RAISE - UM METAMODELO DE INFORMAÇÃO DE RASTREABILIDADE

PEDRO LOPES DA ROCHA LEAL JÚNIOR

RAISE - UM METAMODELO DE INFORMAÇÃO DE RASTREABILIDADE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Rodolfo Sérgio Ferreira de Resende

Belo Horizonte

07 de maio de 2011

Leal Júnior, Pedro Lopes da Rocha
L435m RAISE - Um Metamodelo de Informação de Rastreabilidade / Pedro Lopes da Rocha Leal Júnior. — Belo Horizonte, 2011
xxiv, 162 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Rodolfo Sérgio Ferreira de Resende

1. Computação - Teses. 2. Engenharia de Software - Teses. 3. Orientador. I. Título.

CDU 519.6*32(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

RAISE - Um metamodelo de informação de rastreabilidade

PEDRO LOPES DA ROCHA LEAL JÚNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. RODOLFO SÉRGIO FERREIRA DE RESENDE
Departamento de Ciência da Computação - UFMG

PROF. JOSÉ LUIS BRAGA
Departamento de Informática - UFV

PROF. CLARINDO ISAÍAS P. DA SILVA E PÁDUA
Departamento de Ciência da Computação - UFMG

PROF. EDUARDO MAGNO LAGES FIGUEIREDO
Departamento de Ciência da Computação - ICEx - UFMG

Belo Horizonte, 02 de junho de 2011.

Dedico este trabalho à minha esposa, Selma, que me deu forças e motivação para fazê-lo, e a minha mãe, Sandra, que me conduziu até este ponto.

Agradecimentos

Agradeço a minha esposa, Selma, minha mãe, Sandra, e aos meus irmãos, Leandra e Vinícius pela compreensão e apoio.

Ao meu afilhado, Artur, pelos momentos de descontração.

À Cintya Corgosinho Suzuki pela paciente revisão.

Ao Bruno pela compreensão e apoio.

À Spread Systems por disponibilizar espaço para a avaliação do trabalho.

Ao meu orientador, Rodolfo, pelos ensinamentos.

"Only in the context of a proposition
has a word a meaning."
(Ludwig Wittgenstein)

Resumo

Rastreabilidade, a habilidade de descrever e seguir a vida dos elementos de software, foi concebida como solução de um dos problemas fundamentais do processo de desenvolvimento de software: a garantia da conformidade do software com os seus requisitos. Apesar de já ser estudada há décadas, a Rastreabilidade ainda é pouco e mal usada. Inicialmente o estudo da Rastreabilidade se concentrou na Engenharia de Requisitos com o nome de Rastreabilidade de Requisitos, mas, nos últimos anos, a Rastreabilidade ganhou importância também para pesquisadores de outras áreas da Engenharia de Software. Essa diversidade de iniciativas na pesquisa de Rastreabilidade, trouxe, como consequência, que diferentes grupos de pesquisadores estabelecessem diferentes objetivos e granularidade em suas pesquisas. Como estes pesquisadores costumam ser parte de comunidades onde há pouca comunicação, o avanço na área é desbalanceado.

Diferentes métodos para manter a informação usada para a Rastreabilidade foram criados, mas nenhum ainda foi capaz de fazê-la ser usada sistematicamente. Os métodos representam a informação da Rastreabilidade através de modelos. Existem também muitas propostas de modelos diferentes, mas o estabelecimento de um modelo único para a Rastreabilidade não foi possível devido à dependência do contexto organizacional onde o modelo é usado. A gramática dos Modelos de Informação da Rastreabilidade pode ser definida através de um Metamodelo de Informação de Rastreabilidade. Esse metamodelo deve ser usado na criação de um Modelo de Informação de Rastreabilidade para cada organização ou projeto. O objetivo desse trabalho é apresentar o Metamodelo RAISE, que é um Metamodelo de Informação de Rastreabilidade que permite o registro do contexto onde a informação de rastreabilidade está inserida. O RAISE contextualiza a Informação de Rastreabilidade através de elementos do processo de desenvolvimento de software de uma organização. Acreditamos com isso permitir que os registros da Rastreabilidade possam ser usados com base no seu contexto de utilização, assim como indicou Pohl.

Palavras-chave: Rastreabilidade, Rastreamento, Engenharia de Software, Engenha-

ria de Requisitos.

Abstract

Traceability, the ability to describe and follow the life of software elements, was conceived as a solution of the fundamental problems of software development process: ensuring software compliance with its requirements. Even though it's already been decades of studies, Traceability is still rarely defined and used. Initially, Traceability studies focused on the Requirements Engineering with the name of Requirements Traceability, but in recent years, Traceability has also gained importance for researchers from other areas of Software Engineering. This diversity of Traceability initiatives in research has brought as a result that different research groups would establish different goals and granularity to their research. As these researchers tend to be part of communities where there is little communication, advances in Traceability are unbalanced.

Different methods used to keep the information for Traceability have been created, but none of them has yet been able to make it be used systematically. These methods use models to represent the Traceability information. There are also many proposals for different models, but the establishment of a single model for Traceability has not been possible due to the dependence of the organizational context where the model is used. The syntax and semantics of information models of Traceability can be defined through a metamodel for Traceability Information. This metamodel should be used in creating a model of Traceability Information for each organization or project. This dissertation presents RAISE, a metamodel that allows the recording of comprehensive Traceability Information and its context. RAISE contextualizes Traceability through elements of a software development process from an specific organization. We believe this allows the use of records of Traceability in a more effective and efficient way.

Lista de Figuras

2.1 Exemplo de Metamodelo 20
2.2 Metamodelo de Informação de Rastreabilidade Ramesh e Jarke (adaptado de Ramesh e Jarke) [Ramesh & Jarke, 2001] 25
2.3 Metamodelo de Informação de Rastreabilidade Espinoza e outros (adaptado de Espinoza e outros [Espinoza et al., 2006]) 26
2.4 Metamodelo de Informação da Rastreabilidade MDA-OMG (adaptado da OMG) 30
2.5 Metamodelo de Informação da Rastreabilidade Winkler baseado em modelos MDE (adaptado de Winkler e Pilgrim [Winkler & von Pilgrim, 2009]) . . . 31

3.1 Metamodelo dos conceitos de modelagem de negócio proposto por Eriksson/Penker (extraído do livro "*Business Modeling with UML: Business Patterns at Work*" [Eriksson & Penker, 2000]) 40
3.2 SPEM 2.0 (Extraído do documento "*Software & Systems Process Engineering Meta-Model Specification*" [OMG, 2008b]) 42
3.3 A composição Nested Breakdown Element do modelo SPEM (Adaptado do documento "Software & Systems Process Engineering Meta-Model Specification" [OMG, 2008b]) 43
3.4 Apresentação parcial do Metamodelo RAISE 48
3.5 Exemplo de um modelo baseado no Metamodelo RAISE Parcial 49
3.6 Exemplo de um Rastro gerado pela tarefa Definir Escopo 50
3.7 Destaque dos tipos de Rastros no meta-modelo 53
3.8 Metamodelo de Informação da Rastreabilidade RAISE 56
3.9 Exemplo de Rastro de Inferência 57
3.10 Exemplo de Regra Estrutural 58
3.11 Exemplo da Restrição Operacional/Comportamental 59
3.12 Exemplo de Restrição de Estimulo/Resposta 59

4.1 Modelo RAISE do Rastro Ligação de Alocação de Requisitos 78
4.2 Especificação da Regra Requisito Alocado 79
4.3 Modelo RAISE do Rastro de Satisfação do Requisito 80
4.4 Regra de Negócio Satisfação do Requisito 81
4.5 Modelo RAISE mostrando os detalhes do Rastro "Ligação de Alocação de Questão" 82
4.6 Regra de Negócio Questão Alocada 83
4.7 Modelo RAISE da Ligação da Resolução da Questão 84
4.8 Modelo da Regra de Negócio "QuestãoResolvida" 84
4.9 Implementação da Ligação de Satisfação do Requisito no EMF 85
4.10 Núcleo do meta-metamodelo Ecore 86
4.11 Modelo RAISE para a Organização do Estudo de Caso 87
4.12 Registros do Modelo RAISE da organização 92
4.13 Diagrama Causa e Efeito para identificação das possíveis causas de serem encontrados mais elementos nos Projetos que utilizam o Modelo RAISE . 98
4.14 Gráfico de Dispersão de elementos pesquisados vs Número de Desenvolvedores 102
4.15 Dispersão entre Número de Desenvolvedores e Numero de elementos pesquisado 103

C.1 Evidência da Auditoria 1 160
C.2 Evidência da Auditoria 2 160
C.3 Evidência da Auditoria 3 161
C.4 Evidência da Auditoria 4 161
C.5 Formulário das Tarefas de Pesquisa da Rastreabilidade 162

Lista de Tabelas

1.1 Exemplo de Matriz de Rastreabilidade 2

4.1 Características dos Projetos 74
4.2 CONRAS - Requisitos x Componentes 79
4.3 CONRAS - Requisitos x Código 81
4.4 CONRAS - Questão x Componente 83
4.5 CONRAS - Questão x Código 83
4.6 Indicadores coletados nos projetos 94
4.7 Número de vezes que ocorreu a Execução das Tarefas 98
4.8 Tipo de pesquisa que o modelo satisfaz 99

Sumário

Agradecimentos **ix**

Resumo **xiii**

Abstract **xv**

Lista de Figuras **xvii**

Lista de Tabelas **xix**

1 Introdução **1**

1.1 Contextualização 2
1.2 Visão Geral do Trabalho 4
1.3 Objetivo do Trabalho 7
1.4 Organização do Trabalho 7

2 Métodos e Metamodelos de Rastreabilidade **9**

2.1 A Questão da Rastreabilidade 10
2.2 Conceitos básicos 12
2.3 Métodos para Manter a Rastreabilidade 19
2.4 Modelos de Informação da Rastreabilidade 22
2.5 Metamodelos de Rastreabilidade 23
2.6 Discussão Sobre os Métodos e os Metamodelos 32
2.7 Conclusão do Capítulo 34

3 Rastreabilidade no Metamodelo RAISE **37**

3.1 Objetivo do Metamodelo RAISE 38
3.2 Metamodelos de Processo de Desenvolvimento de Software 39
3.3 Metamodelo de Informação de Rastreabilidade RAISE 44
3.3.1 Descrição do Metamodelo RAISE 44

3.3.2 Elementos do Metamodelo RAISE 60
3.4 Considerações sobre o Metamodelo de Informação RAISE 67
3.4.1 RAISE na Automação da Engenharia de Software 67
3.4.2 Granularidade dos elementos rastreáveis 69
3.5 Conclusão do Capítulo 69

4 Avaliação, Aspectos Práticos e Operacionais 71
4.1 Contextualização 72
4.1.1 Formulário para Coleta de Dados da Utilização da Rastreabilidade 73
4.1.2 Projetos 74
4.1.3 Processos 75
4.2 Aspectos Práticos e Operacionais 77
4.2.1 Rastros encontrados para a Organização 77
4.2.2 Implementação do Modelo RAISE 85
4.3 Metodologia do Estudo de Caso 87
4.4 Subprocesso de Definição 88
4.4.1 Meta Específica 89
4.4.2 Questões 89
4.4.3 Indicadores 90
4.5 Subprocesso de Planejamento 91
4.6 Subprocesso Coleta dos Dados 92
4.7 Subprocesso de Interpretação 94
4.7.1 Resultados dos Indicadores 94
4.7.2 Ameaças à Validade do Estudo de Caso 96
4.7.3 Ameaças Internas à validade 96
4.7.4 Resultado da Interpretação 104
4.8 Conclusão 104

5 Conclusão 107
5.1 Considerações Gerais 108
5.2 Contribuições 109
5.3 Trabalhos Relacionados 109
5.4 Trabalhos Futuros 110
5.4.1 Adaptação dos métodos de Rastreabilidade 110
5.4.2 Área de Conhecimento Rastreamento 111
5.4.3 Utilização do Metamodelo RAISE na Automação da Engenharia de Software 112

5.4.4 Automação do Desenho ("*Design*") do Modelo de Informação da Rastreabilidade 112
5.4.5 Outras propostas de estudos 112

Referências Bibliográficas **115**

Apêndice A Pesquisas Realizadas **125**
A.1 Descrição das pesquisas 127
A.1.1 Quais são os códigos e motivadores diretamente afetados pela alteração do requisito? 127
A.1.2 Quais são os códigos e motivadores indiretamente afetados pela alteração do requisito? 127
A.1.3 Qual é a razão do não cumprimento do Planejado? 128
A.1.4 O código que foi alterado é o previsto para ser alterado? 128
A.1.5 Quais são os outros requisitos afetados pela alteração? 128
A.1.6 "Quem?", "Quando?", "Por que?" da alteração 128
A.1.7 Quais são os requisitos alocados para a entrega? 129
A.1.8 Quais são os Requisitos que estão em estado de implementação? 129
A.1.9 Quais são os requisitos já satisfeitos pelo código até o momento? 129
A.1.10 Quais são os agrupamentos de questões que foram afetadas pela alteração? 129
A.1.11 Quais são as razões de se retirar questões do escopo da versão? 129
A.1.12 Quais questões foram canceladas e por quê? 130
A.1.13 Quais questões estão em implementação? 130
A.1.14 Quais questões estão implementadas? 130
A.1.15 Quais questões foram incluídas na versão? 130
A.1.16 Quais questões foram planejadas no início da versão? 130
A.1.17 Quais questões estão planejadas para a versão? 131
A.1.18 Quais questões foram retiradas do planejamento da versão? . . 131
A.1.19 Quais são as razões da inclusão de questão no planejamento da versão? 131
A.1.20 Quais são as razões para o cancelamento de questões? 131
A.1.21 Quais são os requisitos afetados pela questão planejada? 131
A.1.22 Quais são os requisitos afetados pela questão resolvida? 132

Anexo A Descrição das Tarefas do Estudo de Caso **133**
A.1 Identificar e Cadastrar os Requisitos e os Casos de Uso 134
A.2 Analisar a Questão 136

A.3 Especificar Casos de uso e Requisitos 138
A.4 Controlar Alterações 140
A.5 Verificar Se Os objetivos Estão Sendo Alcançados 142
A.6 Implementar Componentes 143
A.7 Implementar a Questão Técnica 145
A.8 Acompanhar o Projeto 147
A.9 Projetar Caso de Uso 150

Anexo B Notas das Entrevistas **153**
B.1 Entrevista Gerente de projeto 1 154
B.2 Entrevista Gerente de projeto 2 154
B.3 Entrevista com desenvolvedor 1 155
B.4 Entrevista com desenvolvedor 2 156
B.5 Entrevista com desenvolvedor 3 157
B.6 Entrevista com desenvolvedor 4 157

Anexo C Evidências do Estudo de Caso **159**
C.1 Evidências das Auditorias 160

Capítulo 1

Introdução

1.1 Contextualização

A evolução e a complexidade do desenvolvimento de software levaram ao longo das últimas décadas ao surgimento de uma área do conhecimento voltada à especificação, desenvolvimento e manutenção de sistemas de software: a Engenharia de Software [Pressman, 1995]. Dentre outros aspectos, a Engenharia de Software estuda a aplicação sistemática, disciplinada e quantificável de abordagens para o desenvolvimento, operação e manutenção do software [IEEE, 1990]. Essas diferentes abordagens manipulam elementos estruturados em artefatos correlacionados. A habilidade de descrever e seguir a vida desses elementos é conhecida como Rastreabilidade [Lago et al., 2009].

A comunidade de estudo de Engenharia de Requisitos é a responsável pela maior parte das pesquisas em Rastreabilidade [Pohl, 1993]. Seus pesquisadores classificaram-na como um atributo de qualidade de requisito denominada Rastreabilidade de Requisitos. A Rastreabilidade de Requisitos foi concebida como solução de um dos problemas fundamentais do processo de desenvolvimento de software: a garantia da conformidade do software com os seus requisitos, ou, em outras palavras, a garantia de que o que está sendo desenvolvido é exatamente o que foi solicitado.

Uma forma simples de se registrar a Rastreabilidade de Requisitos é através de uma matriz de relacionamento entre requisitos e outros elementos da engenharia de software, chamada de Matriz de Rastreabilidade. Por exemplo, considere um dos requisitos funcionais de um sistema de controle de vendas e de compras de uma mercearia, "*REQ001 - Controle de entrada e saída de mercadorias deve ocorrer manualmente e ser totalmente registrado no sistema*". Considere também o componente "*COMP001 - Controlador de Estoque*", que é um dos componentes lógicos do sistema em questão. Esse componente ajuda a satisfazer o requisito REQ001. A matriz de Rastreabilidade registra o relacionamento entre REQ001 e o COMP001, que pode ser denominado de Rastro ou de Ligação de Rastreabilidade entre REQ001 e COMP001. A Tabela 1.1 mostra esse Rastro.

Exemplo de Matriz de Rastreabilidade				
Requisitos	**Componentes**			
	COMP001	**COMP002**	**COMP003**	**COMP004**
REQ001	X			
REQ002				
REQ003				
REQ004				

Tabela 1.1. Exemplo de Matriz de Rastreabilidade

A Matriz de Rastreabilidade pode ser utilizada em tarefas com diferentes propósitos. Um exemplo seria a análise de impacto de uma alteração de requisito. Caso o requisito "*REQ001 - Controle de entrada e saída de mercadorias deve ocorrer manualmente e ser totalmente registrado no sistema*" sofra a alteração para "*REQ001 - Controle de entrada e saída de mercadorias ou* ***grupo de mercadorias*** *deve ocorrer manualmente e ser totalmente registrado no sistema*", a matriz pode informar quais são os componentes afetados pela alteração durante a análise de impacto. A Matriz de Rastreabilidade é apresentada na Seção 2.3 e é usada na avaliação do metamodelo proposto neste trabalho descrito no Capítulo 4.

Das práticas que pertencem à Engenharia de Requisitos (Elicitação, Análise, Documentação e o Gerenciamento), o Gerenciamento de Requisitos exige um cuidado especial [Nuseibeh & Easterbrook, 2000]. O modelo CMMI-DEV [Chrissis et al., 2006] possui uma área de conhecimento dedicada ao gerenciamento de requisitos (REQM - "*Requirements Management*"). O Gerenciamento de Requisitos é o responsável por controlar o ciclo de vida do requisito. Para essa área, as práticas de Rastreabilidade de Requisitos são as principais atividades utilizadas.

O uso da Rastreabilidade de Requisitos ganhou uma grande importância no desenvolvimento de software. A prática já é recomendada por diferentes normas e padrões de qualidade: CMMI [Chrissis et al., 2006], ISO/IEC-15504 [El-Emam & Garro, 1999], IEEE-830 [Gonçalves et al., 1998], MIL-STD-2167A [DoD (Department of Defense), 1988], MIL-STD-498 [DoD (Department of Defense), 1994], ISO/IEC 25000 [ISO, 2005]. Ramesh e Jarke [Ramesh & Jarke, 2001] afirmam que algumas organizações consideram a Rastreabilidade de Requisitos uma área estratégica. Em 2001, o Departamento de Defesa Norte Americano dedicou à Rastreabilidade quatro por cento de seu orçamento destinado à Tecnologia da Informação [Genvigir, 2009].

Nos últimos anos, a Rastreabilidade na sua forma mais genérica (a habilidade de descrever e seguir a vida dos elementos de software) ganhou importância também para pesquisadores de outras áreas da Engenharia de Software [Winkler & von Pilgrim, 2009]. Por exemplo: Desenho ("*Design*") de Software, Engenharia do Conhecimento, Gerência da Engenharia de Software, Gerência de Processos. Isso ocorreu por haver necessidades específicas dessas áreas de conhecimento para rastreamento de seus artefatos e, em muitos casos, sem haver um vínculo explícito com a Rastreabilidade de Requisitos. Algumas áreas de conhecimento iniciaram suas pesquisas em Rastreabilidade sem perceber a existência de iniciativas semelhantes em outras comunidades [Winkler & von Pilgrim, 2009]. Um exemplo está nos estudos do desenvolvimento orientado a modelos (Model-Driven Engineering - MDE) [Schmidt, 2006]. Nesse caso, a Rastreabilidade é usada para manter os modelos consistentes entre si e suportar a propagação

da alteração entre modelos derivados. A Rastreabilidade no MDE tem sido investigada de forma independente da Rastreabilidade de Requisitos.

Essa diversidade de iniciativas na pesquisa de Rastreabilidade levou pesquisadores de comunidades com pouca comunicação entre si a estabelecerem avanços desbalanceados na área [Winkler & von Pilgrim, 2009].

A literatura da área foca não apenas em aspectos da Rastreabilidade, mas no Rastreamento. Ou seja, não se está discutindo apenas o atributo de qualidade que indica a habilidade de descrever e seguir a vida de elementos da Engenharia de software, mas a ação de rastrear os elementos como uma área de conhecimento: a disciplina de Rastreamento [Pinheiro, 2003; Toranzo et al., 2002; Zisman et al., 2003]. Apesar disso a maioria dos autores ainda utiliza o atributo de qualidade Rastreabilidade para discutir o Rastreamento [Antoniol et al., 2005; Amar et al., 2008; Kelleher, 2005]. Essa dissertação segue os principais autores na área [Ramesh & Jarke, 2001; Mader et al., 2009] e utiliza o termo Rastreabilidade mesmo quando se discute as técnicas e conceitos relacionados à disciplina de Rastreamento.

1.2 Visão Geral do Trabalho

Apesar de já ser estudada há décadas, a Rastreabilidade ainda é pouco e mal usada [Mader et al., 2009]. Essa é a questão que direcionou este trabalho.

A razão para isso pode ser encontrada na afirmação de Egyed e outros de que, apesar dos avanços significativos nas pesquisas em Rastreabilidade, a criação de Ligações de Rastreabilidade ainda continua sendo uma tarefa que requer muito esforço humano ("*human-intensive*") [Egyed et al., 2010]. Como exemplo, considere o Componente COMP001 da Tabela 1.1 que satisfaz o requisito REQ001. Suponha que esse componente também satisfaça dezenas de outros requisitos. Suponha ainda que o referido componente sofresse uma refatoração e fosse completamente eliminado do sistema dando lugar a outros componentes. Nesse cenário, deve-se avaliar quais componentes satisfazem cada um dos requisitos. Como normalmente um sistema possui dezenas de requisitos e componentes, a matriz frequentemente possui dimensões elevadas. Os trabalhos de excluir o antigo componente, criar os novos e rever cada ligação se tornam bastante árduo. Muitas vezes até mesmo maior que a própria manutenção em si.

Esse exemplo expõe outra consideração: qual é a granularidade que deve ser usada nos elementos componentes das ligações de Rastreabilidade? Bianchi e outros [Bianchi et al., 2000] mostraram que a eficiência do processo de manutenção da Informação de Rastreabilidade está relacionada com a granularidade dos elementos envolvidos. Caso

se trabalhe com uma granularidade muito fina a manutenção na informação da Rastreabilidade fica inviável. Caso se trabalhe com pouca granularidade a Rastreabilidade perde sua eficácia.

A pesquisa fragmentada na área e a complexidade dos contextos envolvidos não estão permitindo o desenvolvimento de uma semântica de recuperação, uma semântica de manutenção, uma semântica operacional adequada para as necessidades das partes interessadas [Aizenbud-Reshef et al., 2005]. Assim, questões estruturais e dinâmicas da Rastreabilidade ficam sem respostas adequadas: Como atualizar as Ligações de Rastreabilidade quando um dos elementos ligados for atualizado? Quais formas de recuperação de informação da Rastreabilidade devem existir? Quais devem ser as políticas de criação, atualização e remoção das Ligações de Rastreabilidade e seus registros? Quando existem linhas de produto de software, como as ligações comuns e específicas devem ser administradas?

Existem diferentes métodos que se propõem manter a informação usada para a Rastreabilidade [Aizenbud-Reshef et al., 2006; Evans, 1989; Davis, 1993; West, 2002; Bowen et al., 1990; Cooke & Stone, 1991; Lefering, 1993; Tang, 1997; Smithers et al., 1991]. Cada um deles com um escopo de atuação diferente, e, por isso, nenhum foi capaz de fazer a Rastreabilidade ser largamente usada. A Seção 2.3 lista alguns desses métodos.

Para tornar factível o trabalho de mestrado não foi considerado dentro do escopo a adoção ou definição de métodos ou processos de rastreabilidade que resolvam a questão. Nem tão pouco, propor semânticas de recuperação, de manutenção ou operacional. Mas, colaborar na solução através de uma melhor representação da informação utilizada nos métodos de Rastreabilidade.

Os métodos utilizam diferentes formas de representação da informação de Rastreabilidade. São os Modelos de Informação da Rastreabilidade. Existem muitas propostas de modelos diferentes [Ramesh & Jarke, 2001; Espinoza et al., 2006; Lee et al., 2003; Richardson & Green, 2003; Ramesh et al., 2002; Pinheiro, 2003; Almeida et al., 2006; Drivalos et al., 2008; Amar et al., 2008; Pohl, 1996] (a Seção 2.4 lista algumas delas) e nenhuma efetivamente mais usada que outras. Uma razão para isso é que o Modelo de Informação da Rastreabilidade depende do contexto de sua utilização.

Cada organização possui um processo de desenvolvimento de software particular. Mesmo se estiver usando algum processo baseado em um arcabouço de mercado como o Rational Unified Process [Kruchten, 2003], a organização precisa personalizá-lo para sua realidade. Além disso, cada organização pode utilizar um conjunto de ferramentas diferentes para apoiar o processo de desenvolvimento. O modelo de Rastreabilidade utilizado numa organização precisa considerar essas particularidades. Por exemplo, ob-

servando uma organização que trabalhe com um desenvolvimento orientado a objetos: essa organização modela suas classes numa ferramenta de diagramação que utiliza a linguagem UML. O modelo de Rastreabilidade usado por ela deve ser diferente de outra organização que trabalhe com paradigmas de programação procedimental e modelagens baseadas no modelo de Entidades e Relacionamentos. No primeiro caso, requisitos são ligados a classes, enquanto no segundo, requisitos são ligados a entidades e procedimentos. Sendo assim, como organizações possuem realidades diferentes, deve-se conceber modelos diferenciados para cada organização.

Os modelos particularizados de cada organização podem ser concebidos a partir de um único Metamodelo de Informação de Rastreabilidade. O Metamodelo de Informação de Rastreabilidade descreve a gramática a ser usada nas definições de Modelos de Informação de Rastreabilidade específicos. Já existem algumas propostas de metamodelos [Ramesh & Jarke, 2001; Espinoza et al., 2006; Spanoudakis et al., 2004; Lago et al., 2004; Winkler & von Pilgrim, 2009; Genvigir, 2009] (vide Seção 3.3), que também não são sistematicamente utilizadas [Mader et al., 2009]. Nenhum dos metamodelos revisados para esta dissertação apresenta uma forma estruturada de se registrar o contexto onde a Informação da Rastreabilidade está inserida. Por exemplo: o metamodelo proposto por Ramesh e Jarke [Ramesh & Jarke, 2001] não fornece elementos para se registrar qual era a tarefa que estava sendo executada que culminou a geração de um determinado rastro. O mesmo ocorre no metamodelo de Espinoza [Espinoza et al., 2006]. Nesse caso se registra o tipo do Rastro com mais informações (Descrição, propósito, usos, exemplos), mas não há uma forma estruturada de se registrar que operação de negócio é necessária ocorrer para que este rastro seja gerado. Já no metamodelo proposto por Winkler e Pilgrim [Winkler & von Pilgrim, 2009] há a definição da metaclasse Contexto onde se pode vincular outro metamodelo que estrutura o contexto do rastro. Mas o próprio metamodelo de Winckler e Pilgrim não estrutura o contexto do rastro. Sendo assim, para registrar a tarefa executada que gerou um rastro é necessário modelar as tarefas em outro metamodelo e vincular esse metamodelo ao contexto do rastro no metamodelo de Winckler e Pilgrim.

Para Pohl [Pohl, 1996] a falta de uso da Rastreabilidade está no fato que apenas Informação de Rastreabilidade orientadas ao conteúdo é base para o uso apropriado. Para se usar a informação registrada de forma apropriada é necessário que ela esteja encapsulada em seu contexto. A Matriz de Rastreabilidade da Tabela 1.1 apresenta a ligação entre o requisito REQ001 e o componente COMP001 sem nenhuma informação do contexto de sua criação. "Quem criou", "quando criou", "porque criou" e "com que finalidade criou" são informações que não estão registradas, mas são informações importantes para a análise de impacto de uma alteração em um dos elementos do

Rastro, por exemplo.

1.3 Objetivo do Trabalho

O objetivo desse trabalho foi conceber um Metamodelo de Informação de Rastreabilidade que permite o registro do contexto onde a Informação de Rastreabilidade está inserida. O metamodelo foi denominado de **RAISE** (*tRaceability-Aided Integrated Software Engineering*) por permitir a integração entre as diversas ferramentas que suportam a Engenharia de Software. O RAISE utiliza meta-elementos do processo de desenvolvimento de software para estruturar o registro do contexto da informação da Rastreabilidade. Essa abordagem foi utilizada por entendermos que a Rastreabilidade está inserida no processo de desenvolvimento de software que a organização utiliza, e esse processo dita o contexto da criação e utilização da Rastreabilidade. No nosso ponto de vista, o amplo uso da Rastreabilidade só irá se tornar uma realidade quando a informação de Rastreabilidade estiver vinculada aos diferentes aspectos dos processos de desenvolvimento de software.

1.4 Organização do Trabalho

O Capítulo 2 discute a questão da Rastreabilidade, "por que ainda é pouco e mal usada?", e suas implicações. Os conceitos básicos que foram usados no desenvolvimento do trabalho são apresentados, contextualizando-os nas áreas de conhecimento onde foram moldados. Apresenta os métodos usados para manter a Rastreabilidade e os Modelos de Informação da Rastreabilidade propostos pelas comunidades. Os metamodelos existentes são então apresentados e suas limitações discutidas.

No Capítulo 3 o metamodelo RAISE é apresentado. A abordagem utilizada para a concepção do modelo é mostrada. Alguns elementos dos metamodelos de processos que foram usados como base da criação do RAISE são apresentados. Por fim, faz-se algumas considerações sobre o metamodelo RAISE.

No Capítulo 4 o metamodelo RAISE é avaliado. Infelizmente não foi possível executar uma validação do metamodelo por haver restrição de recursos e tempo. Primeiramente, a metodologia utilizada na avaliação é apresentada. Então, mostra-se o resultado de cada subprocesso da metodologia usada: Definição, Planejamento, Coleta de Dados e Interpretação. É na Interpretação que os aspectos práticos e operacionais do Metamodelo RAISE são discutidos.

No Capítulo 5 a conclusão do trabalho é apresentada, os resultados, contribuições e algumas oportunidades para trabalhos futuros são discutidas.

Capítulo 2

Métodos e Metamodelos de Rastreabilidade

Neste capítulo é apresentada a Rastreabilidade e as principais questões relacionadas ao seu uso. Os conceitos importantes e utilizados neste trabalho são descritos aqui, explorando principalmente o contexto onde estão inseridos. As técnicas propostas e os metamodelos existentes para manter a rastreabilidade são discutidos. Pretende-se, desta forma, contextualizar o conhecimento necessário para o entendimento do atual estado de desenvolvimento da Rastreabilidade na Engenharia de Software.

2.1 A Questão da Rastreabilidade

A Rastreabilidade na Engenharia de Software é uma disciplina emergente e muitos de seus termos e definições ainda não estão inteiramente claros [Winkler & von Pilgrim, 2009]. Como já citado na Seção 1.1, a Rastreabilidade foi concebida como solução de um dos problemas fundamentais do processo de desenvolvimento de software: a garantia da conformidade do software com os seus requisitos, ou, em outras palavras, a garantia de que o que está sendo desenvolvido é exatamente o que foi solicitado. A Rastreabilidade apresentou problemas desde cedo. Em 1994, Gotel e Finkelstein [Gotel & Finkelstein, 1994] listaram uma série de questões que necessitavam ser resolvidas para que a Rastreabilidade pudesse ser largamente utilizada. Gotel, quinze anos depois, é coautor de Mader e outros em "Getting Back to Basics: Promoting the Use of a Traceability Information Model in Practice" [Mader et al., 2009], e afirma que a Rastreabilidade ainda é pouco e mal usada. Nesta dissertação, o fato de Rastreabilidade ser pouco e mal usada é discutido e referenciado como a "Questão da Rastreabilidade".

A essência da Questão da Rastreabilidade permeia o fato que manter a informação necessária para a Rastreabilidade é trabalhoso e consome tempo [Antoniol et al., 2005]. Pohl [Pohl, 1996] detalha o problema do uso da Rastreabilidade em três principais aspectos, que são: primeiro aspecto, o uso da informação de Rastreabilidade depende das Partes Interessadas e das tarefas do processo de desenvolvimento de software, isto é, informação não é usada como é registrada e, consequentemente, recuperações seletivas, de acordo com necessidades atuais precisam ser suportadas. Por exemplo, na tarefa de Análise de Impacto de uma alteração de um requisito, as seguintes recuperações de informação da Rastreabilidade são necessárias: todos os elementos ligados ao requisito alterado, outros requisitos indiretamente afetados, elementos indiretamente afetados, Partes Interessadas nos elementos afetados. Essa informação recuperada é registrada em tarefas distintas de forma distinta. Segundo aspecto, devido ao grande volume de informação produzida durante o processo, apenas ligações orientadas ao conteúdo são base para o uso apropriado, isto é, o uso da informação registrada só é apropriado

se a informação de Rastreabilidade está encapsulada em seu contexto. Por exemplo, ao se recuperar uma ligação de contradição entre dois requisitos, deve-se entender: quando a contradição foi identificada? Quem são as Partes Interessadas para resolver a contradição? A contradição já foi resolvida? Como foi resolvida? Sem esse contexto a informação não é usada de forma apropriada. Terceiro aspecto, as pessoas envolvidas na captura de informação de Rastreabilidade são frequentemente diferentes dos usuários da informação. Como no caso da análise de impacto de uma alteração. A informação de Rastreabilidade utilizada foi registrada pelos desenvolvedores que frequentemente não participam da análise.

A busca para se resolver a Questão da Rastreabilidade e obter uma boa utilização está fragmentada em diferentes comunidades, o que prejudica a solução do problema. Isso ocorreu devido à forma como a disciplina foi inserida na Engenharia de Software.

A Rastreabilidade surgiu na Engenharia de Requisitos, o que justifica o fato de até hoje se encontrar a expressão "Rastreabilidade de Requisitos" quando, em alguns casos, se quer falar de Rastreabilidade em geral. Essa diferenciação é discutida no artigo de Winkler e Pilgrim [Winkler & von Pilgrim, 2009]. Já está claro que a Rastreabilidade é uma área de pesquisa que transcende a Disciplina de Requisitos e alcança todo o ciclo de desenvolvimento de software, apesar de ainda ser muito comum encontrá-la vinculada à requisitos. As primeiras pesquisas feitas pela comunidade da Engenharia de Requisitos focaram principalmente nas áreas de processo de validação e verificação. Com o tempo, outras comunidades de pesquisadores também mostraram interesse na Rastreabilidade, em especial no campo de MDE (Model-Driven Engineering), onde as ligações de Rastreabilidade ganham uma atenção especial por indicarem as transformações dos elementos. Como exemplo, uma classe de nome "ControladorDeEstoque" de um modelo independente de plataforma pode ser transformada nas classes "ControladorDeEstoqueAction" e "ControladorDeEstoqueForm" em um modelo dependente de plataforma. Informação de Rastreabilidade que vincula a classe de origem com as classes de destino devem ser mantidas para permitir as alterações nos modelos. O artigo "Model-driven engineering" de Douglas C. Schmidt [Schmidt, 2006] apresenta o campo de estudo MDE. Apesar de existirem algumas variações do MDE, como o MDD (*Model-driven Development*) e o MDA (*Model-Driven Architecture*), por simplificação, estas variações são tratadas neste trabalho como pertencentes à classe do MDE, já que não há diferenças significativas entre elas do ponto de vista da Rastreabilidade.

As pesquisas da Rastreabilidade na comunidade MDE foram desenvolvidas de forma independente das pesquisas de Rastreabilidade de Requisitos por terem focos distintos e não se mostrarem parte de um todo maior. A percepção de uma unificação do estudo da Rastreabilidade só ocorreu mais recentemente [Winkler & von Pilgrim,

2009] e ainda não é uma realidade.

Ainda há indefinições sobre alguns conceitos utilizados na Rastreabilidade, principalmente devido à fragmentação dos esforços das pesquisas. A seção 2.2 apresenta os conceitos básicos que são usados no restante deste trabalho contextualizados em suas áreas de pesquisas.

2.2 Conceitos básicos

"Rastreabilidade"

A maioria das conceituações existentes de Rastreabilidade está contextualizada em alguma comunidade. Isso ocorre porque as pesquisas em Rastreabilidade são realizadas para resolver questões vinculadas a diferentes áreas de pesquisa. Apesar disto, existem algumas definições genéricas do termo. É o caso do *IEEE Standard Glossary of Software Engineering Terminology* [IEEE, 1990], onde a Rastreabilidade é definida como:

> 1. O grau em que cada relacionamento pode ser estabelecido entre dois ou mais produtos do processo de desenvolvimento, especialmente a relação predecessor-sucessor ou mestre-subordinado. [...]
> 2. O grau em que cada elemento do produto do desenvolvimento de software estabelece sua razão de existir. [1]

Esta definição utiliza o termo "elemento" com o mesmo valor de "artefato", que é mais comumente usado nos artigos de Rastreabilidade, como por exemplo, em Gotel e Finkelstein [Gotel & Finkelstein, 1995] e Ramesh e Jarke [Ramesh & Jarke, 2001]. Nesta dissertação o termo "artefato" é usado como uma instanciação de "elemento". Por exemplo, uma lista de requisitos é um elemento, enquanto o arquivo de nome ERSw - Especificação do Requisitos de Software, que contém a lista de requisitos, é um artefato.

No caso das pesquisas em MDE, utilizam-se os termos "modelo" ou "texto" como sinônimo de "elemento". Isso porque os elementos que são manipulados nessa área de pesquisa são principalmente modelos e textos. Essa generalização, apesar de errada,

[1] Do inglês:

a) The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another. [...]

b) The degree to which each element in a software development product establishes its reason for existing.

não provoca nenhum impacto nos textos do MDE, pois se opera a transformação de modelo para modelo ou de modelo para texto e somente estes são os artefatos manipulados. No caso desta dissertação o termo "modelo" não pode ser sinônimo de elemento, pois a dissertação trata de um contexto mais abrangente do que utilizado no MDE (vide abaixo, no item Modelo e Metamodelo, a definição de modelo usado neste trabalho).

Outro aspecto relevante da definição do *IEEE Standard Glossary of Software Engineering Terminology* é o identificado por Derniame e outros: elementos podem ser composições formadas de outros elementos recursivamente [Derniame et al., 1999].

No domínio da Engenharia de Requisitos, o termo Rastreabilidade é frequentemente sinônimo de "Rastreabilidade de Requisitos" e a definição mais comumente usada é a de Gotel e Finkelstein [Gotel & Finkelstein, 1994] que descreve a Rastreabilidade como:

> "A habilidade de descrever e seguir a vida do requisito, em ambas as direções, para frente e para trás."[2]

Uma definição de Rastreabilidade bastante citada pela comunidade de pesquisa MDE, é a do Paige e outros [Paige et al., 2008]:

> "[...] a habilidade de inter-relacionar cronologicamente entidades unicamente identificáveis da forma que importa [...] Refere à capacidade de rastrear artefatos através de um conjunto encadeado de operações [manuais ou automáticas]." [3]

Outra especificação de Rastreabilidade encontrada na comunidade MDE é a de Aizenbud-Reshef e outros [Aizenbud-Reshef et al., 2005], que foi baseada na de Gotel e Finkelstein da comunidade da Engenharia de Requisitos:

> "Qualquer relacionamento que existe entre artefatos envolvidos no ciclo de vida na Engenharia de Software. Esta definição inclui, mas não está limitada às seguintes ligações:
> - Ligações Explícitas ou mapas que são gerados como um resultado da transformação, para frente (geração de código) ou para trás (engenharia reversa);

[2]Do inglês: "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction."

[3]Do inglês: "[...] the ability to chronologically interrelate uniquely identifiable entities in a way that matters. [...] [It] refers to the capability for tracing artifacts along a set of chained [manual or automated] operations."

- Ligações que são computadas baseadas em informação existente (por exemplo, analise de dependência de código);

- Ligações inferidas estatisticamente. São ligações que são computadas baseadas no histórico provido pelo sistema de gerência de mudança nos itens que são alterados em conjunto como resultado de uma requisição de alteração." [4]

"Rastro" ("*Trace*")

Artefatos são criados e modificados como resultado de tarefas executadas durante o processo de desenvolvimento do software. Essas tarefas deixam Rastros que ligam os artefatos trabalhados.

Rastro é outro conceito relevante por ser o principal elemento de trabalho da Rastreabilidade. Segundo o IEEE Standard Glossary [IEEE, 1990], um Rastro é "(possivelmente) uma indicação não-material ou evidência mostrando o que existiu ou ocorreu"[5]. Se um desenvolvedor trabalha um artefato, ele deixa Rastros. Por exemplo, a biblioteca de gestão de configuração registra "quem" trabalhou no artefato; "quando" o artefato foi trabalhado; e pode ainda registrar "que partes" do artefato foram alteradas. Essas informações, "quem", "o quê", "quando" são atributos de um Rastro resultante da tarefa executada.

As pesquisas sobre o "Rastro" evoluíram bastante no campo da Engenharia de Requisitos. Os conceitos acima citados foram desenvolvidos e diversas formas de classificação dos Rastros foram geradas. Por exemplo, Pinheiro [Pinheiro, 2003] classificou o Rastro como funcional e não-funcional. Essa classificação, apesar de ser concebida para Rastreabilidade de Requisitos, pode ser aplicada em um contexto mais amplo.

Os Rastros funcionais são criados pela transformação de um artefato em outro usando um conjunto definido de regras. Essa transformação não precisa ser necessariamente automática como ocorre em sistemas MDE, mas deve seguir procedimentos sem ambiguidade.

Pinheiro categoriza como Rastros não-funcionais aqueles de natureza informal. São Rastros que se caracterizam por apresentarem a razão de um evento, ou por apresentarem o contexto em que o evento ocorreu ou por apresentarem as decisões tomadas.

[4] Do inglês: "any relationship that exists between artifacts involved in the software-engineering life cycle. This definition includes, but is not limited to the following: - Explicit links or mappings that are generated as a result of transformations, both forward (e.g., code generation) and backward (e.g., reverse engineering) - Links that are computed based on existing information (e.g., code dependency analysis) - Statistically inferred links, which are links that are computed based on history provided by change management systems on items that were changed together as a result of one change request."

[5] Do inglês: "(possibly) non-material indication or evidence showing what has existed or happened"

Já no campo do MDE, a OMG, no documento de especificação da infra-estrutura da UML [OMG, 2009] apresentou o Rastro assim:

> "Um Rastro[...] registra uma ligação entre um grupo de objetos do modelo de entrada e um grupo de objetos do modelo de saída. Essa ligação é associada com um elemento da especificação do modelo de transformação que relaciona os grupos."[6]

No contexto do MDE, Rastros exercem as mesmas funções que exercem na Engenharia de Requisitos. A característica especial do MDE está no uso de modelos e transformações automáticas. Nesse caso, os artefatos em estudo são sempre, ou quase sempre, modelos. Esse contexto influencia as definições dos elementos vinculados à Rastreabilidade.

Há na literatura diferentes denominações para o "Rastro". O termo "Rastro" ("*Trace*") é o mais usado [IEEE, 1990; Egyed et al., 2010; von Knethen, 2002; De Lucia et al., 2005], mas também se encontra "Objeto da Rastreabilidade" [Ramesh & Jarke, 2001], "Elo" [Genvigir, 2009], dentre outros.

"Ligações de Rastreabilidade"

Um Rastro pode, em parte, ser documentado como um conjunto de metadados de um artefato (como a data de criação e modificação, criador, modificador) e em parte como relacionamento entre artefatos [Winkler & von Pilgrim, 2009]. Particularmente, os relacionamentos representam um conceito vital para a Rastreabilidade, e são normalmente referenciados como Ligações de Rastreabilidade ("*traceability link*"). As Ligações de Rastreabilidade documentam as várias dependências, influências, causalidade, etc., existentes entre artefatos.

Gotel e Finkelstein [Gotel & Finkelstein, 1994] introduziram e enfatizaram a classificação das Ligações de Rastreabilidade: pré-especificação dos requisitos ("*pre-requirements specification*") pre-RS e pós-especificação do requisito ("*post-requirements specification*") post-RS. Pre-RS são as ligações de Rastreabilidade que ocorrem durante a elicitação, discussão, e acordos sobre os requisitos até eles serem registrados no documento de especificação. Post-RS são as ligações de Rastreabilidade vinculadas ao desenvolvimento dos requisitos em elementos de projeto e implementação. Gotel e Finkelstein argumentam que as ligações post-RS são mais simples de tratar que as pre-RS. Nas post-RS se deve apenas descobrir os passos da transformação dos modelos, que

[6]Do inglês: "A trace [...] records a link between a group of objects from the input models and a group of objects in the output models. This link is associated with an element from the model transformation specification that relates the groups concerned."

podem ser bastante formais se comparados com as ligações pre-RS. Pre-RS são Rastros que registram problemas de comunicação e informalidade e, portanto, mais complexos de serem tratados.

"Tipos de Ligações de Rastreabilidade"

As Ligações de Rastreabilidade podem ser categorizadas. Existem na literatura muitas propostas diferentes de tipos de ligações de Rastreabilidade.

Ramesh e Jarke [Ramesh & Jarke, 2001] propuseram os seguintes tipos básicos: Satisfação, Dependência, Evolução e Fundamentação.

