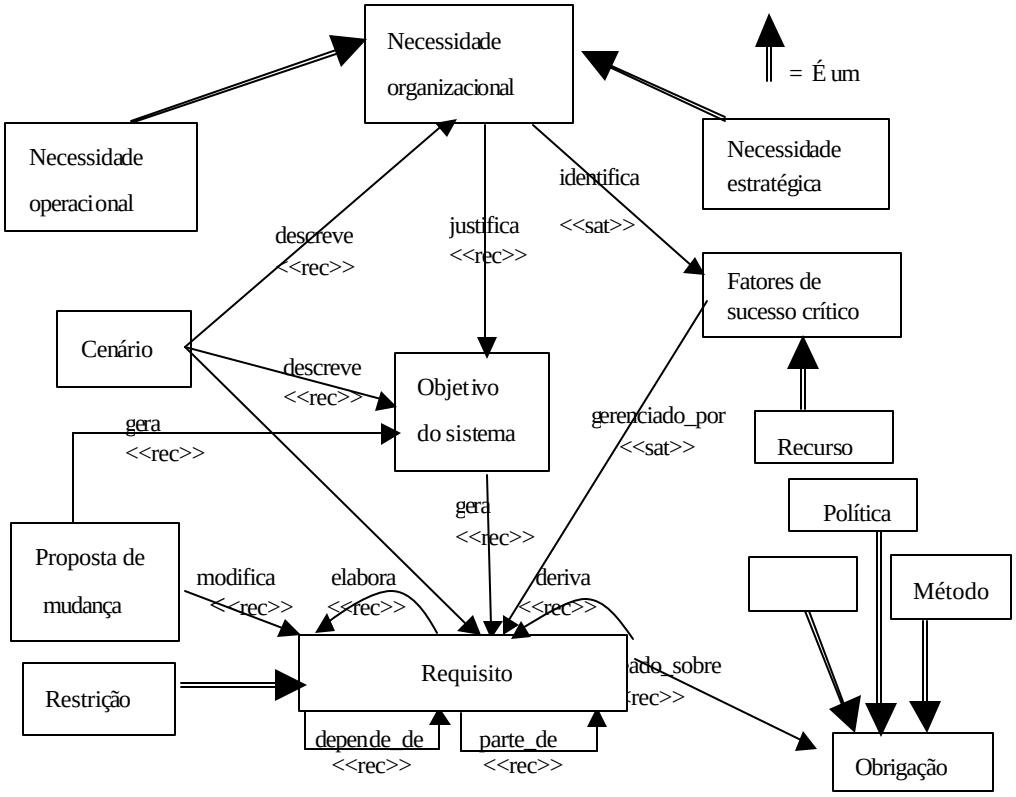


Figura 35: Instanciando do meta-modelo de Toranzo no modelo de Ramesh

A Figura 35 é um outro exemplo que ilustra que todos os elementos e relacionamentos propostos no modelo de gerência de requisitos podem ser mapeados

para alguns dos nossos tipos de relacionamento. Porém, os relacionamentos de Ramesh não incluem os adornos dos nossos tipos de relacionamento. Portanto, os modelos de rastreamento produzidos pelo meta-modelo de Ramesh são semanticamente mais pobres que os modelos produzidos por nosso meta-modelo.

Com esta última comparação concluímos a apresentação da Seção 3.4. A seguir apresentaremos as considerações finais do capítulo.

3.5 Considerações Finais

Neste capítulo apresentamos uma das nossas contribuições para melhorar as atividades de rastreamento de requisitos que consistiu de uma classificação composta de quatro níveis de informação: externo, organizacional, gerencial e desenvolvimento.

Um dos benefícios da classificação é a possibilidade de se ter um ponto de partida para entender a gama de informações existentes que podem formar parte do modelo de rastreamento de um projeto.

O segundo benefício dos níveis de informação é que eles podem ajudar para que as informações candidatas para um modelo de rastreamento possam ser classificadas da mesma forma.

Um terceiro benefício é que o processo de elicitação das informações que farão parte do modelo de rastreamento podem ser fundamentadas nos níveis de informação, isto é, pode ser *top-down* (do externo para o desenvolvimento) ou *bottom-up* (do desenvolvimento para o externo). Por exemplo, no Capítulo 5, apresentaremos nosso processo para construir um modelo de rastreamento fundamentado nos quatro níveis de informação.

O quarto, e último benefício dos níveis de informação é tornar explícita a necessidade de incluir e relacionar as tarefas do plano de projeto em um modelo de rastreamento. A inclusão das tarefas está fundamentada na necessidade dos processos do nível 2 do CMM de que tenham condições de repetir os desenvolvimentos bem sucedidos. Em lugar de visualizar um processo somente como um conjunto de caixas pretas, desejamos que as caixas pretas estejam relacionadas com requisitos porque é difícil falar do sucesso/fracasso de um projeto sem fazer referência aos requisitos. Se foram registradas ligações entre as tarefas e os requisitos, então poderemos compreender melhor a estratégia de trabalho de um gerente de projeto e poderemos identificar alguns dos fatores responsáveis pelo sucesso/fracasso de um projeto. Portanto, poderemos melhorar a estratégia dos gerentes de projetos.

Uma outra contribuição para o rastreamento de requisitos é o meta-modelo de rastreamento que inclui tipos de relacionamento que permitem capturar informações

mais detalhadas do relacionamento entre dois artefatos, por exemplo, grau de dependência, árvore lógica e multiplicidade.

Capítulo 4

Proposta, Aplicação e Validação de um Processo para Desenvolver um Modelo de Rastreamento de Requisitos

---

Os principais objetivos deste capítulo são apresentar um modelo intermediário para rastreamento de requisitos e explicar e ilustrar as diretrizes de um processo para elaborar um modelo de rastreamento sobre um sistema de vídeo locadora.

4.1 Introdução

No capítulo anterior apresentamos um meta-modelo composto de meta-classes cujas instâncias são classes que fazem parte de um diagrama de classe que representa os elementos ou artefatos rastreados em um projeto. A elaboração de um modelo de rastreamento de um projeto tem duas alternativas: começar do zero ou começar a partir da revisão e análise dos modelos de rastreamento existentes. As pesquisas sobre o rastreamento de requisitos (Gotel, Jarke, Pohl, Ramesh e Pinheiro), em geral, apresentam um conjunto de perguntas e sugestões que, como um todo, são insatisfatórias para elaborar um modelo de rastreamento. Nosssa pesquisa apresenta um processo para guiar a elaboração de um modelo de rastreamento de projeto. Oas diretrizes do processo são ilustradas através de um sistema de vídeo locadora.

Este capítulo está estruturado como segue. A Seção 4.2 apresenta e explica um modelo intermediário de rastreamento de requisitos que possui várias classes pré-definidas que são geralmente encontradas nos modelos de rastreamento. A Seção 4.3 apresenta um modelo de raciocínio para registrar a solução de problemas que acontecem no desenvolvimento de software. Uma visão geral do processo para elaborar um modelo de rastreamento é apresentada na Seção 4.4. A Seção 4.5 apresenta uma visão geral do sistema de controle de locadora que será usado para exemplificar o processo proposto para desenvolver um modelo de rastreamento. A apresentação e ilustração do processo é abordada na Seção 4.6. A aplicação de um formulário para registrar o raciocínio e a solução dos problemas acontecidos no desenvolvimento de software é apresentada na Seção 4.7. Finalmente, as considerações finais são apresentadas na Seção 4.8.

4.2 Proposta de um Modelo Intermediário de Rastreamento de Requisitos

Os objetivos desta seção são de apresentar e discutir nosso modelo intermediário de rastreamento. Esse modelo intermediário não pertence a um domínio de aplicação específico. Esse modelo é resultado de uma combinação de fatores, tais como: boas práticas, estudo de caso, abstração, aplicação do nosso meta-modelo, dos níveis de informação e uma extensão de Ramesh pela sua contribuição e experência sobre a elaboração de modelos de rastreamento. A extensão consiste em aplicar uma classificação sobre as informações rastreadas, tipos de relacionamento e um processo para elaborar um modelo de rastreamento. Os principais objetivos do modelo intermediário de rastreamento são:

1. Servir como um ponto de partida para identificação, discussão e construção de um modelo de rastreamento;
2. Identificar as informações encontradas nos modelos de rastreamento dos diferentes projetos. Por exemplo, o modelo intermediário fornece uma visão do rastreamento mais próxima da realidade de um projetista de software porque o modelo identifica os artefatos comuns da fase de projeto;
3. Contribuir na derivação dos modelos de rastreamento dos projetos de software*;*
4. Fornecer uma maior compreensão do nosso meta-modelo de rastreamento de requisitos porque o modelo intermediário é uma instanciação do meta-modelo;
5. Realçar a importância dos tipos de relacionamento do meta-modelo para aumentar o significado e entendimento dos modelos intermediário e particulares de um projeto. No modelo de rastreamento de projeto proposto por Ramesh ([Ramesh, 2001]), o significado atribuído aos relacionamentos é realizado através da nomeação do próprio relacionamento. Por exemplo, será que um leitor qualquer entenderá o significado dos relacionamentos direciona, faz e modifica? Provavelmente não, somente poderá entendê-los se observar atentamente o modelo de Ramesh. Através de exemplos desejamos ilustrar que o significado atribuído a um relacionamento (nome definido pelo usuário) pode ser claro para seu autor, mas não necessariamente para outros usuários;
6. Entender como problemas e soluções aconteceram no processo de software. Independente desse processo, o surgimento dos problemas que afetam o software é inevitável. Para isso, foi proposto um sub-modelo de raciocínio cujo objetivo é capturar as informações necessárias para entender e aprender a partir do erro acontecido.

A Figura 36 mostra o modelo intermediário dentro da camada modelo da arquitetura do meta-dado do MOF (*Meta Object Facility*).

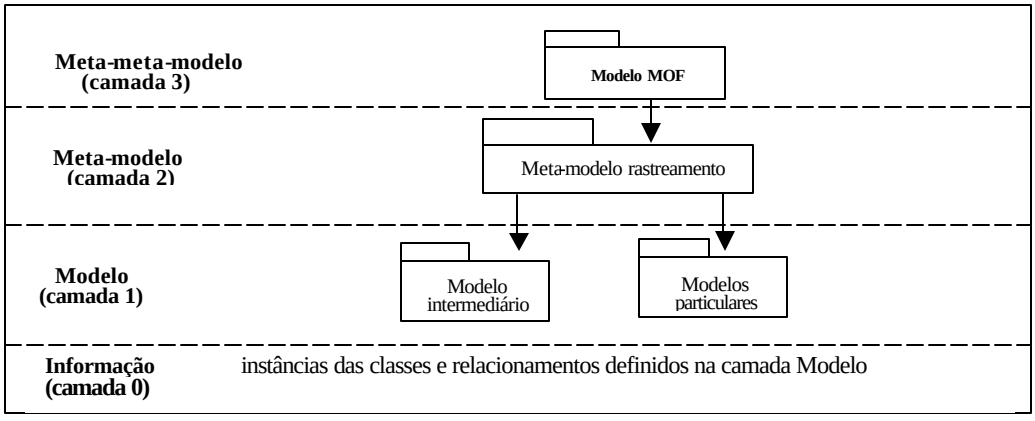


Figura 36: Modelo intermediário de rastreamento na arquitetura de meta-dado do MOF

A Figura 36 mostra que o modelo intermediário e os modelos particulares (modelos de rastreamento) são instâncias do nosso meta-modelo de rastreamento. Os modelos particulares representam os modelos de rastreamento dos diferentes projetos de software desenvolvidos a partir do meta-modelo ou derivados do modelo intermediário. A Figura 37 apresenta o modelo intermediário.

É importante salientar que as propriedades das classes do modelo intermediário devem ser definidas de acordo com as necessidades particulares de cada projeto.

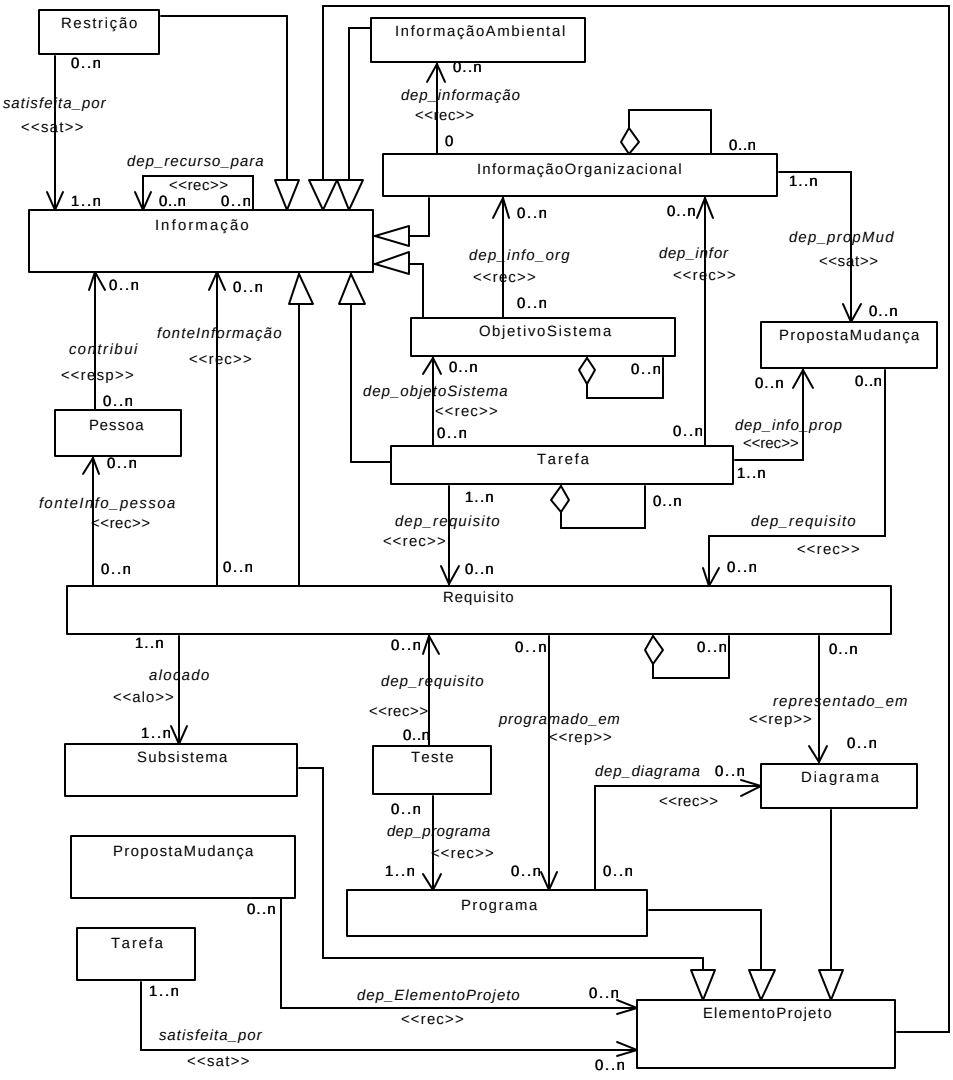


Figura 37: Modelo intermediário para o rastreamento de requisitos

Na Figura 37, a raiz da estrutura hierárquica é a classe *Informação* e algumas das suas subclasses são: *InformaçãoAmbiental*, *InformaçãoOrganizacional*, *ObjetivoSistema*, *Tarefa*, *Requisito*, *Diagrama*, *Programa* e *Subsistema*, *Restrição* e *ElementoProjeto*.

A classe *InformaçãoAmbiental* é uma abstração de todos os possíveis tipos de informações do nível de informação externo, apresentado na Subseção 3.2.1. A classe *InformaçãoOrganizacional* é uma abstração das informações do nível de informação organizacional, apresentado na Subseção 3.2.2. As classes *Tarefa* e *ObjetivoSistema* representam algumas das informações do nível de informação gerencial, apresentado na Subseção 3.2.3. As classes *Requisito*, *Diagrama*, *Programa*, *Subsistema* e *ElementoProjeto* representam algumas das informações do nível de informação de desenvolvimento, apresentado na Subseção 3.2.4.

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. Mesmo com a definição do relacionamento *dep_recurso_para*, decidimos propor outros relacionamentos do tipo de recurso entre algumas classes para facilitar o uso do modelo.

O relacionamento *dep_informação* (de *InformaçãoOrganizacional* para *InformaçãoAmbiental*) é uma abstração das possíveis dependências de recurso das informações do nível de informação organizacional com as do nível de informação externo. Por exemplo, para o rastreamento de um sistema de biblioteca, um tipo de informação do nível externo pode ser o *MARC* [Marc, 2002] que representa um padrão usado para a representação de dados bibliográficos, catalogação de dados, etc.

O relacionamento *dep_info_org* (de *ObjetivoSistema* para InformaçãoOrganizacional) expressa que os objetivos do sistema dependem de algumas informações organizacionais. Os objetivos do sistema estão fundamentados nos objetivos organizacionais sobre o sistema desejado.

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. Por exemplo, uma restrição pode condicionar que todos os requisitos que incluam cálculo deverão conter as fórmulas.

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.

Sabemos que a elaboração de um modelo de rastreamento de um projeto pode necessitar de classes e relacionamentos não definidos no modelo intermediário. De fato, isso pode acontecer porque o modelo intermediário não fornece todas as possíveis classes e relacionamentos usados em todos os modelos de rastreamentos dos diferentes projetos de software. Entretanto, isso não invalida ou inviabiliza a aplicação do modelo intermediário porque o mesmo pode ser melhorado e adaptado às necessidades dos projetos de uma organização.

É importante recomendar o uso de alguns atributos para algumas classes do modelo intermediário para facilitar o uso das mesmas. Por exemplo, considere os seguintes atributos sobre as classes *Requisito* e *InformaçãoOrganizacional*.

a) *Identificação*. identificação da instância;
b) *Descrição*. Representação textual da descrição da classe;
c) *Estado*. Estado da instância da classe (aprovado, reprovado, implementado, por exemplo);
d) *Data de criação*. Data de criação da instância;
e) *Fonte de informação*. Referência à fonte de informação do requisito.

Para facilitar a apresentação do modelo intermediário, o mesmo foi dividido em dois sub-modelos: *gerência de requisitos* e *projeto de* software. Os sub-modelos são modelos de rastreamento.

No decorrer da pesquisa foi constatado que existe um número mínimo de classes do modelo intermediário de rastreamento que devem ser consideradas na elaboração de um modelo de rastreamento, elas são: Tarefa, Requisito, Programa, Objetivos Organizacionais e Objetivos do Sistema. As outras classes do modelo intermediário são opcionais.

4.2.1 Sub-modelo para a Gerência de Requisitos

Independente da natureza do projeto de software, o surgimento de novos requisitos e as mudanças dos requisitos existentes do sistema são inevitáveis [Kotonya, 1998]. Além disso, se acrescentamos a complexidade e o porte do sistema, e as centenas de requisitos a gerenciar, claramente existirão problemas de gerência de requisitos, tais como: a dificuldade de consultar informações relacionadas; a dificuldade da recuperação de informações relacionadas para avaliar uma proposta de mudança; e ter um melhor acompanhamento do projeto em termos do estado dos requisitos (aprovado,

rejeitado, ou implementado). Portanto, é possível afirmar que existirá a necessidade de gerenciar requisitos.

A gerência de requisitos é uma abordagem sistemática para a elicitação, organização e documentação dos requisitos de um sistema [Leffingwell, 2000; Locke, 1999]. Determinar e gerenciar requisitos são duas atividades importantes do processo de software porque os requisitos formam a base para o planejamento, o acompanhamento do desenvolvimento, e a aceitação dos resultados de um projeto [Fiorini, 1998].

A Figura 38 apresenta o sub-modelo para a gerência de requisitos que omite algumas classes e relacionamentos que serão explicados na Subseção 4.2.2 que explica o sub-modelo de rastreamento para o projeto de software.

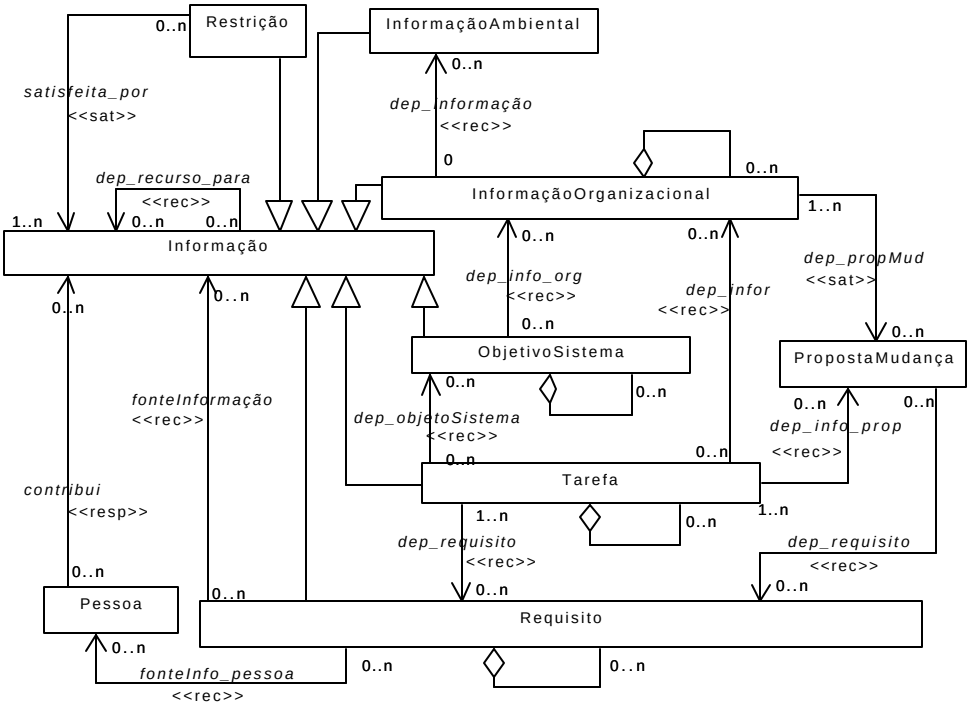


Figura 38: Sub-modelo para a gerência de requisitos

A seguir explicaremos brevemente os relacionamentos do sub-modelo de rastreamento para a gerência de requisitos.

O relacionamento *dep_propMud* (de *InformaçãoOrganizacional* para *PropostaMudança*) expressa que a satisfação de uma informação organizacional depende da satisfação de nenhuma ou muitas propostas de mudança. Por exemplo, uma administradora de cartões de crédito que deseja incentivar os seus clientes a usarem os

seus cartões, optou por sortear viagens entre os mesmos. Cada vez que o cliente paga suas compras (com valor superior a R$50) com um dos cartões da administradora, o sistema garante a geração de um cupom para participar do sorteio das viagens. Para viabilizar isso, a administradora propõe uma mudança no sistema para identificar as compras com valor superior a R$ 50. Portanto, o objetivo de incentivar o uso dos cartões da administradora depende da efetivação da mudança proposta.

O 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*, desejam capturar algumas informações que os gerentes de projetos têm em mente na elaboração de um cronograma de um projeto. Por exemplo, o relacionamento *dep_requisito* (de *Tarefa* para *Requisito*) captura os requisitos que são recursos de informações para as tarefas.

O relacionamento *dep_objetivo_sistema* (de *Tarefa* para *ObjetivoSistema*) captura os objetivos do sistema que são recursos de informação para cada uma das tarefas.

O relacionamento *dep_infor* (de *Tarefa* para *InformaçãoOrganzacional*) especifica que algumas tarefas estão fundamentadas em informações organizacionais (objetivos, estratégias, prazos, etc).

O relacionamento *dep_info_prop* (de *Tarefa* para *PropostaMudança*) expressa que uma tarefa depende do recurso de informação contida em uma proposta de mudança para definir e organizar as tarefas que introduzirão as modificações no sistema.

O relacionamento *dep_requisito* (de PropostaMudança para Requisito) especifica que os requisitos são recursos de informação para uma proposta de mudança.

O relacionamento *fonteInfo_pessoa* (de *Requisito* para Pessoa) expressa que podem existir pessoas que são recursos de informação para os requisitos.

O relacionamento *fonteInformação* (de *Requisito* para *Informação*) expressa que um requisito pode ter como recurso de informação qualquer instância das subclasses da classe *Informação*.

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

4.2.2 Sub-modelo de Rastreamento para o Projeto de Software

Um sub-modelo de rastreamento para o projeto de software identifica os elementos geralmente encontrados na fase de projeto de um sistema.

A Figura 39 apresenta esse sub-modelo de rastreamento para o projeto de software. Algumas classes-chave do sub-modelo são: *Subsistema*, *Programa*, *Diagrama*, *Tarefa* e *PropostaMudança*. A classe *ElementoProjeto* é a superclasse de algumas dessas classes.

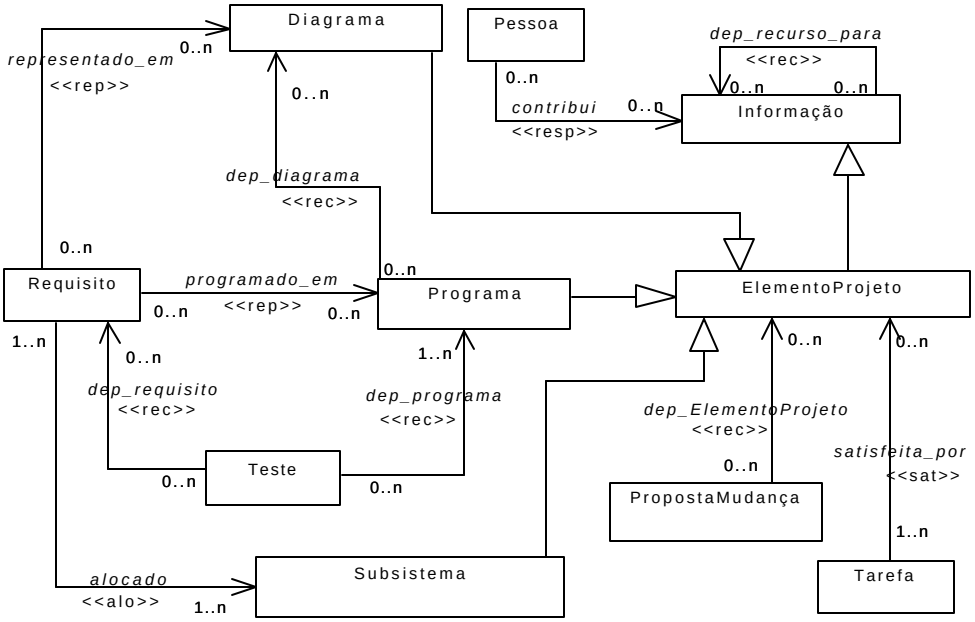


Figura 39: Sub-modelo de rastreamento para o projeto de software

Na Figura 39, o relacionamento *satisfeita_por* (de *Tarefa* para *ElementoProjeto*) expressa que a satisfação de uma tarefa depende da realização de alguns elementos do projeto. Por exemplo, a satisfação de uma tarefa, que especifica a depuração de um programa, depende da realização de uma correção bem sucedida sobre o programa.

O relacionamento *dep_ElementoProjeto* (de *PropostaMudança* para *ElementoProjeto*) expressa que uma proposta de mudança tem uma dependência de recurso de informação com alguns artefatos do projeto de software.

Os relacionamentos *programado_em* e *representado_em* (de Requisito para Programa e Diagrama) expressam que os requisitos são representados em outra linguagem. Por exemplo, requisitos podem ser representados em diagramas (caso de uso e de classe, por exemplo) e programas (C++ e Java).

