



Ministério da  
Ciência e Tecnologia



INPE-15761-TDI/1504

## UM MODELO PARA RASTREABILIDADE DE REQUISITOS DE SOFTWARE BASEADO EM GENERALIZAÇÃO DE ELOS E ATRIBUTOS

Elias Canhadas Genvigir

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada,  
orientada pelo Dr. Nandamudi Lankalapalli Vijaykumar, aprovada em 31 de março  
de 2009.

Registro do documento original:

<<http://urlib.net/sid.inpe.br/mtc-m18@80/2009/03.02.14.1>>

INPE  
São José dos Campos  
2009

**PUBLICADO POR:**

Instituto Nacional de Pesquisas Espaciais - INPE

Gabinete do Diretor (GB)

Serviço de Informação e Documentação (SID)

Caixa Postal 515 - CEP 12.245-970

São José dos Campos - SP - Brasil

Tel.:(012) 3945-6911/6923

Fax: (012) 3945-6919

E-mail: [pubtc@sid.inpe.br](mailto:pubtc@sid.inpe.br)

**CONSELHO DE EDITORAÇÃO:****Presidente:**

Dr. Gerald Jean Francis Banon - Coordenação Observação da Terra (OBT)

**Membros:**

Dr<sup>a</sup> Maria do Carmo de Andrade Nono - Conselho de Pós-Graduação

Dr. Haroldo Fraga de Campos Velho - Centro de Tecnologias Especiais (CTE)

Dr<sup>a</sup> Inez Staciarini Batista - Coordenação Ciências Espaciais e Atmosféricas (CEA)

Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)

Dr. Ralf Gielow - Centro de Previsão de Tempo e Estudos Climáticos (CPT)

Dr. Wilson Yamaguti - Coordenação Engenharia e Tecnologia Espacial (ETE)

**BIBLIOTECA DIGITAL:**

Dr. Gerald Jean Francis Banon - Coordenação de Observação da Terra (OBT)

Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)

Jefferson Andrade Ancelmo - Serviço de Informação e Documentação (SID)

Simone A. Del-Ducca Barbedo - Serviço de Informação e Documentação (SID)

**REVISÃO E NORMALIZAÇÃO DOCUMENTÁRIA:**

Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)

Marilúcia Santos Melo Cid - Serviço de Informação e Documentação (SID)

Yolanda Ribeiro da Silva Souza - Serviço de Informação e Documentação (SID)

**EDITORAÇÃO ELETRÔNICA:**

Viveca Sant´Ana Lemos - Serviço de Informação e Documentação (SID)



Ministério da  
Ciência e Tecnologia



INPE-15761-TDI/1504

## UM MODELO PARA RASTREABILIDADE DE REQUISITOS DE SOFTWARE BASEADO EM GENERALIZAÇÃO DE ELOS E ATRIBUTOS

Elias Canhadas Genvigir

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada,  
orientada pelo Dr. Nandamudi Lankalapalli Vijaykumar, aprovada em 31 de março  
de 2009.

Registro do documento original:

<http://urlib.net/sid.inpe.br/mtc-m18@80/2009/03.02.14.1>

INPE  
São José dos Campos  
2009

Dados Internacionais de Catalogação na Publicação (CIP)

---

Genvigir, Elias Canhadas.

G287m Um modelo para rastreabilidade de requisitos de software baseado em generalização de elos e atributos / Elias Canhadas Genvigir. – São José dos Campos : INPE, 2009.  
200p. ; (INPE-15761-TDI/1504)

Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2009.

Orientador : Dr. Nandamudi Lankalapalli Vijaykumar.

1. Rastreabilidade. 2. Requisitos de software. 3. Modelos. 4. Engenharia de requisitos. 5. Engenharia de software. I. Título.

CDU 004.414.22

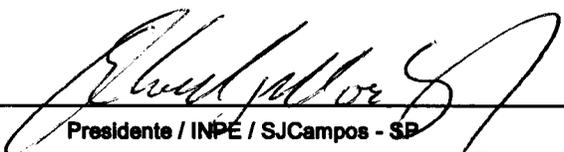
---

Copyright © 2009 do MCT/INPE. Nenhuma parte desta publicação pode ser reproduzida, armazenada em um sistema de recuperação, ou transmitida sob qualquer forma ou por qualquer meio, eletrônico, mecânico, fotográfico, reprográfico, de microfilmagem ou outros, sem a permissão escrita da Editora, com exceção de qualquer material fornecido especificamente no propósito de ser entrado e executado num sistema computacional, para o uso exclusivo do leitor da obra.

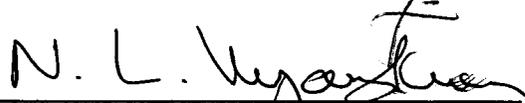
Copyright © 2009 by MCT/INPE. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, microfilming or otherwise, without written permission from the Publisher, with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use of the reader of the work.

Aprovado (a) pela Banca Examinadora  
em cumprimento ao requisito exigido para  
obtenção do Título de Doutor(a) em  
Computação Aplicada

Dr. Antonio Miguel Vieira Monteiro

  
Presidente / INPE / SJCampos - SP

Dr. Nandamudi Lankalapalli Vijaykumar

  
Orientador(a) / INPE / SJCampos - SP

Dr. Mauricio Gonçalves Vieira Ferreira

  
Membro da Banca / INPE / SJCampos - SP

Dr. Rafael Duarte Coelho dos Santos

  
Membro da Banca / INPE / SJCampos - SP

Dr. Celso Massaki Hirata

  
Convidado(a) / ITA / SJCampos - SP

Dr. Carlos Renato Lisboa Francês

  
Convidado(a) / UFPA / Belém - PA

Aluno (a): Elias Canhadas Genvigir

São José dos Campos, 31 de março de 2009



*“Assim diz o SENHOR: Não se glorie o sábio na sua sabedoria, nem o forte, na sua força, nem o rico, nas suas riquezas; mas o que se gloriar, glorie-se nisto: em me conhecer e saber que eu sou o SENHOR e faço misericórdia, juízo e justiça na terra; porque destas coisas me agrado, diz o SENHOR”.*

*“Jeremias 9:23”*



Ao SENHOR, Deus eterno, fonte da sabedoria e salvação.



## AGRADECIMENTOS

A Deus, Senhor Eterno, Rocha e Fortaleza, que em Sua bondade permitiu a realização deste trabalho.

A minha esposa Fabiana.

A minha mãe Leonor e aos irmãos Humberto, Gabriel, Teresa Cristina e Renata.

Ao Prof. Dr. Nandamudi Lankalapalli Vijaykumar, orientador valoroso, exemplo, referência de trabalho, que teve a maestria de ensinar que a pesquisa pode ser gratificante e o pesquisador competente sem perder a sua simplicidade.

Aos professores e pesquisadores da CAP e do LAC, pelas contribuições, sugestões e bons diálogos ao longo do doutorado e pela dedicação e conhecimento repassados nos cursos, especialmente ao Dr. Ezzat S. Chalhoub.

Aos servidores do INPE, especialmente dos laboratórios que participaram do estudo experimental executado nesta tese, fornecendo os dados de seus projetos e despendendo seu tempo e atenção.

À UTFPR campus Cornélio Procópio e aos colegas do grupo de Informática, que contribuíram para a realização deste doutorado.

À CAPES pelo apoio financeiro.



## RESUMO

Estabelecer, adequadamente, o conhecimento sobre o papel que um software deve desempenhar é atividade crítica, e de difícil execução, para a Engenharia de Software, sendo que tal responsabilidade é atribuída às atividades da Engenharia de Requisitos. Entre essas atividades destaca-se, neste trabalho, a Rastreabilidade, que possui como meta a definição dos relacionamentos entre requisitos e demais artefatos produzidos durante o processo de desenvolvimento. Para tanto, a Rastreabilidade faz uso de elos como o principal elemento para manter e representar esses relacionamentos. A Rastreabilidade está diretamente associada à qualidade dos requisitos, exercendo um papel extremamente importante no gerenciamento desses elementos, bem como nas atividades de análise de impacto, de validação de requisitos, de testes de regressão, entre outras. Esta tese discute as principais características da Rastreabilidade. Apresenta-se como a área é tratada através de modelos que, no caso da Rastreabilidade, são criados com base em informações como as necessidades dos envolvidos, as práticas, as metodologias, as normas ou os padrões disponíveis. Neste trabalho são avaliados os pontos positivos e negativos dos principais modelos existentes e como os elos são abordados nessas pesquisas. Com o intuito de apresentar melhorias e facilidades, para a execução da Rastreabilidade, esta tese propõe um modelo para a generalização de diferentes elos, visando permitir a definição de elos e possibilitando a inserção de atributos a esses itens. Também é desenvolvida uma arquitetura de software que possui como objetivo a construção do projeto para a implementação de um protótipo. O protótipo é implementado para execução em ambiente WEB fazendo uso de tecnologias de código aberto. Ao final do trabalho são apresentados os resultados de um estudo experimental. Este estudo teve por objetivo avaliar a aplicação do modelo frente a projetos que não fazem uso das facilidades apresentadas pelo modelo proposto. Os resultados são analisados e discutidos visando à apresentação das contribuições obtidas que podem ser utilizadas para aperfeiçoar a prática e motivar a continuidade da pesquisa.



# A MODEL FOR REQUIREMENTS TRACEABILITY BASED IN GENERALIZATION OF LINKS AND ATTRIBUTES

## ABSTRACT

To correctly provide the knowledge about the role that software should execute is a critical and difficult activity for the Software Engineering, and this responsibility belongs to the activities of the Requirements Engineering. Among these activities the main point, in this work, is the traceability, which has a target to define the relationships between requirements and other artifacts produced during the software development process. It is conducted with the use of links that are a main resource to provide and represent the relationships. The traceability is directly associated with quality requirements, performing an extremely important role in the management of these elements, as well as the activities of impact analysis, requirements validation, testing of regression, among others. This thesis discusses the main features of Traceability. It presents how the area is covered by models which are created based on information on those involved, practices, methods, or available standards. The positive and negative aspects of the main existing models are investigated and describe how the links are addressed in these existing models. In order to make improvements and facilities for the traceability implementation, this thesis proposes a model for the generalization of different links, to allow the definition of links and allowing the insertion of attributes to these items. Also, a software architecture has been developed by implementing a prototype. The prototype is developed to run in WEB using open source technologies. At the end of this thesis the results of an experimental study are presented. This study aimed to evaluate the model comparing it with some projects that do not make use of the facilities presented by this research. The results are analyzed and discussed focusing on illustrating the contributions obtained so that they may be used to improve the practice and to motivate the research in the Traceability.



# SUMÁRIO

Pág.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

## LISTA DE SÍMBOLOS

<b>1</b>	<b>INTRODUÇÃO</b>	<b>27</b>
1.1	Visão geral do trabalho	29
1.2	Objetivos do trabalho	30
1.3	Organização do trabalho	30
<b>2</b>	<b>RASTREABILIDADE DE REQUISITOS DE SOFTWARE</b>	<b>33</b>
2.1	Engenharia de requisitos	33
2.2	Gerenciamento de requisitos	35
2.3	Rastreabilidade	38
2.3.1	Classificação para a rastreabilidade	39
2.3.2	Técnicas de rastreabilidade	42
2.3.3	Elos de rastreabilidade	44
2.3.4	Métricas para rastreabilidade	47
2.3.5	Ferramentas para rastreabilidade	51
2.3.6	Modelos para rastreabilidade	53
2.4	Discussão do problema	56
2.5	Conclusão do Capítulo	58
<b>3</b>	<b>MODELO PARA A GENERALIZAÇÃO DE ELOS DE RAS- TREABILIDADE</b>	<b>61</b>
3.1	Modelo proposto	61
3.2	Formalização do modelo	64
3.3	Definição de artefatos	65
3.4	Conclusão do Capítulo	70

<b>4</b>	<b>UM MODELO DE ARQUITETURA DE SOFTWARE PARA SUPORTE AO MODELO DE RASTREABILIDADE . . . . .</b>	<b>71</b>
4.1	Elicitação dos requisitos para a arquitetura . . . . .	71
4.1.1	Definição do domínio da aplicação . . . . .	72
4.1.2	Compreensão do problema a ser resolvido . . . . .	72
4.1.3	Definição dos requisitos funcionais da aplicação . . . . .	73
4.2	Diagrama de contexto de caso de uso . . . . .	73
4.2.1	Descrição dos casos de uso . . . . .	74
4.3	Arquitetura para implementação do modelo para rastreabilidade de requisitos . . . . .	77
4.3.1	Definição dos elementos da arquitetura . . . . .	78
4.3.2	Comparação entre funcionalidades e casos de uso . . . . .	80
4.4	Desenvolvimento da aplicação . . . . .	81
4.4.1	Interfaces dos módulos da aplicação . . . . .	82
4.5	Conclusão do Capítulo . . . . .	89
<b>5</b>	<b>ESTUDO DE CASO – EXECUÇÃO DE UM ESTUDO EXPERIMENTAL COM O OBJETIVO DE AVALIAR O MODELO PARA RASTREABILIDADE . . . . .</b>	<b>91</b>
5.1	Introdução . . . . .	91
5.2	Fase de definição . . . . .	93
5.2.1	Objetivo global . . . . .	93
5.2.2	Objetivo da medição . . . . .	93
5.2.3	Objetivo do estudo . . . . .	94
5.2.4	Questões . . . . .	94
5.3	Fase de planejamento . . . . .	94
5.3.1	Contexto . . . . .	94
5.3.2	Definição das hipóteses . . . . .	95
5.3.3	Definição dos instrumentos . . . . .	96
5.3.4	Identificação das variáveis . . . . .	97
5.3.5	Seleção dos métodos de análise . . . . .	98
5.4	Fase de coleta de dados . . . . .	98
5.5	Fase de interpretação . . . . .	98
5.5.1	Variável tempo . . . . .	98
5.5.2	Variável <i>rationale</i> . . . . .	105
5.6	Discussão . . . . .	109

<b>6 CONCLUSÃO</b> . . . . .	<b>113</b>
6.1 Resultados . . . . .	113
6.2 Trabalhos futuros . . . . .	117
<b>REFERÊNCIAS BIBLIOGRÁFICAS</b> . . . . .	<b>121</b>
<b>A APÊNDICE A – Questionários utilizados para execução do ex- perimento</b> . . . . .	<b>137</b>
A.1 Questionário sobre o perfil dos participantes do experimento . . . . .	137
A.2 Questionário sobre as práticas de engenharia de requisitos e de engenharia de software adotadas pelo grupo . . . . .	137
A.3 Questionário e itens para avaliação de matriz de rastreabilidade . . . . .	139
A.4 Documentação auxiliar para o líder responder o questionário 3 . . . . .	141
<b>A ANEXO A - ARTIGOS E OUTROS TRABALHOS CIENTÍFI- COS PUBLICADOS DURANTE O DOUTORADO</b> . . . . .	<b>143</b>



## LISTA DE FIGURAS

	<u>Pág.</u>	
2.1	Processo de engenharia de requisitos. . . . .	36
2.2	Alocação da rastreabilidade dentro da engenharia de software. . . . .	38
2.3	Rastreabilidade Horizontal e Vertical. . . . .	40
2.4	Pré e Pós-Rastreabilidade. . . . .	41
2.5	Elementos básicos de um elo. . . . .	45
2.6	Metamodelo para rastreabilidade de requisitos proposto por Ramesh e Jarke (2001). . . . .	54
2.7	Principais elementos do modelo de Toranzo et al. (2002) . . . . .	55
3.1	Elementos do modelo proposto para a rastreabilidade de requisitos. . . . .	62
3.2	Diagrama entidade relacionamento do modelo proposto. . . . .	64
3.3	Diagrama de atividades do processo de criação de um artefato com uso do modelo. . . . .	66
3.4	Valores do artefato $a_1$ . . . . .	67
3.5	Valores do artefato $a_2$ . . . . .	68
3.6	Valores do artefato $a_3$ . . . . .	69
4.1	Diagrama de contexto de caso de uso para a aplicação. . . . .	74
4.2	Arquitetura proposta para dar suporte ao modelo para rastreabilidade. . . . .	78
4.3	Controle de acesso. . . . .	82
4.4	Controle de tipos de artefatos. . . . .	83
4.5	Controle de atributos de artefatos. . . . .	83
4.6	Controle de artefatos. . . . .	84
4.7	Controle de matrizes – matrizes disponíveis para edição. . . . .	85
4.8	Controle de matrizes – elos disponíveis. . . . .	85
4.9	Controle de matrizes – matriz escolhida para edição. . . . .	86
4.10	Controle de matrizes – seleção de artefatos para origem e destino. . . . .	86
4.11	Atributos de um elo em uma matriz. . . . .	87
4.12	Edição de um elo através do módulo de gerenciamento de artefatos. . . . .	88
4.13	Análise de impacto. . . . .	88
4.14	Impacto direto e indireto sobre o requisito ET006. . . . .	89
5.1	Níveis conceituais do GQM. . . . .	92
5.2	Processo de experimentação utilizado. . . . .	92
5.3	Tempo utilizado pelo membro e estimado pelo líder do grupo 1. . . . .	100

5.4	Tempo utilizado pelo membro e estimado pelo líder do grupo 2. . . . .	100
5.5	Tempo utilizado pelo membro e estimado pelo líder do grupo 3. . . . .	101
5.6	Gráfico <i>Box Plot</i> dos dados sobre tempo do grupo 1. . . . .	102
5.7	Gráfico <i>Box Plot</i> dos dados sobre tempo do grupo 2. . . . .	103
5.8	Gráfico <i>Box Plot</i> dos dados sobre tempo do grupo 3. . . . .	104
5.9	Número de respostas corretas e erradas de <i>racionales</i> entre membros e líderes dos três grupos. . . . .	106
5.10	Número de <i>racionales</i> corretos e errados, dos três grupos, com os respectivos valores de resposta para a escala Likert. . . . .	107
6.1	Proposta de trabalho futuro para uma ferramenta para engenharia de requisitos. . . . .	118

## LISTA DE TABELAS

	<u>Pág.</u>
2.1 Exemplos de elos descritos na literatura classificados por seus respectivos autores e subgrupos. . . . .	47
2.2 Comparação entre algumas ferramentas de gerenciamento de requisitos. . .	53
2.3 Matriz representando elos do tipo satisfação entre o requisito $r_1$ e os casos de uso $a_1$ e $a_2$ . . . . .	57
2.4 Matriz representando elos do tipo <i>rationale</i> . . . . .	57
2.5 Matriz representando elos do tipo satisfação entre o requisito $r_1$ e os casos de uso $a_1$ e $a_2$ e a razão como atributo adional. . . . .	58
3.1 Descrição dos elementos do modelo. . . . .	65
3.2 Resultado da expressão 3.1. . . . .	67
3.3 Resultado da expressão 3.2. . . . .	70
4.1 Comparação entre casos de uso e funcionalidades. . . . .	81
5.1 Exemplo de matriz analisada pelos líderes dos grupos para responder as questões. . . . .	97
5.2 Tempo gasto em minutos pelos membros e estimado pelos líderes. . . . .	99
5.3 Estatísticas descritivas dos dados coletados sobre o tempo utilizado nos três grupos. . . . .	99
5.4 Quantidade de acertos e erros (por parte) dos líderes em relação ao tempo. . . . .	101
5.5 Resultados obtidos para o Teste $U$ do grupo 1. . . . .	102
5.6 Resultados obtidos para o Teste $U$ do grupo 2. . . . .	103
5.7 Resultados obtidos para o Teste $U$ do grupo 3. . . . .	104
5.8 Número de respostas corretas e erradas de <i>Rationales</i> entre membros e líderes dos três grupos. . . . .	106
5.9 Porcentagem de acertos e erros dos líderes entre os grupos nas escalas 4 e 5. . . . .	108
5.10 Resultado do teste Qui-quadrado entre as variáveis número de acertos e escalas nos níveis 4 e 5. . . . .	108
5.11 Resultado do teste Qui-quadrado entre as variáveis número de erros e escalas nos níveis 4 e 5. . . . .	108
6.1 Artigos publicados . . . . .	117

A.1 Exemplo de matriz. . . . .	141
A.1 Artigos publicados . . . . .	143

## LISTA DE ABREVIATURAS E SIGLAS

A	–	Artefatos
ABNT	–	Associação Brasileira de Normas Técnicas
AJAX	–	<i>Asynchronous Javascript And XML</i>
AT	–	Atributos_Do_Tipo
BPEL	–	<i>Business Process Execution Language</i>
BPM	–	<i>Business Process Management</i>
CMMI	–	<i>Capability Maturity Model Integration</i>
CMMI-DEV	–	<i>Capability Maturity Model Integration for Development</i>
COV	–	<i>Next-level coverage</i>
COVdown	–	<i>Next Level down Coverage</i>
COVup	–	<i>Next Level up Coverage</i>
DCOV	–	<i>Traceability to the Lowest Level</i>
DHCOV	–	<i>Full Depth and Height Coverage</i>
DRL	–	<i>Decision Representation Language</i>
ECSS	–	<i>European Cooperation for Space Standardization</i>
ER	–	Entidade Relacionamento
FFBD	–	<i>Functional Flow Block Diagram</i>
GQM	–	<i>Goal Question Metric</i>
HCOV	–	<i>Traceability to the Highest Level Coverage</i>
HTML	–	<i>HyperText Markup Language</i>
I	–	Instancias
IBIS	–	<i>Issue-Based Information System</i>
IEC	–	<i>International Electrotechnical Commission</i>
IEEE	–	<i>Institute of Electrical and Electronics Engineers</i>
INCONSE	–	<i>International Council on Systems Engineering</i>
ISO	–	<i>International Organization for Standardization</i>
ITM	–	<i>Inconsistent Traceability Metrics</i>
ITMup	–	<i>Inconsistent Traceability Link upward</i>
JSP	–	<i>JavaServer Pages</i>
MDA	–	<i>Model Driven Architecture</i>
NBR	–	Norma Brasileira
NRM	–	<i>Need-based Requirements Management</i>
OMG	–	<i>Object Management Group</i>
QOC	–	<i>Questions, Options and Criteria</i>
RF	–	Requisito Funcional
RI	–	Recuperação de Informação
SEURAT	–	<i>Software Engineering Using Rationale</i>
SPEM	–	<i>Software Process Engineering Metamodel Specification</i>
SWEBOK	–	<i>Guide to the Software Engineering Body of Knowledge</i>
TF	–	Tipo_De_Artefato
TI	–	Tecnologia de Informação
TIMdown	–	<i>Inconsistent Traceability Link downward</i>
UML	–	<i>Unified Modeling Language</i>
UTM	–	<i>Undefined Traceability Metrics</i>
UTMdown	–	<i>Traceability Links downward</i>
UTMup	–	<i>Traceability Links upward</i>
XML	–	<i>Extensible Markup Language</i>



## LISTA DE SÍMBOLOS

- $\pi$  – Operação projeção da álgebra relacional
- $\sigma$  – Operação seleção da álgebra relacional
- $\bowtie$  – Operação junção natural da álgebra relacional
- $\delta$  – Diferença entre medianas
- $\alpha$  – Nível de significância
- $\chi^2$  – Qui-quadrado
- $p$  – Valor p



## 1 INTRODUÇÃO

Um software é dirigido à realização de tarefas, que estão associadas à solução computacional de um problema do domínio da natureza humana. Devido à amplitude desses problemas é necessário que sua produção atinja padrões básicos de qualidade e produtividade.

Como resposta à necessidade, de melhoria da qualidade e da produtividade, a Engenharia de Software foi evoluindo e se especializando em várias subáreas. Uma classificação para essas subáreas foi realizada pelo projeto *Guide to the Software Engineering Body of Knowledge* – SWEBOK ([IEEE, 2004](#)) que definiu dez áreas de conhecimento para a Engenharia de Software.

A primeira área definida pelo SWEBOK trata dos requisitos de software. Esses elementos são de extrema importância, pois são definidos, na maioria dos processos de desenvolvimento, durante os estágios iniciais do projeto, como uma especificação do que deve ser implementado, descrevendo os atributos ou propriedades do sistema, como o sistema deve comportar-se, ou também podendo constituir-se restrições ao processo ([SOMMERVILLE; SAWYER, 1998](#)).

A importância dos requisitos para o desenvolvimento é tão crítica que nenhuma outra parte do trabalho com software incapacita e prejudica tanto o sistema e, depois de concluído, nenhuma outra parte é mais difícil de corrigir do que os requisitos ([BROOKS, 1995](#)). É possível até afirmar que a medida primária do sucesso de um software está no grau em que ele atinge os requisitos para os quais foi dirigido ([NUSEIBEH; EASTERBROOK, 2000](#)).

A área da Engenharia de Software que trata dos requisitos de software é conhecida como Engenharia de Requisitos, que é o processo de descoberta dos requisitos, de identificação dos envolvidos e suas necessidades e de documentação, de forma que seja útil para a análise, comunicação e a sua subsequente implementação ([NUSEIBEH; EASTERBROOK, 2000](#)).

A Engenharia de Requisitos é composta de diversas atividades como a Elicitação, a Análise, a Documentação e o Gerenciamento. Esta última atividade tem o objetivo controlar a agregação e a evolução dos requisitos ao projeto do sistema, realizando isso através das atividades da Rastreabilidade de Requisitos.

O termo Rastreabilidade é comumente usado para descrever a referência para um grupo coletivo de requisitos baseados em seus relacionamentos, fazendo uso de relacionamentos sobre os requisitos, projeto e implementação de um sistema para prover a qualidade, além de estabelecer mecanismos que podem ser usados para avaliar o impacto de mudanças no sistema.

A importância da Rastreabilidade é tamanha que várias normas e padrões de qualidade fazem referência as suas práticas, como: o CMMI (SEI, 2006), as normas ISO/IEC-15504 (ISO/IEC, 2003) e IEEE-830 (IEEE, 1998), as normas militares como a MIL-STD-2167A (DoD, 1988) e MIL-STD-498 (DoD, 1994), o novo conjunto de normas ISO/IEC 25000 (BØEGH, 2008), e as normas espaciais como as da Agência Espacial Européia ECSS-E-40 (ECSS, 2003; ECSS, 2005).