- Satisfação e Dependência formam um grupo relacionado ao produto, que descrevem as propriedades e relacionamentos com os objetos.
- Evolução e Fundamentação compõem um segundo grupo relacionado ao processo, que pode ser capturado somente se observado o histórico das ações.

Já Toranzo e outros [Toranzo et al., 2002] pensaram nos tipos:

- Satisfação: indica que o elemento de origem satisfaz o elemento destino. Por exemplo: um componente de software satisfaz um requisito;
- Recurso: o elemento de origem é um recurso do elemento destino. Por exemplo, uma proposta de mudança usa como recurso de informação os requisitos do sistema;
- Responsabilidade: aponta a participação, responsabilidade ou ação de participantes sobre os itens gerados;
- Representação: registra a representação ou modelagem dos requisitos em outras linguagens;
- Alocação: representa que elementos de origem são reservados para serem usados por elementos destino. Por exemplo: requisitos são alocados em componentes de software; e
- Agregação: representa a composição entre elementos.

Pohl [Pohl, 1996], em seu modelo de dependência, define dezoito diferentes ligações de dependência que são categorizados em cinco categorias:

- Condição,
- Conteúdo,

- Documentos,
- Evolução e
- Abstração.

De Lucia e outros [De Lucia et al., 2005] apresentam três tipos de ligações:

- Dependência: é uma ligação direta entre um artefato cliente e um artefato supridor. O artefato cliente depende ou é impactado por mudanças no artefato supridor;
- Indireto: indica artefatos que impactam uns aos outros; e
- Composição: um artefato cliente é parte de um artefato supridor, os quais são recuperados por uma ferramenta de recuperação, mas não rastreados por usuários.

Spanoudakis e outros [Spanoudakis et al., 2004] identificaram quatro tipos de ligações entre os elementos Padrões de Requisitos, Casos de Uso e Modelos de Análise de Objetos.

- Ligação de Sobreposição: indica que os elementos conectados representam uma característica comum do sistema ou de seu domínio.
- Ligação de Execução Requerida: é uma ligação entre artefatos e operação. Indica que a sequência dos artefatos requer a execução da operação a que se relaciona.
- Ligação Característica Requerida: indica a relação entre uma parte de uma especificação de um Caso do Uso e um requisito ou entre dois requisitos.
- Ligação Parcialmente Realizável: indica que o Caso de Uso executa parcialmente um determinado requisito.

Já em 2005, Spanoudakis e Zisman [Spanoudakis & Zisman, 2005] identificaram oito tipos de ligações:

- Ligação de Dependência: um elemento “e1” depende de um elemento “e2” se a existência de “e1” baseia-se na existência de “e2”, ou se alterações em “e2” têm de ser refletidas em “e1”.

- Ligação de Generalização ou Refinamento: este tipo de relação é usado para identificar como elementos complexos de um sistema podem ser quebrados em componentes, e como elementos de um sistema podem ser combinados para formar outros elementos, ou ainda, como um elemento pode ser refinado por outro elemento.

- Ligação de Evolução: Relações desse tipo indicam evoluções nos elementos de Engenharia de Software.

- Ligação de Satisfação: um elemento "e1" satisfaz um elemento "e2" se "e1" atende as expectativas, necessidades e desejos de "e2".

- Ligação de Sobreposição: um elemento "e1" sobrepõe um elemento "e2" se "e1" e "e2" se referem a uma característica comum de um sistema ou de um domínio.

- Ligação de Conflito: esse tipo de ligação evidencia conflito entre dois elementos, como por exemplo, requisitos contraditórios.

- Ligação de Racionalização: ligação desse tipo é usada para representar e manter as razões por traz da criação e evolução de elementos, e decisões sobre o sistema em diferentes níveis de detalhes.

- Ligação de Contribuição: são usadas para representar associações entre artefatos e Partes Interessadas que contribuíram na geração de requisitos.

Aizenbud-Reshef e outros [Aizenbud-Reshef et al., 2006] definiram quatro tipos de ligações:

- Imposto: ocorre entre artefatos que existem pela violação do criador do relacionamento;

- Inferido: ocorre entre artefatos que satisfazem uma regra que descreve o relacionamento;

- Manual: é criado e mantido por um usuário humano;

- Computacional: é criado e mantido automaticamente por um computador e é dividido em duas categorias:

  - 1) Derivação: denota que, dado o conteúdo de um artefato, é possível computar a validade do conteúdo deste artefato,

– 2) Análise: é um tipo inferido de relacionamento criado pela análise dos programas que examinam códigos em um grupo de regras.

"Modelo e Metamodelo"

Neste trabalho "Modelo" tem o valor estabelecido por Mellor e outros: "Um modelo é um grupo coerente de elementos que descrevem algo construído através de alguma forma de análise para algum propósito particular, podendo ser expresso por alguma linguagem (textual ou gráfica) que por sua vez possui certo grau de abstração" [Mellor et al., 2003].

Ainda segundo Mellor e outros, a gramática da linguagem de modelagem pode ser definida através de outro modelo. Este modelo é chamado de metamodelo. Por exemplo, o modelo subjacente à linguagem UML é descrito através de um metamodelo que corresponde a um subconjunto da UML. A Figura 2.1 mostra a representação de um metamodelo muito simples. Possui apenas as meta-classes de nome "Classe" e "Associação". O metamodelo indica que uma meta-classe "Classe" pode se associar com diversas meta-classes "Associação", mas uma meta-classe "Associação" só pode se associar com duas meta-classes "Classe". Um modelo derivado do metamodelo do exemplo é apresentado na camada intermediária com três elementos: "Pessoa", "Carro" e uma conexão entre eles. Os elementos "Pessoa" e "Carro" são duas instâncias da meta-classe "Classe". O elemento que aparece conectando "Pessoa" e "Carro" é uma instância da meta-classe "Associação". Observe que a associação segue a semântica indicada no metamodelo: apenas se associa a duas "Classes". Uma representação de objetos instanciados do modelo é mostrada na camada inferior. Um objeto "Pedro" que é uma instância de "Pessoa" e um objeto "Carro do Pedro" que é uma instância de Carro estão representados na figura. A conexão entre "Pedro" e "Carro do Pedro" é uma instância da associação entre as respectivas classes. Uma referência de uma linguagem para definição de metamodelo é a OMG Unified Modeling Language [OMG, 2009].

A implementação de mecanismos para registrar e manter registros de Rastreabilidade conforme as especificações acima está vinculada a algum método. Diferentes métodos foram criados na tentativa de resolver os diferentes aspectos da Rastreabilidade. A seção 2.3 mostra a evolução desses métodos.

2.3 Métodos para Manter a Rastreabilidade

O estabelecimento e manutenção da informação da Rastreabilidade não é tarefa trivial, já que um grande volume de informação é manipulado. Diferentes métodos foram

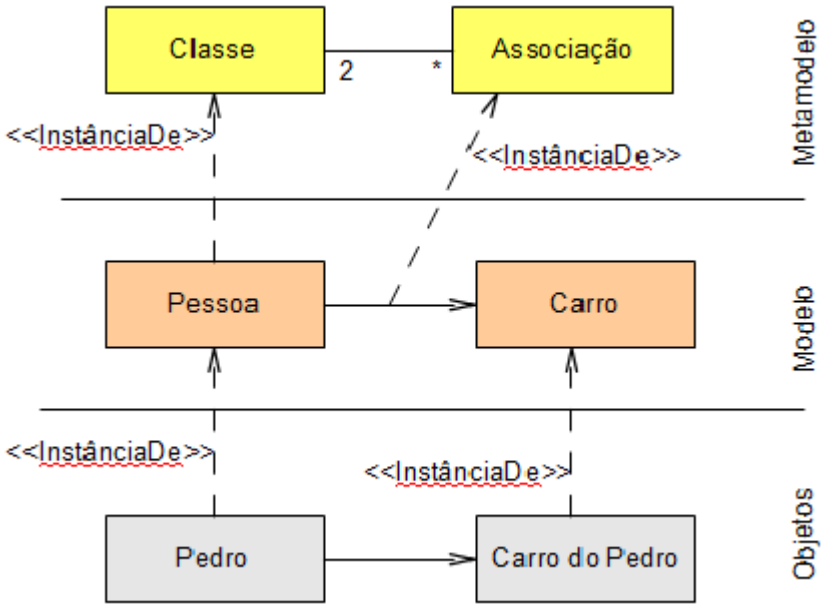


Figura 2.1. Exemplo de Metamodelo

concebidos na tentativa de se tornar viável essa manipulação.

O primeiro método usado, desenvolvido na década de 1970 [Aizenbud-Reshef et al., 2006], foi a referência cruzada [Evans, 1989]. O uso de matrizes [Davis, 1993; West, 2002] como método ocorreu naturalmente em seguida. A Matriz de Rastreabilidade se tornou um método muito usado nas organizações que empregam as práticas de Rastreabilidade. Isso ocorreu pelo fato de ser um método simples e de fácil entendimento. Até mesmos o modelo CMMI [Chrissis et al., 2006] sugere sua utilização como boa prática. O Capítulo 1 apresenta um exemplo de matriz e o Capítulo 4 mostra o seu uso em uma organização.

Como a Matriz de Rastreabilidade não se mostrou ser de fácil utilização [Mader et al., 2009], outras técnicas foram propostas:

- redes de confiança [Bowen et al., 1990];
- arcabouço de desenvolvimento formal [Cooke & Stone, 1991];
- integração incremental entre requisitos e programação [Lefering, 1993];
- hipertexto [Alexander, 2002; Glinz, 2000; Kaindl, 1993]
- sistemas baseados em conhecimento ("*knowledge-based*") [Tang, 1997; Smithers et al., 1991];

- grafos [Pinheiro & Goguen, 1996];
- esquemas dinâmicos [Cleland-Huang et al., 2003; Antoniol et al., 2000; Tryggeseth & Nytro, 1997];
- "*keyphrase based traceability*" [Jackson, 2002];

Métodos que automatizam parcialmente a detecção de existência de Ligações de Rastreabilidade foram criados devido à dificuldade em se conseguir manter as ligações. Como exemplo há o método proposto por Faisal [Faisal, 2005] e a recuperação por construção do Deng e outros [Deng et al., 2005]. Alguns métodos foram baseados nos conceitos de "*Information Retrieval*" - RI [Jung, 2007], outros em LSA - "*Latent semantic analysis*" [Lucia et al., 2004]. Alguns pesquisadores se dedicaram às técnicas baseadas em RI no intuito de mostrar que podem ser usadas para descobrir dinamicamente a existência de ligações de Rastreabilidade [Antoniol et al., 2000; Hayes et al., 2003; Marcus & Maletic, 2003; Spanoudakis, 2002], além de permitir o relacionamento entre vários tipos de artefatos [Deng et al., 2005] como código e documentação [Marcus & Maletic, 2003; Antoniol et al., 2002], requisitos e projeto [Hayes et al., 2003], artefatos e requisitos [Zisman et al., 2003]. Outra característica desses métodos automatizados é que eles podem prover a Rastreabilidade sem a necessidade de manter registros das ligações. Fazem isso, identificando os Rastros diretamente nos artefatos. Por exemplo, Antonial e outros [Antoniol et al., 2002] propuseram um método para recuperar Ligações de Rastreabilidade entre código fonte e documentação de texto livre. Usam identificadores extraídos do código fonte em pesquisas nas documentações para recuperar itens relevantes ao código. Eles assumem que os programadores utilizam palavras relevantes (ex.: nomes derivados do domínio da aplicação).

Muitos autores se concentraram em métodos que automatizam a identificação de Rastreabilidade diretamente entre o código fonte e outros artefatos. É o caso de Murphy e outros que propuseram o método de modelo de reflexão de software [Murphy et al., 2001]. Esse método checa a conformidade da implementação do código com o modelo de projeto usando regras de mapeamento entre o modelo de projeto e o modelo de origem extraído para o código de implementação. Outro trabalho nessa mesma linha é o de Richardson e Green [Richardson & Green, 2004], que usam uma síntese automatizada para inferir ligações de Rastreabilidade.

Automações da busca das ligações de Rastreabilidade ocorreram também com propósitos diferentes de manter ligações. É o caso do método de Chechik e outros [Chechik & Gannon, 2001] que automatizou a análise da consistência entre requisitos e

desenhos usando a identificação automática de ligações de Rastreabilidade entre esses artefatos.

Um Rastro não necessariamente é uma ligação, mas um registro de um evento. Alguns métodos são baseados em eventos [Cleland-Huang et al., 2003]. Eles podem ser usados para rastrear requisitos associados a desempenho em modelos executáveis de desempenho [Cleland-Huang et al., 2003, 2002], e também para rastrear a qualidade de requisitos implementados através de padrões de projetos já conhecidos [Gross & Yu, 2001], que podem ser ligados usando-se seus invariantes [Cleland-Huang & Schmelzer, 2003].

Muitos desses métodos manipulam informação de Rastreabilidade que deve ser registrada e mantida. Um dos caminhos que parece bastante promissor para a manutenção da informação da Rastreabilidade está na construção de modelos de informação de Rastreabilidade (*Traceability Information Model*) que é detalhada na seção 2.4.

2.4 Modelos de Informação da Rastreabilidade

O conceito de Mellor e outros para modelo [Mellor et al., 2003] apresentado na seção 2.2 pode ser adaptado para o contexto da Rastreabilidade. Tem-se que um modelo de informação da Rastreabilidade é uma forma coerente de abstração dos tipos de elementos que compõe ou se relacionam com um tipo de Rastro. Por exemplo, os artefatos dos tipos "Especificação de Requisito de Software" e "Código Fonte" se relacionam com o tipo de Rastro "Satisfaz".

Os Modelos de Informação de Rastreabilidade são usados como base conceitual na concepção dos métodos e na construção de ferramentas que objetivam permitir o registro e controle da Rastreabilidade. Um exemplo é o de Ramesh e Jarke [Ramesh & Jarke, 2001] que propuseram um modelo a partir de investigações do uso real da Rastreabilidade. O artigo cita ainda o uso extensivo desse modelo em uma ferramenta que, segundo os autores, tem certa relevância no mercado. Esse trabalho, um dos mais citados sobre Rastreabilidade, indica o que seria o primeiro esboço para a solução do problema da Rastreabilidade.

Como já dito, um metamodelo de informação de Rastreabilidade é aquele que apresenta a gramática da linguagem de modelagem do Modelo de Informação de Rastreabilidade.

Em todas as áreas de pesquisa há diferentes artigos propondo modelos e metamodelos de Rastreabilidade. Em alguns casos eles são apresentados na forma de esquemas (*"schemas"*) [Espinoza et al., 2006; Lee et al., 2003; Richardson & Green, 2003], em

outros casos, como modelos UML [Ramesh et al., 2002; Pinheiro, 2003; Genvigir, 2009], ou ainda podem ser apresentados em diversas outras linguagens [Almeida et al., 2006; Drivalos et al., 2008; Amar et al., 2008; Pohl, 1996].

A falta de um consenso sobre como representar a Rastreabilidade torna a discussão sobre o metamodelo, neste momento, mais relevante do que a discussão sobre o modelo. Os modelos são muito particulares de cada organização e apresentam grandes variações devido ao contexto onde estão inseridos, enquanto os metamodelos podem ser genéricos e aplicados aos modelos de várias organizações. A seção 2.5, a seguir, apresenta os metamodelos de Rastreabilidade propostos na literatura com maior número de referências até o momento.

2.5 Metamodelos de Rastreabilidade

Esta seção apresenta metamodelos de Rastreabilidade encontrados na literatura. Esses metamodelos mostram as formas de se definir as propriedades de um Rastro e suas relações com os artefatos da Engenharia de Software.

Pesquisadores, a maioria deles na área de Engenharia de Requisitos, tentaram encontrar as propriedades de um Rastro. Em muitos casos a solução é representada como um esquema "*schema*", que pela definição de Mellor e outros [Mellor et al., 2003], é um modelo. Em outros casos a solução é representada como um modelo UML (pesquisadores do campo de MDE o fazem com frequência).

Um dos primeiros e bastante simples metamodelo de Rastreabilidade foi concebido por Ramesh e sua equipe de pesquisadores [Ramesh & Jarke, 2001]. Eles se basearam na definição de Gotel e Finkelstein [Gotel & Finkelstein, 1994] (vide a seção 2.2) e evoluíram as características necessárias para o Metamodelo de Rastreabilidade baseando-se em estudos empíricos. Ampliaram à definição de Rastro do IEEE Standard Glossary [IEEE, 1990] (vide a seção 2.2) ao concluírem a existência de cinco dimensões da Rastreabilidade de requisitos. Na sua visão, cada Rastro deve responder às seguintes perguntas:

- Que informação é representada?
- Quem são as Partes Interessadas que executam papeis na criação e manutenção dos objetos da Rastreabilidade?
- Onde é representada?
- Como é representada?

- Por que certo objeto conceitual é criado ou evoluído?
- Quando a informação foi capturada, modificada ou evoluída?

Ramesh e sua equipe montaram um metamodelo que satisfaz estas dimensões. A Figura 2.2 mostra o metamodelo proposto (neste trabalho todos os metamodelos e modelos estão apresentados em UML para que se possa fazer um paralelo entre eles. Não se utilizou nenhum perfil UML para a descrição dos metamodelos. Os nomes dos elementos modelados foram mantidos conforme concebido por seus autores em inglês).

A dimensão "Que informação é representada?" indica ou evidencia o que aconteceu ou o que existiu e satisfaz a conceituação de Rastro do IEEE Standard Glossary. Nesta dimensão está contida toda a definição de Gotel e Finkelstein. No modelo proposto por Ramesh e Jarke, esta dimensão é representada pelo "Object" e "Trace to". "Objects" que são os artefatos de entradas e saídas do processo de desenvolvimento de software. Exemplos de tipos de objetos são Requisitos, Desenho de Software, Componentes do Sistema, Decisões, Alternativas. Representam elementos conceituais, para os quais a Rastreabilidade precisa ser mantida durante os vários estágios do ciclo de desenvolvimento. "Objects" são criados pelas tarefas do ciclo de desenvolvimento de software. As diferentes formas de ligações de Rastreabilidade que podem ocorrer através dos "Objects" são representadas pelas ligações "Trace to". Por exemplo, uma ligação do tipo "Dependência" (vide Seção 2.2, "Tipos de Ligação de Rastreabilidade") entre dois objetos pode ser representada como uma ligação do tipo "Trace to". Grosso modo, sob o ponto de vista da definição de Rastro de IEEE Standard Glossary [IEEE, 1990] pode-se inferir que elementos do tipo "Object" registram a indicação ou evidência do que existiu, enquanto as ligações "Trace to" indicam ou evidenciam o que aconteceu. Vale dizer que ligações "Trace to" também podem evidenciar o que existiu, já que podem ser especializadas em ligações do tipo "Composição", que mostra que um determinado objeto é parte de outro objeto.

A segunda dimensão, "Quem são as Partes Interessadas que executam papeis na criação e manutenção dos objetos da Rastreabilidade?", é uma novidade para a definição de Gotel e Finkelstein. Sua importância, assim como das próximas dimensões a serem listadas, foi identificada por diversas organizações que trabalhavam com Rastreabilidade e foram entrevistadas pela equipe de Ramesh e Jarke [Ramesh & Jarke, 2001]. No modelo, a meta-classe "Stakeholder" representa esta dimensão. Os "Stakeholders" são os agentes envolvidos no desenvolvimento e manutenção das tarefas do ciclo de vida. Exemplos de "Stakehoders" incluem gerentes de projeto, analistas de sistema, desenhista, etc.

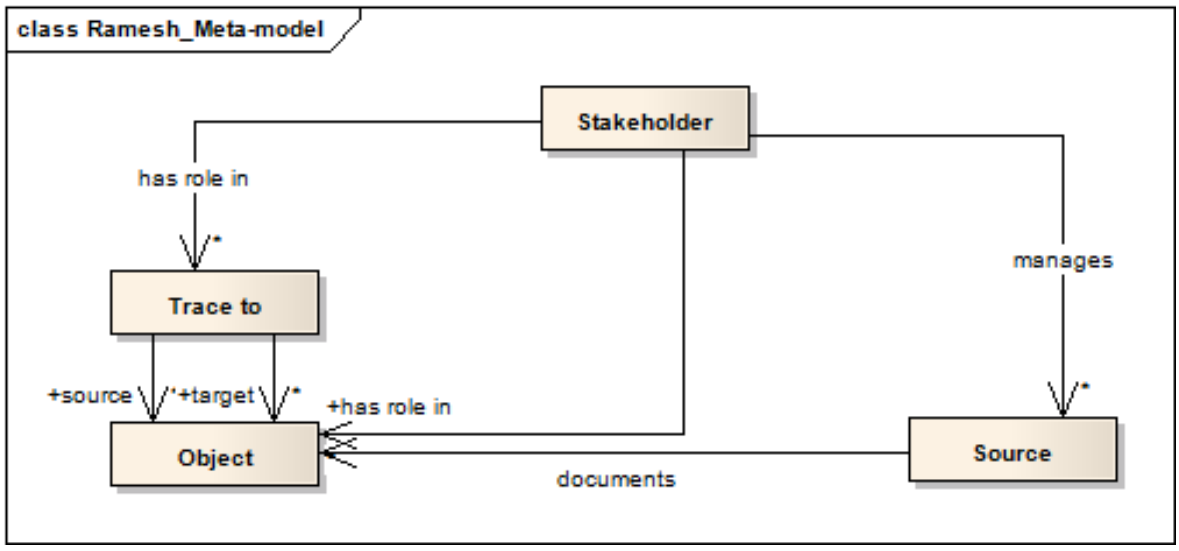


Figura 2.2. Metamodelo de Informação de Rastreabilidade Ramesh e Jarke (adaptado de Ramesh e Jarke) [Ramesh & Jarke, 2001]

A dimensão " Onde é representada?" refere-se ao aspecto da fonte de informação do objeto conceitual. Todos "Objects" são documentados em "Sources" que podem ser um meio físico, como documentos, ou coisas intangíveis, como referências às pessoas ou políticas não documentadas. Exemplos de "Sources" são: Documento de Especificação de Requisitos, Atas de Reunião, Memorandos, Telefonemas, etc. "Stakeholders" gerenciam as fontes (criam, mantêm e utilizam).

A dimensão "Como é representada?" é modelada como um atributo da fonte ("Source"). A informação pode ser documentada formal ou informalmente nas fontes, na forma de modelos ou textualmente.

A dimensão "Por que certo objeto conceitual é criado ou evoluído?" não tem uma representação explícita no metamodelo. Esta dimensão é obtida através da ligação "Trace to" entre "Objects" representando que um objeto é a razão da existência de outro objeto. Para esta representação ser possível deve-se criar uma ligação com esta semântica do tipo da meta-classe "Trace to".

"Quando a informação foi capturada, modificada ou evoluída?" também é representada como atributo de "Trace to" e "Object".

O metamodelo proposto por Ramesh e Jarke resolve as cinco dimensões de Rastreabilidade, mas apresenta um problema: não permite que as cinco dimensões sejam representadas em um único Rastro. Nesse metamodelo os Rastros não são tratados como elementos únicos que possuem as cinco dimensões, mas como instâncias de tipos que representam algumas das dimensões. Por exemplo, uma ligação que diz que um componente do sistema satisfaz um requisito é representada com os elementos "Com-

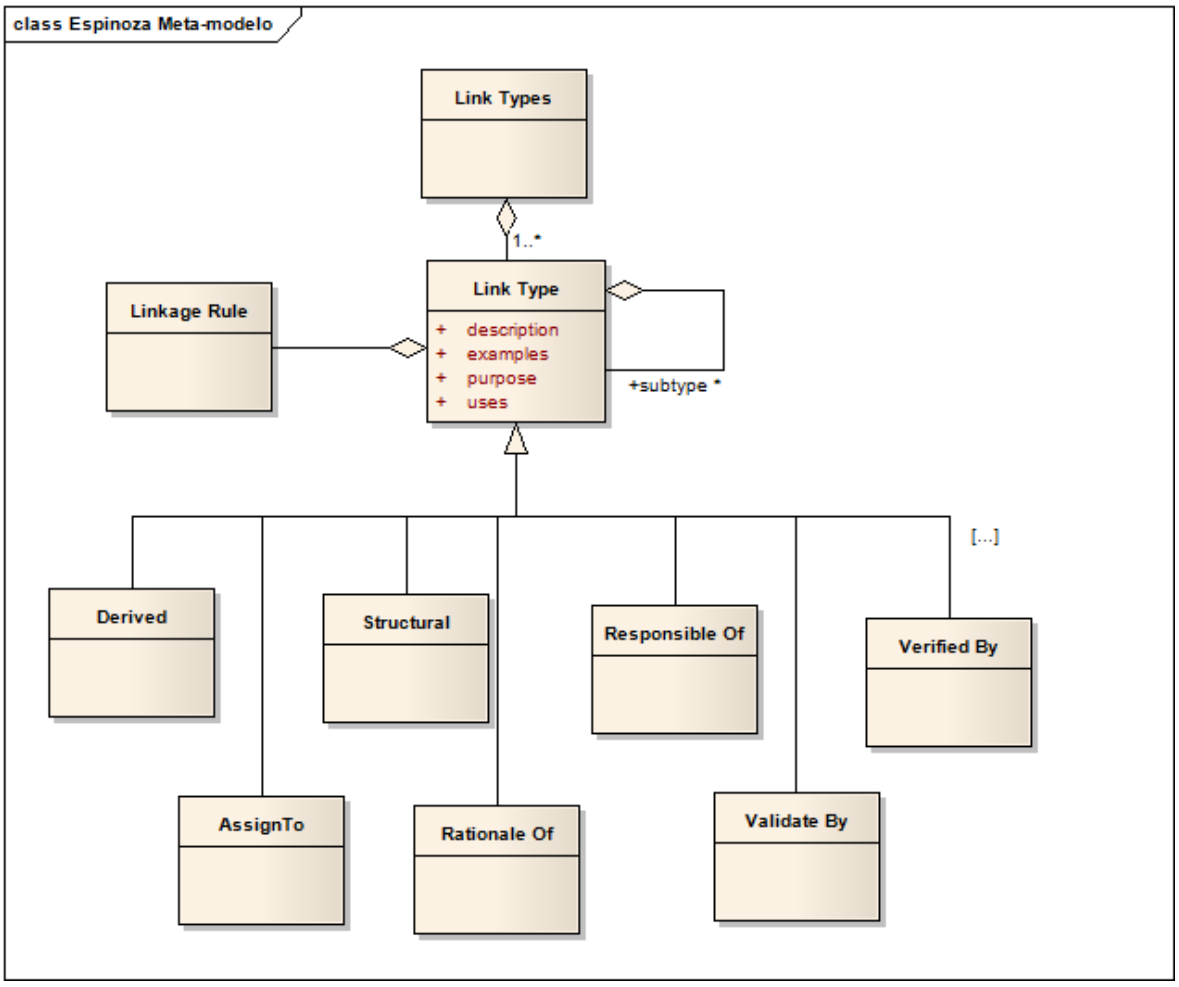


Figura 2.3. Metamodelo de Informação de Rastreabilidade Espinoza e outros (adaptado de Espinoza e outros [Espinoza et al., 2006])

ponente" e "Requisito" do tipo "Object" e a ligação "satisfaz" do tipo "Trace to". Este Rastro não possui as dimensões: "Quem são as Partes Interessadas que executam papeis na criação e manutenção dos objetos da Rastreabilidade?", "Onde é representada?", "Por que certo objeto conceitual é criado ou evoluído?". Outros Rastros ligados a estes são necessários para se ter todas as dimensões. Um metamodelo mais eficiente deveria permitir que cada Rastro possuísse as cinco dimensões.

Outro metamodelo a ser analisado foi especificado por Espinoza e outros [Espinoza et al., 2006] e definido no formato de esquemas (*"schema"*). Espinoza e outros utilizaram a linguagem Caseml (baseada em XML) para especificar a ligação de Rastreabilidade. Uma versão parcial em UML do metamodelo de Espinoza e outros é mostrada na Figura 2.3. Como são muitos os tipos apresentados por Espinoza e outros, não foram todos modelados na figura 2.3, mas todos são discutidos abaixo.

Espinoza e outros fizeram inicialmente uma compilação na literatura sobre Ras-

treabilidade de requisitos, onde identificaram os tipos de ligações de Rastreabilidade até então documentados, e os classificaram. Definiram um grupo de meta-classes denominadas "Traceability meta-type". É nesse grupo que se define todo tipo de Rastreabilidade e as regras de ligações válidas entre os artefatos do desenvolvimento de software. Foi criado um "Conjunto de tipos de Rastreabilidade" (TYS - "*Traceability Type Set*") que foram representadas na figura 2.3 como as heranças do "*Link Type*". A seguir, os tipos contidos no TYS são listados, e é mostrado de onde eles foram extraídos, e uma breve descrição de cada um. Não faz parte do escopo deste trabalho uma discussão aprofundada destas ligações.

***Derived*:** Também identificado por Ramesh e outros [Ramesh & Jarke, 2001]. Registra os requisitos que foram decompostos a partir de outros requisitos. A decomposição pode ser um processo recursivo e indica que o requisito derivado está em um nível de detalhamento maior do que o requisito original.

Apesar do artigo não indicar, podemos deduzir que este tipo de ligação possui a propriedade de derivação transitiva, que produz derivações indiretas. Se do requisito "A" derivou-se o requisito "B", e desse derivou-se o requisito "C", temos que o requisito "C" é também derivado do "A". Provavelmente não deve existir uma ligação de derivação diretamente entre eles. Essa ligação deve ser deduzida a partir das outras duas.

***Multiple Source of Suport*:** de Ramesh e Jarke [Ramesh & Jarke, 2001], é uma representação do fato que podem existir formas alternativas de satisfazer um requisito.

***Degrees of Satisfation*:** de Ramesh e Jarke [Ramesh & Jarke, 2001], especifica que um artefato é parcialmente realizado pelos artefatos com os quais está ligado, e que é mostrado pelo grau de satisfação indicado pelo tipo da ligação.

***Requires Execution Of*:** de Spanoudakis e outros [Spanoudakis et al., 2004], denota que a execução da operação ligada se faz necessária na sequência.

***Can Partially Realize*:** de Spanoudakis e outros [Spanoudakis et al., 2004], indica que a execução de um caso de uso pode realizar parte do requisito com o qual está relacionado.

***Traceability between PF and Product feature*:** de Lago e outros [Lago et al., 2004], mostra que as características de um produto particular suportam as características da família do produto.

***Composed Of*:** de Lago [Lago et al., 2004], expressa que as características da família do produto são decompostas em características do produto no nível do produto.

***Traceability between Product FM and Product CM*:** de Lago e outros [Lago et al., 2004], é uma ligação de Rastreabilidade definida entre cada característica do produto e decisão de desenho (componentes, classes ou interfaces).

***Traceability between Product CM and Implementation*:** de Lago e outros [Lago et al., 2004], é definido entre cada decisão de desenho e ativos de implementação (componentes, módulos, código fonte, etc.) que resolvem o desenho.

***Assign To*:** de Letelier [Letelier, 2002], a ligação de Rastreabilidade "alocado para" liga requisitos a componentes do sistema para registrar a realização de cada requisito nos componentes do sistema. É usada para identificar os componentes para os quais um determinado requisito foi alocado.

***Structural Dependencies*:** de Dahlsted e Persson [Dahlstedt & Persson, 2003], expressa o relacionamento entre requisitos que são organizados numa estrutura hierárquica. Requisitos de negócio de alto nível podem ser decompostos em requisitos de software mais detalhados, formando a hierarquia.

***Goal Dependencies*:** de Ramesh e Jarke [Ramesh & Jarke, 2001], representa a dependência entre artefatos que colaboram para o objetivo do sistema.

***Task Dependencies*:** de Ramesh e Jarke [Ramesh & Jarke, 2001], a ligação indica que a dependência entre os objetos criados por sua dependência comum em uma tarefa cria uma dependência de tarefa.

***Resource Dependencies*:** de Ramesh e Jarke [Ramesh & Jarke, 2001], mostra a dependência entre artefatos que compartilham recursos do sistema.

***Temporal Dependencies*:** de Ramesh e Jarke [Ramesh & Jarke, 2001], existe quando ações relacionadas com os objetos são feitas numa ordem temporal específica.

***Strength Dependencies*:** de Ramesh e Jarke [Ramesh & Jarke, 2001], é uma medida do quanto um objeto afeta outro. Pode ser medida qualitativamente (muito, médio, baixo, etc.) ou quantitativamente (numa escala de 1-10, por exemplo).

***Overlap*:** de Spanoudakis e outros [Spanoudakis et al., 2004], denota que os elementos conectados referem-se a uma característica comum do sistema ou ao seu domínio.

***Requires Feature In*:** de Spanoudakis e outros [Spanoudakis et al., 2004], denota que uma parte relevante do caso de uso não pode ser realizada sem a existência de características estruturais ou funcionais presentes no requisito relacionado.

***Requires and Excludes*:** de Lago e outros [Lago et al., 2004], modela dependências entre as características da família do produto.

***Constrain Dependencies*:** de Dahlstedt e Persson [Dahlstedt & Persson, 2003], são relacionamentos para descrever que os requisitos são dependências ou restrições de outros.

***Cost/Value dependencies*:** de Dahlstedt e Persson [Dahlstedt & Persson, 2003], são devido aos custos envolvidos na implementação dos requisitos em relação ao benefício que a implementação do requisito vai prover ao usuário.

***Rationale Of*:** de Letelier [Letelier, 2002], é a base ou são alternativas associadas com a especificação rastreável.

***Responsible Of*:** de Letelier [Letelier, 2002], indica qual stakeholder é responsável pela definição de um item rastreável.

***Modifies*:** de Letelier [Letelier, 2002], indica a relação entre stakeholders e os itens rastreáveis que são modificados por eles.

***Validated By*:** de Letelier [Letelier, 2002], relaciona as especificações de requisitos com suas respectivas especificações de teste que validam os requisitos.

***Verified By*:** de Letelier [Letelier, 2002], relaciona as especificações de teste que verificam especificações UML.

Além do conjunto de tipos de Rastreabilidade (TYS - "*Traceability Type Set*") descritos acima, a figura 2.3 mostra que o "*Link Type*" possui "*Linkage Rules*", que especifica os *stakehoders* e suas permissões para acessar os tipos de ligações.

Outros metamodelos de Rastreabilidade foram discutidos no âmbito das pesquisas em MDE. Por exemplo, a OMG publicou uma "proposta para modelo de fundação MDA (vide Seção 2.1)" onde a visão do MDE da OMG é modelada. Essa proposta inclui a definição e também o modelo de um registro de transformação ("*Transformation Record*"), que é o equivalente no MDE a uma coleção de Rastros, neste caso, produzido por uma transformação. O núcleo do metamodelo está reproduzido na figura 2.4.

Um Registro de Transformação ("*Transformation Record*") representa um contexto onde uma transformação ocorre e possui uma coleção de Rastros ("*Trace*"). Cada Rastro relaciona objetos ("*Object*")dos modelos. Observe que um Rastro pode não relacionar nenhum objeto, como também pode relacionar mais do que dois objetos. A forma como cada ligação deve ocorrer é definida no "*ModelTransformation Specification*".

Outros autores de MDE apresentam diversas variações de metamodelos de Rastreabilidade, mas o núcleo proposto possui pouca diferença do metamodelo da OMG.

Winkler e Pilgrim [Winkler & von Pilgrim, 2009] propuseram um metamodelo com características comuns aos principais metamodelos do MDE (Figura 2.5). Eles apresentaram o metamodelo através da linguagem TML ("*Traceability metamodeling language*"). Como já dito, Para facilitar a comparação com outros metamodelos, aqui esse metamodelo é representado em UML.

O elemento "*Context*" é usado para registrar certos metadados, como por exemplo, quais metamodelos são usados, ou os valores dos parâmetros de transformação. O "Context" é usado como contêiner de Rastros ("*Traces*").

O elemento Rastro ("*Trace*") também pode conter metadados (por exemplo, quais regras ou operações criaram o Rastro). Existem dois diferentes tipos de modelos: um

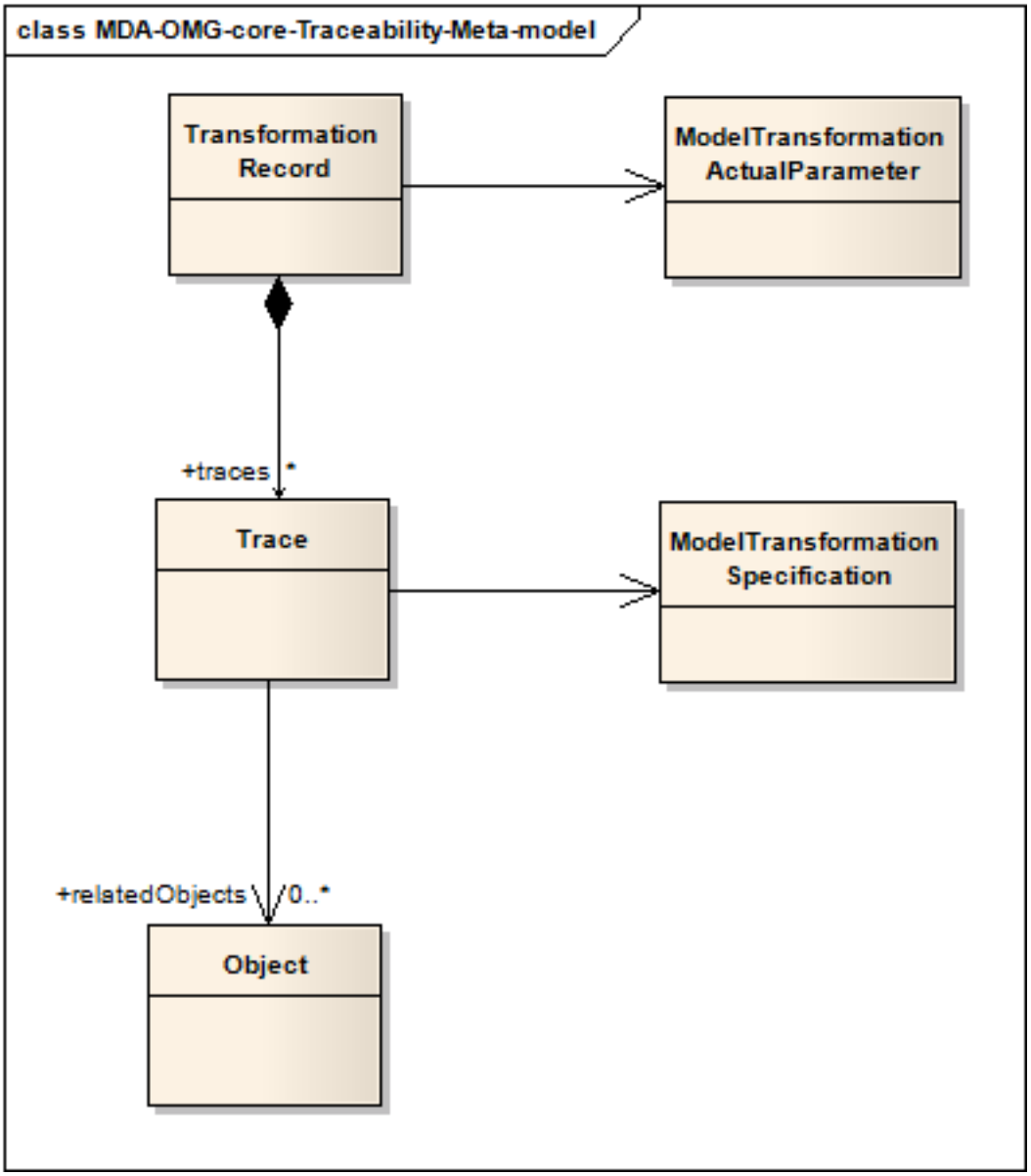


Figura 2.4. Metamodelo de Informação da Rastreabilidade MDA-OMG (adaptado da OMG)

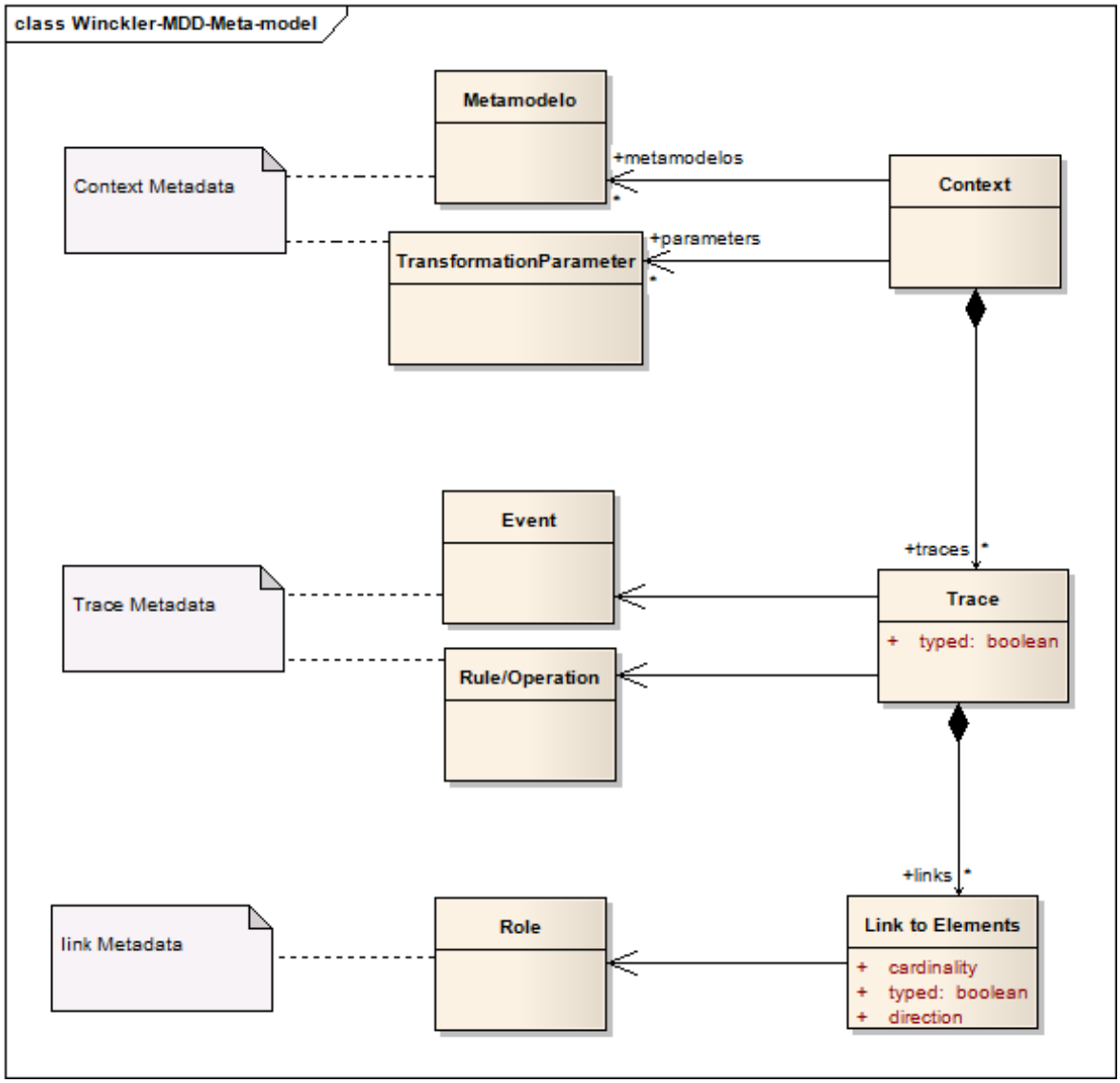


Figura 2.5. Metamodelo de Informação da Rastreabilidade Winkler baseado em modelos MDE (adaptado de Winkler e Pilgrim [Winkler & von Pilgrim, 2009])

usando Rastros não "tipados", outro, criando sofisticados tipos hierárquicos de Rastros. Os tipos de Rastros podem ser vistos como metadados.

As características mais importantes de um "*Trace*" são os "*links*" para elementos do modelo ou artefatos. Cada ligação pode registrar metadados. O metadado "*Role*" é normalmente definido como "*source*" ou "*target*".

A cardinalidade de uma ligação também varia em diferentes modelos: O mais restritivo é quando um Rastro simples conecta exatamente uma fonte a um destino. Mas, há casos onde mais de dois elementos podem ser interconectados (n:m). Alguns modelos esperam no mínimo uma fonte e um destino (2..*), já outros aceitam que os Rastros possam não ter nenhuma ligação.

Os modelos de Rastros são, segundo Winckler e Pilgrim, normalmente registrados em um modelo separado, e as ligações são unidirecionais a fim de manter os modelos independentes entre si. Alternativamente, porém, modelos podem conter as ligações por si mesmo ou definir as ligações como bidirecionais.

2.6 Discussão Sobre os Métodos e os Metamodelos

A Rastreabilidade é, ainda hoje, pouco e mal usada [Mader et al., 2009] apesar de toda a mobilização mostrada neste capítulo para resolver a questão. Um grande esforço para manter os registros dos Rastros continua sendo necessário. Desta forma, fica evidente que nenhum dos métodos propostos para manter a Rastreabilidade resolve por completo o problema. Isso se deve ao fato de que os métodos estão direcionados a um contexto de uso limitado. É difícil que seja de outra maneira: uma das considerações de Pohl [Pohl, 1996] é que devido ao grande número de informação produzida durante o processo de desenvolvimento de software, apenas ligações de Rastreabilidade orientadas ao conteúdo são base para o uso apropriado, isto é, o uso da informação gravada é quase impossível se a informação de Rastreabilidade não está encapsulada em seu contexto.