O relacionamento *aloca* (de *Requisito* para *Subsistema*) expressa que requisitos são atribuídos a um ou muitos subsistemas. Por exemplo, um requisito funcional ("o sistema deverá incluir clientes...") está alocado a um subsistema, enquanto um requisito não-funcional ("o sistema deve ser seguro") pode estar alocado em mais de um subsistema.

Os relacionamentos *dep_requisito* e *dep_programa* (de *Teste* para *Requisito* e *Programa*) expressam que um teste depende das informações contidas nos requisitos e de um programa para sua elaboração.

O relacionamento *dep_diagrama* (de *Programa* para *Diagrama*) expressa 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 muitas decisões são tomadas em função de novos problemas e necessidades. Geralmente essas decisões ficam registradas na mente dos participantes e responsáveis. A Seção 4.3 apresenta um modelo de raciocínio para identificar e estruturar os problemas e as soluções tomadas (raciocínio) no decorrer do desenvolvimento de um software.

4.3 Sub-Modelo para o Raciocínio

Na Subseção 2.4.3 foi apresentado o modelo IBIS [Conklin, 1988] para registrar os problemas (assunto no modelo IBIS) e seu raciocínio e solução acontecidos no decorrer do desenvolvimento de um projeto. Nosso modelo de raciocínio estende o modelo IBIS de duas formas. A primeira, inclui alguns conceitos, tais como: objetivo, suposição e restrição. A segunda, inclui a aplicação dos tipos de relacionamento para melhorar a compreensão e construção de um modelo de raciocínio para um projeto. O modelo é usado na tese para capturar o raciocínio dos problemas (assuntos) e suas soluções tomadas no decorrer de um projeto. Nesta seção apresentaremos um formulário para captura o raciocínio.

A Figura 40 apresenta nosso modelo de raciocínio. Na mesma figura, um *Assunto* representa um problema ou uma preocupação que requer uma discussão e resolução. Independente do assunto, deveriam ser considerados os objetivos do sistema, as restrições do projeto, e a existência de suposições. Por isso que foram considerados os relacionamentos do tipo recurso da classe *Assunto* com as classes *ObjetivoSistema*, *Restrição* e *Suposição*.

O relacionamento *dep_ objetivo* (de *Assunto* para *ObjetivoSistema*) expressa que um assunto (um problema acontecido) tem uma dependência de recurso com os objetivos do sistema, isto é, o assunto deve considerar as informações dos objetivos do sistema para saber se afeta ou está fora do escopo dos mesmos. Por exemplo, os desenvolvedores podem propor a modificação de um relatório visando sua melhoria, porém, um objetivo do sistema estabeleçe que todos os relatórios devem seguir os padrões definidos pelo cliente. Esse exemplo, ilustra que é necessário que um assunto proposto faça uma revisão dos objetivos do sistema.

O relacionamento *dep_restrição* (de *Assunto* para *Restrição*) expressa que um assunto (um problema) deve considerar as restrições existentes. Um assunto pode expressar a escolha de um estilo de documentação, porém, pode existir uma restrição do sistema estabelece que o estilo de programação usado no projeto deve ser a definida e usada pelo cliente.

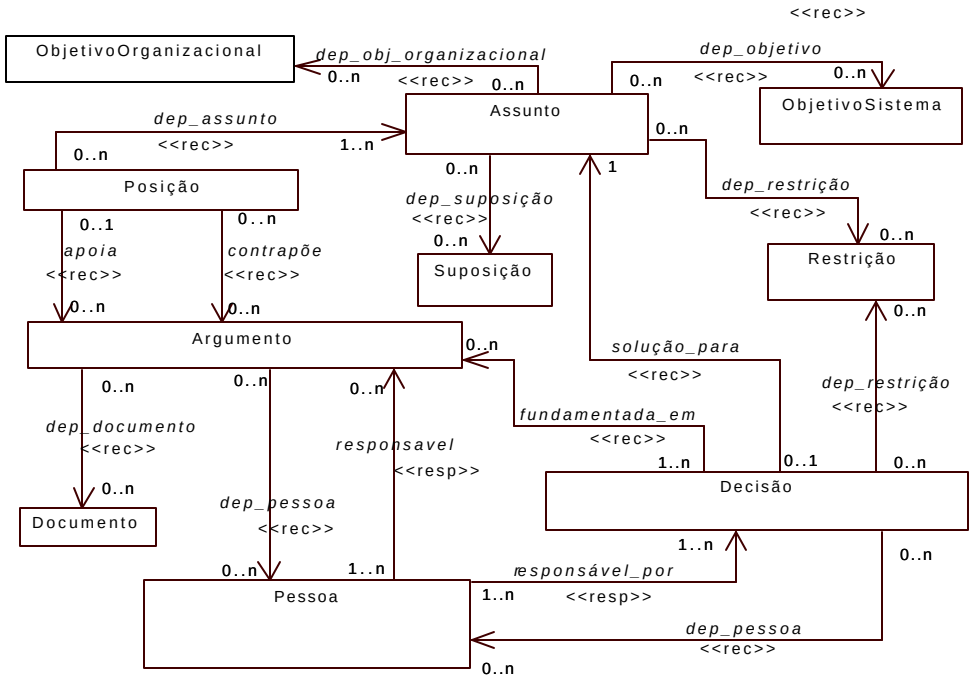


Figura 40: Sub-modelo para o raciocínio

O relacionamento *dep_suposição* (de *Assunto* para *Suposição*) expressa que um assunto pode ter como recurso de informação nenhuma ou muitas suposições. Por exemplo, o problema de integrar várias ferramentas CASE pode ter como suposição que um Sistema Gerenciar de Banco de Dados Orientado a Objetos (SGBDOO) pode permitir uma melhor integração por dados.

No modelo de raciocínio, uma posição representa uma solução alternativa para um assunto. O relacionamento *dep_assunto* (de *Posição* para *Assunto*) expressa que um assunto é um recurso de informação para uma posição. Por exemplo, considerando o problema da integração de ferramentas de um projeto, uma posição pode recomendar a compra de um SGBDOO enquanto que uma outra posição recomenda a compra de um Sistema Gerenciar de Banco de Dados Relacional.

Os relacionamentos *apóia* e *contrapõe* (de *Posição* para *Argumento*) expressam que uma *Posição* pode ter como recurso de informação nenhum ou muitos argumentos a

favor (apóiam) ou contra (contrapõem-se). No 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 em contra pode argumentar que a tecnologia relacional está madura e consolidada no mercado industrial.

Os argumentos podem ter como recursos de informação documento e pessoa. Isso é expresso pelos relacionamentos *dep_documento* e *dep_pessoa* (de *Argumento* para *Documento* e *Pessoa*). Por exemplo, um argumento que recomenda a compra de um Sistema Geranciar de Banco de dados (SGBD) pode ter como recurso de informação uma pessoa especialista na área de integração de ferramentas e ambientes CASE.

O relacionamento *responsável* (de *Pessoa* para *Argumento*) expressa que uma pessoa pode ser responsável por nenhum ou muitos argumentos. Por exemplo, um programador pode ser responsável por um argumento que recomenda a compra de um ambiente de programação para um projeto.

A classe *Decisão* representa uma coleção de decisões. O relacionamento *solução_para* (de *Decisão* para *Assunto*) especifica que uma decisão tem como recursos de informação o assunto. Por exemplo, a decisão da escolha de uma ferramenta para fazer uma integração por dados, está associada a um assunto (um problema).

O relacionamento *fundamentada_em* (de *Decisão* para *Argumento*) especifica que a tomada de decisão tem como recurso de informação alguns argumentos.

O relacionamento *dep_restrição* (de *Decisão* para *restrição*) especifica que uma decisão tem como recurso de informação as restrições do projeto. A idéia é que uma solução não poderia invalidar algumas restrições impostas a um projeto.

O relacionamento *dep_pessoa* (de *Decisão* para *Argumento*) especifica que uma decisão tem como recurso de informação as pessoas para futuros esclarecimentos.

O relacionamento *responsavel_por* (de *Pessoa* para *Decisão*) especifica que as pessoas são responsaveis pela tomada de decisões.

Na Figura 41 sugerimos um formulário para registrar o raciocínio de algum problema. O formulário apresenta um problema fictício de um projeto de pesquisa multi-institucional formado por grupos de pesquisas geograficamente dispersos. O projeto teve como objetivo desenvolver um ambiente de engenharia de software centrado em processo. Para isso, os diferentes grupos desenvolveram diferentes ferramentas que foram integradas. No decorrer do projeto ocorreu o problema de ter que escolher uma ferramenta para integrar os sistemas.

Registro do Raciocínio					
Identificador: Prioridade:				Data:	
Assunto					
O projeto de pesquisa "Laboratório de Engenharia de Software" deverá realizar uma reunião de acompanhamento dos diferentes grupos de pesquisas que compõem o projeto e que estão localizados em diferentes pontos do Brasil. O projeto deverá tomar a decisão de comprar uma ferramenta para integrar as diferentes ferramentas que estão sendo desenvolvidas pelos diferentes grupos de pesquisas					
Objetivos organizacionais	Objetivos do Sistema	Restrições			
Criar um ambiente integrado de engenharia de software	1. desenvolver editores gráficos 2. Desenvolver tradutores para a lógica modal de ações	O projeto tem 12 meses para desenvolver o produto			
Posições					
Apóia		Contrapõe			
Argumento	Fonte Responsável	Argumento		Fonte	Responsável
1) Comprar uma implementação do PCTE (Portable Common Tool Environment) para integrar as ferramentas dos grupos de pesquisa	J. Pinto J. pinto	Não comprar PCTE porque seu mecanismo de armazenamento é relacional		J. Prado	S. Meira
2) Comprar um SGBDOO. Possui APIs para as diferentes linguagens de programação usadas pelos diferentes grupos	Revista J. Castro	Não comprar um SGBDOO porque tecnologia OO para a área de Banco de dados está imatura		E. silva	P. Lucena
Decisão					
1) Será comprado um SGBDOO porque um dos objetivos do projeto é explorar a tecnologia objeto.		Responsável	Coordenadores dos diferentes grupos de pesquisas		
		Fonte	Documentos do projeto aprovados pelo órgão financiador.		
2) Fazer uma seleção e avaliação dos SGBDOO existentes e escolher um para ser comprado para todos os grupos de pesquisa		Responsável	M. Toranzo, C. Gautreau e S. Silva		
		Fonte	M. Toranzo, C. Gautreau e S. Silva		
Restrições sobre a decisão					
1) Devem ser compradas 4 licenças do produto 2) Os produtos deverão ser pagos pelo coordenador do projeto					

Figura 41: Proposta de formulário para registrar o raciocínio sobre um problema

Com a ilustração do formulário, concluímos a apresentação do modelo de raciocínio. A seguir apresentaremos uma visão geral do nosso processo para construir um modelo de rastreamento.

4.4 Uma Visão Geral do Processo e Sistemática do Trabalho Realizado

O objetivos desta seção é fornecer uma visão geral do processo para a elaboraração de um modelo de rastreamento. Os quatro importantes componentes do processo são:

1. A classificação dos quatro níveis de informação para o rastreamento de requisitos, apresentada na Seção 3.2;
2. O meta-modelo de rastreamento, apresentado na Seção 3.3;
3. Um modelo intermediário de rastreamento de requisitos, apresentado na Seção 4.2;
4. Restrições de projeto. Algumas das restrições podem ser padrões organizacionais e restrições sobre os artefatos produzidos no processo de software.

O objetivo da Tabela 3 é organizar (por fase de desenvolvimento) as classes candidatas identificadas por cada uma das diretrizes do processo para elaborar um modelo de rastreamento. A primeira linha da Tabela 3 identifica as fases genéricas de um processo de software. A primeira coluna da tabela Tabela 3 identifica os quatro níveis de informação para o rastreamento. Vale salientar que estas fases foram usadas somente a título de ilustração. Processos diferentes, com certeza, possuem fases distintas. Para cada uma das fases de um processo, como o identificado na Tabela 3, são identificados os elementos/artefatos que serão rastreados. Por exemplo, a Tabela 3 indica que na fase de análise as informações rastreadas serão Leis, pessoas e requisitos. Já na fase projeto serão diagrama e subsistema. As pessoas são incluída para capturar sua participação sobre os elementos/artefatos produzidos para um software.

Níveis de informação	Análise	Projeto	Implementação	Teste
Externo	Lei			
Organizacional				
Gerencial Desenvolvimento	Pessoa Requisito	Diagrama, Subsistema	Programa	

Tabela 3: Tabela para organizar as classes candidatas de um modelo de rastreamento

O processo para o desenvolvimento de um modelo de rastreamento está divido em três fases:

1. Coleta de informações. Engloba as atividades de identificação de informações dos quatro níveis;
2. Estruturação e construção do modelo de rastreamento. Está relacionada com a organização das classes;
3. Definição e preenchimento das matrizes de rastreamento. Consiste da definição, o preencimento e validação das matrizes de rastreamento.

A Figura 42 apresenta um diagrama de atividade da UML (Unified Modeling Language [Booch, 1999]) que identifica as diferentes atividades do processo para desenvolver um modelo de rastreamento.

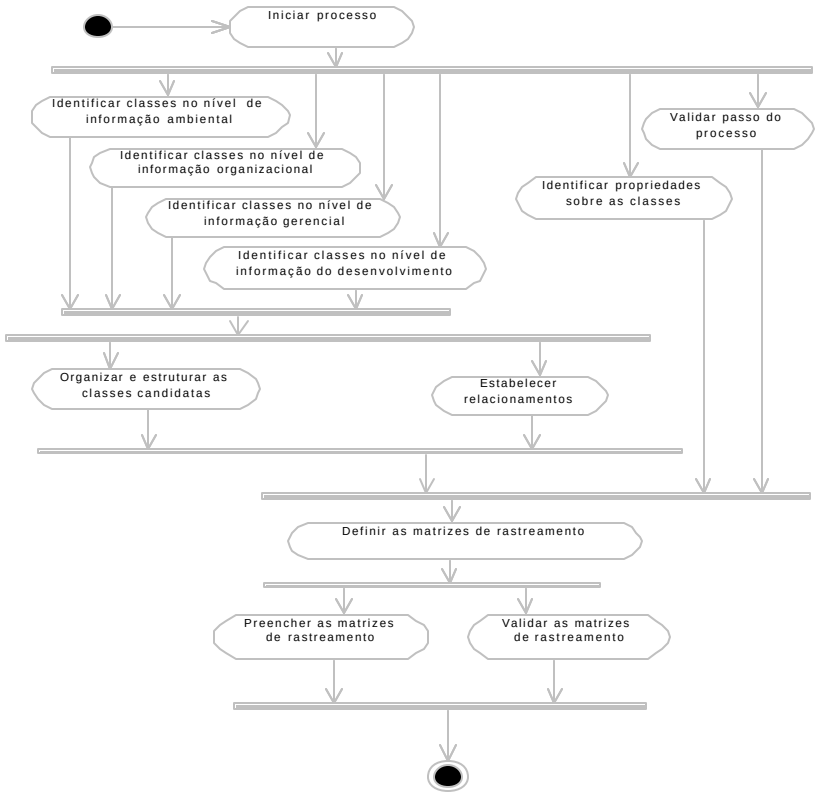


Figura 42: Visão geral do processo para elaborar um modelo de rastreamento

A primeira fase do processo, chamada coleta de informação, está composta de quatro atividades concorrentes: identificar classes no nível de informação externo;

identificar classes no nível de informação organizacional; identificar classes no nível de informação gerencial; e identificar classes no nível de informação do desenvolvimento. O objetivo dessas atividades é identificar classes candidatas dos diferentes níveis de informação que podem compor o modelo de rastreamento de um sistema.

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

A terceira fase, chamada definição e preenchimento das matrizes de rastreamento, está formada pelas seguintes atividades: preencher as matrizes de rastreamento e validá-las. O objetivo das duas atividades é preencher e validar as matrizes, e capturar as informações das dependências (*grau de satisfação, grau de dependência*, etc) 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.

A sistemática de trabalho para elaborar um modelo de rastreamento do sistema de vídeo locadora começo fornecendo aos participantes um treinamento teórico e prático sobre o rastreamento de requisitos. O aspecto teórico foi fundamentado, entre outros, nos trabalhos de [Ramesh, 1998], [Ramesh, 2001], [Wiegers, 1999], [Gotel, 1996a] e [Toranzo, 1999]. Após concluído o treinamento teórico, os participantes trabalharam no desenvolvimento do modelo de rastreamento do sistema de vídeo locadora. Posteriormente, os participantes conheceram o modelo intermediário e o processo para desenvolver um modelo de rastreamento. Ambas informações contribuiram para uma melhor compreensão do rastreamento e da validação do trabalho. Problemas aconteceram no decorrer deste e outros projetos que contribuiram na melhoria das nossas contribuições apresentadas no Capítulo 1. Esses problemas foram identificados porque os projetos foram acompanhados e supervisionados para avaliar o progresso de todos os grupos de trabalhos. Cada um dos trabalhos finais (modelo e a atividade de rastreamento) foram avaliados por pessoas de outros grupos que não participaram na elaboração do modelo, nem no rastreamento do projeto em questão. A estratégia de validação foi checar se as matrizes de rastreamento produzidas contribuiriam na manutenção do próprio sistema que, neste caso, seria realizada por pessoas que não implementaram o sistema em questão. Na Seção 4.8 mostraremos as conclusões sobre o projeto de sistema de vídeo locadora.

Na Seção 4.5 forneceremos uma descrição de um sistema de vídeo locadora que será usado para exemplificar o processo para desenvolver um modelo de rastreamento. Esse

processo é apresentado na Seção 4.6. A aplicação do mesmo processo sobre um estudo de caso é apresentado no Capítulo 5.

4.5 Descrição do Sistema de Controle de Locadora

O objetivo desta seção é fornecer uma descrição geral de um sistema SISLOCA – Sistema de controle de locadora que será usado para exemplificar várias diretrizes do processo para elaborar um modelo de rastreamento. O sistema foi desenvolvido para uma empresa que atua no setor de locação e vendas de fitas de vídeo. SISLOCA possui cinco grandes funcionalidades: gerência de clientes; gerência de produtos (fitas e filmes); locação de filmes; emissão de relatórios; e controle da entrada dos funcionários (cartão ponto). No decorrer do capítulo apresentaremos várias telas desse sistema para sua maior compreensão. A Figura 43 apresenta a tela principal do sistema.

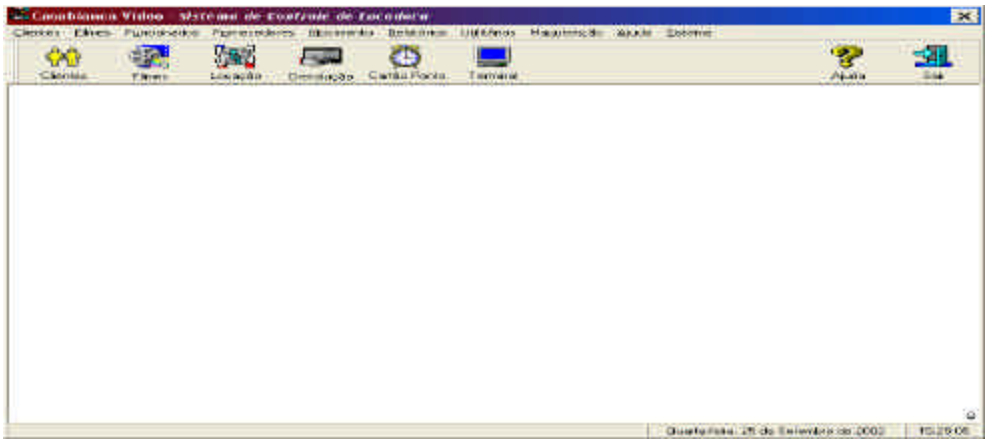


Figura 43: Tela principal do sistema de controle de locadora

A tela principal fornece acesso aos funcionários cadastrados no sistema. A Figura 44 apresenta a tela disponível para os clientes.

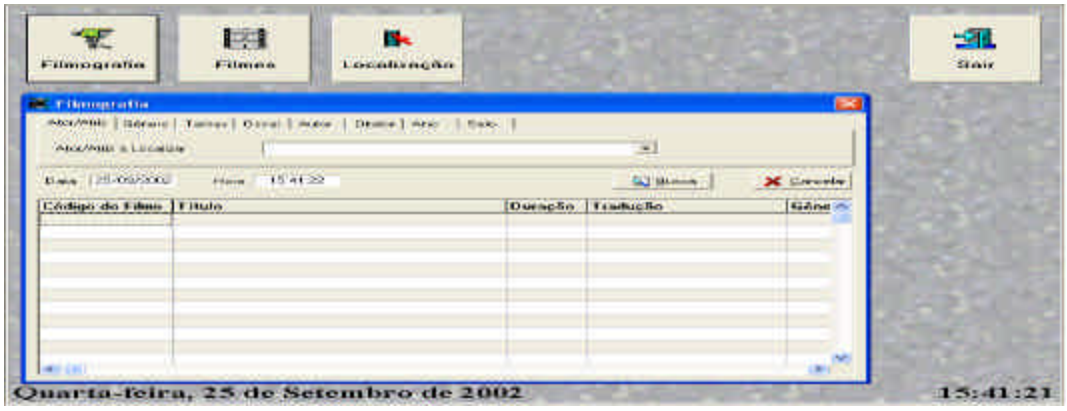


Figura 44: Tela principal dos clientes do sistema de controle de vídeo locadora

A tela disponível para os clientes lhes oferece três funções (Filmografia, Filme e Localização) para fazer pesquisas sobre o acervo de filmes da loja.

Na seção seguinte explicaremos o processo para a construção de um modelo de rastreamento para este sistema.

4.6 Exemplicação de um Processo para Desenvolver um Modelo de Rastreamento

Os objetivos desta seção são apresentar e aplicar as diretrizes do processo para desenvolver e avaliar um modelo de rastreamento de requisitos. As subseções seguintes explicam e exemplificam as diretrizes do processo para elaborar um modelo de rastreamento sobre o sistema de controle de locadora, apresentado na Subseção 4.5. É importante salientar que o processo de rastreamento foi realizado durante o desenvolvimento do sistema e que as telas apresentadas nas próximas subseções correspondem à versão final desse produto.

É importante salientar que o objetivo da Tabela 3 é organizar (por fase de desenvolvimento) as classes candidatas identificadas por cada uma das diretrizes do processo para elaborar um modelo de rastreamento. A atribuição das classes candidatas nas diferentes fases do desenvolvimento depende da experiência do encarregado do rastreamento e dos desenvolvedores.

4.6.1 Informar as Expressões e Convenções usadas nas Matrizes de Rastreamento

A maior parte das informações contidas nesta seção foram apresentadas na Seção 3.3.2, mas foram repetidas aqui para facilitar a leitura e compreensão do processo para

desenvolver um modelo de rastreamento. As expresões e convenções utilizadas para representar um relacionamento do modelo de rastreamento em uma matriz são as seguintes:

1. O nome e tipo do relacionamento representado na matriz são colocados na parte superior esquerda da matriz. O tipo do relacionamento é representado por um rótulo. Por exemplo, o rotulo "<<*rec*>>" representa um relacionamento do tipo recurso.
2. Junto com o nome do relacionamento colocaremos uma seta ("→" ou "←") para identificar as instâncias da classe destino na matriz. Se for usado o símbolo "→", então as instâncias declaradas na primeira linha da matriz são instâncias da classe destino na associação. Caso contrário ("←") as instâncias declaradas na primeira coluna da matriz são instâncias da classe destino. Isso ajuda a entender o sentido do relacionamento e identificar as instâncias com o papel de dependentes.
3. Na primeira coluna e primeira linha das matrizes de rastreamento podem ser usadas expressões (abreviações) para identificar as instâncias de uma classe. A seguir recomendamos algumas expressões que podem ser adaptadas e/ou modificadas para o projeto que será rastreado.

[OBO] = Objetivo organizacional	[PRG] = Programa
[OBS] = Objetivo do sistema	[DGM] = Diagrama
[REQ] = Requisito do sistema	[TAR] = Tarefa da gerência
[RES] = Restrição do sistema	[FUN] = Funcionário
[LEI] = Artigo de Lei	[SUB] = Subsistema do sistema

A Figura 45 ilustra o uso das expressões que estão associadas com números para ajudar a identificar as diferentes instâncias das classes.

Satisfeito_em: <<sat>> ←	[OBO-1] Agendar reunião	[OBO-5] Ter um banco de informações das reuniões
[OBS-3] Requer dos participantes datas de participação...	<A; A>	
[OBS-4] Agendar uma reunião para todos ...	<A; A>	
[OBS–5] Informar a data da reunião.	<A; A>	

Figura 45: Exemplo do uso de expressões

Na Figura 45, as expressões *[OBO-1]* e *[OBO-5]* são instâncias da classe que representa os objetivos organizacionais. As expressões *[OBS-3]*, *[OBS-4]* e *[OBS-5]* são instâncias da classe que representa os objetivos do sistema. Na parte superior

esquerda da mesma figura, aparece o nome do relacionamento *Satisfeito_em* e seu tipo (*Satisfação*).

A seguir abordaremos a identificação das classes candidatas para o modelo de rastreamento.

4.6.2 Identificar Classes no Nível de Informação Externo

Considerando o modelo intermediário de rastreamento de requisitos (Figura 37), preencha a Tabela 3 com os resultados obtidos com cada uma das diretrizes do processo para elaborar um modelo de rastreamento.

Diretriz 1: Identificar as informações que podem afetar o sistema. O objetivo da diretriz é identificar classes no nível de informação externo que podem afetar o sistema. Algumas perguntas que podem ser formuladas são:

1. No domínio de aplicação existem leis ou regulamentações legais que devem ser respeitadas e implementadas? Por exemplo, um sistema de folha de pagamento é afetado diretamente por leis que fixam o valor do salário família, as porcentagens para a retenção do imposto de renda e da previdência social.
2. No domínio de aplicação existem normas ou padrões que serão implementados? Por exemplo, a implementação de um sistema de biblioteca pode incluir normas internacionais de catalogação e formatação de obras bibliográficas.

Na análise do sistema de vídeo locadora, identificamos que uma lei brasileira obriga a emissão de notas fiscais pelos serviços de locação de filmes, porém, como o sistema de vídeo locadora não preencherá notas fiscais, podemos afirmar que a abstração representada pela classe *InformaçãoAmbiental* do modelo intermediário não se aplica ao modelo de rastreamento do sistema.

4.6.3 Identificar Classes no Nível de Informação Organizacional

Considerando o modelo intermediário de rastreamento de requisitos, preenche-se a Tabela 3 com os resultados obtidos com a seguinte diretriz.

Diretriz 2: Identificar os objetivos, estratégias ou regras de negócio que serão rastreadas. O objetivo dessa diretriz é identificar os objetivos, estratégias ou regras registradas que podem afetar o sistema desejado. No nível de informação organizacional, a necessidade de implementar um sistema é expresso freqüentemente em termos gerais (objetivos, regras de negócio e restrições) porque as pessoas têm uma

noção geral do sistema desejado. Algumas perguntas que podem ser formuladas para identificar classes de informação são:

1. Os objetivos organizacionais sobre o sistema foram registrados?
2. Existem estratégias organizacionais que efetam os requisitos do sistema?
3. Existem regras do negócio que efetam os requisitos do sistema? Esse tipo de informação é uma fonte de conhecimento para os requisitos elicitados.

A classe *InformaçãoOrganizacional* do modelo intermediário foi incluída como classe candidata para o modelo de rastreamento porque foram documentados (na fase de análise) os objetivos organizacionais sobre o sistema. Acresecente a classe *InformaçãoOrganizacional* na sua tabela de classes candidatas para o rastreamento (como a Tabela 3). De aqui para frente, acrescente as classes obtidas com a aplicação de cada uma das diretrizes fornecidas pelo processo. A posição das classes na tabela (como a Tabela 3) indicará o seu nível de informação e a fase do processo na qual será rastreada. Na Tabela 16 apresentaremos as classes obtidas com a aplicação das várias diretrizes.