Ramesh e Jarke (2001) afirmam que algumas organizações consideram a Rastreabilidade como área estratégica, tal é o caso do Departamento de Defesa Norte Americano que utiliza 4% de seu orçamento em Tecnologia de Informação (TI) com Rastreabilidade. O orçamento do ano de 2008 para esse departamento está estimado em 481,4 bilhões de dólares, sendo 31,4 bilhões destinados para TI (DOD, 2007). Considerando-se o mesmo percentual de 2001 (4%), afirmado por Ramesh e Jarke (2001), o custo com Rastreabilidade, para 2008, seria (da ordem) de 1,5 bilhões de dólares.

Apesar da sua importância, nem todas as organizações, envolvidas com o desenvolvimento de sistemas, aplicam a rastreabilidade de requisitos de forma adequada. Muitas vezes, os valores e os esforços envolvidos nessa atividade se dão a esmo; os padrões fornecem pouca orientação, apesar dos esforços em estabelecê-los, e os modelos e mecanismos são mal compreendidos e interpretados (RAMESH; JARKE, 2001).

Um caso típico, desse tipo de problemas, pode ser encontrado no próprio INPE. Em uma pesquisa realizada com membros de quatro laboratórios do INPE, envolvidos com o desenvolvimento de sistemas para a área espacial, foi observado que 67% dos entrevistados não fazem uso de Rastreabilidade e 92% não utiliza nenhum tipo de métrica para Rastreabilidade. Mesmo quando práticas básicas da Engenharia de Requisitos são avaliadas, como Elicitação ou Análise de Requisitos, foi observado que 75% dos membros desses laboratórios não adotam práticas para Elicitação e apenas 50% dos entrevistados fazem uso de técnicas para a Análise de Requisitos

(GENVIGIR; VIJAYKUMAR, 2008).

Como justificativa para o não uso dessas práticas, a pesquisa realizada aponta os seguintes fatores: falta de conhecimento (33%), consideraram as práticas muito sofisticadas (56%), falta de tempo para aplicar as práticas (11%).

A falta de conhecimento sobre as práticas de Rastreabilidade e o receio de que elas sejam demasiadamente complexas são alguns dos fatores motivadores desse trabalho. Estabelecer um modelo visando uma Rastreabilidade mais flexível para o uso, a fim de permitir que os próprios envolvidos com a Rastreabilidade possam definir os padrões de informação a serem usados são alguns dos fatores que direcionam este trabalho.

### 1.1 Visão geral do trabalho

Os relacionamentos são estabelecidos entre requisitos e artefatos de software usando-se elos. Elos são elementos necessários para estabelecer a Rastreabilidade, enquanto que artefatos são considerados informações produzidas ou modificadas como parte de um processo de Engenharia de Software (RAMESH; JARKE, 2001). Artefatos podem ser modelos, documentos, código fonte, seqüências de testes, requisitos ou executáveis. Esses são os elementos de um sistema que podem ser rastreados (CLELAND-HUANG; CHRISTENSEN, 2003).

Existe uma série de propostas para modelos de elos de Rastreabilidade que proveem padrões predefinidos de grupos de elos (RAMESH; JARKE, 2001; LETELIER, 2002; PINHEIRO; GOGUEN, 1996; TORANZO et al., 2002). Tais pesquisas fazem extensivas observações das práticas da Rastreabilidade, mas seus modelos são limitados na fixação dos tipos de elos, ou seja, os grupos de elos são definidos para cobrir uma determinada solução para o domínio do problema para o qual o modelo é proposto, o que pode limitar as práticas da Rastreabilidade. Em adição, esses modelos não permitem a inclusão de atributos ou semânticas ricas (DICK, 2002; DICK, 2005), e isso restringe e inflexibiliza a descrição dos elos.

O foco deste trabalho está em propor uma solução para o problema da predefinição de elos de rastreabilidade a fim de minimizar os problemas apontados e facilitando a criação de novos tipos de elos e de outros artefatos. Objetivos mais precisos sobre a solução adota são tratados na próxima seção.

## 1.2 Objetivos do trabalho

O tópico discutido neste trabalho trata sobre modelos que representam elos de rastreabilidade.

Baseado nas limitações, já mencionadas, o trabalho propõe um modelo que permite: (i) representar vários tipos de elos já abordados pelos modelos existentes; e (ii) criar novos tipos de elos para servir a uma necessidade específica de um projeto. O modelo proposto pode também adicionar atributos para melhorar a semântica dos elos.

O trabalho tem por objetivo apresentar uma solução para generalização de elos de rastreabilidade. Tal objetivo é atingido através da execução das seguintes atividades:

- Elaborar um modelo para Rastreabilidade que permita a representação de diferentes artefatos de software, entre eles elos de rastreabilidade e atributos para estes elos.
- Desenvolver um modelo de arquitetura de software para dar suporte ao modelo.
- Implementar um protótipo de software para a arquitetura apresentada.
- Executar estudos experimentais para a coleta de dados sobre o modelo.
- Comparar estatísticas do modelo em relação a outras técnicas, modelos ou projetos, através dos dados dos estudos experimentais.
- Demonstrar a viabilidade do modelo.

O modelo proposto é modelado através de álgebra relacional. A arquitetura é apresentada para dar suporte ao modelo e um protótipo para essa arquitetura é desenvolvido a fim de demonstrar sua viabilidade. Por fim, é realizado um estudo experimental e são apresentados alguns resultados sobre o uso do modelo.

## 1.3 Organização do trabalho

O Capítulo 2 descreve a Rastreabilidade e seus principais elementos. São apresentados os tipos de Rastreabilidade, elos, técnicas e ferramentas. São exploradas as vantagens e desvantagens de alguns dos principais modelos existentes para Rastreabilidade.

O Capítulo 3 apresenta o modelo para a generalização de elos de Rastreabilidade e são discutidos seus elementos. O modelo é representado através de diagramas, formalizado através de álgebra relacional e exemplos são apresentados demonstrando a criação de artefatos e elos.

No Capítulo 4 é apresentada a arquitetura e o protótipo que foi implementado a fim de validar o modelo proposto. A arquitetura é apresentada através da visão de casos de uso, enquanto que o protótipo é exibido através de suas principais interfaces, fazendo uma relação entre casos de uso e interfaces.

O Capítulo 5 descreve alguns conceitos sobre experimentação de software e o escopo do estudo experimental executado. A descrição do estudo é realizada após a demonstração, análise e interpretação dos dados obtidos.

No Capítulo 6 é apresentada a conclusão geral do trabalho, os resultados obtidos e as contribuições para trabalhos futuros.



## 2 RASTREABILIDADE DE REQUISITOS DE SOFTWARE

Neste Capítulo é apresentada a Engenharia de Requisitos e suas principais atividades, com atenção ao gerenciamento e à rastreabilidade de requisitos. São discutidas as classificações para a rastreabilidade, as técnicas existentes, os principais elementos utilizados para representar relacionamentos, as propriedades de elos de rastreabilidade e seus tipos, métricas e algumas das principais ferramentas disponíveis para o gerenciamento. Os principais modelos para rastreabilidade são descritos, explorando-se o contexto, o domínio para o qual foram desenvolvidos, os tipos de elos que eles estabelecem, e é realizada uma discussão sobre suas qualidades e deficiências. Pretende-se, dessa forma, contextualizar, brevemente, o conhecimento que envolve a rastreabilidade de requisitos e apresentar o foco da pesquisa desenvolvida neste trabalho.

### 2.1 Engenharia de requisitos

A atividade de produção de software, frequentemente chamada de processo de desenvolvimento de software, vem se aperfeiçoando ao longo dos anos para atender às necessidades de aumento da qualidade na produção desses produtos.

A demanda por mais qualidade, a diminuição de custos e a introdução de novos elementos tecnológicos forçam a melhoria do processo. Tais fatores não são recentes e, continuamente, também são vivenciados por outras áreas produtivas que, ao longo da história, vêm empregando diferentes técnicas para aperfeiçoar seus modelos de produção. No caso do desenvolvimento de software a área de pesquisa envolvida com a melhoria do processo de desenvolvimento é a Engenharia de Software ([BAUER, 1969](#); [IEEE, 1990](#); [IEEE, 2004](#)).

O primeiro modelo, explícito, de desenvolvimento de software, que veio propor a segmentação do trabalho, é conhecido como modelo cascata. Esse modelo permitiu uma visão estruturada sobre o desenvolvimento de software e sua estrutura foi utilizada para propor novos processos, assim que surgiam novas necessidades.

Existem outros modelos como de prototipação, incremental, evolucionário e espiral.

É interessante observar que, tipicamente, os modelos têm início com a Engenharia de Requisitos que, em termos gerais, é o processo de identificação dos envolvidos e suas necessidades de descoberta dos requisitos e de documentação, de forma que

seja útil para a análise, comunicação, e a subsequente implementação (NUSEIBEH; EASTERBROOK, 2000).

O fato da Engenharia de Requisitos ser a atividade inicial, nos diferentes modelos de desenvolvimento, pode ser analisado através das seguintes afirmações:

*“O principal objetivo da Engenharia de Requisitos é definir o propósito do sistema proposto e delimitar seu comportamento externo”* (ANTONIOU et al., 2000).

*“A medida primária do sucesso de um software está no grau em que ele atinge os requisitos para o qual foi intencionado”* (NUSEIBEH; EASTERBROOK, 2000).

A primeira afirmação trata sobre a definição de propósitos. Essa atividade é uma característica atribuída a atividades iniciais tanto de projetos quanto de metodologias de pesquisa (BASILI et al., 1994; SOLINGEN; BERGHOUT, 1999).

A segunda afirmação trata de atender a requisitos propostos. Se a necessidade de desenvolvimento de um software tem como base a resolução de um dado problema, saber precisamente qual a dimensão desse problema é essencial. Os requisitos são levantados inicialmente no projeto, e conhecê-los adequadamente é condição básica para as fases posteriores de construção do software.

Dessa forma, é possível compreender a importância do levantamento de requisitos e por que a Engenharia de Requisitos ocorre nas fases iniciais do processo de desenvolvimento.

As atividades que compõem a Engenharia de Requisitos podem ser divididas em cinco categorias: Elicitação<sup>1</sup>; Análise e Negociação; Documentação; Validação e Gerenciamento. A definição dessas categorias difere entre vários autores (JARKE; POHL, 1994; SAWYER et al., 1998; GRAHAM, 1998; KOTONYA; SOMMERVILLE, 2000; SOMMERVILLE, 1995) mas os conceitos são similares:

---

<sup>1</sup>Optou-se por utilizar o termo elicitación para a tradução do vocábulo da língua inglesa *elicitation* – que possui como significado: 1) eliciar, fazer sair 2) extrair resposta. O vocábulo, tanto em português quanto em inglês possui como origem o termo da língua latina *elicere*. O termo *elicitation* é de uso corrente na literatura de língua inglesa sobre engenharia de requisitos; o termo *elicitação*, por sua vez, possui como origem o verbo transitivo direto *elicitar* que se origina no particípio passado do latim *elicitus*, ‘extrair’, ‘tirar de’; ingl. *(to) elicit*. Os vocábulos *elicitar* e *elicitação* estão sendo utilizados com certa frequência nas publicações em língua portuguesa na área de engenharia de software (SERRANO, 1997; GOMES et al., 2003; BERTOLIN, 1998)

- Elicitação – É a atividade que envolve a descoberta dos requisitos do sistema. Nesta atividade os desenvolvedores do sistema trabalham junto com os clientes e os usuários para definir o problema a ser resolvido, os serviços que o sistema deverá prover, os requisitos de desempenho do sistema, as características de *hardware* e outros recursos inerentes ao sistema em questão. Existem três níveis de elicitação de requisitos: de negócio, de usuário e funcional.
- Análise e Negociação – As necessidades de usuários e clientes devem estar de acordo com as definições dos requisitos de software. Esta atividade serve para analisar em detalhes e resolver possíveis conflitos de requisitos entre os envolvidos com o sistema.
- Documentação – Os requisitos devem ser documentados a fim de servir de base para o restante do processo de desenvolvimento, essa documentação deve ser feita de forma consistente, seguindo-se um padrão que permita demonstrar, em vários níveis de detalhes, a especificação dos requisitos levantados.
- Validação – Esta atividade requer que a especificação dos requisitos do software seja validada com os requisitos do sistema. Isso quer dizer que serão checados para que se esclareça se possuem uma representação e descrição aceitável além da análise de propriedades como corretude, viabilidade e consistência.
- Gerenciamento – A atividade de gerenciamento de requisitos deve atender à manutenção da evolução dos requisitos ao longo do processo de desenvolvimento, ou seja, abrange todos os processos envolvidos em mudanças nos requisitos do sistema.

## 2.2 Gerenciamento de requisitos

Uma das maiores dificuldades para a Engenharia de Requisitos é o controle e a agregação de novos requisitos ao sistema. Isso ocorre devido ao impacto das propostas de alteração, à inflexibilidade dos processos e à dificuldade de assessorar essas mudanças.

A gerência de requisitos visa à resolução de tais problemas tendo como principais objetivos o gerenciamento das mudanças, o gerenciamento entre requisitos relaci-

onados e o gerenciamento das dependências entre a documentação de requisitos e outros documentos originados durante outros processos da engenharia de software (SOMMERVILLE; SAWYER, 1998).

Leffingwell e Widrig (2000) definem o gerenciamento de requisitos como um esforço sistemático para elicitar, organizar, e documentar um processo de engenharia de requisitos a fim de estabelecer acordo entre clientes, usuários e o grupo de desenvolvimento no que tange às mudanças dos requisitos de um sistema, tarefa árdua. Porém, a mudança dos requisitos, durante o desenvolvimento, é admitida como algo natural e inevitável, fato que se deve à própria natureza do software.

Dentro dessa ótica o gerenciamento pode ser visto como um processo paralelo de suporte a outras atividades da engenharia de requisitos (SOMMERVILLE; SAWYER, 1998; KOTONYA; SOMMERVILLE, 2000), sendo executado mesmo após a fase de especificação. O processo de engenharia de requisitos e a atividade de gerenciamento de requisitos, ocorrendo ao longo do processo, podem ser visualizados na Figura 2.1.

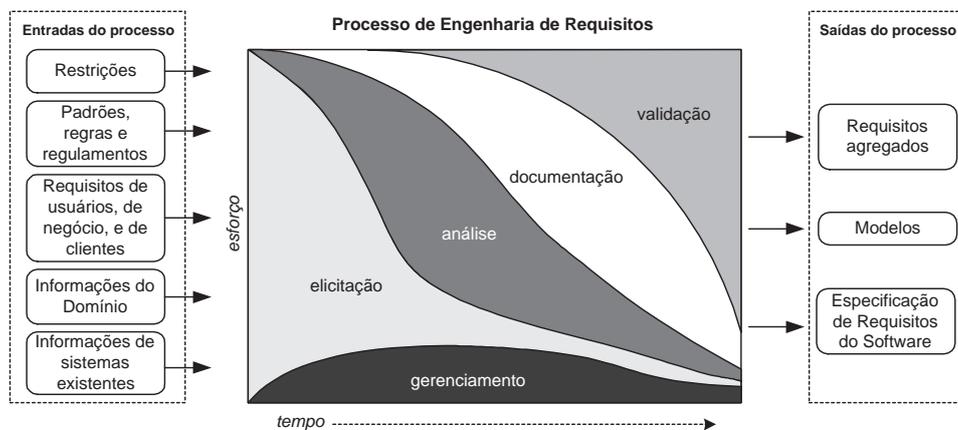


Figura 2.1 - Processo de engenharia de requisitos.

Fonte: Adaptado de NASA (1990) e Kotonya e Sommerville (2000)

O gerenciamento de requisitos pode ser dividido em três áreas específicas:

- a) Identificação – Atividades relacionadas à identificação e armazenamento de requisitos. As práticas de identificação de requisitos focam a assinatura única para cada requisito; basicamente, existem 4 técnicas para sua

execução: Esquema de Identificação Natural para Requisitos Hierárquicos, que identifica requisitos filhos à seus requisitos pais (LEFFINGWELL; WIDRIG, 2000); Renumeração Dinâmica, que permite automaticamente renumerar documentos e parágrafos quando informações são inseridas. Para isso utilizam-se editores de texto e referência cruzada (KOTONYA; SOMMERVILLE, 2000); Identificação de Registros em Base de Dados, cada requisito inserido na base de dados é identificado de forma única por ser uma entidade, podendo assim ser versionado, referenciado e gerenciado (KOTONYA; SOMMERVILLE, 2000); e Identificação Simbólica, técnica em que os requisitos são identificados usando-se nomes simbólicos relacionados ao contexto do requisitos, como, por exemplo, RF-1 (Requisito Funcional 1), RF-2, RF- $n$  (KOTONYA; SOMMERVILLE, 2000).

- b) Gerenciamento de mudanças – Atividades associadas a mudanças nos requisitos como análise de impacto, comunicação, mensuração de estabilidade, e incorporação de elementos ao sistema.

Os requisitos são inerentemente mudáveis ao longo do processo de desenvolvimento; gerenciar cuidadosamente essas mudanças pode prover um sistema com melhor utilidade, com custo e tempo aceitáveis (KOBAYASHI; MAEKAWA, 2001). Leffingwell e Widrig (2000) definem cinco etapas gerais para o processo de gerenciamento de mudanças: 1) reconhecer que o processo de mudança é inevitável e se planejar para isso, 2) Criar *baselines* para os requisitos, 3) estabelecer um canal simples para controlar as mudanças, 4) usar um sistema de controle de mudanças para capturá-las e 5) hierarquizar o gerenciamento de mudanças. Além de etapas gerais existem vários autores que definem métodos para lidar com o gerenciamento de mudanças de requisitos, como Crnkovic et al. (1999) que propõem suporte ao gerenciamento de mudanças durante o ciclo de vida do produto, Lindsay et al. (1997) que enfatizam o suporte em granularidade fina, Lam et al. (1999) que introduzem o uso de métricas para o suporte, sendo a mensuração a atividade central das decisões do gerenciamento, e Kobayashi e Maekawa (2001) que criaram o modelo NRM (do inglês *Need-based Requirements Management*) baseado nas necessidades dos interessados no projeto e por analisar essas necessidades dentro de quatro contextos: onde, quem, porque e o quê.

- c) Rastreabilidade – Atividades de rastreabilidade como a definição de elos

entre requisitos e outros artefatos produzidos pelo processo de desenvolvimento.

A rastreabilidade é o foco principal deste trabalho e será explorada detalhadamente nas próximas seções. A fim de concluir a apresentação sobre o ponto no qual a rastreabilidade se encontra na Engenharia de Software, a [Figura 2.2](#) apresenta sua alocação dentro dessa área de pesquisa (não são consideradas as outras áreas da Engenharia de Software).

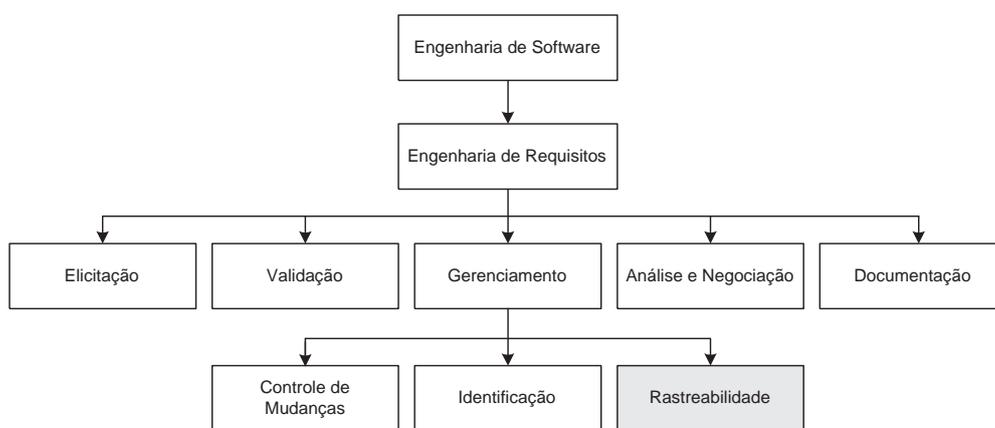


Figura 2.2 - Alocação da rastreabilidade dentro da engenharia de software.

### 2.3 Rastreabilidade

A rastreabilidade de requisitos tem seu início desde a elicitação avançando através do desenvolvimento do sistema até a manutenção e constitui a principal atividade do gerenciamento de requisitos ([NUSEIBEH; EASTERBROOK, 2000](#)).

A rastreabilidade está intimamente associada ao processo de produção de software, especificamente aos requisitos e à capacidade de estabelecer vínculos entre esses requisitos e outros artefatos (modelos, documentos, código fonte, seqüências de testes ou executáveis) que os satisfaçam. Essa característica é observada por [Letelier \(2002\)](#) que afirma que a rastreabilidade de requisitos ajuda a garantir uma contínua concordância entre os requisitos dos interessados no sistema e os artefatos produzidos ao longo do processo de desenvolvimento. Na mesma linha, [Palmer \(2000\)](#) afirma que a rastreabilidade dá assistência essencial para a compreensão do relacionamento que existe com e entre requisitos de software, projeto e implementação. Esses relacionamentos auxiliam o projetista a mostrar quais elementos do projeto satisfazem

os requisitos.

Sobre as vantagens da rastreabilidade é observado que seu uso ajuda a estimar variações em cronogramas e em custos do projeto (PALMER, 2000; DAVIS, 1993; DÖMGES; POHL, 1998; PINHEIRO, 2004). Além disso, a rastreabilidade pode auxiliar gerentes de projeto a: verificar a alocação de requisitos a componentes de software; resolver conflitos entre requisitos; verificar requisitos nos processos de testes; corrigir defeitos através da identificação do componente de origem do erro; validar o sistema junto aos clientes; analisar o impacto na evolução dos sistemas; prever custos e prazos; e gerenciar riscos e reuso de componentes (SAYÃO; LEITE, 2004).

Dependendo de sua semântica, a rastreabilidade também pode ser usada para: (a) assistir o processo de verificação dos requisitos para um sistema, (b) estabelecer o impacto de mudanças na especificação de requisitos através de seus artefatos ou da documentação (Ex. projeto, teste e implementação de artefatos), (c) compreender a evolução de um artefato, (d) compreender os aspectos do projeto, e (e) dar suporte à captura das razões por trás das decisões de projeto, área comumente definida como *Design Rationales*<sup>2</sup> (TANG et al., 2007). Dessa forma, a geração e a manutenção de tais relações podem fornecer uma base mais eficaz para a garantia da qualidade do sistema, a gerência das mudanças e a manutenção do software (SPANOUKAKIS et al., 2004).

### 2.3.1 Classificação para a rastreabilidade

A capacidade de rastrear um requisito até seus refinamentos é definida como rastrear para frente (*Forwards*), e a de rastrear um refinamento até sua origem é definida como rastrear para trás (*Backwards*) (DAVIS, 1993). Essas duas capacidades devem estar presentes em todos os tipos de rastreabilidade, ou seja, é uma propriedade básica para a realização completa das funções dessa atividade. Um processo de rastreabilidade é falho se não executa uma das duas capacidades.

Sobre os tipos de rastreabilidade basicamente existem duas classificações gerais: i) rastreabilidade horizontal e vertical, e ii) pré e pós-rastreabilidade.

A rastreabilidade horizontal é a rastreabilidade entre diferentes versões ou variações de requisitos, ou outros artefatos, em uma particular fase do ciclo de vida. Enquanto

---

<sup>2</sup>Adotou-se para este trabalho o termo da lingua inglesa *Rationale* devido à falta de trabalhos científicos em lingua portuguesa que obordem ou traduzam o termo.

que a rastreabilidade vertical é realizada entre requisitos e artefatos produzidos pelo processo de desenvolvimento ao longo do ciclo de vida do projeto (BELFORD et al., 1976; RAMESH; EDWARDS, 1993; GOTEL, 1995). Uma visão simplificada sobre esses dois tipos de rastreabilidade é apresentada na Figura 2.3.

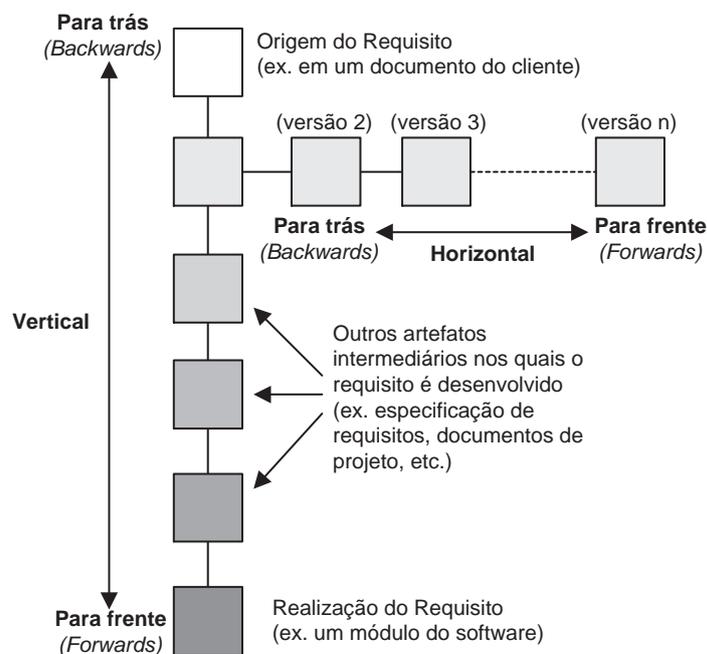


Figura 2.3 - Rastreabilidade Horizontal e Vertical.