O vínculo entre o contexto e a Rastreabilidade é essencial para garantir sua utilização. Os métodos usados para capturar, manter e recuperar ligações de Rastreabilidade devem ser encapsulados em seus contextos, o que provavelmente irá conduzir a solução da questão da Rastreabilidade para a utilização de mais de um método, dependendo dos contextos de uso. A integração dos diferentes métodos deve ocorrer através de uma representação da informação da Rastreabilidade única e compartilhada. É o modelo de informação de Rastreabilidade. Em uma organização, um modelo de Rastreabilidade deve ter um núcleo único e capaz de registrar toda a informação necessária para a manutenção da Rastreabilidade. E esse modelo deve ser representado a partir de

um metamodelo geral que especifique a sintaxe e a semântica necessária e suficiente para se modelar a informação da Rastreabilidade em diferentes contextos. Mas, assim como ocorre com os métodos, também há grandes dificuldades em se estabelecer um metamodelo para a Rastreabilidade.

Pohl [Pohl, 1996] também vislumbrou a razão dessa dificuldade: o uso de informação de Rastreabilidade depende das Partes Interessadas e das tarefas do processo de desenvolvimento de software utilizado, que é diferente em cada organização. Desta forma, a Rastreabilidade está intimamente associada à cultura organizacional. O processo personalizado e as ferramentas utilizadas para automatizar parte desse processo são particulares de cada organização e interferem de forma significativa no metamodelo utilizado para estabelecer modelos de informação de Rastreabilidade.

Deve-se considerar as particularidades de uma organização e projetos envolvidos para se estabelecer métodos de Rastreabilidade eficazes. Particularidades como o nível de maturidade do processo de desenvolvimento de software utilizado pela organização e seu grau de automação. Um metamodelo de Rastreabilidade deve ser coerente com os elementos do processo de desenvolvimento de software que a organização utiliza e se integrar com as ferramentas usadas na automação desse processo para realmente ser útil a uma organização.

A relação entre a automação do desenvolvimento de software e o processo utilizado na organização é discutida por diversos autores. O'Neill [O'Neill, 1982] relata quatro anos de observações sobre o relacionamento da automação do desenvolvimento de software com a maturidade dos processos de Engenharia de Software. Já O'Halloran [O'Halloran, 2000] questiona as interferências das particularidades dos processos de desenvolvimento de software na automação de *safety critical software*. Cordy [Cordy, 2003] fez uma pesquisa para comprovar que o motivo da baixa utilização da automação na manutenção de software está nas particularidades de cada projeto. Por fim, Liu [Liu, 2009] propõe que a automação deve ser orientada a processo para que possa ser utilizada ao longo do ciclo de vida do projeto.

Existem muitas ferramentas disponíveis no mercado que propõem diversas formas de automatização do desenvolvimento (Aksyonov e outros [Aksyonov et al., 2009] analisam algumas delas), que apresentam diferentes formas de personalizações para se adaptarem a realidade da organização e dos projetos. Mas a complexidade, o esforço requerido, e o tempo necessário que envolvem todo o ambiente de desenvolvimento de software, não permitem, ou, pelo menos dificultam muito, que as personalizações ocorram da melhor forma possível. Além disso, há muitos problemas na integração entre ferramentas de diferentes fabricantes, o que impede que a automação possa ser usada contemplando amplos aspectos do desenvolvimento. Algumas ten-

tativas de criação de padrões para integração dessas ferramentas já ocorrem desde a década de 1980: ATIS/CATIS/CIS - A Tool Integration Standard, CAIS/CAIS-A (MIL-STD-1838/MIL-STD-1838A)- The Common Ada Programming Support Environment (APSE) Interface Set (CAIS), CDIF - The CASE Data Interchange Format, IEEE (P1175), IRDS (ANSI/ISO) - The Information Resource Dictionary Standard, PCTE/PCTE+ - The Portable Common Tools Environment, SIGMA - The Software Industrialized Generator and Maintenance Aids. Até o momento, nenhum desses padrões está sendo usado em larga escala.

É nesse cenário que se deve conceber um metamodelo de Rastreabilidade coerente com os elementos das ferramentas de automação e com os processos de desenvolvimento de software usados nas diferentes organizações. Amelunxen e outros [Amelunxen et al., 2008] estudaram alguns metamodelos que relacionam elementos da Engenharia de Software mantidos por cada uma das ferramentas envolvidas, mas não aprofundaram na complexidade da Rastreabilidade. Em nenhum caso observado nas pesquisas deste trabalho se encontrou uma preocupação nos estudos da Rastreabilidade com as iniciativas de automação do processo de desenvolvimento de software como um todo. Nos metamodelos vistos, a complexidade do ambiente onde a Rastreabilidade está envolvida é registrada de forma não-estruturada em meta-classes de contexto, com pouco aprofundamento nas descrições destes contextos.

No nosso ponto de vista, a Rastreabilidade só irá se tornar uma realidade quando estiver vinculada aos diferentes aspectos dos processos de desenvolvimentos de software. Um modelo de Rastreabilidade deve estar agregado com o modelo do processo de desenvolvimento de software utilizado na organização. Dessa forma, um metamodelo de Rastreabilidade deve estar integrado com o metamodelo de processo de desenvolvimento de software.

2.7 Conclusão do Capítulo

Neste capítulo foi apresentada a Questão da Rastreabilidade, sua importância e o fato de que até o momento não haver uma solução satisfatória. Mostrou-se também os conceitos básicos utilizados nas pesquisas e suas variações dependendo do contexto onde estão inseridas, focando principalmente no contexto da Engenharia de Requisitos e nos estudos de MDE.

Fez-se a análise dos diferentes métodos para se manter a Rastreabilidade até então concebidos pela comunidade científica. Ponderou-se que a solução para a Questão da Rastreabilidade deve estar numa conjunção de diferentes métodos e esta conjunção

deve ter como base algum modelo de informação de Rastreabilidade. Deve haver um metamodelo único capaz de produzir modelos de Rastreabilidade com todas as variações necessárias, devido à diversidade de contextos onde os modelos de informação da Rastreabilidade devem ser aplicados. O capítulo apresentou alguns metamodelos propostos e discutiu o fato de não haver consenso sobre qual é o melhor metamodelo.

Por fim, as causas da Questão da Rastreabilidade ainda não ter sido resolvida e a necessidade de haver um metamodelo vinculado a metamodelos de processo de desenvolvimento de software foram discutidas nos seus diferentes aspectos.

No capítulo 3 é apresentada uma proposta de metamodelo de Rastreabilidade vinculada à meta-elementos de processo de desenvolvimento de software, onde se registra todo o contexto onde um Rastro é gerado.

Capítulo 3

Rastreabilidade no Metamodelo RAISE

Neste capítulo, o Metamodelo RAISE é apresentado como uma proposta de gramática da representação da informação da Rastreabilidade. O RAISE foi concebido considerando a informação de Rastreabilidade integrada com elementos do processo de desenvolvimento de software utilizado, e a Seção 3.1 justifica essa abordagem. Os elementos dos metamodelos de processos de desenvolvimento de software que foram usados na criação do RAISE são descritos e detalhados na Seção 3.2 antes da apresentação do RAISE propriamente dito na Seção 3.3.

Este capítulo utiliza exemplos baseados no processo Praxis descritos, não somente nos livros, como também em artigos [de Pádua Paula Filho, 2001]. O processo Praxis [de Pádua Paula Filho, 2003] é baseado no Processo Unificado (*"Unified Process"* - UP) [Jacobson et al., 1999] e normas do IEEE [IEEE, 1994]. O Praxis foi escolhido por ser mais simples que o RUP ("*Rational Unified Process*")[Kruchten, 2003], mas mesmo assim adequado para exemplificação do Metamodelo RAISE. Exemplos da aplicação do Praxis presentes no sítio do autor [de Pádua Paula Filho, 2010] também foram utilizados.

3.1 Objetivo do Metamodelo RAISE

O Metamodelo de Informação de Rastreabilidade RAISE objetiva prover uma gramática para a criação de modelos de informação de Rastreabilidade integrado ao processo de desenvolvimento de software utilizado na organização.

Na Seção 2.6 foi discutida a vantagem do metamodelo de Rastreabilidade estar integrado ao metamodelo de processo de desenvolvimento de software. Dele deve ser possível criar modelos de informação de Rastreabilidade onde os Rastros estejam contextualizados nos aspectos do processo de desenvolvimento utilizado na organização de forma estruturada.

Para adotar essa abordagem, um metamodelo de Rastreabilidade precisa permitir, ou até mesmo exigir, que cada Rastro esteja vinculado ao seu contexto. É o caso do metamodelo proposto por Winckler e Pilgrim [Winkler & von Pilgrim, 2009] apresentado na Seção 2.5. Winckler e Pilgrim definiram a meta-classe de Contexto dos Rastros (vide Figura 2.5) que deve sempre existir para cada Rastro. Os atributos dessa meta-classe são definidos através de um metamodelo diferente para cada situação. Winckler e Pilgrim não estruturaram a forma de especificação desse contexto, deixando que a contextualização seja modelada sem nenhuma gramática previamente estabelecida.

A abordagem utilizada no RAISE se diferencia dos demais metamodelos ao estruturar o contexto dos Rastros tendo os elementos do processo de desenvolvimento

de software como base. Assim, a forma como uma organização desenvolve software pode ser descrita no seu modelo de informação de Rastreabilidade, permitindo que a informação dos Rastros seja capturada encapsulada em seu contexto, como vislumbrou Pohl [Pohl, 1996] (vide Seção 2.1).

A próxima seção, 3.2, descreve e detalha os elementos dos metamodelos de processos utilizados na estrutura do RAISE.

3.2 Metamodelos de Processo de Desenvolvimento de Software

O Metamodelo SPEM 2.0 (Software Process Engineering Metamodel) especificado pela OMG [OMG, 2008b] foi utilizado como base para a criação do Metamodelo RAISE. Isso porque o SPEM é um dos metamodelos para o processo de desenvolvimento de software mais referenciados na literatura acadêmica e possui muito dos elementos necessários para a definição do RAISE. Entretanto, a capacidade de expressão do SPEM é insuficiente para descrever o RAISE. Por exemplo, a semântica de regras de negócio de uma tarefa não está estruturada no SPEM, mas é necessária para o RAISE. Elementos necessários para a descrição do Metamodelo RAISE não encontrados no SPEM foram cobertos pelo metamodelo especificado por Eriksson e Penker para processos de negócio [Eriksson & Penker, 2000]. Apesar de esse metamodelo ser mais genérico que o SPEM 2.0 (lembrando que o "processo de desenvolvimento de software" é um "processo de negócio", quando o negócio é desenvolver software), há nele definições úteis para a Rastreabilidade e que são aplicáveis ao processo de desenvolvimento de software. Esta seção descreve alguns conceitos do metamodelo de Eriksson e Penker e do metamodelo SPEM 2.0 com o detalhamento necessário para o entendimento do Metamodelo RAISE.

Para se entender o metamodelo de Eriksson e Penker, deve-se inicialmente entender o seu conceito de processo de negócio: "tarefas executadas para o negócio que fazem o estado dos recursos de negócio mudar". Sendo que Recursos são "objetos do negócio, como pessoas, material, informação, e produtos, que são usados ou produzidos no negócio". A Figura 3.1 mostra esses elementos no metamodelo.

Nesse contexto, toda tarefa executada que muda o estado do negócio é um Processo ("*Process*"). Os Processos descrevem como o trabalho é executado. Processo é composto de Processos ("*part of*"), e, principalmente, processos são governados por regras de negócio ("*rule*").

Uma regra de negócio é uma sentença que pode controlar ou afetar a execução de um processo de negócio tanto quanto as estruturas dos recursos do negócio. As

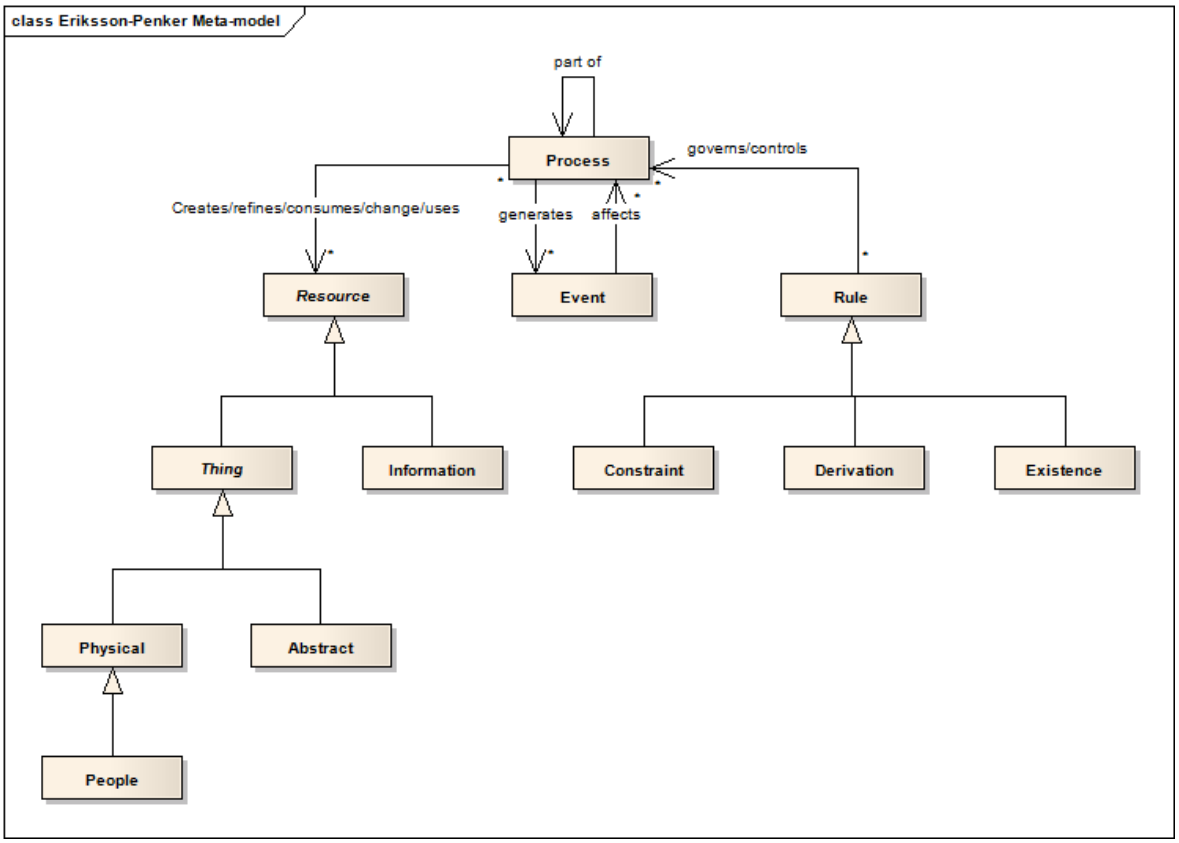


Figura 3.1. Metamodelo dos conceitos de modelagem de negócio proposto por Eriksson/Penker (extraído do livro "*Business Modeling with UML: Business Patterns at Work*" [Eriksson & Penker, 2000])

regras de negócio são conceitos importantes para o entendimento do Metamodelo de Informação de Rastreabilidade RAISE e por isso são detalhadas a seguir.

James Odell [Odell, 1998] divide as regras de negócio em regras de derivação ("*derivation*") e regras de restrição ("*constraint*"). Regras de derivação definem como o conhecimento ou a informação pode ser transformada em outro conhecimento ou informação. Uma derivação pode ser uma regra computacional (ex. uma fórmula para calcular um valor) ou uma regra de inferência (ex. se algum fato é verdade, então outro fato inferido precisa ser verdade)[Eriksson & Penker, 2000, pág. 153].

Regras de restrições restringem tanto a estrutura quanto o comportamento de objetos ou processos; restringem a forma como objetos são relacionados uns com outros ou a forma como mudanças de estados de objetos ou processos podem ocorrem.

Uma categoria adicional de regras de negócio definida por Eriksson e Penker e não mencionada por Odell é a categoria que engloba as regras de existência ("*existence*"). Regras de existência definem sobre que circunstâncias alguma coisa pode existir (o ciclo de vida de um objeto) e quando pode vir a existir (isto é, quando um objeto é criado

ou destruído).

As regras de negócio operam sobre os recursos ("*resource*"). Recursos são objetos que agem ou são usados nos negócios. São consumidos, produzidos, transformados ou usados pelo processo de negócio. Exemplos incluem material, energia, produtos, pessoas, informação e serviços. Eriksson e Penker definem diferentes tipos de recursos. Um metamodelo dos tipos de recursos está indicado na Figura 3.1, e é inspirado nas definições de Gale e Eldred de quatro tipos recursos concretos [Gale & Eldred, 1997].

- Físico ("*Physical*"): Uma entidade com realidade material que ocupa um volume no espaço. É alguma coisa que pode ser tocado.
- Abstrato ("*Abstract*"): Uma ideia ou conceito, normalmente uma composição de outros objetos. Envolve coisas ou conceitos que não são físicos e não podem ser tocados, mas são de importância para o negócio.
- Objeto de informação ("*Information*"): Uma representação de um conceito, coisa, ou outro objeto de informação. É importante separar o objeto de informação do conceito ou coisa que ele representa.
- Pessoa ("*People*"): Um ser humano que age no negócio.

São esses os conceitos do metamodelo de Eriksson e Penker que são explorados neste trabalho. Quando necessário servirão como complemento às definições do SPEM 2.0 (*Software Process Engineering Metamodel*). A Figura 3.2 descreve parte do Metamodelo do SPEM. O SPEM, especificado e mantido pela OMG [OMG, 2008b], foi propositalmente concebido com o mínimo de elementos necessários para definir qualquer processo de desenvolvimento de software. O seu principal objetivo é acomodar uma grande faixa de métodos e processos de desenvolvimento de diferentes estilos, contextos sociais, níveis de formalismos, modelos de ciclo de vida e comunidades.

No SPEM, o trabalho executado no processo de desenvolvimento de software é especificado pelo elemento *"Work Definition"*. Já o *"Task Definition" e "Step"* descrevem o desenvolvimento do trabalho. Essa descrição mostra o comportamento de um papel definido que executa o trabalho ou de muitos papéis definidos participantes e colaboradores durante a execução do *"Work Definition"*.

O "*Work Definition*" é um "*Classifier*" que generaliza todas as definições de trabalho no SPEM 2.0. Pode conter um conjunto de pré e pós-condições com restrições que necessitam ser validadas antes da execução dos trabalhos e após eles terem terminados. Além dessas restrições há ainda restrições que são atribuídas ao "*Work Definition*" por ser um "*Classifier*".

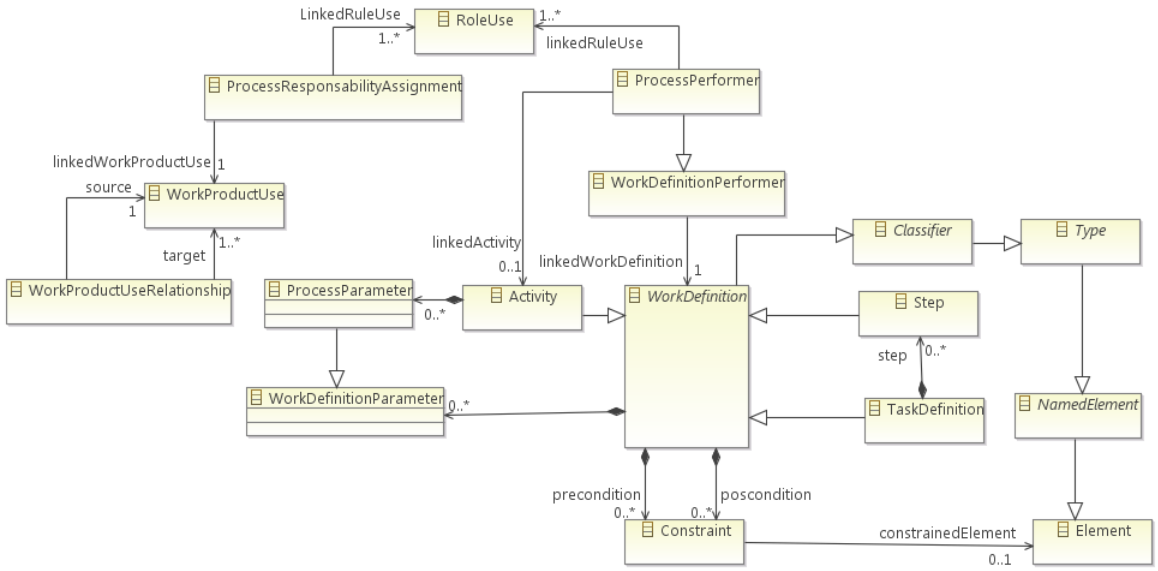


Figura 3.2. SPEM 2.0 (Extraído do documento "*Software & Systems Process Engineering Meta-Model Specification*" [OMG, 2008b])

O "*Work Definition Parameter*" é uma generalização abstrata para elementos que representam parâmetros das definições do trabalho. É usado para declarações de entradas e saídas. As meta-classes dos tipos concretos de entrada ou saída devem ser definidas como subtipos de "*Work Definition Parameter*".

O "*Work Definition Performer*" representa o relacionamento do executor do trabalho com a definição do trabalho. Diferentes especializações do trabalho irão introduzir diferentes tipos de executores.

"*Role Use*" representa tanto o executor do trabalho (definido no "*Work Definition*"), como algum participante do trabalho. Se for o executor, o "*Role Use*" se associa ao "*Work Definition Performer*" através da especialização "*Process Performer*". Sendo um participante, o "*Role Use*" é registrado na composição "*nested Breakdown Element*" (vide Figura 3.3) da tarefa e pode ser usada por uma das sub-tarefas como o executor ou "*Process Responsibility Assignment*". "*Role Use*" é valido apenas no contexto de uma tarefa. Não é reusado entre tarefas.

O "*Work Product Use*" é um elemento que representa um tipo de entrada ou saída para uma tarefa ou representa um participante da tarefa. Se for uma entrada ou saída, então o "*Work Product Use*" precisa ser relacionado com a tarefa através do "*Process Parameter*". Sendo um participante, então o "*Work Product Use*" é gravado na composição "*nested Breakdown Element*" (vide Figura 3.3) da tarefa e pode ser usado por uma das sub-tarefas como uma entrada ou saída ou ser relacionado a um

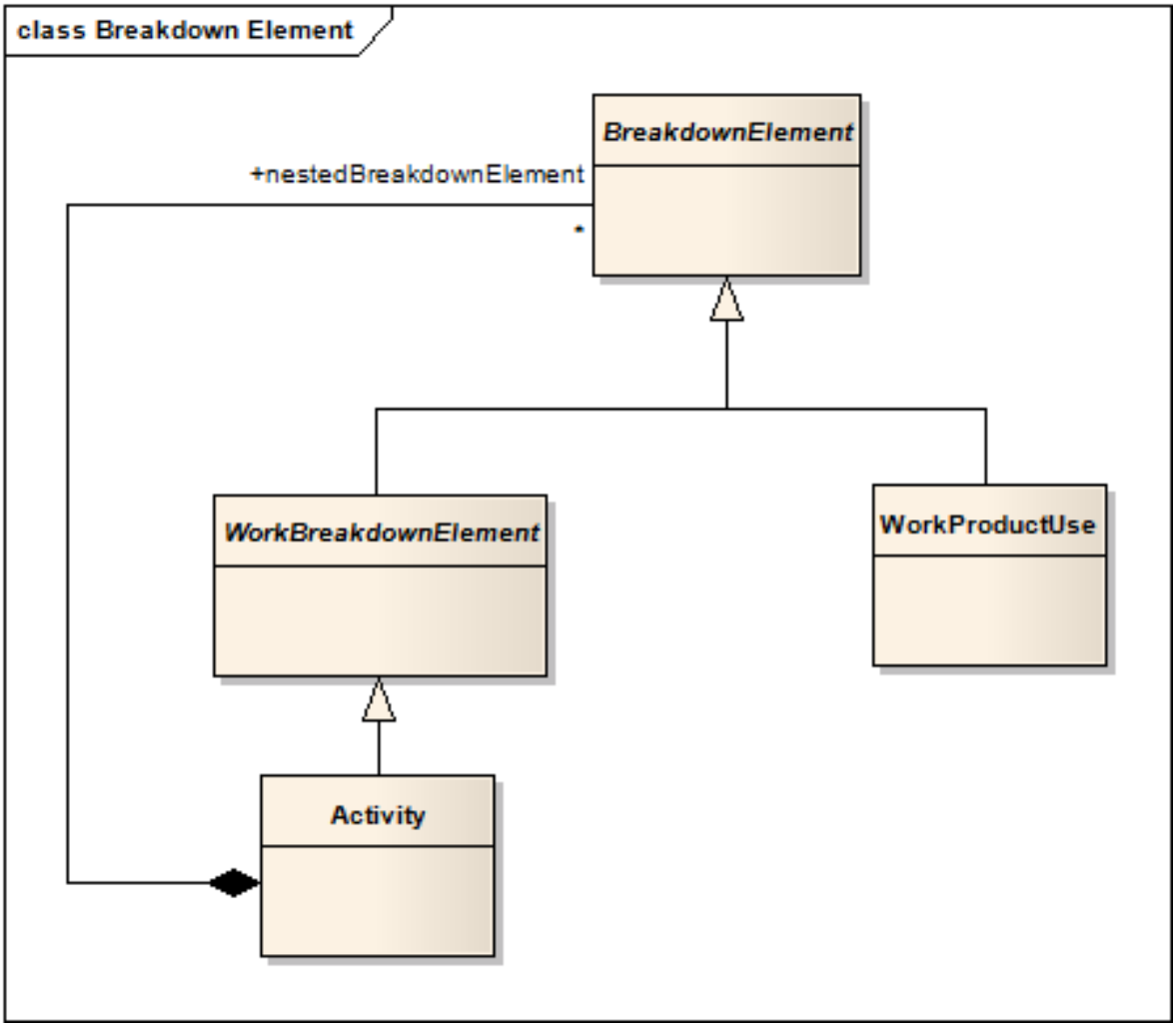


Figura 3.3. A composição Nested Breakdown Element do modelo SPEM (Adaptado do documento "Software & Systems Process Engineering Meta-Model Specification" [OMG, 2008b])

"*Role Use*" via "*Process Responsibility Assignment*".

"*Work Product Use Relationship*" expressa uma relação entre produtos de trabalho. Pode ser usado para expressar diferentes tipos de relacionamentos entre "*Work product Uses*". Os tipos de relacionamentos típicos descritos na especificação do SPEM 2.0 são:

- "composição": expressa que uma instância de produto de trabalho é parte de outra instância de outro produto de trabalho.

- "agregação": indica que o um produto de trabalho é usado com outro produto de trabalho.

- "dependência" ou "impactado por": indica que um produto de trabalho impacta outro produto. Esse tipo de relacionamento indica dependência de Rastreabilidade entre as instâncias dos produtos de trabalho que devem ser consideradas quando se cria ou altera esses produtos de trabalho.

Os tipos de relacionamentos entre os "*Work Product Use*", principalmente o de "dependência", devem ser usados para registrar Ligações de Rastreabilidade. Apesar disso, esses relacionamentos não são usados no Metamodelo RAISE. Outros mecanismos com semântica mais rica do ponto de vista da Rastreabilidade são utilizados para representar essas ligações.

São esses os elementos do SPEM 2.0 e do metamodelo Eriksson e Penker relevantes para este trabalho. O Metamodelo de Informação de Rastreabilidade RAISE, que utiliza os elementos aqui apresentados, é mostrado na seção seguinte, 3.3.

3.3 Metamodelo de Informação de Rastreabilidade RAISE

3.3.1 Descrição do Metamodelo RAISE

O Metamodelo RAISE foi descrito neste trabalho de forma incremental. A descrição explorou a relação do metamodelo com os conceitos de Rastreabilidade mostrados na Seção 2.2 e com os metamodelos de processos mostrados na Seção 3.2. A partir de cada conceito de Rastreabilidade se justificou a necessidade de elementos do Metamodelo RAISE. Alguns conceitos já mostrados foram reproduzidos novamente aqui. Suas definições foram exploradas na explicação do metamodelo que facilita o entendimento da evolução da concepção do RAISE. Os elementos RAISE foram então descritos a partir dos elementos dos processos SPEM e Eriksson e Penker que cobriram o contexto onde o conceito se insere. Dessa forma se assegurou que o RAISE cobriu todos os aspectos da Rastreabilidade que conceitualmente deveriam ser cobertos a partir de elementos de processos já definidos.

Outro aspecto da descrição que deve ser ressaltado é que a representação do Metamodelo RAISE nesta dissertação não utilizou nenhum perfil UML. Apesar da linguagem UML permitir a especificação do perfil na definição de um metamodelo, o uso deste formalismo na descrição do Metamodelo RAISE não foi necessário pelo fato de cada elemento possuir sua descrição detalhada na forma textual.

A Figura 3.4 mostra uma visão parcial do Metamodelo RAISE que satisfaz a definição de Gotel e Finkelstein (descrita na Seção 2.2 e revista abaixo), utilizando

elementos do SPEM e Eriksson e Penker. O Metamodelo considera os Rastros ("*Trace*") como eventos ("*Event*") gerados pelas instâncias dos "*Work Definition*".

A definição de Rastreabilidade mais comumente referenciada [Pinheiro, 2003] é de Gotel e Finkelstein sobre Rastreabilidade de requisitos [Gotel & Finkelstein, 1994], e é naturalmente a base para a montagem do Metamodelo RAISE. A definição diz:

> "Rastreabilidade de requisitos se refere à habilidade de descrever e seguir a vida do requisito nas direções de avanço e retorno (isto é, da sua origem, através de seu desenvolvimento e especificação, para o seu uso subsequente, através de refinamentos e iterações continuadas)"[1].

Pinheiro [Pinheiro, 2003] fez uma análise desta definição, e, segundo sua interpretação, a necessidade de descrever a vida do requisito através de Rastros torna-se explicita. Pois, rastreá-lo, é a única forma de se seguir seu ciclo vida. Tanto Pinheiro quanto Winkler e Pilgrim [Winkler & von Pilgrim, 2009] concordam que para melhor entender esta definição deve-se entender o que é um Rastro. O Oxford English Dictionary, citado nos dois artigos, define Rastro como:

> "(possivelmente) uma indicação ou evidência não-material mostrando o que existiu ou aconteceu" [2].

Rastro pode ser:

- Uma indicação ou evidência do que aconteceu.
- Uma indicação ou evidência do que existiu.

Trazendo esta definição para o contexto da Engenharia de Software: "O que aconteceu" é um evento ocorrido no ciclo do desenvolvimento de software. Esse evento pode ter sido provocado por uma tarefa pertencente ao próprio processo ou um evento externo que interfere no processo (Mudança em alguma legislação que afeta o sistema em desenvolvimento). Por exemplo, o Analista de Requisitos João identificou as necessidades dos usuários Márcio e Carlos e as registrou no documento ERSw, Seção 2 [de Pádua Paula Filho, 2003], em reunião ocorrida no dia 05/06/2010 e também registrou as razões na ata ATA-20100605. O modelo SPEM 2.0 não representa os eventos gerados pela tarefa no processo. Apenas os descrevem em cada "*Work Definition*" de

[1] Do inglês: "Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)".

[2] Do inglês: "(possibly) non-material indication or evidence showing what has existed or happened"

forma textual e não estruturada. Como já dito anteriormente, um Rastro é um evento gerado por algum "*Work Definition*", e, para se registrar o Rastro, a sua representação se faz necessária. No Metamodelo RAISE foi utilizada a meta-classe "*Event*" (do metamodelo Eriksson e Penker), já que representa um evento de negócio gerado por um processo. Como no Metamodelo RAISE o equivalente ao processo de Eriksson e Penker é o "*Work Definition*", um "*Event*" foi concebido sendo gerado pelo "*Work Definition*".

Eriksson e Penker observaram que um processo é afetado por eventos. Eventos ocorrem no ambiente onde o processo está inserido ou são gerados por outros processos (ou sub-processos) e podem ativar um determinado processo. Um evento é identificado sempre que ocorrer uma mudança de estado nos elementos do processo de desenvolvimento de software. Um trabalho qualquer no processo de desenvolvimento de software (tarefa, um passo dentro de uma tarefa) provoca um ou mais eventos, já que esta tarefa deve alterar algum estado de algum elemento. O elemento no Metamodelo SPEM que representa o trabalho no desenvolvimento de software que foi utilizada no RAISE é a meta-classe genérica "*Work Definition*" (Figura 3.2 e Figura 3.4). O "*Work Definition*" pode ser especializado em "*Activity*", "*Step*" ou "*Task Definition*", que provocam eventos que modificam o estado do processo de desenvolvimento de software.

Retomando a definição de Rastro do Oxford English Dictionary sobre o ponto de vista do RAISE: "A indicação ou evidência do que aconteceu" está vinculada a algum registro que evidencie direta ou indiretamente uma ocorrência de um determinado "*Work Definition*". E essa evidência ou indicação pode ser um registro de um evento ("*Event*") ocorrido.

Já "a evidência ou indicação do que existiu" está vinculada a algum registro que evidencie direta ou indiretamente um estado de um ou vários produtos de trabalho intermediários ou finais. O SPEM, assim como o RAISE, representa os produtos de trabalho através da classe genérica "*Work Product Use*", que permite a especificação de diferentes estados. O registro destes elementos e seus estados caracterizam Rastros pela definição do Oxford English Dictionary.

A definição de Gotel e Finkelstein, apesar de ser a mais citada, não é a única. Ramesh e outros [Ramesh & Jarke, 2001] evoluíram as características necessárias para o metamodelo de Rastreabilidade baseando-se em estudos empíricos. Ampliaram a definição de Rastro ao concluírem a existência de cinco dimensões da Rastreabilidade de requisitos. Na sua visão, cada Rastro deve responder as seguintes perguntas:

- Que informação é representada?
- Quem são as partes interessadas que executam papéis na criação e manutenção dos Rastros?

- Onde é representada?
- Como é representada?
- Por que certo elemento é criado ou evoluído?
- Quando a informação foi capturada, modificada ou evoluída?

Ramesh e Jarke [Ramesh & Jarke, 2001] montaram um metamodelo baseado nessas definições que foi analisado na Seção 2.5. Nessa seção se observou que o Metamodelo de Ramesh e Jarke não permite que as cinco dimensões sejam representadas em um único Rastro. Os Rastros não são tratados como elementos únicos que possuem estas dimensões, mas como instâncias de tipos que representam algumas das dimensões. Um metamodelo mais eficiente deveria permitir que cada Rastro possuísse estas cinco dimensões. A Seção 2.2 apresenta uma discussão detalhada sobre cada dimensão. Do ponto de vista do Metamodelo RAISE, as dimensões foram observadas da seguinte forma:

A definição de Gotel e Finkelstein, que foi discutida nos parágrafos anteriores, responde a primeira dimensão, "Que informação é representada?". Dessa forma, os elementos do Metamodelo RAISE já discutidos são capazes de representar essa dimensão. As novas quatro dimensões podem também ser modeladas naturalmente através dos elementos dos metamodelos de processo.

A segunda dimensão, "Quem são as partes interessadas que executam papéis na criação e manutenção dos Rastros?", é mapeada para quem executa a tarefa no processo. Nos metamodelos SPEM 2.0 e RAISE(Figura 3.2, Figura 3.4) essa atribuição é representada por alguma instância do "*Work Definiton Performer*", que, caso o "*Work Definition*" seja uma "*Activity*", o "*Work Definition Performer*" deve ser especializado em "*Process Performer*".

As dimensões "Onde é representada?" e "Como é representada?" não possuem representantes especializados no modelo SPEM 2.0, mas aparecem no modelo de Eriksson e Penker como recursos do tipo "*Thing*". O elemento "*Thing*" é onde se representa a informação (elemento "*Information*"), que no Metamodelo RAISE é uma especialização de "*Work Product Use*". O "*Work Product Use*" é o equivalente ao "*Resource*" do metamodelo de Eriksson e Penker e por isso se tornou a generalização do elemento "*Thing*". Assim, na Figura 3.4 se vê os elementos "*Thing*" e suas especializações "*Physical*" e "*Abstract*" e ainda o elemento "*Information*". Todos como especialização de "*Work Product Use*". Desta forma, o ''*Work product Use*" é abstrato neste metamodelo. Sendo que, quando ele for a informação utilizada pelo "*Work Definition*", deve

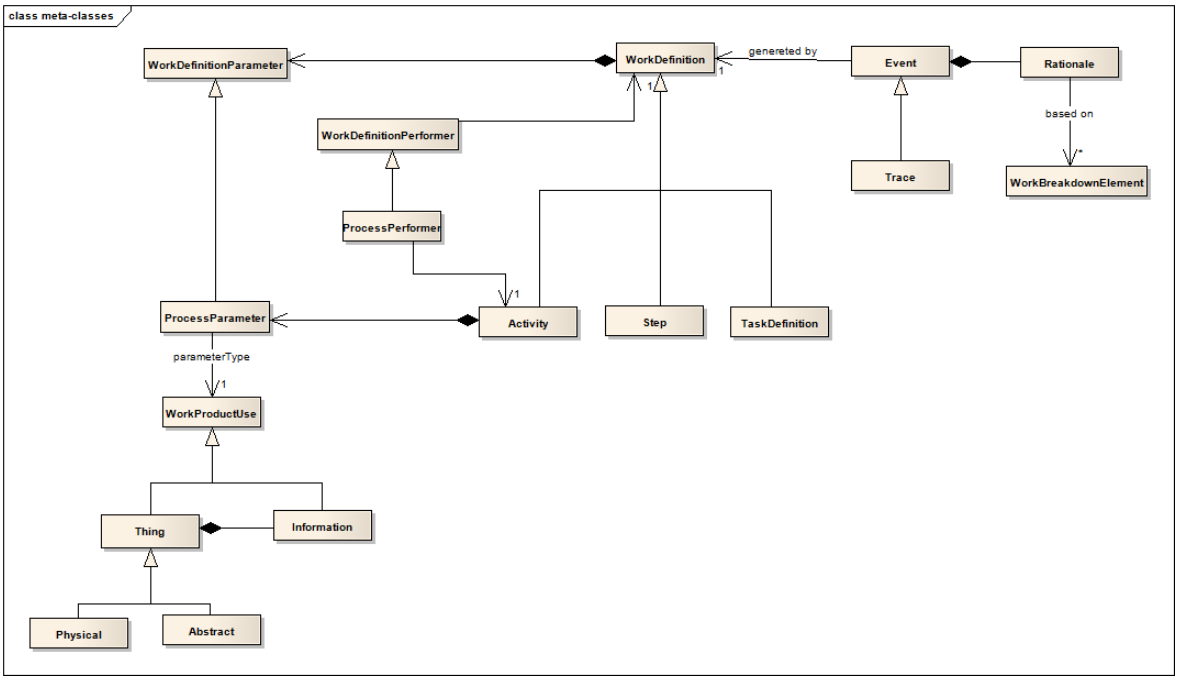


Figura 3.4. Apresentação parcial do Metamodelo RAISE

ser representado como "*Information*", e quando for o artefato onde a informação está contida deve ser do tipo "*Thing*".

A dimensão "Por que certo elemento é criado ou evoluído?" foi modelada no RAISE baseando-se no submodelo de Ramesh e Jarke [Ramesh & Jarke, 2001] que representa a classe "*Rationale*" ligada à classe "*object*" pela relação "*based on*". Essa ligação indica que fundamentações, razões, podem ser baseadas em outros elementos ("*Objects*"). Na Figura 3.4 se vê então o elemento "*Event*" possuindo o elemento "*Rationale*", que se baseia em algum "*Breakdown Element*". Considerando que "*Activity*" e "*Work Product Use*" são especializações do "*Work Breakdown Element*", a ligação indica que qualquer recurso ou atividade do processo de desenvolvimento de software pode ser usado para justificar um evento ("*Event*") e consequentemente, um Rastro ("*Trace*").

A última dimensão, "Quando a informação foi capturada, modificada ou evoluída?", foi modelada como um atributo do "*Work Definition*" indicando que o Rastro foi feito no momento da execução da tarefa.

A tarefa "Definição de Escopo" do processo Praxis [de Pádua Paula Filho, 2003] é usada como exemplo da utilização deste metamodelo parcial. Um dos passos da tarefa é "identificar os benefícios do produto". A Figura 3.5 mostra uma proposta de modelo, orientado ao metamodelo parcial da Figura 3.4, para o passo exemplificado. Observe que o Rastro proposto possui o nome de "*pre-requirement*". Isto se deve ao fato

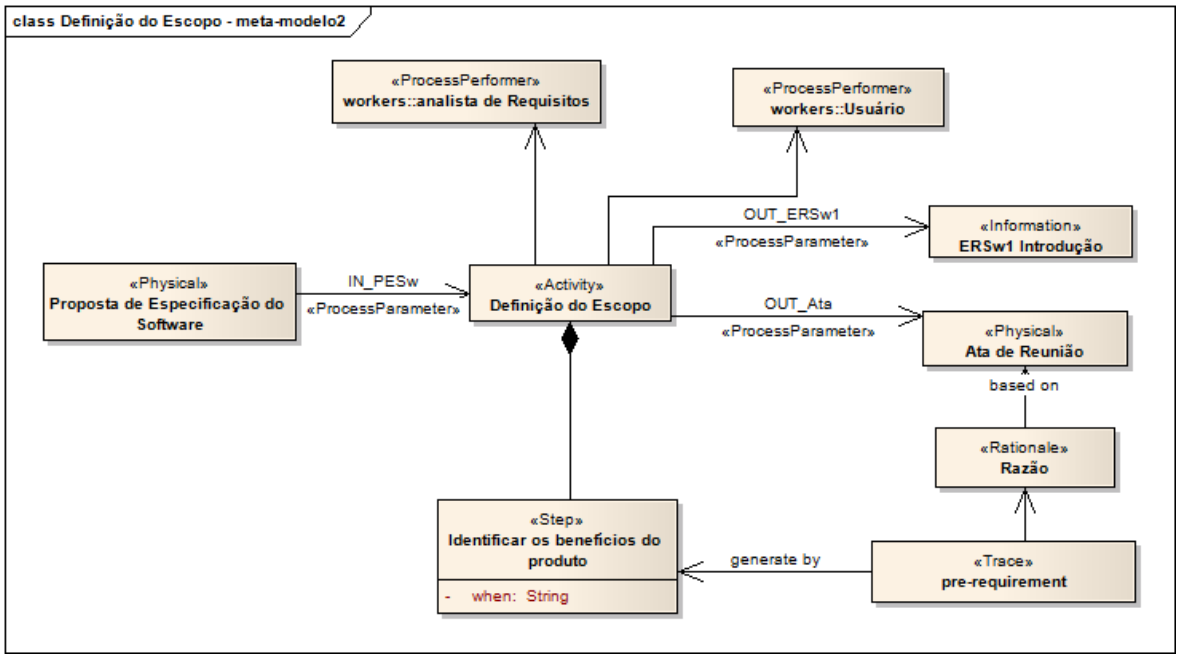


Figura 3.5. Exemplo de um modelo baseado no Metamodelo RAISE Parcial

que o que é registrado obedece ao conceito de "*pre-requirement*" proposto por Gotel e Finkelstein [Gotel & Finkelstein, 1994]. Na Figura 3.6 pode-se ver um Rastro gerado por este modelo. O Rastro indica que o artefato "ERSw 1 Introdução" foi atualizado na execução do passo "Identificar os benefícios do produto" da tarefa "Definição do Escopo" em 05/06/2010, executado por João, analista de requisitos, Márcio, Usuário e Carlos, usuário. As razões que levaram ao artefato final estão registradas na ata de reunião ATA20100605.

Desta forma, o metamodelo proposto satisfaz então a definição de Gotel e Finkelstein, a definição de Rastro do Oxford English Dictionary e as dimensões de Ramesh e Jarke para Rastreabilidade de requisitos, mas ainda não está completo. Outra definição de Rastreabilidade, que está presente no IEEE Standard Glossary of Software Engineering Terminology [IEEE, 1990], abrange mais do que apenas a evolução do requisito, mas qualquer tipo de ligação de Rastreabilidade que possa existir durante o desenvolvimento de software.

> "A Rastreabilidade é:
>
> 1. O grau em que cada relacionamento possa ser estabelecido entre dois ou mais produtos do processo de desenvolvimento, especialmente a relação predecessor-sucesso ou mestre-subordinado. [...]
>
> 2. O grau em que cada elemento do produto do desenvolvimento de software estabelece sua razão de existir."