A Tabela 4 apresenta os objetivos organizacionais que foram extraídos da especificação de requisitos da vídeo locadora.

Código	Descrição
[OBO-1]	Facilitar o acesso dos clientes às informações sobre os vídeos da loja
[OBO-2]	Controlar a entrada e saída dos funcionários
[OBO-3]	Gerenciar as informações dos clientes
[OBO-4]	Gerenciar as informações dos produtos (fitas e filmes)
[OBO-5]	Obter relatórios para tomada de decisão de compra de filmes
[OBO-6]	Obter uma carteira de identificação da loja para os clientes
[OBO-7]	Obter um contrato entre a vídeo locadora e o cliente

Tabela 4: Objetivos organizacionais do sistema de controle de locadora

4.6.4 Identificar Classes no Nível de Informação Gerencial

O objetivo é apresentar uma diretriz que recomenda a inclusão das classes *Tarefa*, *ObjetivoSistema* e *Pessoa* do modelo intermediário de rastreamento no modelo de rastreamento do projeto.

Diretriz 3: Incluir classes de informação da gerência de projeto no modelo de rastreamento. O objetivo dessa diretriz é recomendar a inclusão das tarefas do

cronograma do projeto, os objetivos do sistema e pessoas (desenvolvedores) nos modelos de rastreamento dos projetos porque desejamos reduzir o *gap* da gerência de projeto com a gerência de requisitos. Para isso, devemos associar e registar os relacionamentos das tarefas com requisitos para identificar e entender melhor a distribuição dos requisitos no decorrer do projeto e compreender o sucesso ou fracaso de um projeto, do ponto de vista da gerência de projeto. Também, os relacionamentos entre objetivos do sistema e tarefas podem contribuir para entender como os objetivos do sistema são considerados no decorrer do projeto para serem satisfeitos. Porém, sabemos que existem outros fatores da gerência de projeto que podem contribuir para o sucesso ou o fracasso de um projeto, mas eles estão fora do contexto deste trabalho.

Finalmente, as classes *Tarefa*, *ObjetivoSistema* (objetivos do sistema) e *Pessoa* do modelo intermediário foram incluídos como classes candidatas para o modelo de rastreamento do sistema de controle de locadora. A Tabela 5 apresenta os objetivos do sistema que foram elaborado a partir dos objetivos organizacionais. Os objetivos do sistema guiaram o desenvolvimento do sistema de vídeo locadora. Vale salientar que nossa estratégia de trabalho é identificar e relacionar informações que podem afetar e guiar o desenvolvimento de um sistema.

Codigo	Descrição
[OBS-1]	Implementar um módulo para os clientes acessar as informações dos vídeos disponíveis na loja
[OBS-2]	Implementar um cartão ponto no sistema
[OBS-3]	Ter um cadastro de todos os clientes da loja
[OBS-4]	Ter um cadastro de todos as fitas e filmes da loja
[OBS-5]	Gerar vários relatórios que ajudem na compra de materiais para loja
[OBS-6]	Gerar carteiras dos clientes
[OBS-7]	Ter e imprimir um contrato de responsabilidade do cliente com a loja
[OBO-8]	Ter um cadastro de todos os funcionários da loja

Tabela 5: Objetivos do sistema de controle de locadora

A Figura 46 apresenta algumas das tarefas do cronograma de trabalho do desenvolvimento do sistema.

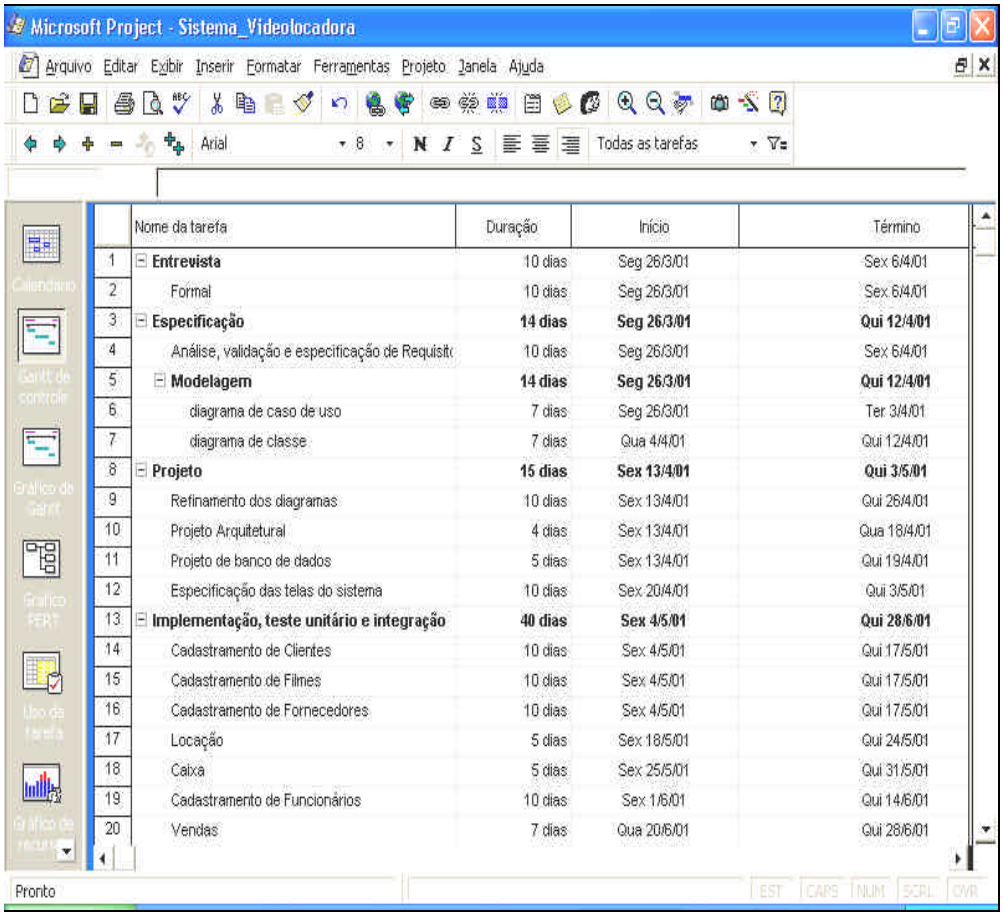


Figura 46: Tarefas para desenvolver o sistema de controle de locadora

A título de exemplo, a ferramenta Project 98, da Microsoft, foi usada para especificar as tarefas.

A Tabela 6.a identifica algumas pessoas que participaram do projeto, enquanto que a Tabela 6.b apresenta os papéis definidos no desenvolvimento.

Pessoa		
	Identificação	Código Papel
[PES-1]	Luciano Lopez	A, P, G
[PES-2]	Salete Samburgo	A, P
[PES-3]	Danielle Chrusciak	A, P

a)

Papel	
Código	Descrição
A	Analista
P	Programador
G	Gerente de projeto

b)

Tabela 6: Tabelas de pessoas e papéis usados no rastreamento de requisitos

A seguir vamos identificar alguns classes de informação no nível de informação do desenvolvimento.

4.6.5 Identificar Classes no Nível de Informação do Desenvolvimento

O objetivo desta subseção é identificar classes no nível de informação de desenvolvimento.

Diretriz 4: Identificar subsistemas. O objetivo dessa diretriz é identificar os subsistemas que compõem o sistema desenvolvido. Se os requisitos foram alocados para os subsistemas existentes, então deve ser incluída a classe *Subsistema*, como o sugerido no modelo intermediário de rastreamento.

O resultado da aplicação da diretriz foi a inclusão da classe *Subsistema* no modelo de rastreamento. A Tabela 7 codifica, identifica e descreve os subsistemas.

Subsistemas do Sistema Locadora de Vídeo		
Código	Identificação	Descrição
[SUB-1]	Gerência de cadastros	Manipulação das informações dos clientes, funcionários, fornecedores e filmes
[SUB-2]	Gerência de movimentos	Controla as informações das locações, compra e vendas
[SUB-3]	Gerência de controle financeiro	Controle do caixa
[SUB-4]	Gerência de configurações	Controla os indexadores, boletos, elenco e oscar
[SUB-5]	Gerência de utilitários	Calculadora, calendário e backup

Tabela 7: Subsistemas do sistema de controle de locadora

A seguir apresentaremos uma diretriz que recomenda a inclusão da classe *Requisito* do modelo de rastreamento.

Diretriz 5: Identificar requisitos. Apesar de óbvio, essa diretriz indica que é obrigatória a inclusão da classe *Requisito* do modelo intermediário porque trabalhar sobre o rastreamento de requisitos não faz sentido sem fazer referência aos próprios requisitos.

O resultado da aplicação da diretriz foi a inclusão da classe *Requisito*. Em lugar de apresentar uma única tabela com todos os requisitos do sistema, decidimos apresentar alguns deles que serão acompanhados com uma figura do sistema para facilitar um maior e melhor entendimento.

A apresentação dos requisitos será feita em várias tabelas que serão acompanhadas de figuras que ilustram sua implementação no sistema.

A Tabela 8 apresenta alguns requisitos relacionados com a gerência de funcionários do sistema de controle de locadora.

[REQ-6]	O sistema deverá permitir ao supervisor a inclusão de funcionários com as seguintes informações do sistema: código, data e hora de inclusão, senha, supervisor(sim/não), nome, foto, endereço, complemento, bairro, cidade, estado, CEP, telefone, celular, E-mail, sexo, data de nascimento, naturalidade, estado civil, grau de instrução, CPF, CGC, RG, órgão expedidor, referências, outras locadoras, nome do pai, nome da mãe, local de trabalho, profissão, cargo, tempo de trabalho, endereço de trabalho, bairro de trabalho, cidade de trabalho, cidade de trabalho, CEP do lugar de trabalho, telefone de trabalho, ramal de trabalho. As informações requeridas do cônjuge são: nome do cônjuge, data de nascimento do cônjuge, celular cônjuge, local de trabalho, profissão do cônjuge, cargo do cônjuge, tempo de trabalho do cônjuge, endereço de trabalho do cônjuge, bairro de trabalho do cônjuge, cidade de trabalho do cônjuge, cidade de trabalho do cônjuge, CEP do lugar de trabalho do cônjuge, telefone de trabalho do cônjuge, ramal de trabalho do cônjuge, e observação.
[REQ-7]	O sistema deverá permitir ao supervisor a exclusão de funcionários do sistema
[REQ-8]	O sistema deverá permitir ao supervisor consultar as informações dos funcionários por código, nome, ou parte do nome. O sistema apresentará todas as informações do funcionário e não permitirá editar as informações.
[REQ-9]	O sistema deverá permitir ao supervisor modificar as informações dos funcionários, com exceção do código.
[REQ-50]	O sistema deverá permitir ao funcionário registrar a hora de entrada e saída do trabalho na loja
[REQ-51]	O sistema deverá gerar um relatório das horas extras trabalhadas pelos funcionários

Tabela 8: Requisitos relacionados com a gerência de funcionários

A título de exemplo, Figura 47 é a tela do sistema de vídeo locadora que permite modificar as informações dos funcionários.

Figura 47: Cadastramento dos funcionários

A Figura 47 apresenta a implementação do requisito “[REQ-9]”.

Na Tabela 9 apresentamos alguns requisitos relacionados com a gerência de filmes.

[REQ-11]	O sistema deverá permitir ao funcionário a inclusão de filmes com os seguintes campos de informações: código do file, título, gênero, cor (colorido ou pret o e branco), tradução (legendado ou dublado), título original, fornecedor, distribuidor, produtora, selo, valor, duração em minutos, tema, tipo (VHF ou DVD), autor, diretor, Oscar (categorias), elenco (ator e atriz), sinopse, trailer, e observações.
[REQ-12] [REQ-14]	O sistema deverá permitir somente ao supervisor a exclusão dos filmes. O sistema não deverá excluir um filme que esteja locado. O sistema deverá permitir somente ao funcionário a consulta dos filmes da loja pelo código, título ou parte do título.
[REQ-15]	O sistema deverá permitir a modificação dos dados do filme. O código não filme não pode ser mudado.

Tabela 9: Requisitos relacionados com a gerência de filmes

A Figura 48 é a tela que representa a implementação do requisito de alteração dos dados do filme ([REQ-15]).

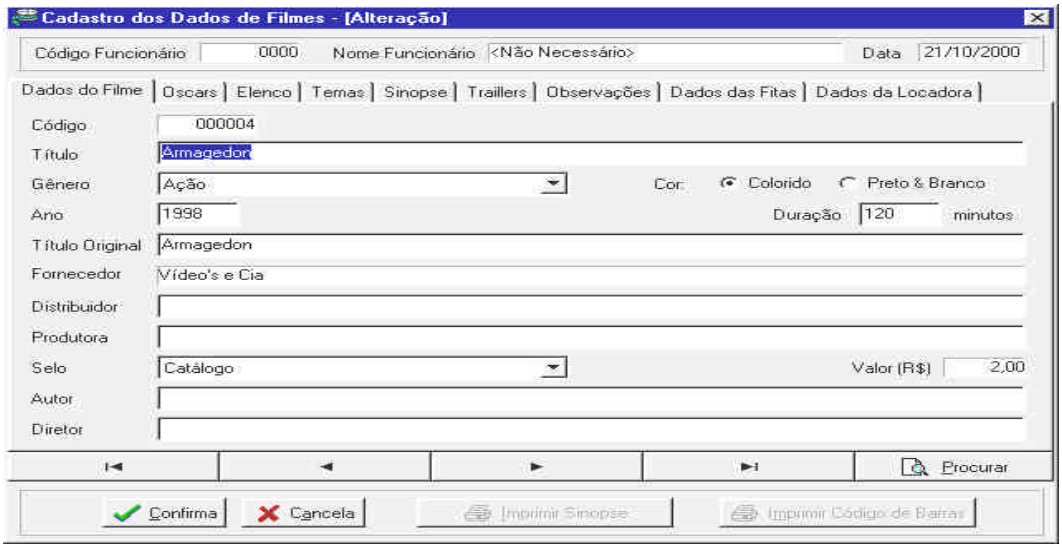


Figura 48: Alteração dos dados do filme

A Tabela 10 apresenta alguns requisitos relacionados com a gerência de clientes.

[REQ-45]	O sistema deverá permitir ao funcionário incluir clientes. As informações que o sistema deverá gerenciar do cliente são as seguintes: código, data e hora da inclusão, nome, foto, endereço, complemento, bairro, cidade, estado, CEP, telefone, celular, E-mail, sexo, data de nascimento, naturalidade, estado civil, grau de instrução, CPF, CGC, RG, órgão expedidor, referências, outras locadoras, nome do pai, nome da mãe, local de trabalho, profissão, cargo, tempo de trabalho, endereço do trabalho, bairro do trabalho, cidade do trabalho, CEP do lugar de trabalho, telefone do trabalho e ramal. As informações requeridas do cônjuge são: nome, data de nascimento, celular, local de trabalho, profissão, cargo, tempo de serviço, endereço, bairro do trabalho, cidade do trabalho, CEP do lugar de trabalho, telefone do trabalho, ramal, e observação.
[REQ-46]	O sistema deverá permitir ao funcionário alterar as informações do cliente, com exceção do código de identificação do cliente.
[REQ-47]	O sistema deverá permitir aos funcionários consultar as informações dos clientes por um dos seguintes campos de informação: código, nome, e parte do nome. Todas as informações dos clientes deverão ser apresentadas.

Tabela 10: Requisitos relacionados com a gerência de clientes

A título de exemplo, a Figura 49 apresenta a tela do sistema para fazer a alteração das informações dos clientes (veja requisito "[REQ-46]" na Tabela 10).

Figura 49: Cadastro de clientes

A seguir aplicaremos outras diretrizes recomendadas pelo processo para identificar classes para desenvolver um modelo de rastreamento.

Diretriz 6: Identificar diagramas. Os objetivos desta diretriz são identificar os tipos de diagramas nos quais foram ou serão representados os requisitos do sistema e definir os caminhos lógicos para acessar as representações dos requisitos nos diferentes tipos de diagrama. O termo "caminho lógico" foi apresentado na Subseção 3.3.8. Se estabelecemos relacionamentos e definimos caminhos lógicos, então poderemos verificar com maior facilidade se todos os requisitos do sistema foram modelados nos diagramas.

O resultado da aplicação da diretriz foi a inclusão da classe *Diagrama* para registrar e representar todos os diagramas que serão usados no rastreamento de requisitos e a definição de caminhos lógicos para acessar os requisitos modelados nos diagramas.

A Tabela 11 apresenta a definição dos caminhos lógicos usados no rastreamento do sistema de controle de locadora. Nos caminhos lógicos, o símbolo ponto e vírgula (";") separa informações de diferentes tipos, por exemplo, atributo e operação. Já o símbolo vírgula (",") é usado para separar informações do mesmo tipo, por exemplo, uma lista de atributos. Podemos dizer que a definição da classe *Diagrama* pode ter os atributos: nome do diagrama e tipo de diagrama.

Caminhos lógicos para o diagrama de classe	
Tipo referência	Caminho lógico
de requisito para classe.	"C:nome_classe", onde "C" é uma abreviação para classe e nome_classe é nome de uma classe.
de requisito para um atributo da classe	"C: nome_classe; A: nome_atributo$_1$,...,nome_atributo$_n$", onde "A" representa atributos da classe. A expressão "nome_atributo$_1$,...,nome_atributo$_n$" é uma lista dos atributos nos quais o requisito foi modelado.
de requisito para uma método da classe.	"C:nome_classe; O:nome_operação$_1$,...,nome_atributo$_n$" , onde "O" é uma abreviação para operação e a expressão "nome_operação$_1$,...,nome_atributo$_n$" é o nome da operação que implementa o requisito.
Caminho lógico para diagrama de caso de uso	
Tipo referência	Caminho lógico
De requisito para um caso de uso	"CA: nome_casoUso", onde CA é uma abreviação para caso de uso e "nome_casoUso" é o nome do caso de uso que modela o requisito do sistema.

Tabela 11: Definição de caminhos lógicos rastrear requisitos para diagramas

A Tabela 12 apresenta a codificação de alguns diagramas da UML usados na modelagem do sistema vídeo locadora. Isso significa que um atributo deverá ser definido sobre a classe diagrama para identificar (através de código) as instâncias dos diferentes diagramas.

Código	Tipo de diagrama
1	Diagrama de Classe
2	Diagrama de Caso de uso
3	Diagrama de Atividade
4	Diagrama de Componentes
5	Diagrama de implantação

Tabela 12: Condificação de alguns diagramas da UML

O objetivo da codificação é facilitar a identificação dos diferentes tipos de diagrama elaborados. A Tabela 13 apresenta alguns nomes e tipos de diagramas desenvolvidos para o sistema. A tabela está dividida em três colunas. A coluna *código* representa a identificação dos diagramas. A coluna *nome lógico do diagrama*. A coluna *tipo de diagrama* codifica os diferentes tipos de diagrama (veja a Tabela 13).

Código	Nome lógico do Diagrama	Tipo diagrama
[DGM-1]	AberturaCaixa	1
[DGM –2]	CartãoPonto	1
[DGM –3]	filmografia_filmes	1
[DGM –4]	gerencia_filmes	1
[DGM –5]	configuração_Sistema	1
[DGM –6]	gerencia_clientes	1
[DGM –7]	gerencia_Cliente	1
[DGM –8]	GerenciaFilmes	1
[DGM –9]	movimentacão_do_funcionário	1
[DGM –50]	Utilitário	2
[DGM –51]	TerminalCliente	2
[DGM –52]	CadastroClientes	2
[DGM –53]	CadastroFilmes	2
[DGM –54]	CadastroFuncionários	2

Tabela 13: Diagramas do Sistema de Vídeo de Locadora

A Figura 50 apresenta um diagrama de caso de uso no qual o caso de uso *cartão ponto* representa o requisito "[REQ-50] O sistema deverá permitir ao funcionário registrar a hora de entrada e saída do trabalho na loja", da Tabela 8. O caso de uso "Sobre o sistema" especifica que o sistema deve fornecer informações sobre o mesmo (desenvolvedores, versão, ano, etc).

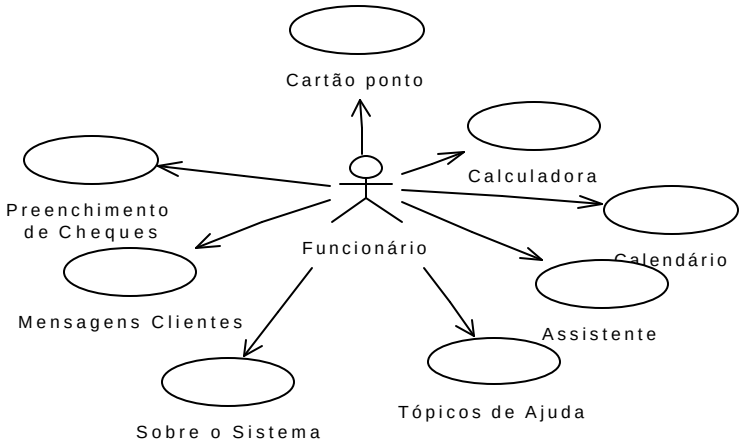


Figura 50: Algumas funcionalidades disponíveis para o funcionário

A Figura 51 apresenta um diagrama de classe que representa os funcionários que operam o sistema de controle de locadora. A classe *Funcionário* possui todos os atributos definidos no requisito "[REQ-6] O sistema deverá permitir ao supervisor a inclusão de funcionários com as seguintes informações do sistema: código, data e hora de inclusão, senha, supervisor(sim/não), nome, foto, endereço,...", da Tabela 8.

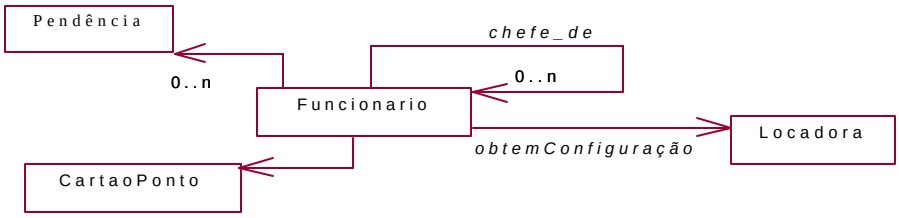


Figura 51: Diagrama de classe para representar funcionários e cartão ponto

A Figura 52 apresenta um diagrama de classe que modela o cliente do sistema. A classe *Cliente* possui todos os atributos especificados no requisito "[REQ-45] O sistema deverá permitir ao funcionário incluir clientes. As informações que o sistema deverá gerenciar do cliente são as seguintes: código, data e hora da inclusão, nome, foto, endereço, complemento, bairro, cidade, estado, CEP, telefone, celular, E-mail, sexo,... ", da Tabela 10.

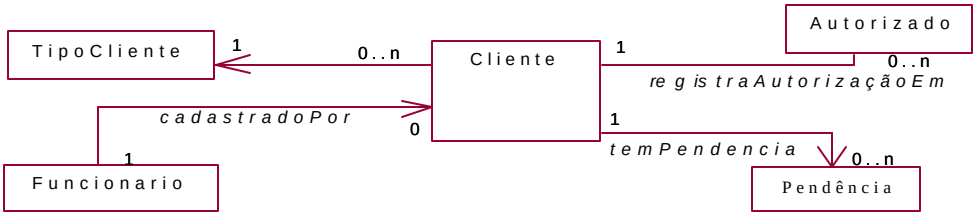


Figura 52: Diagrama de classe sobre a classe cliente

Finalmente, a Figura 53 inclui a modelagem do requisito "[REQ-11] O sistema deverá permitir ao funcionário a inclusão de filmes com os seguintes campos de informações: código do file, título, gênero, cor (colorido ou preto e branco), tradução (legendado ou dublado), título original, fornecedor, distribuidor, produtora, selo, valor, duração em minutos, tema, tipo (VHF ou DVD), autor, diretor,...", da Tabela 9.

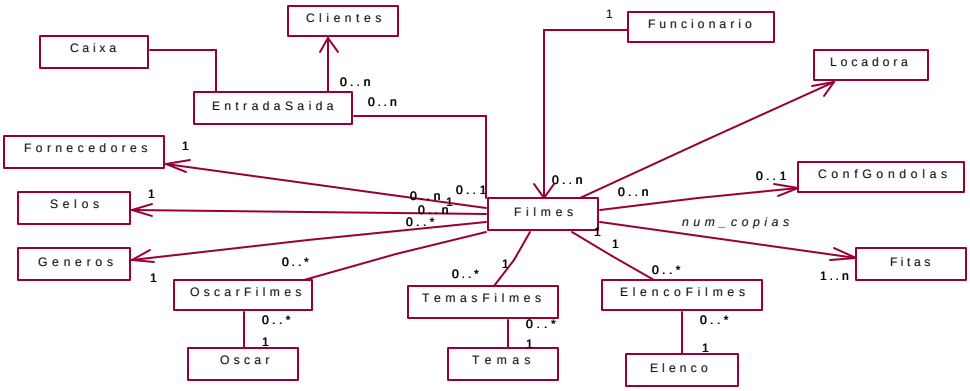


Figura 53: Diagrama de classe sobre filmes

A seguir continuamos apresentando outras diretrizes para identificar classes candidatas para o modelo de rastreamento.

Diretriz 7: Identificar programas. O objetivo dessa diretriz é identificar os programas nos quais foram representados os requisitos do sistema e definir um caminho lógico para acessar a representação dos requisitos nos programas. A Tabela 14 apresenta o caminho lógico usado no processo de rastreamento do Sistema de controle de locadora.

Caminho lógico para relacionar requisitos com programas	
Tipo referência	Caminho lógico
De requisito para *unit*	"U:nome_unidade", onde "U" é uma abreviação para unidade onde foi implemento o requisito
De requisito para um procedimento.	"C:nome_unidade; P:nome_procedimento$_1$,.....,nome_ procedimento $_n$" , onde "P" é uma abreviação para procedimento principal que implementa o requisito.

Tabela 14: Definição de caminhos lógicos para rastrear requisitos em programas

Na Tabela 14, a para *unit* representa um módulo no ambiente de programação Delphi.

A Tabela 15 apresenta alguns dos programas desenvolvidos para o sistema. Os programas foram implementados no ambiente de programação *Delphi*.

Nomes de programas que implementam os requisitos do Sistema de controle de locadora	
[PRG-1]	AberturaCaixa
[PRG-3]	AdicionaFita
[PRG-4]	AlteraCartaoPonto
[PRG-5]	AlteraHoraCartao
[PRG-7]	ArquivoMorto
[PRG-8]	Autorizados
[PRG-12]	CartaoPonto
[PRG-14]	DadosFilmes
[PRG-15]	Filmografia
[PRG-19]	Funcionarios

Tabela 15: Nomes de programas do sistema de controle de locadora

A Figura 54 apresenta outros nomes de programas desenvolvidos para o sistema.