Fonte: Adaptada de Gotel (1995)

Kotonya e Sommerville (2000) enfatizam a direção da rastreabilidade, para frente e para trás, estendendo a rastreabilidade vertical com os atributos ‘para’ (do inglês *to*) e ‘de’ (do inglês *from*), além de apontar a rastreabilidade entre requisitos e outros artefatos produzidos pelo processo de desenvolvimento. Esses conceitos reforçam a natureza bi-direcional da rastreabilidade (KOTONYA; SOMMERVILLE, 2000; DAVIS, 1993):

- Rastreabilidade de frente-para (*Forward-to traceability*): rastreabilidade de origens (requisitos de clientes, requisitos no nível de sistema, etc.) para requisitos.
- Rastreabilidade de frente-de (*Forward-from traceability*): rastreabilidade de requisitos para especificações de projeto.

- Rastreabilidade de trás-para (*Backward-to traceability*): rastreabilidade de especificações de projeto para requisitos.
- Rastreabilidade de trás-de (*Backward-from traceability*): rastreabilidade de requisitos para suas origens (requisitos de clientes, requisitos no nível de sistema, etc.).

A segunda classe de rastreabilidade trata da pré-rastreabilidade, que está concentrada no ciclo de vida dos requisitos antes de serem incluídos na especificação de requisitos; e a pós-rastreabilidade, que está concentrada no ciclo de vida dos requisitos após serem incluídos na especificação de requisitos (GOTEL; FINKELSTEIN, 1994).

A Figura 2.4 ilustra a pré e a pós-rastreabilidade; pode-se observar que o conhecimento sobre os requisitos está distribuído e inserido em sucessivas representações assim como a complicação sobre as iterações e a propagação de mudanças.

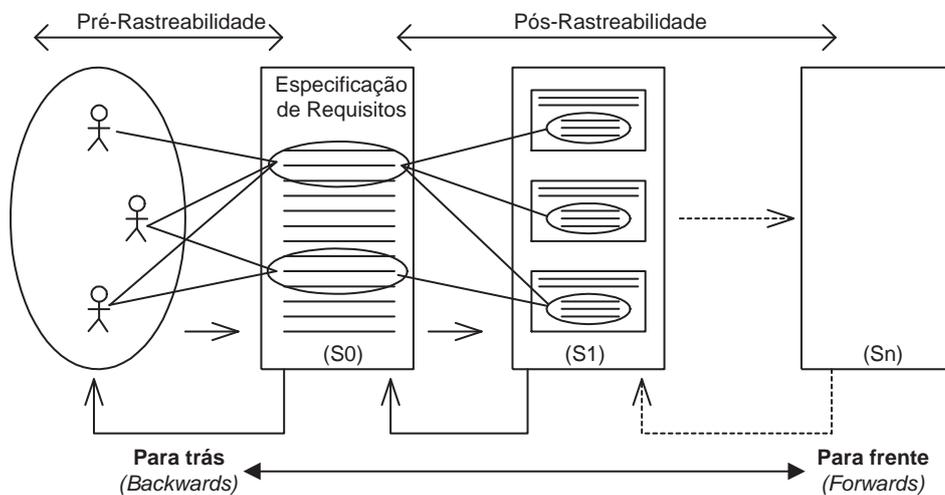


Figura 2.4 - Pré e Pós-Rastreabilidade.  
Fonte: Adaptado de Gotel (1995)

Gotel e Finkelstein (1994) enfatizam a separação entre a pré e a pós-Rastreabilidade, embora necessárias, torna-se importante compreender suas diferenças, pois cada um dos tipos impõe suporte distinto aos requisitos.

A principal diferença entre a pós e a pré-Rastreabilidade, envolve as informações com que elas podem lidar.

A pós depende da habilidade de rastrear requisitos para frente e para trás de um ponto de referência (*baseline*), neste caso, a especificação de requisitos (Figura 2.4), através de uma sucessão de artefatos nos quais os requisitos estão relacionados. Mudanças na *baseline* necessitam ser propagadas através desses artefatos.

A pré-rastreabilidade depende da habilidade para rastrear requisitos, para frente e para trás, de seus padrões originais ou fontes (clientes, usuários, normas, etc.), através dos artefatos gerados pelo processo de engenharia de requisitos (requisitos funcionais, não-funcionais, validações, documentos de análise e acordo, etc.). Esses requisitos são eventualmente integrados em uma especificação de requisitos relativamente simples. Mudanças no processo de produção de requisitos precisam ser refletidas na especificação de requisitos, enquanto que mudanças na especificação necessitam ser realizadas com referência a este processo, podendo ser propagadas para suas fontes. Isso requer visibilidade para os inter-relacionamentos sutis que existem entre os requisitos iniciais do projeto (GOTEL; FINKELSTEIN, 1994).

### 2.3.2 Técnicas de rastreabilidade

O estabelecimento, a manutenção e a representação dos relacionamentos, existentes nas diferentes classes de rastreabilidade, são realizados através de alguma técnica de rastreabilidade. O desenvolvimento e o uso dessas técnicas tiveram início na década de 1970 (AIZENBUD-RESHEF et al., 2006) e o primeiro método usado para expressar e manter a rastreabilidade foi a referência cruzada (EVANS, 1989).

Outras técnicas podem ser usadas tanto para representar como para estabelecer relacionamentos incluindo-se matrizes (DAVIS, 1993; WEST, 1991); dependência de frases chaves (JACKSON, 1991); integração de documentos (LEFERING, 1993); hipertexto (ALEXANDER, 2002; GLINZ, 2000; KAINDL, 1993), grafos (PINHEIRO; GOGUEN, 1996); métodos formais (COOKE; STONE, 1991); esquemas dinâmicos (CLELAND-HUANG; CHRISTENSEN, 2003; ANTONIOL et al., 2000; TRYGGESETH; NYTRØ, 1997); sistemas baseados em suposição da manutenção de verdade (TANG, 1997; SMITHERS et al., 1991); e redes de confiança (BOWEN et al., 1990).

Técnicas automatizadas e semi-automatizadas de rastreabilidade também foram desenvolvidas, como a detecção de existência de elos entre documentos de requisitos,

documentos de projeto, e código fonte usando-se uma máquina de aprendizagem (FAISAL, 2005); técnicas de análise altamente escaláveis para análise automática de consistência entre requisitos e projetos (CHECHIK; GANNON, 2001); técnicas para recuperação de informação – RI (do inglês *Information Retrieval*) (JUNG, 2007; LANCASTER, 1968; Van RIJSBERGEN, 1979) como a indexação por análise da semântica latente (De LUCIA et al., 2004; DEERWESTER et al., 1990) que são usadas para recuperar elos de rastreabilidade ou na recuperação por construção de elos sobre a formalização de semânticas (DENG et al., 2005).

Pesquisas vêm mostrando que as técnicas baseadas em RI podem ser usadas para descobrir dinamicamente a existência de elos (ANTONIOLO 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; ANTONIOLO et al., 2002), requisitos e projeto (HAYES et al., 2003), artefatos e requisitos (ZISMAN et al., 2003). Outra característica dessas técnicas é que elas podem prover a rastreabilidade sem a necessidade de manter armazenados os elos.

A técnica conhecida como modelo de reflexão de software (MURPHY et al., 2001) 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 (2004), que usa uma síntese automatizada para inferir elos. Perturbações no artefato de origem são analisadas e usadas para identificar os elos entre os artefatos de origem e destino.

Além da rastreabilidade entre artefatos, algumas técnicas, como a rastreabilidade baseada em eventos (CLELAND-HUANG; CHRISTENSEN, 2003), podem ser usadas para rastrear requisitos associados a desempenho em modelos de desempenho executáveis (CLELAND-HUANG; CHRISTENSEN, 2003; CLELAND-HUANG et al., 2002; CLELAND-HUANG et al., 2003), 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).

### 2.3.3 Elos de rastreabilidade

O principal recurso utilizado para manter e representar os relacionamentos da rastreabilidade é o elo (do inglês *link*).

Várias propostas têm sido feitas para elos de rastreabilidade assim como para modelos que suportem: (i) algum padrão focado na manutenção do processo e (ii) a representação desses elos.

Basicamente os elos estão associados a uma metodologia ou a alguma informação do domínio do processo de desenvolvimento. Nesse aspecto, elos foram criados para orientação a objetos (SPANOUKAKIS *et al.*, 2004), orientação a aspectos (EGYED, 2004), desenvolvimento centrado em visão (SABETZADEH; EASTERBROOK, 2005), e desenvolvimento dirigido a modelo (AIZENBUD-RESHEF *et al.*, 2006; ALMEIDA *et al.*, 2006). Além disso, elos podem ser criados por aspectos ambientais e organizacionais (RAMESH; JARKE, 2001); no tipo de informação a ser rastreada (TORANZO *et al.*, 2002); ou ainda por configurações que integram especificações textuais (LETELIER, 2002). O principal problema desses elos é que eles são criados e usados somente para um propósito específico.

No caso do processo de engenharia de requisitos, os elos são usados nas atividades de: validação, análise, evolução e referência cruzada entre requisitos e artefatos (PALMER, 2000). Também fazem uso desse recurso os processos de gerência de projeto (auxiliar a predição de custo, restrições ou impactos); de manutenção; de teste (geração de casos de teste baseados em cenários ou modelos); e no processo da garantia da qualidade (validação do software)(CLELAND-HUANG; CHRISTENSEN, 2003; PINHEIRO; GOGUEN, 1996; PALMER, 2000; GOTEL; FINKELSTEIN, 1994; JARKE, 1998; PAPADOPOULOU, 2002; RIEBISCH; HUBNER, 2005).

Os elos podem ser formalizados como:  $A = \{a_1, a_2, a_3, \dots, a_n\}$  representa todos os artefatos produzidos e identificados no processo de desenvolvimento;  $E = \{e_1, e_2, e_3, \dots, e_n\} \subset A$  é um subgrupo dos artefatos que representam elos, e  $R = \{r_1, r_2, r_3, \dots, r_n\} \subset A$  é um subgrupo de  $A$  que representa os requisitos.

Um elo representa explicitamente o relacionamento definido entre dois artefatos  $a_1$  e  $a_2$ .  $a_1 \rightarrow a_2$  são considerados como diretamente ligados enquanto um elo indireto usa um artefato intermediário como em  $a_1 \rightarrow a_2 \rightarrow a_3$  (CLELAND-HUANG; CHRISTENSEN,

SEN, 2003). Um elo é estabelecido entre um artefato origem e um artefato destino, Figura 2.5.

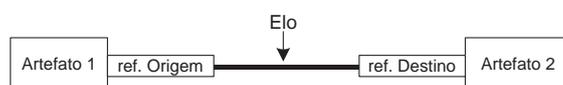


Figura 2.5 - Elementos básicos de um elo.

Informações sobre a origem e o destino são suficientes para suportar a rastreabilidade para frente e para trás, mas outras atividades da engenharia de requisitos, como o auxílio à previsão de custos e prazos e a resolução de conflitos, necessitam de outras informações para sua execução (DICK, 2002; DICK, 2005).

Várias categorias de elos podem ser determinadas usando-se como base os atributos e propriedades ou a aplicação desses elos no processo de desenvolvimento. Devido a essas características existem muitos tipos de elos descritos na literatura.

Ramesh e Jarke (2001) definem quatro tipos de elos: Satisfação, Dependência, Evolução e *Rationales*. Satisfação e Dependência formam um grupo chamado relacionado ao produto, que descreve propriedades e relacionamentos aos objetos, enquanto Evolução e *Rationales* compõem um segundo grupo chamado relacionado ao processo que pode ser capturado somente se observado o histórico das ações feitas no processo de rastreabilidade desses próprios elos.

Toranzo et al. (2002) propõem seis tipos: Satisfação – indica que o elemento de origem tem dependência de satisfação com a classe destino; Recurso – a classe de origem tem dependência de recurso com a classe destino; 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 o relacionamento entre a classe de origem e destino, sendo esta última um subsistema; e Agregação – representa a composição entre elementos.

Pohl (1996), em seu modelo de dependência, define dezoito diferentes elos de dependência (DAHLSTEDT; PERSSON, 2003) que são categorizados em cinco classes: Condição, Conteúdo, Documentos, Evolução e Abstração.

De Lucia et al. (2005) apresentam três tipos de elos: Dependência – um elo direto entre um artefato escravo e um artefato mestre; o artefato escravo depende ou é impactado por mudanças no artefato mestre; Indiretos – artefatos que impactam uns aos outros; e Composição – um artefato mestre é parte de um artefato escravo, os quais são recuperados por uma ferramenta de recuperação mas não rastreados por usuários.

Spanoudakis et al. (2004) baseados em estudos da documentação de sistemas de software identificaram quatro tipos de elos entre padrões de requisitos, casos de uso e modelos de análise de objetos: Elo de Sobreposição – denota que os elementos conectados consultam uma característica comum do sistema subjacente ou de seu domínio. Elo Execução\_Requerida – denota que a sequência dos termos do artefato requer a execução da operação a que se relaciona. Elo Característica\_Requerida – indica a relação entre uma parte de uma especificação de um caso do uso e um requisito ou entre dois requisitos. Elo Parcialmente\_Realizável – indica que o caso de uso executa parcialmente um determinado requisito.

Aizenbud-Reshef et al. (2006) definem quatro tipos de elos: 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 atrás de um grupo de regras.

Um resumo dos tipos de elos apresentados pode ser visualizado na [Tabela 2.1](#).

Tabela 2.1 - Exemplos de elos descritos na literatura classificados por seus respectivos autores e sub-grupos.

<b>Autor</b>	<b>Grupos</b>	<b>Tipos de elo</b>	
Ramesh e Jarke (2001)	Relacionado ao produto	Satisfação Dependência	
	Relacionado ao processo	Evolução <i>Rationales</i>	
Toranzo et al. (2002)		Satisfação Recurso Responsabilidade Representação Alocação Agregação	
	Condição	Restrições Pré-condições	
		Documentos	Exemplos Propósito Caso de teste Comentários Segundo Plano
	Abstração	Refinado Generalizado	
	Evolução	Elaborado Formalizado Baseado em Satisfação Substituído	
Conteúdo	Similar Comparação Contradição Conflito		
	De Lucia et al. (2005)	Dependência Composição Indireto	
		Spanoudakis et al. (2004)	Sobreposição Execução_Requerida Característica_Requerida Parcialmente_Realizável
			Aizenbud-Reshef et al. (2006)
Computacional	Derivação Análise		

### 2.3.4 Métricas para rastreabilidade

As métricas, de maneira geral, são recursos que podem ser utilizados para trazer informações úteis para o acompanhamento, gerenciamento e controle do processo de desenvolvimento (COSTELLO; LIU, 1995). Elas também provêm informações quan-

titativas para a realização dessas tarefas ([NORMAN; MARTIN, 2000](#)), além de poderem ser utilizadas para aperfeiçoar a produtividade e a qualidade tanto do produto quanto do processo ([SEI, 1988](#)).

Segundo [Roche \(1994\)](#), as métricas de software estão inseridas no contexto da mensuração do software e podem ser exploradas dentro dos seguintes tópicos:

- Evolução de desempenho e de modelos;
- Complexidade computacional e algorítmica;
- Estimação de custo e esforço;
- Mensuração de produtividade e de modelos;
- Qualidade;
- Mensuração estrutural e de complexidade;
- Mensuração de projeto;
- Modelos de confiabilidade.

Especificamente as métricas para rastreabilidade usam dados sobre os elos e outras técnicas de representação para mostrar como uma organização mantém a mensuração sobre o relacionamento dos requisitos. As informações obtidas podem determinar se todos os relacionamentos requeridos por uma dependência são atendidos, expondo especificações incompletas ou excessivamente complexas para o sistema, e também podem ser utilizadas para evitar interpretações incorretas de outras métricas. Além disso, elas podem ser usadas para expor o surgimento de requisitos de baixo nível com origens não válidas no nível de sistema ([COSTELLO; LIU, 1995](#)).

[Costello e Liu \(1995\)](#) definem cinco tipos de métricas: cobertura de próximo nível (COV), profundidade plena e alta cobertura (DHCOV), vinculação estatística, rastreabilidade inconsistente (ITM), e rastreabilidade indefinida (UTM).

As métricas COV incluem o número de requisitos que são rastreados consistentemente para o próximo nível acima (COVup), o número de requisitos rastreados consistentemente para o próximo nível baixo (COVdown), bem como o número de requisitos rastreados consistentemente para o próximo nível em ambos os sentidos

(COV). Métricas DHCov envolvem uma análise semelhante, mas avaliam a rastreabilidade em níveis de especificação mais altos (HCov) e mais baixos (DCov). Note-se que se os sistemas possuem três ou mais níveis de especificação o COV e DHCov serão idênticos. Vinculação estatística – mede a complexidade dos dados da rastreabilidade fornecendo a contagem dos requisitos de baixo e alto nível para o qual cada requisito de uma especificação é rastreado. ITM inclui os números de requisitos em uma especificação que tenham pelo menos um elo inconsistente ascendente (ITMup) e descendente (TIMdown). Por fim, a rastreabilidade indefinida (UTM) que inclui o número de requisitos em uma especificação que não tenha nenhum elo ascendente e descendente (UTMup)(UTMdown).

[Hull et al. \(2002\)](#) também estabelecem uma discussão sobre métricas para rastreabilidade. Tendo como base o relacionamento do tipo satisfação, e movendo-se para baixo através dos requisitos, os autores definem três dimensões e dois outros elementos para a mensuração da rastreabilidade.

Para determinar quais das três dimensões são úteis, em termos de medição do processo de engenharia de requisitos, é necessário distinguir entre dois tipos de indicadores: i) métricas de camadas: medições relativas a uma única fase do desenvolvimento, por exemplo, apenas para a camada dos requisitos do sistema; e ii) métricas globais: medições abrangendo várias fases de desenvolvimento ([HULL et al., 2002](#)).

A seguir são apresentadas as três dimensões e os outros dois elementos:

- Dimensões:
  - a) Largura – Diz respeito à cobertura e, como tal, é uma camada métrica. A cobertura pode ser utilizada para medir o progresso do processo que cria a rastreabilidade em uma única camada. Avalia a extensão na qual os requisitos são cobertos pelas camadas adjacentes, acima, abaixo ou ao lado.
  - b) Profundidade – É focada no número de camadas que a rastreabilidade estende para cima ou para baixo a partir de um determinado nível, o que faz dela uma métrica global.
  - c) Crescimento – Está relacionada com o impacto de mudanças, examinando de que maneira alguns requisitos, em níveis mais baixos, estão relacionados a um único requisito em um nível mais alto.

- Outros elementos:
  - a) Balanço – Seu foco está na observação da distribuição de fatores de crescimento para requisitos individuais entre duas camadas, examinando aqueles que estão no exterior dos quartis da distribuição. O objetivo consiste em identificar os requisitos, que têm alguma anormalidade, tanto alta quanto baixa, no fator de crescimento, e submetê-los a um escrutínio especial.
  - b) Mudança Latente – A mobilização de um único pedido de mudança pode introduzir uma sequência de possíveis mudanças latentes no sistema. Em tais circunstâncias seria altamente desejável acompanhar os progressos e estimar o trabalho consequente.

As métricas para rastreabilidade apresentadas têm por objetivo propiciar a melhoria da qualidade dos requisitos rastreados, o que é muito apropriado. Porém, um olhar mais crítico, sobre as métricas existentes, pode apontar a falta de preocupação com a qualidade de elos, item que pode ser útil para outras atividades da engenharia de requisitos.

Todas as métricas propostas obtêm informações da rastreabilidade para determinar a qualidade de requisitos, enquanto que a qualidade da rastreabilidade em si não é avaliada.

Ao considerar-se as normas existentes para qualidade de software segue-se um padrão de refinamento partindo-se de elementos mais amplos do processo de desenvolvimento até que seus produtos ou processos atinjam os elementos em níveis mais refinados. Um exemplo disso pode ser observado nas normas NBR ISO/IEC 12207 (ABNT/NBR, 1998), que trata sobre o ciclo de vida do processo de desenvolvimento; na norma IEEE-830 (IEEE, 1998), que trata sobre a especificação de requisitos de software; e na norma IEEE-1233 (IEEE, 1998), que aborda o desenvolvimento da especificação de requisitos para sistemas.

O mesmo padrão de refinamento não ocorre com a rastreabilidade, ou seja, o principal elemento de manutenção e representação de relacionamentos, o elo, utilizado no nível mais baixo do processo de gerenciamento de requisitos, não é explorado no quesito qualidade sendo apenas utilizado como instrumento para aferir a qualidade de elementos de níveis superiores.

Apesar da importância da rastreabilidade ainda é evidente a falta de padrões para a sua qualidade, fato que pode ser observado nas normas e modelos como o CMMI-DEV (SEI, 2006) que possui a prática específica 1.4 – Manter a rastreabilidade bidirecional para os requisitos, ou a norma ISO/IEC 15504-2 (ISO/IEC, 2003), que possui as práticas base: ENG. 3.4 – Estabelecer a rastreabilidade e PRO. 4.5 – Manter a rastreabilidade. Nesses dois casos propõe-se apenas que a rastreabilidade seja executada, mas nenhum padrão para a qualidade da rastreabilidade é estabelecido. Como afirmado por Angelina et al. (2006), os padrões e normas não incluem métricas de rastreabilidade para garantir a consistência, a completude, a confiabilidade, a usabilidade e a eficiência da implementação da rastreabilidade.

### 2.3.5 Ferramentas para rastreabilidade

Existe uma grande quantidade de ferramentas comerciais que variam em seu nível de suporte para as atividades do gerenciamento de requisitos. A maioria dessas ferramentas fornece uma visão orientada a texto no gerenciamento de requisitos (tal como DOORS, RTM, Document Director e outros) e possuem pouca integração com o processo de desenvolvimento (TORANZO et al., 2002; SPANOUDAKIS, 2002; ALEXANDER, 2003). Algumas exceções sobre suporte para o processo é a ferramenta CORE que fornece uma execução orientada ao processo de engenharia de requisitos.

De 1998 até 2008 o INCONSE (do inglês *International Council on Systems Engineering*) (INCONSE, 2008) apresenta uma comparação entre os mais importantes pacotes de software para sistemas de gerenciamento de requisitos. A seguir são descritas algumas das ferramentas analisadas pelo INCONSE.

CALIBER – Ferramenta de gerenciamento de requisitos elaborada para ambiente WEB. Contém recursos para manutenção e rastreamento de alterações de requisitos e elaboração de relatórios.

CORE – Baseado no desenvolvimento em equipe, utiliza uma linguagem de especificação formal RSL e suporta várias técnicas de diagramação (ER, *hierarchy*, FFBD e N2). Também permite fazer elos de documentos externos com seus modelos, ajuda a verificar as especificações formais e fornecer rastreabilidade.

RDT – Foi projetado para capturar o processo e gerenciar os requisitos do sistema. O processo de gerenciamento de requisitos é controlado a partir do tempo de produção de uma proposta de projeto.

DOORS – É um sistema de gerenciamento de requisitos orientado a objetos. Ele foi projetado para gerenciar componentes do projeto aos requisitos. Seu principal papel é gerenciar, rastrear e navegar elos entre muitos tipos de dados, de forma que seja relevante ao processo de requisitos.

RequisitePRO – É um sistema cooperativo como grupo de trabalho para gerenciamento de requisitos. O sistema adapta-se à suite de outros produtos e permite criar documentos de requisitos de multimídia com o Microsoft Word.

A [Tabela 2.2](#) é um resumo da pesquisa do INCONSE ([INCONSE, 2008](#)) sobre as ferramentas descritas. São apresentadas apenas as atividades iniciais da captura e análise de requisitos. Essa tabela mostra que a maioria das ferramentas analisadas tem capacidades para identificar requisitos, através de análise, detecção de palavra-chave, classificação (automática ou semi-automática), visualização (em texto e gráfica), alocação (para processamento adicional), melhoramento (através da anotação (explicação/observação) e inter-relacionamentos com outros documentos) e rastreabilidade (que inclui verificação parcial e referência cruzada).

Tabela 2.2 - Comparação entre algumas ferramentas de gerenciamento de requisitos.

Legenda:	CaliberRM	CORE	DOORS	RDT	RequisitePRO
■ Característica completamente suportada					
□ Característica parcialmente suportada					
⊗ Característica não suportada					
<b>1. Captura/Identificação de Requisitos</b>					
1.1. Enriquecimento de documentação	■	■	■	■	■
1.1.1. Mudança na documentação	■	■	□	□	■
1.2. Analisador gramatical automático de requisitos	⊗	■	■	■	□
1.3. Identificador semi-automático de requisitos	■	■	■	■	□
1.4. Identificador manual de requisitos	■	■	■	■	■
1.5. Modo de operação em lote (do inglês <i>batch</i> )	⊗	■	■	□	■
1.5.1. Modo em lote de atualização de vínculos	■	■	□	□	■
1.6. Classificação de requisitos	■	■	■	■	■
<b>2. Captura de elementos estruturais do sistema</b>					
2.1. Captura gráfica de elementos estruturais	■	■	■	■	■
2.2. Captura textual elementos estruturais	■	■	■	■	■
<b>3. Fluxo inverso de requisitos</b>					
3.1. Derivação de requisitos	■	■	■	■	■
3.2. Alocação de desempenho aos requisitos	■	■	■	■	■
3.3. Vínculo de requisitos com elementos do sistema	■	■	■	■	■
3.4. Anotação de requisitos	■	■	■	■	■
<b>4. Análise de rastreabilidade</b>					
4.1. Identificação de inconsistências	■	■	■	■	■
4.2. Visibilidade de vínculos com a implementação	■	■	■	■	■
4.3. Verificação de requisitos	■	■	■	■	■
4.4. Verificação de desempenho de elementos do sistema	■	■	■	■	⊗

Fonte: Adaptado de [INCONSE \(2008\)](#)

### 2.3.6 Modelos para rastreabilidade

A principal área de pesquisa para a rastreabilidade abordada neste trabalho são os modelos. Segundo [Mellor et al. \(2003\)](#), um modelo é um grupo coerente de elementos que descrevem algo construído através de alguma forma de análise para algum propósito particular, podendo ser expressado por uma linguagem (textual ou gráfica) que por sua vez possui um certo grau de abstração.