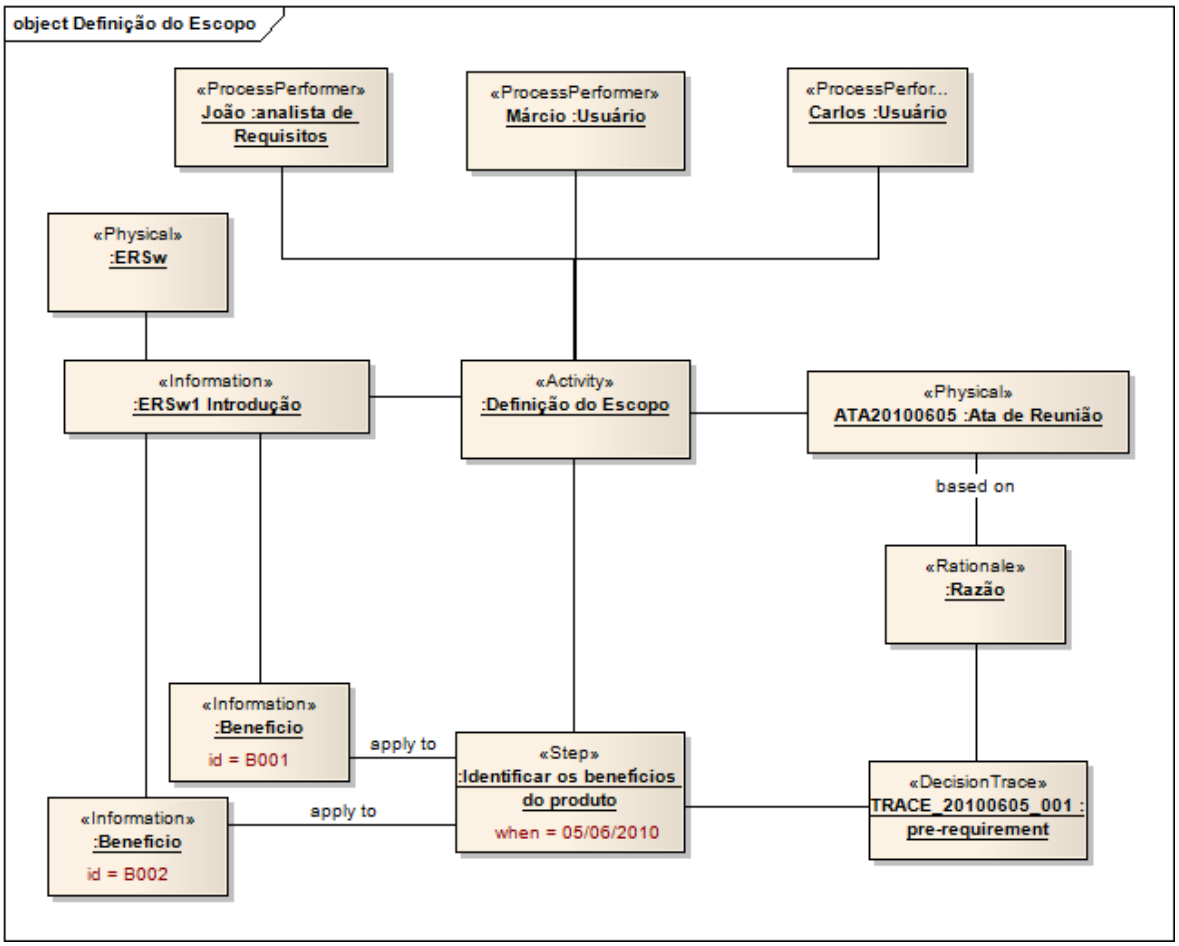


Figura 3.6. Exemplo de um Rastro gerado pela tarefa Definir Escopo

A definição engloba qualquer tipo de ligação de Rastreabilidade, mas evidencia três tipos:

- Relação predecessor-sucessor.
- Relação mestre-subordinado.
- Relação de fundamentação.

Primeiramente, faz-se necessário observar que a definição restringe a informação de Rastreabilidade apenas às relações entre elementos. Não está incluído na definição que um Rastro também possa ser "uma evidência ou indicação do que existiu". A definição inclui apenas que um Rastro possa ser uma evidência "do que aconteceu". Nesse aspecto, o Metamodelo RAISE é mais abrangente do que essa definição. Mas também se deve notar que o Metamodelo RAISE parcial não satisfaz essa definição por completo, pois não representa o grau em que cada relacionamento entre dois ou

mais produtos do processo de desenvolvimento pode ser estabelecido; apenas registra eventos dos relacionamentos. Também não representa o grau em que cada elemento do produto do desenvolvimento de software estabelece sua razão de existir; apenas mostra que um evento possui uma razão para ocorrer.

Os graus das relações são categorizados na definição através dos "tipos de ligação" e a definição ainda destaca algumas categorias. O SPEM 2.0 permite representar estas relações através da meta-classe "*Work Product Use Relationship*", mas não deixa claro qual é a categoria da relação que está representando. No Metamodelo RAISE se faz necessário categorizar a classe "*Trace*" para representar os diferentes graus das relações, e por isso a meta-classe "*Work Product Use Relationship*" não foi utilizada. Na Seção 2.2 foram mostrados diversos tipos de Ligações de Rastreabilidade que poderiam ser usados como categorias no metamodelo e na Seção 2.5 foram mostrados diferentes metamodelos de informação de Rastreabilidade com propostas de categorias para o Rastro. Todas as propostas apresentando outras categorias além das três destacadas na definição IEEE Standard Glossary of Software Engineering Terminology.

Dessas propostas, a do Pinheiro [Pinheiro, 2003], de classificar os Rastros como funcionais e não-funcionais, foi utilizada no metamodelo RAISE. Para Pinheiro "Rastros não-funcionais são aqueles relacionados à intenção, propósito, metas, responsabilidades e outros conceitos intangíveis. Por outro lado, os Rastros funcionais são os relacionados com mapeamento bem estabelecidos entre objetos" [3]. Os Rastros funcionais possuem regras.

Os Rastros não-funcionais são classificados em quatro grupos:

a) Razão, compreendendo, dentre outros, os Rastros com a justificativa e interpretação de requisitos e outros artefatos, para definição de metas e propósitos, e para a explanação de tarefas.

b) Contexto, compreendendo, dentre outros, os Rastros relacionados com o ambiente da organização e aspectos sociais.

c) Decisão, compreendendo, dentre outros, os Rastros relacionados com decisões feitas durante o processo de desenvolvimento de software, e para as comunicações entre as Partes Interessadas e desenvolvedores. [4]

[3] Do inglês: "Non-functional traces are those traces related to intentions, purpose, goals, responsibilities, and other intangible concepts. On the other hand, the functional traces are those related to well established mappings between objects."

[4] Do inglês: a) Reason, comprising, among others, the traces related to the justification and interpretation of requirements and other artefacts, to the definition of goals and purpose, and to the explanation of activities.

b) Context, comprising, among others, the traces related to the environment of both the application domain and the software development process, including organisation and social aspects;

c) Decision, comprising, among others, the traces related to the decisions made during the software

A classificação de Pinheiro é interessante por permitir separar os Rastros estruturados dos descritivos. Os Rastros funcionais são regidos por regras, enquanto os não-funcionais devem ser registrados justamente por não serem regidos por regras claras. Um exemplo de Rastro não-funcional é o Rastro "pre-requirement" do tipo decisão(Figura 3.5). Os benefícios listados no documento ERSw1 Introdução são decisões tomadas durante comunicações entre usuários e desenvolvedores.

A Figura 3.7 detalha os tipos de Rastro ("*Trace*") no metamodelo RAISE. Nessa figura se observa que o primeiro nível de classificação de um Rastro segue a sugerida por Pinheiro. Mas a classificação de Pinheiro é mais genérica do que as categorias destacadas na definição do IEEE Standard Glossary of Software Engineering Terminology (Relação predecessor-sucessor, mestre-subordinado, Relação de fundamentação).

A relação predecessor-sucessor é uma relação existente entre dois elementos que são gerados em sequência temporal. O seu vínculo é em função da evolução do ciclo do desenvolvimento de software que deve ocorrer entre elementos de entrada e saída de um "*Work Definition*". O SPEM 2.0 já permite o registro desse vinculo através da meta-classe "*Work Product Use Relationship*". Do ponto de vista do Metamodelo RAISE, um Rastro que permite essa relação é um Rastro funcional (Figura 3.7).

Já a relação mestre-subordinado não é uma ligação vinculada à evolução do ciclo de desenvolvimento de software, mas representa um vínculo de dependência entre os elementos, que também é um vinculo funcional.

A última relação indicada nesta definição é a ligação de fundamentação, que ocorre quando determinados elementos são a razão de outros existirem. Observa-se que não há necessariamente uma relação de temporalidade entre os elementos, pois os elementos que fundamentam podem ser criados após os outros elementos. Na classificação de Pinheiro essa relação é não-funcional.

Mas estas relações ainda não caracterizam subclasses do Rastro, "*Trace*", por serem incompletas.

Há conceituações no âmbito das transformações usadas no MDE que ajudam a entender melhor os Rastros funcionais. Falleri e outros [Falleri et al., 2006] propuseram algumas definições para transformações. A definição de número três tem uma relevância para o Metamodelo RAISE por se referir às ligações da transformação. Deve-se considerar essa definição avaliando-a no contexto da Rastreabilidade. As ligações de

development process, and to the communications among the stakeholders and developers.

d) Technical, comprising, among others, the traces related to non-functional requirements.

"These groups may overlap and in practice non-functional traces are seldom classified into only one group. For example, a reason is usually situated in a context and it may be given to support some decision related to a non-functional requirement."

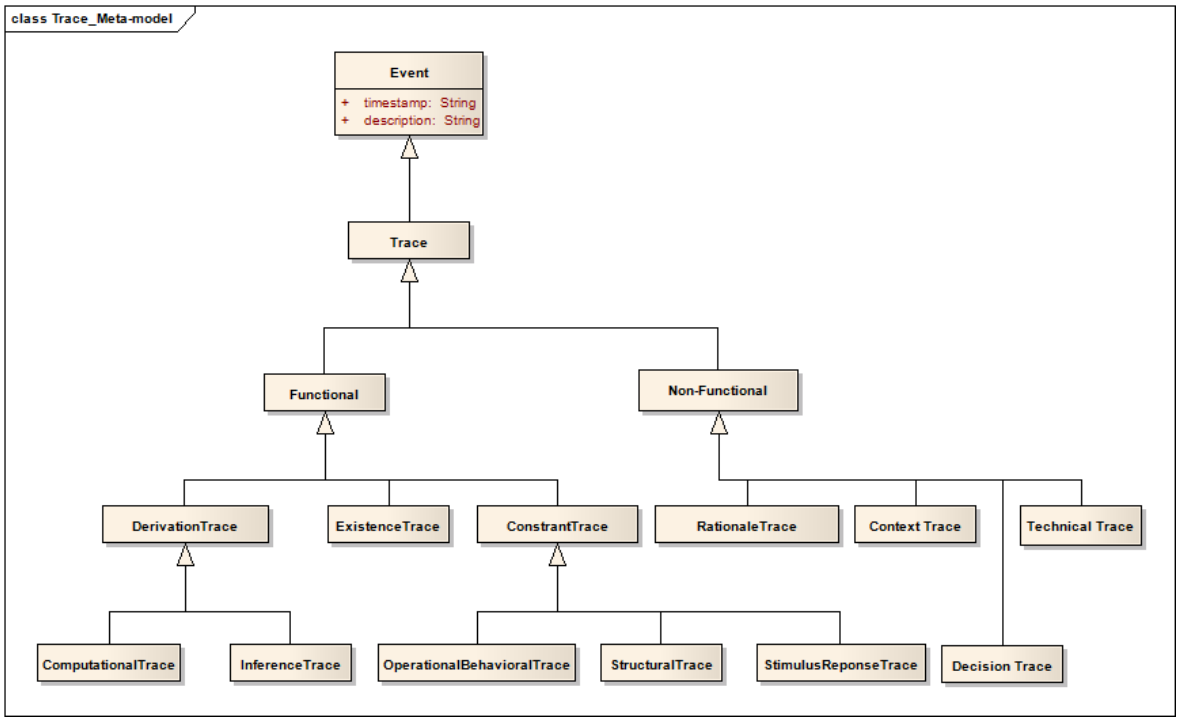


Figura 3.7. Destaque dos tipos de Rastros no meta-modelo

transformações podem ser vistas como Ligações de Rastreabilidade [Winkler & von Pilgrim, 2009].

A definição três, de Falleri e outros, diz que "a ligação de Rastreabilidade de transformação é um grafo bipartido. Os nodos são particionados em duas categorias: nodos fontes e nodos destinos". [5]

Nesse contexto, os nodos são objetos de modelos participantes da transformação. A ligação indica que os elementos destinos são resultados da transformação a partir de elementos origens. Pode-se inferir que a transformação considerada nessa definição é o resultado de alguma tarefa (ou etapa de uma tarefa) do ciclo de desenvolvimento de software. Sendo assim, aplicando a definição ao contexto do Metamodelo RAISE, pode-se afirmar que a ligação de transformação é um Rastro funcional. Pinheiro diz que os Rastros funcionais são relacionados com o aspecto funcional do desenvolvimento de software, que podem ser descritos em termos de transformações de estados bem definidos do desenvolvimento de software, seus modelos e artefatos. [6]

Tanto a definição de Pinheiro para Rastros funcionais, quanto a de Falleri para

[5]Do inglês: "A transformation trace is a bipartite graph. The nodes are partitioned into two categories: source nodes and target nodes".

[6]Do inglês: "Functional tracing is related to the functional aspects of software development, those that may be described in terms of transformations on well-defined states of software development, its models, and its artefacts."

transformação, ressaltam a existência de regras que regem a transformação. Pode-se ainda complementar o entendimento de Rastros funcionais através da descrição feita por Winkler e Pilgrim [Winkler & von Pilgrim, 2009]:

> "Rastros funcionais são estes criados para transformar um artefato em outro usando um conjunto de regras bem definidas. A transformação não necessariamente deve ser executada de forma automática, mas deve seguir procedimentos inequívocos, como uma transcrição de uma gravação em áudio de uma entrevista. Jacobson e Lindvall chamam isso de "*seamless transformation*". Um subconjunto de Rastros funcionais é o conjunto de Rastros explícitos que são criados juntos com o artefato como um subproduto da transformação, ou aqueles que podem ser inequivocamente reconstruídos a qualquer momento analisando os artefatos originais, o artefato transformado, e as regras de transformações. Rastros funcionais, e particularmente Rastros explícitos, são também usualmente encontrados em modelos que possuem sintaxe e semântica formalmente definidas". [7]

Desta forma pode-se concluir que **as regras da transformação são as regras da tarefa executada**, pois a transformação é o resultado da tarefa especificada. **O Rastro funcional registra as ligações entre os elementos de entrada e saída das regras de negócio.** Essa é uma conclusão relevante para a categorização dos tipos de Rastros funcionais. Os Rastros podem ser categorizados segundo o tipo de regra das tarefas executadas que os regem [grifo intencional].

No SPEM as tarefas executadas são representadas pelo "*Work Definition*", mas não há representação explícita das regras que regem o trabalho (vide Seção 3.2). Essas regras são representadas no Metamodelo de Eriksson e Penker como regras de negócio "*Rule*". E foram categorizadas em "*Constraint*", "*Derivation*" e "*Existence*".

O Metamodelo RAISE utilizou as regras ("*Rules*") de Eriksson e Penker, mas especializadas para o processo de desenvolvimento de software. As regras foram detalhadas por tipos de regras de negócio que podem ser aplicadas ao processo de desenvolvimento de software e mapeadas no metamodelo. Esse detalhamento pode ser visto na Figura3.8 junto com todo o Metamodelo RAISE.

[7] Do inglês: "Functional traces These are created by transforming one artifact into another using a defined rule set. The transformation is not required to be performed automatically,but it has to follow unambiguous procedures, such as transcribing an audio recording of an interview. Jacobson and Lindvall attribute this as a seamless transformation. A subset of functional traces is the set of explicit traces which are either created together with the artifact as a by-product of the transformation, or which can be unambiguously reconstructed at any time by analyzing the original artifact, the transformed artifact, and the transformation rules. Functional traces, and particularly explicit traces are also usually found in models with formally defined syntax and semantics."

Odell [Odell, 1998] diz que uma regra de derivação, "*Derivation*", pode ser divida em duas subcategorias: Regras de inferências ("*Inference*"), e regras computacionais ("*Computacional*"). As regras de inferências especificam conclusões que podem ser tiradas quando certos fatos são verdadeiros. Por exemplo, a tarefa de análise "Identificação das Classes" do Praxis, que descreve a identificação das classes de análise do produto com base principalmente nos fluxos iniciais de casos de uso (Figura 3.9). As classes são inferidas dos fluxos de caso de uso através de regras bem definidas no Praxis:

Classes são extraídas de substantivos existentes nos fluxos, desde que os seguintes fatos sejam verdade:

- Os substantivos não devem conter aspectos de implementação e informação irrelevante quanto à missão do produto;
- Os substantivos devem resultar em classes, não em objetos, relacionamentos ou atributos de classes;
- Os substantivos devem ser do nível lógico, não incluindo detalhes de interfaces, arquivos, estruturas de dados etc.;
- Os substantivos devem pertencer ao escopo do produto, evitando classes não conexas com a missão deste.

A descrição destas regras pode ser feita através de listas textuais, como a lista mostrada acima, mas, caso seja possível a automação, uma linguagem de transformação como o ATL ("*Atlas Transformation Language*") [Jouault & Kurtev, 2005] ou QVT (Query View Trasnformation) [OMG, 2008a; Jouault & Kurtev, 2006] pode ser usada.

As regras computacionais são matemáticas por natureza e são expressas como uma equação. Elas são similares às regras de inferência por derivarem seus resultados de alguma outra informação. Regras computacionais derivam seus resultados do processamento de um algoritmo, e são usadas para derivar atributos ou comportamento de operações. Um exemplo dessa regra pode ser encontrado na tarefa "Realização dos Casos de Uso" do Praxis, no passo de "Cadastramento dos itens de análise" quando se usa APF (Análise de Ponto de Função) [Vazquez et al., 2010] para estimativa de tamanho. O processo pede para que seja identificado o TAR (Número de classes persistentes que compõem o arquivo lógico) por classe de fronteira. Esse número pode ser calculado a partir do modelo de colaboração. Modelo que apresenta os fluxos que a partir de uma classe de fronteira se alcança as classes persistentes. Um algoritmo que segue este fluxo pode ser usado para calcular o TAR.

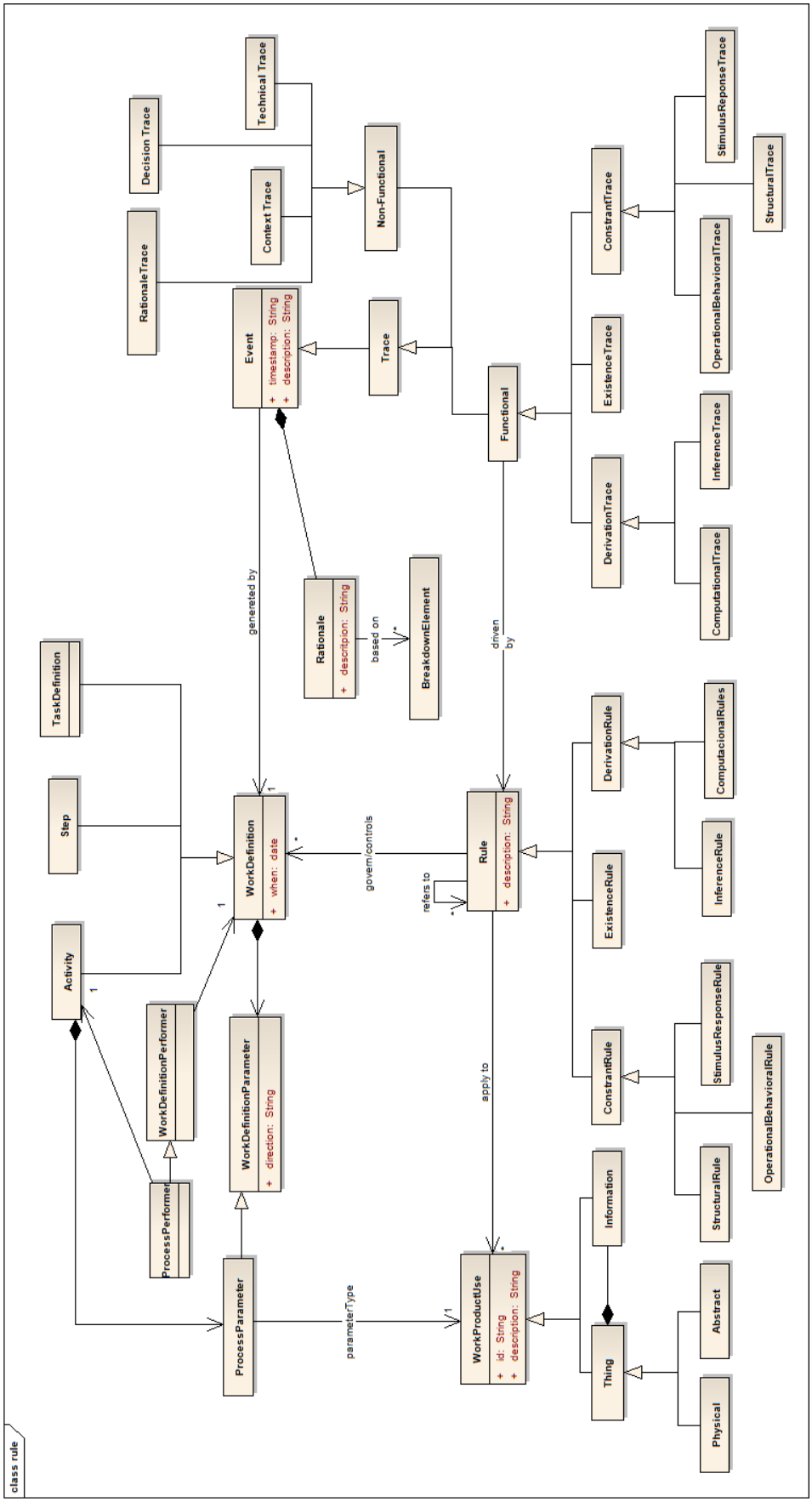


Figura 3.8. Metamodelo de Informação da Rastreabilidade RAISE

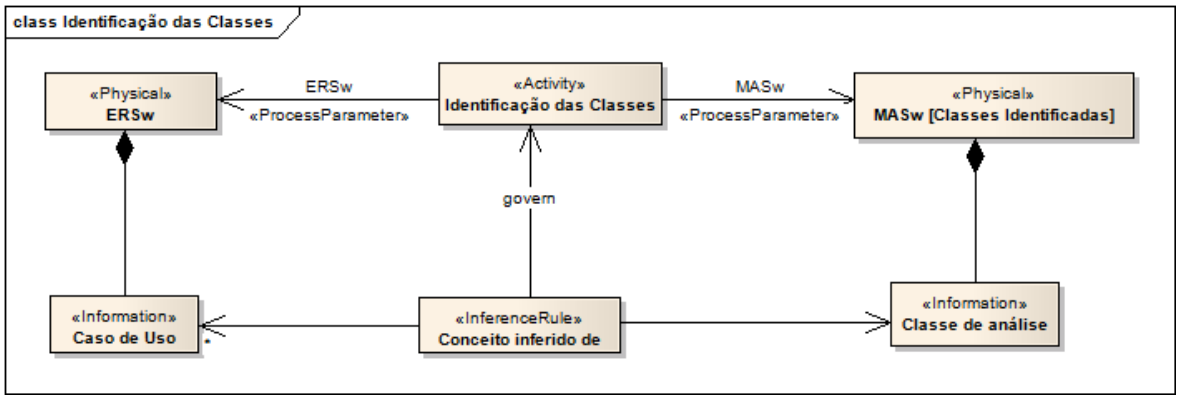


Figura 3.9. Exemplo de Rastro de Inferência

As regras de restrições governam a possível estrutura ou comportamento de objetos ou processos; isto é, a forma como os objetos se relacionam ou quando uma mudança de estado de um processo pode ocorrer. Restrições são usadas para apoiar a integridade de objetos de negócio quando são criados e seus relacionamentos alterados, e para controlar o comportamento de objetos e processos. Restrições, "*Constraints*", são divididas em três categorias: "*Structural*", "*Operational/Behavioral*", e "*Stimulus/Reponse*".

Regras de restrições estruturais, "*Structural*", são aplicadas sobre tipos de objetos de negócio e associações entre eles, e especificam que algumas condições que regulam a estrutura precisam sempre ser verdadeiras. Expressam o aspecto estático do negócio. Estas regras podem ser documentadas utilizando-se os modelos estáticos da UML junto com a linguagem OCL. Por exemplo, no Praxis, o ERSw, na Seção 2.1.2, pede para se documentar a coleção ordenada de telas do sistema. Cada tela, sem exceção, deve estar associada a um, e somente um, caso de uso identificado e documentado na Seção 2.1.1 (como visto na Figura 3.10). Essa é uma regra estrutural. Sua especificação permite que a verificação do documento possa ser parcialmente automatizada, já que não há grandes dificuldades em verificar se cada tela está associada a um caso de uso especificado na Seção 2.1.1.

Restrições do tipo operacional/comportamental,"*Operational/Behavioral*", definem pré e pós-condições que restringem o que deve ser verdade antes ou após uma tarefa ser executada. Uma pré-condição expressa restrições sobre o que deve ser verdade antes de se executar uma tarefa. Uma pós-condição indica o que deve ser verdade após a tarefa. Este tipo de regra indica a mudança de estado do sistema e pode ser expressa através de um modelo de estado UML. O grau de especificação dos estados participantes da restrição define o grau de automação que esta regra permite. Por exemplo, a tarefa "Revisão dos Requisitos" no Praxis possui uma regra de restrição do tipo operacional/comportamental, indicando que o requisito precisa estar no estado

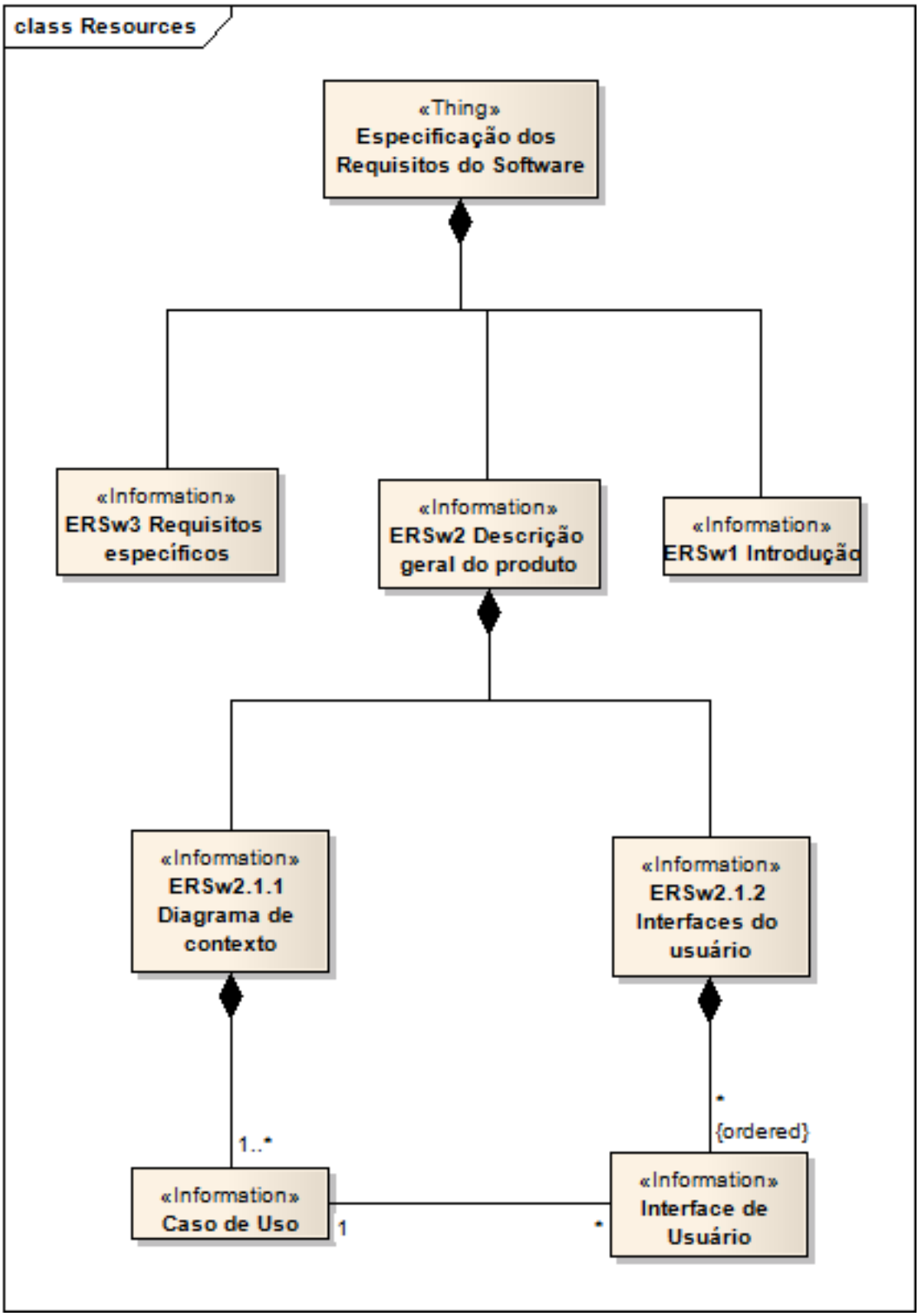


Figura 3.10. Exemplo de Regra Estrutural

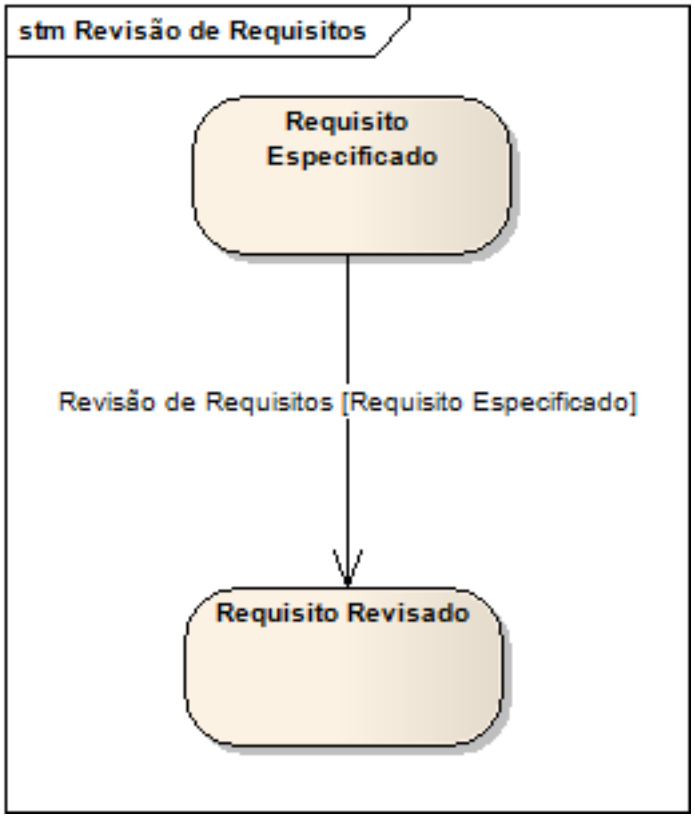


Figura 3.11. Exemplo da Restrição Operacional/Comportamental

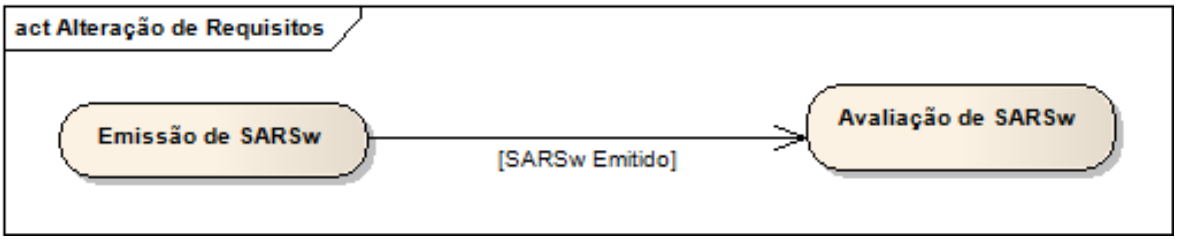


Figura 3.12. Exemplo de Restrição de Estimulo/Resposta

especificado antes de ser revisado (Figura 3.11).

Restrições de estímulo/resposta, "*Stimulus/Response*", especificam que certas ações deveriam ser executadas quando certos eventos são gerados no negócio. Eles possuem a forma de "Quando isto ocorrer, faça aquilo". Estas regras podem ser expressas através de um diagrama de tarefas. Um bom exemplo da aplicação desta restrição está na tarefa Avaliação da SARSw (Solicitação de Alteração de Requisitos). Esta tarefa deve ser executada sempre que for emitido um SARsw (Figura 3.12).

Por fim, as regras de existência, "*Existence*", governam quando um objeto de negócio específico pode existir.

Essas regras de negócio estão associadas às Ligações de Rastreabilidade funcionais e os Rastros devem ser diferenciados para cada tipo de regra de negócio [grifo intencional].

Essa visão vinculando Ligações de Rastreabilidade com as regras de negócio lança um novo entendimento sobre a definição de tipos de Ligações de Rastreabilidade destacados pelo IEEE Standard Glossary of Software Engineering Terminology [IEEE, 1990]. O IEEE Standard Glossary evidência as relações predecessor-sucessor, mestre-subordinado, Relação de fundamentação. A relação predecessor-sucessor (de evolução) é a indicação ou evidência de que uma regra de negócio de derivação foi executada. Assim como a relação mestre-subordinado (de dependência) indica ou evidencia que uma regra de restrição foi aplicada. O último tipo de ligação destacado foi a relação de fundamentação, que é não-funcional e, portanto, não está vinculada a nenhuma regra.

Desta forma, faz-se coerente subclassificar os Rastros funcionais com tipos correspondentes aos das regras que eles indicam ou evidenciam: "*DerivationTrace*", com suas subclasses, "*ComputationalTrace*" e "*InferenceTrace*"; "*ConstrantTrace*" com as subclasses "*OperacionalBehaviorTrace*", "*StructuralTrace*" e "*StimulusResponseTrace*"; e por fim o "*ExistenceTrace*". Todas estas classes estão na Figura 3.7.

As regras foram os últimos elementos que precisavam ser explorados para o entendimento completo do Metamodelo RAISE. Nesta seção, então, foi apresentado o metamodelo, que tem como base elementos do processo de desenvolvimento de software e definições de Rastreabilidade. Percebe-se que essa representação é mais natural e mais próxima do uso do contexto onde a Rastreabilidade está inserida que os outros modelos já propostos. A seção seguinte, 3.3.2, lista todos os elementos do metamodelo com sua descrição e atributos.

3.3.2 Elementos do Metamodelo RAISE

3.3.2.1 Abstract

Descrição:

"*Abstract*" é um "*Thing*" que representa uma ideia ou conceito, normalmente uma composição de outros objetos. Envolve coisas ou conceitos que não são físicos e não podem ser tocados, mas são de importância para o negócio.

3.3.2.2 Activity

Descrição:

"Activity" é um "*WorkBreakdownElement*" e um "*WorkDefinition*" que define unidades básicas de trabalho em um processo. Pode ser decomposto em etapas ("*Step*").

3.3.2.3 BreakdownElement

Descrição:

"BreakdownElement" é uma generalização abstrata para qualquer tipo de elemento de uma repartição do processo.

3.3.2.4 ComputacionalRule

Descrição:

Regras computacionais ("*ComputacionalRule*") são Regras de Derivação ("*DerivationRule*") e derivam seus resultados do processamento de um algoritmo, e são usadas para derivar atributos ou comportamento de operações.

3.3.2.5 ComputacionalTrace

Descrição:

São Rastros Funcionais ("*Functional*") dirigidos para registrar eventos gerados a partir de uma regra de negócio ("*Rule*") de algum "*WorkDefinition*".

3.3.2.6 ContextTrace

Descrição:

São Rastros Não-Funcionais ("*Non-Functional*") compreendendo, dentre outros, os Rastros relacionados com o ambiente da organização e aspectos sociais.

3.3.2.7 ConstraintRule

Descrição:

Regras de restrições ("*ConstraintRule*") restringem tanto a estrutura quanto o comportamento de objetos ou processos. Restringem a forma como objetos são relacionados uns com outros ou a forma como mudanças de estados de objetos ou processos podem ocorrem.

3.3.2.8 ConstraintTrace

Descrição:

São Rastros funcionais ("*Functional*") dirigidos pela regra de negócio ("*Role*") "*ConstraintRule*".

3.3.2.9 DecisionTrace

Descrição:

São Rastros Não-Funcionais ("*Non-functional*") compreendendo, dentre outros, os Rastros relacionados com decisões feitas durante o processo de desenvolvimento de software, e para as comunicações entre as Partes Interessadas e desenvolvedores.

3.3.2.10 DerivationRule

Descrição:

Regras de derivação definem como o conhecimento ou a informação pode ser transformada em outro conhecimento ou informação.

3.3.2.11 DerivationTrace

Descrição:

São Rastros Funcionais ("*Functional*") dirigidos pela regra de negócio ("*Role*") "*DerivationRule*".

3.3.2.12 Event

Descrição:

É um evento ocorrido no ciclo do desenvolvimento de software.

Atributos:

timestamp: o atributo deve ser preenchido com uma sequência de caracteres indicando a data e hora que o evento ocorreu.

description: O atributo deve ser preenchido com o registro da descrição do evento, caso seja necessária.

3.3.2.13 ExistenceRule

Descrição:

Regras de existência ("*ExistenceRule*") definem sobre que circunstâncias alguma coisa pode existir (o ciclo de vida de um objeto) e quando pode vir a existir (isto é, quando um objeto é criado ou destruído).

3.3.2.14 ExistenceTrace

Descrição:

São Rastros funcionais ("*Functional*") dirigidos pela regra de negócio ("*Role*") "*ExistenceRule*".

3.3.2.15 Functional

Descrição:

Os Rastros funcionais ("*Functional*") são os relacionados com mapeamento bem estabelecidos entre objetos. Os Rastros funcionais são regidos por regras. O aspecto funcional do desenvolvimento de software, que podem ser descritos em termos de transformações de estados bem definidos do desenvolvimento de software, seus modelos e artefatos são representados pelos Rastros Funcionais.

3.3.2.16 InferenceRule

Descrição:

As regras de inferências ("*InferenceRule*") especificam conclusões que podem ser tiradas quando certos fatos são verdadeiros.

3.3.2.17 InferenceTrace

Descrição:

São Rastros Funcionais ("*Functional*") dirigidos pela regra de negócio ("*Role*") "*InferenceRule*".

3.3.2.18 Information

Descrição:

"*Information*" é um "*WorkProductUse*" que representa um conceito, coisa, ou outro objeto de informação.

3.3.2.19 ProcessParameter

Descrição:

Um "*Process Parameter*" é um "*Work Definition Parameter*" e um "*Breakdown Element*" que define tipo de entradas e saídas que serão os "*Work Product Use*". Deve ser representado como uma associação entre uma Activity e um WorkProductUse.

3.3.2.20 Step

Descrição:

Um "*Step*" é um "*WorkDefinition*" que descreve o desenvolvimento de uma etapa dentro de um trabalho maior.

3.3.2.21 TaskDefinition

Descrição:

"TaskDefiniton" é um "WorkDefinition" que define o trabalho executado por uma tarefa que subdividade em "*Steps*".

3.3.2.22 Non-Functional

Descrição:

Rastros não-funcionais ("*Non-Functional*") são Rastros ("*Traces*") relacionados à intenção, propósito, metas, responsabilidades e outros conceitos intangíveis.

3.3.2.23 OperationalBehavioralRule

Descrição:

Restrições do tipo operacional e comportamental,"*Operational/Behavioral*", definem pré e pós-condições que restringem o que deve ser verdade antes ou após uma tarefa ser executada. Uma pré-condição expressa restrições sobre o que deve ser verdade antes de se executar uma tarefa. Uma pós-condição indica o que deve ser verdade após a tarefa.

3.3.2.24 OperationalBehavioralTrace

Descrição:

São Rastros funcionais ("*Functional*") dirigidos pela regra de negócio ("*Role*") "*OperationalBehavioralRule*".

3.3.2.25 Physical

Descrição:

Físico ("*Physical*") é um "WorkProductUse" com realidade material que ocupa um volume no espaço. É alguma coisa que pode ser tocado.

3.3.2.26 ProcessPerformer

Descrição:

"*ProcessPerformer*" é um "*BreakdownElement*" que representa o executor da atividade ("*Activity*").

3.3.2.27 Rationale

Descrição:

"*Rationale*" indica que a razão de um determinado "Event" ocorrer. Pode ser baseada em outros "*BreakdownElements*".

Atributos:

description: Deve ser preenchido com a descrição da razão do evento estar relacionado com algum "BreakdownElement".

3.3.2.28 RationaleTrace

Descrição:

São Rastros Não-Funcionais ("*Non-functional*") compreendendo, dentre outros, os Rastros com a justificativa e interpretação de requisitos e outros artefatos, para definição de metas e propósitos, e para a explanação de tarefas.

3.3.2.29 Rule

Descrição:

Uma regra de negócio ("*Rule*") é uma sentença que pode controlar ou afetar a execução de um processo de negócio tanto quanto as estruturas dos recursos do negócio.

Atributos:

description: deve ser preenchido com a descrição da regra. Essa descrição pode ser na forma textual ou utilizando alguma linguagem como ATL ou QVT. Também é possível usar um modelo para descrever a regra, como por exemplo, um modelo UML.

3.3.2.30 StimulusResponseRule

Descrição:

Restrições de estímulo/resposta, "*Stimulus/Response*", especificam que certas ações deveriam ser executadas quando certos eventos são gerados no negócio.

3.3.2.31 StructuralRule

Descrição:

Regras de restrições estruturais, "*Structural*", são Regras Funcionais ("*Functional*") aplicadas sobre tipos de objetos de negócio e associações entre eles, e especificam que algumas condições que regulam a estrutura precisam sempre ser verdadeiras. Expressam o aspecto estático do negócio.

3.3.2.32 StructuralTrace

Descrição:

São Rastros Funcionais ("*Functional*") dirigidos pela regra de negócio ("*Role*") "*StructuralRule*".

3.3.2.33 Thing

Descrição:

"*Thing*" é um "*WorkProductUse*" que possui a informação concreta representada pelo elemento "*Information*".

3.3.2.34 TechnicalTrace

Descrição:

"TechnicalTrace" é um Rastro Não-Funcional ("*Non-Functional*") compreendendo, dentre outros, os Rastros relacionados com requisitos não-funcionais.

3.3.2.35 Trace

Descrição:

"*Trace*" é um "*Event*" que indica ou evidencia o que existiu ou aconteceu.

3.3.2.36 WorkBreakdownElement

Descrição:

Representa elementos do processo que descrevem trabalhos realizados.

3.3.2.37 WorkDefinition

Descrição:

O trabalho executado no processo de desenvolvimento de software é especificado pelo elemento *"Work Definition"*. *"Work Definition"* é uma meta-classe abstrata que generaliza toda definição de trabalho.

3.3.2.38 WorkDefinitionPerformer

Descrição:

Representa o relacionamento entre o "Work Performer" com o "Work Definition".

3.3.2.39 WorkDefinitionParameter

Descrição:

É o elemento que representa os parâmetros de entrada e saída do "*Work Definition*". Deve ser representado como uma associação entre WorkDefinition e outro elemento do modelo.

Atributos:

direction: Pode ser "in" ou "out" para representar parâmetros de entrada e saída respectivamente.

3.3.2.40 WorkProductUse

Descrição:

O "*WorkProductUse*" é um "BreakdownElement" que representa um tipo de entrada ou saída para uma tarefa ou representa um participante da tarefa.

Atributos:

id: deve ser preenchido com um identificador da instância do WorkProductUse.

description: deve ser preenchido com uma descrição da instância do WorkproductUse, caso o identificador não seja suficiente.

3.4 Considerações sobre o Metamodelo de Informação RAISE

3.4.1 RAISE na Automação da Engenharia de Software

Uma consequência da abordagem de vincular o Metamodelo de Informação da Rastreabilidade aos elementos do processo de desenvolvimento de software é que o metamodelo pode ser mais facilmente utilizado na automação do desenvolvimento de software. Isso ocorre porque a sintaxe da ligação expressa no metamodelo está vinculada às tarefas de desenvolvimento de software que podem ser suportadas por automação. A definição de um Rastro funcional nos parece indicar também possibilidade de automação conforme explicado a seguir. Considere que existam Rastros que mostram a derivação de um elemento a partir de outro. Para esses Rastros o metamodelo pode estar vinculado às tarefas que executam a derivação. A descrição das tarefas pode indicar como a derivação ocorre através de uma linguagem apropriada (como por exemplo, o QVT da OMG [OMG, 2008a] ou ATL do projeto Eclipse [Jouault & Kurtev, 2005]). Rastros desse

tipo são os que existem, por exemplo, entre o modelo conceitual resultado da análise do sistema e o modelo dependente de plataforma resultado do desenho do projeto.

Também há Rastros que simplesmente mostram como determinados elementos devem se relacionar seguindo determinadas regras estruturais. Por exemplo, a regra que todo Caso de Uso deve satisfazer alguma necessidade do cliente. Nesse caso, existe uma relação entre elementos da Engenharia de Software que podem ser descritas através de modelos estáticos UML, utilizando OCL, onde se evidenciam as relações destes elementos. Cada Caso de Uso deve ser ligado a uma ou mais necessidade do cliente.

Esses dois tipos de Ligações de Rastreabilidade exemplificados possuem sintaxes diferentes, mas quando explicitadas, podem auxiliar na automação do desenvolvimento de software. No caso, os Rastros de derivação podem ser implementados usando mecanismos de transformação. Já as ligações estruturais podem ajudar na integridade entre os elementos da Engenharia de Software. Pode-se automatizar a criação dos elementos, inibir as inconsistências, ou criar mecanismos de verificação para identificar as inconsistências. De qualquer forma, conhecendo as Ligações de Rastreabilidade pela sua sintaxe se torna possível utilizá-la na automação do desenvolvimento.

Existem muitas ferramentas disponibilizadas no mercado que propõem diversas formas de automatização do processo de desenvolvimento de software [Aksyonov et al., 2009], e que, para se adaptarem a realidade da organização e dos projetos, estas ferramentas apresentam diferentes formas de personalizações. Mas a complexidade, o esforço requerido, e o tempo necessário para adaptar os softwares às particularidades de cada organização e projetos, não permitem, ou, pelo menos dificultam muito, que as personalizações ocorram da melhor forma possível [Cordy, 2003; Xiaoli, 2009].

Uma abordagem para se resolver o problema, usada por Amelunxen e outros [Amelunxen et al., 2008], propõe que essas personalizações sejam obtidas a partir de um modelo que relaciona elementos da Engenharia de Software mantidos por cada uma das ferramentas envolvidas. Este modelo seria usado para se configurar as ferramentas e gerar canais de comunicação entre elas, quando os mesmos ainda não existem. Acreditamos que o metamodelo de informação RAISE pode ser usado para dar suporte a configuração de ferramentas e estabelecimento de relações entre elas. O nome de metamodelo **RAISE** (*tRaceability-Aided Integrated Software Engineering*) é derivado dessa possibilidade que deve ser estudada em trabalhos futuros.