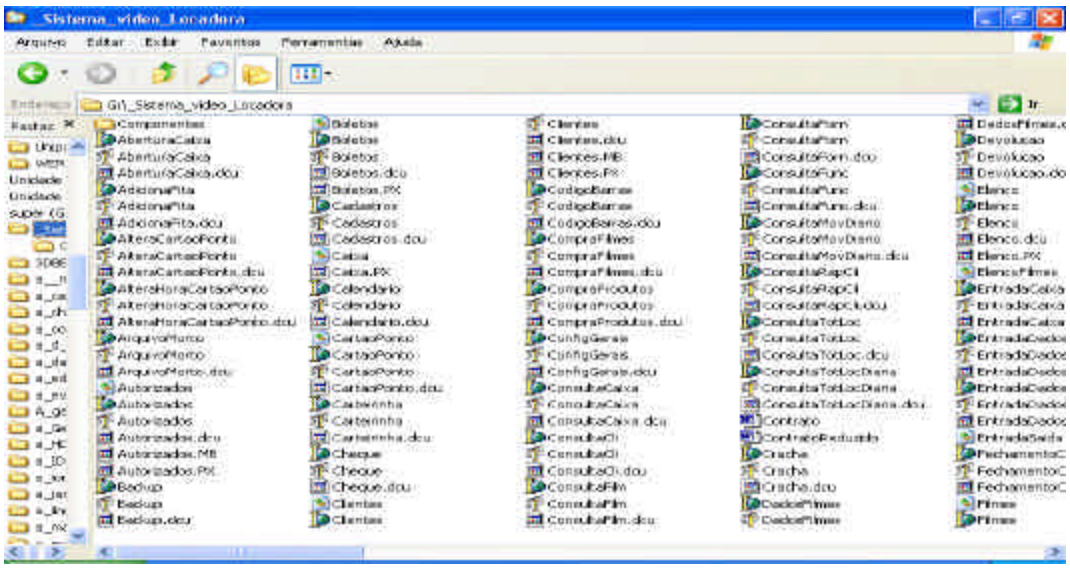


Figura 54: Visão parcial os programas do sistema de controle de locadora

Na Figura 55, a seta pontilhada identifica o nome unidade de programação *CartaoPonto* (programa [PRG-12] da Tabela 15) que implementa o cartão ponto. Além disso, é apresentada a tela do sistema para registrar a hora de entrada/saída do sistema.

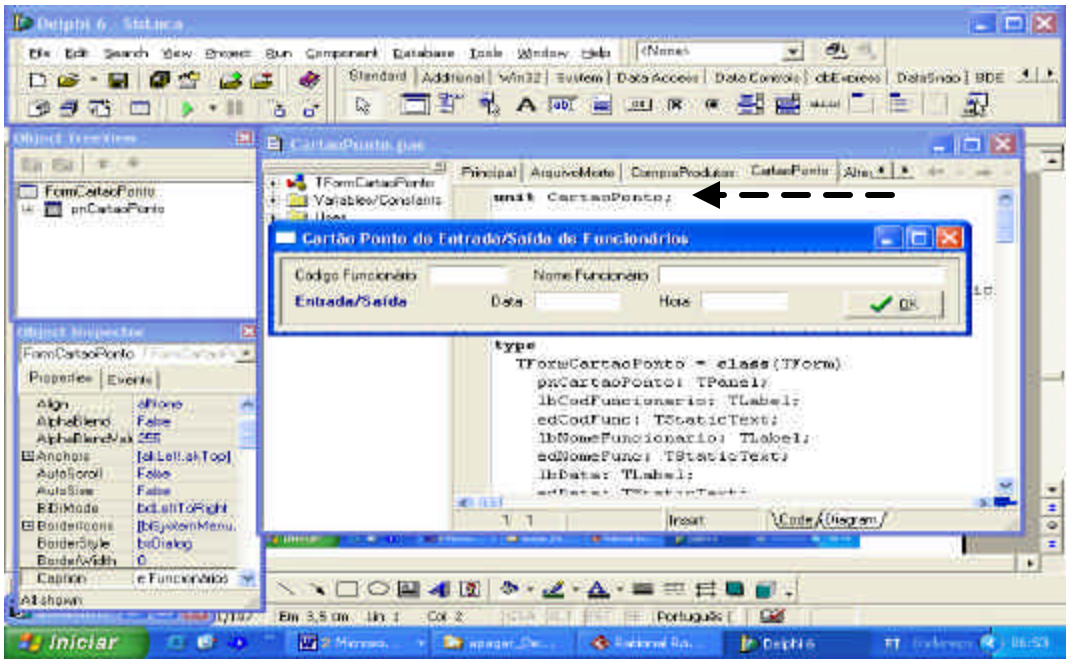


Figura 55: Visão da implementação do cartão ponto

A Figura 56 apresenta um trecho da implementação do requisito “[REQ-51] O sistema deverá gerar um relatório das horas extras trabalhadas pelos funcionários” , da Tabela 8 .

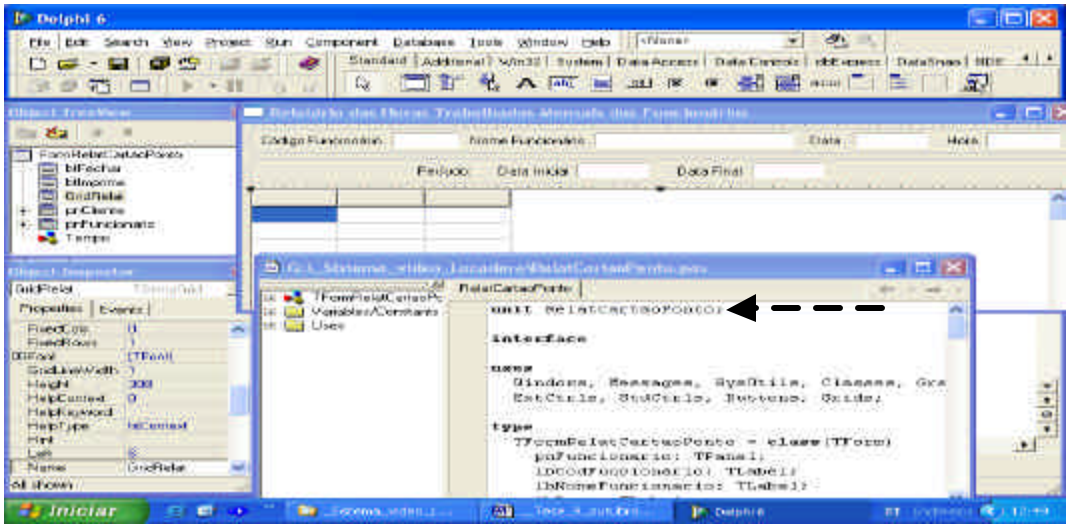


Figura 56: Relatório das horas mensais trabalhadas pelos funcionários

Diretriz 8: Identificar teste. O objetivo desta diretriz é identificar documentos ou fragmentos de documentos que especificam como os requisitos serão testados.

O sistema de vídeo locadora foi testado, mas seus testes não foram documentados. Portanto, o modelo de rastreamento do sistema de vídeo locadora não possui uma classe para representar os testes do sistema.

No que diz respeito a testes, podemos dizer que os testes unitários foram inicialmente realizados durante e no final do desenvolvimento dos programas. Posteriormente foram realizados testes de integração para checar se dois programas conseguiam trabalhar juntos. Finalmente, o desenvolvimento concluiu com os testes do sistema para checar todos os programas ao mesmo tempo.

Na próxima subseção organizaremos as classes das diferentes fases.

4.6.6 Organizar e Estruturar as Classes Candidatas

O resultado dos passos das diretrizes anteriores é um conjunto de classes candidatas. As classes devem ser revisadas e organizadas seguindo as diretrizes seguintes. É importante salientar que as próximas três diretrizes não devem ser aplicadas necessariamente em forma seqüencial.

Diretriz 9: Remover as classes de informação irrelevantes. O objetivo da diretriz é alertar a necessidade de remover as classes candidatas que são irrelevantes para o modelo de rastreamento. Para isso, é necessário fazer uma lista de perguntas, tais como:

1. Qual é o objetivo da classe candidata no modelo de rastreamento? Se não obtemos uma resposta convincente, então é recomendável avaliar a inclusão da classe.
2. Qual é a contribuição da classe no modelo de rastreamento? O objetivo da pergunta é verificar se a classe contribui para responder questões importantes sobre o rastreamento de requisitos para o projeto.
3. A classe candidata representa informações disponíveis para o rastreamento? O objetivo da pergunta é identificar e remover classes de informações que não estão disponíveis para serem rastreadas. Não se deve remover uma classe que representa informações parcialmente disponíveis.

Diretriz 10: Integrar as classes com o mesmo significado. A análise do siginificado e responsabilidade das classes candidatas pode ajudar a identificar classes com nomes diferentes, mas com o mesmo significado. Por exemplo, as classes ObjetivoSistema e ObjetivoOrganizacional tem diferentes nomes, mas a essência delas é a mesma. Mas, mesmo assim, decidimos trata-lás como informações diferentes.

Diretriz 11: Integrar novas classes. Para integrar novas classes se deve seguir e aplicar as diretrizes 9 e 10.

O resultado das aplicações das diretrizes 9, 10 e 11 é um conjunto de classes representativas dos elementos/artefatos que desejamos. A Tabela 16 apresenta as classes candidatas a serem rastreadas no sistema de vídeo locadora.

Níveis de informação	Análise	Projeto	Implementação
Externo			
Organizacional Gerencial	Objetivo organizacional Objetivo do sistema Tarefa, Pesssoa	Tarefa, Pesssoa	Tarefa, Pesssoa
Desenvolvimento	Requisito	Diagrama, Subsistema	Programa

Tabela 16: Lista de classes candidatas

A seguir estabelecemos relacionamentos entre as classes.

4.6.7 Estabelecer Relacionamentos

Nesta subseção forneceremos diretrizes para estabelecer relacionamentos entre as classes candidatas.

Diretriz 12: Organizar as classes. O objetivo dessa diretriz é organizar e estruturar as classes candidatas usando generalização. O modelo de rastreamento do sistema de vídeo locadora não uso a generalização.

Diretriz 13: Estabelecer relacionamentos. Para relacionar as classes candidatas existem duas alternativas:

1. A classe candidata existe no modelo intermediário. Nesse caso, o usuário deverá verificar se alguns dos relacionamentos propostos no modelo intermediário podem ser aplicados sobre a classe em questão. Caso contrário, será necessário conhecer o objetivo ou responsabilidade da classe para associá-la.
2. A classe candidata não existe no modelo intermedário. Nesse caso, o usuário deve conhecer o objetivo ou responsabilidade da classe para identificar outras classes com as quais a classe candidata deve relacionar-se.

Repita as alternativas da diretriz 13, quantas vezes for necessário, para identificar outras classes e estabelecer outros relacionamentos.

A Figura 57 apresenta o modelo de rastreamento para o sistema.

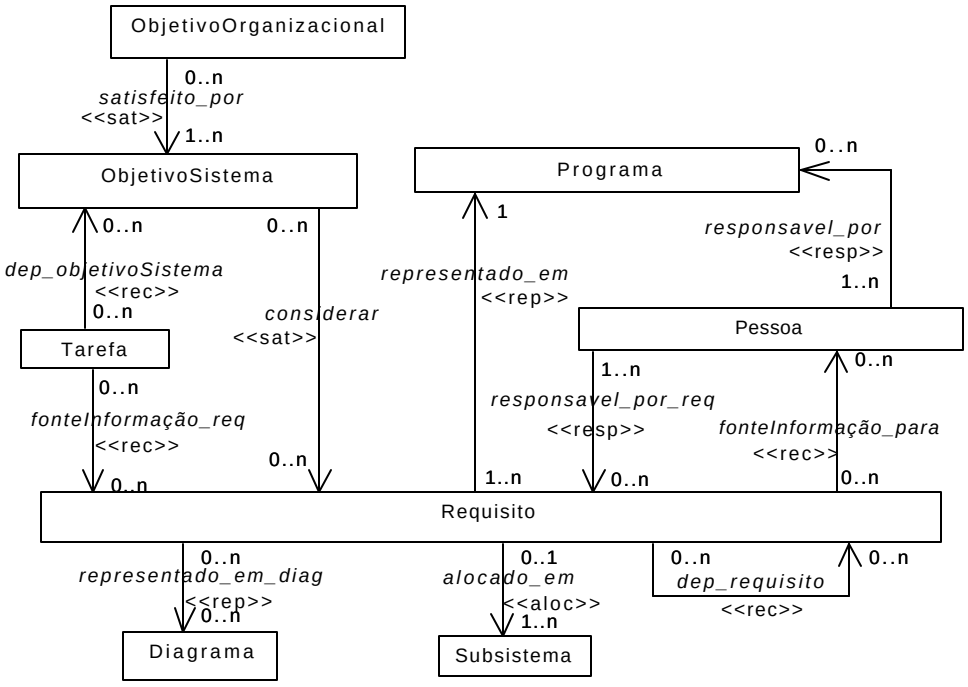


Figura 57: Modelo de rastreamento do sistema de controle de locadora

As classes candidatas foram organizadas conforme as duas últimas diretrizes.

Após aplicar as primeiras treze diretrizes do processo, podemos concluir que:

1. O processo apresenta uma forma sistemática e simples para identificar, estruturar e relacionar classes;
2. O modelo intermediário usado no processo facilita a identificação de várias classes que compõem o modelo de rastreamento do sistema de controle de locadora;
3. Podemos afirmar que o modelo de rastreamento do sistema de controle de locadora possui vários das classes e associações especificadas no nosso modelo intermediário de rastreamento.

Em geral, as diretrizes podem ser automatizadas, porém, existem algumas diretrizes, por exemplo 9 e 10, que requerem da análise e experiência da pessoa encarregada do rastreamento de requisitos. A seguir recomendamos alguns atributos para as classes modelada s.

4.6.8 Identificar as Propriedades sobre as Classes

O objetivo desta subseção é fornecer uma diretriz para identificar os atributos das classes do modelo de rastreamento. Na aplicação dessa diretriz, o usuário deve considerar a hierarquia de classes, se for necessário.

Diretriz 14: Recomendar atributos sobre as classes.

1. Considere os seguintes atributos sobre a classe requisito.
   a) *Prioridade*. Relevância da instância da classe. Isso ajuda ao gerente de projeto a planejar as versões do software que conterão determinados requisitos;
   b) *Subsistema*. Subsistema onde foi alocado o requisito;
   c) *Data de criação*. Data de criação da instância.
   d) *Fonte de informação*. Referência à fonte de informação.
2. Considere os seguintes atributos sobre as classes Diagrama e/ou Programa.
   a) *Identificação*. Rótulo de identificação das instâncias das classes, segundo a Subseção 4.6.1 e o exemplo da Figura 45;
   b) *Tipo*. Para diferenciar o tipo de diagrama (caso de uso ou de classe) ou programa Delphi (unidade ou parte gráfica do programa do formulário).

Esses atributos complementam os outros atributos da classe *Requisito*, do modelo intermediário, que foram pre-definidos na meta-classe *Classe* do meta-modelo de rastreamento apresentado na Seção 3.3. Esse exemplo é suficiente para iniciar a identificação das propriedades das classes do modelo de rastreamento do sistema de vídeo locadora. A seguir fornerecemos algumas recomendações para gerar algumas matrizes de rastreamento.

4.6.9 Definir e Preencher as Matrizes de Rastreamento

A subseção apresenta uma diretriz que trata a questão da representação matricial dos relacionamentos que compõem o modelo de rastreamento do sistema. A seguir apresentamos a diretriz.

Diretriz 15: Definir uma matriz para cada relacionamento do modelo. O objetivo desta diretriz é desenvolver várias matrizes ou listas para representar os relacionamentos do modelo de rastreamento.

A seguir identificamos os relacionamentos cuja representação matricial será apresentada.

1. *satisfeito_por*, que associa as classes *ObjetivoOrganizacional* e *ObjetivoSistema*.
2. *considerar*, que associa as classes *ObjetivoSistema* e *Requisito*.
3. *representado_em_diag*, que associa as classes *Requisito* e *Diagrama*.
4. *representado_em*, que associa as classes *Requisito* e *Programa*.
5. *fonteInformação_req*, que associa as classes *Tarefa* e *Requisito*.

A Figura 58 apresenta uma representação matricial do relacionamento *Satisfeito_por* do modelo de rastreamento (Figura 57). A Figura 58 inclui alguns pares de instâncias relacionadas das classes *ObjetivoOrganizacional* e *ObjetivoSistema* do modelo de rastreamento. As expressões contidas na matriz da Figura 58 expressam que todos os objetivos organizacionais têm uma alta dependência de satisfação com os objetivos do sistema, e que todas as dependências dos objetivos organizacionais devem ser satisfeitas.

satisfeito:<<sat>> →	[OBS-1] Implementar um módulo para os clientes acessar as informações dos vídeos...	[OBS-2] Implementar um cartão ponto no sistema	[OBS-3] Ter um cadastro de todos os clientes da loja	[OBS-4] Ter um cadastro de todas as fitas e filmes da loja	[OBS-5] Gerar vários relatórios que ajudem na compra de materiais para loja	[OBS-6] Gerar carteiras dos clientes	[OBO-8] Ter um cadastro de todos os funcionários da loja e
[OBO-1] Facilitar aos clientes o acesso às informações sobre...	<A, A>						
[OBO-2] Controlar a entrada e saída dos funcionários		<A, A>					
[OBO-3] Gerenciar as informações dos clientes			<A,A>			<A,A>	
[OBO-4] Gerenciar as informações dos produtos...							
[OBO-5] Gerar relatórios para tomada de decisão...				<A,A>	<A,A>		

Figura 58: Representação matricial do relacionamento *satisfeito_por*

A título de exemplo, a satisfação do objetivo organizacional "[OBO-2] Controlar a entrada e saída dos funcionários" depende da satisfação do objetivo do sistema "[OBS-2] implementar um cartão ponto no sistema". De forma similar podemos dizer que a satisfação do objetivo organizacional "[OBO-5] gerar relatórios para a tomada de decisão..." depende que os dois objetivos do sistema ([OBS-4] e [OBS-5]) sejam satisfeitos.

A Figura 59 é a representação matricial do relacionamento *Considerar*. A matriz mostra as associações de algumas instâncias das classes *ObjetivoSistema* e *Requisito* do modelo de rastreamento. O relacionamento expressa que a satisfação dos objetivos do sistema depende dos requisitos sejam sejam implementados de forma correta.

considerar: <<sat>> ←	[OBS-2] Implementar um cartão ponto no sistema	[OBS-3] Ter um cadastro de todos os clientes da loja	[OBS-4] Ter um cadastro de todas as fitas e filmes da loja	[OBO-8]Ter um cadastro de todos os funcionários da loja
[REQ-6] O sistema deverá permitir ao supervisor a inclusão de funcionários ...				<A,A>
[REQ-7] O sistema deverá permitir ao supervisor a exclusão de funcionários ...				<A,A>
[REQ-8] O sistema deverá permitir ao supervisor consultar as informações dos funcionários				<A,A>
[REQ-50] O sistema deverá permitir ao funcionário registrar a hora de entrada..	<A,A>			
[REQ-51] O sistema deverá gerar um relatório das horas extras trabalhadas..	<A,A>			

Figura 59: Representação matricial do relacionamento *Considerar*

A representação matricial do relacionamento *Representado_em_diag* é apresentado na Figura 60.

representado_em_diag: <<rep>> →	[DGM –54] (Figura 50)	[DGM –2] (A Figura 51)
[REQ-6] O sistema deverá permitir ao supervisor a inclusão de funcionários da seguinte informação do sistema: código...		<rep, C:Funcionario>
[REQ-7] O sistema deverá permitir ao supervisor a exclusão de funcionários ...		<rep, C:Funcionario>
[REQ-8] O sistema deverá permitir ao supervisor consultar as informações dos funcionários		<rep, C:Funcionario>
[REQ-50] O sistema deverá permitir ao funcionário registrar a hora de entrada e saída...	<rep, CA:cartãoPonto>	<rep, C:CartãoPonto>
[REQ-51] O sistema deverá gerar um relatório das horas extras trabalhadas pelos funcionários		<rep, C:Funcionario>

Figura 60: Representação matricial do relacionamento *representado_em_diag*

Na Figura 60 a expressão "<rep; CA:cartãoPonto >" significa que o requisito "[REQ-50]" foi modelado em um diagrama de caso de uso (veja em [DGM-2] da Tabela 13), especificamente, no caso de uso chamado *cartãoPonto* (veja Figura 50).

Na Figura 61 apresenta a matriz para o relacionamento *representado_em* do tipo representação. Na matriz, a expressão "<imp, U: cartaoPonto>" expressa que o requisito "[REQ-50]" foi implementado ("imp") na unidade *CartaoPonto*.

representado_em: <<rep>> →	[PRG-12]	[PRG –19]
[REQ-6] O sistema deverá permitir ao supervisor a inclusão de funcionários da seguinte informação do sistema: código, data e hora de inclusão...		<imp, U:Funcionario>
[REQ-7] O sistema deverá permitir ao supervisor a exclusão de funcionários do sistema		<imp, U:Funcionario>
[REQ-8] O sistema deverá permitir ao supervisor consultar as informações dos funcionários por código, nome....		<imp, U:Funcionario>
[REQ-50] O sistema deverá permitir ao funcionário registrar a hora de entrada e saída do trabalho na loja	<imp, U: CartaoPonto>	
[REQ-51] O sistema deverá gerar um relatório das horas extras trabalhadas pelos funcionários		<imp, U:Funcionario>

Figura 61: Representação matricial do relacionamento *representado_em*

A Figura 62 apresenta uma forma alternativa para representar o relacionamento *representado_em* da Figura 61. A Figura 62 reduz o número de colunas da matriz, porém, não é fácil identificar todos os requisitos implementados em um programa.

representado_em: <<rep>> →	Programa
Requisito	
[REQ-6]O sistema deverá permitir ao supervisor a inclusão de funcionários da seguinte informação do sistema: código, data e hora de inclusão....	<imp, U:Funcionario>
[REQ-7]O sistema deverá permitir ao supervisor a exclusão de funcionários do sistema	<imp, U:Funcionario>
[REQ-8]O sistema deverá permitir ao supervisor consultar as informações dos funcionários por código, nome...	<imp, U:Funcionario>
[REQ-50] O sistema deverá permitir ao funcionário registrar a hora de entrada e saída do trabalho na loja	<imp, U: CartaoPonto>
[REQ-51]O sistema deverá gerar um relatório das horas extras trabalhadas pelos funcionários	<imp, U:Funcionario>

Figura 62: Forma alternativa para representar o relacionamento *representado_em*

O desenvolvimento e registro das informações contidas nas diferentes matrizes ajudarão a manter a consistência dos artafatos relacionados. Porém, é importante salientar que o trabalho manual é cansativo e requer a dedicação de uma pessoa.

A Figura 63 apresenta o cronograma de trabalho do Sistema de controle de locadora. Na coluna direita da mesma figura estão alguns dos requisitos implementados no sistema.

Conforme indicado no Capítulo 1, no decorrer da tese, foram realizados cinco estudos de casos: Sistema de Contabilidade, Sistema de Biblioteca (apresentado Capítulo 5), Sistema de controle de locadora, Sistema de condomínio, Sistema de Produção e Custo de Leite, e Sistema de Avaliação de Consulta. Os resultados dos estudos de casos não apresentados neste tese serão gerados com relatórios técnicos do Depto. De Informática da UNIOESTE. No rastreamento de requisitos de quatro sistemas foram relacionados as tarefas com os requisitos e os resultados obtidos pelos desenvolvedores podem ser resumidos como segue:

1. O fato de relacionar tarefas com requisitos contribuiu para que os desenvolvedores pensassem mais no planejamento das atividades que deveriam ser realizadas porque tiveram conhecimento que as tarefas seriam relacionadas com requisitos. Conseqüentemente, aconteceu um maior interesse pela definição dos requisitos;
2. O cronograma de trabalho normal, como o da Figura 46, ajudou outros desenvolvedores a entender somente a organização das tarefas. Porém, um cronograma, como o da Figura 63, permitiu conhecer os requisitos tratados em cada tarefa e compreender melhor o tempo atribuído para o desenvolvimento dos requisitos associados com as tarefas;
3. Foi possível identificar a tarefa que teve mais requisitos a serem desenvolvidos, mas isso não significou necessariamente que a tarefa seja a mais complexa, a mais importante ou a mais trabalhosa;
4. O relacionamento das tarefas com os requisitos lhes permitiu entender a importância de escrever requisitos completos, corretos e não ambíguos para se dimensionar e fundamentar melhor a decisão de atribuir um tempo apropriado para a tarefa que implementaria os requisitos;
5. Um cronograma, como o da Figura 63, permitiu a muitos desenvolvedores (externos a um projeto) entender e compreender melhor a organização das tarefas do um projeto e da distribuição dos requisitos entre as tarefas;

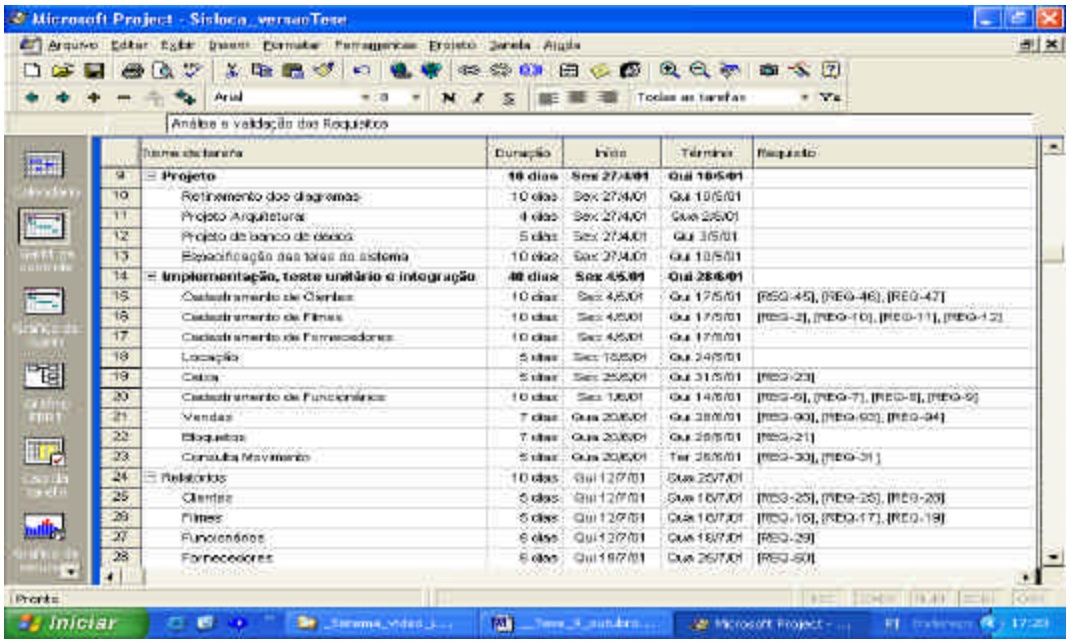


Figura 63: Cronograma de trabalho do sistema de controle de locadora

4.6.10 Validar as Matrizes de Rastreamento

Esta subseção apresenta a realização de uma diretriz que trata a questão da validação matricial dos relacionamentos que compõem o modelo de rastreamento do sistema. A seguir apresentamos a diretriz.

Diretriz 16: Validar o modelo de rastreamento. O objetivo desta diretriz é usar as matrizes preenchidas na Subseção 4.6.9 para validar a importância e as facilidades de se ter um modelo de rastreamento.

Mostraremos inicialmente como usar o modelo de rastreamento para fazer análise de impacto. Considere a seguinte situação na qual a organização que encarregou o desenvolvimento do Sistema de controle de locadora comprou e instalou um aparelho para marcar o cartão ponto. Logo, a organização decide que o objetivo organizacional "[OBO-2] implementar um cartão ponto no sistema" sobre o sistema de controle de locadora não faz mais sentido, e portanto, tal funcionalidade deve ser retirada do sistema. Diante da situação descrita, podemos seguir os seguintes passos:

1. Revisar e identificar no modelo de rastreamento a classe que foi afetada diretamenta pela mudança. Nesse caso a classe afetada é *ObjetivoOrganizacional*,

especificamente, a instância "[OBO-2] Controlar a entrada e saída dos funcionários".