No caso da rastreabilidade, os modelos são desenvolvidos com base nas informações sobre um determinado domínio desse processo (usuários, práticas, metodologias, normas, etc.).

Vários autores fazem uso de modelos para representar conceitos relacionados à ras-

treabilidade como a definição de diferentes tipos de elos.

Ramesh e Jarke (2001) criaram um metamodelo para rastreabilidade, Figura 2.6, utilizando como base uma extensa pesquisa empírica sobre as consequências dos diferentes usos, perspectivas e necessidades de informação dos envolvidos no processo de desenvolvimento.

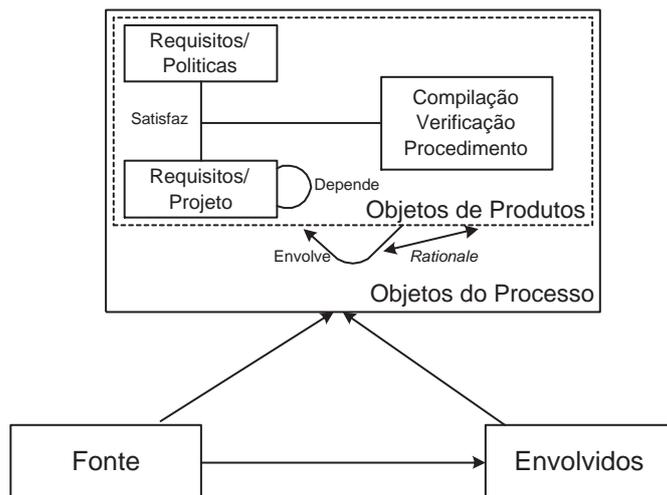


Figura 2.6 - Metamodelo para rastreabilidade de requisitos proposto por Ramesh e Jarke (2001).

Fonte: Adaptado de Ramesh e Jarke (2001)

Os autores encontraram três categorias de informações associadas à rastreabilidade que foram utilizadas para estabelecer o metamodelo: envolvidos; fontes (padrões, normas, regras institucionais, etc.); e objetos (objetos conceituais, modelos, ou artefatos).

Além das três classes de informação a pesquisa permitiu aos autores criar uma classificação dos usuários da rastreabilidade. Esses foram divididos, quanto a suas práticas, em dois grupos distintos: usuários sofisticados e normais (do inglês *high-end e low-end users*). Com base nessa classificação dois modelos foram criados para representar seus processos de rastreabilidade.

Adicionalmente à criação do metamodelo, conceitualmente simples mas muito amplo em seus elementos e elos, o trabalho de Ramesh e Jarke (2001) apresenta uma classificação muito interessante sobre os usuários da rastreabilidade, dando grande

valor às práticas utilizadas por estes nas organizações. Entretanto, esse modelo pre-estabelece os tipos de elos a serem utilizados, permitindo apenas a divisão das classes preestabelecidas. Outro problema encontrado no modelo é que ele não possibilita a adição de atributos aos elos, o que permitiria o enriquecimento da semântica desses artefatos, apesar da existência do tipo de elo *Rationale* que é importante para determinar as razões da criação ou alteração de artefatos.

O modelo de [Toranzo et al. \(2002\)](#) faz uso de representação gráfica através de diagramas de classes da UML ([Figura 2.7](#)). Esse modelo tem como conceito a classificação das informações a serem rastreadas, que são divididas em quatro classes: Ambientais, Organizacionais, Gerenciais e de Desenvolvimento. Com base na classificação, os autores desenvolveram um metamodelo; um modelo intermediário e um processo para aplicação das estratégias anteriores sobre a rastreabilidade. É dada atenção a aspectos gerenciais, como a definição do elo do tipo responsabilidade. Esse aspecto é notado pela influência dos contextos da classificação definida (Ambientais, Organizacionais, Gerenciais e de Desenvolvimento). Entretanto, os tipos de elos também são predefinidos sendo focados apenas nos contextos utilizados no modelo; devido a essa característica outros padrões de gerência ou de rastreabilidade, como dirigida à visão ou a modelos, poderão não ser completamente expressos.

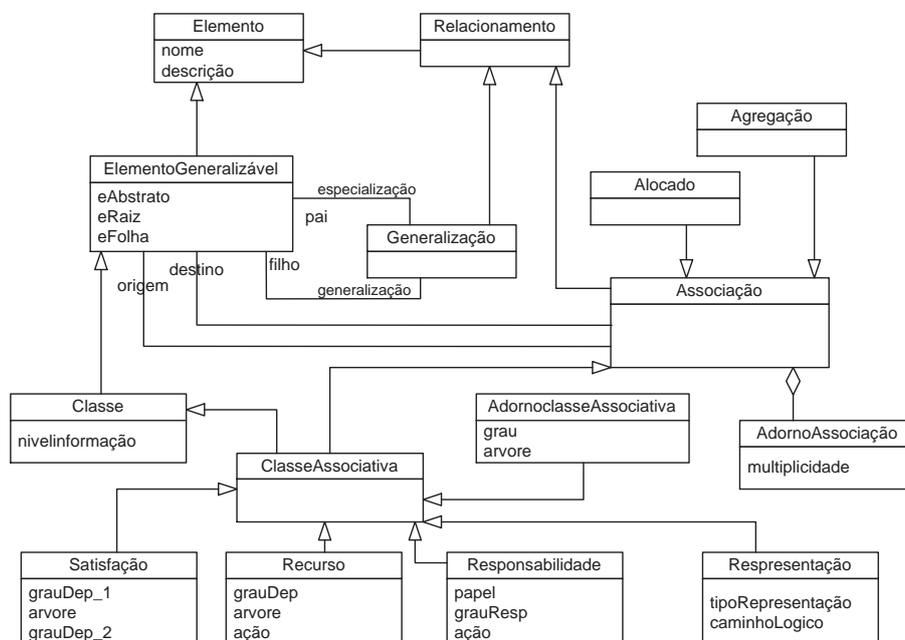


Figura 2.7 - Principais elementos do modelo de [Toranzo et al. \(2002\)](#)  
 Fonte: Adaptado de [Toranzo et al. \(2002\)](#)

Uma discussão sobre o problema da predefinição de elos nos modelos de rastreabilidade é apresentada por [Aizenbud-Reshef et al. \(2006\)](#). Os autores propõem uma solução, para este problema, usando o conceito de *Model Driven Architecture* – MDA. Esta proposta considera que a criação de modelos para sistemas deve incluir um metamodelo de requisitos; os modelos resultantes, que incluem um modelo explícito de requisitos, podem prover a rastreabilidade entre requisitos e outros artefatos. A existência de tais modelos não provê mecanismos que dispõem como produzir, automaticamente, os elos entre requisitos e suas dependências, ou seja, segundo os autores, o fato de se dispor de uma semântica para produzir elos significativos não garante uma efetiva estratégia de rastreabilidade. Assim os elos necessitam ser construídos e mantidos manualmente.

Segundo [Aizenbud-Reshef et al. \(2006\)](#), o ideal seria gerar os artefatos, ou suas estruturas básicas, e produzir os elos entre eles e, se necessário, os elos seriam detalhados. Para tanto, a MDA poderia ser uma alternativa para realizar essas tarefas.

Apesar da crítica sobre os modelos de rastreabilidade que predefinem os tipos de elos, a proposta de [Aizenbud-Reshef et al. \(2006\)](#) está vinculada a uma solução baseada em arquitetura, o que pode ser um problema para processos de desenvolvimento baseados em outros conceitos. Entretanto, esta proposta pode ser muito relevante para os envolvidos com a MDA.

## 2.4 Discussão do problema

Sobre os modelos de rastreabilidade existentes, e apresentados neste trabalho, é possível observar que eles provêm padrões predefinidos de grupos de elos que podem ser criados pelas suas respectivas soluções apresentadas. Esses trabalhos fazem extensas observações das práticas da rastreabilidade, mas seus modelos são limitados quanto à fixação dos tipos de elos, ou seja, são definidos para cobrir uma determinada abordagem de solução para o domínio do problema para o qual seus modelos são propostos ([AIZENBUD-RESHEF et al., 2006](#)).

Outro ponto que apresenta complicações é a representação dos elos em matrizes de rastreabilidade. Um elo pode possuir um determinado tipo que permite determinar qual a natureza do relacionamento que está sendo realizado entre dois artefatos. No caso de uma matriz de rastreabilidade os diversos elos, ali existentes, representam, em geral, a mesma natureza de relacionamento, ou pertencem a um mesmo grupo

de elos. Caso se deseje criar a representação de outro comportamento é necessário criar uma nova matriz com os mesmos artefatos mediante novos relacionamentos. Ao permitir a inserção de atributos nos elos dá-se a capacidade de aperfeiçoar a semântica desses elementos, permitindo-se que mais de um papel seja atributo a um mesmo relacionamento estabelecido pelo elo, além de permitir que sejam adicionados atributos que venham servir como recurso para mensuração.

Como exemplo: supondo-se a existência do relacionamento entre um requisito ( $r_1$ ) e dois casos de usos ( $a_1$  e  $a_2$ ), apresentados como  $r_1 \rightarrow a_1$  e  $r_1 \rightarrow a_2$ , em que os  $\rightarrow$  são:  $e_1$  e  $e_2$  e considerando-se apenas o estabelecimento de  $e_1$  e  $e_2$ , sendo do tipo satisfação, esses artefatos podem ser representados em uma matriz como na [Tabela 2.3](#).

Tabela 2.3 - Matriz representando elos do tipo satisfação entre o requisito  $r_1$  e os casos de uso  $a_1$  e  $a_2$ .

Origem \ Destino	$a_1$	$a_2$	...	$a_n$
$r_1$	x	x		
⋮				
$r_n$				

Supondo que além dos elos do tipo satisfação exista também outro comportamento entre  $r_1 \rightarrow a_2$ , como exemplo, o caso de uso  $a_2$  foi criado não apenas para modelar um determinado requisito mas para facilitar o uso de um componente já existente, que será utilizado no projeto ou, ainda, que este caso de uso foi criado apenas para facilitar um acordo/negociação realizado com o cliente pois este compreenderia melhor a representação do problema. Dessa forma um comportamento de decisão de projeto e outro de acordo/negociação não pode ser demonstrado na matriz de elos do tipo satisfação ([Tabela 2.3](#)), sendo assim necessário criar uma nova matriz que contenha os elos do tipo *Rationale*, como demonstrado na [Tabela 2.4](#).

Tabela 2.4 - Matriz representando elos do tipo *rationale*.

Origem \ Destino	$a_2$	...	...	$a_n$
$r_1$	x			
⋮				
$r_n$				

Ao permitir a inserção de atributos de informação nos elos, apenas uma matriz é

usada na qual a informação de *Rationale* pode ser inserida diretamente nos elos (Tabela 2.5).

Tabela 2.5 - Matriz representando elos do tipo satisfação entre o requisito  $r_1$  e os casos de uso  $a_1$  e  $a_2$  e a razão como atributo adional.

Origem \ Destino	$a_1$	$a_2$	...	$a_n$
$r_1$	x	x		
⋮				
$r_n$				

Origem  
Destino  
Autor  
Data  
Versão  
Rationale

Origem  
Destino  
Autor  
Data  
Versão  
Rationale

Explorando a representação de elos em matrizes surge a oportunidade de melhor explorar a descrição dos elos. Os elos, na maioria dos casos, representam apenas a ligação entre dois elementos do processo de desenvolvimento, ou seja, os modelos apresentados não permitem a inclusão de atributos ou semânticas mais detalhadas, assim, informações necessárias para estabelecer análises para a rastreabilidade (como avaliação de impacto, derivação ou cobertura (DICK, 2002; DICK, 2005; HULL et al., 2002)) não são apresentadas, representadas ou mesmo suportadas por esses modelos.

Uma alternativa para o problema da predefinição dos tipos de elos é a definição de um modelo que se baseia na generalização de todos os tipos de artefatos que possam fazer parte do processo de rastreabilidade, focando a representação das informações. Essa característica permite tanto a criação de novos tipos de elos quanto de outros artefatos, conforme a necessidade dos envolvidos na rastreabilidade.

## 2.5 Conclusão do Capítulo

Neste capítulo foi apresentada uma visão sobre os principais aspectos que envolvem a rastreabilidade de requisitos, úteis para o desenvolvimento do conhecimento e necessários para ampliar a discussão do tema. Foram apresentados os conceitos, as tecnologias e algumas das ferramentas existentes, além de serem abordados os esforços de pesquisas realizados na definição de modelos para rastreabilidade. Foi discutida a questão da pré-especificação de elos e a melhoria de semântica desses artefatos.

O próximo capítulo apresenta a abordagem proposta para tratar os problemas apresentados.



### 3 MODELO PARA A GENERALIZAÇÃO DE ELOS DE RASTREABILIDADE

Neste capítulo é apresentado o modelo proposto para a generalização de elos de rastreabilidade e seus elementos. Também é realizada uma discussão sobre sua aplicação. Após a apresentação do modelo são apresentados alguns exemplos de criação de artefatos e como os elos e seus atributos podem ser criados.

O modelo apresentado neste capítulo contribui para a solução do problema da pré-especificação de grupos de elos focando a generalização das categorias dos elos e outros artefatos. Além da generalização dos elos, o modelo permite a melhoria da semântica desses elementos através da inclusão de atributos, sendo que tais propriedades não são apresentadas nos modelos existentes para rastreabilidade.

#### 3.1 Modelo proposto

O modelo desenvolvido neste capítulo propõe uma solução que inova a representação de elos de rastreabilidade permitindo a generalização desses elementos, sendo que tal capacidade não é encontrada nos modelos avaliados na literatura sobre modelos para rastreabilidade que foram explorada no capítulo anterior.

Os modelos apresentados no Capítulo 2 são limitados no que diz respeito aos tipos de elos, ou seja, são fixados, e conseqüentemente são definidos para lidar apenas com soluções específicas para um domínio específico. Isso pode ser visualizado nos trabalhos de [Toranzo et al. \(2002\)](#) e [Ramesh e Jarke \(2001\)](#) em que os elos são estabelecidos, respectivamente, para o domínio das informações gerenciais da rastreabilidade e dos envolvidos no processo de rastreabilidade, enquanto o trabalho de [Aizenbud-Reshef et al. \(2006\)](#) tem foco nas soluções que fazem uso de MDA.

Com base nas restrições dos modelos, já apresentados, é proposto um modelo baseado em quatro elementos que podem representar vários tipos de artefatos. Diferente das outras abordagens o modelo proposto permite que sejam definidos os padrões dos requisitos, dos artefatos e dos elos a serem utilizados ao longo do projeto. A [Figura 3.1](#) apresenta os quatro elementos que compõem o modelo.

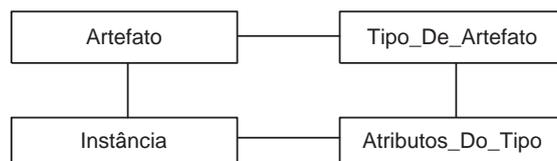


Figura 3.1 - Elementos do modelo proposto para a rastreabilidade de requisitos.

Vários são os tipos de artefatos, produzidos ao longo do processo de desenvolvimento, que devem ser incluídos no processo de rastreabilidade. Tal afirmação pode ser mais bem compreendida quando a pré e a pós-rastreabilidade, proposta por [Gotel \(1995\)](#), é analisada. Essa proposta afirma a existência de características específicas no tratamento da rastreabilidade antes da agregação dos requisitos ao documento de especificação de requisitos de software e após essa fase, ou seja, diferentes tipos de elos, artefatos e atividades existem no contexto desse processo.

Outro ponto a ser considerado é a granularidade em que se deseja executar a rastreabilidade; quanto mais baixo o nível maior o controle na análise de impacto, entretanto aumenta-se a complexidade na evolução e alteração dos elos em níveis muito refinados de rastreabilidade, além da diversidade de artefatos a serem utilizados.

Com base nessas características, o modelo, definido através de quatro elementos, tem como meta possibilitar a definição de diferentes tipos de artefatos permitindo a melhoria da semântica através da adição de atributos a esses artefatos. Os elementos do modelo são discutidos a seguir:

- a) Tipo de Artefato – Um *Tipo de Artefato* consiste da descrição de um grupo de artefatos, criados ao longo do processo de desenvolvimento, que possuem, coletivamente, as mesmas características, ou seja, pertencem a uma mesma classe ou ordem, podendo, assim, ser agrupados em um *Tipo de Artefato*. Como exemplo de um *Tipo de Artefato* pode-se citar o Requisito Funcional. Artefatos desse tipo são criados no processo de Engenharia de Requisitos, na atividade de Elicitação de Requisitos. Outros exemplos de Tipos de Artefatos podem ser citados como: requisito não-funcional, caso de uso, cenários, diagramas, matrizes, código, casos de teste, módulos, classes, objetos, funções, documentos, etc.
- b) Atributos do Tipo – Cada artefato, criado ao longo do processo de desen-

volvimento, possui informações que permitem detalhar seu comportamento ou suas propriedades. O elemento *Atributos do Tipo* permite que as propriedades, inerentes a um conjunto de artefatos, sejam apresentadas, ou seja, esse elemento permite adicionar aos *Tipos de Artefatos* atributos que podem ser usados para melhorar a documentação e a identificação entre diferentes *Tipos de Artefatos*. Como exemplo de *Atributos do Tipo* para o tipo Requisito Funcional pode-se especificar um identificador, um código, a descrição do requisito, os usuários envolvidos com o requisito, a fonte, quem coletou o requisito, a data da elicitação, versão, verbo, substantivo, nível de risco, nível de prioridade, etc.

- c) Artefato – São os próprios artefatos gerados em um projeto. Esse elemento permite referir os artefatos reais criados ao longo do processo de desenvolvimento. Um artefato está relacionado a um *Tipo de Artefato*, só possuindo os atributos pertencentes ao tipo ao qual ele está associado. Desta forma, todos os artefatos de um tipo, documentados pelo modelo, para um dado projeto, possuem os mesmos atributos.
- d) Instâncias – O elemento *Instâncias* permite que os diferentes valores dos *Atributos do Tipo*, ao qual um determinado artefato faz referência, sejam representados. Uma instância é um valor real, alocado a um atributo de um tipo, ou seja, os valores reais para os atributos de um artefato, criado durante o desenvolvimento, são representados por este elemento.

O modelo se baseia na generalização de todos os tipos de artefatos que possam fazer parte do processo de rastreabilidade, focando a representação das informações. Essa característica permite tanto a criação de novos tipos de elos quanto de outros artefatos, conforme a necessidade dos envolvidos na rastreabilidade. Dois casos particulares justificam essa generalização. O primeiro trata sobre a agregação de novos campos para um dado tipo de elo ou artefato. O segundo trata sobre a inserção de novos tipos de artefatos.

Para o primeiro caso os elos constituem, na maioria das aplicações existentes, apenas a existência da ligação entre dois artefatos: informações importantes não são agregadas a essa ligação. Ao constituir um elo de rastreabilidade deve-se permitir que sejam inseridas informações do processo ou adjacentes a esse, como informações sobre qualidade e *rationalidade*.

O segundo caso consiste da necessidade de inclusão de novos tipos de elos ou outros artefatos ao longo do projeto ou para um novo projeto. Permitir a definição dos tipos de elos somente no início do projeto não condiz com a necessidade de evolução e adaptação do projeto diante das alterações que possam vir a ser realizadas.

### 3.2 Formalização do modelo

A relação entre os elementos do modelo pode ser visualizada através de um diagrama entidade relacionamento, [Figura 3.2](#).

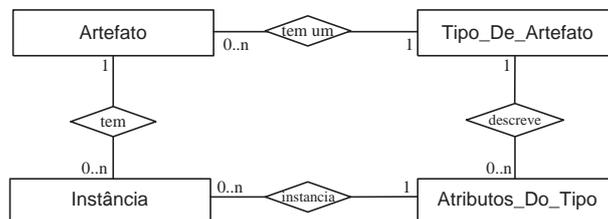


Figura 3.2 - Diagrama entidade relacionamento do modelo proposto.

No modelo relacional um esquema de relação é dado por  $P(A, K)$ , em que  $P$  é o nome dado à relação e  $A = \{A_1, A_2, \dots, A_n\}$  é um finito grupo de atributos. Cada atributo  $A_i$  representa um papel desempenhado por algum domínio  $D$ ,  $\text{dom}(A_i)$ . O número  $n$  de atributos define o grau da relação. Enquanto que  $K$  consiste de um grupo finito de atributos chaves  $K = \{K_1, K_2, \dots, K_n\}$  em que  $K \subseteq A$ . Dizer que  $K = \{K_1, K_2, \dots, K_n\}$  é a chave de um esquema de relação é dizer que qualquer relação válida  $p$  em  $P$  tem a propriedade que para qualquer tupla distinta  $t_1$  e  $t_2$  em  $p$ ,  $t_1(K) \neq t_2(K)$ , e que nenhum outro subgrupo de  $K$  tem essa propriedade.

Assim como no modelo relacional o modelo proposto é definido como um grupo  $S$  formado por quatro esquemas de relação  $S = (\text{Tipo\_De\_Artefato } (TF), \text{ Atributos\_Do\_Tipo } (AT), \text{ Instancias } (I), \text{ Artefatos } (A))$ . A [Tabela 3.1](#) mostra as relações e seus atributos.

Tabela 3.1 - Descrição dos elementos do modelo.

Nome da Relação	Nomes dos Atributos	Domínio
Tipo_De_Artefato	TipoDoArtefatoID	chave primária
	Nome	nomes de artefatos
	Descrição	descrição dos artefatos
Atributos_Do_Tipo	AtributosDoTipoID	chave primária
	TipoDeArtefatoFK	chave estrangeira para $TF$
	Nome	nomes dos atributos da relação $TF$
	Descrição	descrição dos atributos
Instância	InstânciaID	chave primária
	ArtefatoFK	chave estrangeira da relação $A$
	AtributosDoTipoFK	chave estrangeira da $AT$
	Valor	valores para os atributos do artefato
Artefato	ArtefatoID	chave primária
	TipoDeArtefatoFK	chave estrangeira de $TF$
	Data	data da criação do artefato
	Versão	versão do artefato
	Descrição	descrição do artefato

Cada elemento de  $S$  pode ser instanciado. Por exemplo, *Artefato* pode ser instanciado como  $a_i(A)$  tendo grau 5 (domínios: ArtefatoID, TipoDeArtefatoFK, Data, Versão, e Descrição), exemplo:

$$a_i(A) \subseteq (dom(ArtefatoID) \times dom(TipoDeArtefatoFK) \times dom(Data) \times dom(Versão) \times dom(Descrição))$$

O mesmo conceito se aplica aos outros elementos de  $S$ .

### 3.3 Definição de artefatos

A definição de um artefato  $a_i$ , no modelo proposto, passa por três etapas básicas detalhadas a seguir:

- a) Definir o tipo de artefato – Especificar o nome e a descrição de uma categoria na qual uma instância de artefato fará parte, como: código fonte, documento, requisito, elo, caso de uso, etc.
- b) Determinar os atributos do tipo do artefato – Os atributos são utilizados para detalhar o tipo de artefato. Por exemplo um tipo de artefato “caso de uso” pode possuir os seguintes atributos: nome, ator, precondições, fluxo principal, etc.
- c) Criar um artefato – A criação de um artefato depende do tipo ao qual o novo artefato será instanciado e do domínio de valores que serão atribuídos

a ele. Suas características são definidas pela relação *Atributos\_Do\_Tipo*; assim, através das chaves dessa relação é possível associar os atributos ao artefato e suas instâncias.

Os valores dos atributos de  $a_i$  são designados na relação *Instância* a qual que, através de suas restrições, estabelece que esses valores pertencem exclusivamente ao artefato  $a_i$ .

O processo de criação de um artefato pode ser visualizado através de um diagrama de atividades usando-se a notação do *Software Process Engineering Metamodel Specification*–SPEM (GENVIGIR et al., 2003; OMG, 2005) proposto pelo Object Management Group – OMG, Figura 3.3.

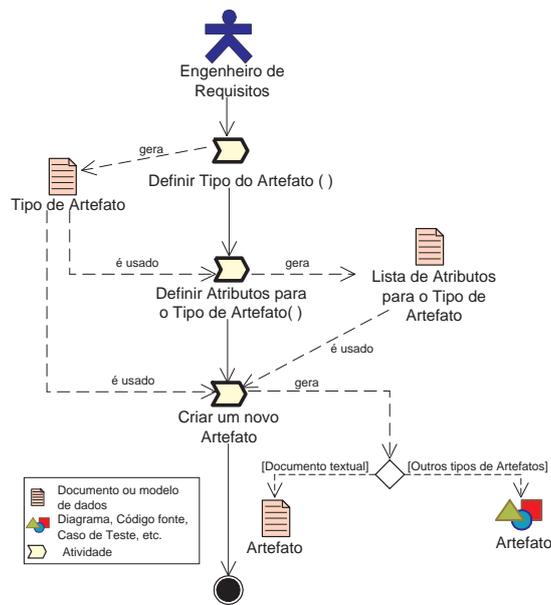


Figura 3.3 - Diagrama de atividades do processo de criação de um artefato com uso do modelo.

O artefato  $a_i$  é definido pelos atributos da relação *Artefato*, pelo tipo definido na relação *Tipo\_Do\_Artefato*, e pelos valores armazenados na relação *Instância*.

Na Figura 3.4 um artefato  $a_i$  é definido como sendo do tipo Requisito Funcional com dois atributos (Descrição e Fator de Risco). Cada um desses atributos tem suas instâncias (“o sistema deve...” e “Risco1”) que estão associados ao artefato  $a_i$ .