3.4.2 Granularidade dos elementos rastreáveis

Outra consideração que deve ser feita é sobre a granularidade dos elementos rastreáveis. Bianchi e outros [Bianchi et al., 2000] mostraram que a eficiência do processo de manutenção da Informação de Rastreabilidade está relacionada com a granularidade dos elementos envolvidos.

Um modelo especificado a partir da semântica e sintaxe do Metamodelo RAISE é capaz de manter a granularidade planejada pela organização. Há a meta-classe "*Information*" e a meta-classe "*Thing*". O metamodelo exige que um modelo deve definir quais são as informações rastreáveis e registrá-las como "*Information*" e os artefatos ou ferramentas onde estas informações são mantidas devem ser registradas como "*Thing*".

Exemplo: Se a organização decide rastrear um método de uma classe com um requisito, deve-se modelar o "Método" do tipo "Information", e "Arquivo da Classe" do tipo "*Thing*", associar os dois. Modelar "Requisito" do tipo "*Information*" e o "ERSw" do tipo "*Thing*".

Dessa forma, o problema da granularidade foi transferido para o desenho ("*design*") da informação da Rastreabilidade, que deve ocorrer em toda organização.

3.5 Conclusão do Capítulo

Neste capítulo a abordagem usada para a concepção do Metamodelo RAISE e seu vínculo forte com o metamodelo de desenvolvimento de software e de negócio foram explicados. Os elementos relevantes para o RAISE dos metamodelos base (SPEM e Eriksson e Penker) foram detalhados e, por fim, o Metamodelo RAISE foi explorado de forma incremental em todos os seus detalhes. O Capítulo 4 irá mostrar a avaliação do modelo e sua utilização em sistemas reais.

Capítulo 4

Avaliação, Aspectos Práticos e Operacionais

A avaliação do metamodelo proposto, seu aspecto prático e considerações de uso são apresentados neste capítulo. Uma validação do metamodelo não foi possível de ser feita devido à indisponibilidade de recursos e tempo necessários para medir e acompanhar as possibilidades de uso do RAISE. O uso experimental do Metamodelo RAISE em projetos reais foi executado com o objetivo de medir sua eficácia na captura e uso da informação de Rastreabilidade em um contexto específico.

A experimentação seguiu a técnica de Estudo de Caso [Wohlin et al., 2000], com avaliação através do método comparativo paralelo, combinada com a metodologia proposta por Van Soligen e Berghout [Van Solingen & Berghout, 1999] e teve como foco a capacidade do metamodelo de capturar a informação necessária para a utilização da Rastreabilidade em um determinado contexto de uso.

A próxima seção, Seção 4.1, contextualiza o estudo de caso na organização onde foi executada.

4.1 Contextualização

O Estudo de Caso foi realizado com equipes de quatro projetos diferentes na organização Spread Systems - Unidade MSA-Infor. Os projetos utilizaram processos e métodos diferentes. Seis pessoas durante três meses participaram do Estudo de Caso: dois gerentes de projetos e quatro desenvolvedores.

Como o Estudo de Caso foi em sistemas de clientes da organização, algumas informações dos participantes e dos projetos não podem ser divulgadas por acordos de confidencialidade.

Inicialmente, os diferentes processos de desenvolvimento de software utilizados pela organização foram identificados. As necessidades de Rastreabilidade, requisitos para Rastreabilidade e suas implantações foram localizadas em cada um dos processos. Tarefas que alimentavam e que utilizavam informação de Rastreabilidade foram então observadas. Das tarefas, aquelas que possuem as características relevantes para avaliar o metamodelo sob os aspectos identificados foram então selecionadas.

Um Modelo de Informação de Rastreabilidade foi concebido a partir do Metamodelo RAISE para as ligações de Rastreabilidade relevantes às tarefas selecionadas.

Como a forma de avaliação utilizada foi o Comparativo Paralelo (vide Seção 4.3), a execução do estudo foi planejada em dois grupos de pares de projetos semelhantes, onde, em cada grupo, um projeto utiliza o Metamodelo RAISE, e outro utiliza o modelo matricial. Quatro projetos foram então selecionados para a execução do Estudo de Caso. A escolha dos projetos obedeceu aos seguintes critérios. Dois pares de projetos,

sendo que:

- um projeto deve ser o mais semelhante possível ao seu par.
- um projeto deve ter característica distinta dos projetos do outro par.

Os critérios para decidir as semelhanças e distinções entre projetos foram:

- Processo de desenvolvimento de software e a personalização.
- Tamanho.
- Complexidade do Escopo.
- Arquitetura básica.

Em cada par de projeto, um continuou a utilizar a forma de Rastreabilidade já adotada na organização (matriz de Rastreabilidade - vide Seção 2.3), enquanto o outro utilizou o modelo gerado a partir do RAISE.

Para os dois projetos que utilizaram o RAISE temos que as equipes foram treinadas na utilização dos modelos em suas tarefas.

Em todos os quatro projetos, formulários foram fornecidos para aqueles que estavam executando tarefas que seriam medidas. Nele se apontou dados sobre a utilização dos modelos de Rastreabilidade através de questões descritivas e uma escala Likert [Likert, 1932] para avaliar o grau de confiança das respostas obtidas. O preenchimento dos formulários ocorreu sempre que as tarefas Consumidoras (tarefas que utilizam Informação de Rastreabilidade definida na Seção 4.1.3) foram executadas.

A Seção 4.1.1 apresenta os itens presentes nos formulários.

4.1.1 Formulário para Coleta de Dados da Utilização da Rastreabilidade

1. Projeto
2. Método de Rastreabilidade utilizado: RAISE, Matriz.
3. Executor
4. Para cada análise executada informar:
   a) Tarefa
   b) Quando a análise foi realizada

c) O que se procura?

d) Qual o Grau de Confiabilidade de sua resposta: Nenhuma, Pouca, Média, Muita, Total.

e) A informação foi obtida através de: Rastreabilidade, Outros Artefatos diretos, Afirmações?

f) Qual o Grau de Confiabilidade de sua resposta: Nenhuma, Pouca, Média, Muita, Total.

g) Quantos eram os elementos que deveriam ser encontrados pela pesquisa?

h) Qual o Grau de Confiabilidade de sua resposta: Nenhuma, Pouca, Média, Muita, Total.

i) Número de elementos encontrados pela Rastreabilidade?

j) Qual o Grau de Confiabilidade de sua resposta: Nenhuma, Pouca, Média, Muita, Total.

O item "O que se procura?" foi usado para uma consolidação das pesquisas sobre os elementos de engenharia de software normalmente utilizados nos projetos participantes. A consolidação dessas pesquisas pode ser vista no Apêndice A.

Cada pergunta sobre a tarefa executada é acompanhada de uma avaliação da confiabilidade da resposta, onde se utiliza a escala Likert [Likert, 1932]. Para o Estudo de Caso, apenas as respostas com confiabilidade "Muita" e "Total" foram utilizadas.

4.1.2 Projetos

A organização estava com um portfólio de mais de 20 projetos no momento em que a seleção de projetos ocorreu. Os projetos selecionados foram quatro, aqui referenciados pelo nome fantasia de P1, P2, P3, P4.

Características dos Projetos				
	P1	**P2**	**P3**	**P4**
Processo	PCVPS	PFSW	PManut	PManut
Tamanho	491PF	420PF	1255UMM	854UMM
Escopo	Complexidade Média	Complexidade Média	Complexa	Complexa
Arquitetura	ARCO	ARCO	Multi-arquitetural	Multi-arquitetural

Tabela 4.1. Características dos Projetos

Os objetivos e clientes dos projetos não foram detalhados por razões de confidencialidade, o que não prejudicou a redação do Estudo de Caso. As características relevantes para o Estudo de Caso foram apontadas na Tabela 4.1 e detalhadas nesta seção.

Os projetos P1 e P2 compuseram o Grupo Um. Eles são bastante semelhantes entre si, além de possuírem o mesmo gerente de projetos. Apesar de utilizarem processos diferentes, eles compartilham muitas tarefas. O processo PFSW (Processo de Fábrica de Software) é um subconjunto do PCVPS (Processo de Ciclo de Vida do Produto de Software) com algumas tarefas diferenciadas (vide Seção 4.1.3 para compreensão dos processos utilizados no Estudo de Caso). As tarefas monitoradas foram tarefas semelhantes entre os processos. A arquitetura utilizada nestes dois projetos é um arcabouço desenvolvido pela própria companhia, ARCO - Arquitetura Corporativa, que se baseia em Java.

O Grupo 2, P3 e P4, foi composto de projetos que utilizam o Processo de Manutenção - PMANUT e possuem arquiteturas de alta complexidade. O P3 é um projeto de um sistema multi-plataformas (Mainframe Unisys e Sistemas Linux), envolvendo linguagens COBOL, Algol, Java no ambiente JEE e C++. O P4 é um projeto de um sistema dominado pela arquitetura Java, mas com diversos elementos em C++ e muitas integrações. Apesar dos projetos possuírem tamanhos distintos, as suas complexidades os tornam semelhantes do ponto de vista da Rastreabilidade.

O tamanho dos projetos do Grupo 2 é medido em UMM (Unidade de Medida de Manutenção). UMM é uma medida desenvolvida e utilizada apenas nesta organização e não pode ser usada para se comparar com outras médidas (por exemplo Ponte de Função). Como os dois projetos do grupo 2 utilizam UMM como medida, seus tamanhos são comparáveis.

A formação do UMM é confidencial. Sendo uma medida particular da organização ela precisaria ser validada, o que não ocorreu devido à confidencialidade. O erro que pode ser provocado pela medida foi considerado na avaliação das ameaças às conclusões do Estudo de Caso na Seção 4.7.2.

4.1.3 Processos

A organização possui alguns modelos de processos definidos que são utilizados e personalizados à cada projeto novo. O grau de personalização depende da aderência do processo selecionado ao escopo do projeto. Os projetos selecionados utilizaram os processos denominados pela organização de "PCVPS - Processo de Ciclo de Vida do Projeto

de Software", "PFSW - Processo de Fábrica de Software", e o "PManut - Processo de Manutenção de Produtos".

Todos esses processos possuem uma base comum e compartilham diversas tarefas. Pode-se observar que o PFSW é derivado do PCVPS.

Para a identificação de quais seriam os Rastros capturados e monitorados pelo Estudo de Caso, foram executadas entrevistas com as equipes dos projetos participantes. Ligações de Rastreabilidade já usadas na organização foram identificadas e rotuladas. O Anexo B apresenta as entrevistas realizadas e na Seção 4.2.1 está o estudo de identificação dos Rastros.

As seguintes tarefas foram delimitadas como resultado das entrevistas. Optou-se por restringir o estudo apenas aos Rastros entre código fonte, componentes, requisitos e questões por serem Rastros já capturados na organização. As tarefas identificadas em todos os processos da organização são:

- Identificar e Cadastrar os Requisitos e Casos de Uso.
- Analisar a Questão.
- Especificar Casos de uso e Requisitos.
- Controlar Alterações.
- Verificar Se Os Objetivos Estão Sendo Alcançados.
- Implementar Componentes.
- Implementar a Questão Técnica.
- Acompanhar o Projeto.
- Projetar Caso de Uso.

A especificação de cada uma das tarefas conforme descrito pelos processos da organização está reproduzida no Anexo A.

As tarefas "Identificar e Cadastrar os Requisitos e Casos de Uso", "Especificar Casos de uso e Requisitos", "Analisar a Questão", "Implementar Componentes", "Implementar a Questão Técnica", "Projetar Caso de Uso" foram chamadas de "Tarefas Geradoras" neste estudo. Isso porque elas foram usadas apenas para criar Rastros. Mesmo se na execução dessas tarefas ocorrem consultas a alguns Rastros, como é o caso de "Especificar Casos de Uso e Requisitos", essas consultas não foram usadas no Estudo de Caso.

As tarefas "Acompanhar o Projeto", "Controlar Alterações" e "Verificar Se Os objetivos Estão Sendo Alcançados" são tarefas onde os Rastros são consultados. Foram chamadas de "Tarefas Consumidoras" neste estudo.

O Modelo de Informação da Rastreabilidade da organização, gerado do Metamodelo RAISE, foi parcialmente construído com base nas Tarefas Geradoras e Consumidoras selecionadas.

As Tarefas Consumidoras foram utilizadas para medir a eficiência do modelo gerado.

4.2 Aspectos Práticos e Operacionais

4.2.1 Rastros encontrados para a Organização

Baseando-se nas tarefas selecionadas da organização e na classificação de Rastros documentada na literatura (Seção 2.2), os seguintes tipos de Rastros foram localizados:

- Ligação de Alocação de Requisito
- Ligação de Satisfação de Requisito
- Ligação de Alocação de Questão
- Ligação de Resolução de Questão

A seguir, os modelos de informação de Rastreabilidade usados no estudo de caso são mostrados. Os modelos são o matricial, chamado de CONRAS- Controle da Rastreabilidade, já usado na organização, e o Modelo RAISE, assim chamado por ter sido gerado a partir do Metamodelo RAISE.

4.2.1.1 Alocação do Requisito

Essa ligação vincula o requisito ao componente que deve satisfazer o requisito. Derivado tipo "Alocação" do modelo de Toranzo e outros [Toranzo et al., 2002], mostrado na Seção 2.2, e também no tipo proposto por Letelier [Letelier, 2002] e usado no metamodelo de Espinoza e outros [Espinoza et al., 2006], descrito na Seção 2.5.

A Figura 4.1 mostra o modelo da Ligação de Alocação de Requisito no Modelo RAISE.

Nesse caso, a Ligação de Alocação de Requisito é orientada ("*drivenBy*") pela regra de negócio "Requisito Alocado". Essa regra vincula requisitos a componentes. Como é uma regra estrutural, ela foi especificada em UML e é vista na Figura 4.2.

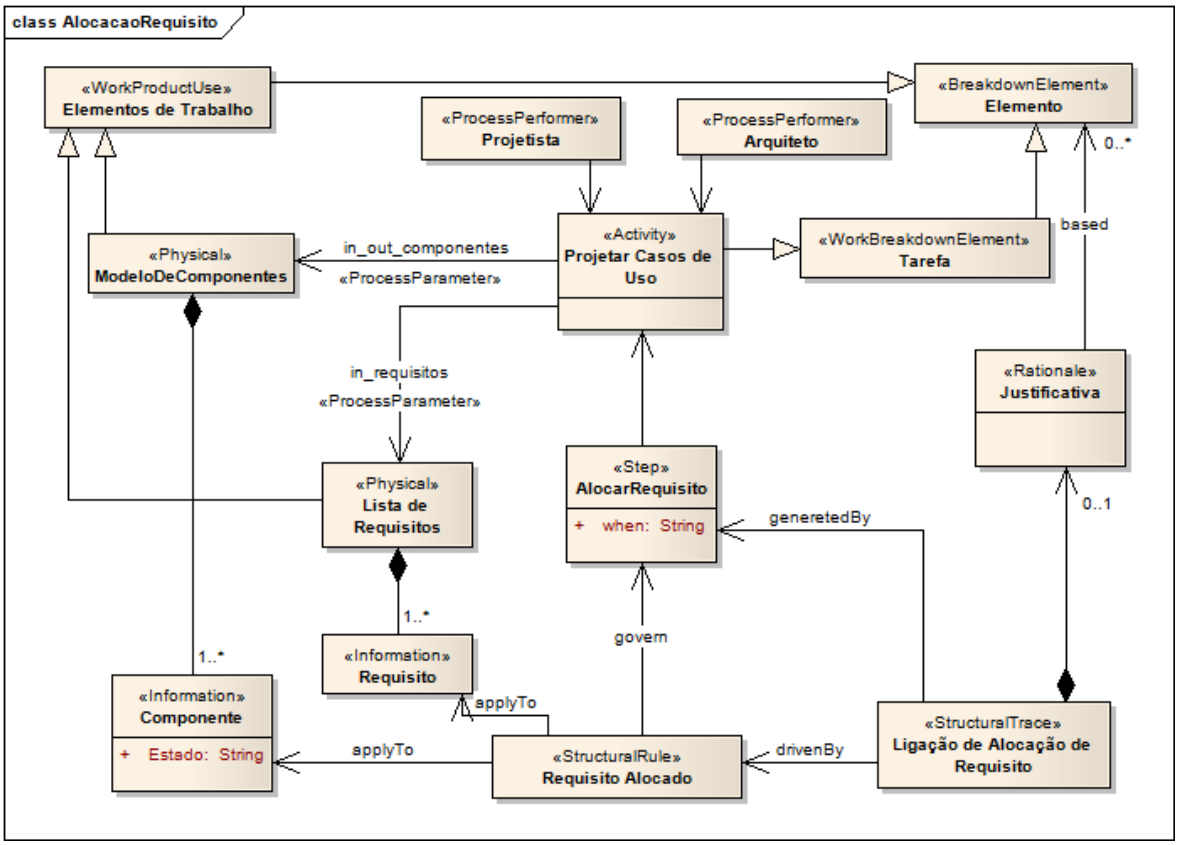


Figura 4.1. Modelo RAISE do Rastro Ligação de Alocação de Requisitos

A regra indica que um requisito pode ser alocado a diversos componentes. E cada componente pode satisfazer diversos requisitos.

A Figura 4.1 ainda indica que o Rastro Ligação de Alocação de Requisito é gerado pela etapa "Alocar Requisito" da tarefa "Projetar Caso de Uso". Essa etapa é descrita na Seção A.9 do Anexo A como uma indicação que determinados requisitos devem ser satisfeitos em códigos do componente projetado.

O Projetista é o executor da tarefa ("*ProcessPerformer*"). De todos os elementos de entrada da tarefa, apenas os elementos "Componente" e "Requisito" foram modelados por serem os únicos relevantes para o Rastro modelado. Eles são do tipo "*Information*" por serem elementos de informação contidos nos artefatos "Modelo de componentes" e "Lista de Requisitos". Esse elementos estão referenciados na tarefa através dos parâmetros de processo ("*processParameter*") "inProjetar componentes" e "inProjetar Requisitos".

Deve-se também observar que o Rastro Ligação de Alocação de Requisito pode ter uma justificativa de sua existência registrada, onde se aponta elementos que evidenciam essa justificativa. O elemento "Justificativa" não é de preenchimento obrigatório.

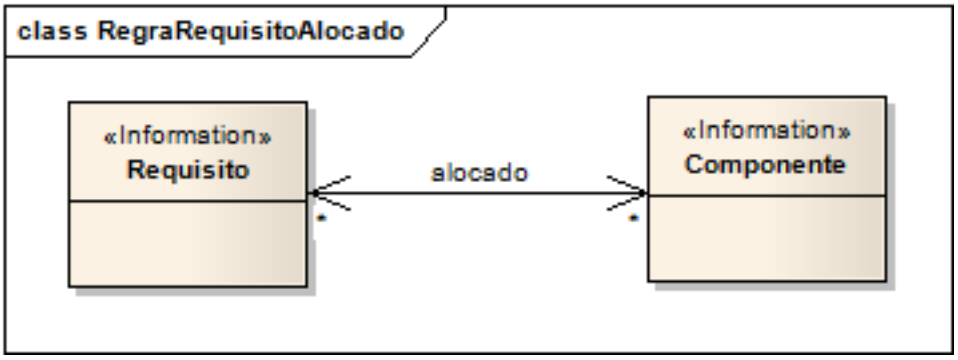


Figura 4.2. Especificação da Regra Requisito Alocado

Apenas para quando o Projetista achar que deve explicar algo além do que o modelo já mostra.

A Tabela 4.2 mostra o mesmo Rastro implementado no modelo matricial usado na organização: CONRAS - Controle de Rastreabilidade da ligação de alocação.

CONRAS - Requisito por Componente				
Requisitos	**Componentes**			
	Comp001	**Comp002**	**Comp003**	**Comp004**
ReqCli001		X		
ReqCli002			X	
ReqCli003	X			
ReqCli004				

Tabela 4.2. CONRAS - Requisitos x Componentes

A matriz possui os componentes sendo representados nas colunas e os requisitos nas linhas. Onde ocorrer a alocação do requisito deve-se marcar com um X a interseção da linha do requisito com a coluna do componente.

4.2.1.2 Satisfação do Requisito

Essa ligação vincula o requisito com o código que o satisfaz.

A Figura 4.3 mostra a modelagem do Rastro Satisfação de Requisito no Modelo RAISE. O tipo do Rastro foi concebido com base nos tipos "Satisfação" propostos por Ramesh e Jarke [Ramesh & Jarke, 2001], Toranzo e outros [Toranzo et al., 2002], Spanoudakis e Zisman [Spanoudakis & Zisman, 2005], e no tipo "Inferido" de Aizenbud-Reshef e outros [Aizenbud-Reshef et al., 2006]. Todos esses tipos foram descritos na Seção 2.2. A solução é muito parecida com o Rastro Alocação de Requisitos, por ser quase que a mesma informação que foi rastreada. A diferença está no fato que a alocação do requisito ocorre como planejamento dos componentes, enquanto a satisfação

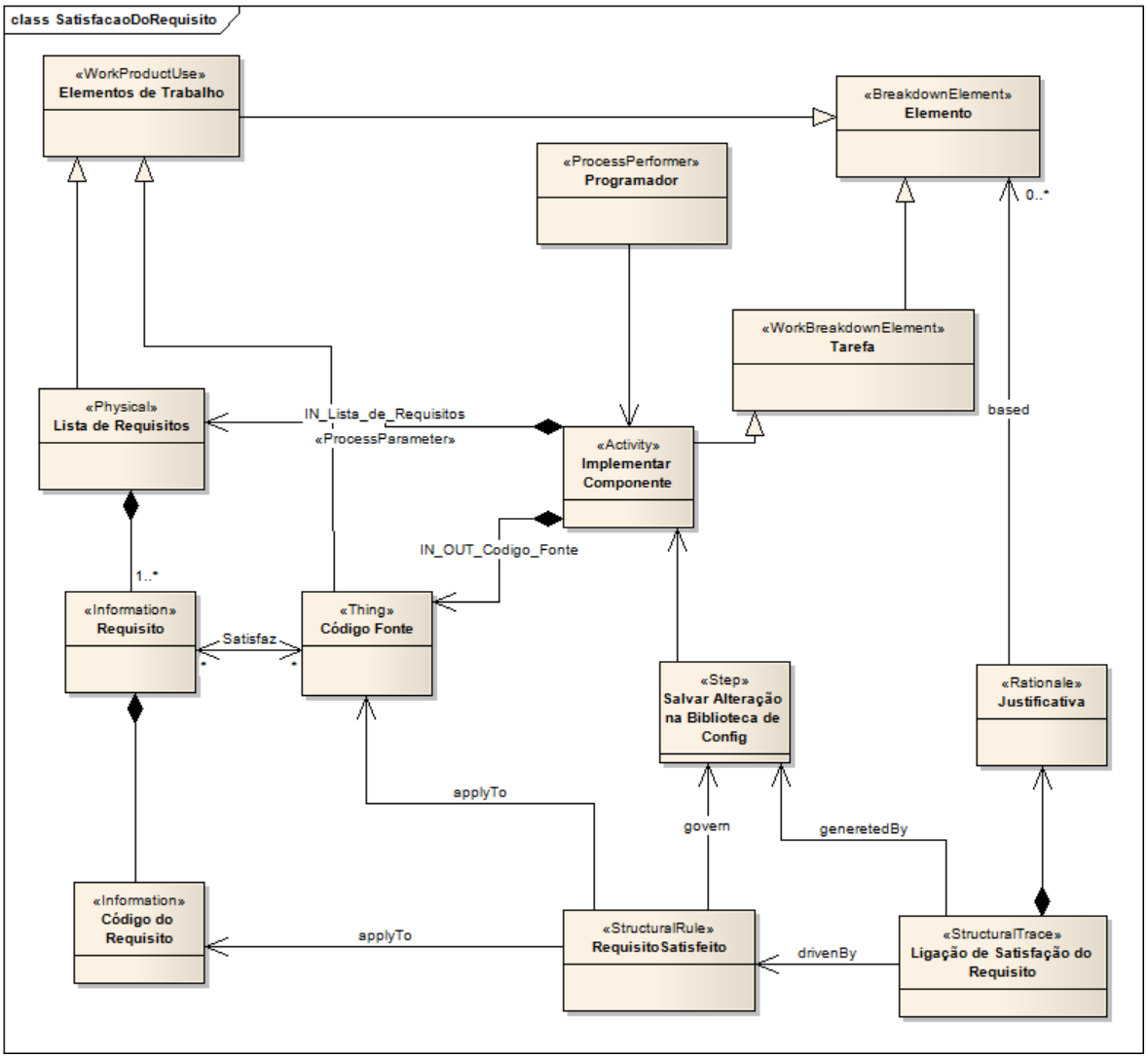


Figura 4.3. Modelo RAISE do Rastro de Satisfação do Requisito

dos requisitos ocorre quando o código é construído e o requisito realizado no código. Uma diferença está na regra de negócio. Nesse caso, a regra é "Requisito Satisfeito" do passo "Salvar Alteração na biblioteca de Configuração". É uma regra Estrutural e sua especificação pode ser vista na Figura 4.4. A figura também traz a regra de alocação para permitir visualizar a relação entre as duas regras estruturais.

Ainda na Figura 4.3 pode-se observar que a tarefa que gera o Rastro é "Implementar Componente" descrito no Anexo A, Seção A.6. O executor principal da tarefa é o Programador. Dos diversos artefatos de entrada da tarefa, a "Lista de Requisitos" e "Código Fonte" foram modelados pelo fato de serem usados pela regra de negócio RequisitoSatisfeito.

A Tabela 4.3 mostra o mesmo Rastro implementado no modelo matricial usado

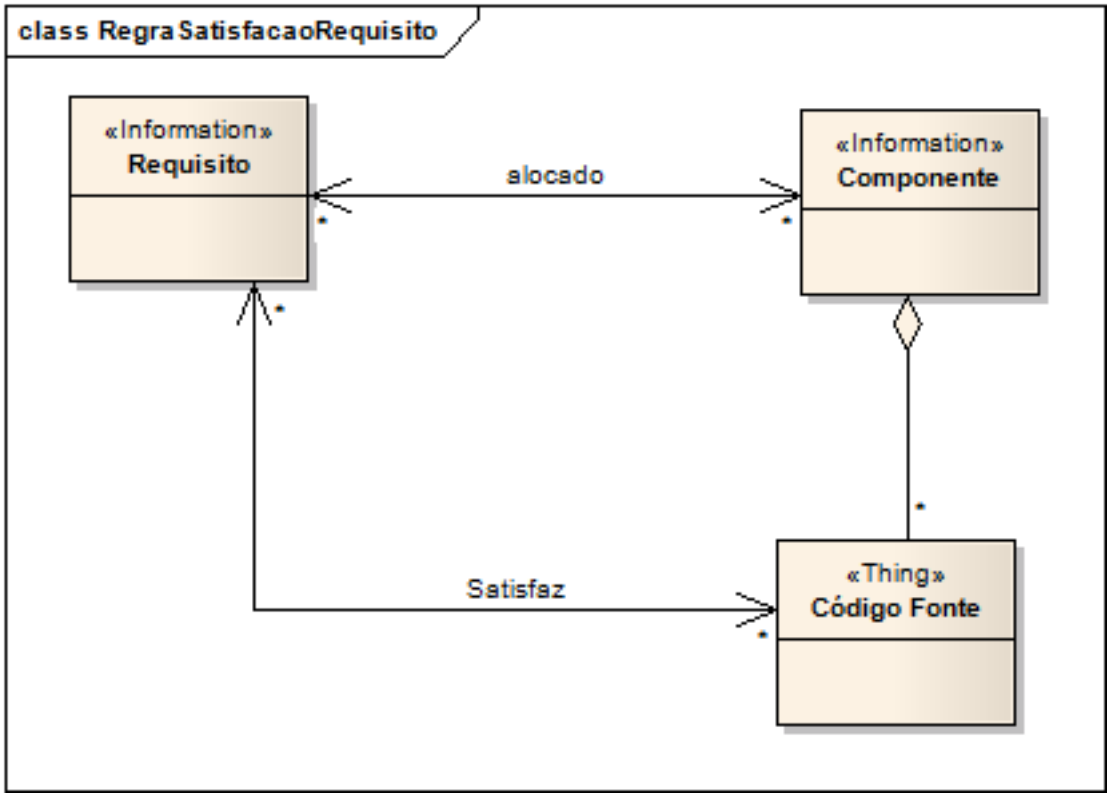


Figura 4.4. Regra de Negócio Satisfação do Requisito

na organização, CONRAS - Controle de Rastreabilidade.

CONRAS - Requisito por Código				
Requisitos	**Componentes**			
	Class0001	**Class0002**	**Class0003**	**Class0004**
ReqCli001		X	X	
ReqCli002			X	
ReqCli003				X
ReqCli004		X		

Tabela 4.3. CONRAS - Requisitos x Código

4.2.1.3 Alocação da Questão

O Modelo RAISE do Rastro Ligação de Alocação de Questão pode ser encontrado na Figura 4.5. É um Rastro bastante semelhante ao de Ligação de Alocação de Requisito e foi baseado nos mesmos tipos propostos na seção 2.2. Entretanto, nesse caso, o elemento alocado é a questão.

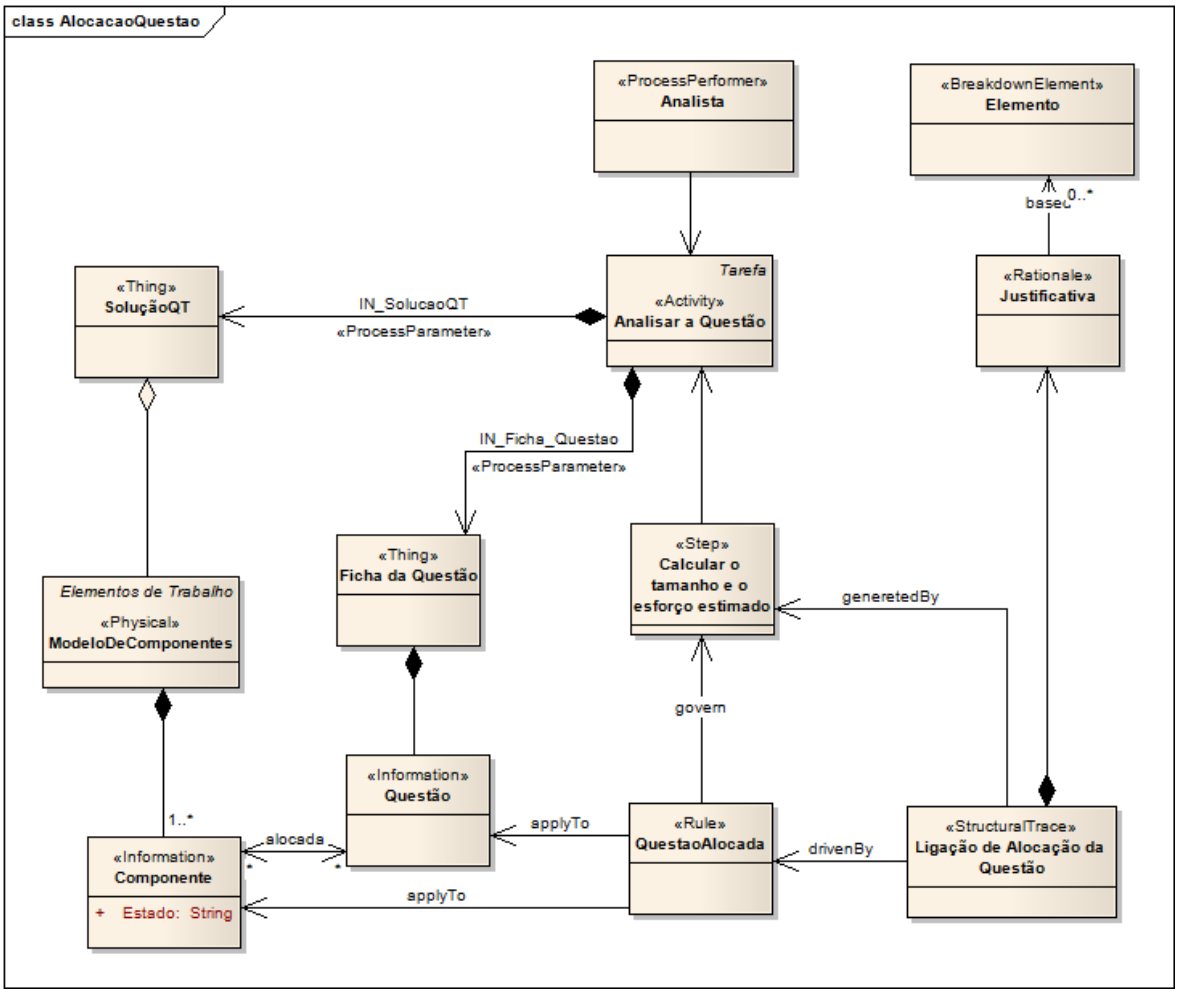


Figura 4.5. Modelo RAISE mostrando os detalhes do Rastro "Ligação de Alocação de Questão"

Uma questão é do tipo "*Information*" e está localizada em uma Ficha de Questão ("*Thing*"), que é um elemento de entrada para a tarefa "Analisar a Questão" descrita no Anexo A, Seção A.2. A etapa "Calcular o Tamanho e o Esforço Estimado", dentre outras ações, registra o componente que deve ser alterado para resolver a questão. Essa ação é governada pela regra de negócio "Questão Alocada". A regra está modelada na Figura 4.6 e mostra que um componente pode alocar diferentes Questões, e cada Questão pode ser alocada em diferentes componentes. A classe "Elemento" no modelo é a generalização dos elementos de trabalho, como a SoluçãoQT e a "Ficha da Questão". Os relacionamentos de generalização da classe Elemento não estão sendo mostrados nesse modelo (para visualizar a generalização pode-se consultar o modelo da Figura 4.3).

No modelo matricial esse Rastro é registrado conforme Tabela 4.4.

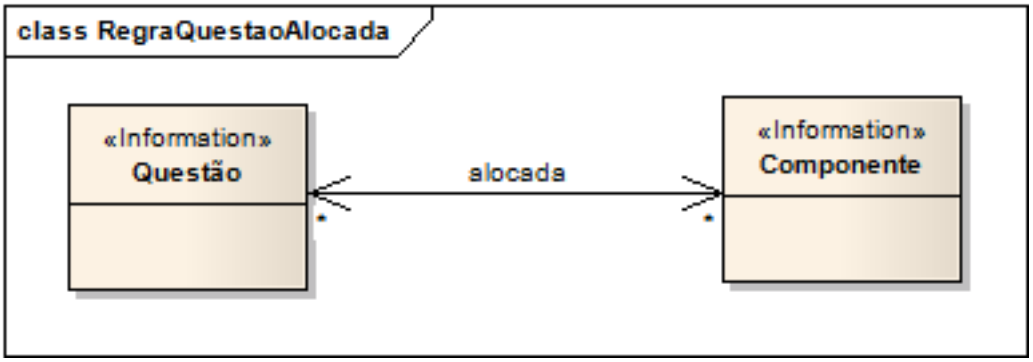


Figura 4.6. Regra de Negócio Questão Alocada

CONRAS - Questão por Componente				
Questão	**Componentes**			
	Comp0001	**Comp0002**	**Comp0003**	**Comp0004**
Ques001				X
Ques002		X		
Ques003			X	
Ques004	X			

Tabela 4.4. CONRAS - Questão x Componente

4.2.1.4 Resolução da Questão

A Figura 4.7 mostra a Ligação de Resolução da Questão. O Rastro gerado da tarefa "Implementar Questão Técnica" descrita no Anexo A, Seção A.7. A etapa "Liberar Código" pede para o "Programador" registrar os códigos alterados pela resolução da questão. O registro é regido pela regra de negócio "Questão Resolvida" modelada na Figura 4.8. A regra indica que uma Questão pode ser resolvida por zero ou mais códigos e cada código pode resolver de zero a muitas questões. A figura ainda mostra a ligação de alocação para ajudar o entendimento da relação de uma ligação com a outra.

Esse mesmo Rastro representado no modelo matricial apresenta o formato da Tabela 4.5.

CONRAS - Questão por Código				
Questão	**Códigos**			
	Class0001	**Class0002**	**Class0003**	**Class0004**
Ques001		X		
Ques002				X
Ques003				X
Ques004	X			

Tabela 4.5. CONRAS - Questão x Código

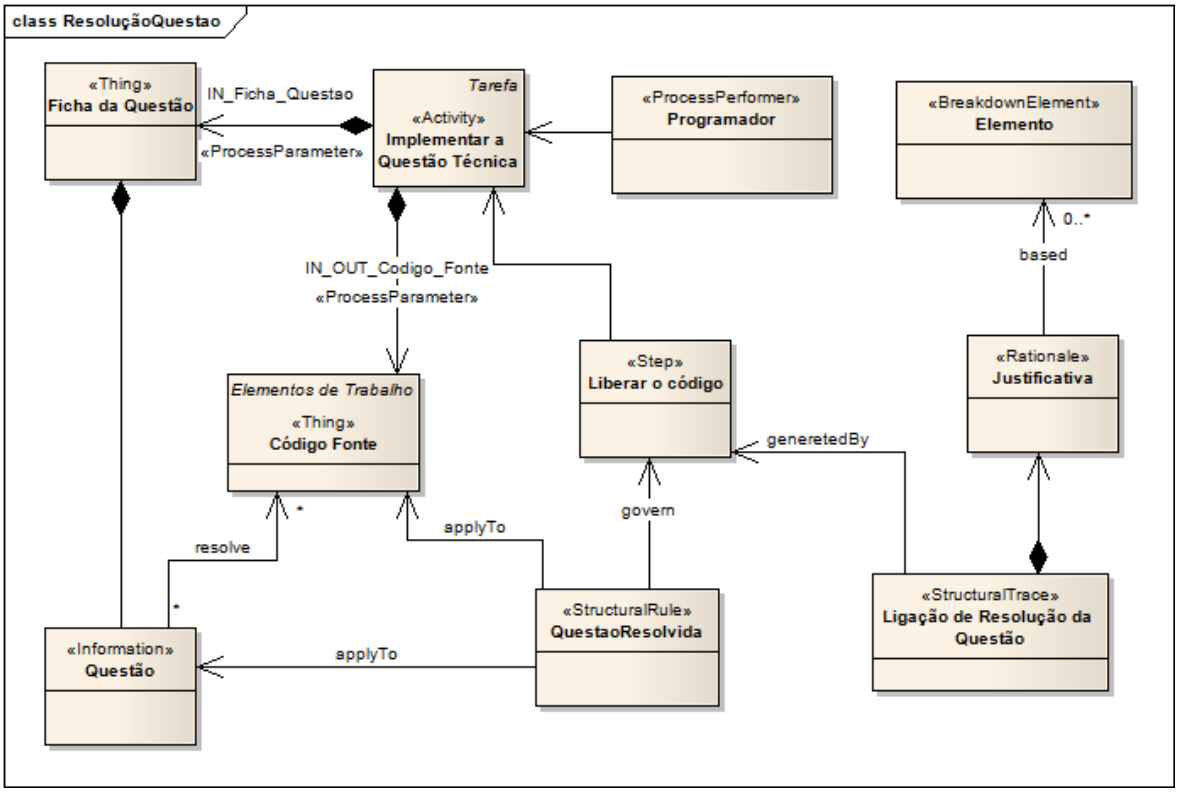


Figura 4.7. Modelo RAISE da Ligação da Resolução da Questão

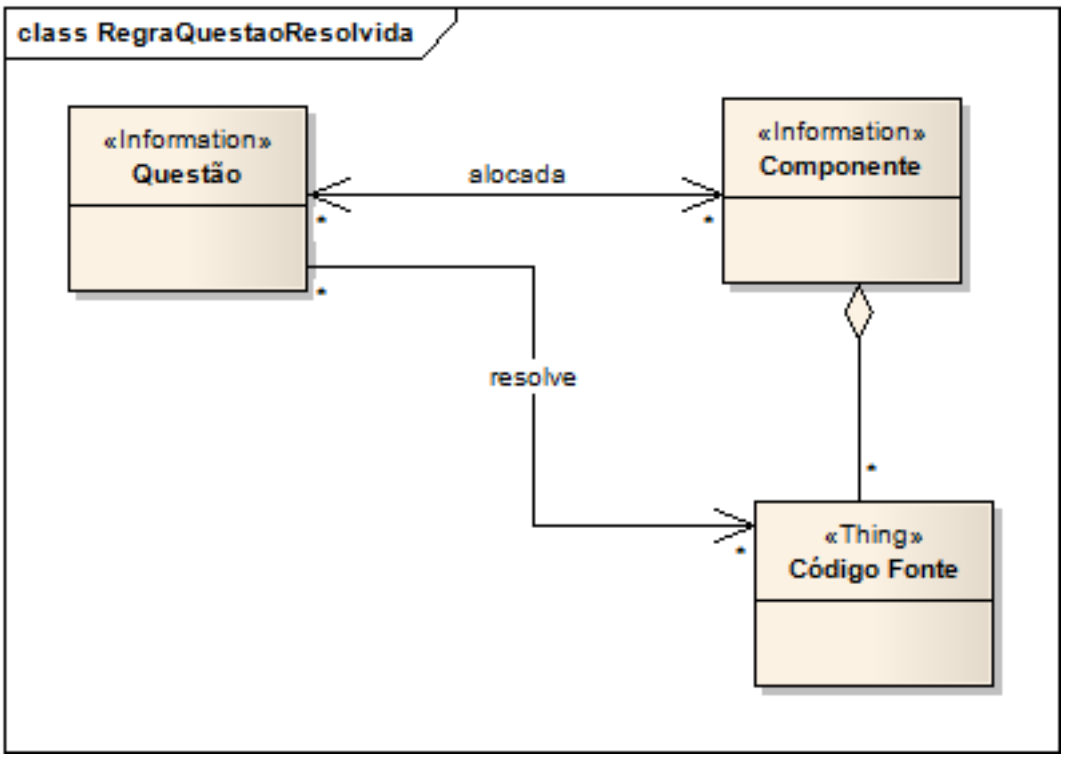


Figura 4.8. Modelo da Regra de Negócio "QuestãoResolvida"

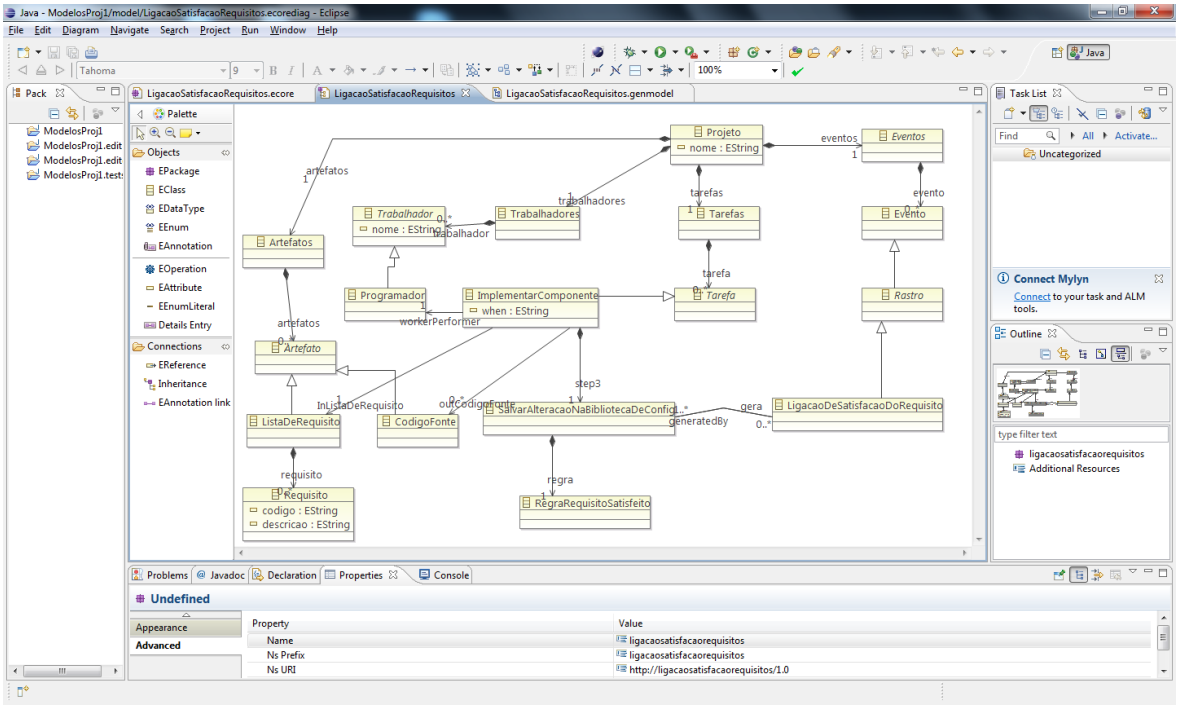


Figura 4.9. Implementação da Ligação de Satisfação do Requisito no EMF

4.2.2 Implementação do Modelo RAISE

Os Rastros mostrados na Seção 4.2.1 em seus respectivos modelos foram implementados em uma ferramenta de desenvolvimento para permitir seu registro. No caso da Matriz de Rastreabilidade, a implementação já é a usada pela organização. Uma planilha Microsoft Excel de nome CONRAS - Controle de Rastreabilidade permite o registro de cada uma das ligações mostradas.

No caso do Modelo RAISE, a implementação foi realizada no Eclipse EMF. A Figura 4.9 mostra essa implementação. Utilizou-se o IDE Eclipse Versão Helios, Build: 20100617-1415, com plug-ins das ferramentas de modelagem do projeto Eclipse ("*Eclipse Modeling Tools*")

O EMF ("*Eclipse Modeling Framework*") é um arcabouço para modelagem e geração de código com base no modelo para a construção de ferramentas e outras aplicações baseadas em um modelo de dados estruturado [Eclipse, 2011a]. A partir de uma especificação de um metamodelo em XMI, o EMF provê ferramentas e suporte em tempo de execução para produzir um conjunto de classes na linguagem Java para o modelo. Para mais informação, pode-se consultar o livro de Steinberg e outros [Steinberg et al., 2009].