2. Revisando o modelo de rastreamento identificamos que a classe ObjetivoOrganizacional está relacionada com a classe *ObjetivoSistema* através do relacionamento *Satisfeito_por* (Figura 58). Analisando a matriz desse relacionamento foi identificado uma instância do objetivo que expressa: "[OBS-2] Implementar um cartão ponto no sistema ";
3. Revisando o modelo de rastreamento identificamos que a classe *Requisito* está relacionada com a classe *ObjetivoSistema* através do relacionamento *considerar* (Figura 59). Revisando a matriz desse relacionamento conseguimos identificar que os requisitos [REQ-50] e [REQ-51] estão relacionados com o objetivo do sistema [OBS-2]. O requisito [REQ-50] expressa que "o sistema deverá permitir ao funcionário registrar a hora de entrada e saída...", enquanto o requisito [REQ-51] expresa que "o sistema deverá gerar um relatório das horas extras trabalhadas pelos funcionários";
4. Revisando o modelo de rastreamento identificamos que a classe *Diagrama* está relacionada com a classe *Requisito* através do relacionamento *representado_em_diag* (Figura 60). A título de exemplo, a análise da matriz desse relacionamento determinou que o requisito [REQ-50] fosse modelado nos diagramas [DGM-2] e [DGM-54] (Tabela 13), e que o requisito [REQ-51] no diagrama [DGM-2];
5. De forma similar ao item anterior, analisando a matriz do relacionamento *representado_em* (Figura 61) foi identificado de que os requisitos [REQ-50] e [REQ-51] foram implementados nas unidades *cartaoPonto* e *funcionário*, respectivamente.

Em resumo, temos mostrado que o modelo de rastreamento fornece uma visão concreta dos elementos relacionados do sistema. Os relacionamentos fornecem uma maior compreensão de como as classes do modelo de rastreamento estão relacionadas entre si (o significado dos relacionamentos). O modelo pode ser considerado um meio de comunicação para trocar idéias entre os desenvolvedores acerca do rastreamento de requisitos.

4.7 Registrando o Raciocínio dos Problemas

Na Subseção 2.4.3 apresentou-se o modelo IBIS (*Issue Based Information System* [Conklin, 1988]) para registrar os problemas e os raciocínios para se obter uma solução. A título de exemplo, a Figura 64 apresenta um exemplo do registro do raciocínio.

<table>
<tr><td colspan="6">Registro do Raciocínio</td></tr>
<tr><td colspan="2">Identificador: 3</td><td colspan="3">Prioridade: 10</td><td>Data: 27/4/2001</td></tr>
<tr><td colspan="6">Assunto</td></tr>
<tr><td colspan="6">Considerando o volume de informação dos registro de locação, a possibilidade de interligar o sistema com outras futuras loja de locação, e a possibilidade de permitir um acesso pelo internet às reserva e locação de filme, é preciso decidir a compra de um SGBD.(SQL-Server ou Access)</td></tr>
<tr><td colspan="2">Objetivos organizacionais</td><td colspan="2">Objetivos do Sistema</td><td colspan="2">Restrições</td></tr>
<tr><td colspan="2">[OBO-1] Facilitar o acesso dos clientes às informações sobre os vídeos da loja</td><td colspan="2">[OBS-1] Implementar um módulo para os clientes acessar as informações dos vídeos disponíveis na loja</td><td colspan="2">O orçamento da loja para o sistema é pequeno.</td></tr>
<tr><td colspan="6">Posições</td></tr>
<tr><td colspan="3">Apóia</td><td colspan="3">Contrapõe</td></tr>
<tr><td>Argumento</td><td>Fonte</td><td>Responsável</td><td>Argumento</td><td>Fonte</td><td>Responsável</td></tr>
<tr><td>1) Comprar o SQL Server porque é mais robusto e porque a Microsoft fez grande investimento na melhoria desse produto</td><td>internet</td><td>S. Samburgo</td><td>1) A compra do SQL Server aumentaria os custos do sistema e na empresa não tem os recursos financeiros para fazer isso.</td><td>Dono da loja</td><td>Dono da loja</td></tr>
<tr><td>2) Os desenvolvedores tem mais experiência no uso desse produto</td><td>L. Lopez</td><td>L. Lopez</td><td>2) Aproveitar o Access porque ele está disponível e não necessita ser comprado</td><td>Dono da loja</td><td>Dono da loja</td></tr>
<tr><td>3) O desempenho do SQL Server é superior ao Access</td><td>internet</td><td>L. Lopez</td><td></td><td></td><td></td></tr>
<tr><td colspan="6">Decisão</td></tr>
<tr><td colspan="3" rowspan="2">1) Será usado o Access porque está disponível junto com o sistema operacional do Windows.</td><td>Responsável</td><td colspan="2">L. Lopez e dono da loja</td></tr>
<tr><td>Fonte</td><td colspan="2">Dono da loja</td></tr>
<tr><td colspan="3" rowspan="2"></td><td>Responsável</td><td colspan="2"></td></tr>
<tr><td>Fonte</td><td colspan="2"></td></tr>
<tr><td colspan="6">Restrições sobre a decisão</td></tr>
<tr><td colspan="6">Não existem restrições</td></tr>
</table>

Figura 64: Registro do problema de escolher SQL Server ou Access

Já a Figura 65 apresenta o registro do raciocínio para identificar o funcionário responsável pelo cadastramento de um cliente no sistema.

Registro do Raciocínio					
Identificador: 5 Prioridade: 5				Data: 7/5/2001	
Assunto					
Durante a implementação do módulo cliente surgeu a dúvida se o sistema deveria registrar o código do funcionário responsavél pelo cadastramento de um cliente porque no sistema manual de vídeo locadora as fichas dos clientes tinham informações incompletas e erradas.					
Objetivos organizacionais	Objetivos do Sistema		Restrições		
[OBO-3] Gerenciar as informações dos clientes	[OBS-3] Ter um cadastro de todos os clientes da loja				
Posições					
Apóia			Contrapõe		
Argumento	Fonte	Responsável	Argumento	Fonte	Responsável
1) O sistema deveria registrar o funcionário que cadastra um cliente porque identificaríamos o responsável pelo problema	Dono da loja	Dono da loja	Não existes argumentos		
2) O sistema deverá solicitar a senha do funcionário para confirmar o cadastro do cliente	Dono da loja	Dono da loja	Não existes argumentos		
3) a automação disso é fácil de implementar	L. Lopez	L. Lopez			
Decisão					
1) Toda inclusão de cliente deverá levar o código do funcinário			Responsável	Dono da loja e L. Lopez	
			Fonte	Dono da loja e L. Lopez	
			Responsável		
			Fonte		
Restrições sobre a decisão					
Não existem restrições					

Figura 65: Registro do problema de identificar o funcionário responsável pela cadastro de um cliente

Na próxima seção apresentaremos as considerações finais do capítulo.

4.8 Considerações Finais

O capítulo apresentou e exemplificou um modelo intermediário de rastreamento e um conjunto de diretrizes (um processo) para desenvolver um modelo de rastreamento. Ambos os dois foram aplicados e exemplificados sobre o desenvolvimento de um modelo de rastreamento de um sistema de vídeo locadora.

O modelo intermediário possui a maioria dos elementos/artefatos encontrados na maioria dos modelos de rastreamento dos diferentes software. Por sua vez, o processo é composto de um conjunto de diretrizes que orientam a identificação dos elementos/artefatos que irão compor o modelo de rastreamento de um software. Porém, é importante salientar que ambos os dois devem ser adaptados às reais necessidades de um projeto de software.

A sistemática de trabalho dos participantes que desenvolveriam e rastreariam o sistema de vídeo locadora consistiu de um treinamento teórico e prático.

O treinamento teórico foi fundamentado nos trabalhos de [Ramesh, 1998], [Ramesh, 2001], [Wiegers, 1999], [Gotel, 1996a] e [Toranzo, 1999]. O ênfases do aspecto teórico foi sobre a importância (tempo, custo e facilidade) do rastreamento na manutenção de um sistema.

O treinamento prático consistiu no desenvolvimento e gerenciamento das informações representadas pelo modelo de rastreamento do sistema em questão. Na primeira parte do trabalho prático, os participantes foram supervisionados, mas não foi prestada ajuda na identificação dos elementos que deveriam compor o modelo de rastreamento do sistema em questão. Nesta parte do trabalho foi constatado que as pessoas entendem o rastreamento, mas apresentaram dificuldades para identificar as informações que desejam rastrear. Diante dessa situação, consideramos oportuno apresentar e explicar o modelo intermediário e o processo para desenvolver um modelo de rastreamento. O ensino do modelo e do processo contribuiram na melhoria da compreensão do rastreamento e da atividade que os participantes deveriam cumprir. A apreciação dos participantes sobre o modelo intermediário de rastreamento pode ser resumida como "*O modelo intermediário ajudou na identificação e compreensão das classes (elemento/artefato) e os relacionamentos que podem compor o modelo de rastreamento do sistema de vídeo locadora. Os relacionamentos ajudaram a entender melhor as dependências entre as classes e compreender melhor da importância de alguns requisitos para a organização que hospedará o sistema*."

A experiência obtida no estudo de caso do sistema de vídeo locadora mostrou que, apesar das pessoas conhecerem e entenderem o que significa rastrear requisitos, a maioria delas tiveram grandes dificuldades para identificar e relacionar os elementos/artefatos a serem rastreados porque não sabiam que tipo de informação

deveriam incluir. O modelo intermediário e o processo ajudaram na estruturação e centralização dos esforços das discussões sobre os elementos que iriam compor o modelo de rastreamento do projeto. Além disso, facilitou a validação do modelo proposto pelos participantes.

A validação dos dados contidos nas matrizes de rastreamento foi realizada por pessoas que não participaram no rastreamento e desenvolvimento do sistema, mas tinham conhecimento do rastreamento de requisitos. Essa estratégia foi realizada para checar se o modelo, as informações rastreadas e as representações matricias dos tipos de relacionamentos contribuiriam na manutenção do sistema de vídeo locadora. O resultado da validação mostrou que todo o material apresentado ajudou a simplificar as tarefas realizadas.

A estratégia da validação do modelo de rastreamento consistiu de quatro passos. O primeiro passo foi entender o problema. O segundo passo foi identificar no modelo de rastreamento a classe que representa a informação onde se originou a mudança. O terceiro passo foi identificar os relacionamentos da classe afetada com as outras classes. O quarto passo foi identificar a informação que seria modificada na matriz de rastreamento as outras informações afetadas. Uma vez identificada as informações afetadas, o quarto passo foi novamente repetido sobre as matrizes que continham as informações afetadas.

Neste capítulo também apresentamos a proposta de um formulário para registrar o raciocíonio dos problemas acontecidos no desenvolvimento de um software. Um formulário foi proposto e aplicado para registrar problemas e soluções tomadas no decorrer do sistema de vídeo locadora. O formulário foi importante e uma pequena fonte de informação acerca do processo de desenvolvimento e não simplesmente um documento técnico de como foi projetado um sistema. É difícil encontrar documentação que identifique os problemas acontecidos no projeto. Portanto, o registro do raciocínio permite apreender e evitar os problemas documentados do projeto.

No capítulo seguintes aplicaremos e exemplificaremos o processo apresentado neste capítulo sobre um sistema de biblioteca que está sendo desenvolvido na Universidade Estadual do Oeste do Paraná. É importante salientar que a mesma estratégia de trabalho e validação da nossa proposta de trabalho foram realizadas sobre outros estudos de casos.

Capítulo 5

Estudo de Caso: Sistema de Biblioteca

Este capítulo apresenta o estudo de caso do rastreamento de requisitos sobre um sistema de biblioteca.

5.1 Introdução

Neste capítulo forneceremos um estudo de caso que ilustra a aplicação e o uso do processo para desenvolver um modelo de rastreamento de um sistema de biblioteca. Também ilustraremos o uso de matrizes e listas no rastreamento. A parte restante do capítulo está estruturada como segue. A Seção 5.2 apresenta uma visão geral do sistema de biblioteca. A Seção 5.3 apresenta e exemplifica as diretrizes para desenvolver um modelo de rastreamento. Finalmente, a Seção 5.4 apresenta as considerações finais do capítulo.

5.2 Descrição do Sistema de Biblioteca

O Sistema de Biblioteca visa gerenciar e integrar os dados bibliográficos de todas as bibliotecas das diferentes sedes da Universidade Estadual do Oeste do Paraná (UNIOESTE), localizadas em diferentes cidades do Estado do Paraná. O sistema gerenciará todas as informações dos seus usuários (alunos, professores e funcionários), operadores (atendentes, bibliotecária, administradores), das obras (livros, multimídia, etc), circulação de obras (por exemplo, empréstimo, reserva, devolução), e fornecerá dados estatísticos referentes ao acervo e circulação das obras.

Alguns dos fatores que contribuíram para a escolha do sistema de biblioteca foram:

1. Carência de um processo definido e documentado da engenharia de requsitos. Logo, os requisitos não são documentados e estão na mente dos desenvolvedores. A aplicação do rastreamento de requisitos introduz algumas mudanças sobre o desenvolvimento do sistema, tais como: documentação dos requisitos, elaboração de alguns diagramas da UML (classe e caso de uso) para entender melhor o sistema. Sabemos que essas mudanças são insuficientes, porém, se começou a entender melhor a importância de documentar os requisitos;

2. Sensibilizar aos desenvolvedores da importância do rastreamento de requisitos. Existe uma falta de cultura no rastreamento de requisitos. Logo, através do trabalho da biblioteca desejamos contribuir e melhorar essa cultura no desenvolvimento de software;

3. Oportunidade da realização de um trabalho de conclusão de curso sobre rastreamento de requisitos. Um trabalho de conclusão de curso no Curso de informática da UNIOESTE foi oportuno e apropriado para a aplicação do nosso

processo de rastreamento. Um dos objetivo do trabalho de conclusão de curso é ser um projeto piloto de como usar e aplicar o rastreamento de requisitos;

4. Necessidade de reduzir e melhorar a manutenção do sistema de biblioteca. Atualmente, existem vários projetos em desenvolvimento na diretoria de informática cujas manutenções consomem muito tempo e são muito confiáveis porque elas introduzem outros erros no sistema.

A seguir apresentaremos o processo para desenvolver o modelo de rastreamento do sistema da biblioteca.

5.3 Processo para Desenvolver um Modelo de Rastreamento

Nesta seção lembraremos e aplicaremos as diretrizes do processo para construir um modelo de rastreamento. No decorrer das aplicações das diretrizes serão apresentadas algumas telas do sistema de biblioteca visando uma melhor compreensão das informações rastreadas.

5.3.1 Informar as Expressões e Convenções usadas nas Matrizes de Rastreamento

As expressões usadas nas matrizes de rastreamento são apresentadas na Tabela 17.

Expressões usadas nas matrizes de rastreamento	
[OBO] = Objetivo organizacional	[PRO] = Programa do sistema
[OBS] = Objetivo do sistema	[DGM] = Diagrama
[REQ] = Requisito do sistema	[PES] = Pessoa
[SUB] = Subsistema do sistema	[FMT] = Formatação bibliográfica

Tabela 17: Expressões usadas nas matrizes de rastreamento

A subseção seguinte apresenta as diretrizes para a identificação das classes candidatas para o modelo de rastreamento.

5.3.2 Identificar Classes no Nível de Informação Externo

Considerando o modelo intermediário de rastreamento de requisitos, preenche-se a a Tabela 18 com os resultados obtidos com a seguinte diretriz.

Níveis de informação	Análise	Projeto	Implementação	Teste
Externo				
Organizacional				
Gerencial				
Desenvolvimento				

Tabela 18: Tabela para organizar as classes candidatas

Diretriz 1: Identificar as informações que podem afetar o sistema. O objetivo da diretriz é identificar classes no nível de informação externo que podem afetar o sistema.

O resultado da aplicação dessa diretriz foi a inclusão da classe *InformaçãoFormato* para referenciar as informações dos formatos de visualização e de catalogação de obras, chamados *MARC* (*MAchine-Readable Cataloging*) e *AACR* (*Anglo-American Cataloging Rules*). A Tabela 19 é um fragmento do formato bibliográfico *MARC*.

Informações do Ambiente externo	
[FMT-2]	O sistema também deve seguir normas definidas no padrão *MARC.* Estrutura do Formato USMARC Dados fixos: Campo 001 – Número de Controle Campo 003 – Identificador do Número de Controle Campo 005 – Data e Hora da Última Operação Campo 008 – Informações Gerais Dados Variáveis: Campo 020 – ISBN Campo 040 – Fonte de Catalogação Campo 041 – Código da Lingua Campo 043 – Código de área geográfica Campo 045 – Código Cronológico Campo 082 – Classificação Decimal de Dewey 1XX – Área da entrada principal (Nome pessoal, Entidade coletiva, Evento, Título Uniforme). 24X – Área do Título (Título Uniforme, Título principal) 250 – Área da Edição 260- Área da Publicação

Tabela 19: Exemplo de campos para a catalogação de obras

A seguir abordaremos a identificação das classes no nível de informação organizacional.

5.3.3 Identificar Classes no Nível de Informação Organizacional

Considerando o modelo intermediário de rastreamento de requisitos, continua-se preenchendo a Tabela 18 com os resultados obtidos com as seguintes diretrizes.

Diretriz 2: Identificar os objetivos, estratégias ou regras de negócio que serão rastreadas. O objetivo dessa diretriz é identificar os objetivos, estratégias ou regras registradas que podem afetar o sistema desejado.

Após aplicar essa diretriz, foi decidido a inclusão da classe *ObjetivoOrganizacional*, do modelo intermediário, no modelo de rastreamento do sistema de biblioteca porque os objetivos organizacionais foram registrados e disponibilizados. A Tabela 20 apresenta alguns dos objetivos organizacionais da UNIOESTE sobre o sistema de biblioteca.

Objetivos Organizacionais	
[OBO-1]	Integrar as informações do acervo bibliográfico das bibliotecas dos diferentes campus da UNIOESTE
[OBO-2]	Controle dos diversos tipos de materiais existentes na biblioteca (Monografias, Periódicos, Multimídias)
[OBO-3]	Gerenciar e centralizar todas as informações de todos os usuários do sistema de biblioteca
[OBO-4]	Controlar o acesso e uso do sistema de biblioteca
[OBO-5]	Incorporar sistemas nacionais e internacionais de catalogação
[OBO-6]	Emitir relatórios conforme as necessidades das bibliotecas e o Ministério de Educação
[OBO-7]	Emitir dados estatísticos conforme as necessidades das bibliotecas
[OBO-8]	Disponibilizar consultas dos diversos materiais contidos no acervo da biblioteca aos seus usuários
[OBO-9]	Automatizar e controlar os empréstimos e reservas das obras
[OBO-10]	Incluir normas internacionais para gerenciar e/ou representar as informações bibliográficas

Tabela 20: Objetivos organizacionais do Sistema de Biblioteca

A seguir apresentaremos a identificação das classes no nível de informação gerencial.

5.3.4 Identificar Classes no Nível de Informação Gerencial

O objetivo desta seção é apresentar uma diretriz que recomenda a inclusão das classes *Tarefa, ObjetivoSistema* e *Pessoa* do modelo intermediário de rastreamento no modelo de rastreamento de um projeto.

Diretriz 3: Incluir classes de informação da gerência de projeto no modelo de rastreamento. O objetivo dessa diretriz é recomendar a inclusão das tarefas do

cronograma do projeto, os objetivos do sistema e Pessoas (desenvolvedores) no modelo de rastreamento do projeto porque desejamos reduzir o *gap* da gerência de projeto com a gerência de requisitos.

A classe *Tarefa* não foi incluída porque não foram registradas as atividades para desenvolver o sistema. A Tabela 21 apresenta os objetivos do sistema e a Tabela 22 mostra as pessoas e os papéis dos desenvolvedores do sistema.

Objetivos do sistema	
[OBS-1]	Controlar débitos referentes ao empréstimo de obras
[OBS-2]	Integrar e reusar as informações existentes das obras das diferentes bibliotecas da UNIOESTE.
[OBS-4]	Controlar dos diversos tipos de materiais existentes na biblioteca (Monografias, Periódicos, Multimídias)
[OBS-5]	Gerenciar o controle de acesso.
[OBS-6]	Gerenciar o empréstimo e reserva das obras.
[OBS-8]	Emitir relatórios estatísticos sobre a circulação dos materiais do acervo da biblioteca.
[OBS-11]	Gerenciar as classificações dos diversos materiais do acervo atendendo as tabelas CDD (*Classificação decimal Dewey*) ou CDU (Classificação decimal universal)
[OBS-12]	Gerenciar a catalogação de obras atendendo as normas especificadas no AACR (Anglo-American Cataloging Rules) e MARC (*Machine-Readable Cataloging*)

Tabela 21: Objetivos do sistema de biblioteca

Pessoa		
	Identificação	Código Papel
[PES-1]	Elton Mello	A, P, G
[PES-2]	Marcio Veronez	A, P
[PES-3]	Clovis Tomadão	P

a)

Papel	
Código	Descrição
A	Analista
P	Programador
G	Gerente de projeto

b)

Tabela 22: Tabelas das pessoas e papeis definidos no desenvolvimento do sistema

A seguir apresentaremos a aplicação de outras diretrizes para identificar informações no nível de informação de desenvolvimento.

5.3.5 Identificar Classes no Nível de Informação do Desenvolvimento

O objetivo desta seção é identificar classes no nível de informação de desenvolvimento do modelo intermediário de rastreamento.

Diretriz 4: Identificar subsistemas. O objetivo dessa diretriz é identificar os subsistemas que compõem o sistema.

O resultado da aplicação da diretriz foi a inclusão da classe *Subsistema* para registrar e representar os subsistemas do sistema de biblioteca. A Tabela 23 codifica, identifica e descreve os subsistemas do sistema de biblioteca.

Subsistemas do Sistema de Biblioteca		
Código	Identificação	Descrição
[SUB-1]	Gerência de Obra	Manipulação das obras da biblioteca
[SUB-2]	Gerência de Usuário	Controla as informações dos usuários
[SUB-3]	Gerência Circulação de obra	Controla os empréstimos, reserva, devolução e consultas locais
[SUB-4]	Gerência de Relatórios	Responsável pela geração dos diferentes relatórios do sistema
[SUB-5]	Gerência de cadastro	Responsável pela geração de fichas das obras, impressão de etiquetas, código de barra
[SUB-6]	Gerência de Tabelas genéricas	Manipulação das informações genéricas, tais como, nacionalidade, estado

Tabela 23: Os subsistemas do sistema de biblioteca

A seguir incluiremos uma classe relacionada com requisitos.

Diretriz 5: Identificar requisitos. Essa diretriz indica que é obrigatório incluir a classe *Requisito*, do modelo intermediário, como uma classe candidata para o modelo de rastreamento do sistema de biblioteca.

O resultado da aplicação da diretriz foi a inclusão da classe *Requisito* para registrar e representar os requisitos do sistema. Para facilitar o entendimento dos requisitos e do sistema de biblioteca, dividimos a apresentação dos requisitos em várias tabelas. A Tabela 24 apresenta os requisitos relacionados com a gerência dos usuários do sistema.

[REQ-1]	O sistema deverá permitir a inclusão de usuários com os seguintes campos de informação: Código de Pessoa, Nome Completo, Endereço (Código do Endereço, Endereço, Tipo de endereço, Bairro, Cidade, UF, País, CEP, Caixa postal, Telefone, Fax), sobrenome, sexo, data de nascimento, nacionalidade (Tabela do Sistema), RG, Órgão Expedidor, Tipo de documento (carteira ou cédula), Data de expedição do documento, número da carteira, E-mail, biblioteca de origem.
[REQ-2]	O sistema deverá permitir a inclusão de dados referentes ao usuário caso este seja um operador do sistema de bibliotecas com o seguinte campo de formação: Nível de acesso.
[REQ-3]	O sistema deve permitir a consulta dos dados do usuário pelos campos: Nome, Sobrenome ou Nº de identificação. O retorno da consulta mostra todos os dados do usuário.
[REQ-4]	O sistema deve permitir alterar os dados do usuário. O único campo que não pode ser alterado é o Código da Pessoa Física.

Tabela 24: Requisitos relacionados com a gerência de usuários

A Figura 66 apresenta o cadastro de usuários da biblioteca . Veja o requisito [REQ-1] de la Tabela 24.

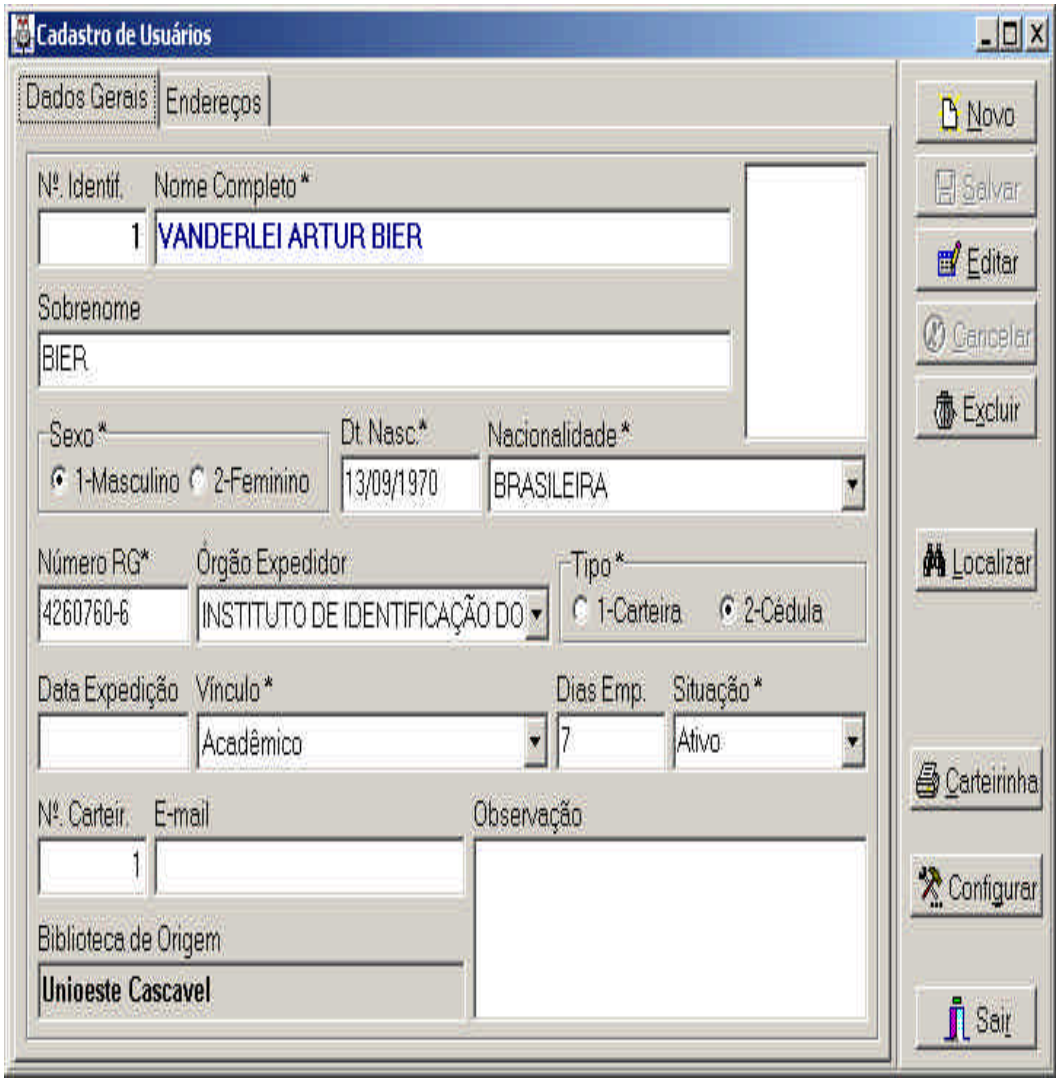


Figura 66: Tela de inclusão de usuários

A seguir, Tabela 25 apresenta alguns requisitos relacionados com a gerência de obras, especificamente, com livros.