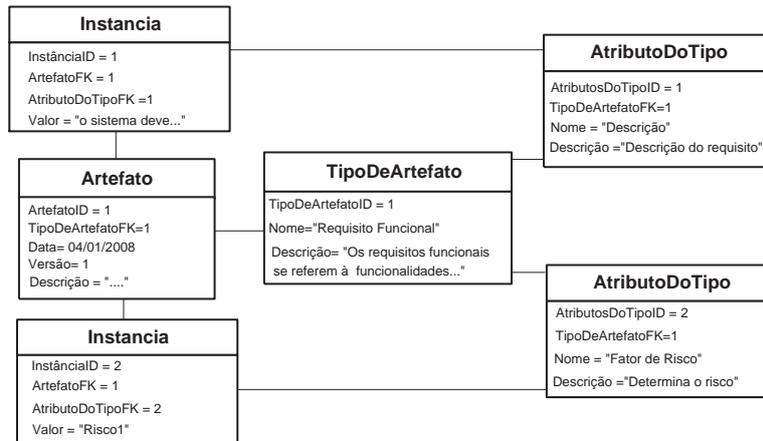


Figura 3.4 - Valores do artefato  $a_1$ .

A visualização do artefato  $a_1$  pode ser feita através de três operações conhecidas na álgebra relacional: projeção( $\pi$ ), seleção( $\sigma$ ) e junção natural ( $\bowtie$ ):

$$\pi_{a.ArtefatoID,tf.Nome,at.Nome,i.Valor}(\sigma_{a.ArtefatoID=1}(a \bowtie i \bowtie tf \bowtie at)). \quad (3.1)$$

A Tabela 3.2 mostra o resultado da Expressão 3.1.

Tabela 3.2 - Resultado da expressão 3.1.

<i>a.ArtefatoID</i>	<i>tf.Nome</i>	<i>at.Nome</i>	<i>i.Valor</i>
1	Requisito Funcional	Descrição	O sistema deve...
1	Requisito Funcional	Fator de Risco	Risco1

Todos os atributos necessários para documentar um determinado artefato podem ser criados através da relação *at*. Para estabelecer um elo é necessário considerar a criação de pelo menos mais um artefato, dito  $a_2$ . Aqui  $a_2$  é do tipo *tf*="Caso de Uso", que possui dois atributos *at*="Identificador" e *at*="Nome", que têm valores instanciados *i.valor* ="2" e *i.valor* ="Controlar dados do cliente". O artefato  $a_2$  é apresentado na Figura 3.5.

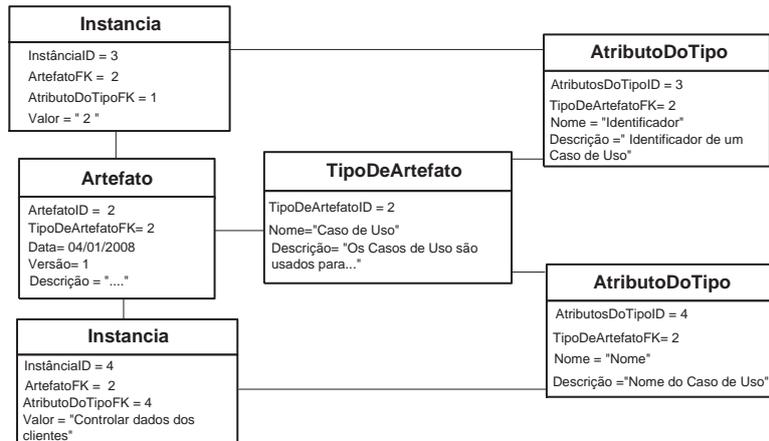


Figura 3.5 - Valores do artefato  $a_2$ .

Após a definição de  $a_1$  e  $a_2$  um elo pode ser estabelecido entre estes dois artefatos, sendo o procedimento para criar o elo similar ao utilizado para outros artefatos. O elo definido, para este caso, é um artefato  $a_3$ , do tipo  $tf$ ="Elo", que consiste de três atributos  $at.nome$ ="Identificador",  $at_1.nome$ ="Código da Origem",  $at_2.nome$ ="Código do Destino" que tem como valores de instâncias  $i_1.valor=1$ ,  $i_2.valor=1$ , e  $i_3.valor=2$ . A Figura 3.6, ilustra  $a_1 \rightarrow a_2$  que é executado por  $a_3$ .

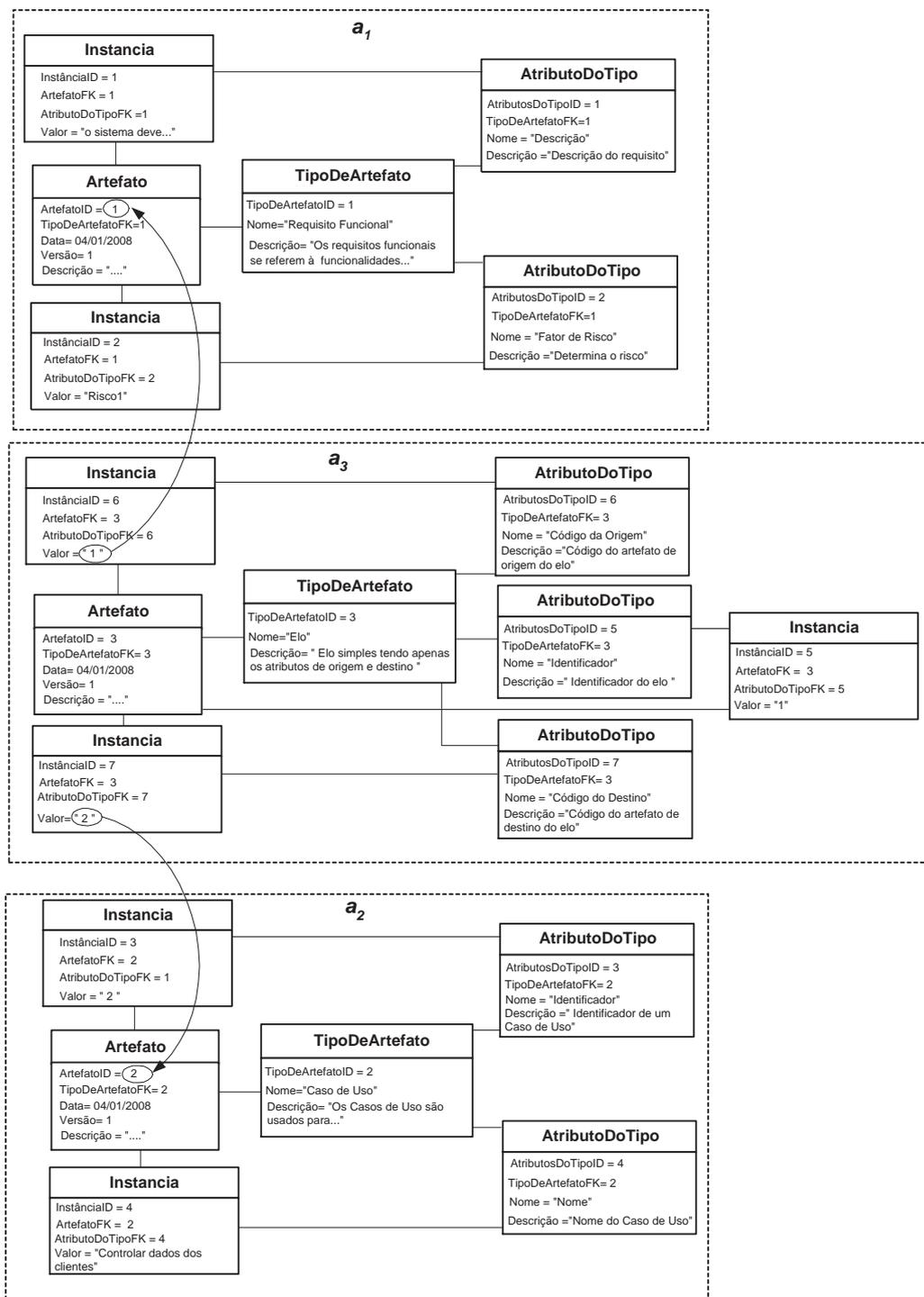


Figura 3.6 - Valores do artefato  $a_3$ .

**Nota:** O artefato  $a_3$  é do tipo "elo" usado para relacionar os artefatos  $a_1$  e  $a_2$ .

O elo apresentado por  $a_3$  pode ser visualizado mediante o uso das operações de projeção ( $\pi$ ), seleção ( $\sigma$ ) e junção natural ( $\bowtie$ ):

$$\pi_{a.ArtefatoID,at.AtributoID,tf.Nome,i.Valor}(\sigma_{a.ArtefatoID=3}(a \bowtie i \bowtie tf \bowtie at)). \quad (3.2)$$

O resultado da Expressão 3.2 é apresentado na Tabela 3.3.

Tabela 3.3 - Resultado da expressão 3.2.

<i>a.ArtefatoID</i>	<i>at.AtributoID</i>	<i>at.Nome</i>	<i>i.Valor</i>
3	5	Identificador	1
3	6	Código da Origem	1
3	7	Código do Destino	2

Através do resultado da Expressão 3.2, é possível verificar que *i.Valor* apresenta os valores que correspondem ao código do artefato de origem e o código do artefato de destino referentes a um elo. Além dos atributos de identificação, da origem, e do destino, outros atributos podem ser adicionados ao elo, o que permite aumentar a capacidade desse elemento dentro do processo de rastreabilidade.

### 3.4 Conclusão do Capítulo

Neste capítulo foi apresentado o modelo para generalização de elos de rastreabilidade, seus elementos e sua formalização através da algebra relacional. Foram apresentados alguns exemplos para a criação de artefatos e, foi indicado como o modelo pode estabelecer elos entre artefatos.

O próximo capítulo apresenta a arquitetura e o protótipo elaborado a fim de validar, através de implementação, o modelo apresentado neste capítulo.

## 4 UM MODELO DE ARQUITETURA DE SOFTWARE PARA SU- PORTE AO MODELO DE RASTREABILIDADE

Este capítulo apresenta uma arquitetura de software que objetiva dar suporte para modelo proposto. Os requisitos para a aplicação e sua modelagem, através de casos de uso, são apresentados, bem como alguns elementos de projeto como as interfaces.

Compreende-se, neste trabalho, o termo arquitetura de software como uma estrutura que permite o entendimento de componentes de um sistema e seus inter-relacionamentos, especialmente aqueles atributos que são consistentes ao longo do tempo (FILHO, 2002); (SHAW, 2001), e compreendem-se os termos aplicação e ferramenta como a implementação desta arquitetura.

A Elicitação dos requisitos tem como base a descrição do modelo de rastreabilidade modelando os requisitos através de casos de uso, que devem ser atendidos para que o modelo seja implementado. É apresentado um diagrama de contexto, em nível de visão de caso de uso, visando detalhar os recursos que a arquitetura deve prover e, após essa atividade, é proposta a arquitetura.

### 4.1 Elicitação dos requisitos para a arquitetura

A atividade de Elicitação tem papel importante para o sucesso do projeto. Caso os requisitos não sejam levantados adequadamente as fases posteriores do desenvolvimento podem ser prejudicadas ou, até mesmo, acarretar o fracasso do projeto. Sistemas que não atendem as necessidades de seus usuários poderão ser descontinuados ou fadados ao fracasso.

Para este trabalho a atividade de Elicitação foi simplificada. Isso ocorreu devido aos objetivos propostos para a elaboração da arquitetura, que são: definir os requisitos que implementem as funcionalidades para dar suporte à representação de diferentes tipos de elos, de outros artefatos e permitir a inserção de atributos a esses artefatos para o modelo apresentado. Assim, a atividade de elicitação objetivou apresentar os requisitos para uma ferramenta, em forma de protótipo, para dar suporte ao modelo proposto a fim de validá-lo funcionalmente, e não elicitar os requisitos de um ambiente corporativo em que estão inseridos diferentes envolvidos (*stakeholders*), pontos de vista, necessidades, normas, entre outros. Devido aos objetivos e às características da aplicação, diversas atividades da elicitação, da negociação e validação, da análise e da documentação de requisitos foram simplificadas ou não realizadas.

A atividade de Elicitação utilizada segue o modelo proposto por (GENVIGIR, 2004; GENVIGIR; SANT'ANNA, 2007). A seguir são descritas as atividades que foram realizadas bem como os requisitos elicitados.

#### 4.1.1 Definição do domínio da aplicação

Segundo Jackson (1995), o primeiro elemento de estudo da elicitação consiste em estruturar e analisar o domínio da aplicação. O domínio da aplicação é onde os requisitos particulares dos envolvidos são encontrados. Se o domínio da aplicação não for identificado corretamente, não há aptidão para focalizar os requisitos. Zave (1997) explica que todas as descrições envolvidas na engenharia de requisitos poderão ser descrições de ambiente.

A identificação do ambiente ou domínio da aplicação onde está inserido o problema que será tratado é o passo inicial para o planejamento das atividades de elicitação dos requisitos inerentes ao contexto do negócio.

- **Domínio da Aplicação:** Modelos para a rastreabilidade de requisitos de software.

#### 4.1.2 Compreensão do problema a ser resolvido

Um problema ao ser tratado agrega características inerentes ao fator humano do querer, do saber, do poder e, principalmente, da comunicação e do entendimento do requisito. Gause e Weinberg (1989, 1990) apresentam várias abordagens sobre esse tema.

O problema, no contexto da elicitação de requisitos, é a razão principal para o entendimento, a especialização e o domínio do conhecimento. Identificar qual é o problema, qual é a definição do problema, quem tem o problema e qual a essência do problema, sob o ponto de vista de quem o tem, caracteriza a complexidade do processo. Faz-se, então, necessário distinguir claramente entre a definição do problema (conhecimento dos requisitos) e a solução do problema. Neste trabalho foram realizadas duas tarefas para a realização da atividade Compreender o problema a ser resolvido: 1) Definir os objetivos da aplicação, 2) Definir as complexidades da aplicação.

- 1 **Objetivos da aplicação:** Dar suporte ao modelo proposto através de uma

ferramenta em formato de protótipo.

## 2 Complexidades da aplicação:

- O modelo possui como ponto principal a generalização dos artefatos envolvidos no processo de rastreabilidade. Essa característica impõe problemas de acoplamento e coesão, o que pode acarretar dificuldades para o desenvolvimento;
- A aplicação deve ser desenvolvida usando-se tecnologia WEB;
- A aplicação deve ser desenvolvida usando-se tecnologias de código aberto.

### 4.1.3 Definição dos requisitos funcionais da aplicação

Os requisitos funcionais capturam a natureza da interação entre o produto e o seu ambiente, devendo definir objetos, funções e estados; limitar ou controlar as ações a eles associadas e definir os relacionamentos entre eles.

Os requisitos funcionais levantados para a aplicação foram os seguintes:

RF-1 O sistema deverá controlar os tipos de artefatos.

RF-2 O sistema deverá controlar os atributos dos tipos de artefatos.

RF-3 O sistema deverá controlar os tipos de artefatos.

RF-4 O sistema deverá controlar as instâncias dos atributos dos tipos de artefatos.

RF-5 O sistema deverá gerenciar as matrizes de rastreabilidade.

RF-6 O sistema deve controlar o acesso dos usuários.

RF-7 O sistema deve implementar dois padrões de análise: impacto e esforço.

## 4.2 Diagrama de contexto de caso de uso

O diagrama de contexto de caso de uso (BOOCH *et al.*, 1999) demonstra uma visão ampla das funcionalidades do sistema. Na [Figura 4.1](#) é demonstrada a interação entre os atores e os principais casos de uso que compõem as funções do sistema.

Cada um dos casos de uso apresentados nesse diagrama será descrito assim como suas funcionalidades.

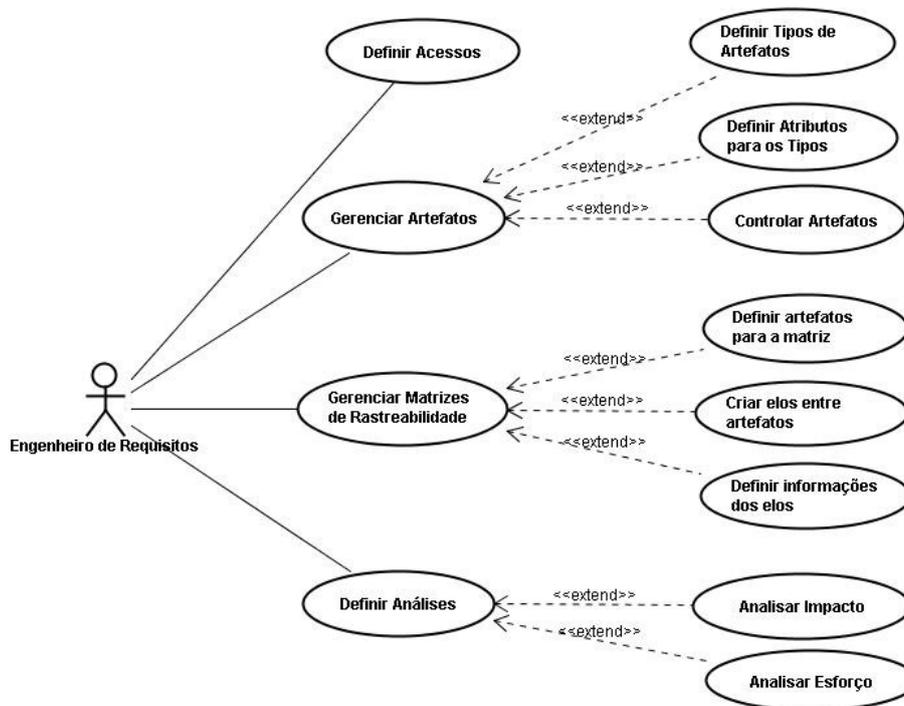


Figura 4.1 - Diagrama de contexto de caso de uso para a aplicação.

#### 4.2.1 Descrição dos casos de uso

Como apresentado anteriormente, o diagrama de contexto demonstra as principais interações entre atores e ações do sistema. Para demonstrar um maior nível de detalhes sobre a visão da aplicação são apresentados, nesta seção, os casos de uso que compõem o diagrama de contexto.

- Caso de uso definir acessos – O controle de acesso consiste em determinar os perfis e as permissões dos usuários em relação ao sistema. A ferramenta deve manter a integridade e o sigilo das informações do projeto, pois o sistema será desenvolvido para operar ambiente WEB. Uma interface inicial deverá ser apresentada solicitando-se os dados de acesso dos usuários além de um gerenciador para os perfis.
- Caso de uso gerenciar artefatos – Os artefatos são os principais elementos

do modelo proposto para a rastreabilidade.

Este caso de uso deve controlar as atividades relacionadas aos artefatos do projeto. Por questões de organização do projeto, as atividades deste caso de uso foram decompostas em outros três casos de uso: a) Definir tipos de artefatos; b) Definir atributos para os tipos; e c) Controlar artefatos.

a) Caso de uso definir tipos de artefatos – A primeira atividade do processo de rastreabilidade consiste em definir quais serão as categorias de artefatos, produzidos durante o desenvolvimento, que farão parte do processo de rastreabilidade. Algumas organizações realizam a rastreabilidade apenas entre requisitos, outras incluem elementos de outros processos como projeto, implementação, testes, entre outros.

A ferramenta deve permitir o registro dos tipos de artefatos que farão parte do processo de rastreabilidade permitindo sua inclusão, exclusão e alteração.

b) Caso de uso definir atributos para os tipos – Quando um tipo de artefato é registrado no sistema novos atributos poderão ser associados a esse tipo. Os atributos permitem incluir tanto informações de produto quanto do processo associado ao tipo de artefato.

Ao incluir um atributo num tipo, todas as instâncias desse tipo deverão possuir esse atributo. A ferramenta deverá listar os tipos já cadastrados e permitir a inclusão, exclusão e alteração dos atributos associados a um determinado tipo.

c) Caso de uso controlar artefatos – Esse caso de uso deverá permitir o controle sobre os artefatos que farão parte do processo de rastreabilidade. A aplicação deverá prover essa identificação através da descrição dos atributos associados ao tipo do artefato e, se necessário, a anexação de arquivos de outros aplicativos que correspondam às fontes do artefato.

- Caso de uso gerenciar matrizes de rastreabilidade – As matrizes de rastreabilidade serão adotadas como recursos para representar os relacionamentos entre artefatos no sistema. As atividades desse caso de uso são realizadas por três outros casos de uso: a) Definir artefatos da matriz; b) Criar elos entre artefatos; e c) Definir informações dos elos.

a) Caso de uso definir artefatos da matriz – Um artefato do tipo matriz deve permitir a associação entre os artefatos de origem e os artefatos de

destino. Definir quais são esses artefatos para uma determinada matriz é o papel desse caso de uso. A aplicação deverá disponibilizar uma listagem dos artefatos, já cadastrados e classificados pelos tipos, e permitir que estes sejam alocados como artefatos de origem ou como artefatos de destino.

b) Caso de uso criar elos entre artefatos – Neste caso de uso, os elos deverão ser criados entre os artefatos de origem e destino. A aplicação deverá disponibilizar, na apresentação da matriz, um recurso gráfico em linguagem de marcação que permita a inserção do elo. O ator, ao clicar sobre esse recurso, define a existência de um elo entre dois artefatos e, ao clicar sobre um elo já definido, removendo a marcação, o elo deve ser excluído. A aplicação também deverá tratar o atributo artefato de origem do elo e o atributo artefato destino do elo, que são essenciais para a implementação do modelo.

A definição dos artefatos que serão associados será realizada pelo ator Engenheiro de Requisitos de fará uso de seu conhecimento sobre o projeto para definir os elos.

c) Caso de uso definir informações dos elos – Assim como outros artefatos os elos podem possuir atributos. Ao definir um elo, a aplicação deverá disponibilizar a apresentação dos atributos definidos para o elo.

Neste caso de uso o ator poderá controlar as informações dos elos existentes nas matrizes do projeto, inserindo, removendo ou alterando essas informações.

As informações dos elos deverão ser apresentadas em forma de listagens agrupadas em janelas individuais associadas a cada elo. A alteração de uma informação em um elo não deverá implicar na reconstrução de toda a matriz em sua tela de apresentação, ou seja, apenas as informações deste elo deverão ser reapresentadas.

- Caso de uso definir análises – As análises permitem demonstrar algumas capacidades do modelo em agregar informações ao processo de rastreabilidade, o que pode auxiliar tanto a rastreabilidade quanto outros processos como o de gerência de projeto.

Este caso de uso tem por objetivo a implementação de duas análises: a de esforço e a de impacto. A análise de impacto é tradicionalmente realizada

pelas ferramentas que implementam a rastreabilidade, porém a análise de esforço não.

Devido à realização de mais de um tipo de análise as atividades deste caso de uso são implementadas por outros dois casos de uso: a) analisar impacto e b) analisar esforço, apresentados a seguir.

a) Caso de uso analisar impacto – Este caso de uso deve realizar uma análise de impacto na qual o ator escolherá qual artefato deseja analisar e a aplicação deverá realizar a busca dos artefatos associados através dos elos criados nas matrizes. Todos os artefatos que estejam associados ao artefato escolhido deverão ser apresentados o que permitirá visualizar quais artefatos seriam impactados caso alguma alteração for realizada no artefato selecionado.

A análise de impacto está diretamente associada ao nível de granularidade da rastreabilidade realizada.

b) Caso de uso analisar esforço – A análise de esforço visa aumentar a capacidade das análises que podem ser realizadas com as informações adicionadas aos artefatos. A aplicação deverá permitir que o usuário escolha um determinado artefato e será apresentado o esforço associado ao artefato escolhido, aos elos e aos demais artefatos ligados ao artefato escolhido.

### **4.3 Arquitetura para implementação do modelo para rastreabilidade de requisitos**

Com a análise dos casos de uso, elaborados a partir do modelo proposto para rastreabilidade de requisitos, foram definidas quatro funcionalidades que devem ser atendidas pela arquitetura a fim de atender o modelo:

- 1 Gerenciamento de artefatos.
- 2 Gerenciamento de matrizes.
- 3 Controle de acessos.
- 4 Ferramentas suplementares: Análise de impacto e Análise de esforço.

As funcionalidades são implementadas na arquitetura, [Figura 4.2](#), sendo esta definida em três camadas: camada de interface com o usuário que define a forma de

interação entre usuário e sistema; camada de serviços que determina os serviços que a arquitetura deve implementar para prover suporte ao modelo; e camada de armazenamento que possui os repositórios utilizados pelas demais camadas.

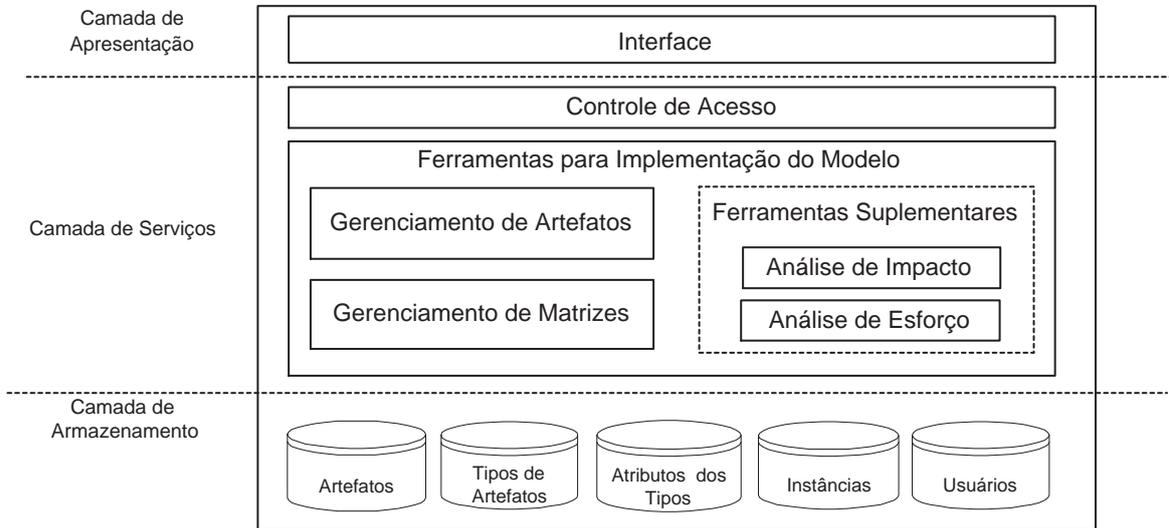


Figura 4.2 - Arquitetura proposta para dar suporte ao modelo para rastreabilidade.