Para o desenho do metamodelo em EMF usou-se o GMF (Graphical Modeling Framework) que fornece uma interface gráfica para o desenho do metamodelo. Consulte

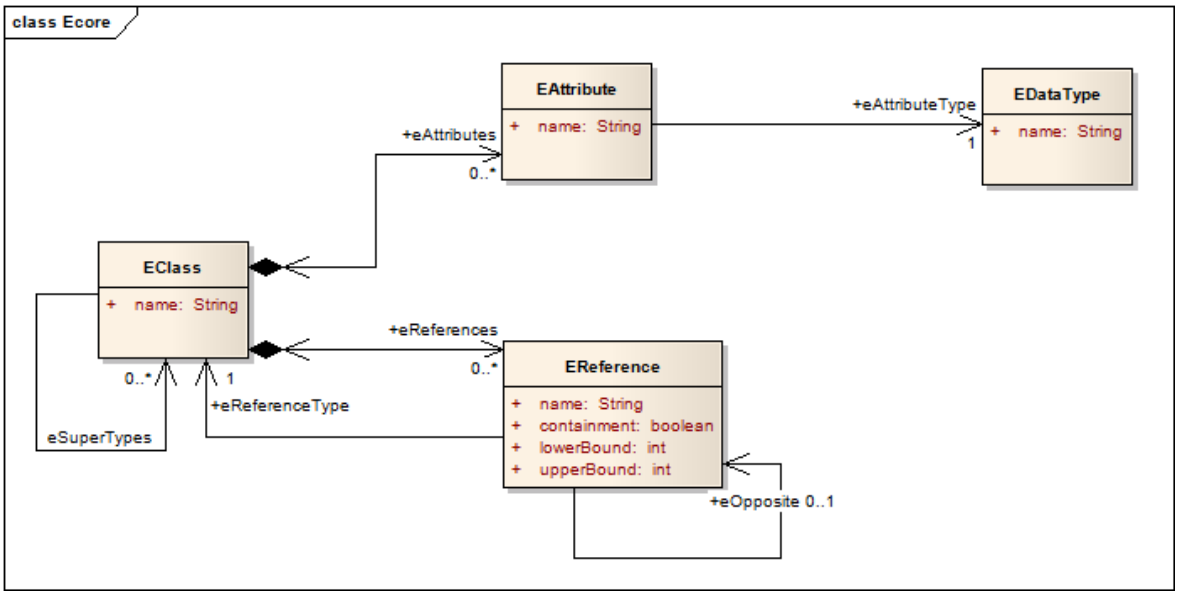


Figura 4.10. Núcleo do meta-metamodelo Ecore

o site do projeto GMF [Eclipse, 2011b] para mais informação .

A especificação do Modelo RAISE no EMF utilizou o metamodelo Ecore. O Ecore é uma linguagem para modelagem estática inspirada na UML e é parte integrante do arcabouço EMF.

O núcleo do Ecore é mostrado na Figura 4.10.

Esse modelo possui quatro meta-classes. A primeira o EClass modela as classes. As classes são identificadas pelo nome e podem conter atributos e referências. Para suportar heranças, uma classe pode referenciar a um número de outras classe como super-tipos.

Um EAttribute modela atributos das classe. Eles são identificados pelo nome e possuem um tipo.

O EDataType modela os tipos de atributos, representando tipos primitivos. Os tipos de dados são também identificados pelo nome.

Por fim, o EReference é usado na modelagem de associações entre classes. Como os atributos, as referências são identificadas pelo nome e possuem um tipo. Entretanto, esse tipo deve ser EClass da outra ponta da associação.

A implementação do Modelo RAISE em Ecore no EMF (Figura 4.11) exigiu que elementos que não são parte do Metamodelo RAISE fossem tratados. O arcabouço EMF pede que todas as metas-classe tenham uma ligação direta ou indireta do tipo de "*containment*" com uma classe raiz. É o caso do elemento "Projeto". Ele é o elemento raiz que contem todos os elementos que são usados pelo modelo. Há, portanto, a lista

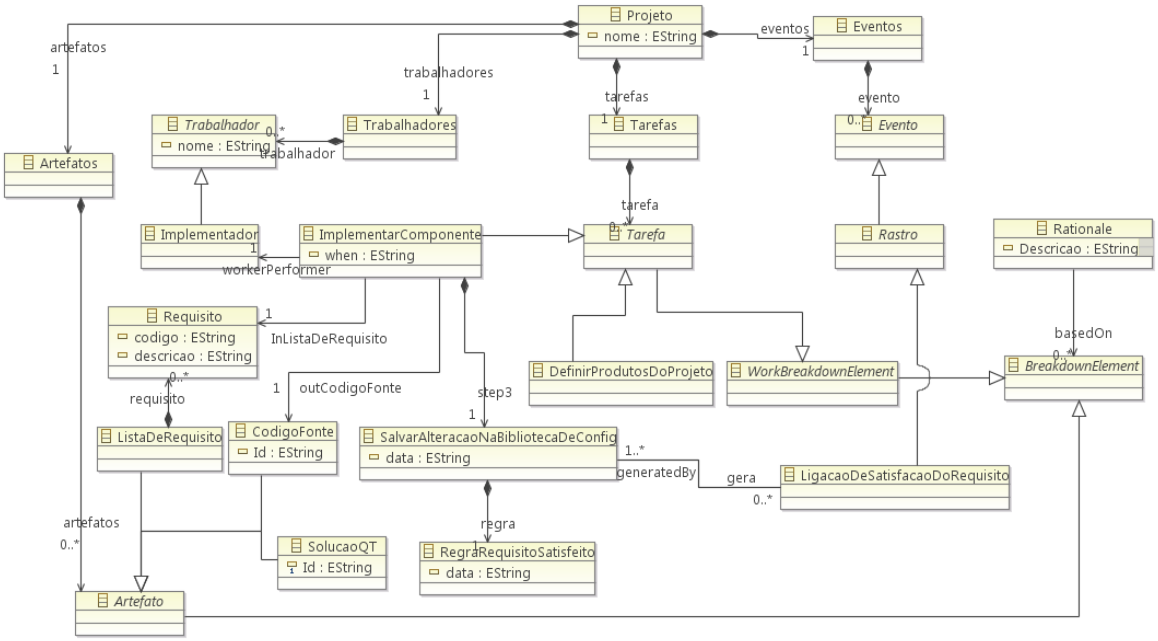


Figura 4.11. Modelo RAISE para a Organização do Estudo de Caso

de "Trabalhadores" no projeto em seus respectivos papéis, a lista de "Tarefas" usadas no projeto. No caso foram modeladas apenas as Tarefas que fizeram parte do Estudo de Caso. A Lista de "Artefatos" usados no Estudo de Caso. E por fim a lista de "Eventos" que contem todos os "Rastros" tratados no Estudo de Caso.

O Modelo RAISE pode ser visto na Figura 4.11.

4.3 Metodologia do Estudo de Caso

Como dito no inicio do capítulo, a avaliação do Metamodelo RAISE foi executada utilizando a técnica de Estudo de Caso conforme explicado por Wohlin e outros [Wohlin et al., 2000]. O Estudo de Caso é um acompanhamento de projetos sem a interferência sistemática do pesquisador. O controle sobre o ambiente é menor que no uso de experimentos.

Wohlin e outros propõem três meios de avaliação em um Estudo de Caso:

- Estudo através de Linha de Base: compara os resultados obtidos contrapondo-os aos resultados de uma linha de base anterior;

- Estudo Comparativo Paralelo: compara os resultados de duas execuções paralelas, cujas características sejam semelhantes (Do Inglês: "*paired comparison design*");

- Estudo randômico: aplica o método para algumas execuções individuais e randomicamente não aplica para outras.

O Metamodelo RAISE foi avaliado através do Estudo Comparativo Paralelo.

A técnica de Estudo de Caso não foi utilizada integralmente na forma sugerida por Wohlin. A técnica foi combinada com a metodologia proposta por Van Soligen e Berghout [Van Solingen & Berghout, 1999]. Essa metodologia se divide nos subprocessos de Planejamento, Definição, Coleta de dados e Interpretação.

Para estruturar o subprocesso de Definição, foi utilizado o modelo GQM - Goal Question Metric [Basili et al., 1994] como um guia. Esse modelo pede que indicadores quantitativos sejam identificados a partir de metas. As metas compõem a definição conceitual do modelo. Delas se derivam os indicadores através das questões. As questões representam o aspecto operacional do modelo . O GQM não foi usado com todo o seu formalismo, por se tratar de um Estudo de Caso. O modelo GQM apenas serviu de orientador na busca de indicadores relevantes para a avaliação do Metamodelo RAISE.

O principal objetivo do Subprocesso de Definição foi indicar a direção da realização do Estudo de Caso. Já o Subprocesso de Planejamento determinou o escopo, prazos e restrições do experimento.

No Subprocesso de Coleta de Dados se executou a coleta dos dados identificados na Definição através dos métodos definidos no planejamento e usados na Interpretação. Ocorreu em paralelo com o Subprocesso de Interpretação.

Os dados coletados foram analisados no Subprocesso de Interpretação e uma conclusão do Estudo de Caso foi atingida.

Os subprocessos desse modelo possuem saídas inter-relacionadas: As metas, questões e indicadores, gerados na Definição foram respondidos no subprocesso Interpretação. Os dados obtidos da Coleta de Dados foram direcionados pelos indicadores encontrados no subprocesso Definição. Através desses indicadores, as questões foram respondidas no subprocesso Interpretação e finalmente se verificou a satisfação da meta do Estudo de Caso.

As seções seguintes apresentam as discussões e resultados de cada um dos subprocessos aplicados ao Metamodelo RAISE.

4.4 Subprocesso de Definição

Neste subprocesso se estabeleceu a meta que orientou o Estudo de Caso.

A intenção do Estudo de Caso foi de avaliar a capacidade do Metamodelo RAISE em definir modelos que capturam a informação necessária para a utilização da Rastre-

abilidade em um determinado contexto de uso. Como não era possível fazer um estudo sistemático de possíveis modelos a serem gerados e compreender suas limitações, a meta estabelecida foi direcionada a um determinado contexto.

4.4.1 Meta Específica

A meta específica do Estudo de Caso foi obtida a partir dos aspectos identificados por Pohl [Pohl, 1996] e discutidos no Capítulo 2.

Pohl afirmou que o uso da informação da Rastreabilidade depende da Parte Interessada e que apenas informação orientada ao contexto é base para uso apropriado. Como o Metamodelo RAISE se propõe a permitir o registro estruturado do contexto do Rastro, a meta específica do Estudo de Caso foi:

- Avaliar se a informação capturada por um modelo definido a partir do Metamodelo RAISE satisfaz mais que o modelo matricial as necessidades dos agentes que executam tarefas de verificação para se tomar alguma decisão.

O Escopo do Estudo de Caso foi bastante restritivo conforme já discutido anteriormente por não haver nem tempo hábil nem recursos suficiente para uma pesquisa mais ampla. A tarefa de verificação foi escolhida porque alguns artigos bastante referenciados apontam essa tarefa como uma que mais utiliza informação de Rastreabilidade [Cleland-huang et al., 2009; Ramesh & Jarke, 2001; Oliveto et al., 2007]. Isso ocorre devido à necessidade de se verificar Rastros para a tomada de decisões. Como exemplo, temos: análise de impacto de alguma alteração, verificação de atendimento de requisitos para tomar ações corretivas nos projetos.

4.4.2 Questões

Questões foram formuladas para conduzir o processo de avaliação do metamodelo em função da meta do Estudo de Caso identificada.

Para se concluir que o modelo gerado a partir do Metamodelo RAISE satisfez mais as necessidades das Partes Interessadas que o modelo matricial no contexto do Estudo de Caso, deve-se saber qual informação de Rastreabilidade foi necessária para as Partes Interessadas (dentro do escopo do Estudo de Caso).

Ramesh e Jarke [Ramesh & Jarke, 2001] fizeram essa pergunta a diferentes organizações e montaram modelos baseados nas respostas obtidas (vide Seção 2.4). Nesta dissertação foi mostrado que essa informação depende da cultura da organização e, portanto, o modelo de Rastreabilidade depende do contexto onde está inserido. Mas

dentro de um determinado contexto se faz necessário saber qual informação de Rastreabilidade é relevante.

Além de se conhecer os Rastros necessários, também foi importante conhecer se os modelos estavam atendendo às necessidades de Informação de Rastreabilidade. Esse dado foi usado como base comparativa entre o modelo matricial e o modelo baseado no RAISE.

Dessa forma as questões identificadas foram:

- Qual informação foi pesquisada nos modelos de Informação de Rastreabilidade pelas Partes Interessadas?
- Qual informação foi obtida dos modelos de Informação de Rastreabilidade pelas Partes Interessadas?

A resposta a essas duas questões servem como base para a avaliação se o modelo obtido a partir do RAISE satisfaz as necessidades das Partes Interessadas mais do que o modelo matricial.

4.4.3 Indicadores

Indicadores que podem ajudar na avaliação do Metamodelo RAISE foram identificados com base nas questões formuladas. Esses indicadores foram usados apenas como apoio durante a avaliação de metamodelo e não representam métricas que permitam uma validação do Metamodelo RAISE.

Durante uma avaliação de alguma situação onde é necessário tomar decisões, Partes Interessadas buscam através de mecanismos de Rastreabilidade a informação necessária à tomada de decisão.

Para cada pesquisa realizada se observou os elementos necessários ("o que foi pesquisado") para que a pesquisa fosse um sucesso e os elementos efetivamente encontrados.

Assim, os elementos pesquisados ("o que foi pesquisado"), número de elementos pesquisados e número de elementos encontrados são dados coletados para serem usados na avaliação do Metamodelo RAISE.

Elementos pesquisados são elementos que compõem os artefatos do processo de desenvolvimento de software utilizado pela organização e que são pesquisados através dos Rastros. Por exemplo, um caso de uso, um requisito, um código fonte. Normalmente um elemento pesquisado é representado por uma pergunta, para a qual, a resposta é uma coleção de elementos pesquisados. Para satisfazer uma necessidade do

usuário é necessário pesquisar alguns elementos diferentes. Por exemplo, durante a tarefa de acompanhamento, o Gerente de Projeto pesquisa os requisitos alocados para uma determinada entrega, os requisitos em implementação e os requisitos já implementados e as razões de decisões tomadas sobre o escopo da entrega. A composição desta informação ajuda ao Gerente de Projeto a entender o atual estado do desenvolvimento do produto e comparar com o planejado.

Número de elementos pesquisados é o número de elementos que deveria ser encontrado na pesquisa. Por Exemplo, o Gerente de Projetos pesquisa os requisitos que estão alocados para uma entrega. Sendo que a entrega possui 32 requisitos alocados, esse é o número total de elementos que a pesquisa deve retornar.

Número de elementos encontrados é o número real de elementos que foi encontrado na pesquisa. Ele pode ser menor que o número de elementos pesquisados porque o modelo pode não ser capaz de registrar todos os Rastros que levam aos elementos. Por exemplo, o modelo baseado em matriz utilizado na organização em estudo registra os requisitos alocados para os códigos fontes, mas não registra se ocorreu alteração no escopo e o porquê dessa alteração.

O número de elementos pesquisados é difícil de se precisar em alguns casos, mas é possível identificar que o modelo não foi capaz de registrar todas as situações necessárias para a captura de alguma informação de Rastreabilidade.

4.5 Subprocesso de Planejamento

O subprocesso Planejamento determinou o escopo, prazos e restrições do Estudo de Caso. O escopo foi delimitado nos projetos selecionados e discutidos na Seção 4.1 deste capítulo.

O prazo foi delimitado em 3 meses para a execução, coleta de dados e avaliação, ocorridos de dezembro de 2010 até fevereiro de 2011.

O prazo é a principal restrição do Estudo de Caso. E para que o prazo fosse cumprido, foi necessário restringir o escopo (como visto na Seção 4.1).

Para a análise quantitativa, se trabalhou com duas classes de variáveis: dependente e independente [Wohlin et al., 2000]. A variável independente foi escolhida com base nos objetivos definidos na Seção 4.4. Há uma variável independente no Estudo de Caso: o tipo do modelo usado. Essa variável é preenchida com o Modelo RAISE ou o modelo matricial já utilizado na organização. As variáveis dependentes são os indicadores definidos na Seção 4.4.

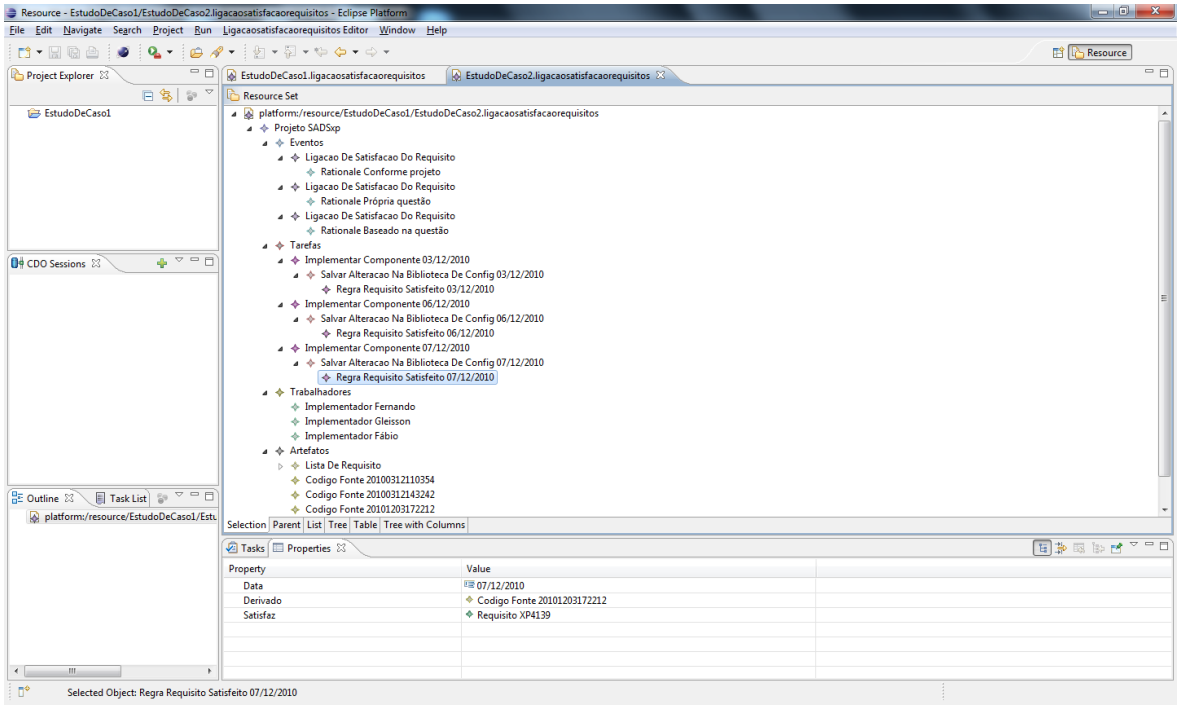


Figura 4.12. Registros do Modelo RAISE da organização

Nesse Estudo de Caso, a analise quantitativa foi usada apenas como orientador da analise qualitativa.

4.6 Subprocesso Coleta dos Dados

Informação de Rastreabilidade foi registrada nos modelos durante a execução das tarefas geradoras. Para os projetos que utilizaram a técnica matricial, o registro era feito na planilha CONRAS (vide Seção 4.2). Já para o caso do uso do Modelo RAISE, o executor utilizou a implantação do Modelo RAISE no EMF para registrar os Rastros. A Figura 4.12 mostra uma lista de registros na ferramenta.

Para as Tarefas Consumidoras, o executor utilizou o formulário eletrônico com os itens mostrados na Seção 4.1.1 para anotar informação logo em que eram realizadas. Os dados foram consolidados para serem analisados durante o subprocesso de Interpretação (Seção 4.7).

Os dados são então utilizados nas tarefas de pesquisas indicadas acima. Nessas tarefas foram anotadas as pesquisas que seriam necessárias e o quanto delas o modelo de Rastreabilidade forneceu respostas.

Para garantir a semelhança do uso da Rastreabilidade nos projetos com modelos diferentes, as pesquisas executadas para cada tarefa foram identificadas junto aos

gerentes de projetos. Sua consolidação está apontada no Apêndice A. Formas alternativas de obter o resultado correto foram identificadas para avaliar o resultado dos modelos e assim obter o número de elementos pesquisados (vide Seção 4.4.2).

Além disso, foi coletada informação sobre os participantes do Estudo de Caso a partir do formulário contendo os itens mostrados abaixo:

1. Nome
2. Projeto
3. Tempo no Projeto
4. Cargo
5. Funções (papéis) no projeto
6. Quais atividades das listadas abaixo você realiza?
    - Identificar e Cadastrar os Requisitos e Casos de Uso.
    - Analisar a Questão.
    - Especificar Casos de uso e Requisitos.
    - Controlar Alterações.
    - Verificar Se Os objetivos Estão Sendo Alcançados.
    - Implementar Componentes
    - Implementar a Questão Técnica.
7. Para cada tarefa que você realiza identifique se:
    a) Você foi treinado para realizá-la?
    b) Você foi treinado nas ferramentas utilizadas na execução da tarefa?
    c) Você se classifica como na execução da tarefa: Junior, Pleno, Sênior?
8. você conhece a técnica de Rastreabilidade utilizada em seu projeto?
9. Você foi treinado na técnica de Rastreabilidade utilizada em seu projeto?

O formulário preenchido pelos participantes está no Anexo C.

4.7 Subprocesso de Interpretação

A interpretação foi executada após o subprocesso de coleta de dados. Em seguida, apresentamos o resultado da análise dos dados.

4.7.1 Resultados dos Indicadores

Os dados coletados e consolidados apresentaram os resultados mostrados na Tabela 4.6.

Indicadores				
	Grupo 1		**Grupo 2**	
Indicador	**P1 (RAISE)**	**P2**	**P3 (RAISE)**	**P4**
Pesquisas realizadas	32	33	44	34
Elementos pesquisados	539	1043	1784	1754
Elementos Encontrados	508	696	1181	890
Média de elementos encontrados	0,94	0,67	0,66	0,50

Tabela 4.6. Indicadores coletados nos projetos

Primeiramente, deve-se notar que em todos os casos um número superior a 30 pesquisas foram realizadas. Conforme já dito, as finalidades dessas pesquisas foram consolidadas e estão detalhadas no Apêndice A). Uma lista das pesquisas é mostrada abaixo:

- Quais são os códigos e motivadores diretamente afetados pela alteração do requisito?
- Quais são os códigos e motivadores indiretamente afetados pela alteração do requisito?
- Qual é a razão do não cumprimento do Planejado?
- O código que foi alterado é o previsto para ser alterado?
- Quais são os outros requisitos afetados pela alteração?
- “Quem?”, “Quando?”, “Por que?” da alteração
- Quais são os requisitos alocados para a entrega?
- Quais são os Requisitos que estão em estado de implementação?
- Quais são os requisitos já satisfeitos pelo código até o momento?

- Quais são os agrupamentos de questões que foram afetadas pela alteração?
- Quais são as razões de se retirar questões do escopo da versão?
- Quais questões foram canceladas e porque?
- Quais questões estão em implementação?
- Quais questões estão implementadas?
- Quais questões foram incluídas na versão?
- Quais questões foram planejadas no início da versão?
- Quais questões estão planejadas para a versão?
- Quais questões foram retiradas do planejamento da versão?
- Quais são as razões da inclusão de questão no planejamento da versão?
- Quais são as razões para o cancelamento de questões?
- Quais são os requisitos afetados pela questão planejada?
- Quais são os requisitos afetados pela questão resolvida?

Para cada pesquisa que um Gerente de Projeto fez, ele preencheu o formulário indicado na Seção 4.6 onde indica a razão da pesquisa. Essa lista é a consolidação do preenchimento das pesquisas realizadas pelos Gerentes de Projeto.

Os valores médios de elementos encontrados por pesquisa realizada indicam que o Modelo RAISE satisfaz mais as necessidades das Partes Interessadas do que os projetos que utilizaram o modelo matricial. Essa conclusão se baseia no fato que a média de elementos encontrados nos projetos que utilizaram o Modelo RAISE (tanto no Grupo 1 quanto no Grupo 2) é superior à média de elementos encontrados no projetos que utilizam o modelo matricial. A conclusão do Estudo de Caso não pode ser generalizada, mas pode ser usada como ponto de partida para uma investigação mais minuciosa do uso do Metamodelo RAISE.

Uma análise de ameaças ao Estudo de Caso foi executada e é mostrada na Seção 4.7.2.

4.7.2 Ameaças à Validade do Estudo de Caso

Segundo Wohlin, para se generalizar o resultado do experimento deve-se confrontar o experimento com algumas ameaças a sua validade. Wohlin propõe o estudo das ameaças sobre:

- Ameaças Internas à Validade: são influências que podem afetar a variável independente no que diz respeito à causalidade, sem o conhecimento dos pesquisadores.
- Ameaças Externas à Validade: cobre questões que limitam a possibilidade de generalizar os resultados.
- Ameaças à Validade da Conclusão: são ameaças que tratam de problemas que afeta a habilidade de alcançar a conclusão correta.
- Ameaças à Validade Teórica ("*Construct Validity*"): cobre problemas relacionados ao projeto do experimento e problemas relativo ao comportamento dos participantes do experimento.

O Estudo de Caso não se mostrou resistente a algumas ameaças. Ao confrontar o experimento às ameaças externas, percebe-se que o escopo do Estudo de Caso não é representativo e, portanto, não pode ser generalizado para qualquer uso do Metamodelo RAISE.

Quanto às ameaças à validade da conclusão, se percebe que não se usou ferramentas estatísticas que garantam a validade da conclusão, o que aumenta o risco de uma conclusão equivocada. Isso porque apenas quatro projetos foram utilizados no experimento. Também se considerou o uso da medida UMM para escolha de projetos de mesmo tamanho. Como a medida não foi validada o risco dos projetos não serem de tamanhos semelhantes também existe.

Dessa forma, a conclusão não pode ser generalizada, e o modelo não foi efetivamente validado. Mas também se verificou no escopo do Estudo de Caso e para os projetos investigados que a conclusão é valida. Assim uma investigação mais detalhada das ameaças internas foi executada.

4.7.3 Ameaças Internas à validade

As ameaças internas à validade do Estudo de Caso foram identificadas e analisadas tendo como guia a técnica de Wohlin e outros [Wohlin et al., 2000].

Como método comparativo paralelo foi utilizado no Estudo de Caso, os tipos de ameaças internas identificadas foram para grupos múltiplos. No caso foram discutidas as seguintes categorias de ameaças:

- Ameaças de Seleção-História
- Ameaças de Seleção-Maturação
- Ameaças de Seleção-Teste
- Ameaças de Seleção-Instrumentação
- Ameaças de Seleção-Mortalidade
- Ameaças de Seleção-Regressão

A sessão para identificação das ameaças foi realizada com os gerentes de projeto utilizando a técnica baseada no Brainstorming [Osborn, 1957]. A sessão teve uma duração de 40 minutos tendo o autor da dissertação como moderador e "*timekeeper*".

As ameaças localizadas e discutidas foram documentadas em um Diagrama de Causa e Efeito baseado no proposto por Ishikawa [Ishikawa, 1982] e mostrado na Figura 4.13. O diagrama construído não possuem as mesmas categorias propostas por Ishikawa por ter sido concebido com base nas Ameaças Internas de Wohlin. O diagrama foi usado apenas como forma de documentação da conclusão da sessão de Brainstorming.

Cada causa potencial identificada no diagrama de Causa e Efeito foi investigada. Essa investigação e conclusão alcançada são apresentadas a seguir:

Causa potencial "Tarefas mais adequadas ao Modelo RAISE": Esta causa investiga se a diferença das médias se deve ao fato que as tarefas selecionadas para o Estudo de Caso foram mais adequadas ao Modelo RAISE do que ao modelo matricial CONRAS. E se devido a isso o Modelo RAISE satisfez as necessidades das Partes Interessadas mais do que o CONRAS. A conclusão alcançada foi que esta não foi a causa da diferença da média. Isso porque o Modelo RAISE foi gerado para se adaptar ao processo de desenvolvimento de software já utilizado na organização que está estruturado esperando um modelo matricial. Pode-se dizer que as tarefas estão mais adaptadas ao modelo matricial do que ao Modelo RAISE.

Causa potencial "Tarefas não executadas adequadamente": esta causa investiga se a diferença nas médias se deve ai fato que as tarefas do processo não foram adequadamente executadas. Auditorias são realizadas sobre as tarefas e artefatos gerados na

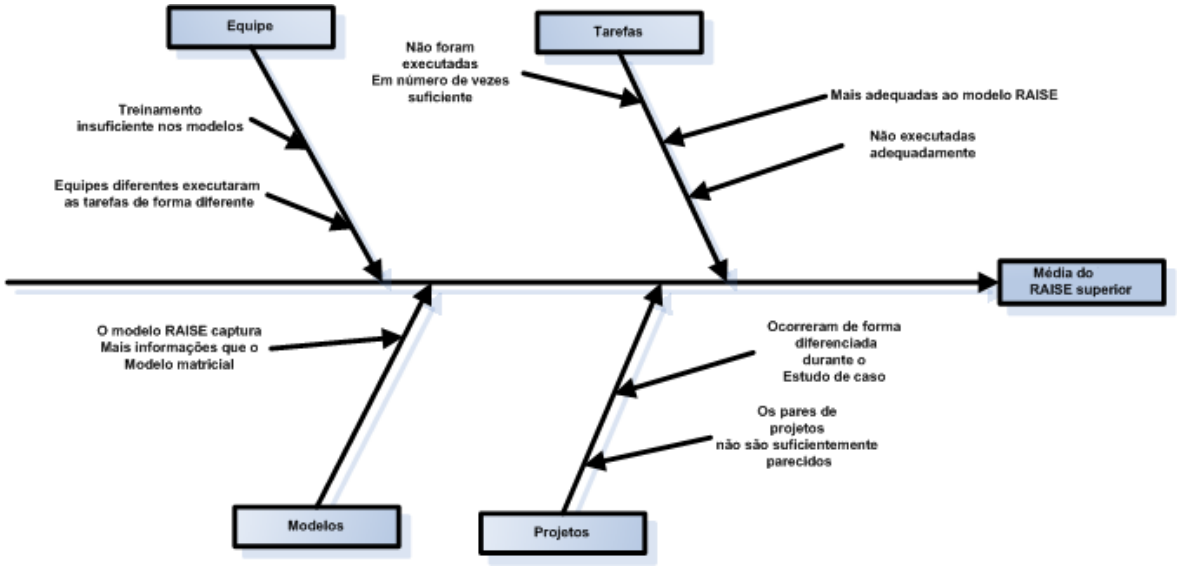


Figura 4.13. Diagrama Causa e Efeito para identificação das possíveis causas de serem encontrados mais elementos nos Projetos que utilizam o Modelo RAISE

corporação em estudo. O resultado das auditorias pode ser visto no Anexo C e indica que as tarefas utilizadas no Estudo de Caso foram executadas adequadamente.

Execução das Tarefas				
	Grupo 1		**Grupo 2**	
Tarefas Consumidoras	**P1**	**P2**	**P3**	**P4**
Acompanhar o Projeto	23	29	32	26
Controlar Alterações	6	2	3	0
Verificar se os Objetivos Estão Sendo Alcançados	0	0	7	7

Tabela 4.7. Número de vezes que ocorreu a Execução das Tarefas

Causa potencial "As tarefas não foram executadas um número suficiente de vezes". Esta causa investiga se a diferença das médias se deve ao fato que as tarefas do processo de desenvolvimento de software escolhidas para o Estudo de Caso não foram executadas um número suficiente de vezes, o que poderia alterar as médias apresentadas. A Tabela 4.7 mostra o número de vezes que cada tarefa foi executada. Observa-se que a tarefa de acompanhamento foi a mais executada, portanto, a que mais influenciou o resultado. E observa-se também que o número de vezes que a tarefa foi executada é semelhante entre os projetos.

Uma análise por elemento pesquisado foi realizada para entender quais tipos de pesquisas um modelo é capaz de satisfazer e quais o modelo não satisfaz.

A Tabela 4.8 mostra o resultado da análise. As pesquisas realizadas e consolidadas

	Grupo 1		Grupo 2	
Pesquisas Realizadas	**P1**	**P2**	**P3**	**Ferram.**
Quais são os códigos e motivadores diretamente afetados pela alteração do requisito?	Sim	Sim		
Quais são os códigos e motivadores indiretamente afetados pela alteração do requisito?	Sim			
Qual é a razão do não cumprimento do Planejado?	Parcial	Não		
O código que foi alterado é o previsto para ser alterado?	Sim			
Quais são os outros requisitos afetados pela alteração?	Sim			
"Quem?", "Quando?", "Por que?" da alteração	Sim	Não		
Quais são os requisitos alocados para a entrega?	Sim	Sim	Não	
Quais são os Requisitos que estão em estado de implementação?	Parcial	Não	Sim	
Quais são os requisitos já satisfeitos pelo código até o momento?	Sim	Sim	Sim	Não
Quais são os agrupamentos de questões que foram afetadas pela alteração?			Sim	
Quais são as razões de se retirar questões do escopo da versão?			Sim	
Quais questões foram canceladas e porque?			Sim	Não
Quais questões estão em implementação?			Sim	Não
Quais questões estão implementadas?			Sim	Não
Quais questões foram incluídas na versão?			Sim	Sim
Quais questões foram planejadas no início da versão?			Sim	Não
Quais questões estão planejadas para a versão?			Não	
Quais questões foram retiradas do planejamento da versão?			Sim	Não
Quais são as razões da inclusão de questão no planejamento da versão?			Sim	Não
Quais são as razões para o cancelamento de questões?			Sim	Não
Quais são os requisitos afetados pela questão planejada?			Sim	
Quais são os requisitos afetados pela questão resolvida?			Sim	Sim

Tabela 4.8. Tipo de pesquisa que o modelo satisfaz

formam as linhas da tabela. Para cada projeto foi indicado se o modelo usado satisfez a pesquisa realizada. A tabela foi montada com os valores:

- "Sim", para quando o modelo satisfez a pesquisa,
- "Não", para quando o modelo não satisfez a pesquisa,
- "Parcial", para quando o modelo satisfez a pesquisa parcialmente e,
- em branco, para quando a pesquisa não foi realizada no projeto ou para quando a pesquisa foi realizada mas não há valores confiáveis coletados (os valores são medidos como confiáveis ou não segundo a escala Likert [Likert, 1932] mostrada na Seção 4.1.1).

A tabela apresenta a seguinte informação:

Das pesquisas que apresentam informação conclusiva, as seguintes representam um melhor desempenho do Modelo RAISE:

- Qual é a razão do não cumprimento do Planejado?
- "Quem?", "Quando?", "Por que?" da alteração
- Quais são os Requisitos que estão em estado de implementação?
- Quais são os requisitos já satisfeitos pelo código até o momento?
- Quais questões foram canceladas e porque?
- Quais questões estão em implementação?
- Quais questões estão implementadas?
- Quais questões foram planejadas no início da versão?
- Quais questões foram retiradas do planejamento da versão?
- Quais são as razões da inclusão de questão no planejamento da versão?
- Quais são as razões para o cancelamento de questões?

O conjunto de pesquisas realizadas no Grupo 1 é diferente do conjunto de pesquisas realizadas no Grupo 2. Isso se deve a diferença contextual entre os projetos do Grupo 1 e do Grupo 2, principalmente pelo fato que os projetos do Grupo 1 são de desenvolvimento enquanto os projetos do Grupo 2 são de manutenção. Por exemplo, pesquisas sobre o elemento "Questão" só foram realizadas no Grupo 2.

Na maioria das pesquisas realizadas se observa um valor absoluto: ou o modelo possui a informação requerida ou não possui. Estas pesquisas dependem apenas do modelo onde estão sendo pesquisadas e sua compatibilidade com o projeto. Por exemplo, o Modelo RAISE é capaz de retornar a informação da pesquisa "Quais são os códigos e motivadores diretamente afetados pela alteração do requisito?" sempre que ela for pedida, não importando quantas vezes isto irá ocorrer.

Mas ocorreram casos como "Qual é a razão do não cumprimento do Planejado?" e "Quais são os Requisitos que estão em estado de implementação?" onde o modelo gerado a partir do RAISE não foi capaz de retornar a informação completa. Para o primeiro caso, a razão do não cumprimento está na existência de elementos que foram planejados para a versão mas não foram registrados como Rastros. Por exemplo, um código planejado mas ainda não desenvolvido e, portanto, sem Rastro entre ele e o requisito.

Já o caso da pesquisa "Quais são os Requisitos que estão em estado de implementação?" ocorre a mesma situação do "Qual é a razão do não cumprimento do Planejado?". Os desenvolvedores nesta organização não registram todos os códigos que estão em desenvolvimento, mas apenas quando o desenvolvimento alcança um ponto de estabilidade. Apenas nesse momento é registrado o Rastro de satisfação entre código e os requisitos.

Sendo assim, para que a causa potencial "As tarefas não foram executadas um número suficiente de vezes" tenha sido a real razão do melhor desempenho do Modelo RAISE, precisou-se entender se "Qual é a razão do não cumprimento do Planejado?" e "Quais são os Requisitos que estão em estado de implementação?" poderiam ter comportamento distinto do observado se executados mais vezes.

O "Quais são os Requisitos que estão em estado de implementação?" foi usado pelos gerentes de projeto durante a tarefa de "Acompanhamento do Projeto". Essa pesquisa é uma de 4 pesquisas comumente realizada nesta tarefa: "Quais são os requisitos alocados para a entrega?", "Quais são os requisitos já satisfeitos pelo código até o momento?", "Quais são os Requisitos que estão em estado de implementação?".

Os gerentes de projeto identificaram que o "Quais são os Requisitos que estão em estado de implementação?" depende do número de desenvolvedores atuando na equipe no momento. Um gráfico de dispersão foi criado para verificar a correlação a partir da informação de número de elementos em implementação (que é coincidente com o número de elementos pesquisados no modelo de Rastreabilidade e o número de desenvolvedores nos projetos). Obteve-se o gráfico da Figura 4.14.

O coeficiente de correlação de Pearson foi calculado com o valor de 0,903, indicando uma correlação forte positiva. Desta forma pode-se dizer que o número de vezes

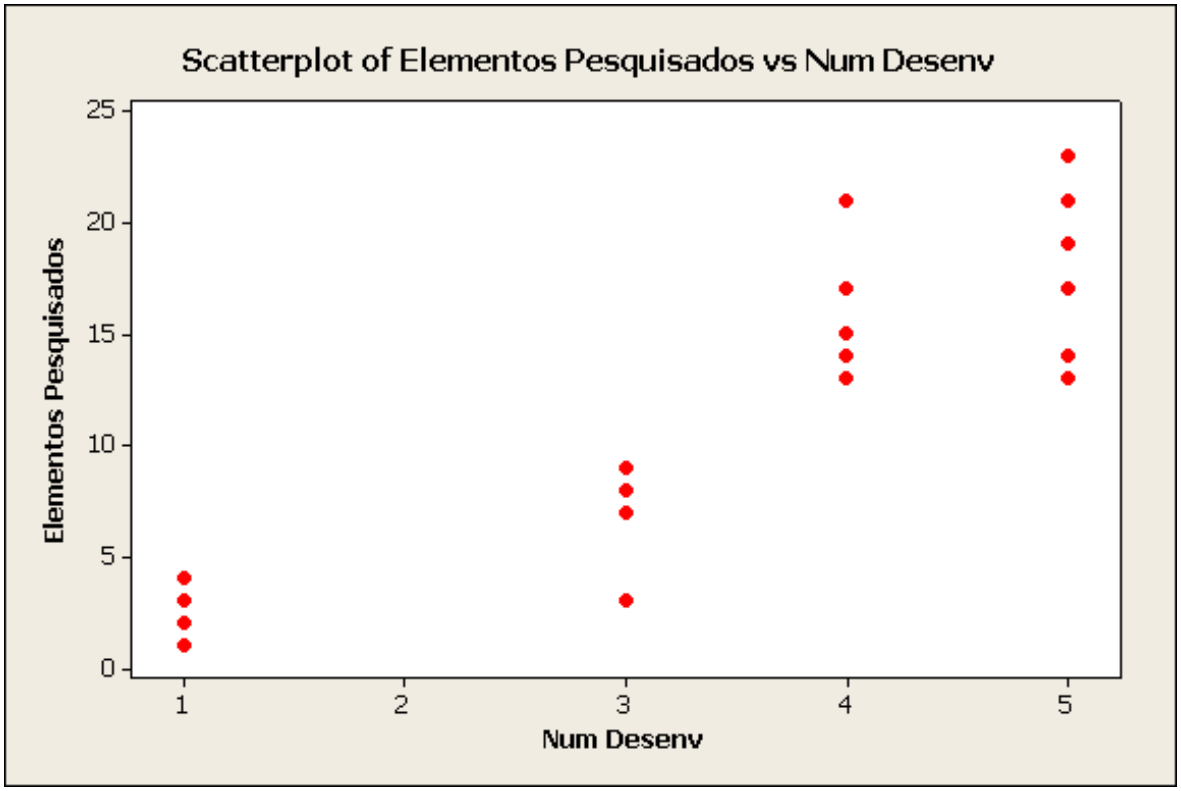


Figura 4.14. Gráfico de Dispersão de elementos pesquisados vs Número de Desenvolvedores

que a tarefa de acompanhamento for executada não irá mudar o cenário, desde que se mantenha o número de desenvolvedores.

A outra pesquisa que precisou ser investigada foi "Qual é a razão do não cumprimento do Planejado?". Ele também está vinculado ao número de desenvolvedores. A variação do valor do "Qual é a razão do não cumprimento do Planejado?" é dependente do desenvolvimento que está ocorrendo e que ainda não gerou Rastros. O número de desenvolvimento ocorrendo simultaneamente depende do número de desenvolvedores trabalhando simultaneamente. A Figura 4.15 mostra a dispersão desse caso.

O coeficiente de correlação de Pearson foi calculado de 0,709, com o p-value de 0,022. A correlação existente não é tão forte quanto o indicado no primeiro caso, mas o suficiente para indicar que uma alteração significativa dos valores não irá ocorre sem uma mudança no número de desenvolvedores na equipe.

Desta forma pode-se concluir que a causa "As tarefas não foram executadas um número suficiente de vezes" não é a causa do Modelo RAISE satisfazer mais que o modelo matricial.

A causa potencial "Treinamento Insuficiente no Modelo" que indica que a causa da média apontar para um melhor desempenho para o Modelo RAISE é devido à falta

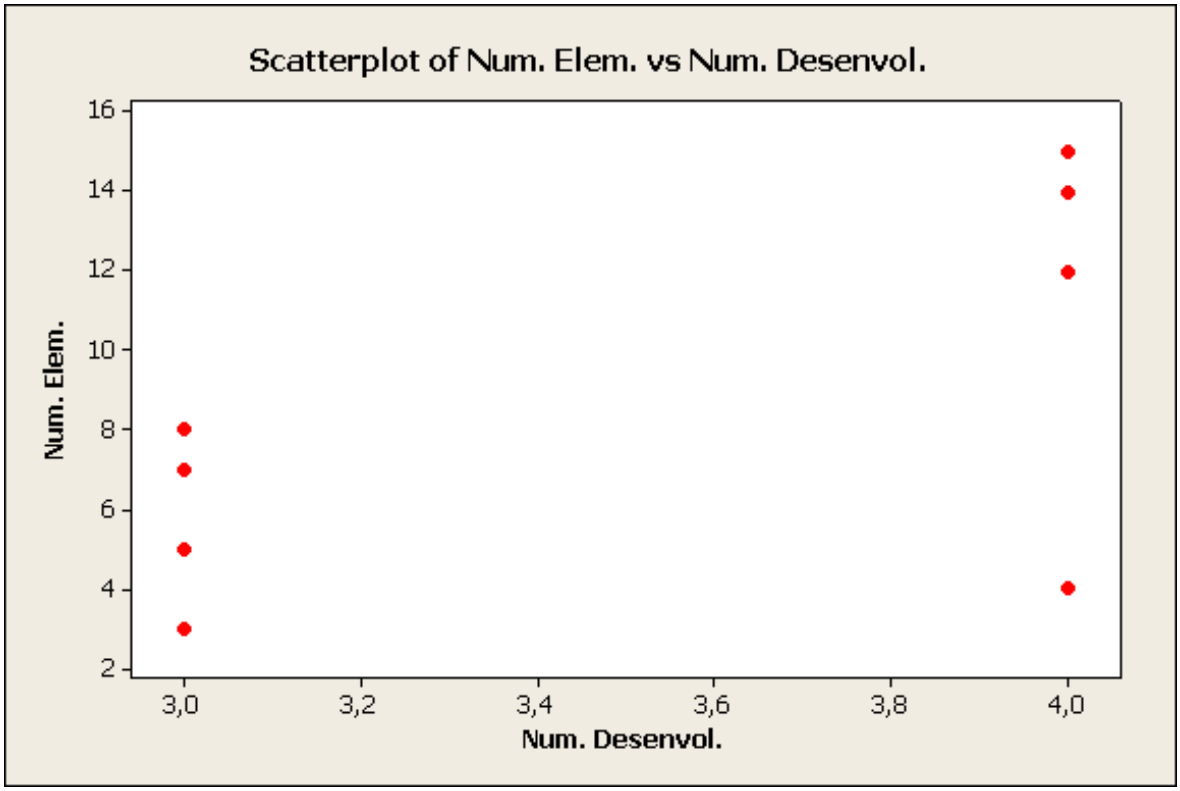


Figura 4.15. Dispersão entre Número de Desenvolvedores e Numero de elementos pesquisado

de treinamentos dos profissionais envolvidos no Estudo de Caso. Este caso não é a causa raiz do problema porque todos os participantes foram treinados previamente no modelo e sofreram auditoria de suas tarefas. O resultado da auditoria está no Anexo C e mostra o total domínio dos modelos utilizados.