[REQ-101]	O sistema deverá incluir obras com as seguintes informações: Código da Obra (Chave primária - incremental), Título, Tipo de Entrada (Autor, Entidade, Título), Público Alvo (Anexo 1 item 22), Forma Literária (Anexo 1 item 33) e Língua Principal (Anexo IV);
[REQ-102]	O sistema deverá permitir a consulta das Obras pelos campos: Título, Assunto (Tópico, Geográfico, Autor, Entidade), ISBN e palavra-chave. O sistema também deve prover para busca de obras um sistema de busca boleana com os campos Titulo, Assunto e Autor; e os operadores "E" e "OU". A consulta retorna os campos: Título, Classificação, Nº Autor e Ficha ...
[REQ-104]	O sistema deverá permitir a exclusão de uma obra somente depois que todas as suas dependências forem excluídas.
[REQ-105]	O sistema deverá incluir Títulos adicionais de obras com os seguintes campos de informação: Código do Subtítulo (Chave primária - incremental), Subtítulo, ...
[REQ-106]	O sistema deverá relacionar obras com sua classificação. O sistema deve permitir somente uma classificação de cada tipo para o livro por Biblioteca, isto é um livro pode ter uma classificação CDD e uma CDU, mais nuca duas classificações CDD ou CDU na Biblioteca. O sistema deve permitir que uma biblioteca ...

Tabela 25: Requisitos relacionados com a gerência de obras

A Figura 67 apresenta a tela para o cadastro de obras. Veja o requisito [REQ-101] da Tabela 25.

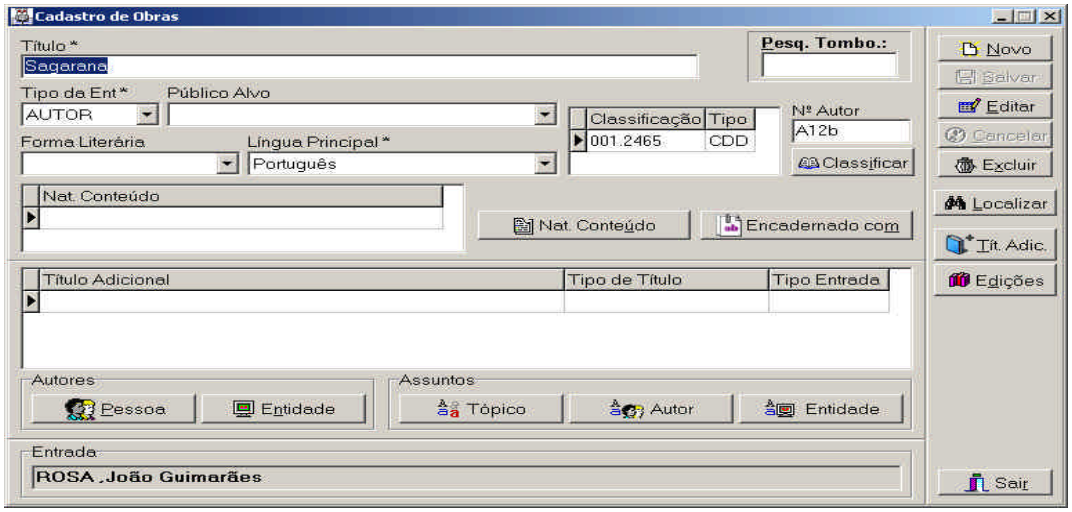


Figura 67: Tela do cadastro de obras

A Figura 68 apresenta a tela do sistema para consultar os livros. Veja o requisito [REQ-102] da Tabela 25.

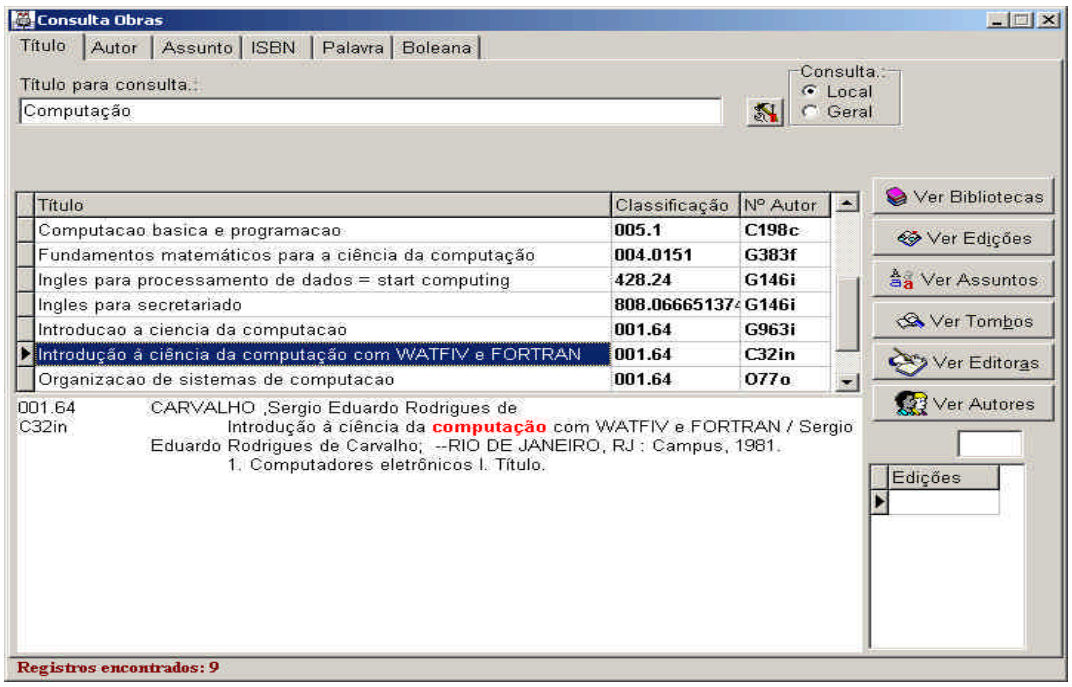


Figura 68: Tela de consulta de obras

A Tabela 26 apresenta alguns requisitos relacionados com a circulação dos livros.

[REQ-201]	O sistema deverá permitir empréstimos de livros desde que o usuário não tenha nenhum débito com a biblioteca e que o livro esteja disponível para o empréstimo. Os campos necessários para efetuar o empréstimo são: Tombo (Registro do livro) e Código de Pessoa Física.
[REQ-202]	O sistema deverá permitir a devolução de livros, verificando se existe algum atraso na devolução, caso haja o sistema deverá calcular e informar o valor da multa.
[REQ-203]	O sistema deverá permitir a reserva de livros desde que o usuário não tenha nenhum débito com a biblioteca e que o livro esteja disponível para a reserva. Os campos necessários para efetuar a reserva são: Tombo (Registro do livro) e Código de Pessoa Física.
[REQ-204]	O sistema deverá calcular o débito a ser pago por um usuário.

Tabela 26: Requisitos relacionados com circulação de obras

A Figura 69 apresenta a tela do sistema para fazer empréstimo de livros. Veja o requisito [REQ-201] da Tabela 26.

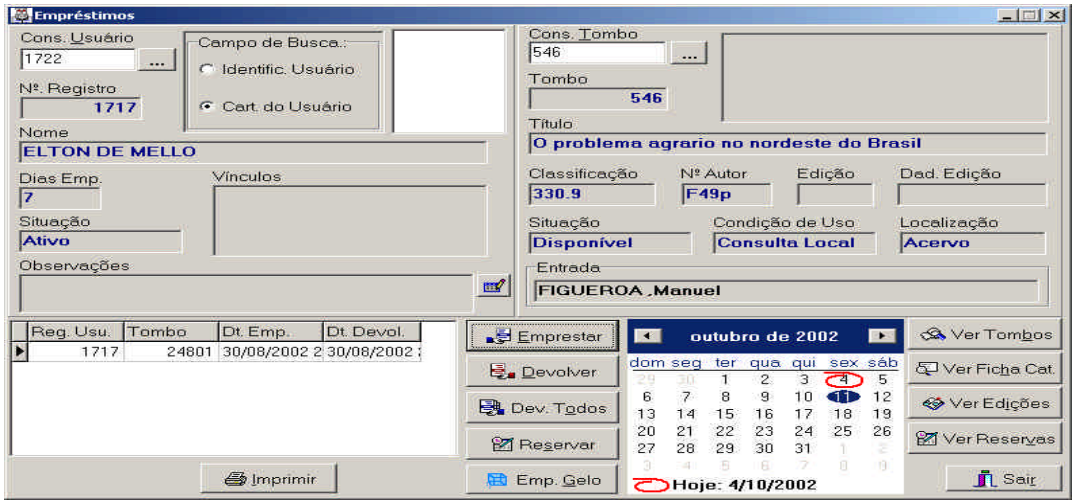


Figura 69: Tela de empréstimos de livros

A Tabela 27 apresenta alguns requisitos relacionados com a gerência de relatórios.

[REQ-301]	O sistema deve gerar relatórios de livros dando como entrada intervalos de classificação ou data de registro, como retorno os seguintes campos são requeridos: Título, Edição, Data de Publicação, Entrada principal, classificação, número do autor, número de exemplares e data de registro.
[REQ-302]	O sistema deve gerar os seguintes tipos de relatório: Padrão (Modelo da biblioteca com os campos do requisito [REQ-301]), Referência Bibliográfica, MARC, e AACR2.
[REQ-303]	O sistema deve gerar relatório de livros atrasados, o relatório deve conter as seguintes informações: Nome do usuário, Telefone do usuário, Título do livro, Tomo do exemplar, data de devolução e valor da multa.
[REQ-304]	O sistema deve gerar relatório de livros emprestados por intervalo de data e ou usuário, o relatório deve conter as seguintes informações: Nome do usuário, tombo, título, data de empréstimo e data de devolução.

Tabela 27: Requisitos relacionados com relatórios

A Figura 70 apresenta um relatório no formato *MARC*. Veja o requisito [REQ-302] Tabela 27.

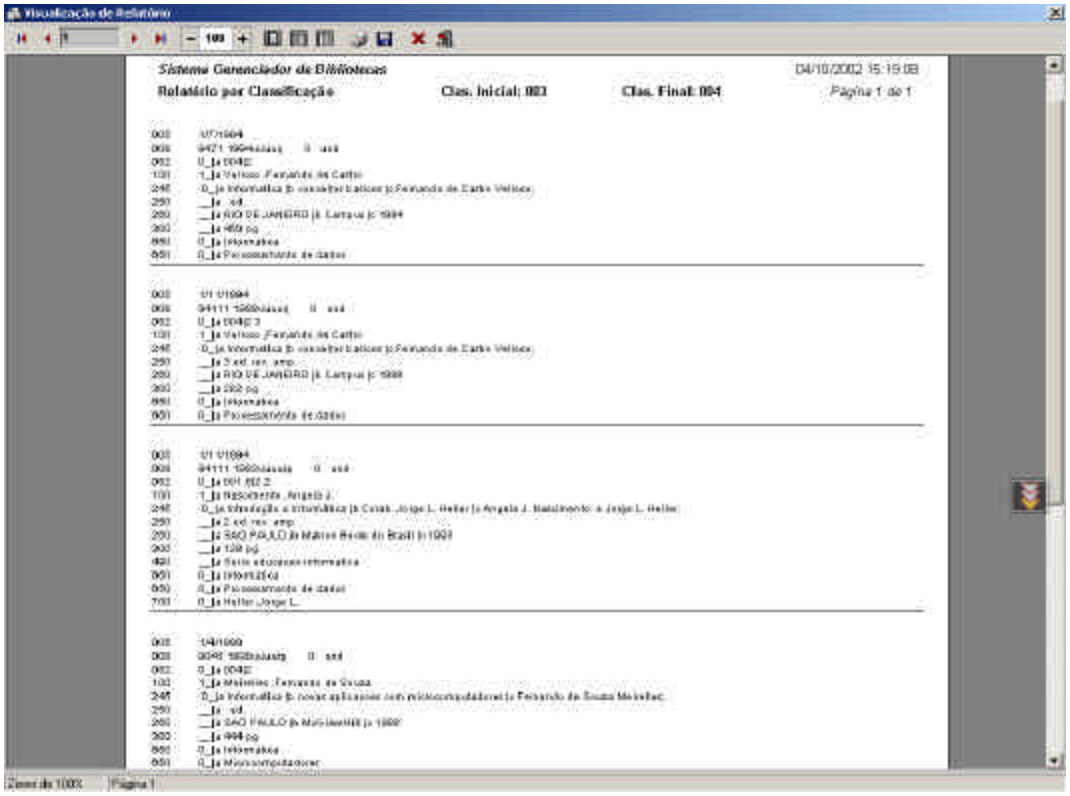


Figura 70: Tela de visualização de relatório

Alguns requisitos relacionados com geração de fichas bibliográficas são apresentados na Tabela 28.

[REQ-401]	O sistema deve gerar fichas catalográficas com os seguintes tipos de entrada: Entrada principal, Autor, Assunto, Tombo, Série, Título e Encadernação. As fichas devem seguir as normas definidas pelo AACR2.
[REQ-402]	O sistema deverá gerar etiquetas de código de barra para identificação dos exemplares de livros.
[REQ-403]	O sistema deverá gerar etiquetas para adesão na lombada do livro, para identificação deste na prateleira.

Tabela 28: Requisitos relacionados com a geração de ficha

A Figura 70 apresenta a tela de ficha bibliográfica do sistema de biblioteca. Veja o requisito [REQ-401] da Tabela 28.

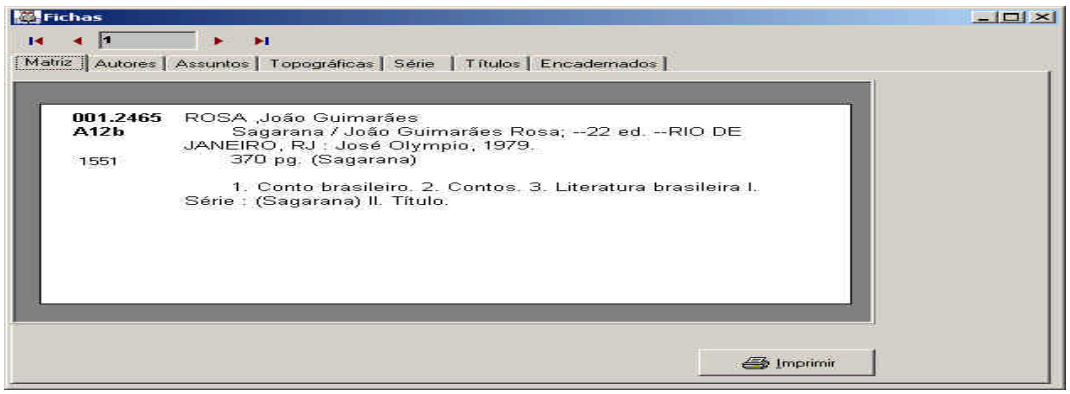


Figura 71: Tela de ficha bibliográfica

O sistema possui vários requisitos que incluem informações genéricas, tais como: instituição, cadastramento de bibliotecas, cadastramento de autores, assunto (área do conhecimento), etc. Na Tabela 29 somente incluímos requisitos relacionados com a Classificação e Autor.

[REQ-501]	O sistema deverá incluir Classificações com as seguintes informações Código da Classificação (Chave primária - incremental), Classificação, Tipo da Classificação. As classificações podem ser: CDD (Classificação Decimal Dewey), ou CDU (Classificação Decimal Universal).
[REQ-502]	O sistema deverá permitir a consulta das classificações pelos campos: classificação e tipo de classificação
[REQ-503]	O sistema deverá permitir a edição das classificações nos campos: Classificação e tipo de classificação.
[REQ-504]	O sistema deverá permitir a exclusão de uma classificação somente depois que todas as suas dependências forem excluídas.
[REQ-505]	O sistema deverá incluir Autores com os seguintes campos de informação: Código do Autor (Chave primária - incremental), Nome, Sobrenome, Numeração (Ex. João Paulo II), data de nascimento e ou morte, titulações (Tabela com campo relacionado ex. Príncipe, Doutor, etc.) e Pseudônimos do autor (Tabela).
[REQ-506]	O sistema deverá permitir a consulta dos autores pelos campos sobrenome e nome. A consulta retorna os campos: sobrenome, nome, numeração e data de nascimento/Morte.
[REQ-507]	O sistema deverá permitir a edição de todos os campos referentes ao autor, excluso o código do autor.
[REQ-508]	O sistema deverá permitir a exclusão de um autor somente depois que todas as suas dependências forem excluídas.

Tabela 29: Requisitos relacionados com informações genéricas

A seguir, a Figura 72 apresenta o cadastro de classificação. Veja o requisito [REQ-501] da Tabela 29.

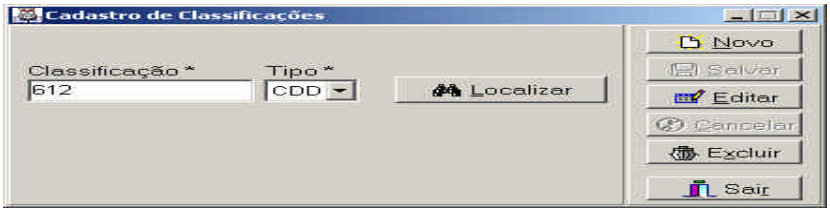


Figura 72: Cadastro de classificações

A seguir, a Figura 73 apresenta o cadastro de classificação. Veja o requisito [REQ-505] da Tabela 29.

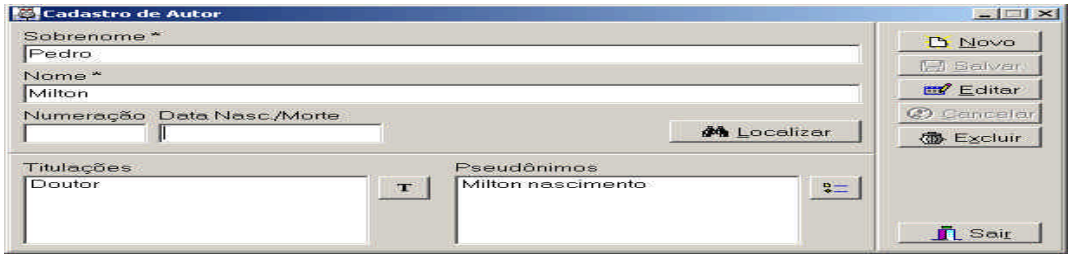


Figura 73: Cadastro de autor

A seguir identificaremos os diagramas nos quais foram representados os requisitos.

Diretriz 6: Identificar diagramas. Os objetivos desta diretriz são identificar os tipos de diagramas nos quais foram ou serão representados os requisitos do sistema e definir os caminhos lógicos para acessar as representações dos requisitos nos diferentes tipos de diagrama.

O resultado da aplicação da diretriz foi a inclusão da classe *Diagrama* que representará todos os diagramas nos quais foram modelados os requisitos. Também foi definido o caminho lógico.

A Tabela 30 apresenta a definição do caminho lógico no qual o símbolo ponto e vírgula (";") separa informações de diferentes tipos, como atributo e operação. Já o símbolo vírgula (",") foi usado para separar as informações do mesmo tipo, como uma lista de atributo.

Caminho lógico para acessar os requisitos no diagrama de classe	
Tipo referência	Caminho lógico
de requisito para classe	"C:nome_classe", onde "C" é uma abreviação para classe e "nome_classe" é o nome de uma classe

Tabela 30: Definição do caminho lógico para acessar requisitos no diagrama de classe

Diretriz 7: Identificar programas. O objetivo dessa diretriz é identificar os programas nos quais foram representados (implementados) os requisitos do sistema e definir um caminho lógico para acessar a representação dos requisitos nos programas.

O resultado da diretriz é a inclusão da classe *Programa* e a elaboração de caminhos lógicos para acessar a implementação dos requisitos. A Tabela 31 apresenta o caminho lógico usado no rastreamento do sistema.

Caminho lógico para relacionar requisitos com programas	
Tipo referência	Caminho lógico
de requisito para *unit*	"U:nome_unidade", onde "U" é uma abreviação para unidade onde foi implemento o requisito
de requisito para um procedimento.	"U:nome_unidade; P:nome_procedimento" , onde "P" é o nome do procedimento principal que implementa o requisito.

Tabela 31: Definição dos caminhos lógicos

A seguir apresentamos algumas unidades nas quais foram implementados os requisitos.

Nomes de unidades nas implementam os requisitos do Sistema de Biblioteca	
[PRG-1]	FcadUsuario
[PRG-3]	FconsUsuario
[PRG-4]	FcadUsuario
[PRG-5]	FcadObra
[PRG-7]	FcadObra
[PRG-8]	FcadObra
[PRG-12]	FcadEmprestimo
[PRG-14]	FcadReserva
[PRG-15]	FrelatorioObra
[PRG-19]	FrelatorioFicha
[PRG-21]	FimprimeLombadaObra
[PRG-22]	FcadClassificacao
[PRG-26]	FcadAutor
[PRG-27]	FconsAutor

Tabela 32: Nome de unidades do sistema de biblioteca

Diretriz 8: Identificar teste. O objetivo desta diretriz é identificar os documentos ou os fragmentos de documentos que especificam como os requisitos foram ou serão testados.

O desenvolvimento do sistema biblioteca incluiu testes que não foram documentados. Portanto, não foi incluída a classe *Teste* do modelo intermediário no modelo de rastreamento do sistema.

Deve-se checar se foram consideradas todas as classes a serem rastreadas da fase de projeto. A Tabela 33 apresenta as classes candidatas para o modelo de rastreamento do sistema de biblioteca.

	Análise	Projeto	Implementação	Teste
Externo	InformaçãoFormato			
Organizacional	ObjetivoOrganizacional			
Gerêncial	Pesssoa, ObjetivoSistema			
Desenvolvimento	Requisito	Diagrama, Subsistema	Programa	

Tabela 33: Classes candidatas

Na próxima subseção organizaremos as classes das diferentes fases.

5.3.6 Organizar e Estruturar as Classes Candidatas

Nesta subseção aplicaremos as seguintes diretrizes:

1. Diretriz 9: Remover as classes de informação irrelevantes;
2. Diretriz 10: Integrar as classes com o mesmo significado;
3. Diretriz 11: Integrar novas classes;

Após aplicar essas diretrizes, observou-se que não houve alterações sobre o número e significado das classes identificadas. Portanto, as classes da Tabela 33 foram preservadas.

5.3.7 Estabelecer Relacionamentos

O objetivo da subseção é fornecer e aplicar algumas diretrizes para estabelecer relacionamentos entre as classes candidatas. As diretrizes aplicadas foram:

1. **Diretriz 12: Organizar as classes**. O objetivo dessa diretriz é organizar e estruturar as classes candidatas em forma hierárquica.

2. **Diretriz 13: Estabelecer relacionamentos**. O objetivo é relacionar as classes candidatas.

Considerando as classes candidatas da Tabela 33, a Figura 74 apresenta o modelo de rastreamento para o sistema de biblioteca.

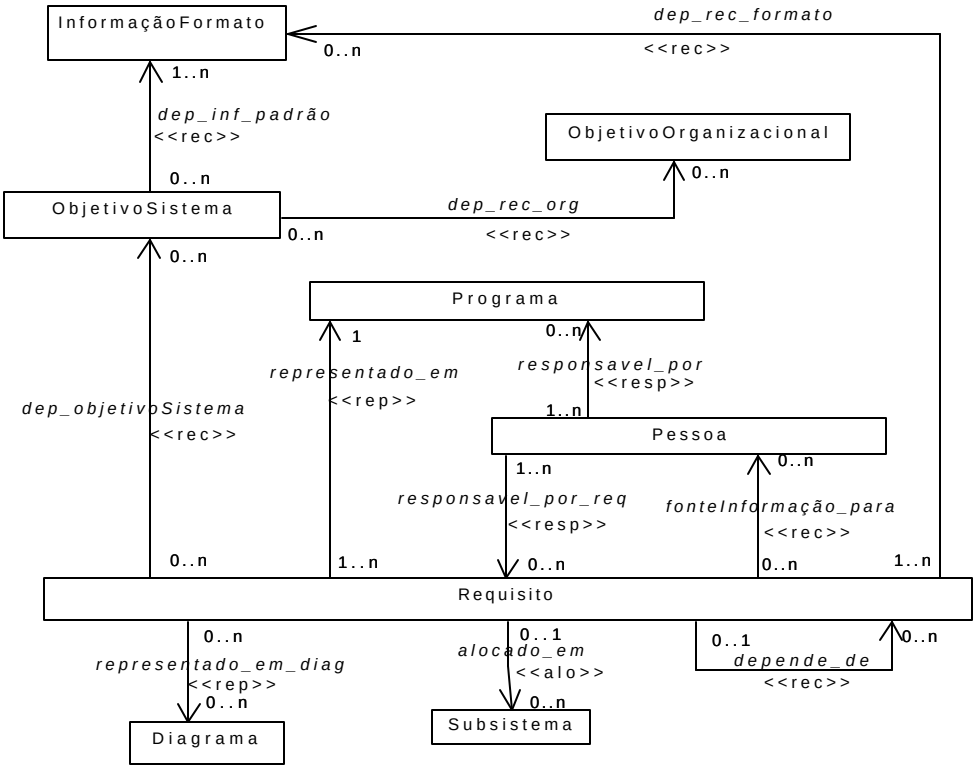
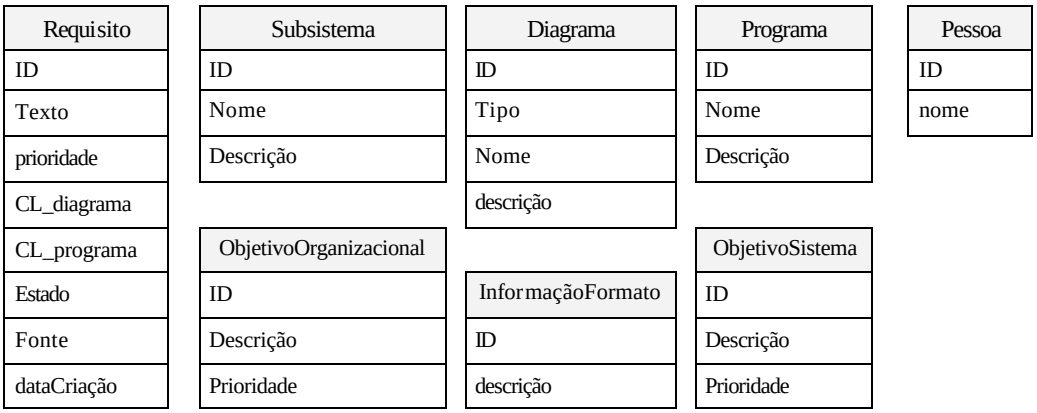


Figura 74: Modelo de rastreamento do sistema de biblioteca

A seguir aplicaremos outras diretrizes que recomendam alguns atributos para as classes.

5.3.8 Identificar as Propriedades sobre as Classes

A diretriz 14 (Recomendar atributos sobre as classes) foi aplicada e os seus resultados são apresentados a seguir.



Muitos dos atributos das classes já foram apresentados. A seguir fornerecemos algumas recomendações para gerar algumas matrizes de rastreamento.

5.3.9 Definir e preencher as Matrizes de Rastreamento

Nesta subseção apresentaremos e preencheremos a representação matricial de alguns relacionamentos do modelo de rastreamento para sua validação.

Diretriz 15: Definir uma matriz para cada relacionamento do modelo. O objetivo desta diretriz é desenvolver várias matrizes ou listas para representar os relacionamentos do modelo de rastreamento.