A arquitetura fornece uma visualização orientada ao modelo, o que permite fornecer o suporte para o processo de rastreabilidade e a representação de artefatos. A arquitetura é proposta de forma a implementar o modelo a fim de demonstrar sua viabilidade. Os requisitos, assim como os demais artefatos, são definidos sem o uso de um processo específico, mas a ferramenta, ao implementar o modelo, permite a representação desses elementos a partir de diferentes perspectivas como orientada ao projeto ou a processos.

#### 4.3.1 Definição dos elementos da arquitetura

- Interfaces – As interfaces devem ser construídas de modo a propiciar a interação entre os usuários e entre as camadas mais interiores da aplicação e devem permitir o uso de meios de hipermídia, o que facilita a visualização dos artefatos.
- Controle de acesso – Por motivos de segurança, esse serviço deverá proporcionar a adequação da aplicação, a cada ator. Isso permitirá que cada usuário tenha seu próprio ambiente, que será montado dinamicamente a partir de permissões.

- Gerenciamento de artefatos – As relações existentes no modelo são integradas através dessa funcionalidade que permite o cadastro dos tipos de artefatos, dos atributos associados aos artefatos e do gerenciamento dos próprios artefatos do projeto que são instâncias das relações: artefatos, atributos do tipo e instâncias.

Cada artefato, por sua vez, é uma instância de um tipo definido no modelo. Os artefatos devem possuir versões, que devem ser controladas por outra ferramenta, ou a de gerenciamento de versão de requisitos ou a de controle de versão geral.

Essa funcionalidade é composta por três subsistemas: controle de tipos, controle de atributos do tipo e controle de artefatos. Cada um desses subsistemas apresenta ao usuário uma lista dos elementos já cadastrados permitindo a inclusão, alteração ou exclusão.

- Gerenciamento de matrizes – As matrizes correspondem a um dos principais recursos para demonstrar a relação existente entre artefatos em um processo de rastreabilidade. A aplicação deve permitir a criação de matrizes através da inserção de artefatos como sendo do tipo origem ou destino. Além da inserção dos artefatos essa funcionalidade permite que sejam criados os elos entre os artefatos e o preenchimento das informações nos elos. A inserção de informações nos elos está relacionada aos atributos que podem ser adicionados aos artefatos do tipo elo. As informações são construídas de forma dinâmica, o que permite a apresentação de atributos inseridos em qualquer fase do processo. Os atributos origem e destino do elo são obrigatórios enquanto os outros atributos são definidos pelo engenheiro de requisitos.
- Análise de impacto – Essa funcionalidade visa demonstrar a viabilidade do modelo em dar suporte a algumas atividades que fazem uso da rastreabilidade. A análise de impacto possui dois focos principais: 1) viabilizar a identificação dos artefatos que podem estar envolvidos em uma determinada mudança; e 2) viabilizar a identificação das consequências da mudança.

- Análise de esforço – A análise de esforço é o segundo elemento escolhido para demonstrar a capacidade do modelo em incorporar informações adicionais aos elos. Ao definir um atributo esforço<sup>1</sup> a um tipo é possível atribuir o valor despendido na elaboração de uma determinada instância desse tipo. Da mesma forma, o atributo esforço, que é definido para um artefato do tipo elo, será utilizado para incluir o esforço gasto para a elaboração do elo. A inclusão desta informação aos elos permite que o usuário, ao escolher um determinado artefato, obtenha os valores dos artefatos associados ao elemento de origem e também dos elos envolvidos na rastreabilidade para trás e para frente. Isso é importante, pois, em um processo manual de criação dos elos, o esforço aplicado pode ser expressivo, o que nem sempre é contabilizado pelas gerências ao avaliar o custo ou esforço despendido.

#### 4.3.2 Comparação entre funcionalidades e casos de uso

A [Tabela 4.1](#) apresenta como as funcionalidades da arquitetura proposta respondem às necessidades levantadas nos casos de uso. As funcionalidades são numeradas e avaliadas com os casos de uso e, quando estes são atendidos por uma ou mais funcionalidades, ele recebe o valor “x”.

---

<sup>1</sup>O esforço é definido neste trabalho como  $esforço = (t \cdot qp)$  em que  $t$  é tempo gasto em horas/minutos utilizado para a elaboração do artefato e  $qp$  é a quantidade de pessoas empregadas, enquanto o custo é definido como  $custo = esforço \cdot ch$  em que  $ch$  é o valor da hora de trabalho do profissional envolvido na elaboração do artefato.

Tabela 4.1 - Comparação entre casos de uso e funcionalidades.

Casos de Uso \ Funcionalidades	Funcionalidades				
	1	2	3	4	5
Definir acessos	x				
Gerenciar artefatos		x			
Definir tipos de artefatos		x			
Controlar artefatos		x			
Definir tipos de artefatos		x			
Definir atributos para os tipos		x			
Controlar artefatos		x			
Gerenciar matrizes de rastreabilidade			x		
Definir artefatos da matriz			x		
Criar elos entre artefatos			x		
Definir informações dos elos			x		
Definir análises				x	x
Analisar esforço				x	
Analisar impacto					x

**Nota:** Funcionalidades: 1 – Controle de acesso; 2 – Gerenciador de artefatos, 3 – Gerenciador de matrizes, 4 – Analisador de esforço, 5 – Analisador de impacto.

#### 4.4 Desenvolvimento da aplicação

Com o objetivo de fazer uso de tecnologias de código aberto a aplicação foi desenvolvida utilizando os seguintes recursos:

- Linguagem de programação: Java plataforma EE – Servlets, *JavaServer Pages* – JSP e JavaBeans;
- Servidor de aplicação: Apache Tomcat versão 6.0
- Servidor de banco de dados: PostgreSQL versão 8.2;
- Linguagem de *script* JavaScript;
- Linguagem de marcação: *HyperText Markup Language* – HTML;
- Ferramenta para *debug* de *script* Firebug versão 1.2;
- Tecnologia *Asynchronous Javascript And XML* – Ajax;
- Estrutura de desenvolvimento NetBeans versão 6.1;
- Ferramenta de administração de banco de dados: pgAdmin III versão 1.6.2;
- Ferramenta de modelagem UML: JUDE.

#### 4.4.1 Interfaces dos módulos da aplicação

O acesso à aplicação é realizado através da interface de controle de acesso, [Figura 4.3](#).



Nome	<input type="text"/>
Senha	<input type="password"/>
<input type="button" value="OK"/>	

Done

Figura 4.3 - Controle de acesso.

Após acessar a aplicação, a primeira etapa a ser realizada pelo usuário é a criação dos Tipos de Artefatos que serão utilizados pelo processo de rastreabilidade. A [Figura 4.4](#) apresenta essa funcionalidade e são listados cinco tipos de artefatos que foram criados.

Na [Figura 4.4](#) é possível observar que, ao inserir um tipo de artefato, apenas três campos estão disponíveis: identificador do tipo de artefato, nome do tipo de artefato e descrição, esses três campos compõem o elemento Tipo de Artefatos do modelo.



Figura 4.4 - Controle de tipos de artefatos.

Após a inclusão dos Tipos de Artefatos é possível definir os atributos para esses tipos. Essa funcionalidade é realizada pelo Controle de Atributos dos Tipos de Artefatos. A Figura 4.5 mostra essa funcionalidade e pode ser visualizado o tipo Elo composto por cinco atributos, listados logo abaixo na interface.

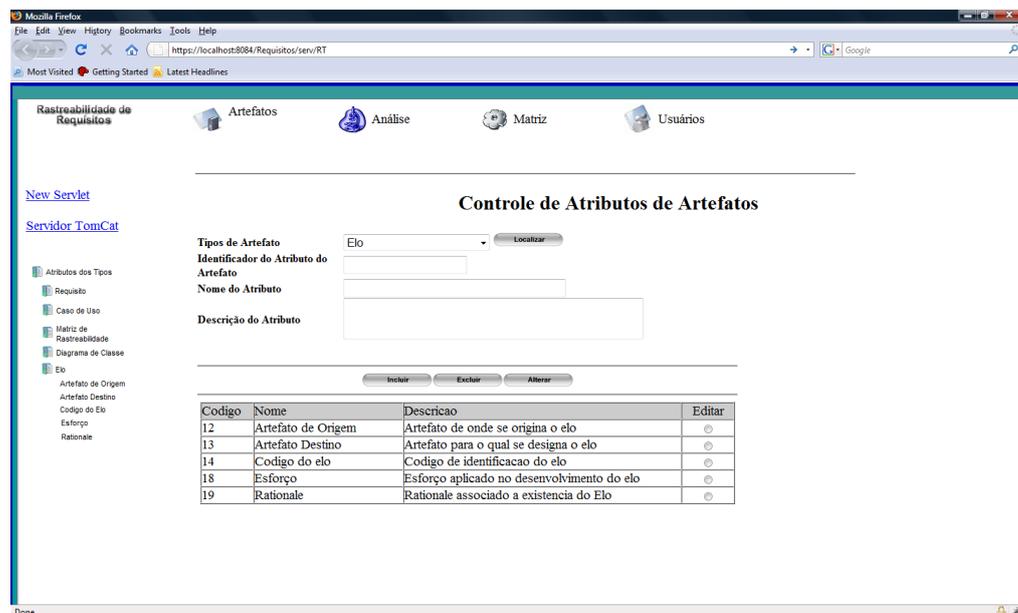


Figura 4.5 - Controle de atributos de artefatos.

Ao incluir um atributo a um tipo as seguintes informações devem ser inseridas: o indentificador do atributo, o nome e a descrição do atributo, conforme estabelecido pelo modelo.

Após o cadastro dos Tipos de Artefatos e de seus atributos, a aplicação está pronta para o registro dos artefatos elaborados ao longo do desenvolvimento. Esse cadastro é realizado através do Controle de Artefatos, [Figura 4.6](#). É possível observar na [Figura 4.6](#) um artefato do tipo Requisito que possui sete atributos.



Figura 4.6 - Controle de artefatos.

Os atributos do tipo Requisitos, apresentados na [Figura 4.6](#), foram criados anteriormente no Cadastro de Atributos dos Tipos. Entretanto, é possível adicionar, se desejável, novos atributos aos tipos ao longo do projeto. Caso isso seja feito todos os artefatos do tipo, em que foi inserido o novo atributo, passarão a possuir o novo atributo inserido. Todavia, não é possível excluir um atributo de um tipo se este possuir instâncias cadastradas pelo Controle de Artefatos. Isso se deve à integridade referencial do modelo, ou seja, não é possível remover um atributo de um tipo que possui instâncias associadas a um artefato, embora a aplicação possa implementar a exclusão de um artefato como um todo, se isso for desejável.

As matrizes devem ser inseridas como artefatos, e após esse procedimento é possível editar seus elementos.

O controle de matrizes é feito em três passos, o primeiro (Figura 4.7) apresenta ao usuário as matrizes cadastradas no módulo de Controle de Artefatos, após escolher a matriz, que será editada, é apresentada ao usuário a tela dos elos disponíveis para uso na matriz (Figura 4.8) e por fim a própria edição da matriz escolhida, Figura 4.9.

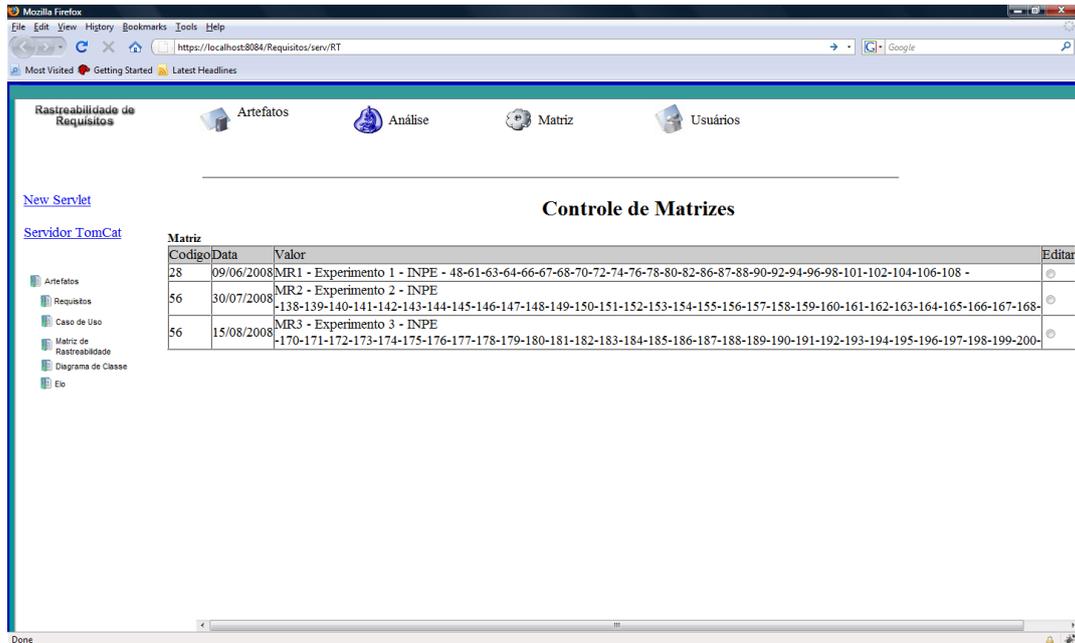


Figura 4.7 - Controle de matrizes – matrizes disponíveis para edição.

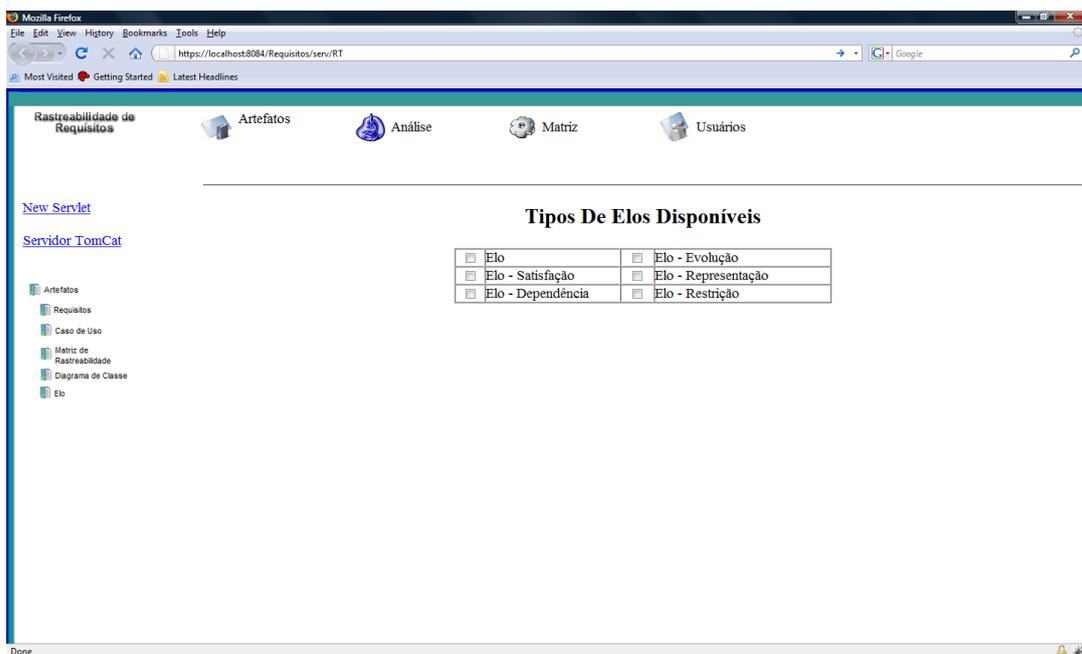


Figura 4.8 - Controle de matrizes – elos disponíveis.



Figura 4.9 - Controle de matrizes – matriz escolhida para edição.

O Controle de Matrizes permite definir os artefatos que serão inseridos como origem e destino, Figura 4.10. A aplicação controla a inclusão de artefatos, tanto na origem quanto no destino, não permitindo repetições em uma mesma matriz, gerencia os elos temporários, a inclusão e a exclusão dos elos, assim como o controle dos atributos dos elos da matriz.

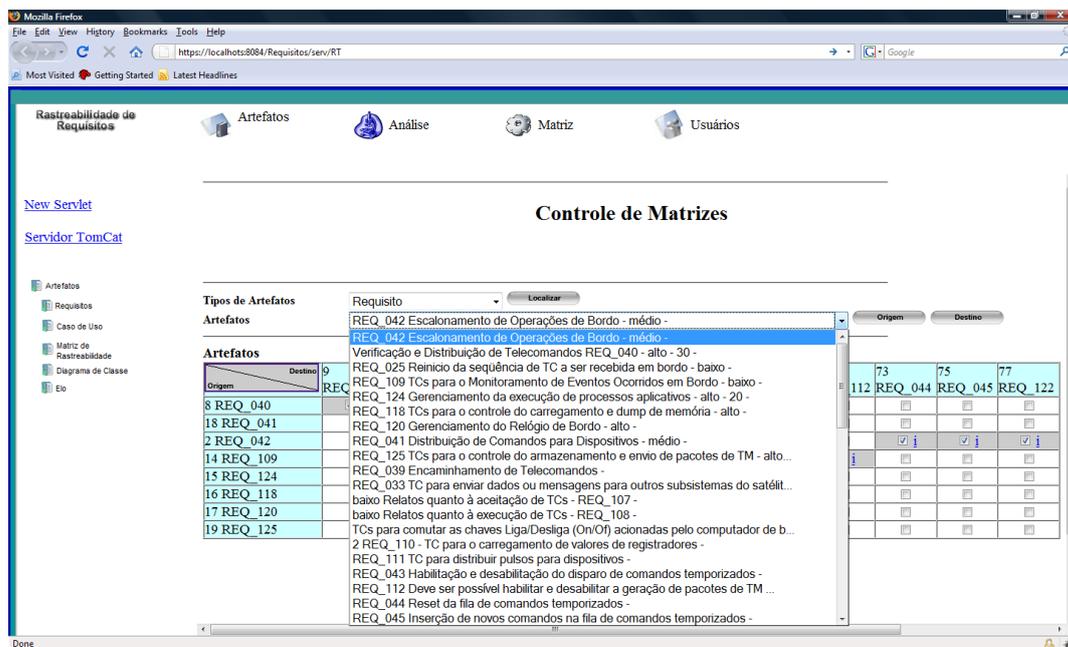


Figura 4.10 - Controle de matrizes – seleção de artefatos para origem e destino.

A Figura 4.11 mostra os atributos pertencentes a um elo. São apresentados o código do elo, a versão, o código do artefato de origem e destino, o esforço (em minutos) despendido para criação do elo e o *Rationale* associado ao elo. Os atributos para o Tipo Elo foram definidos, anteriormente, no controle de Atributos de Tipos.

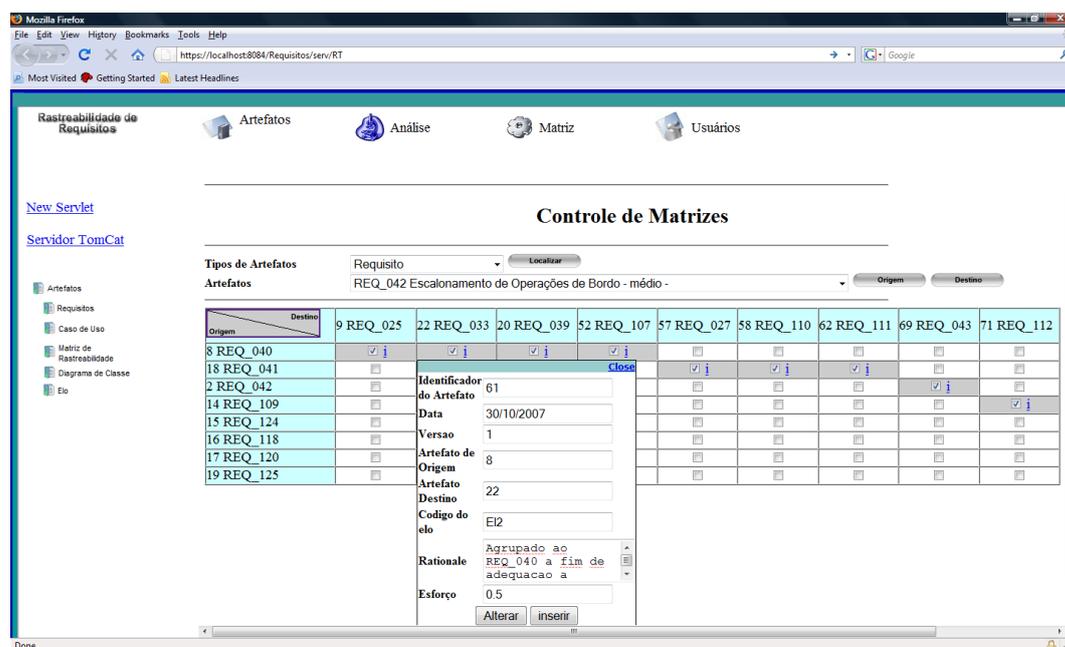


Figura 4.11 - Atributos de um elo em uma matriz.

Ao modificar um atributo de um elo, apenas os dados desse elo são alterados. Isso é realizado a fim de evitar a reconstrução de toda a matriz na tela de apresentação. Esta funcionalidade tem por objetivo diminuir o tempo de resposta ao usuário, facilitando a interação com a aplicação e reduzindo a espera na reconstrução de toda a matriz. Tal funcionalidade é realizada através da tecnologia Ajax.

Os elos inseridos na matriz podem ser editados pelo Controle de Matrizes ou também pelo Controle de Artefatos, isso porque um elo também é um artefato, sendo apresentado como tal no controle de Artefatos. A Figura 4.12 mostra um elo sendo editado pelo controle de artefatos.



Figura 4.12 - Edição de um elo através do módulo de gerenciamento de artefatos.

Após realizar a edição das matrizes é possível realizar algumas análises, como a de impacto. Para este caso é facultada ao usuário a opção de escolher qual artefato será modificado, Figura 4.13, e são mostrados os artefatos que poderão ser impactados pela alteração, Figura 4.14.

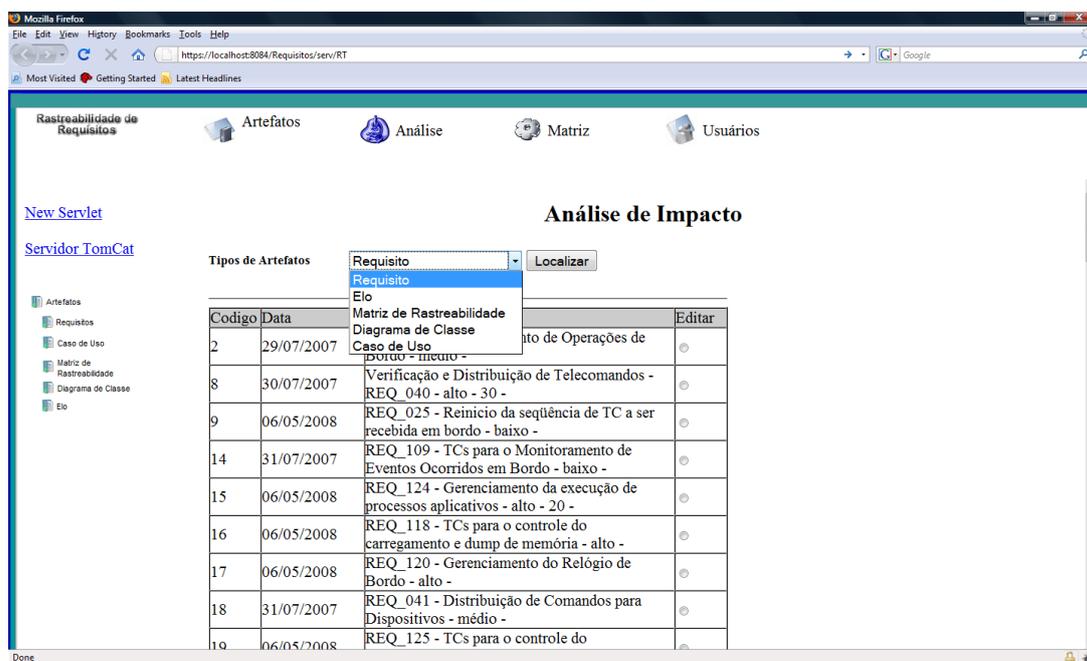


Figura 4.13 - Análise de impacto.