A causa potencial "Equipes diferentes executam as tarefas de forma diferentes", que significa que, sendo equipes diferentes entre os projetos, as tarefas podem ser executadas de forma diferenciada e essa se a causa do melhor desempenho do Modelo RAISE. Também não se mostra verdadeira devido ao fato que as ações medidas foram executadas pelos gerentes de projetos e para cada grupo um mesmo gerente de projeto executou as mesmas tarefas nos dois projetos, com modelos distintos.

A causa potencial "os projetos ocorreram de forma diferenciada durante o Estudo de Caso" não é causa nesse caso. Isso pode ser verificado através da Tabela 4.7, que mostra que para cada tarefa, os projetos do mesmo grupo apresentam o número de vezes que foram executados bastante próximos entre si.

Desta forma, mesmo se os projetos foram conduzidos de forma diferente nas outras tarefas, essas não afetaram as tarefas monitoradas pelo Estudo de Caso.

A causa potencial "Os pares de projeto não são suficientemente parecidos" não se

mostra verdadeiro nessa situação, já que os critérios utilizados na seleção dos projetos foram atendidos.

Por fim, a causa "O Modelo RAISE é mais eficaz que o modelo matricial" parece ser a mais próxima da verdade neste Estudo de Caso.

Avaliou-se as pesquisas aos elementos executadas pelos gerentes de projetos em que o Modelo RAISE foi mais eficiente que o modelo matricial, e se observou que a razão disso é que o Modelo RAISE registra informação do contexto do Rastro que o modelo matricial não registra. Por exemplo: na pesquisa ""Quem?", "Quando?", "Por que?" da alteração" o Modelo RAISE registra cada informação em seus elementos de contexto, enquanto o modelo matricial não possui este registro.

4.7.4 Resultado da Interpretação

Observou-se então que no escopo do Estudo de Caso os resultados são válidos. Mas não podem ser generalizados e considerados verdadeiros para outros projetos.

Dessa forma, observando a meta do Estudo de Caso, temos que o resultado aponta para que o modelo gerado a partir do Metamodelo RAISE satisfez mais do que o modelo matricial as necessidades dos agentes que executam tarefas de verificação para se tomar alguma decisão (gerente de projetos no caso) nos projetos selecionados.

4.8 Conclusão

Neste capítulo foi exibido o Estudo de Caso para avaliar o Metamodelo RAISE. Primeiramente, se contextualizou o Estudo de Caso na organização onde foi executado. Os projetos e processos da organização foram apresentados. Então a metodologia utilizada foi mostrada, para em seguida se apresentar os resultados obtidos para cada subprocesso da metodologia.

No Subprocesso de Definição, as metas, questões e métricas que dirigiram todo o Estudo de Caso foram definidas. Alguns indicadores foram identificados para ajudar a avaliar o quanto o modelo satisfaz as necessidades das Partes Interessadas.

Na Seção Subprocesso de Planejamento todo o planejamento do Estudo de Caso foi mostrado. Também se mostrou quais foram os sistemas utilizados, os critérios de seleção; como e quando o Estudo de Caso aconteceu; as implementações dos modelos.

A seção seguinte, Subprocesso Coleta de Dados, descreveu como ocorreu a coleta de dados, para que na próxima seção, Subprocesso de Interpretação, os dados coletados fossem analisados.

A conclusão da análise é que no Estudo de Caso modelo definido através do Metamodelo RAISE satisfez mais as necessidades das Partes Interessadas do que o modelo matricial nos projetos avaliados.

Capítulo 5

Conclusão

5.1 Considerações Gerais

Rastreabilidade é uma prática importante para a Engenharia de Software. Apesar disso ainda é pouco e mal usada [Mader et al., 2009]. O seu desenvolvimento ocorreu até o momento de forma desbalanceada. Diferentes comunidades investem na Rastreabilidade como um atributo exclusivo de seus elementos de trabalho [Winkler & von Pilgrim, 2009]. Esforços para torná-la uma questão que envolve toda a Engenharia de Software se iniciaram mais recentemente.

Diferentes métodos focando em diferentes aspectos da Rastreabilidade foram criados para permitir o seu bom uso, mas até o momento nenhum obteve o sucesso pretendido [Aizenbud-Reshef et al., 2006; Evans, 1989; Bowen et al., 1990; Cooke & Stone, 1991; Lefering, 1993; Alexander, 2002; Tang, 1997]. Para que a Rastreabilidade seja efetivamente usada, a diversidade de necessidades das partes interessadas deve ser entendida e métodos focados nessas necessidades devem ser adotados [Egyed et al., 2007]. As necessidades distintas das partes interessadas conduzem uma única organização a usar uma conjunção de diferentes métodos para suprir todas suas necessidades. Então, uma base de informação de Rastreabilidade compartilhada entre os métodos se faz necessária. Essa base é concebida através de um Modelo de Informação de Rastreabilidade, que é dependente da cultura organizacional onde a Rastreabilidade está inserida. O que significa que diferentes organizações possuem diferentes Modelos de Informação de Rastreabilidade. Os diferentes modelos devem ser especificados usando uma sintaxe e semântica única. Algumas propostas de gramáticas para a Informação da Rastreabilidade já existem na literatura. Neste trabalho estas propostas foram apresentadas no formato de metamodelos na Seção 2.5.

Há ainda muita divergência sobre o que é um bom metamodelo de Informação de Rastreabilidade [Winkler & von Pilgrim, 2009]. O critério usado neste trabalho para definir a qualidade do metamodelo de Informação da Rastreabilidade está nos aspectos identificados por Pohl [Pohl, 1996] e discutidos na Seção 2.1. São metamodelos que permitem a especificação de Modelos de Informações da Rastreabilidade, cujos Rastros estejam inseridos em seus contextos. Esses modelos registram informações mais ricas para as partes interessadas daquelas que apenas registram os Rastros.

No Capítulo 3 propomos o metamodelo de informação da Rastreabilidade RAISE. O RAISE é um metamodelo que define uma gramática para Modelos de Informação de Rastreabilidade onde o registro de um Rastro está estruturado em seu contexto no processo de desenvolvimento de software. A avaliação do Metamodelo RAISE foi descrita no Capítulo 4. Como resultado da avaliação, percebeu-se que nos projetos avaliados houve mais eficiência nessa forma de registro que na forma matricial. A

avaliação também demonstra que a principal característica do metamodelo RAISE, a de estar vinculado ao contexto dos processos de desenvolvimento de software, o aproxima mais das necessidades das partes interessadas do que o modelo matricial.

5.2 Contribuições

A principal contribuição deste trabalho é a definição de um Metamodelo de Informação de Rastreabilidade inserido no processo de desenvolvimento de software. A literatura já indicava para a necessidade da contextualização de um Rastro, mas os metamodelos até então concebidos não estruturavam esse contexto. A utilização de elementos de metamodelos de processo de desenvolvimento de software permitiu a estruturação do contexto do Rastro. Essa estruturação facilita a integração do modelo de informação da rastreabilidade com as ferramentas de suporte ao desenvolvimento de software da organização.

Uma contribuição conceitual do trabalho é a relação entre Regras de Negócio de uma tarefa do processo de desenvolvimento com os tipos de Rastros. A percepção que um Rastro pode ser definido por um conjunto de regras ("Rastros Funcionais") e essas regras são Regras de Negócio descritas na especificação de uma tarefa representa uma visão diferente das mostradas na literatura revisada (vide Capítulo 2).

5.3 Trabalhos Relacionados

Existe na literatura diversas propostas de metamodelos de rastreabilidade como já mostrado no na Seção 2.5. Um metamodelo bastante referenciado é o proposto por Ramesh e Jarke [Ramesh & Jarke, 2001]. É um metamodelo simples que utiliza elementos da contextualização do rastro assim como nesta dissertação. Entretanto, não permite que um rastro seja representado com todos os elementos do seu contexto. Também não representa o trabalho gerador de cada rastro.

Outro metamodelo foi proposto por Spanoudakis e outros [Spanoudakis et al., 2004]. A característica de permitir expansão torna o metamodelo bastante flexível, apesar de não modelar o contexto do rastro diretamente. Informação do contexto é modelada através de meta-elementos de critério: "*artefact creation*", "*artefact editing*", "*artefact viewing*", etc.

Espinoza e outros [Espinoza et al., 2006] fizeram um metamodelo baseado nas propostas de modelos encontrados na literatura. Sua proposta foi apresentada no formato de esquemas (*"schema"*) e utilizando a linguagem Caseml (baseada em XML)

para especificar a ligação de Rastreabilidade. Espinoza e outros definiram um grupo de meta-classes denominadas "Traceability meta-type". É nesse grupo que se define todo tipo de Rastreabilidade e as regras de ligações válidas entre os artefatos do desenvolvimento de software. Foi criado um "Conjunto de tipos de Rastreabilidade" (TYS - "*Traceability Type Set*"). Também definiram grupos para regras e conjunto de usuários ("*Rule and User set*"), conjunto mínimo de ligações ("Minimal Link Set") e métricas ("Metrics Set"). É uma proposta que foca no o aspecto da utilização dos rastros, entretanto, não contextualiza o rastro como o RAISE.

A preocupação com a família de produtos é o aspecto central da proposta de Lago e outros [Lago et al., 2004]. Nesse metamodelo há a relação de elementos em três camadas: família de produtos, produto, características do produto. Apesar de não contextualizar o rastro, o metamodelo mostra que a generalização para famílias de produtos é interessante para a rastreabilidade.

Em sua tese de doutorado, Gengivir [Genvigir, 2009], apresenta um metamodelo que apenas representa os elementos e suas relações. A preocupação do contexto do rastro, assim como modelada no RAISE, não é representada nesse modelo.

No artigo em que propõem uma integração dos estudos da Rastreabilidade, Winkler e Pilgrim [Winkler & von Pilgrim, 2009] apresentam um metamodelo simples, mas com a presença de um elemento para representar o contexto. Entretanto, o elemento contexto é muito genérico, o que não facilita a concepção do modelo de Rastreabilidade.

5.4 Trabalhos Futuros

5.4.1 Adaptação dos métodos de Rastreabilidade

A Rastreabilidade é ainda pouco e mal usada [Mader et al., 2009]. Apesar do Metamodelo RAISE não ser a solução para essa questão, pode ser usado como uma base para sustentar métodos que permitam um melhor uso da Rastreabilidade.

Dessa forma, o trabalho de desenvolver ou adaptar métodos de Rastreabilidade para o uso do Metamodelo RAISE se faz necessário. Os métodos avaliados e apresentados na Seção 2.3 foram pensados sobre algum modelo de Rastreabilidade. É interessante que esses métodos sejam adaptados para usarem o Metamodelo RAISE. Um trabalho futuro está em conceber uma ferramenta que utilize o Metamodelo RAISE para se especificar um Modelo de Informação de Rastreabilidade. A ferramenta deve permitir que os seus usuários utilizem de métodos de rastreabilidade no modelo por eles concebido.

O fato do Metamodelo RAISE estar inserido no processo de desenvolvimento de software permite que uma ferramenta possa gerar modelos integrados com outras ferramentas que suportam o processo de desenvolvimento utilizado em uma organização. E assim se possa facilitar a utilização da Rastreabilidade. Por exemplo, um programador, ao terminar a tarefa de codificar e salvar seu trabalho em uma biblioteca de gestão de configuração, deve registrar o ocorrido em um Modelo de Informação de Rastreabilidade. O Modelo de Informação de Rastreabilidade gerado pela ferramenta pode estar integrado com o ambiente de codificação e já registrar automaticamente os rastros com seus atributos (elementos, tarefa executada, executor, data hora de execução).

5.4.2 Área de Conhecimento Rastreamento

Outro trabalho futuro é a estruturação da área de conhecimento Rastreamento utilizando o Metamodelo RAISE como suporte às tarefas da área.

Para o uso adequado do Metamodelo RAISE é necessário haver uma tarefa específica na organização para se definir o Modelo de Informação de Rastreabilidade utilizado em cada projeto. A tarefa de desenho ("*design*") do modelo de Rastreabilidade deve ser descrita e incorporada no processo utilizado. Caso a organização possua processos corporativos que são instanciados e personalizados em cada projeto, a informação da Rastreabilidade deve também ser modelada de uma forma corporativa.

Essa tarefa deve ser acompanhada de outras com a finalidade de manter a informação de Rastreabilidade. Segundo Egyed e outros, "mais esforço se faz necessário nas tarefas de manutenção e melhorias das informações de Rastreabilidade do que nas tarefas de definição" [Egyed et al., 2007].

Dessa forma, a manutenção na informação da Rastreabilidade pode ser vista como uma disciplina ou área de conhecimento, o Rastreamento, onde essas tarefas são mantidas. Winkler e Pilgrim [Winkler & von Pilgrim, 2009] já anunciavam a Rastreabilidade como uma disciplina emergente. E essa visão é importante para que as diferentes comunidades que estudam a Rastreabilidade possam somar esforços. Nesse aspecto o Metamodelo RAISE pode ajudar por permitir a representação dos elementos das diferentes áreas de conhecimento, pois essas áreas são representadas por subprocessos do processo de desenvolvimento de software.

5.4.3 Utilização do Metamodelo RAISE na Automação da Engenharia de Software

Um terceiro trabalho futuro é a utilização do Metamodelo RAISE como identificação de oportunidades de automação do processo de desenvolvimento de software.

Conforme discutido na Seção 3.4, o fato do RAISE direcionar para que modelos sejam integrados com elementos do processo de desenvolvimento de software utilizados na organização permite a identificação de ligações entre elementos. Ligações que podem ser automatizadas.

O trabalho futuro deve objetivar o desenvolvimento de um método para se gerar mecanismos que possam configurar e integrar um conjunto de ferramentas que apoiam o desenvolvimento de software a partir de um modelo de Informação de Rastreabilidade concebido pelo Metamodelo RAISE.

5.4.4 Automação do Desenho ("*Design*") do Modelo de Informação da Rastreabilidade

Deve-se também desenvolver uma ferramenta que automatiza o desenho ("*Design*") e a implementação de um Modelo de Informação de Rastreabilidade. Sugerimos a apropriação de conceitos do MDE para executar transformações para modelos dependentes de plataforma (que, nesse caso, devem ser modelos de configuração e integração entre ferramentas) a partir de um modelo conceitual de rastreabilidade, montado pelo usuário, com base no Metamodelo RAISE previamente configurado na ferramenta.

Para isso ser possível, a ferramenta proposta deve se valer de modelos de transformações para cada ferramenta utilizada, e um metamodelo com tipos de ligações de rastreabilidade vinculadas às oportunidades de automação derivadas de cada um deles.

5.4.5 Outras propostas de estudos

Outro aspecto que merece um estudo futuro é a aplicação do Metamodelo RAISE em Sistemas. O Metamodelo RAISE foi concebido para Software, mas pode ser expandido para ser usado em Sistemas.

Por último, deve-se pensar que a Rastreabilidade é um segmento da Engenharia de Software que manipula artefatos de todo o ciclo de desenvolvimento. Como já discutido na Seção 3.4, existem muitas ferramentas disponibilizadas no mercado que propõem diversas formas de automatização do processo de desenvolvimento de software [Aksyonov et al., 2009]. A automação de um modelo de Rastreabilidade deve

estar integrada com a automação de todo o processo de desenvolvimento. Assim o Metamodelo RAISE pode ser revisto, e integrado em um metamodelo de Informação de Engenharia de Software. Ao invés de se ter elementos do metamodelo de processo de Desenvolvimento de Software dentro do Metamodelo RAISE, deve-se ter o Metamodelo RAISE dentro do metamodelo de processo de desenvolvimento de software.

Das iniciativas para automação da Engenharia de Software, a ALM (Application Lifecycle Management) está ganhando espaço [Schwaber, 2006]. Nessa abordagem há uma integração de ferramentas que suportam o desenvolvimento de software com base em um barramento de elementos da Engenharia de Software. Um exemplo é o Jazz.net desenvolvido pela IBM Rational e disponibilizada como projeto "Open Source" [IBM, 2011]. O Metamodelo RAISE pode se integrar com sistemas ALM e trabalhar com elementos definidos em seus barramentos.

Referências Bibliográficas

Aizenbud-Reshef, N.; Nolan, B.; Rubin, J. & Shaham-Gafni, Y. (2006). Model traceability. *IBM Systems Journal*, 45(3):515--526.

Aizenbud-Reshef, N.; Paige, R.; Rubin, J.; Shaham-Gafni, Y. & Kolovos, D. (2005). Operational semantics for traceability. In *ECMDA-Traceability Workshop, Nuremberg*, pp. 7--14.

Aksyonov, K.; Spitsina, I.; Bykov, E.; Kai, W. & Smoliy, E. (2009). Multiple approaches integration for computer-supported software development. In *Proceedings of the 21st annual international conference on Chinese control and decision conference*, pp. 4960--4964. IEEE Press.

Alexander, I. (2002). Towards automatic traceability in industrial practice. In *Proc. of the 1st Int. Workshop on Traceability*, pp. 26--31.

Almeida, J.; Van Eck, P. & Iacob, M. (2006). Requirements traceability and transformation conformance in model-driven development. In *10th IEEE International Enterprise Distributed Object Computing Conference, 2006. EDOC'06*, pp. 355--366.

Amar, B.; Leblanc, H. & Coulette, B. (2008). A traceability engine dedicated to model transformation for software engineering. In *ECMDA Traceability Workshop (ECMDA-TW)*, pp. 7--16.

Amelunxen, C.; Klar, F.; Königs, A.; Rötschke, T. & Schürr, A. (2008). Metamodel-based tool integration with MOFLON. In *Proceedings of the 30th international conference on Software engineering*, pp. 807--810. ACM.

Antoniol, G.; Canfora, G.; Casazza, G.; De Lucia, A. & Merlo, E. (2002). Recovering traceability links between code and documentation. *Software Engineering, IEEE Transactions on*, 28(10):970--983.

Antoniol, G.; Casazza, C. & Cimitile, A. (2000). Traceability recovery by modeling programmer behavior. In *Reverse Engineering, 2000. Proceedings. Seventh Working Conference on*, pp. 240--247. IEEE.

Antoniol, G.; Merlo, E.; Guéhéneuc, Y. & Sahraoui, H. (2005). On feature traceability in object oriented programs. In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pp. 73--78. ACM.

Basili, V.; Caldiera, G. & Rombach, H. (1994). The goal question metric approach. *Encyclopedia of software engineering*, 1:528--532.

Bianchi, A.; Fasolino, A. & Visaggio, G. (2000). An exploratory case study of the maintenance effectiveness of traceability models. In *Program Comprehension, 2000. Proceedings. IWPC 2000. 8th International Workshop on*, pp. 149--158. IEEE.

Bowen, J.; O'Grady, P. & Smith, L. (1990). A constraint programming language for life-cycle engineering. *Artificial Intelligence in Engineering*, 5(4):206--220.

Chechik, M. & Gannon, J. (2001). Automatic analysis of consistency between requirements and designs. *Software Engineering, IEEE Transactions on*, 27(7):651--672.

Chrissis, M. B.; Konrad, M. & Shrum, S. (2006). *CMMI for Development, version 1.2*. Software Engineering Institute of Carnegie Mellon University.

Cleland-Huang, J.; Chang, C. & Christensen, M. (2003). Event-based traceability for managing evolutionary change. *IEEE Transactions on Software Engineering*, pp. 796--810.

Cleland-Huang, J.; Chang, C.; Sethi, G.; Javvaji, K.; Hu, H. & Xia, J. (2002). Automating speculative queries through event-based requirements traceability. *10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02)*, p. 289.

Cleland-huang, J.; Hayes, J. H. & Domel, J. M. (2009). Model-Based Traceability School of Computing. *Management*, pp. 6--10.

Cleland-Huang, J. & Schmelzer, D. (2003). Dynamically tracing non-functional requirements through design pattern invariants. In *Workshop on Traceability in Emerging Forms of Software Engineering, in conjunction with IEEE International Conference on Automated Software Engineering*.

Cooke, J. & Stone, R. (1991). A formal development framework and its use to manage software production. In *Tools and Techniques for Maintaining Traceability During Design, IEE Colloquium on*, p. 10. IET.

Cordy, J. R. (2003). Comprehending Reality - Practical Barriers to Industrial Adoption of Software Maintenance Automation. *11th IEEE International Workshop on Program Comprehension (IWPC03)*, p. 196.

Dahlstedt, A. G. & Persson, A. (2003). Requirements interdependencies-moulding the state of research into a research agenda. *Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2003), held in conjunction with CAiSE*, pp. 71--80.

Davis, A. (1993). *Software requirements: objects, functions, and states*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2 edição.

De Lucia, A.; Fasano, F.; Oliveto, R. & Tortora, G. (2005). Adams re-trace: A traceability recovery tool. In *Ninth European Conference on Software Maintenance and Reengineering, 2005. CSMR 2005*, pp. 32--41.

de Pádua Paula Filho, W. (2001). Requirements for an educational software development process. *ACM SIGCSE Bulletin*, 33(3):65--68.

de Pádua Paula Filho, W. (2003). *Engenharia de Software*. LTC, 2 edição.

de Pádua Paula Filho, W. (2010). http://homepages.dcc.ufmg.br/~wilson/praxis/.

Deng, M.; Stirewalt, R. & Cheng, B. (2005). Retrieval by construction: A traceability technique to support verification and validation of UML formalizations. *International Journal of Software Engineering and Knowledge Engineering*, 15(5):837--872.

Derniame, J.; Kaba, B. & Wastell, D. (1999). *Software process: principles, methodology, and technology*. Springer, 1 edição.

DoD (Department of Defense) (1988). MIL STD 2167A.

DoD (Department of Defense) (1994). IVNL-STD-498 SOFTWARE AND DEVELOPMENT DOCUMENTATION.

Drivalos, N.; Paige, R. F.; Fernandes, K. J. & Kolovos, D. (2008). Towards Rigorously Defined Model-to-Model Traceability. *ECMDA Traceability Workshop (ECMDA-TW)*, pp. 17 -- 26.

Eclipse, F. (2011a). http://www.eclipse.org/modeling/emf/.

Eclipse, F. (2011b). http://www.eclipse.org/modeling/gmp/.

Egyed, A.; Graf, F. & Grunbacher, P. (2010). Effort and Quality of Recovering Requirements-to-Code Traces: Two Exploratory Experiments. *2010 18th IEEE International Requirements Engineering Conference*, pp. 221--230.

Egyed, A.; Grunbacher, P.; Heindl, M. & Biffl, S. (2007). Value-Based Requirements Traceability: Lessons Learned. *15th IEEE International Requirements Engineering Conference (RE 2007)*, pp. 115--118.

El-Emam, K. & Garro, I. (1999). ISO/IEC 15504. International Organization for Standardization.

Eriksson, H. & Penker, M. (2000). *Business Modeling with UML: Business Patterns at Work*. John Wiley & Sons, Inc.

Espinoza, A.; Alarcon, P. & Garbajosa, J. (2006). Analyzing and systematizing current traceability schemas. In *Software Engineering Workshop, 2006. SEW'06. 30th Annual IEEE/NASA*, volume 30, pp. 21--32.

Evans, M. (1989). *The software factory: a fourth generation software engineering environment*, volume 8. John Wiley & Sons, Inc. New York, NY, USA.

Faisal, M. (2005). *Toward automating the discovery of traceability links*. PhD thesis, University of Colorado at Boulder.

Falleri, J. R.; Huchard, M. & Nebut, C. (2006). Towards a traceability framework for model transformations in kermeta. In *ECMDA-TW Workshop*.

Gale, T. & Eldred, J. (1997). *Getting results with the object-oriented enterprise model*. Cambridge University Press.

Genvigir, E. (2009). *Um Modelo para Rastreabilidade de Requisitos de Software Baseado em Generalizaçao de Elos e Atributos*. Tese de doutorado, Instituto Nacional de Pesquisas Espaciais - INPE.

Glinz, M. (2000). A lightweight approach to consistency of scenarios and class models. In *Requirements Engineering, 2000. Proceedings. 4th International Conference on*, pp. 49--58. IEEE.

Gonçalves, A.; Martins, F. & Carreira, P. (1998). IEEE Std 830 Prática Recomendada Para Especificações de Exigências de Software Standard Internacional.

Gotel, O. & Finkelstein, A. (1995). Contribution structures [Requirements artifacts]. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering, 1995.*, pp. 100--107.

Gotel, O. C. Z. & Finkelstein, A. C. W. (1994). An Analysis of the Requirements Traceability Problem. *Development*.

Gross, D. & Yu, E. (2001). From non-functional requirements to design through patterns. *Requirements Engineering*, 6(1):18--36.

Hayes, J.; Dekhtyar, A. & Osborne, J. (2003). Improving requirements tracing via information retrieval. In *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE*, pp. 138 -- 147. IEEE Computer Society.

IBM (2011). http://jazz.net.

IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*.

IEEE (1994). IEEE Standards Collection - Software Engineering.

Ishikawa, K. (1982). *Guide to quality control*. Asian Productivity Organization Minato-Ku, Japan, segunda ed edição.

ISO (2005). ISO / IEC 25000 - Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. International Organization for Standardization.

Jackson, J. (2002). A keyphrase based traceability scheme. In *Tools and Techniques for Maintaining Traceability During Design, IEE Colloquium on*, p. 2. IET.

Jacobson, I.; Booch, G. & Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley Professional.

Jouault, F. & Kurtev, I. (2005). Transforming models with ATL. In *Satellite Events at the MoDELS 2005 Conference*, pp. 128--138, Montego Bay, Jamaica. Springer.

Jouault, F. & Kurtev, I. (2006). On the architectural alignment of ATL and QVT. *Proceedings of the 2006 ACM symposium on Applied computing - SAC '06*, p. 1188.

Jung, J. (2007). Ontological framework based on contextual mediation for collaborative information retrieval. *Information Retrieval*, 10(1):85--109.

Kaindl, H. (1993). The missing link in requirements engineering. *ACM SIGSOFT Software Engineering Notes*, 18(2):30--39.

Kelleher, J. (2005). A reusable traceability framework using patterns. In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pp. 50--55, New York, NY, USA. ACM.

Kruchten, P. (2003). *The Rational Unified Process: An Introduction (3rd Edition)*. Addison-Wesley Professional.

Lago, P.; Muccini, H. & van Vliet, H. (2009). A scoped approach to traceability management. *Journal of Systems and Software*, 82(1):168--182.

Lago, P.; Niemela, E. & Van Vliet, H. (2004). Tool support for traceable product evolution. In *Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings. Eighth European Conference on*, pp. 261--269.

Lee, C.; Guadagno, L. & Jia, X. (2003). An Agile Approach to Capturing Requirements and Traceability. In *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003)*.

Lefering, M. (1993). An incremental integration tool between requirements engineering and programming in the large. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pp. 82--89. IEEE.

Letelier, P. (2002). A framework for requirements traceability in UML-based projects. In *1st International Workshop on Traceability in Emerging Forms of Software Engineering, In conjunction with the 17th IEEE International Conference on Automated Software Engineering*.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1 -- 55.

Liu, X. (2009). Process Oriented Analysis for Software Automation. In *Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on*, volume 1, pp. 1131--1133. IEEE.

Lucia, A. D.; Fasano, F. & Oliveto, R. (2004). Enhancing an artefact management system with traceability recovery features. *2004. Proceedings. 20th.*

Mader, P.; Gotel, O. & Philippow, I. (2009). Getting back to basics: Promoting the use of a traceability information model in practice. In *Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, pp. 21--25. IEEE Computer Society.

Marcus, A. & Maletic, J. (2003). Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pp. 125--135. IEEE.

Mellor, S. J.; Clark, A. N. & Futagami, T. (2003). Model-driven development. *Software, IEEE*, 20(5):14--18.

Murphy, G.; Notkin, D. & Sullivan, K. (2001). Software reflexion models: Bridging the gap between design and implementation. *Software Engineering, IEEE Transactions on*, 27(4):364--380.

Nuseibeh, B. & Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pp. 35--46. ACM.

Odell, J. (1998). *Advanced object-oriented analysis and design using UML*. Cambridge Univ Pr.

O'Halloran, C. (2000). Issues for the automatic generation of safety critical software. In *The Fifteenth IEEE International Conference on Automated Software Engineering, 2000. Proceedings ASE 2000*, pp. 277--280.

Oliveto, R.; Marcus, A. & Hayes, J. H. (2007). Software Artefact Traceability: the Never-Ending Challenge. *ICSM*, pp. 485--488.

OMG (2008a). Meta Object Facility (MOF) 2.0 Query View Transformation Specification.

OMG (2008b). Software & Systems Process Engineering Meta-Model Specification.

OMG (2009). OMG Unified Modeling Language TM (OMG UML), Infrastructure.

O'Neill, L. (1982). A retrospective on software engineering in design automation. In *Proceedings of the 19th Design Automation Conference*, pp. 10--14. IEEE Press.

Osborn, A. (1957). *Applied imagination: Principles and procedures of creative problem-solving*. C. Scribner's Sons; Rev. ed edition, Nova York, NY, terceira e edição.

Paige, R.; Olsen, G.; Kolovos, D.; Zschaler, S. & Power, C. (2008). Building model-driven engineering traceability classifications. In *Proc. 4th Workshop on Traceability, ECMDA, Berlim, Alemanha*.

Pinheiro, F. & Goguen, J. (1996). An object-oriented tool for tracing requirements. *IEEE SOFTWARE*, pp. 52--64.

Pinheiro, F. A. C. (2003). Requirements Traceability. *Perspectives on Software Requirements*, pp. 91--113.

Pohl, K. (1993). The Three Dimensions of Requirements Engineering. *CAiSE '93: Proceedings of Advanced Information Systems Engineering*, 19(3):275--292.

Pohl, K. (1996). PRO-ART: Enabling requirements pre-traceability. In *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, pp. 76 -- 84. Published by the IEEE Computer Society.

Pressman, R. (1995). *Engenharia de software*. São Paulo: Makron Books, 6 edição.

Ramesh, B.; Dwiggins, D.; DeVries, G. & Edwards, M. (2002). Towards requirements traceability models. In *Systems Engineering of Computer Based Systems, 1995., Proceedings of the 1995 International Symposium and Workshop on*, pp. 229--232. IEEE.

Ramesh, B. & Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 27(1):58--93.

Richardson, J. & Green, J. (2003). Traceability through automatic program generation. In *Proceedings of Second International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003)*.

Richardson, J. & Green, J. (2004). Automating traceability for generated software artifacts. In *Automated Software Engineering, 2004. Proceedings. 19th International Conference on*, pp. 24 -- 33. Published by the IEEE Computer Society.

Schmidt, D. (2006). Model-driven engineering. *IEEE computer*, 39(2):25--31.

Schwaber, C. (2006). The Changing Face Of Application Life-Cycle Management. Forrester Research, Inc.

Smithers, T.; Tang, M.; Tomes, N. & Intelligence, U. E. D. A. (1991). *The maintenance of design history in AI-based design*. University of Edinburgh, Department of Artificial Intelligence.

Spanoudakis, G. (2002). Plausible and adaptive requirement traceability structures. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pp. 135--142. ACM.

Spanoudakis, G. & Zisman, A. (2005). Software traceability: a roadmap. *Handbook of Software Engineering and Knowledge Engineering*, III:1--35.

Spanoudakis, G.; Zisman, A.; Pérez-Minana, E. & P (2004). Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, 72(2):105 -- 127.

Steinberg, D.; Budinsky, F.; Paternostro, M. & Merks, E. (2009). *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley Professional, 2a edição.

Tang, M. (1997). A knowledge-based architecture for intelligent design support. *The knowledge engineering review*, 12(4):387--406.

Toranzo, M.; Castro, J. & Mello, E. (2002). Uma proposta para melhorar o rastreamento de requisitos. In *V Workshop on Requirements Engineering*, pp. 194--209, Valencia.

Tryggeseth, E. & Nytro, I. (1997). Dynamic traceability links supported by a system architecture description. In *Software Maintenance, 1997. Proceedings., International Conference on*, pp. 180--187. IEEE.

Van Solingen, R. & Berghout, E. (1999). *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill Cambridge, UK.

Vazquez, C.; Simões, G. & Albert, R. (2010). *Análise de Pontos de Função - Medição, Estimativas e Gerenciamento de Projetos de Software*. Érica, 10 edição.

von Knethen, A. (2002). Automatic change support based on a trace model. In *Proceedings of the Traceability Workshop, Edinburgh, UK*.

West, M. (2002). Quality function deployment in software development. In *Tools and Techniques for Maintaining Traceability During Design, IEE Colloquium on*, p. 5. IET.

Winkler, S. & von Pilgrim, J. (2009). A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling*, pp. 1--37.

Wohlin, C.; Runeson, P. & Höst, M. (2000). *Experimentation in software engineering: an introduction.* Springer Netherlands.

Xiaoli, L. (2009). Process Oriented Analysis for Software Automation. *Computer-Aided Design*, pp. 1131--1133.

Zisman, A.; Spanoudakis, G. & Peres-Minana, E. (2003). Tracing software engineering artifacts. *on Software Engineering.*

Apêndice A

Pesquisas Realizadas

Este apêndice mostra a consolidação das "pesquisas aos elementos de Engenharia de Software" feitas através de Rastros pelos Gerentes de Projeto participantes do Estudo de Caso (vide Seção 4).

Toda pesquisa realizada através dos modelos de Informação da Rastreabilidade foi registrada em formulários, cujos modelos foram apresentados na Seção 4.1.1. O resultado foi consolidado junto aos Gerentes de Projetos. Pesquisas que foram descritas de forma distintas mas possuíam o mesmo valor foram uniformizadas numa descrição única. O resultado é mostrado abaixo:

- Quais são os códigos e motivadores diretamente afetados pela alteração do requisito?
- Quais são os códigos e motivadores indiretamente afetados pela alteração do requisito?
- Qual é a razão do não cumprimento do Planejado?
- O código que foi alterado é o previsto para ser alterado?
- Quais são os outros requisitos afetados pela alteração?
- "Quem?", "Quando?", "Por que?" da alteração
- Quais são os requisitos alocados para a entrega?
- Quais são os Requisitos que estão em estado de implementação?
- Quais são os requisitos já satisfeitos pelo código até o momento?
- Quais são os agrupamentos de questões que foram afetadas pela alteração?
- Quais são as razões de se retirar questões do escopo da versão?
- Quais questões foram canceladas e por quê?
- Quais questões estão em implementação?
- Quais questões estão implementadas?
- Quais questões foram incluídas na versão?
- Quais questões foram planejadas no início da versão?
- Quais questões estão planejadas para a versão?

- Quais questões foram retiradas do planejamento da versão?
- Quais são as razões da inclusão de questão no planejamento da versão?
- Quais são as razões para o cancelamento de questões?
- Quais são os requisitos afetados pela questão planejada?
- Quais são os requisitos afetados pela questão resolvida?

Cada uma das pesquisas identificadas é explicada na Seção A.1

A.1 Descrição das pesquisas

A.1.1 Quais são os códigos e motivadores diretamente afetados pela alteração do requisito?

Esta pesquisa é executada durante a tarefa de Controlar Alterações (descrita no Anexo A.4). Ela se faz necessária para se analisar o impacto da alteração sugerida.

O conhecimento do código afetado é a base para se estimar o esforço necessário para se executar a alteração proposta.

Os motivadores afetados são os requisitos ou questões que motivaram o desenvolvimento do código afetado. Esses requisitos ou questões são ligados ao código através de Rastros como "Satisfação do Requisito" e "Resolução da Questão".

A.1.2 Quais são os códigos e motivadores indiretamente afetados pela alteração do requisito?

Esta pesquisa é executada durante a tarefa de Controlar Alterações (descrita na Seção A.4). Ela se faz necessária para se analisar o impacto da alteração sugerida e complementa a pesquisa "Quais são os códigos e motivadores diretamente afetados pela alteração do requisito?".

Uma alteração pode afetar um código desenvolvimento para satisfazer mais de um requisito ou questão. A alteração no código pode afetar os outros requisitos e questões vinculados a ele. Por exemplo: a alteração exemplificada no Capítulo 1, "REQ001 - Controle de entrada e saída de mercadorias deve ocorrer manualmente e ser totalmente registrado no sistema" sofra a alteração para "REQ001 - Controle de entrada e saída de mercadorias **ou grupo de mercadorias** deve ocorrer manualmente e ser totalmente registrado no sistema" afeta o componente COMP001. Esse componente pode também

estar satisfazendo o requisito "REQ008 - o desempenho do sistema deve ser superior a 3 segundos por operação" que pode ser afetado pela alteração. E consequentemente todos os outros componentes vinculados ao REQ008 são indiretamente afetados pela alteração.

A.1.3 Qual é a razão do não cumprimento do Planejado?

A pesquisa é executada durante a tarefa "Verificar se os objetivos Estão Sendo Alcançados" e deve retornar a situação dos requisitos ou questões alocadas para uma determinada entrega.

Deve ser capaz de identificar os componentes que não estão terminados no prazo estipulado. E as razões do não cumprimento do planejado.

A.1.4 O código que foi alterado é o previsto para ser alterado?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 do Anexo A, "Controlar Alterações" descrita na Seção A.4 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5. A pesquisa deve retornar os códigos afetados pela alteração e os códigos que foram efetivamente alterados por conta da alteração.

A.1.5 Quais são os outros requisitos afetados pela alteração?

Esta pesquisa é executada durante a tarefa de Controlar Alterações (descrita na Seção A.4 do Anexo A). Ela complementa a pesquisa "Quais são os códigos e motivadores indiretamente afetados pela alteração do requisito?".

Como já descrito na pesquisa "Quais são os códigos e motivadores indiretamente afetados pela alteração do requisito?", uma alteração pode afetar um código desenvolvimento para satisfazer mais de um requisito. A alteração no código pode afetar os outros requisitos e questões vinculados a ele.

A.1.6 "Quem?", "Quando?", "Por que?" da alteração

A pesquisa é executada nas tarefas "Acompanhar o Projeto" descrita na Seção A.8, "Controlar Alterações" descrita na Seção A.4 do Anexo A. Deve retornar quem foi o responsável pela alteração, porque a alteração ocorre e quando ela ocorreu. A pes-

quisa é usada em situação de análise de alguma implementação que pode ter afetado negativamente parte do código.

A.1.7 Quais são os requisitos alocados para a entrega?

A pesquisa é executada principalmente na tarefa "Acompanhar o Projeto" descrita na Seção A.8 do Anexo A. Deve retornar os requisitos alocados para a entrega considerando as alterações no escopo ocorridas ao longo do projeto.

A.1.8 Quais são os Requisitos que estão em estado de implementação?

A pesquisa é executada na tarefa "Acompanhar o Projeto" descrita na Seção A.8. Deve retornar todo requisito que algum código que o satisfaz esteja em implementação no momento. Essa pesquisa é usada para acompanhar a evolução da satisfação dos requisitos.

A.1.9 Quais são os requisitos já satisfeitos pelo código até o momento?

A pesquisa é executada na tarefa "Acompanhar o Projeto" descrita na Seção A.8 do Anexo A. Deve retornar através dos Rastros todos os requisitos que os componentes que os satisfazem já estejam implementados. A pesquisa deve considerar as alterações de escopo ocorridas durante a execução do projeto.

A.1.10 Quais são os agrupamentos de questões que foram afetadas pela alteração?

Esta pesquisa é executada durante a tarefa de Controlar Alterações (descrita na Seção A.4 do Anexo A). O resultado da pesquisa é um grupo de questões que serão afetadas pela alteração. É usada numa análise de impacto de projetos de manutenção.

A.1.11 Quais são as razões de se retirar questões do escopo da versão?

A pesquisa é executada principalmente na tarefa "Acompanhar o Projeto" de manutenção descrita na Seção A.8 do Anexo A. Deve retornar as razões de uma questão não pertencer mais ao escopo. A pesquisa é feita sobre os Rastros de alteração do escopo.

A.1.12 Quais questões foram canceladas e por quê?

A pesquisa é executada principalmente na tarefa "Acompanhar o Projeto" de manutenção descrita na Seção A.8 do Anexo A. O resultado da pesquisa mostra as questões que formavam o escopo do projeto de manutenção e tiveram sua execução cancelada. A utilidade da pesquisa está em momentos de investigação das causas das mudanças de escopo do projeto.

A.1.13 Quais questões estão em implementação?

A pesquisa é executada principalmente na tarefa "Acompanhar o Projeto" de manutenção descrita na Seção A.8 do Anexo A. O resultado da pesquisa apresenta as questões que estão em andamento no momento. O gerente de projeto utiliza o resultado em conjunto com as tarefas atribuídas a partir do cronograma e compara os resultados.

A.1.14 Quais questões estão implementadas?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8, ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A pesquisa deve retornar os códigos afetados pela alteração. O resultado da pesquisa apresenta todas as questões já implementadas para uma determinada versão do produto. A razão da pesquisa ser realizada está na necessidade do gerente de projetos acompanhar o estado atual do projeto.

A.1.15 Quais questões foram incluídas na versão?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A pesquisa deve retornar os códigos afetados pela alteração. O resultado da pesquisa aponta todas as questões que foram incluídas no projeto após o escopo ter sido fechado. A razão da pesquisa ser realizada está na necessidade de acompanhar junto ao cliente as alterações e impactos no projeto caso haja alguma dúvida.

A.1.16 Quais questões foram planejadas no início da versão?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A pesquisa deve retornar as questões planejadas para a versão sem considerar as alterações ocorridas ao longo do projeto.

A.1.17 Quais questões estão planejadas para a versão?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A pesquisa deve retornar as questões planejadas para a versão considerando as alterações ocorridas ao longo do projeto.

A.1.18 Quais questões foram retiradas do planejamento da versão?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A pesquisa deve retornar as questões planejadas para a versão, mas que foram retiradas ao longo do projeto.

A.1.19 Quais são as razões da inclusão de questão no planejamento da versão?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A pesquisa deve retornar as questões que não foram planejadas para a versão, mas que foram incluídas ao longo do projeto.

A.1.20 Quais são as razões para o cancelamento de questões?

A pesquisa é executada durante a tarefa de "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. O resultado da pesquisa mostra as questões que formavam o escopo do projeto de manutenção e tiveram sua execução cancelada. A utilidade da pesquisa está em momentos de investigação das causas das mudanças de escopo do projeto.

A.1.21 Quais são os requisitos afetados pela questão planejada?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A necessidade da pesquisa está na analise de impacto da nova versão no produto.

A pesquisa retorna todos os requisitos que foram implementados no produto e que podem ser alterados pela implementação da questão. Esses requisitos são ligados ao código através de Rastros como "Satisfação do Requisito".

A.1.22 Quais são os requisitos afetados pela questão resolvida?

A pesquisa é executada durante a tarefa de "Acompanhar o Projeto" descrita na Seção A.8 ou "Verificar se os objetivos Estão Sendo Alcançados" descrita na Seção A.5 do Anexo A. A necessidade da pesquisa está na analise de impacto real da nova versão do produto.

A pesquisa retorna todos os requisitos que foram implementados no produto e que foram alterados pela implementação da questão.

Anexo A

Descrição das Tarefas do Estudo de Caso

A.1 Identificar e Cadastrar os Requisitos e os Casos de Uso

Executor Principal: Analista de Requisitos

Objetivo: Identificar e cadastrar os requisitos e casos de uso identificados com informações suficientes para controle e acompanhamento.

Entradas:

- DOCVISAO - Documento de Visão
- Entrevistas com Usuários-Chave, Representantes do Cliente e/ou Especialistas

Saídas:

- CDU - Especificação de Casos de Uso
- CONRAS - Controle de Rastreabilidade
- Diagrama de Casos de Uso
- DOCVISAO - Documento de Visão
- Requisitos, Componentes e Casos de Uso

Descrição: O propósito desta atividade é mapear os envolvidos ou grupos de envolvidos que detêm o conhecimento necessário para o desenvolvimento ou manutenção do software e identificar quais são as suas necessidades em relação ao sistema. As informações coletadas dos interessados servem para gerar uma lista de requisitos do software.

Esta atividade é de responsabilidade do Analista de Sistemas e é realizada em conjunto com representantes do cliente, usuários-chave, analistas de negócio e outros especialistas da área de aplicação, além do responsável comercial, quando necessários.

Para identificar os requisitos do software durante seu desenvolvimento, são considerados o “Documento de Visão - DOCVISAO” e as informações obtidas durante as entrevistas com os usuários/representantes/especialistas, os requisitos estatutários e regulamentares aplicáveis ao software e os itens identificados e estabelecidos em acordo com o cliente (produtos a serem entregues, tecnologia utilizada, prazos de entrega, forma de entrega, etc.), que devem ter sido repassados pelo responsável comercial à equipe do software. Técnicas para a elicitação destes requisitos a partir das várias fontes de informação existentes estão descritas no "Guia de Elicitação de Requisitos".

Também são levadas em consideração as informações obtidas a partir de trabalhos de consultorias especializadas desenvolvidas no cliente, bem como versões já existentes do software.

Caso o cliente forneça algum produto para ser incorporado ao software ou para ser consultado com a finalidade de extrair do mesmo informações relevantes ao desenvolvimento do software em questão, devem ser seguidas as diretrizes estabelecidas na atividade Controlar os Produtos Fornecidos pelo Cliente da disciplina de Gestão do Software.

Ao final da presente atividade, deve-se ter a lista de requisitos funcionais e não-funcionais identificados para o software, além da identificação preliminar e cadastro dos casos de uso que atendam aos requisitos levantados.

Para melhor entendimento, a seguir são listadas algumas considerações comparativas entre requisitos e casos de uso:

- um requisito é uma necessidade que o sistema deve atender;
- um caso de uso é a especificação detalhada de uma funcionalidade do sistema;
- um caso de uso pode atender a vários requisitos;
- um requisito pode ser atendido por vários casos de uso.

Os requisitos devem ser classificados em funcionais ou não-funcionais.

Requisitos funcionais descrevem as funções que o produto deverá realizar em benefício dos usuários.

Requisitos não-funcionais tratam de características do produto e incluem os requisitos de desempenho, requisitos lógicos de dados, requisitos de qualidade do software, requisitos estatutários e regulamentares.