A seguir apresentaremos a representação matricial de vários relacionamentos utilizados na validação do modelo para facilitar mais o entendimento do nosso trabalho.

1. *dep_inf_padrão*, entre as classes *ObjetivoSistema* e *InformaçãoFormato*;
2. *dep_rec_org*, entre as classes *ObjetivoOrganizaconal* e *ObjetivoSistema*;
3. *dep_rec_formato* , entre as classes *InformaçãoFormato* e *Requisito* ;
4. *dep_objetivoSistema*, entre as classes *ObjetivoSistema* e *Requisito*;
5. *representado_em*, entre as classes *Requisito* e *Programa*.

dep_inf_padrão: <<rec>> →	
	FMT-2
[OBS-11] Gerenciar as classificações dos diversos materiais do acervo atendendo as tabelas CDD (*Classificação decimal Dewey*) ou CDU (Classificação decimal universal)	<A,A,cri>
[OBS-12] Gerenciar a catalogação de obras atendendo as normas especificadas no AACR (Anglo-American Cataloging Rules) e MARC (*Machine-Readable Cataloging*)	<A,A,cri>

Figura 75: Representação matricial do relacionamento *dep_inf_padrão*

A parte superior esquerda da matriz especifica que a matriz representa o relacionamento *dep_inf_padrão* (do tipo recurso) do modelo de rastreamento. A seta indica que as informações da primeira coluna, que representa os objetivos do sistema, depende da informação contida na primeira linha que representa informações do formato bibliográfico *MARC* (veja a Tabela 19).

Nas interseções dessas informaçãoes foi colocada a expressão "<A,A,cri>" que especifica que as informações do formato foram recursos de informação para a elaboração dos objetivos.

A Figura 76 apresenta a representação matricial do relacionamento *dep_rec_org* do modelo de rastreamento (veja Figura 74). É importante salientar que as informações dos objetivos apresentados na matriz foram omitidas para evitar replicação de informação, caso contrário seria difícil gerenciar os requisitos.

dep_rec_org:<<rec>> ←	[OBS-1]	[OBS-2]	[OBS-4]	[OBS-5]	[OBS-6]	[OBS-8]	[OBS-11]	[OBS-12]
[OBO-1] [OBO-2]		<A,A,cri>	<A,A,cri>					
[OBO-3]								
[OBO-4] [OBO-7]				<A,A,cri>		<A,A,cri>		
[OBO-8]								
[OBO-9]	<A,A,cri>				<A,A,cri>			
[OBO-10]							<A,A,cri>	<A,A,cri>

Figura 76: Representação matricial do relacionamento *dep_rec_org*

Na Figura 76, a parte superior esquerda mostra que a matriz representa o relacionamento *dep_rec_org* e que os objetivos do sistema (primeira linha) dependem dos objetivos da organização (primeira coluna). Por exemplo, se o objetivo "[OBO-10] Incluir normas internacionais para gerenciar e/ou representar as informações

bibliográficas" for excluído, então os seguintes objetivos do sistema: "[OBS-11] Gerenciar as classificações dos diversos materiais do acervo atendendo as tabelas CDD (*Classificação decimal Dewey*) ou CDU (Classificação decimal universal)" e "[OBS-12] Gerenciar a catalogação de obras atendendo as normas especificadas no AACR (Anglo-American Cataloging Rules) e MARC (*Machine-Readable Cataloging*)"

Na Figura 76, as expressões "<A,A,cri>" expressam que os objetivos do sistema são tem um alto grau de dependência sobre as informações contidas nos objetivos organizacionais. A sub-expressão "cri" expressa que os objetivos organizacionais foram utilizados para a elaboração dos objetivos do sistema. Logo, os objetivos do sistema precisam ser revisados se os objetivos organizacionais são modificados.

A Figura 77 apresenta a representação matricial do relacionamento *dep_rec_formato* do modelo de rastreamento (veja Figura 74)

dep_rec_formato: <<rec>> →	
	[FMT-2]
[REQ-102] O sistema deverá permitir a consulta das Obras pelos campos: Título, Assunto (Tópico, Geográfico, Autor, Entidade), ISBN e palavra-chave. O sistema também deve prover para busca de obras um sistema de busca boleana com os campos Titulo, Assunto e Autor; e os operadores "E" e "OU". A consulta retorna os campos: Título, Classificação, Nº Autor e Ficha	
[REQ-106] O sistema deverá relacionar obras com sua classificação. O sistema deve permitir somente uma classificação de cada tipo para o livro por Biblioteca, isto é um livro pode ter uma classificação CDD e uma CDU, mais nuca duas classificações CDD ou CDU na Biblioteca. O sistema deve permitir que uma biblioteca	<M,A,cri>
[REQ-302] O sistema deve gerar os seguintes tipos de relatório: Padrão (Modelo da biblioteca com os campos do requisito [REQ-301]), Referência Bibliográfica, MARC, e AACR2.	<A,A,cri>
[REQ-401] O sistema deve gerar fichas catalográficas com os seguintes tipos de entrada: Entrada principal, Autor, Assunto, Tombo, Série, Título e Encadernação. As fichas devem seguir as normas definidas pelo AACR	<A,A,cri>
[REQ-501] O sistema deverá incluir Classificações com as seguintes informações Código da Classificação (Chave primária - incremental), Classificação, Tipo da Classificação. As classificações podem ser: CDD (Classificação Decimal Dewey), ou CDU (Classificação Decimal Universal).	<A,A,cri>
[REQ-502] O sistema deverá permitir a consulta das classificações pelos campos: classificação e tipo de classificação	<M,A,cri>
[REQ-503] O sistema deverá permitir a edição das classificações nos campos: Classificação e tipo de classificação. [REQ-504] O sistema deverá permitir a exclusão de uma classificação somente depois que todas as suas dependências forem excluídas.	<M,A,cri> <M,A,cri>

Figura 77: Representação matricial do relacionamento *dep_rec_formato*

Na Figura 77, representação matricial do relacionamento *dep_rec_formato* relaciona os requisitos com informação do formato *MARC*. Do ponto de vista dos

requisitos, a informação do formato *MARC* é um recurso de informação. Por exempo, O requisito [REQ-501] tem um alto grau de dependência de informação com [FMT-2]. A sub-expressão "cri" expressa que as informações do formato foram utilizadas para a elaboração dos requisitos. Logo, os requisitos precisam ser revisados se [FMT-2] é modificado.

A Figura 78 apresenta a representação matricial do relacionamento *dep_objetivoSistema* (do tipo recurso) do modelo de rastreamento (veja Figura 74).

dep_objetivoSistema: <<rec>> →							
	[OBS-1]	[OBS-2]	[OBS-4]	[OBS-5]	[OBS-6]	[OBS-11]	[OBS-12]
[REQ-1]				<A;A,cri>			
[REQ-2]				<A;A,cri>			
[REQ-3]							
[REQ-4]				<M;A,cri>			
[REQ-101]		<A;A,cri>				<A;A,cri>	
[REQ-102]		<A;A,cri>				<M;A,cri>	
[REQ-104]		<A;A,cri>	<A;A,cri>				
[REQ-105]		<A;A,cri>	<A;A,cri>				
[REQ-106]		<A;A,cri>				<A;A,cri>	<A;A,cri>
[REQ-107]							
[REQ-201]	<M;A,cri>				<A;A,cri>		
[REQ-202]	<A;A,cri>				<A;A,cri>		
[REQ-203]					<A;A,cri>		
[REQ-204]	<M;A,cri>						
[REQ-302]							<A;A,cri>
[REQ-401]							<A;A,cri>
[REQ-402]						<A;A,cri>	<A;A,cri>
[REQ-403]						<A;A,cri>	<A;A,cri>
[REQ-501]						<A;A,cri>	
[REQ-502]						<M;A,cri>	
[REQ-503]						<M;A,cri>	
[REQ-504]						<M;A,cri>	

Figura 78: Representação matrical do relacionamento *dep_objetivoSistema*

Na Figura 78, a representação matricial do relacionamento *dep_objetivoSistema* especifica que os objetivos do sistema foram fonte de informação (recurso) para a

elaboração dos requisitos. Particularmente, a elaboração do requisito [REQ-101] utilizou como fontes de informação os objetivos [OBS-2] e [OBS-11]. O requisito [REQ-101] especifica que: o sistema deverá incluir obras com as seguintes informações: Código da Obra (Chave primária - incremental), Título, Tipo de Entrada (Autor, Entidade, Título), Público Alvo (Anexo 1 item 22), Forma Literária (Anexo 1 item 33) e Língua Principal (Anexo IV)"). O objetivo [OBS-2] especifica: controle dos diversos tipos de materiais existentes na biblioteca (Monografias, Periódicos, Multimídias). Já o objetivo [OBS-11] especifica: gerenciar as classificações dos diversos materiais do acervo atendendo as tabelas CDD (*Classificação decimal Dewey*) ou CDU (Classificação decimal universal).

Na linha do requisito [REQ-101], da Figura 78, as expressões indicam que esse requisito tem um alto grau de dependência com os objetivos com os quais está relacionado e que mudança nos requisitos poderiam modificar o requisito.

A Figura 79 apresenta a representação matricial do relacionamento *representado_em* (do tipo representação) do modelo de rastreamento (veja Figura 74).

As linhas da matriz especificam onde foi implementado cada requisito. Lembramos que a segunda coluna da matriz apresenta várias expressões (caminhos lógicos) que representa onde foram implementado os requisitos. Nas expressões, são utilizada as sub-expressões "imp", "U:" e "P:". A expressão "imp" especifica que trata-se de uma implementação. A expressão "U:" identifica o nome da unidade do programa implementado no ambiente de programa Delphi. A expressão "P:" identifica o nome do procedimento principal que implementa o requisito. Por exemplo, a linha dois da matriz expressa o que requisito [REQ-1] foi impementado na unidade fCadUsuario, especificamente no procedimento TF_CadUsuario.Frame_Botoes_Cadastro_CadUsuarioBtNovoClick. Já o requisito [REQ-2] foi implementado na unidade FcadOperador, especificamente, no procedimento TF_CadOperador.Frame_Botoes_Cadastro_CadUsuarioBtEditarClick.

A importância da matriz da Figura 79 é a identificação do trecho de um programa em que foi inicialmente implementado um requisito. Isso contribui na redução do tempo que um programador poderia levar em identificar o trecho do programa que implementa o requisito.

Representado _em: <<rep>> →	Programas
[REQ-1]	<imp,U: fCadUsuario; P: TF_CadUsuario.Frame_Botoes_Cadastro_CadUsuarioBtNovoClick(Sender: TObject)>
[REQ-2]	<imp,U: fCadOperador; P: TF_CadOperador.Frame_Botoes_Cadastro_CadUsuarioBtEditarClick(Sender: TObject)>
[REQ-3]	<imp,U: fConsUsuario; P: TF_ ConsUsuario.EditPesquisaChange(Sender: Tobject)>
[REQ-4]	<imp,U: fCadUsuario; P: TF_CadUsuario.Frame_Botoes_Cadastro_CadUsuarioBtEditarClick(Sender: TObject)>
[REQ-101]	<imp,U: fCadObra; P: TF_CadObra.FrameCadObraBtNovoClick(Sender: Tobject)>
[REQ-102]	<imp,U: fConsObra; P: TF_ConsObra.EditConsTituloKeyPress(Sender: TObject; var Key: Char);
[REQ-103]	<imp,U: fCadObra; P: TF_CadObra.FrameCadObraBtEditarClick(Sender: TObject)>
[REQ-104]	<imp,U: fCadObra; P: TF_CadObra.FrameCadObraBtExcluirClick(Sender: TObject)>
[REQ-105]	<imp,U: fCadTituloAdicional; P: F_CadTituloAdicional.Frame_Botoes_Cadastro_CadTituloAdicionalBtNovoClick(Sender:TObject)>
[REQ-106]	<imp,U: fRelItemClassificacao; P: TF_RelItemClassificacao.BtRelacionarClick(Sender: TObject)>
[REQ-107]	<imp,U: fRelDadItemClass; P: TF_RelDadItemClass.BtOkClick(Sender: TObject)>
[REQ-201]	<imp,U: fCadEmprestimo; P: TF_CadEmprestimo.BitBtnEmpClick(Sender: Tobject)>
[REQ-202]	<imp,U: fCadEmprestimo; P: TF_CadEmprestimo.BitBtnDevolverClick(Se nder: TObject)>
[REQ-203]	<imp,U: fCadEmprestimo; P: TF_CadEmprestimo.BitBtnReservarClick(Sender: TObject)>
[REQ-301]	<imp,U: fRelatorioObra; P: TF_RelatorioObra.BtLocalizarEdClick(Sender: TObject)>
[REQ-302]	<imp,U: fRelatorioObra; P: TF_RelatorioObra.Frame_Botao_ImprimirEdBtImprimirClick(Sender: TObject)>
[REQ-303]	<imp,U: fRelatorioEmprestimo; P: TF_RelatorioEmprestimo.BitBtnPesqAtrasadosClick(Sender: TObject)>
[REQ-304]	<imp,U: fRelatorioEmprestimo; P: TF_RelatorioEmprestimo.BitBtnPesqUsuDtClick(Sender: TObject)>
[REQ-401]	<imp,U: fRelatorioFicha; P: TF_RelatorioFicha.BtImprimeMatrizClick(Sender: TObject)>
[REQ-501] [REQ-502]	<imp,U: unit fCadClassificacao; P: TF_CadClassificacao.Frame_Botoes_CadastroBtNovoClick(Sender: TObject); <imp,U: unit fConsClassificacao; P: TF_ConsClassificacao.EditPesquisaChange(Sender: TObject);
[REQ-503]	<imp,U: unit fCadClassificacao; P:TF_CadClassificacao.Frame_Botoes_CadastroBtEditarClick(Sender: TObject);
[REQ-504]	<imp,U: unit fCadClassificacao; P: TF_CadClassificacao.Frame_Botoes_CadastroBtExcluirClick(Sender: TObject);

Figura 79: Representação matricial do relacionamento *representado_em*

5.3.10 Validar as Matrizes de Rastreamento

A subseção apresenta a realização de uma diretriz que trata a questão da validação matricial dos relacionamentos que compõem o modelo de rastreamento do sistema.

Diretriz 16: Validar o modelo de rastreamento. O objetivo desta diretriz é usar as matrizes preenchidas na Subseção 5.3.9 para validar a importância e as facilidades de se ter um modelo de rastreamento.

A validação das matrizes consistiu em fazer uma análise de impacto. Para isso, considere o seguinte cenário na que a UNIOESTE decidiu pela retirada do padrão *MARC* do sistema de biblioteca. Portanto, deveremos identificar as partes afetadas do sistema. A seguir aplicamos alguns passos para identificar os elementos afetados.

1. Revisando o modelo de rastreamento foi constatado que a mudança afeta diretamenta alguns objetivos do sistema. Logo precisamos identificá-los na matriz da Tabela 21. Logo, foi determinado que a classe *ObjetivoSistema* foi afetada, especificamente, os objetivos [OBS-11] e [OBS-12];
2. No modelo de rastreamento devemos identificar os relacionamentos existentes da classe *ObjetivoSistema* com outras classes. Após revisar o modelo de rastreamento foi determinado que o relacionamento *dep_objetivoSistema* (Figura 78) deveria ser examinado;
3. Para cada uma das representações matriciais dos relacionamentos conectados com os objetivos do sistema é necessário fazer uma revisão para identificar outras instâncias das classes relacionadas com [OBS-11] e [OBS-12]. Após revisar os objetivos [OBO-11] e [OBO-12] na representação matricial do relacionamento *dep_objetivoSistema* (Figura 78), identificamos que os requisitos afetados foram: [REQ-101], [REQ-102] [REQ-106], [REQ-302], [REQ-401], [REQ-402], [REQ-403], [REQ-501], [REQ-502], [REQ-503] e [REQ-504].
4. Observando o modelo de rastreamento foi possível identificar que a classe *Requisito* está relacionada com as classes *Programa* e Diagrama. Para identificar os programas afetados deve-se revisar a representação matricial do relacionamento *representado_em* (Figura 79). Para alguns dos requisitos afetados, a Tabela 34 apresenta os procedimentos que precisam ser revisados.

Requisitos	Programas
[REQ-102]	<imp,U: fConsObra; P: TF_ConsObra.EditConsTituloKeyPress(Sender: TObject; var Key: Char);
[REQ-106]	<imp,U: fRelItemClassificacao; P: TF_RelItemClassificacao.BtRelacionarClick(Sender: TObject)>
[REQ-302]	<imp,U: fRelatorioObra; P: TF_RelatorioObra.Frame_Botao_ImprimirEdBtImprimirClick(Sender: TObject)>
[REQ-401]	<imp,U: fRelatorioFicha; P: TF_RelatorioFicha.BtImprimeMatrizClick(Sender: Tobject)>
[REQ-501]	<imp,U:unit fCadClassificacao; P:TF_CadClassificacao.Frame_Botoes_CadastroBtNovoClick (Sender: TObject);
[REQ-502]	<imp,U:unit fConsClassificacao; P:TF_ConsClassificacao.EditPesquisaChange(Sender: TObject);
[REQ-503]	<imp,U:unit fCadClassificacao; P:TF_CadClassificacao.Frame_Botoes_CadastroBtEditarClick(Sender: TObject);
[REQ-504]	<imp,U:unit fCadClassificacao; P: TF_CadClassificacao.Frame_Botoes_CadastroBtExcluirClick(Sender: TObject);

Tabela 34: Lista de programas afetados

Nessa forma, temos apresentado uma simples avaliação do modelo de rastreamento. Outros tipos de validação do modelo poderiam ser realizadas, mas, seriam similar com a validação apresentada. A título de exemplo, se um requisito mudar, então deveremos identificar o programa que implementa o requisito e os diagramas que contem a modelagem dele.

A seguir apresentaremos as considerações finais do capítulo.

5.4 Considerações Finais

O capítulo apresentou um estudo de caso do rastreamento de requisitos sobre um sistema de biblioteca que está sendo desenvolvido na Universidade Estadual do Oeste do Paraná. O estudo de caso não está completamente concluído porque o sistema ainda está em desenvolvimento.

A aplicação da nossa proposta sobre o sistema de biblioteca teve efeitos colaterais positivos porque foram documentados os requisitos, as pessoas compreenderam a importância da documentação dos requisitos como um meio de comunicação e entendimento, houve maior interesse pela redação dos requisitos, e entenderam da importância de ter um processo, não somente para rastrear requisitos, mais também para

outras atividades (elicitação de requisitos, análise de sistema) do desenvolvimento de software.

Foi constatado que a transcrição gradual dos relacionamentos entre as informações do sistema para as matrizes contribuiram para um maior entendimento e compreensão do sistema para os desenvolvedores e para nosso trabalho. Em resumo, com o registro das informações relacionadas, o sistema deixou de ser uma caixa preta, sendo possível ter uma maior compreensão da sua estruturação.

O estudo de caso ajudou a constatar que o rastreamento de requisitos precisa de uma ou duas pessoas para sua realização e que os desenvolvedores deveriam ter conhecimentos de rastreamento e da importância dos artefatos serem rastreados para facilitar a manutenção de um sistema. Em geral, foi possível perceber que existe uma falta de cultura entre os desenvolvedores no que diz respeito ao rastreamento de requisitos e outras áreas do desenvolvimento do sistema. Logo, o rastreamento por si mesmo é insuficiente para introduzir boas práticas e garantir um desenvolvimento de software de qualidade.

Uma das maiores dificuldades encontradas no sistema de biblioteca, mas também nos outros estudos de casos foi fazer compreender a importância do rastreamento dos requisitos para a manutenção dos software. Para facilitar a compreensão de tal benefício, foi realizada uma troca da documentação do rastreamento dos sistemas entre os grupos participantes para que cada grupo realizasse uma análise de impacto sobre um sistema implementado por outros participantes, os resultados foram animadores porque as pessoas não tiveram problemas para identificar as partes afetadas do software com a introdução de uma mudança e porque utilizaram o modelo de rastreamento do sistema como um mapa para identificar as classes afetadas. Foi possível constatar que, em média, as pessoas que utilizaram as matrizes de rastreamento demoraram de cinco a dez minutos para realizar uma análise de impacto. Em geral, e sem especificar o tempo, os participantes concordaram que sem o rastreamento poderiam demorar muitos mais tempo em fazer uma manutenção de software.

A aplicação das diretrizes contribuiram para os participantes identificarem e compreenderem gradualmente a importância do sistema para a organização onde seriam hospedados porque o rastreamento incluiu os objetivos do sistema e os objetivos organizacionais. A realização do estudo de caso contribuiu para os participantes desenvolverem e compreenderam a importância do rastreamento de requisitos. O tempo utilizado para os participantes praticar e trabalhar inicialmente sobre o rastreamento foi de aproximadamente de 45 horas.

Apesar dos resultados parcias obtidos sobre o sistema de biblioteca, a Diretória de Informática da UNIOESTE não possui uma política e padrões definidos para o desenvolvimento de software. Esperamos que os resultados apresentados neste capítulo

possam ser posteriormente revistos para introduzir o rastreamento de requisitos no processo de desenvolvimento.

A seguir apresentaremos as conclusões do nosso trabalho.

Capítulo 6

Conclusões

Este capítulo apresenta as conclusões da tese e os trabalhos futuros.

6.1 Considerações Finais e Trabalhos Futuros

Este trabalho apresentou um estudo sobre o rastreamento de requisitos. No Capítulo 1 apresentamos uma visão da tese em termos da sua contribução e estruturação. Em seguida, no Capítulo 2 apresentamos uma revisão do estado da arte do rastreamento de requisitos. Para cada uma das pesquisas estudadas no Capítulo 2 foram apresentadas alguns dos seus prós e contras. Além disso, foi apresentada uma revisão de algumas ferramentas para a gerência de requisitos.

No Capítulo 3 foi apresentado, instanciado e comparado o nosso meta-modelo de rastreamento com outros meta-modelos existentes. Além disso, foi apresentada e exemplificada uma proposta para classificar as informações a serem rastreadas. O nosso trabalho, em relação a outros trabalhos ([Jarke, 1998], [Pohl, 1996a], [Pinheiro, 1996a], [Ramesh, 2001] e [Gotel, 1996b]), propôs uma classificação das informações a serem rastreadas em quatro níveis: externo, organizacional, gerencial e de desenvolvimento. O nível de informação externo representa as informações do contexto político, econômico e padrão (normas) que são externas à organização e que podem afetar alguns dos seus sistemas. O nível de informação organizacional representa as informações (recurso, processo, objetivo organizacional, regra, etc) que podem afetar o desenvolvimento dos sistemas da própria organização. O nível de informação gerencial representa algumas das informações (tarefa, objetivo do sistema, restrição, etc) empregadas pela gerência de projetos. O nível de informação de desenvolvimento inclui as informações que representam os diferentes elementos/artefatos produzidos no desenvolvimento de software, por exemplo, requisitos funcionais, documento, diagrama e programa. A classificação fornece dois grandes benefícios. Primeiro, a identificação, elicitação e validação das informações rastreadas podem ser organizadas e entendidas em função dos quatros níveis. Segundo, a aplicação da classificação é independente do pré e pós-rastreamento.

No Capítulo 4 foi apresentado e exemplificado um processo para desenvolver um modelo de rastreamento sobre um sistema de controle de locadora. Também, uma proposta para registrar o raciocínio de alguns problemas acontecidos no processo de desenvolvimento foi apresentada e exemplificada.

No Capítulo 5 foi apresentado um estudo de caso de rastreamento sobre um sistema de biblioteca que visa integrar as diferentes bibliotecas das diversas sedes da UNIOESTE - Universidade Estadual do Oeste do Paraná. A seguir apresentaremos as conclusões dos resultados obtidos.

Neste capítulo escreveremos declarações entre aspas dobles e itálico para resumir nossa apreciação respeito da experiência e comentários obtidos dos participantes.

A experiência e apreendizado dos participantes com respeito à classificação das informações a serem rastreadas podem ser resumidas como: "*A classificação das informações a serem rastreadas ajudou na identificação estruturada das informações que seriam rastreadas. Eles alertaram que provalmente poderíamos obter o mesmo resultado sem a classificação, mas poderiamos demorar mais tempo porque não teríamos um ponto de referência para elicitar, analisar e validar as informações*".

Neste trabalho definimos um conjunto de relacionamentos para melhorar a construção e entendimento dos modelos de rastreamento dos projetos. Os estudos de casos confirmaram que os tipos de relacionamentos ajudaram a entender melhor as dependências entre as classes do modelo de rastreamento. A experiência dos participantes respeito dos tipos de relacionamentos pode ser resumida como: *"Os tipos de relacionamentos ajudaram a entender a importância dos requisitos de um sistema para uma organização porque existem objetivos organizacionais que são satisfeitos pela implementação de alguns requisitos. Além disso, conseguimos entender melhor o significado dos relacionamentos entre as classes do modelo de rastreamento"*.

No decorrer desta pesquisa revisamos vários trabalhos que apresentam modelos de rastreamento, mas não mostram o meta-modelo usado para construir os modelos propostos. Conseqüentemente, podemos afirmar que o rastreamento de requisitos não possui um meta-modelo que unifique os diversos meta-modelos dos diferentes pesquisas para construir um modelo conceitual de rastreamento. Visando minimizar o problema citado, este trabalho propôs um meta-modelo que foi instanciado para gerar os modelos de pré-rastreamento de Gotel ([Gotel, 1996a]), de Ramesh ([Ramesh, 2001]) e o modelo intermediário de rastreamento.

Também foi proposto um modelo intermediário de rastreamento que foi fundamentado no trabalho de Ramesh ([Ramesh, 2001]). Porém, foram incluídos relacionamentos entre tarefas e requisitos para capturar a distribuição dos requisitos entre as tarefas do cronograma de projeto. A experiência obtida dos desenvolvedores com aplicação do modelo intermediário de rastreamento pode ser resumida como: *"O modelo intermediário ajudou os membros das equipes de desenvolvimento a entender e identificar as classes e os tipos de relacionamentos que poderiam compor o modelo de rastreamento do projeto.Em resumo, muitas das informações que dos modelos de rastreamentos dos projetos foram derivados do modelo intermediário"* .

Neste trabalho foi proposto um processo (um conjunto de diretrizes) para desenvolver um modelo de rastreamento. Muitas das pesquisas partem da suposição que um modelo de rastreamento correto já foi elaborado, porém, a aplicação de qualquer contribuição para rastrear poderá produzir resultados suspeitos ou errados porque o modelo de rastreamento pode estar com problemas. É importante salientar que nosso processo engloba e utiliza todas as contribuições citadas anteriormente. A proposta do processo para desenvolver um modelo de rastreamento pode ser resumida como segue:

"o processo proposto inclui um conjunto de diretrizes que contribuem na elaboração de um modelo de rastreamento de um projeto. O processo pode e dever ser adatado à realidade de cada projeto. Processo usa uma abordagem Top-Down para a elaboração de um modelo" .

Em termos da validação dos modelos de rastreamento dos diferentes estudos de casos, os resultados foram animadores porque as pessoas não tiveram dificuldades para fazer análise de impacto quando as informações estavam completas. A experiência dos participantes na validação dos seus modelos de rastreamento pode ser resumida como: *"O tempo e esforço que demanda o preenchimento das matrizes é alto, mas tive seu retorno recompensado porque as pessoas (externas ao grupo de desenvolvimento) compreenderam e reconheceram que o uso das matrizes geradas reduzem o tempo para identificar os artefatos afetados por uma mudança, mais ainda, elas são de grande ajuda porque a memória é frágil e os nomes dos procedimentos dos programas são esquecidos com facilidade. Os participantes reconheceram a importância de escrever requisitos claro e detalhados porque algumas análises de impacto identificaram requisitos incompletos ou ambíguos. Por exemplo, requisitos afetados e que estiveram relacionados com relatórios não incluíram todos os campos de informação porque se assumiu um conhecimento tácito entre os desenvolvedores,porém, isso não funcionou com as pessoas que fizeram a análise de impacto"*.