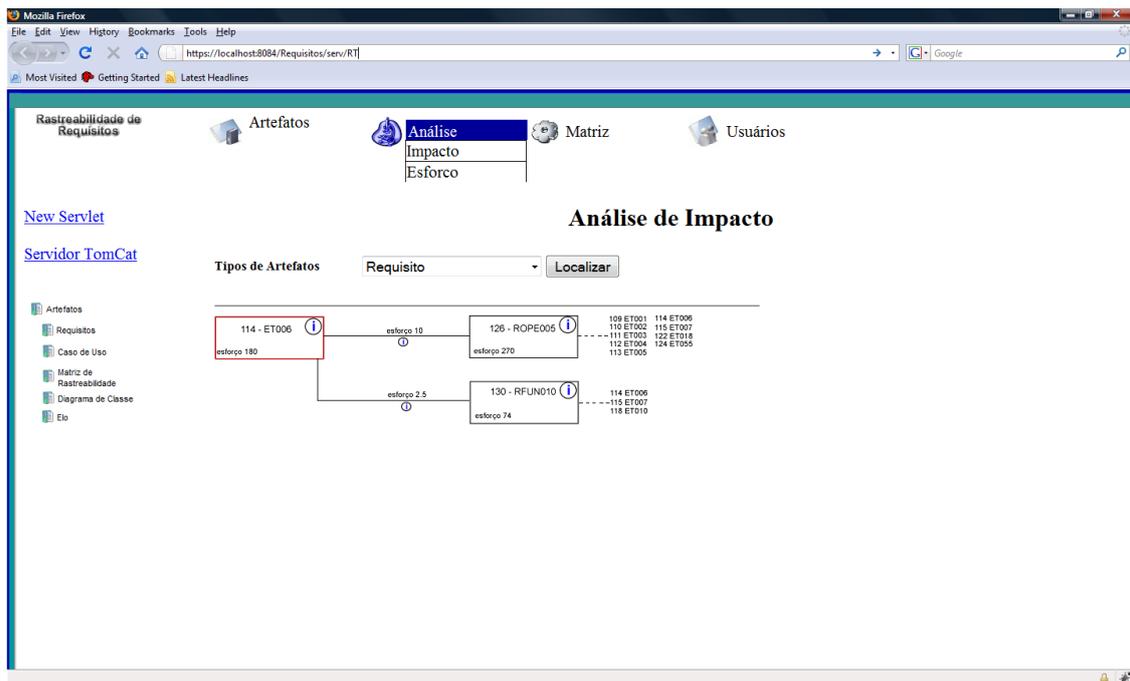


Figura 4.14 - Impacto direto e indireto sobre o requisito ET006.

## 4.5 Conclusão do Capítulo

Neste capítulo foi apresentado o modelo de arquitetura e a aplicação desenvolvida para dar suporte ao modelo para rastreabilidade de requisitos. No capítulo anterior este modelo foi explorado e no próximo será descrito um estudo de caso aplicado para avaliá-lo.

Sobre a arquitetura, vale ressaltar que seu desenvolvimento teve por objetivo demonstrar a viabilidade do modelo conceitual e não cobrir todas as funcionalidades existentes nas aplicações comerciais. A principal meta da aplicação foi servir como um protótipo para validação do modelo e a execução de estudos experimentais e não ser apresentada como um produto final destinado a fins comerciais.

O próximo capítulo apresenta a execução de estudos experimentais realizados para avaliar o modelo.



## 5 ESTUDO DE CASO – EXECUÇÃO DE UM ESTUDO EXPERIMENTAL COM O OBJETIVO DE AVALIAR O MODELO PARA RASTREABILIDADE

Neste capítulo é apresentada a execução de um estudo experimental realizado para demonstrar a viabilidade do modelo proposto. Seu foco está concentrado nas propriedades do modelo para rastreabilidade apresentado neste trabalho.

O estudo experimental foi executado em forma de estudo de caso e teve como foco a definição de matrizes de rastreabilidade, considerando a inserção de informações aos elos da matriz.

Os estudos de caso são conduzidos com o propósito de investigar uma entidade ou fenômeno dentro de um espaço de tempo específico. Esse tipo de estudo experimental é usado para monitorar atributos presentes em projetos, atividades, tecnologias ou modelos.

Na condução de um estudo de caso os dados são coletados e análises estatísticas são realizadas de forma a permitir a avaliação de um atributo ou a relação entre atributos.

A principal diferença entre um estudo de caso e um experimento se dá no nível de controle exercido sobre as variáveis do estudo. Segundo [Zelkowitz e Wallace \(1998\)](#), em um estudo de caso, o nível de controle é menor do que em um experimento e devido a isso, frequentemente, um estudo de caso é caracterizado por ser um estudo de observação, enquanto um experimento é caracterizado por ser um estudo controlado.

### 5.1 Introdução

O estudo experimental foi realizado utilizando-se como referência o modelo GQM (do inglês *Goal Question Metric*) ([BASILI et al., 1994](#)). Este modelo é orientado a metas e sua estrutura é composta por três níveis: Conceitual, que define as metas do experimento; Operacional, que define as questões a serem usadas para caracterizar o que vai ser avaliado; e Quantitativo, que define as métricas que irão compor o grupo de dados associado às questões com o fim de responder a elas de forma quantitativa. A [Figura 5.1](#) apresenta a organização dos níveis do GQM.

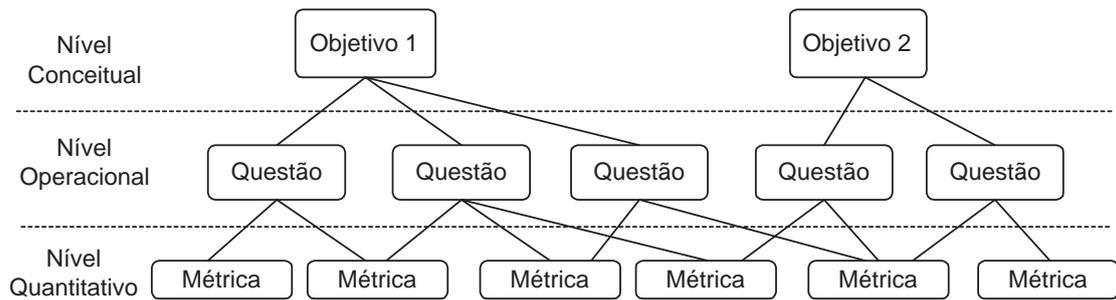


Figura 5.1 - Níveis conceituais do GQM.

Fonte: Adaptado de [Basili et al. \(1994\)](#)

O processo de experimentação aplicado é um refinamento do GQM proposto por [Solingen e Berghout \(1999\)](#). Esse processo é composto pelas atividades de Definição, Planejamento, Interpretação e de Coleta de Dados, [Figura 5.2](#).

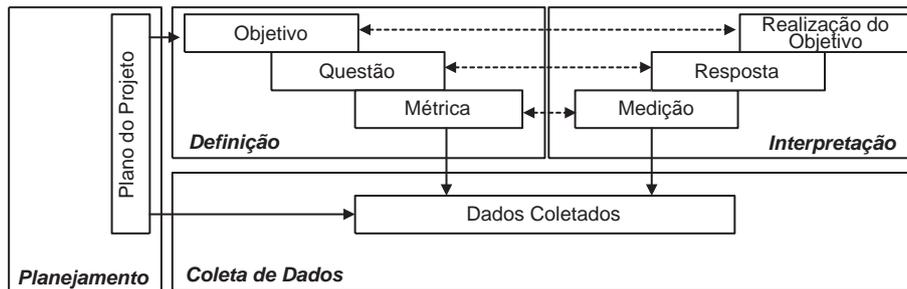


Figura 5.2 - Processo de experimentação utilizado.

Fonte: Adaptado de [Solingen e Berghout \(1999\)](#)

A fase de Definição inclui os três níveis conceituais do GQM em que são descritos os objetivos, as questões e as métricas. O principal resultado desta atividade é fornecer o conhecimento e a direção geral para a realização do experimento.

O Planejamento determina a fundamentação do experimento. Definem-se o contexto, as hipóteses, os instrumentos, identificam-se as variáveis e a selecionam-se os métodos de análise.

A Coleta de Dados define como os dados serão coletados e armazenados para uso

posterior na fase de Interpretação. E, por fim, a Interpretação considera as formas de apresentação dos dados coletados e a elaboração dos resultados obtidos através da análise estatística.

A seguir serão detalhadas cada uma das fases que foram aplicadas na execução do estudo de caso.

## **5.2 Fase de definição**

### **5.2.1 Objetivo global**

O estudo experimental teve por objetivo analisar o modelo apresentado, no que diz respeito à capacidade de fornecer suporte à inserção de atributos para elos de rastreabilidade em matrizes. Especificamente, foram analisados dois atributos: tempo e *Rationales*. Foi considerado o ponto de vista de engenheiros de requisitos e desenvolvedores de software no contexto de três laboratórios de pesquisa do INPE.

### **5.2.2 Objetivo da medição**

Tendo-se como base matrizes de rastreabilidade, utilizadas pelos grupos de desenvolvimento de software na área espacial do INPE, o objetivo é analisar:

- 1 Se o modelo proposto apresenta a inserção de atributos que tratam sobre o tempo usado para a elaboração de cada elo.
  - Qual é o número de elos, em uma matriz que faz uso do modelo proposto, obtidos com informações de tempo;
  - Qual é o número de elos, em uma matriz que não faz uso do modelo proposto, obtidos com informações de tempo;
- 2 Se o modelo proposto apresenta a inserção de atributos que tratam de informações sobre *Rationale* para cada um dos elos.
  - Qual é a razão para a existência dos elos em uma matriz gerada através do modelo.
  - Qual é a razão para a existência dos elos em uma matriz gerada usando-se outro modelo.

### 5.2.3 Objetivo do estudo

O objetivo do estudo é analisar dois padrões de métricas que podem ser estabelecidas no modelo diante das técnicas de representação de matrizes tradicionais, definindo quantitativamente o quanto essas matrizes se diferenciam.

### 5.2.4 Questões

Após a elaboração dos objetivos do experimento são elaboradas as questões que visam caracterizar o que vai ser avaliado a fim de atingir os objetivos estabelecidos.

Cada uma das questões possui uma ou várias métricas que permitirão responder às questões de forma quantitativa.

A seguir são listadas as questões do experimento:

**Q1** - É possível determinar o tempo utilizado em cada elo para as duas matrizes?

**Métricas:** Valor do tempo.

**Q2** - É possível determinar a razão da existência de cada um dos elos das duas matrizes?

**Métricas:** Informações sobre *Rationales* adjacentes aos elos.

## 5.3 Fase de planejamento

### 5.3.1 Contexto

O estudo de caso foi realizado com três equipes de funcionários de laboratórios do INPE, envolvidos com o desenvolvimento de software aplicado à área espacial, que utilizam matrizes de rastreabilidade em seus projetos. Participaram do experimento dois membros de cada uma das três equipes e o experimento teve duração total de cinco meses.

É importante ressaltar que dois grupos solicitaram que suas informações não fossem divulgadas (nome dos grupos e dos membros e, principalmente, os requisitos das aplicações). Tal solicitação foi feita devido à afirmação de que tais informações, utilizadas no experimento, possuem certo grau de sigilo institucional, o que envolve restrições de divulgação. Assim, não serão apresentadas as matrizes com a descrição

completa dos requisitos, e tal fato não comprometeu a realização do experimento nem os resultados finais.

Foi coletada uma matriz de cada grupo com pelo menos trinta elos cada. Após definir-se qual matriz seria avaliada um dos membros do grupo, que participou dos trabalhos originais, refez a matriz, baseada no modelo proposto, criando os relacionamentos, preenchendo os elos e adicionando os novos atributos requeridos para os elos da nova matriz (tempo e *Rationale*). O auxílio de um dos membros foi necessário para se obter essas informações que não estavam presentes na matriz original, como as razões da existência e o tempo empregado na elaboração de cada elo.

O líder do grupo, que não participou dos trabalhos de construção da nova matriz, respondeu às questões do experimento. Para tanto ele analisou a matriz original (sem os atributos de tempo e *Rationale* preenchidos pelo membro) e respondeu ao questionário (Seção A.2) tentando levantar as informações quanto ao tempo empregado para a elaboração de cada elo, o tempo total da construção dos elos da matriz e os *Rationales* dos elos. A participação do líder da equipe visava diminuir a interferência nos resultados, já que os líderes não trabalharam na elaboração das matrizes originais, enquanto que os membros tinham conhecimento sobre as informações dos elos, pois trabalharam na confecção da segunda matriz que requeria tais informações.

- **Vantagens:** Mantém-se o conhecimento prévio do entrevistado para comparar com os resultados reais inseridos na matriz com base no modelo.
- **Desvantagem:** Torna-se um estudo de caso, é mais difícil repetir o estudo experimental.

### 5.3.2 Definição das hipóteses

A definição das hipóteses consistiu na formulação de afirmações que relaciona os itens observados no estudo experimental. Coube, portanto, constatar a veracidade ou não dessas afirmações.

Para cada um dos itens avaliados foram definidas as seguintes hipóteses:

$H_0$  (hipótese nula): Os elos elaborados nas duas categorias de matrizes, com o modelo e sem o modelo, possuem o mesmo grau de qualidade do item avaliado. Assim,

considera-se a hipótese nula por  $H_0 : \delta = 0$  (sem diferença significativa).

$H_A$  (hipótese alternativa): Os elos elaborados nas duas categorias de matrizes, com o modelo proposto e sem o modelo, não possuem o mesmo grau de qualidade do item avaliado. Assim, considera-se a hipótese alternativa por  $H_A : \delta \neq 0$  (existência de diferença significativa).

### 5.3.3 Definição dos instrumentos

Como instrumentos para obtenção das informações para a fase de Coleta de Dados foram utilizados três tipos de questionários (Apêndice A). O primeiro levantou informações sobre o perfil dos participantes (Seção A.1), o segundo sobre as práticas de engenharia de software e rastreabilidade utilizadas pelo grupo (Seção A.2) e o terceiro sobre as matrizes de rastreabilidade e seus elos (Seção A.3). Os dois primeiros questionários tiveram por objetivo avaliar a existência ou não de associações/independência estatísticas sobre o grau de conhecimento e as práticas utilizadas, enquanto que o terceiro forneceu informações para as análises estatísticas dos dados coletados sobre as matrizes.

O questionário 3 (Seção A.3) foi o principal instrumento utilizado com a finalidade de coletar informações para o experimento, sendo composto de questões descritivas e de uma escala Likert (LIKERT, 1932; DEVELLIS, 2003; DAWES, 2008) que teve por objetivo avaliar o grau de confiança das respostas obtidas (Nenhum, Pouco, Mais ou menos, Muita, Completa).

Este último questionário foi utilizado em duas etapas distintas: i) A primeira foi realizada com o líder do grupo que, para responder às questões, teve de analisar a matriz original de seu grupo e determinar o tempo e o *Rationale* dos elos existentes nesta matriz. O líder teve de preencher os itens solicitados no questionário 3 (valores de tempo e o *Rationale* do elo) além do grau de confiança, que ele tinha, para responder a cada um dos dois itens avaliados para todos os elos que compunham a matriz. Vale ressaltar que a matriz, analisada pelo líder, não possuía tais informações, embora o líder tivesse liberdade para pesquisar outras fontes como ferramentas ou a documentação existente, mas nunca consultando outros membros do grupo.

A Tabela 5.1 apresenta um exemplo de matriz a que o líder teve acesso para responder ao questionário 3. Além da matriz o líder recebeu uma documentação complementar, com o objetivo de auxiliar a compreensão da matriz: descrição dos elementos

que compõem a matriz e descrição detalhada desses elementos. Um exemplo desses documentos pode ser visualizado na [Seção A.4](#).

Tabela 5.1 - Exemplo de matriz analisada pelos líderes dos grupos para responder as questões.

Origem \ Destino	Rr1	Rr2	...	Rrn
	R1	x		
R2		x		
⋮		x		
Rn				

### 5.3.4 Identificação das variáveis

Duas classes de variáveis foram consideradas: dependentes e independentes (WOHLIN et al., 2000). As independentes são as entradas do processo de experimentação, podendo ser chamadas de fatores quando são controladas, representam a causa que afeta o resultado do processo de experimentação. As dependentes referem-se às saídas do processo de experimentação sendo afetadas durante o processo, ou seja, representam o efeito da combinação dos valores das variáveis independentes.

As variáveis independentes do experimento são: tempo em minutos e *Rationales* adjacentes à existência dos elos.

As variáveis dependentes do experimento são:

- a) O tempo para cada elo.
  - Valores admissíveis:
    - Iguais para as duas matrizes;
    - Diferentes para as duas matrizes;
    - Não foi possível determinar o tempo.
- b) O tempo total da matriz.
  - Valores admissíveis:
    - Iguais para as duas matrizes;
    - Diferentes para as duas matrizes;

– Não foi possível determinar o tempo.

c) O *Rationale* associado a cada elo.

- Valores admissíveis:
  - Iguais para as duas matrizes;
  - Diferentes para as duas matrizes;
  - Não foi possível determinar o *Rationale*.

### 5.3.5 Seleção dos métodos de análise

Como padrão em experimentação foram utilizados os métodos estatísticos para a análise dos dados coletados.

A estatística descritiva e os testes de análise de distribuição foram realizados e seus resultados utilizados para definir os testes de hipóteses e independência/associação. Foram utilizados testes não-paramétricos como Qui-quadrado e o teste  $U$  de Mann-Whitney.

Como ferramentas para suporte e realização dos testes estatísticos foram utilizados os softwares: SPSS versão 7.0, Excel versão 2007 e SigmaStat versão 3.5, e também o software GraphPad Prism versão 5.0 para a elaboração de alguns gráficos.

## 5.4 Fase de coleta de dados

Os dados foram coletados em “campo”, ou seja, nos laboratórios definidos na fase de planejamento. A coleta foi feita através de questionários e suas informações foram transferidas para planilhas utilizadas para as análises estatísticas.

## 5.5 Fase de interpretação

A fase de Interpretação foi realizada após as fases de coleta e análise dos dados. Primeiramente, são apresentados os valores coletados para as variáveis tempo e *Rationale* e por fim é apresentada uma discussão sobre os resultados obtidos.

### 5.5.1 Variável tempo

A primeira variável analisada foi o Tempo, definida pelo intervalo em minutos que o membro, de cada um dos três grupos, demorou para estabelecer cada um dos trinta

elos da matriz e pelo intervalo em minutos que o líder, de cada um dos três grupo, estimou que o membro de seu grupo utilizou para realizar a tarefa. A [Tabela 5.2](#) apresenta os dados coletados para estabelecer manualmente os elos nas matrizes pelos membros (real) e o tempo que os líderes consideraram que foi utilizado (estimado).

Tabela 5.2 - Tempo gasto em minutos pelos membros e estimado pelos líderes.

Elos	Grupo1		Grupo2		Grupo3	
	Tempo real(membro)	Tempo estimado(líder)	Tempo real(membro)	Tempo estimado(líder)	Tempo real(membro)	Tempo estimado(líder)
1	15	60	1	0,1	7,5	5
2	20	60	1	0,1	8	5
3	30	30	2	0,5	12	10
4	15	60	3	1	10	10
5	22	60	5	1	14	10
6	14	60	2	0,5	13	10
7	40	30	20	0,5	11	5
8	18	60	15	1	8	5
9	23	60	1	0,5	9	15
10	26	30	20	1	14	10
11	27	60	1	1	13	10
12	17	60	4	0,1	5	5
13	13	30	4	0,1	10	10
14	30	30	4	0,5	9	5
15	12	60	1	0,5	8	5
16	24	60	4	2	13	10
17	19	30	1	2	15	5
18	10	60	20	3	6,5	5
19	32	60	20	3	7	5
20	16	60	2	0,5	8	5
21	36	60	5	0,5	15	5
22	25	60	1	1	8	5
23	14	60	1	1	12	5
24	41	60	1	0,1	14	5
25	7	30	1	1	3	5
26	21	60	1	1	6	5
27	23	60	4	2	10	5
28	13	60	1	2	5	5
29	12	60	1	1	6	15
30	22	60	15	3	4	15

A [Tabela 5.3](#) apresenta as médias e outras estatísticas descritivas dos dados coletados.

Tabela 5.3 - Estatísticas descritivas dos dados coletados sobre o tempo utilizado nos três grupos.

Estatísticas	Grupo1		Grupo2		Grupo3	
	Tempo real(membro)	Tempo estimado(líder)	Tempo real(membro)	Tempo estimado(líder)	Tempo real(membro)	Tempo estimado(líder)
Média	21,20	53,00	5,40	1,10	9,50	7,30
Desvio padrão	8,70	12,90	6,79	0,87	3,43	3,40
Variância	75,70	166,55	46,18	0,76	11,01	11,61
Mediana	20,5	60	2	1	9	5

É interessante observar a diferença entre as médias dos membros e dos líderes de cada grupo. Com exceção do grupo 1, as médias apontam uma subestimação do tempo real utilizado na elaboração da matriz de rastreabilidade, ou seja, os líderes dos grupos 2 e 3 acreditam que o tempo envolvido para estabelecer os elos na matriz é menor que o tempo real utilizado. Apenas no grupo 1 houve o caso contrário, uma superestimação do tempo real gasto; seu líder aponta um tempo muito maior para realizar a tarefa do que realmente foi despendido.

As diferenças existentes nos dados sobre o tempo real e o estimado, de cada um dos grupos, podem ser observadas através de gráficos de linhas. A [Figura 5.3](#) apresenta os dados do grupo 1, a [Figura 5.4](#), do grupo 2 e a [Figura 5.5](#), do grupo 3.

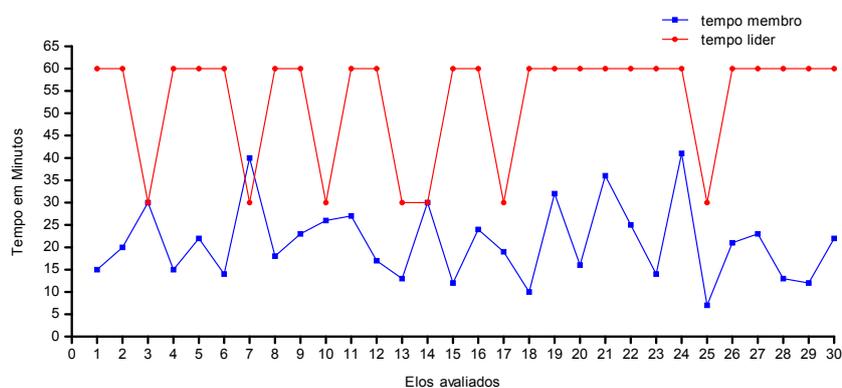


Figura 5.3 - Tempo utilizado pelo membro e estimado pelo líder do grupo 1.

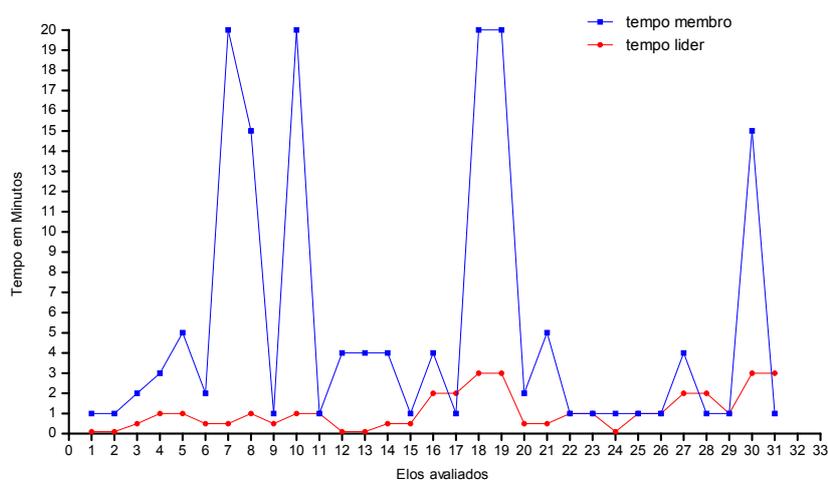


Figura 5.4 - Tempo utilizado pelo membro e estimado pelo líder do grupo 2.

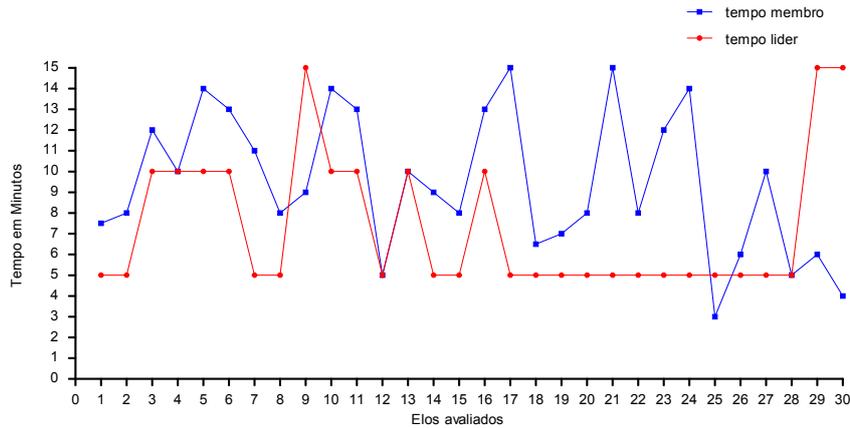


Figura 5.5 - Tempo utilizado pelo membro e estimado pelo líder do grupo 3.

Todavia para os três grupos existe uma variação considerável, no tempo estimado (para mais ou para menos). Ao comparar o tempo real utilizado e o tempo estimado pelo líder é possível observar um número expressivo de erros, [Tabela 5.4](#).

Tabela 5.4 - Quantidade de acertos e erros (por parte) dos líderes em relação ao tempo.

Padrão avaliado	Líderes		
	Grupo1	Grupo2	Grupo3
Número de acertos	2	6	4
Número de erros	28	24	26

Ao considerar o teste de hipótese, sobre os dados, também são observadas diferenças significativas sobre as médias das amostras. Para esta última análise foram consideradas as hipóteses nula e alternativa para as médias dos membros e dos líderes para os três grupos:

$H_0 : \delta = 0$  (sem diferença significativa) e  $H_A : \delta \neq 0$  rejeitar  $H_0$  se  $p \leq 0,05$ . Caso contrário, não é possível afirmar que as médias são significativamente diferentes.

Foi aplicado o teste de normalidade, que evidenciou que a distribuição não se assemelha à curva normal; devido a isso foi utilizado o teste  $U$  de Mann-Whitney para verificar a existência de diferença significativa entre as médias dos tempos dos membros e dos líderes.

A [Tabela 5.5](#) apresenta os resultados obtidos pelo software SPSS para os dados

sobre tempo do grupo 1, enquanto que a [Figura 5.6](#) apresenta um gráfico *Box Plot* demonstrando o primeiro e o terceiro quartis e a mediana entre o tempo despendido pelo líder e o despendido pelo membro.