Após encerrar essa atividade, o responsável deverá atualizar a situação dos respectivos requisitos e/ou casos de uso no SCARAB, para que estejam de acordo com a situação do projeto/produto naquele momento.

Etapas:

1. Identificar e listar os atores.
2. Identificar e listar os requisitos funcionais.
3. Identificar e listar os requisitos não-funcionais.
4. Identificar os casos de uso.

5. Elaborar o Diagrama de Casos de Uso.

6. Cadastrar os requisitos e casos de uso identificados.

7. Elaborar/atualizar a matriz de rastreabilidade.

Depois de cadastrados os requisitos e casos de uso, a matriz de rastreabilidade de requisitos (funcionais e não-funcionais) e casos de uso, presente no documento "CONRAS - Controle de Rastreabilidade" deve ser elaborada.

Essa matriz identifica a rastreabilidade entre: (a) requisitos do cliente (funcionais e não funcionais no DOCVISAO) e requisitos do produto (funcionais e não funcionais na ferramenta); (b) requisitos do produto (funcionais e não funcionais na ferramenta) e casos de uso; (c) casos de uso e componentes (identificados posteriormente, na atividade Elaborar a Arquitetura ou Avaliar uma Arquitetura Preliminar).

Todos os requisitos funcionais devem ser cobertos por, pelo menos, um caso de uso. E cada caso de uso deverá cobrir um ou vários requisitos levantados e cadastrados.

Análise da Atividade: Essa tarefa na etapa "Elaborar/atualizar a matriz de rastreabilidade." gera diversos rastros entre os requisitos e casos de uso. Esses são rastros importantes para a garantia que o projeto de software irá implementar todos os requisitos identificados. Para o Estudo de Caso foi selecionado para ser monitorado apenas os rastros entre Requisitos e Código. Portanto esses rastros aqui identificados não serão avaliados. Mas a tarefa é importante para o Estudo de Caso devido ao fato que os Requisitos levantados devem ser registrados no método re rastreabilidade usado.

A.2 Analisar a Questão

Executor Principal: Analista de Sistemas

Objetivo: Entender os requisitos da questão técnica em análise e calcular seu tamanho e esforço relacionado.

Entradas:

- Questões Técnicas

Saídas:

- Esforço Estimado
- Planilha de Tamanho do Projeto
- Questões Técnicas

- SOLUCAOQT - Projeto de Solução de Questão Técnica

Descrição:

Através desta atividade, os requisitos da questão são melhor entendidos e o tamanho estimado da manutenção é calculado. Caso as informações cadastradas no SCARAB não sejam suficientes para a análise da questão, o Implementador pode solicitar maiores informações sobre os requisitos e necessidades da questão para:

- o atendente ou pessoa que registrou a necessidade na ferramenta de "Controle de Manutenções";
- o contato no cliente, que está cadastrado na ferramenta de "Controle de Manutenções".

Para o caso de uma manutenção corretiva, é recomendável que o Implementador tente reproduzir ou verificar o problema, ainda em ambiente de desenvolvimento.

Após a análise, caso fique decidido que a solução do problema não será implementada:

- o atendente responde ao cliente, se necessário;
- o implementador e/ou atendente atualizam os dados do atendimento;
- a questão é encerrada.

Após análise, se ficar definido que a solução definitiva do problema será implementada, o atendente informa ao cliente, se necessário, e aguarda a informação do término pelo responsável pela distribuição.

Uma questão poderá afetar mais de uma versão do produto e, como consequência, poderá ter que ser tratada em várias versões existentes do produto. Pelo fato de sempre se trabalhar com o cronograma da próxima versão do produto a ser liberada, o tratamento nas versões anteriores, caso realmente seja necessário, somente poderá ocorrer via hotfix. A correção definitiva ocorrerá somente para a próxima versão do produto, de acordo com a necessidade levantada durante a análise da questão.

A abrangência e o impacto da questão devem ser registradas na análise da questão. Deve-se observar também a urgência ou relevância de correção. Na análise, deve-se também levar em consideração outros produtos ou sistemas que poderão ser afetados ao se tratar a questão.

O esforço para implementação da correção da questão técnica deverá ser calculado a partir do tamanho total da questão, em UMMs, e da taxa de produção para

resolução de questões técnicas da equipe. Para esse cálculo, é indiferente o fato do item estar relacionado com a inserção de um novo requisito/CDU/componente ou um item relacionado com alteração, correção do software ou exclusão de funcionalidades.

Ao final desta atividade, a questão técnica a ser tratada estará suficientemente conhecida e será possível determinar se será implementada imediatamente, em uma versão futura, de acordo com execução das atividades Planejar Versão da Produção e Planejar Linha de Base, ou simplesmente não será implementada.

Etapas:

1. Entender a questão técnica.
2. Solicitar maiores informações sobre os requisitos e necessidades, caso necessário.
3. Tentar reproduzir ou verificar o problema.
4. Calcular o tamanho e o esforço estimado para tratar a questão técnica.

Na última etapa "Calcular o tamanho e o esforço estimado para tratar a questão técnica" a questão deve ser registrada na ferramenta de rastreabilidade e vinculada ao projeto e compoentes onde a solução da questão será impelmentada.

Análise da Atividade:

Do ponto de vista da Rastreabilidade essa tarefa possui a mesma função da tarefa "Identificar e Cadastrar os Requisitos e os Casos de Uso". Deve-se registrar as questões e alocá-las a componentes do software.

A.3 Especificar Casos de uso e Requisitos

Executor Principal: Analista de Requisitos

Objetivo:

Uma vez cadastrados e priorizados, os requisitos do produto (funcionais e não-funcionais) devem ser especificados até um nível de detalhe suficiente para o planejamento da próxima iteração.

Os requisitos funcionais são especificados através dos casos de uso nos documentos de "CDU - Especificação de Casos de Uso".

Normalmente, ao final desta atividade, os casos de uso são especificados no Nível Básico, segundo orientações descritas no ANEXO D - NÍVEIS DE CASOS DE USO

Os requisitos não-funcionais são especificados na própria ferramenta de controle de requisitos e casos de uso. Entretanto, se for necessário um maior detalhamento ou

uma especificação muito elaborada, pode-se utilizar o documento RNF - Especificões de Requisitos Não-Funcionais para facilitar a documentação.

Esta atividade é de responsabilidade dos Analistas de Sistemas e pode envolver os Usuários-Chave e Analistas de Negócio. As reuniões feitas com o intuito de realizar esta atividade devem ser registradas em NDR - Notas de Reunião.

Na execução desta atividade, pode ser necessário o uso de técnicas de elicitação de requisitos como, por exemplo, prototipação, entrevistas, reuniões JAD entre outras. Para orientar a seleção de qual técnica deve ser utilizada, foi elaborado o Guia de Elicitação de Requisitos que apresenta as principais técnicas e seus pontos fortes e fracos.

Após encerrar essa atividade, o responsável deverá atualizar a situação dos respectivos requisitos e/ou casos de uso no SCARAB, para que estejam de acordo com a situação do projeto/produto naquele momento.

Entradas:

1. CDU - Especificação de Casos de Uso
2. CONRAS - Controle de Rastreabilidade
3. Diagrama de Casos de Uso
4. DOCVISAO - Documento de Visão

Saídas:

- CDU - Especificação de Casos de Uso
- Diagrama de Casos de Uso
- Requisitos, Componentes e Casos de Uso
- RNF - Especificações de Requisitos Não-Funcionais

Descrição:

Uma vez cadastrados e priorizados, os requisitos do produto (funcionais e não-funcionais) devem ser especificados até um nível de detalhe suficiente para o planejamento da próxima iteração.

Os requisitos funcionais são especificados através dos casos de uso nos documentos de "CDU - Especificação de Casos de Uso".

Normalmente, ao final desta atividade, os casos de uso são especificados no Nível Básico, segundo orientações descritas no ANEXO D - NÍVEIS DE CASOS DE USO

Os requisitos não-funcionais são especificados na própria ferramenta de controle de requisitos e casos de uso. Entretanto, se for necessário um maior detalhamento ou uma especificação muito elaborada, pode-se utilizar o documento RNF - Especificações de Requisitos Não-Funcionais para facilitar a documentação.

Esta atividade é de responsabilidade dos Analistas de Sistemas e pode envolver os Usuários-Chave e Analistas de Negócio. As reuniões feitas com o intuito de realizar esta atividade devem ser registradas em NDR - Notas de Reunião.

Na execução desta atividade, pode ser necessário o uso de técnicas de elicitação de requisitos como, por exemplo, prototipação, entrevistas, reuniões JAD entre outras. Para orientar a seleção de qual técnica deve ser utilizada, foi elaborado o Guia de Elicitação de Requisitos que apresenta as principais técnicas e seus pontos fortes e fracos.

Após encerrar essa atividade, o responsável deverá atualizar a situação dos respectivos requisitos e/ou casos de uso no SCARAB, para que estejam de acordo com a situação do projeto/produto naquele momento.

Etapas:

- Marcar as reuniões de detalhamento dos requisitos.
- Especificar os requisitos funcionais através de casos de uso.
- Especificar os requisitos não-funcionais.

Para cada novo requisito identificado deve-se registrar o novo elemento e seu rastro com os outros elementos.

Análise da Atividade:

Nessa tarefa se encontra novos requisitos associados com requisitos já existentes. Os novos requisitos devem ser registrados no Modelo de informação de Rastreabilidade escolhido com as associações adequadas com os outros requisitos.

A.4 Controlar Alterações

Executor Principal: Gerente do Projeto

Objetivo: Analisar e providenciar o devido tratamento para qualquer tipo de alteração solicitada e/ou ocorrida no projeto.

Entradas:

- Requisição de Alterações

- Controle Financeiro Atualizado do Projeto ou da Versão de Produção
- CRONOGRAMA - Cronograma
- PLANSOFT - Plano do Software
- CONRAS

Saídas:

- Alterações
- CRONOGRAMA - Cronograma
- PLANSOFT - Plano do Software
- CONRAS

Descrição:

O projeto no qual se está trabalhando pode sofrer alterações por diversos motivos. Dentre os vários motivos, pode-se citar os seguintes:

- alterações no contexto do projeto ou produto;
- descoberta de defeitos, inadequações e insuficiências nos requisitos originais;
- falta de detalhamento suficiente nos requisitos originais;
- melhor entendimento do problema por parte dos usuários ou dos implementadores;
- fatores externos (por exemplo: mudança de legislação; alterações tecnológicas, gerenciais, legais e políticas, etc.);
- inadequação, incompatibilidade ou insuficiência de recursos materiais e humanos;
- alterações nos processos de negócio;
- alteração do planejamento das atividades.

Considerando que as alterações podem ter impacto nos prazos e custos planejados, tais alterações devem acontecer de maneira controlada e, para isso, os passos apresentados a seguir devem ser executados.

Há um anexo com parâmetros para controle de alterações, que apresenta um quadro-resumo dos critérios e parâmetros para a visualização geral dos passos a serem seguidos para a realização das alterações.

Etapas:

1. Receber uma requisição de alteração.
2. Registrar a alteração.
3. Analisar o impacto das alterações.
4. Aprovar a alteração.
5. Executar a alteração.
6. Atualizar documentação impactada.
7. Encerrar a alteração.
8. Comunicar aos envolvidos.

Análise da Atividade:

Essa atividade utiliza o Modelo de Informação da Rastreabilidade para identificar o impacto da alteração no projeto.

A.5 Verificar Se Os objetivos Estão Sendo Alcançados

Executor Principal: Gerente do Projeto

Objetivo: Verificar se os objetivos definidos para o projeto estão sendo alcançados.

Entradas:

- AMP - Acompanhamento de Métricas do Projeto
- MP - Modelo Preditivo
- PLANSOFT - Plano do Software
- Processo Customizado para o Projeto
- CONRAS - Controle de Rastreabilidade

Saídas:

- AMP - Acompanhamento de Métricas do Projeto
- CRONOGRAMA - Cronograma
- PLANSOFT - Plano do Software

Descrição:

Com auxílio de um representante da área GDQ, o responsável por esta atividade deve verificar se os objetivos para cada subprocesso do projeto estão sendo cumpridos.

Etapas:

1. Revisar o desempenho dos subprocessos e sua capacidade de alcançar os objetivos
2. Usar os modelos preditivos para estimar o progresso dos subprocessos do projeto
3. Identificar e gerenciar os riscos associados com o alcance dos objetivos do projeto
4. Determinar e documentar ações necessárias para resolver os problemas identificados

Análise da Atividade:

Na atividade se utiliza a rastreabilidade para ajudar a avaliar se os objetivos do projeto estão sendo alcançados.

A.6 Implementar Componentes

Executor Principal: Programador

Objetivo: Codificação de um componente do software.

Entradas:

- ARQUITETURA - Documento de Arquitetura do Software
- CDU - Especificação de Casos de Uso
- COMP - Especificação de Componente
- COMPONENTES - Componentes do Software
- CONRAS - Controle de Rastreabilidade
- Demais Diagramas UML

- Diagrama de Classes
- Dicionário de Dados
- Guia de Codificação do Produto
- Interfaces de Usuário
- Modelo de Dados
- Questões Técnicas
- Requisitos, Componentes e Casos de Uso
- RNF - Especificações de Requisitos Não-Funcionais
- Código Fonte

Saídas:

- Componentes implementados
- Código Fonte

Descrição:

Nesta atividade, devem ser implementados um componente do software. Esta implementação deverá seguir as orientações técnicas dos documentos ARQUITETURA, COMPONENTES, CDU, COMP e dos modelos e digramas do sistema. Todas as Decisões Arquiteturais existentes no documento ARQUITETURA devem ser validadas, se possível.

A sequência e o procedimento de implementação e os critérios de aceite dos componentes do software deverão ser respeitados nesta atividade. Caso não exista um padrão de codificação definido especificamente para o projeto, pode-se utilizar o Guia de Codificação como padrão.

Caso esta atividade esteja sendo também executada na fase de Elaboração, neste momento, pode ser iniciada também a atividade Planejar os Testes (da disciplina de "Testes"), que tem o objetivo de elaborar o planejamento dos testes de integração - testes funcionais e não-funcionais. Em seguida, tais testes serão projetados na atividade Projetar os Testes (da disciplina de Testes).

Etapas:

1. Implementar Componente

2. Verificar Implementação

3. Liberar os componentes

A etapa "Liberar os componentes" que é a que interessa a rastreabilidade é descrito como a "disponibilização do código para os demais desenvolvedores na ferramenta de Gestão de Configuração, seguindo a estratégia de liberação descrito no documento de CONFIGURAÇÃO."

A estratégia de liberação indica que deve-se escrever no comentário da liberação na ferramenta de gestão de configuração o código do componente, requisitos, questões, funcionalidades que motivaram a implementação.

Análise da Atividade:

Esta atividade gera um rastro durante a etapa "Liberar os componentes" que liga o código fonte salvo na ferramenta de gestão de configuração ao requisito que a motivou. Em respostas aos questionários se percebeu que a empresa acredita que este seja um rastro importante para a garantia que todo requisito tenha sido implementado e toda a implementação tem como base algum requisito. Além disso, utiliza-se essa informação na análise de impacto de uma solicitação de mudança.

A.7 Implementar a Questão Técnica

Executor Principal: Programador

Objetivo:

Implementar a questão técnica e testar (teste de unidade), corrigindo os erros dessa implementação.

Entradas:

- ARQUITETURA - Documento de Arquitetura do Software
- CDU - Especificação de Casos de Uso
- COMP - Especificação de Componente
- COMPONENTES - Componentes do Software
- CONRAS - Controle de Rastreabilidade
- Demais Diagramas UML
- Diagrama de Classes

- Diagramas de Seqüência
- Dicionário de Dados
- Guia de Codificação do Produto
- Interfaces de Usuário
- Modelo de Dados
- Questões Técnicas
- RNF - Especificações de Requisitos Não-Funcionais

Saídas:

- Questões Técnicas

Descrição:

Nesta atividade, as questões técnicas1 , junto com os respectivos testes de unidade, são implementados. Em seguida, os testes de unidade são efetuados com a finalidade de descobrir erros de codificação e realiza-se a correção antes das atividades de integração.

A questão técnica é implementada de acordo com os resultados da análise e projeto da questão técnica. O Implementador também deve seguir as orientações do Documento de Arquitetura e os padrões definidos no Guia de Codificação, caso este tenha sido elaborado. Todas as Decisões Arquiteturais existentes no Documento de Arquitetura devem ser validadas se possível.

Caso a questão técnica tenha que ser tratada em mais de uma versão de um produto, podem ocorrer impactos diferentes nas correções das diferentes versões. Recomenda-se, nesse caso documentá-las no SCARAB, indicando tais diferenças. Deve-se ter o cuidado de manter a correspondência correta das versões, uma vez que isso é controlado pelo cronograma.

A criação, codificação e execução dos testes de unidade são de responsabilidade do Implementador. Dessa forma, os testes de unidade são criados, codificados e, por fim, realizados pelo próprio Implementador nas unidades de código por ele construídas. O código é então corrigido, caso necessário. Para questões técnicas, é possível reutilizar testes de unidade já existentes para a implementação em questão, seja integralmente ou com algumas correções e adaptações. Os testes de unidade podem seguir o Guia de Testes, caso este tenha sido elaborado, e podem ser criados e executados com ajuda de ferramentas de testes específicas para tal. Pelo menos os testes manuais devem

ser realizados antes da disponibilização do código para os demais desenvolvedores na ferramenta de Gestão de Configuração. Não é necessário gerar uma documentação para relatar os testes de unidade.

A atividade termina quando o Implementador declarar o fim dos testes de unidade da parte que estiver desenvolvendo (componente, caso de uso, etc) e liberar o código para a integração.

Etapas:

1. Implementar a questão técnica.
2. Criar os testes de unidade.
3. Realizar os testes de unidade.
4. Corrigir os erros encontrados.
5. Liberar o código.

A.8 Acompanhar o Projeto

Executor Principal: Gerente do Projeto

Objetivo: Acompanhar o andamento da execução do planejamento do projeto.

Entradas:

- Alterações
- Controle Financeiro Atualizado do Projeto ou da Versão de Produção
- CRONOGRAMA - Cronograma
- Evoluções
- GUPGQ - Guia de Utilização do Processo e Garantia da Qualidade
- Não-conformidades e observações
- NDRAC - Notas de Reunião de Análise Crítica
- PLANSOFT/Aba MÉTRICAS - Métricas Adicionais
- PLANSOFT - Plano do Software
- Processo Customizado para o Projeto

- CONRAS - Controle de Rastreabilidade
- Questões Técnicas
- Requisitos, Componentes e Casos de Uso
- To-do List
- WBSCOMP - Estrutura Analítica do Projeto por Componentes

Saídas:

- Controle Financeiro Atualizado do Projeto ou da Versão de Produção
- CRONOGRAMA - Cronograma
- CONRAS- Controle de Rastreabilidade
- GUPGQ - Guia de Utilização do Processo e Garantia da Qualidade
- NDRAC - Notas de Reunião de Análise Crítica
- PLANSOFT/Aba SITUAÇÕES ESPECIAIS - Histórico de Situações Especiais
- PLANSOFT - Plano do Software
- Processo Customizado para o Projeto
- To-do List
- WBSCOMP - Estrutura Analítica do Projeto por Componentes

Descrição:

Ao longo de todo o projeto são realizadas atividades de acompanhamento técnico e gerencial, através das quais o trabalho de execução do projeto é verificado, em relação ao planejado.

Esse acompanhamento compreende um conjunto de esforços para o efetivo monitoramento, registro e controle das atividades realizadas, e seus respectivos progressos alcançados.

O Gerente do Projeto é responsável por definir e registrar no "Plano do Software - PLANSOFT (Comunic e Reuniões)" a periodicidade e forma com que fará o acompanhamento da execução do planejamento junto aos demais envolvidos (implementadores, analistas, equipe de testes, gerência sênior, etc.).

Durante esta atividade, são permanentemente envolvidos e mobilizados todos os responsáveis pelo desempenho e suporte à execução do planejamento e os demais envolvidos (stakeholders) que possam influenciar no custo, cronograma e qualidade do projeto.

A Gerência Sênior pode também participar e se envolver, de acordo com a conveniência, no acompanhamento gerencial da evolução da execução do planejamento realizado.

Com o acompanhamento gerencial, o Gerente do Projeto visa controlar e impedir:

- desvio das medidas e métricas estimadas inicialmente;
- não-atendimento dos compromissos, internos ou externos;
- mudanças significativas nas situações dos riscos;
- questões relativas à representação e envolvimento dos envolvidos.

Etapas:

1. Acompanhar e manter o envolvimento dos stakeholders do projeto/produto (envolvidos e interessados).
2. Controlar as horas trabalhadas pelos participantes da equipe do software.
3. Controlar a evolução do planejamento.
4. Executar simulações com Modelos Preditivos (Modelos de Desempenho de Processo).
5. Identificar alterações do planejamento.
6. Garantir a integridade do ambiente de trabalho.
7. Revisar escopo, objetivos e equipe do projeto ou da versão de produção.
8. Acompanhar as ações referentes aos riscos.
9. Acompanhar a execução financeira (receitas e despesas) da execução do planejamento do produto de software.
10. Acompanhar a evolução e solução das questões técnicas, não-conformidades, observações e alterações.
11. Monitorar as ações identificadas nas análises de causa.

12. Registrar o resultado do acompanhamento.

13. Registrar e acompanhar pendências.

A.9 Projetar Caso de Uso

Executor Principal: Projetista de Sistemas

Objetivo: Elaborar os modelos necessários para cada um dos casos de uso do projeto de modo a permitir a sua implementação.

Entradas:

- ARQUITETURA - Documento de Arquitetura do Software
- CDU - Especificação de Casos de Uso
- CENARIO - Cenário de Caso de Uso
- COMP - Especificação de Componente
- COMPONENTES - Componentes do Software
- CONRAS - Controle de Rastreabilidade
- Diagrama de Casos de Uso
- Guia de Projeto de Banco de Dados
- Modelos Preliminares do Software
- Protótipo
- Requisitos, Componentes e Casos de Uso
- RNF - Especificações de Requisitos Não-Funcionais

Saídas:

- CDU - Especificação de Casos de Uso
- CENARIO - Cenário de Caso de Uso
- CONRAS - Controle de Rastreabilidade
- Demais Diagramas UML

- Diagramas de Seqüência
- Dicionário de Dados
- Modelo de Dados
- Requisitos, Componentes e Casos de Uso
- RNF - Especificações de Requisitos Não-Funcionais

Descrição:

O projeto do software tem o objetivo de especificar a implementação dos casos de uso do software, refinando os modelos até um nível de detalhamento suficiente para o trabalho do Implementador. Esse refinamento é feito pelos Projetistas de Sistema, de Banco de Dados e de Interface de Usuário. Ao final das execuções desta atividade, os modelos e especificações do software estarão devidamente detalhados para permitir os trabalhos de implementação.

A estrutura a ser utilizada na ferramenta CASE é importante para permitir a rastreabilidade dos casos de uso até as classes modeladas e, conseqüentemente, até o código. Assim, qualquer que seja a ferramenta de modelagem utilizada, devem ser gerados modelos para cada caso de uso do software. Dessa forma, os vários modelos de classes e de seqüência a serem gerados deverão estar organizados por casos de uso.

No entanto, caso a ferramenta selecionada para a execução do projeto não permita esta divisão dos modelos por casos de uso, o Gerente de Projeto ou Coordenador de Produto deverá incluir no documento de Especificação de Casos de Uso a parte do modelo ou descrição das classes que o realizam.

Após encerrar essa atividade, o responsável deverá atualizar a situação dos respectivos requisitos e/ou casos de uso no SCARAB, para que estejam de acordo com a situação do projeto/produto naquele momento.

Etapas:

1. Projetar as classes/entidades.
2. Alocar Requisito.
3. Projetar os fluxos de dados (Análise Estruturada).
4. Projetar os fluxos das rotinas batch.
5. Projetar o banco de dados.
6. Projetar as interfaces de usuário.

7. Projetar a distribuição do software.

Anexo B

Notas das Entrevistas

B.1 Entrevista Gerente de projeto 1

Data da Entrevista: 26/11/2010

Qual é sua função hoje na empresa?

Sou Gerente de projetos do P3 e do P4. Além disso, gerencio os recursos da área Plataforma Alta e os procedimentos da Informática Interna.

Quais são os processos utilizados em seus projetos?

Tanto no P4 como no P3 utilizamos o processo de manutenção.

Os processos estão bastante customizados?

Não. A aderência é grande. Não me lembro direito agora, mas no P3 está entre 10 e 17 porcento de customização. No P4 a customização é maior. Algo em torno de 30 porcento.

Você utiliza alguma forma de Rastreabilidade nos projetos?

Sim. Usamos o CONRAS.

Nos dois projetos?

Sim.

Usam de forma diferenciada do processo?

Não. Nos dois casos usamos exatamente como o processo pede.

Em quais atividades você utiliza o CONRAS?

No momento em que as questões estão sendo analisadas, preenchemos o CONRAS ligando questão ao código. Então distribuo as questões para desenvolvedores que conhecem os códigos afetados. Toda alteração eu analiso a CONRAS para replanejar as questões e os desenvolvedores.

No acompanhamento semanal eu olho a CONRAS para verificar o que deveria ser feito e avaliar o que já foi feito e o que falta ser feito.

Todas essas informações são extraídas da CONRAS?

não. Lá eu sei apenas o que foi planejado para ser feito. As outras informações eu utilizo os registros do CVS e as atividades com horas apropriadas.

Mais alguma atividade?

Uma vez por mês eu tenho que verificar se o projeto está de acordo com os objetivos corporativos. Eu verifico se as alterações e as questões estão sendo desenvolvidas de acordo com os requisitos da versão. Uso a CONRAS para isso.

B.2 Entrevista Gerente de projeto 2

Data da Entrevista: 29/11/2010

Qual é sua função hoje na empresa?

Sou Gerente de projetos.

Quais são os projetos que você está gerenciando?

Atualmente estou gerenciando o P1, 0 P2 e outros.

Para os projetos P1 e P2, quais são os processos utilizados em seus projetos?

O P2 faz parte de um contrato maior que é um desenvolvimento em formato de fábrica de software. Por isso estamos utilizando o processo padrão de fábrica de software. Já o P1 é um projeto de ciclo completo.

Qual é o processo para o P1?

É o PCVPS.

Os processos estão bastante customizados?

Não. Os projetos são bastante típicos.

Você utiliza alguma forma de Rastreabilidade nos projetos?

Sim. É obrigação nossa usar o que está descrito nos processos

Qual é o artefato que cuida da rastreabilidade?

Usamos o CONRAS.

Nos dois projetos?

Sim.

Usam de forma diferenciada do processo?

Não.

Em quais atividades você utiliza o CONRAS?

Os analistas e desenvolvedores preenchem a CONRAS quando estão registrando os casos de uso.

Você sabe em qual atividade isso ocorre?

Identificar e Cadastrar os Requisitos e Casos de Uso e Especificar Casos de uso e Requisitos.

Mais alguma atividade?

Eu consulto os rastros no Controlar Alterações ou quando verifico os Objetivos ou no acompanhamento do projeto.

A planilha CONRAS fornece todas as informações necessárias?

não. Eu converso com as pessoas e verifico em outros documentos. O PLANSOFT, CONFIGURAÇAO... outros.

B.3 Entrevista com desenvolvedor 1

Data da Entrevista: 30/11/2010

Qual é sua função hoje na empresa?

Analista e desenvolvedor do P3.

Quais são os projetos que você está participando?

P3

Você conhece o processo utilizado em seu projeto?

Claro. Sou auditado nele todo mês.

Qual é?

É o processo de manutenção.

Você utiliza alguma ferramenta de Rastreabilidade?

Nós usamos a planilha CONRAS. Não se foi isso que você perguntou.

Foi sim. Como você usa a CONRAS?

Quando resolvo uma questão eu indico os fontes que foram alterados.

Que atividade é essa?

Implementar questão.

Só nesse caso?

Não. No Analisar a questão também.

B.4 Entrevista com desenvolvedor 2

Data da Entrevista: 1/12/2010

Qual é sua função hoje na empresa?

Desenvolvedor analista.

Quais são os projetos que você participa?

P3 e P4 e eu ainda ajudo em problemas pontuais em outros projetos.

Você conhece o processo utilizado em seu projeto?

O de manutenção.

Você utiliza alguma ferramenta de Rastreabilidade?

CONRAS

Como você usa a CONRAS?

Durante a análise, eu ligo a questão ao componente que deve ser afetado. E quando eu desenvolvo, eu ligo o código alterado à questão.

Que atividades são essas?

Analisar e implementar questão.

B.5 Entrevista com desenvolvedor 3

Data da Entrevista: 1/12/2010

Qual é sua função hoje na empresa?

Desenvolvedor Java.

Quais são os projetos que você participa?

P1 e outros.

Você conhece o processo utilizado em seu projeto?

conheço. É o processo de fábrica de software.

Você utiliza alguma ferramenta de Rastreabilidade?

A rastreabilidade é feita através da CONRAS.

Como você usa a CONRAS?

Quando termino de escrever o código. quando faço o check out no CVS eu também escrevo no CONRAS o vínculo de qual requisito foi implementado no código.

Qual o nome da atividade que você registra na CONRAS?

Implementar caso de uso.

B.6 Entrevista com desenvolvedor 4

Data da Entrevista: 1/12/2010

Qual é sua função hoje na empresa?

Projetista e desenvolvedor.

Quais são os projetos que você participa?

Atualmente estou apenas no P1 e outros

Você conhece o processo utilizado em seu projeto?

Fábrica de software.

Você utiliza alguma ferramenta de Rastreabilidade?

Ferramenta, você diz é o que?

Planilha é uma ferramenta

Usamos sim. A CONRAS.

Como você usa a CONRAS?

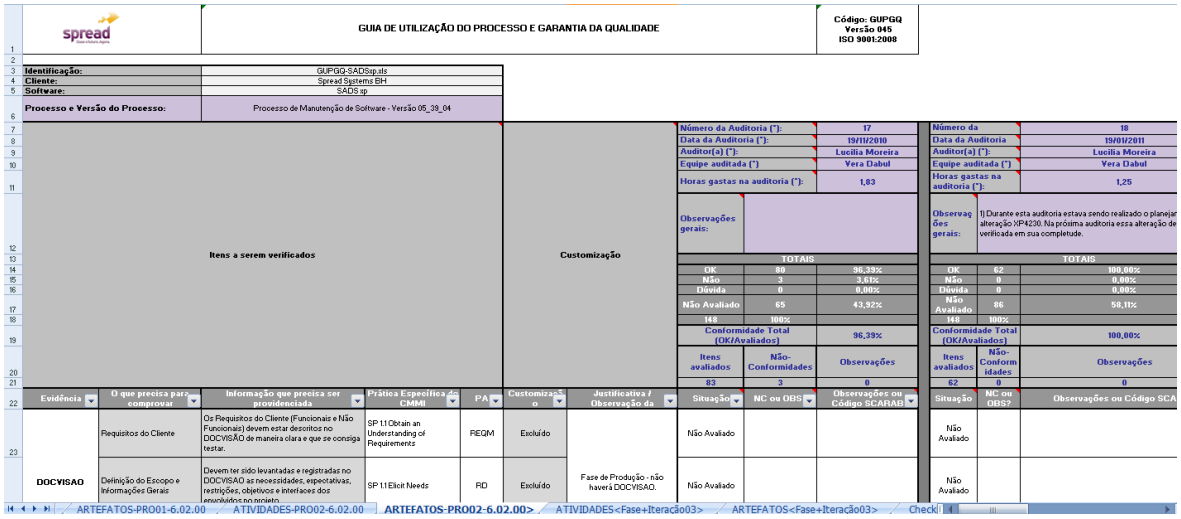
Escrevo qual requisito será implementado em qual componente.

Qual o nome da atividade que você registra na CONRAS?

A de projetar.

Anexo C

Evidências do Estudo de Caso



spread	GUIA DE UTILIZAÇÃO DO PROCESSO E GARANTIA DA QUALIDADE	Código: GUPGQ Versão 045 ISO 9001:2008

Identificação:	GUPGQ-SADSsp.xls
Cliente:	Spread Systems BH
Software:	SADS sp
Processo e Versão do Processo:	Processo de Manutenção de Software - Versão 05_39_04

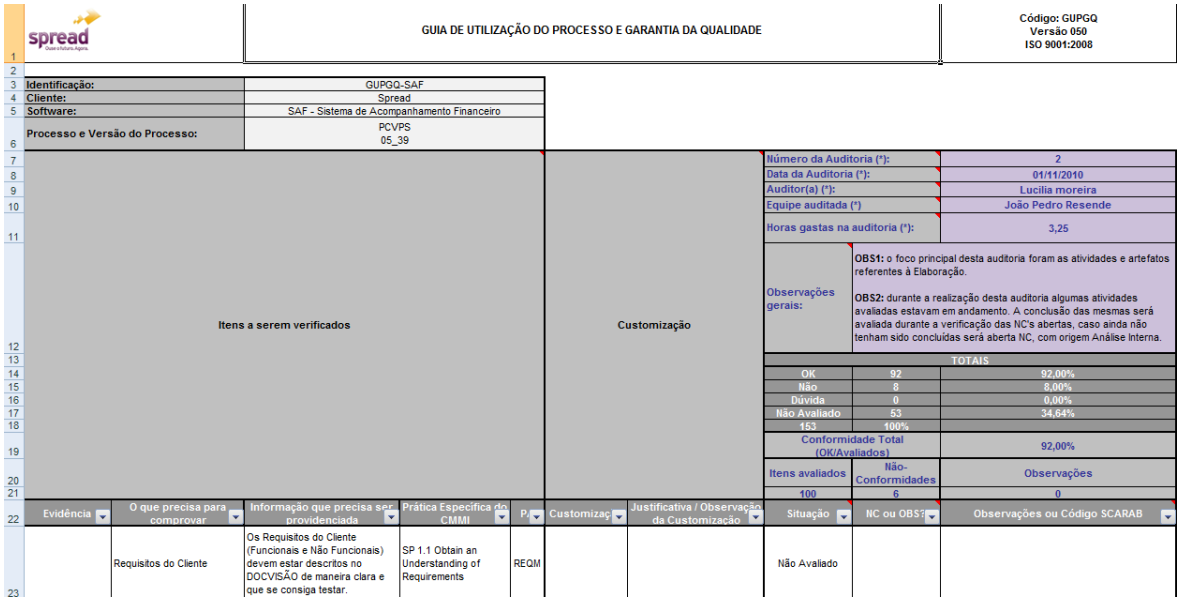
Itens a serem verificados	Customização	Número da Auditoria (*):	17	Número da	18
		Data da Auditoria (*):	19/11/2010	Data da Auditoria	19/01/2011
		Auditor(a) (*):	Lucilia Moreira	Auditor(a) (*):	Lucilia Moreira
		Equipe auditada (*)	Vera Dabul	Equipe auditada (*)	Vera Dabul
		Horas gastas na auditoria (*):	1,83	Horas gastas na auditoria (*):	1,25
		Observações gerais:		Observaç ões gerais:	1) Durante esta auditoria estava sendo realizado o planejar alteração XP4230. Na próxima auditoria essa alteração de verificada em sua completude.

TOTAIS			TOTAIS		
OK	80	36,39%	OK	62	100,00%
Não	3	3,81%	Não	0	0,00%
Dúvida	0	0,00%	Dúvida	0	0,00%
Não Avaliado	65	43,92%	Não Avaliado	86	58,11%
148	100%		148	100%	
Conformidade Total (OK/Avaliados)		96,39%	Conformidade Total (OK/Avaliados)		100,00%
Itens avaliados	Não-Conformidades	Observações	Itens avaliados	Não-Conform idades	Observações
83	3	0	62	0	0

Evidência	O que precisa para comprovar	Informação que precisa ser providenciada	Prática Específica do CMMI	PA	Customizaçã o	Justificativa / Observação da	Situação	NC ou OBS	Observações ou Código SCARAB	Situação	NC ou OBS?	Observações ou Código SCA
DOCVISAO	Requisitos do Cliente	Os Requisitos do Cliente (Funcionais e Não Funcionais) devem estar descritos no DOCVISÃO de maneira clara e que se consiga testar.	SP 1.1 Obtain an Understanding of Requirements	REQM	Excluído		Não Avaliado			Não Avaliado		
	Definição do Escopo e Informações Gerais	Devem ter sido levantadas e registradas no DOCVISAO as necessidades, expectativas, restrições, objetivos e interfaces dos envolvidos no projeto	SP 1.1 Elicit Needs	RD	Excluído	Fase de Produção - não haverá DOCVISAO.	Não Avaliado			Não Avaliado		

ARTEFATOS-PRO01-6.02.00 | ATIVIDADES-PRO02-6.02.00 | ARTEFATOS-PRO02-6.02.00> | ATIVIDADES<Fase+Iteração03> | ARTEFATOS<Fase+Iteração03> | Check

Figura C.1. Evidência da Auditoria 1



spread	GUIA DE UTILIZAÇÃO DO PROCESSO E GARANTIA DA QUALIDADE	Código: GUPGQ Versão 050 ISO 9001:2008

Identificação:	GUPGQ-SAF
Cliente:	Spread
Software:	SAF - Sistema de Acompanhamento Financeiro
Processo e Versão do Processo:	PCVPS 05_39

Itens a serem verificados	Customização	Número da Auditoria (*):	2
		Data da Auditoria (*):	01/11/2010
		Auditor(a) (*):	Lucilia moreira
		Equipe auditada (*)	João Pedro Resende
		Horas gastas na auditoria (*):	3,25
		Observações gerais:	**OBS1:** o foco principal desta auditoria foram as atividades e artefatos referentes à Elaboração. **OBS2:** durante a realização desta auditoria algumas atividades avaliadas estavam em andamento. A conclusão das mesmas será avaliada durante a verificação das NC's abertas, caso ainda não tenham sido concluídas será aberta NC, com origem Análise Interna.

TOTAIS		
OK	92	92,00%
Não	8	8,00%
Dúvida	0	0,00%
Não Avaliado	53	34,64%
153	100%	
Conformidade Total (OK/Avaliados)		92,00%
Itens avaliados	Não-Conformidades	Observações
100	6	0

Evidência	O que precisa para comprovar	Informação que precisa ser providenciada	Prática Específica do CMMI	PA	Customizaç	Justificativa / Observação da Customização	Situação	NC ou OBS?	Observações ou Código SCARAB
	Requisitos do Cliente	Os Requisitos do Cliente (Funcionais e Não Funcionais) devem estar descritos no DOCVISÃO de maneira clara e que se consiga testar.	SP 1.1 Obtain an Understanding of Requirements	REQM			Não Avaliado		

Figura C.2. Evidência da Auditoria 2

C.1 Evidências das Auditorias

As Figuras C.1, C.2, C.3, C.4 mostram os resultados das auditorias nos projetos participantes.

A Figura C.5 mostra os formulários de tarefas preenchido pelos executores das tarefas com informações sobre as pesquisas realizadas, seus resultados e suas razões. s pesquisas

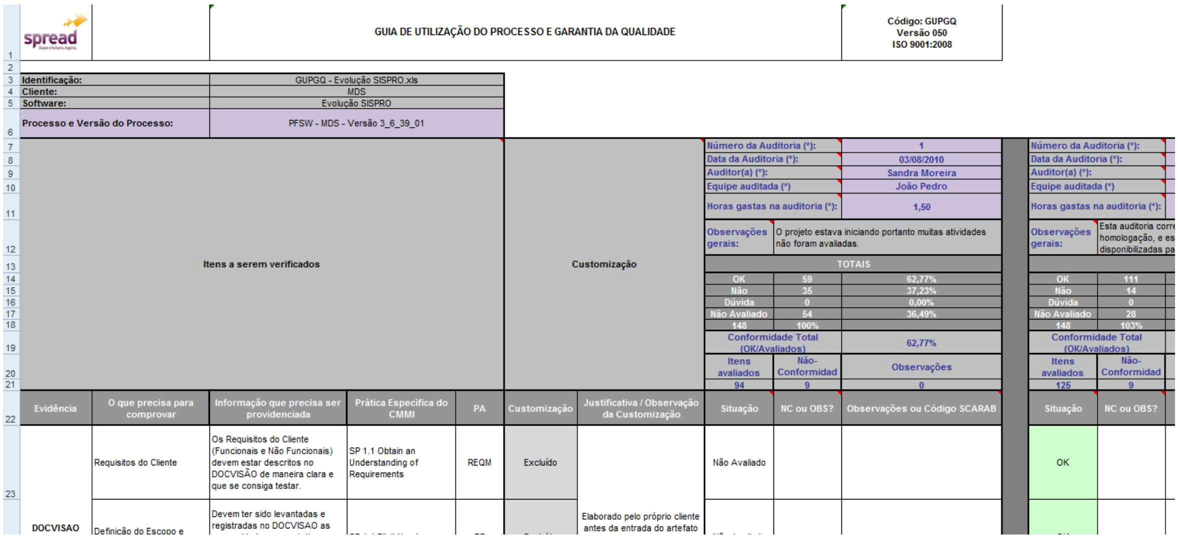


Figura C.3. Evidência da Auditoria 3

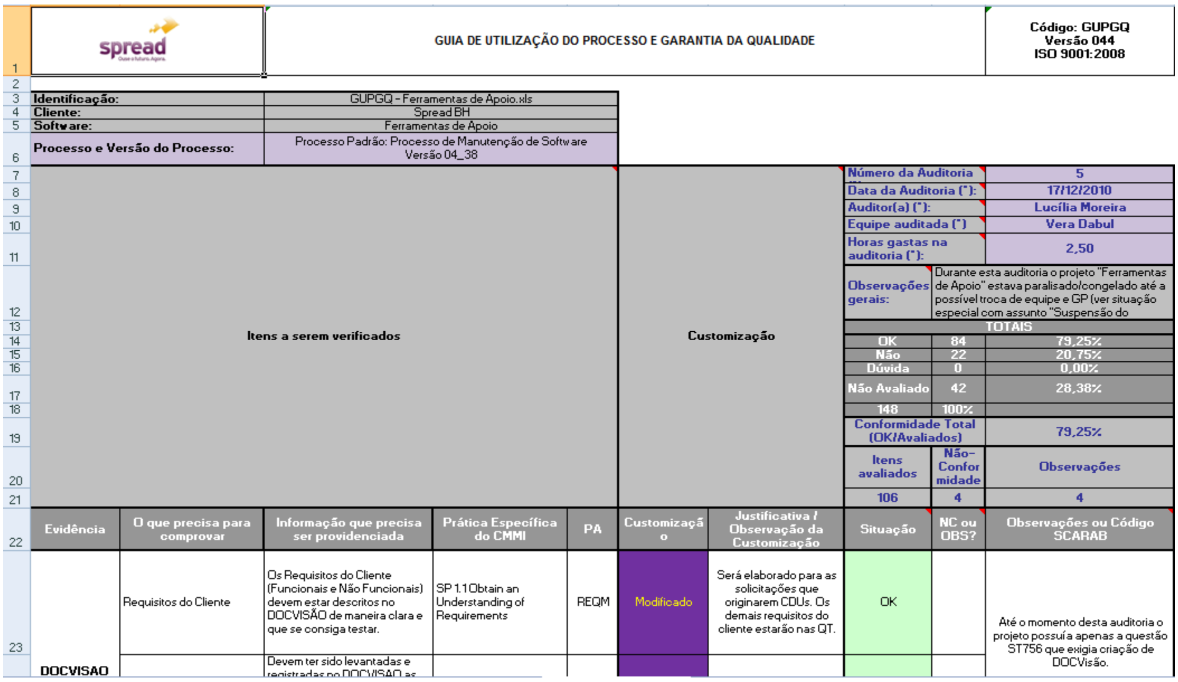


Figura C.4. Evidência da Auditoria 4

Método	Tarefa	Executor	Data	O que	No. elementos pesq	No. de elementos encontr
RAISE	Acompanhamento	João Pedro	03/12/2010	Requisitos alocados para a entrega	35	
RAISE	Acompanhamento	João Pedro	03/12/2010	Requisitos em implementação	7	
RAISE	Controlar Alteração	João Pedro	06/12/2010	Códigos afetados pela alteração do requisto SAF 236	1	
RAISE	Controlar Alteração	João Pedro	06/12/2010	Outros requisitos afetados	10	
RAISE	Controlar Alteração	João Pedro	06/12/2010	Códigos indiretamente afetados	42	
RAISE	Acompanhamento	João Pedro	10/12/2010	Requisitos alocados para a entrega	35	
RAISE	Acompanhamento	João Pedro	10/12/2010	requisitos ja satisfeitos	3	
RAISE	Acompanhamento	João Pedro	10/12/2010	requisitos em implementacao	7	
RAISE	Acompanhamento	João Pedro	17/12/2010	Requisitos alocados para a entrega	35	
RAISE	Acompanhamento	João Pedro	17/12/2010	requisitos ja satisfeitos	6	
RAISE	Acompanhamento	João Pedro	17/12/2010	requisitos em implementacao	9	
RAISE	Controlar alteração	João Pedro	04/01/2011	Códigos afetados pela alteração do requisito SAF123	13	
RAISE	Controlar alteração	João Pedro	04/01/2011	Outros requisitos afetados	17	
RAISE	Controlar alteração	João Pedro	04/01/2011	Códigos indiretamente afetados	3	
RAISE	Acompanhamento	João Pedro	07/01/2011	requisitos alocados para a entrega	35	
RAISE	Acompanhamento	João Pedro	07/01/2011	requisitos ja satisfeitos	12	
RAISE	Acompanhamento	João Pedro	07/01/2011	requisitos em implementacao	8	
RAISE	Acompanhamento	João Pedro	07/01/2011	motivo do nao cumprimento do planejado	3	
RAISE	Validação da alteração	João Pedro	14/01/2011	O codigo alterado é o previsto	26	
RAISE	Validação da alteração	João Pedro	14/01/2011	quem quando e porque da alteração no código 131220	3	

Figura C.5. Formulário das Tarefas de Pesquisa da Rastreabilidade