O nosso trabalho também possui algumas limitações que a seguir serão apresentadas.

O meta-modelo de rastreamento não é uma contribuição completa, nem fechada, porque é necessário mais estudos de caso em outros domínios de aplicação e com projetos de software maiores. Nesta tese exemplificamos como os relacionamentos propostos ajudam a capturar melhor os significados das associações entre classes (representado elementos que serão rastreados), mas novos tipos de relacionamentos podem ser propostos e acrescentados ao meta-modelo para incrementar o poder de expressão.

A inclusão de dados nas matrizes de rastreamento consome muito tempo e dedicação. Entre maior seja o projeto, maior será o esforço requerido para realizar a atividade de rastreamento porque existirão mais informações a serem gerenciadas. Logo, faz-se necessário de uma ferramenta para automatizar algumas das atividades do processo de rastreamento.

Alguns dos trabalhos futuros incluem:

1. Implementação de um sistema para a gerência de requisitos que inclua a questão do rastreamento de requisitos. Atualmente, no Depto. de Informática da Universidade Estadual do Oeste do Paraná estamos desenvolvimento um protótipo que suporta o meta-modelo proposto. É importante salientar que a ferramenta não automatizará

todas as diretrizes do processo porque existem algumas delas que requerem da experiência da pessoa, por exemplo, eliminar classes irrelevantes ou integrar as classes com o mesmo significado.

2. Melhoria do processo para desenvolver um modelo de rastreamento. Atualmente, iniciamos a aplicação desse processo sobre novos estudos de caso visando sua melhoraria.
3. Aplicação do processo sobre uma técnica de análise orientada a objetos. No Depto. de Informática da Universidade Estadual do Oeste do Paraná, o autor desta tese está trabalhando sobre a proposta de uma técnica de análise de sistema orientado a objeto que inclua a questão do rastreamento para manter consistentes as informações relacionadas.
4. Incorporação do rastreamento de requisitos não-funcionais. Para isso será necessário a definição de classes que permitam manipular um conjunto de relacionamentos com uma unidade de lógica porque os requisitos não-funcionais afetam diferentes partes de um sistema. Por exemplo, um requisito não-funcional pode especificar que "o sistema deverá ser seguro", diante disso, deveremos usar vários relacionamentos para associar o requisito com várias partes do software.
5. Propor métricas para auxiliar melhor o desempenho da nossa proposta. No decorrer da pesquisa não foram identificados trabalhos de métricas sobre o rastreamento de requisitos. Porém, pelos comentários e experiências obtidas, podemos afirmar que mais do 80% dos participaram concordaram que as contribuições apresentadas neste trabalho ajudaram na melhoria da atividade do rastreamento. Aproximadamente, 80% dos participantes indicaram que os trabalhos revisados no Capítulo 2 seriam mais interessantes se tivessem um processo que ajudasse na elaboração dos modelos de rastreamento. Além disso, cerca do 90% dos participantes manisfestou que uma contribuição importante do nosso trabalho, em relação a outros, é a proposta dos níveis de informação porque propõe uma forma simples e objetiva de classificar as informações que podem fazer parte de um modelo de rastreamento. Alguns futuros trabalhos de métricas podem incluir: o grau de entendimento das dependências dos elementos do modelo de rastreamento; e comparar as facilicidades para instanciar as diferentes propostas de rastreamento de requisitos visando medir a produtividade das pessoas.

Referências Bibliográficas

[Alencar, 1999] F. M. Alencar. Integação da Modelagem Organizacional com Especificação. Tese de Doutrado. Universidade Federal de Pernambuco, Departamento de Informática, Brasil, 1999.

[Receita, 2001] Receita da Fazenda. http://www.receita.fazenda.gov.br/Legislacao/LegisAssunto/Cpmf.htm. Acessada em abril de 2001.

[Alencar, 2000] F. Alencar, J. Castro, G. Cysneiros and J. Mylopoulos. From Early Requirements Modeled by the i* Technique to Later Requirements Modeled in Precise UML. In Anais do III Workshop de Engenharia de Requisitos, pp. 92-109. RJ, Brasil, 2000.

[Arnold, 1993] R. Arnold and S. Bohner. Impact Analysis – Toward A Framework for Comparation. Software Change Impact Analysis. In S. Shaw and R. Arnold (Eds.), IEEE Computer Society Press, pp. 34-43, 1993.

[Bailin, 1997] S. Bailin. Object Oriented Requirements Analysis. Software Requirements Engineering. In R. Thayer and M. Dorfman (Eds.), IEEE Computer Society Press, pp. 286-307, 1997.

[Bennet, 1999] S. Bennet, S. McRobb and R. Farmer. Object-Oriented Systems Analysis and Design: Using UML. McGraw-Hill, 1999.

[Bohner, 196] S. Bohner and R. Arnold. An Introduction to Software Change Impact Analysis. In S. Shaw and R. Arnold (Eds.), IEEE Computer Society Press, pp. 1-26, 1996.

[Booch, 1994] G. Booch. Object-Oriented Analysis and Design with Applications, Second Edition. Addison Wesley, 1994.

[Booch, 1999] G. Booch, J. Rumbaugh and I. Jacobson. Unified Modeling Language: Users Guide. Addison Wesley, 1999.

[Brown, 1992] A. Brown, A. Earl and J. McDermid. Software Engineering Environments: Automated Support for Software Engineering. McGraw-Hill, 1992.

[Caputo, 1998] K. Caputo. CMM Implementation Guide: Choreographing Software Process Improvement. Addison-Wesley, 1998.

[Carvalho, 2001] A. E. Carvalho. 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.

[Castro, 1996a] J. Castro, C. Gautreau and M. Toranzo. Multiview: An integrated Environment to Support Requirements Elicitation and Formalisation. In Proceedings XXIII Seminar on Integrated Hardware and Software, pp. 445-456. Recife, Brasil, 1996.

[Castro, 1996b] J. Castro, C. Gautreau and M. Toranzo. Towards an Environment to Support Requirements Formalization. In Proceedings of the 10th Symposium Brazilian of Software Engineering, pp. 189-206. São Carlos, Brasil, 1996.

[Castro, 1997a] J. Castro, and M. Toranzo. Towards Software Quality: The Multiview Case. In Proceedings of the 3rd Workshop on Requirements Engineering: Foundation For Software Quality, pp. 107-118. Barcelona, Espanha, 1997.

[Castro, 1997b] J. Castro, M. Toranzo, C. Gautreau and M. Bueno. Multiview: Requirements Modeling and Formalization. In Proceedings of the 11th Symposium Brazilian of Software Engineering, pp. 481-486. Fortaleza, Brasil, 1997.

[Castro, 2000] J. Castro, M. Kolp and J. Mylopoulos. Developing Agent-Oriented Information Systems for the Enterprise. In Proceedings of the 2nd International Conference on Enterprise Information Systems, Lectures on Computer Science, Springer-Verlag, pp. 234-245. Stafford, UK., 2000.

[Castro, 2001a] J. Castro, M. Kolp and J. Mylopoulos, *A Requirements-Driven Development Methodology*. In Proceedings of the 13th Conference on Advanced Information Systems Engineering, Lectures on Computer Science, Springer-Verlag, pp. 220-230. Interlaken, Switzerland, 2001.

[Castro, 2001b] J. Castro, F. Alencar, G. Cysneiros and J. Mylopoulos. Integrating Organizational and Object Oriented Modeling. In Proceedings of the 5th International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 110-117. Toronto, Canada, 2001.

[Chung, 2000] L. Chung, B. Nixon, E. Yu and J. Mylopoulos. *Non-Functional Requirements in* Software *Engineering.* Kluwer Publishing, 2000.

[Conklin, 1988] J. Conklin and M. Begeman. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transaction on Office Information Vol. 6, No. 4, pp. 303-331, 1988.

[DoD, 1984] Military Standard: Defense System Software Development. U.S Department Of Defense, 1984.

[Dorfman, 1997] M. Dorfman. Requirements Engineering. In R. Thayer and M. Dorfman (Eds.), Software Requirements Engineering, IEEE Computer Society Press, pp. 286-307, 1997.

[ECMA, 1991] ECMA. A Reference Model for Frameworks of Computer-Assisted Software Engineering Environments. Technical Report, European Computer Manufacturers Association, 1991.

[Erikson, 2000] H. Erikson and M. Penker. Business Modeling with UML: Business Patterns at Work. OMG Press. John Wileys & Sons Inc., 2000.

[Finkelstein, 1991] A. Finkelstein, Tracing Back from Requirements. In Proceedings of the Colloquium on Tools and Techniques for Maintaining Traceability During Design, pp. 7/1-7/2, 1991.

[Finkelstein, 1997] A. Finkelstein, O. Gotel, and R. Steven. Tutorial notes: Requirements traceability. Tutorial of the 3rd International Symposium on Requirements Engineering, 1997.

[Fiorini, 1998] S. Fiorini, A. Von Staa e M. Baptista. Engenharia de Software com CMM. Brasport, 1998.

[Fowler, 197] M. Fowler. Analysis Patterns: Reusable Object Models. Addison-Wesley, 1997.

[Fuggetta, 1996] A. Fuggetta and A. Wolf. Software Process. John Wiley and Sons Inc., 1996.

[Gamma, 1995] E. Gamma, R. Helm, R. Johnson and J. Vlissides. Design Pattern – Elements of reusable Object-Oriented Software. Addison Wesley, 1995

[Gane, 1983] C. Gane and T. Sarson. Análise Estruturada de Sistema. Editora Livros Técnicos e Científicos. 1983.

[Goguen, 1993] J. Goguen and C. Linde. Techniques for Requirements Elicitation. In Proceedings of the 1st International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 152-164, San diego, USA, 1993.

[Gotel, 1993] O. Gotel and A. Finkelstein. An Analise of the Requirements Traceability Problem. Technical Report TR-93-41, Imperial College of Science, Technology and Medicine, Department of Computing, 1993.

[Gotel, 1994a] O. Gotel and A. Finkelstein. Modeling the Contribution Structure Underlying Requirements. In Proceedings of the 1st International Workshop on Requirements Engineering: Foundation of Software Quality, Augustinus-Verlag, pp.71-81. Utrecht, Germany, 1994.

[Gotel, 1994b] O. Gotel and A. Finkelstein. Contribution Structures. Technical Report, Imperial College of Science, Technology and Medicine, Department of Computing, 1994.

[Gotel, 1994c] O. Gotel and A. Finkelstein. An Analise of the Requirements Traceability Problem. In Proceedings of the 1st International Conference Requirements Engineering, IEEE Computer Society Press, pp. 94 – 101. Colorado Springs, USA, 1994.

[Gotel, 1995] O. Gotel and A. Finkelstein. Contribution Strutures. In Proceedings of the 2nd International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 100-107. York, England, 1995.

[Gotel, 1996a] O. Gotel. Contribution Structures for Requirements Engineering. Ph.D Thesis. Imperial College of Science, Technology and Medicine, Department of Computing, U.K., 1996.

[Gotel, 1996b] O. Gotel and A. Finkelstein. 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.

[Harmon, 1997] P. Harmon and M. Watson. Understanding UML: The Development´s Guide with a Web-Based Application in Java. Morgam Kauffmann, 1997.

[Hsia, 1993] P. Hsia, A. Davis and D. Kung. Status Report: Requirements Engineering. IEEE Software. Nov., Vol. 10, No. 6, pp. 5-79, 1993

[IEEE-610.2, 1991] IEEE-610.2. IEEE Standard Glossary of Software Engineering Terminology, 1991.

[IEEE-830, 1984] IEEE-830. Guide to Software Specification, 1984.

[ISO-9003] 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.

[Jacobson, 1999] I. Jacobson, G. Booch and J. Rumbaugh. The Unified Software Development Process. Reading, MA: Addison Wesley, 1999.

[Jardim, 1999] N. Jardim, M. Toranzo, J. Cunha, J. Castro, S. Kovacecic, R. Dave, J. Tarby, M. Collinscope, and M. Harmelen. Lectures on Computer Science, pp. 34-58, 1999.

[Jarke, 1993] M. Jarke, J.Bubenko, C. Rolland, A. Sutcliffe, and Y. Vassiou. Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis. In Proceedings of the 1st International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 19-33. San Diego, USA, 1993.

[Jarke, 1995] M. Jarke, K. Pohl, R. Dömges, S. Jacobs, and H. Nissen. Requirements Information Management: The Nature Approach. Technical Report 95-8, RWTH Aachen, 1995.

[Jarke, 1996] M. Jarke. Meta Models For Requirements Engineering. In Knowledge Acquisition Workshop 1996. http://ksi.cpsc.ucalgary.ca/KAW/KAW96/jarke/Jarke.html. Acessada em setembro de 2001.

[Jarke, 1998] M. Jarke. Requirements Tracing. Communication of the ACM, Vol. 41, No. 12, pp. 32-36, 1998.

[Kontoya, 1998] G. Kotonya and I. Sommerville. Requirements Engineering: Processes and Techniques. John Wiley and Sons Inc., 1998.

[Krutchten, 2000] P. Krutchten. The Rational Unified Process: An Introduction. Second edition, Addison Wesley, 2000.

[Lam, 1998] W. Lam. Change Analysis and Management in a Reuse-Oriented Software Development Setting. In Proceedings of the 10th Conference Advanced Information Systems Engineering, Lecture Notes in Computer Science, Springer-Verlag, pp. 219-235. Pisa, Italy, 1998.

[Leffingwell, 2000] D. Leffingwell and D. Widrig. Managing Software Requirements: A Unified Approach. Addison-Wesley, 2000.

[Lehman, 1994] M. Lehman, Software Evolution. Encyclopedia of Software Engineering, pp. 1202 –1208, 1994.

[Lindgreen, 2000] P. Lindgreen. Towards a Useful Information Concept. In J. Brinkkemper and A. Solvberg (eds.). Lecture Notes in Computer Science, Springer-Verlag, pp. 249-260, 2000.

[Loucopoulos, 1995] P. Loucopoulos and V. Karakostas. System Requirements Engineering. McGraw-Hill, 1995.

[Macfarlane, 1995] I. Macfarlane and I. Reilly. Requirements Traceability in an Integrated Development Environment. In Proceedings of the 2nd International Symposium on Requirements Engineering, IEEE Computer Society Press, pp 116-123. York, England, 1995.

[Marc, 2002] MARC 21 Concise Format. Library of Congress Network Development and MARC Standards Office. http://lcweb.loc.gov/marc/concise/concise.html#general_intro. Acessada em fevereiro /2002.

[Marconi, 1996] Marconi System Technology. RTM (Requirements and Traceability Management. Technical Report, Marconi System Technology, 1996.

[Michelis, 1998] G. de Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, J. Schmidt, C. Woo and E. Yu. A Three-Faceted View of Information Systems. Communication of the ACM, Vol. 41, No. 12, pp. 64-70, 1998.

[Morris, 1994] P. Morris, A. Coombes and J. McDermid. Requirements and Traceability. In Proceeding 4th of the International Workshop on Requirements Engineering: Foundation of Software Quality, Augustinus-Verlag, pp. 82-87. Utrecht, Germany, 1994.

[Muller, 1979] G. P. Mullery. CORE: A Method for Controlled Requirements Expression. In Proceedings of the 4th International Conference on Software Engineering, IEEE Computer Society Press, pp. 126-135. Munich, Germany, 1979.

[Mylopoulos, 2000a] J. Mylopoulos, J. Castro and M. Kolp. Tropos: Toward Agent-Oriented Information Systems Engineering. In Proceedings of the 2nd International Bi-Conference Workshop on Agent-Oriented Information Systems. Bologna, Italy, 2000.

[Mylopoulos, 2000b] J. Mylopoulos and J. Castro. Tropos: A Framework for Requirements-Driven Software Development In J. Brinkkemper and A. Solvberg (eds.), Lecture Notes in Computer Science, Springer-Verlag, pp. 261-273, 2000.

[Mylopoulos, 2000c] J. Mylopoulos. Tropos at the age of 10 months, Tropos Meeting, University of Toronto, Department of Computer Science, 2000.

[Naur, 1969] P. Naur and Randell. Software Engineering: Report on a Conference Sponsored by the NATO Science Commission, October Brusels, 1969.

[Nunes, 1999] N. J. Nunes, M. Toranzo, M. Collins-Cope, J. Cunha, S. Kovacevic, D. Roberts, J. Tarby, J. Castro, M. Harmelen . W17. Workshop an Interactive System Design and Object Models-WISDOM'99. Lectures on Computer Sciences, pp. 34-58, 1999.

[OMG, 2000] Object Management Group. Meta Object Facility Specification (Version 1.3). http://www.omg.org, Acessada em julho de 2002.

[Orlikowski, 1993] W. Orlikowski. CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Developments. MIS, pp. 309-340, 1993.

[Palmer, 1997] J. Palmer. Traceability. Sofware Requirements Engineering. In R. Thayer and M. Dorfman (Eds.), IEEE Computer Society Press, pp. 364-374, 1997.

[Paulk, 1993] M. Paulk, M. Chissis, and C. Weber. Capability Maturity Model for Software: Version V1.1. Technical Report SEI-93-TR-24. Carnegie Mellon University, Software Engineering Institute, 1993.

[Pinheiro, 1996a] F. A. C. Pinheiro. Design of a Hyper-Environment for Tracing Object-Oriented Requirements. Ph.D Thesis. University of Oxford, Department of Computing, U.K., 1996.

[Pinheiro, 1996b] F. Pinheiro and J. Goguen. An Object-Oriented Tool for Tracing Requirements. IEEE Software, Vol. 13, No. 2, pp. 52-64, 1996.

[Pinheiro, 1996c] F. Pinheiro e J. Goguen, An Object-Oriented Tool for Tracing Requirements. In Proceedings of the 2nd International Conference on Requirements Engineering, IEEE Computer Society Press, pp. 19-27. Colorado Springs, USA, 1996.

[Pinheiro, 1999] F. Pinheiro, An Object-Oriented Library for Tracing Requirements. II Workshop on Requirements Engineering, pp. 187-197. Buenos Aires, Argentina, 1999.

[Pinto, 2002] R. Pinto, J. Castro e M. Toranzo, Requirements Traceability in Agent Oriented Development. In Proceedings of the 1st International Workshop on Software Engineering for Large Scale Multi-Agent Systems, Lecture Notes on Computer Science, pp. 21 – 25. Orlando, USA, 2002.

[Pohl, 1993] K. Pohl. Three Dimensions of Requirements Engineering. In the Proceeding of the 5th International Conference on Advanced Information Systems Engineering, Lecture Notes on Computer Science, pp. 275-292. Paris, France, 1993.

[Pohl, 1994a] K. Pohl, R. Dömges and M. Jarke. PRO-ART: Process Based on Requirements Pre-Traceability. Technical Report 94-7, RWTH Aachen, 1994.

[Pohl, 1994b] K. Pohl. Three Dimensions of Requirements Engineering: A Framework and its Application. Information Systems Press, pp. 243-258, 1994.

[Pohl, 1996a] K. Pohl. Process-Centered Requirements Engineering. John Wiley and Sons Inc., 1996.

[Pohl, 1996b] K. Pohl. PRO-ART: Enabling Requirements Pre-Traceability. In Proceeding of the 2nd International Conference on Requirements Engineering, IEEE Computer Society Press, pp. 31-38. Colorado Springs, USA, 1996.

[Pohl, 1996c] K. Pohl R. Domges and K. Schreck. A Filter-Mechanism for Method-Driven Trace Capture. In Proceedings of the 10th Conference on Advanced Information Systems Engineering, Lectures on Computer Science, pp. 237-250. Pisa, Italy, 1998.

[Pressman, 2001] R. Pressman. Software Engineering: A Practitioner's Approach. McGraw-Hill Book Company, 2001.

[QSS, 2000] QSSRequireit. http://www.qssrequireit.com. Acessada em março de 2001.

[Ramamoorthy, 1986] C. Ramamoorthy, V. Garg and A. Prakash. Programming in Large. IEEE Transacions on Software Engineering, Vol. 28, No. 12, pp. 769-783, 1986.

[Ramesh, 1992] B. Ramesh and V. Dhar. Supporting Systems Development by Capturing Deliberations During Requirements Engineering. IEEE Transactions on Software Engineering, Vol. 18, No. 6, pp. 498-510, 1992.

[Ramesh, 1993] B. Ramesh and M.Edwards. Issues in the Development of a Model of Requirements Traceability. In Proceedings of the 1st International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 256 – 259. San Diego, USA, 1993.

[Ramesh, 1995a] B. Ramesh, T. Powers, C. Stubbs and M.Edwards. Implementating Requirements Traceability: A Case Study. In Proceedings of the 2nd International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 89 – 95. York, England, 1995.

[Ramesh, 1995b] B. Ramesh, C. Stubbs, T. Powers and M. Edwards. Lessons Learned from Implementing Requirement Traceability. Technical Report, Naval Postgraduate School, Department of Computer Science, 1995.

[Ramesh, 1997] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards. Requirements Traceability-Theory and Practice. In proceedings of the 19th International Conference on Software Engineering (Vol. 3), pp. 397-4515. Boston, USA, 1997.

[Ramesh, 1998a] B. Ramesh and M. Jarke. Towards Reference Models For Requirements Traceability. Technical Report, Georgia State University, Department of Computer Science, 1998.

[Ramesh, 1998b] B. Ramesh. Factors Influencing Requirements Traceability Pratice. Communication of the ACM, Vol. 41, No.12, pp. 37-44, 1998.

[Ramesh, 2001] B. Ramesh and M. Jarke. Towards Reference Models For Requirements Traceability. IEEE Transacions on Software Engineering, Vol. 27, No.1, pp. 58-93, 2001.

[RequisitePro, 2001] Rational Software Corporation. http://www.rational.com. Acessada em setembro de 2002.

[Rose, 1998] T. Rose. Visual Assessment of Engineering Process in Virtual Enterprises. Communications of the ACM, Vol. 41, No.12, pp. 45-52, 1998.

[Rose, 2000] Rational Software Corporation. http://www.rational.com. Acessada em novembro de 2002.

[Rumbaugh, 1996] J. Rumbaugh. OMT Insights. New York: SIGS Books, 1996.

[Rumbaugh, 1999] J. Rumbaugh, I. Jacobson and G. Booch. The Unified Modeling Language: Reference Manual. Addison Wesley, 1999.

[Shaw, 1996] M. Shaw. Some patterns for software architectures. In J.Vlissides, J. Coplien, and N. KerthVlissides (eds.). Pattern Languages of Program Design, pp. 255-269, Addison-Wesley, 1996.

[Sommerville, 2000] I. Sommerville. Software Engineering. Addison-Wesley, 1996.

[Sommerville, 1997] I. Sommerville and P. Sawyer. Requirements Engineering: A Good Practice Guide. John Wiley and Sons Inc., 1997.

[Spence, 2000] I. Spence. And L. Probasco. Traceability Strategies for Requirements Management with Use Cases. White Papers, Rational Software Corporation, 2000.

[Standish, 1995] The Standish Group. The CHAOS Report. The Standish Group International, 1995. http://www.standishgroup.com). Acessada em maio de 2001.

[Standish, 1996] The Standish Group. The Unfinished Voyages Report. The Standish Group International, 1996,

http://www..standishgroup.com. Acessada em maio de 2001.

[Thayer, 1997a] R. Thayer and M. Dorfman (Eds.), Software Requirements Engineering. IEEE Computer Society Press, 1997.

[Thayer, 1997b] R. Thayer. Software Engineering Project Manager, Software Requirements Engineering. In R. Thayer and M. Dorfman (Eds.), IEEE Computer Society Press, pp. 286-307, 1997.

[Toranzo, 1998a] M. Toranzo and J. Castro. A Requirement Traceability Model. Latin-American Conference on Informatics CLEI'98 (in Portuguese) , pages 45-56. Bogotá, Colombia, 1998.

[Toranzo, 1998b] M. Toranzo and J. Castro. A Proposal for Requirement Traceability and Representation of Information. In proceedings of the 1st Workshop on Requirements Engineering, pp. 129-137. Maringá, Brasil, 1998.

[Toranzo, 1999a] M. Toranzo, J. Castro. A Comprehensive Traceability Model to Support the Design of Interactive Systems. WISDOM'99 Workshop. Disponível em http://math.uma.pt/wisdom99. Acessada em dezembro de 2000.

[Toranzo, 1999b] M. Toranzo e J. Castro. Multiview ++ Environment: Requirement Traceability from the Perspective of Different Stakeholders. In Proceedings of the 2nd Workshop on Requirement Engineering – WER'99, pp. 198–216, Buenos Aires, Argentina, 1999.

[Toranzo, 2000] M. Toranzo. Towards to Reference Models for Requirements Traceability. Em anais do III Fórum de Tecnologia: X Seminário Regional de Informática, pp. 12-22. Santo Angelo, Brasil, 2000.

[Toranzo, 2002] M. Toranzo, J. Castro e E. Mello. Uma proposta para Rastreamento de Requisitos. In proceedings of the V Workshop de Engenharia de Requisitos. pp. 31 46. Valencia, Espanha, 2002.

[Wiegers, 1996] K. Wiegers. Creating a Software Engineering Culture. Dorset House Publishing, 1996.

[Wiegers, 1999] K. Wiegers. Software Requirements. Microsoft Press, 1999.

[Woodcook, 1996] J. Woodcook. Using Z: Specification, Refinement, and Proof. Prentice Hall, 1996.

[Yourdon, 1994] Edward Yourdon. Análise Estruturada de Moderna. Editora Campus, 1990.

[Yu, 1994] E. Yu and J. Mylopoulos. Understanding Why in Software Process Modelling, Analysis and Design. In Proceedings of the 16th International Conference on Software Engineering, IEEE Computer Society Press, pp. 159-168. Sorrento, Italy, 1994.

[Yu, 1995a] E. Yu, P. Dubois and J. Mylopoulos. From Organizationn Models to System Requirements – A Cooperating Agents Approach. In Proceeding of the 2nd International Conference on Cooperative Information Systems, Lectures on Computer Science, Springer-Verlag, pp. 2-9. Chicago, USA, 1995.

[Yu, 1995b] E. Yu. Modelling Strategic Relationship for Process Reengineering. PhD Thesis. University of Toronto, Computer Science Department, Canada, 1995.

[Yu, 1997a] E. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In Proceedings of the 3rd International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 226-235. Annapois, USA, 1997.

[Yu, 1997b] E. Yu. Why Agent-Oriented Requirements Engineering. In Proceedings of the International Workshop on Requirements Engineering: Foundations for Software Quality, pp. 5-12. Barcelona, Espanha, 1997.

[Yu, 1997c] E. Yu and J. Mylopoulos. Modelling Organizacional Issues for Enterprise Integration. In Proceedings of the 1st International Conference on Enterprise Integration and Modelling Technology, IEEE Computer Society Press, pp. 12-21. Turin, Italy, 1997.

[Yu, 1998] E. Yu and J. Mylopoulos. Why Goal-Oriented Requirements Engineering. In Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality, Lectures on Computer Science, Springer-Verlag, pp. 15-22. Pisa, Italy, 1998.

[Yu, 1999] E. Yu. Strategic Modelling for Enterprise Integration. In Proceedings of the 14th World Congress of International Federation of Automatic Control, pp.34-45, 1999.

[Zave, 1997] P. Zave and M. Jackson. Four Dark Corners of Requirements Engineering. Em anais do XI Simpósio Brasileiro de Engenharia de Software, pp. 3-18. Fortaleza, Brasil, 1997.