Tabela 5.5 - Resultados obtidos para o Teste *U* do grupo 1.

Grupo 1	Número de elos	Média dos Postos	Somatória dos Postos
Membro	30	16,67	500
Líder	30	44,33	1330
Total	60		

Variáveis de controle para o teste <i>U</i>		
Mann–Whitney <i>U</i>		35,0
Nível de Significância $\alpha$		0,05
Valor <i>Z</i>		- 6,327
Valor crítico para <i>Z</i>	<i>p</i>	<0,001
Se $p \leq 0,05$ rejeitar $H_0$		Rejeitar

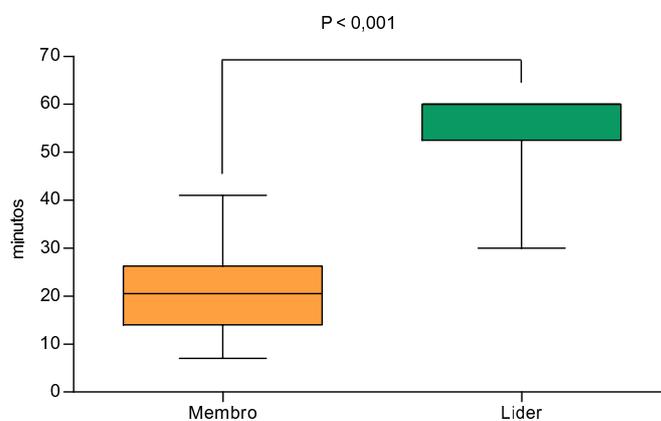


Figura 5.6 - Gráfico *Box Plot* dos dados sobre tempo do grupo 1.

A [Tabela 5.6](#) apresenta os resultados obtidos pelo software SPSS para os dados sobre tempo do grupo 2, enquanto que a [Figura 5.7](#) apresenta um gráfico *Box Plot* demonstrando o primeiro e terceiro quartis e a mediana entre o tempo despendido pelo líder e o despendido pelo membro.

Tabela 5.6 - Resultados obtidos para o Teste  $U$  do grupo 2.

Grupo 2	Número de elos	Média dos Postos	Somatória dos Postos
Membro	30	39,75	1192,50
Líder	30	21,25	637,50
Total	60		

Variáveis de controle para o teste $U$		
Mann-Whitney $U$		172,50
Nível de Significância $\alpha$		0,05
Valor $Z$		- 4,236
Valor crítico para $Z$	$p$	<0,001
Se $p \leq 0,05$ rejeitar $H_0$		Rejeitar

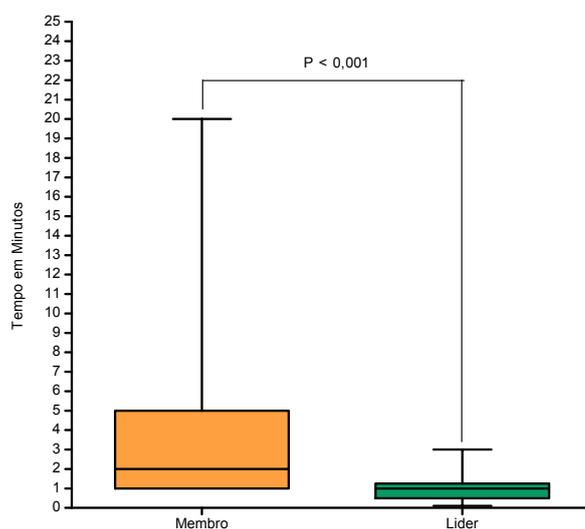


Figura 5.7 - Gráfico *Box Plot* dos dados sobre tempo do grupo 2.

A [Tabela 5.7](#) apresenta os resultados obtidos pelo software SPSS para os dados sobre tempo do grupo 3, enquanto que a [Figura 5.8](#) apresenta um gráfico *Box Plot* demonstrando o primeiro e terceiro quartis e a mediana entre o tempo despendido pelo líder e o despendido pelo membro.

Tabela 5.7 - Resultados obtidos para o Teste  $U$  do grupo 3.

<b>Grupo 3</b>	<b>Número de elos</b>	<b>Média dos Postos</b>	<b>Somatória dos Postos</b>
Membro	30	36,03	1081,00
Líder	30	24,97	749,00
Total	60		

<b>Variáveis de controle para o teste <math>U</math></b>		
Mann-Whitney $U$		284
Nível de Significância $\alpha$		0,05
Valor $Z$		- 2,518
Valor crítico para $Z$	$p$	= 0,012
Se $p \leq 0,05$ rejeitar $H_0$		Rejeitar

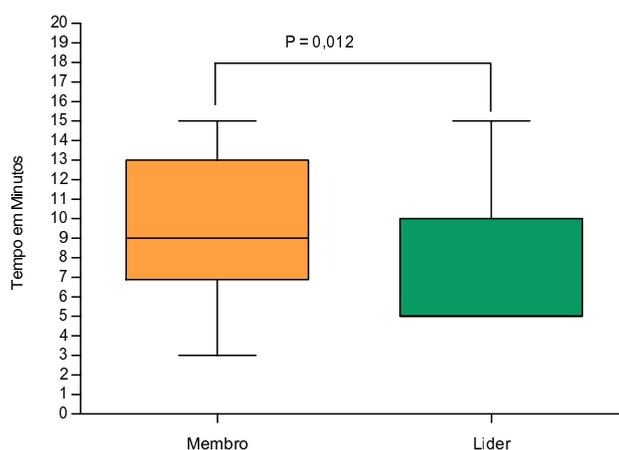


Figura 5.8 - Gráfico *Box Plot* dos dados sobre tempo do grupo 3.

Segundo os resultados observados, ao nível de confiança de 95%, é rejeitada a hipótese de que os tempos, utilizados e estimados, sejam iguais para as avaliações coletadas, ou seja, para os três grupos existe diferença significativa entre os tempos utilizados pelos membros, para desenvolver os elos na matriz, e o tempo que os líderes estimaram que foi usado para esta tarefa.

Tal afirmação permite verificar que informações como o tempo, empregado para estabelecer um elo manualmente em uma matriz, é uma importante variável, não podendo ser controlada simplesmente por experiência prévia. Dessa forma, torna-se

importante o armazenamento desta informação junto ao artefato elo.

Esse tipo de informação seria útil para áreas como a gerência de projeto, na qual poderia ser considerado o cronograma estimado para o projeto e o cronograma real com os tempos reais despendidos para a realização da tarefa. Possivelmente, o tempo real aplicado seria diferente entre o cronograma proposto e o executado; além disso, a informação tempo, utilizado para criar os elos, poderia ser útil na análise de impacto, apresentando não só os artefatos que seriam impactados em uma mudança, mas também definindo qual tempo de desenvolvimento seria associado a esses artefatos e aos elos envolvidos. Assim, caso fosse necessário criar novos elos, a gerência de projeto poderia utilizar tais dados para propor uma previsão de custo e tempo com um maior grau de refinamento e precisão.

### 5.5.2 Variável *rationale*

A segunda variável analisada foi o *Rationale*. Este recurso permite capturar as razões e as justificativas das decisões tomadas ao longo do processo de desenvolvimento.

Existem diferentes formas de representação para *Rationales*, como: representações informais que capturam os *Rationales* através de descrição em linguagem natural (LEE, 1997); registros de áudio ou vídeo, gráficos ou figuras; representações baseadas em argumentação que usam nós e elos para representar o conhecimento e os relacionamentos — tal técnica inclui métodos como Questões, Opções e Critérios (do inglês *Questions, Options and Criteria* (QOC))(MACLEAN et al., 1991), o método Sistema de Informação Baseado em Questões (do inglês *Issue-Based Information System* (IBIS))(RITTEL; WEBBER, 1974) e o método Linguagem de Representação de Decisão (do inglês *Decision Representation Language* (DRL))(LEE, 1991); representações baseadas em argumentação semiformais como o método SEURAT (do inglês *Software Engineering Using Rationale*)(BURGE; BROWN, 2008); e, por fim, representações baseadas em exemplos.

Nesse experimento, o foco da representação utilizada foi a linguagem natural, pois não houve o objetivo implementar nenhuma técnica específica de coleta e representação de *Rationales* nem avaliar vantagens ou desvantagens entre as diferentes técnicas.

Foram avaliados os *Rationales* considerados pelo membro, para cada um dos trinta elos existentes na matriz, e os respectivos *Rationales* que os líderes consideraram

existir para os mesmos elos. Além dos elos foi disponibilizada uma escala Likert para cada um dos *Rationales* respondidos pelos líderes. Dessa forma, para cada descrição de *Rationale*, o líder deveria escolher um item da escala Likert correspondente ao grau de confiança de sua resposta.

A escla Likert utilizada era composta de cinco itens (1 – Nenhum, 2 – Pouco, 3 – Mais ou menos, 4 – Muita, 5 – Completa (Seção A.3)), e tinha por objetivo avaliar o grau de confiança na resposta dada pelo líder.

Após coletar os *Rationales* reais, descritos pelos membros de cada grupo, foram coletados os *Rationales*, que os líderes consideravam existentes, para os mesmos elos, a escala Likert e, por fim, foram comparadas as respostas entre membros e líderes.

A Tabela 5.8 apresenta os escores para acertos e erros, das respostas dos líderes, dos três grupos para a variável *Rationale*. Esses dados também podem ser observados através de um gráfico de barras na Figura 5.9.

Tabela 5.8 - Número de respostas corretas e erradas de *Rationales* entre membros e líderes dos três grupos.

Grupos	Número de acertos	Número de erros
Grupo1	12	18
Grupo2	9	21
Grupo3	6	24

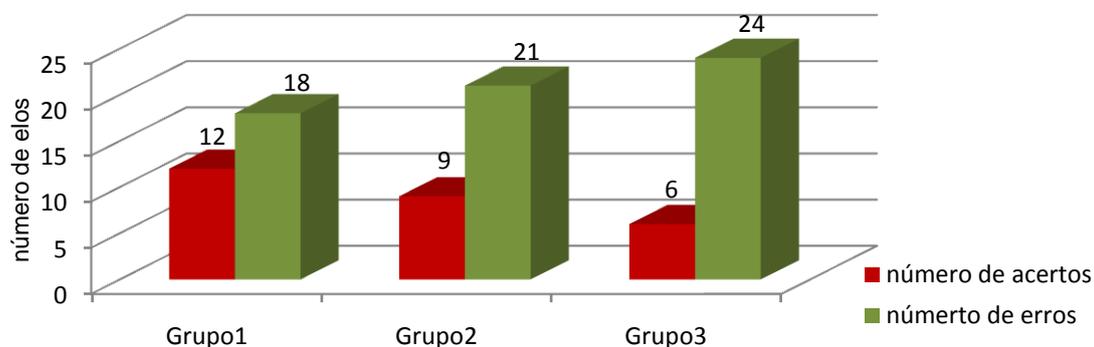


Figura 5.9 - Número de respostas corretas e erradas de *rationales* entre membros e líderes dos três grupos.

Além do número de acertos e erros obtidos pelo líder é possível analisar o número de acertos e erros dos líderes dos três grupos em relação às respostas dadas para a escala Likert (Figura 5.10).

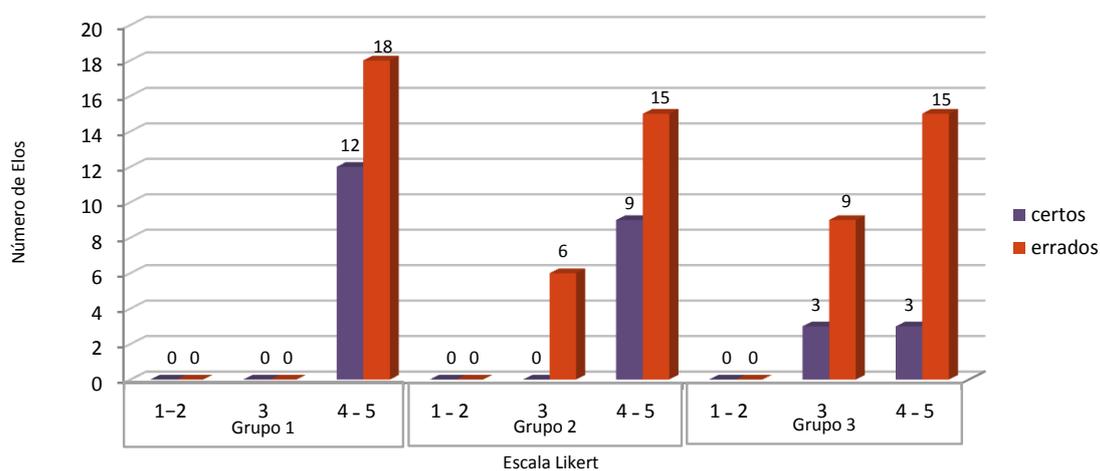


Figura 5.10 - Número de *racionais* corretos e errados, dos três grupos, com os respectivos valores de resposta para a escala Likert.

Segundo Likert (1932), os itens da escala Likert podem ser divididos em três padrões: i) escalas 1 e 2 – negativamente, ii) escala 3 – indiferente e iii) escala 4 e 5 – positivamente. Através da análise dos dados apresentados é possível observar que os líderes, em todos os casos, consideraram que possuíam alto grau de confiança nas respostas que estavam dando, visto que nenhuma resposta para a escala Likert foi dada aos itens 1 e 2.

Dada a característica dos dados, descritivos e escalares, foi possível realizar testes de independência/associação, Qui-Quadrado, entre as variáveis porcentagem de acertos e erros e padrões de resposta para a escala.

Os testes consideraram as seguintes hipóteses:

$H_0$ : As variáveis são independentes e  $H_A$ : rejeitar  $H_0$  se  $p \leq 0,05$ , as variáveis estão associadas.

Foram avaliados os dois níveis mais altos da escala Likert (4 e 5), em razão da ausência ou do baixo número de respostas para os níveis inferiores (Tabela 5.9). Assim, foi

testada a independência/associação entre acertos e altos níveis de confiança e entre erros e altos níveis de confiança.

Tabela 5.9 - Porcentagem de acertos e erros dos líderes entre os grupos nas escalas 4 e 5.

Porcentagem de Acertos			
Escala	Grupo1	Grupo2	Grupo3
4	20%	13%	0%
5	20%	17%	10%
Porcentagem de Erros			
Escala	Grupo1	Grupo2	Grupo3
4	57%	27%	50%
5	3%	23%	0%

Nota: Os dados das escalas 1–2 e 3 não são apresentados nesta tabela; devido a isso apenas os dados do grupo 1, que possui todas as respostas na escala 4–5 possui a somatória de 100% em seus valores.

Os resultados obtidos pelo software SPSS para o teste Qui–quadrado, para associação/independência, são apresentados nas Tabelas 5.10 e 5.11.

Tabela 5.10 - Resultado do teste Qui–quadrado entre as variáveis número de acertos e escalas nos níveis 4 e 5.

Variáveis de controle do teste Qui–quadrado – Acertos	
$\chi^2$	8,339
nível de significância $\alpha$	0,05
graus de liberdade	2
valor $p$	= 0,015
Se $p \leq 0,05$ rejeitar $H_0$	Rejeitar

Tabela 5.11 - Resultado do teste Qui–quadrado entre as variáveis número de erros e escalas nos níveis 4 e 5.

Variáveis de controle do teste Qui–quadrado – Erros	
$\chi^2$	47,798
nível de significância $\alpha$	0,05
graus de liberdade	2
valor $p$	<0,001
Se $p \leq 0,05$ rejeitar $H_0$	Rejeitar

Pelos testes obtidos é possível rejeitar as hipóteses nulas para os dois casos. Ao aceitar a hipótese alternativa assume-se que existe associação entre as respostas erradas (por parte) dos líderes e um alto grau de confiança na escala Likert; é possível observar que também existe associação entre o número de respostas corretas e os altos valores na escala Likert.

O teste Qui-quadrado não permite inferir causa/efeito; desta forma é possível deduzir que os líderes de grupos possuem alto grau de confiança em sua análise; entretanto, os dados apresentam um número elevado de erros nessas respostas.

A existência de associação entre erros e alto grau de confiança e acertos e alto grau de confiança mostra que os gerentes são confiantes em suas respostas, mesmo quando estão errados; por outro lado, permite considerar que eles realmente não se recordam ou não sabem as razões da existência de certos artefatos no projeto, neste caso os elos em matrizes.

Pesquisas (TANG et al., 2007) demonstram que 74,2% dos gerentes de projetos esquecem em um curto espaço de tempo, de suas próprias decisões. Se decisões em nível de projeto são esquecidas em um curto espaço de tempo, as decisões sobre a existência de determinados elos entre artefatos também podem ser afetadas pelo mesmo problema.

Informações sobre a razão pela qual um determinado elo foi estabelecido nem sempre seguem padrões técnicos; como exemplo, é possível citar a existência de diversos casos em que os artefatos foram agrupados, desassociados, ou mesmo criados a fim de atender normas ou requisitos para a organização do projeto, sendo que os mesmos não estavam previstos nas especificações iniciais do projeto.

## 5.6 Discussão

Ao longo da análise, alguns comentários foram realizados a respeito das hipóteses investigadas e foram apresentados vários resultados obtidos. A fim de complementar a análise é feita, a seguir, uma discussão sobre o estudo experimental executado.

No contexto estatístico, os resultados da análise foram obtidos ao nível de significância de 95%. Com relação ao tamanho da amostra deve-se ter consciência de que é pequeno; no entanto, o número de elos analisados (noventa elos) é estatisticamente representativo e, além disso, deve ser considerada a importância do ambiente em que

os dados foram coletados: laboratórios de desenvolvimento de software do segmento espacial em projetos reais.

Sobre o propósito do estudo experimental, houve bons resultados. Primeiramente foi analisada a capacidade de inserção de dois tipos de atributos: tempo e *Rationale*. Foram considerados esses dois atributos porque eles ainda não foram explorados, academicamente, no contexto dos elos de rastreabilidade. Além disso, esses dois atributos, fazem parte do domínio de conhecimento dos envolvidos; mesmo de forma *ad hoc*, os envolvidos no processo de desenvolvimento possuem conhecimentos básicos sobre esses atributos.

Os trabalhos de pesquisa, realizados atualmente na área de rastreabilidade e *Rationales*, possuem foco nos *Rationales* de projeto sendo que poucos trabalhos abordam os *Rationales* no contexto da rastreabilidade, e, quando o fazem, a pesquisa é dirigida à definição de um tipo de elo para relacionar os artefatos aos seus *Rationales* de projeto.

O experimento permitiu afirmar que os *Rationales* também podem ser empregados nos elos, não como elo, mas como atributo para documentar os *Rationales* existentes em um determinado elo. Para o caso de aplicações espaciais, abordadas na pesquisa, foram encontrados alguns padrões para a justificativa de elos como adaptação, agregação, agrupamento e desassociação.

Os principais itens apontados para a existência de razões para certos elos se deu no contexto de adaptação à norma, como no caso das normas ECSS, em que vários componentes foram criados, agrupados ou removidos a fim de atender os padrões estabelecidos.

Sobre os valores coletados para as variáveis do experimento cabem algumas observações. Nas Figuras 5.3, 5.4 e 5.5 podem ser visualizadas algumas diferenças significativas entre os dados dos líderes e dos membros e também é visualizadas respostas seguindo certo padrão, principalmente por parte dos líderes. Entretanto, para o tipo de estudo experimental executado (estudo de caso) tais fatores são dificilmente controlados.

Uma alternativa para contornar as distorções observadas nos dados seria a adoção de outros tipos de estudos experimentais que permitam, ao mesmo tempo, avaliar um número pequeno de grupos ou indivíduos e um maior controle sobre as variáveis.

Para tanto os experimentos cruzados (do inglês *cross-over*) podem ser uma alternativa para a análise do estudo experimental a ser realizado. Todavia, [Kitchenham et al. \(2007\)](#) e [Babar et al. \(2006\)](#) apontam severas ressalvas para esse tipo de estudo quando aplicado a engenharia de software. Os autores apontam a existência de problemas no uso de experimentos cruzados devido à natureza dos tratamentos utilizados na engenharia de software. Tal característica torna difícil a garantia da não existência do efeito período por interação do tratamento. Esse efeito ocorre quando o tratamento é afetado pelo período no qual ele é realizado implicando a existência de efeitos residuais (do inglês *carryover*) (efeito de tratamento que persiste depois do período de observação experimental).

Para solucionar os problemas existentes na análise cruzada deve-se ter certeza, antes da realização do estudo, da não existência do período por interação do tratamento ou que este é pequeno quando comparado com os efeitos do tratamento utilizado, para tanto o uso de randomização nas avaliações por período pode ser um recurso que minimize o efeito período por interação do tratamento. Contudo, a randomização como recurso de seleção de tratamento não é comumente aplicada a estudos de caso em ambientes reais, pois tal alternativa pode modificar as práticas do trabalho inerentes ao grupo em estudo, o que transformaria o estudo de caso em um experimento. Outro ponto é que para o estudo apresentado as variáveis analisadas (tempo e rationale) são afetadas pelo controle randômico e possuem período por interação do tratamento, desta forma os valores observados no experimento apresentam dados reais do ambiente de desenvolvimento, em que fatores ambientais como estímulos e suas faltas estão presentes.

Outro ponto a ser considerado é que não foi executado no estudo de caso a inserção de diferentes tipos de elos. Isso ocorreu devido à ausência de projetos que fazem uso de vários tipos de elos, o que impediu a execução de um estudo experimental que envolve tais artefatos. [Ramesh e Jarke \(2001\)](#) afirmam que usuários sofisticados lançam mão de vários tipos de elos. Entretanto, ao analisar dez instituições, públicas e privadas, para executar o estudo experimental, os autores deste trabalho, não encontraram nenhum projeto que fizesse uso de vários tipos de elos, mesmo quando avaliaram setores de empresas, como fábricas de projeto, com nível de certificação CMMI nível 4. Em razão disso não foi possível realizar análises sobre diferentes tipos de elos no estudo experimental executado nesta tese. Todavia, isso não compromete a capacidade do modelo em definir diferentes tipos de elos, além do fato de que

outros estudos de caso poderão vir a ser executados, no futuro, em projetos que empreguem vários tipos de elos.

Um item que deve ser observado é que a simples opção para alocação de uma informação junto a um elo já permite estabelecer um padrão de documentação, o que não existia até então nos projetos envolvidos no estudo de caso.

O estudo experimental permitiu demonstrar como o modelo, apresentado neste trabalho, pode ser aplicado em casos reais e que o mesmo apresenta uma alternativa para a melhoria da rastreabilidade de requisitos.

O próximo capítulo apresenta as conclusões do trabalho.

## 6 CONCLUSÃO

O uso adequado da rastreabilidade pode facilitar uma série de atividades do processo de desenvolvimento, podendo também auxiliar na melhoria da qualidade, tanto para o processo quanto para o produto de software.

Apesar de sua grande importância alguns pontos permanecem em aberto para serem explorados como futuras linhas de pesquisa.

Ao longo desta tese foram apresentados alguns dos resultados obtidos na pesquisa realizada e este capítulo tem por objetivo apresentar a conclusão do trabalho. Outros resultados relevantes e trabalhos futuros, se realizados, poderão beneficiar a área de rastreabilidade.

### 6.1 Resultados

O trabalho apresentou um modelo para rastreabilidade de requisitos que permite a representação de diferentes artefatos de software, com atenção especial aos elos de rastreabilidade e aos atributos que podem fazer parte desses elos. Foi conduzida uma análise sobre as características dos atuais modelos e, através da modelagem, foi possível elaborar um modelo que generaliza essas características. Apesar do modelo ser aparentemente simples, ele permite incluir os elos propostos nos outros modelos pesquisados e outros tipos de artefatos, além dos atributos para esses elementos.

A pesquisa realizada mostra que os atuais esforços, sobre modelos para rastreabilidade, focam a definição dos modelos para áreas específicas, como o padrão de informação utilizada ou os usuários envolvidos com o processo de rastreabilidade. Contudo, a prática mostra que as técnicas em rastreabilidade utilizadas pela indústria de software são simples. Também é observado que a expectativa dos envolvidos com a rastreabilidade tem foco em itens específicos do processo e não possui interesse no uso de linguagens complexas para descrever os requisitos a fim de estabelecer os elos através de recursos automatizados.

Ainda sobre os modelos, ficou evidente a perspectiva de que uma melhor compreensão das relações existentes entre modelos, modeladores e o que é modelado poderá servir de base para novas pesquisas e práticas sobre a engenharia de requisitos. Tal fundamento poderá lidar com um melhor entendimento do modo como as propriedades da rastreabilidade poderão ser identificadas ou integradas em modelos ([MORRIS](#);

[GOTEL, 2007](#)).

A arquitetura foi elaborada a fim de facilitar o projeto para o protótipo. Seu desenvolvimento permitiu explorar outros aspectos como uma introdução ao processo de rastreabilidade. Um exemplo disso é encontrado no Caso de Uso “Gerenciar Artefatos”, que foi modelado anteriormente, através da linguagem SPEM. Nesse Caso de Uso é necessário estabelecer quais os tipos de artefatos que serão tratados pelo processo de rastreabilidade. Essa tarefa acaba por conduzir o responsável pela rastreabilidade a focar aspectos do processo e não simplesmente de ferramental.

A implementação do protótipo foi uma tarefa árdua, mas muito produtiva. Ao se pesquisar sobre as tecnologias de código livre foi possível ampliar o conhecimento sobre as atuais tendências de desenvolvimento e avaliar, na prática, o desenvolvimento de um protótipo. Essa experiência permitiu verificar a diferença de um protótipo, que teve por objetivo implementar um modelo conceitual, e as ferramentas comerciais que exigem um maior nível no controle da qualidade na produção e de facilidades que devem ser oferecidas aos usuários.

É interessante mostrar que outros trabalhos também exploram arquiteturas e outras técnicas para estabelecer elos de rastreabilidade. [Angelina et al. \(2006\)](#) afirmam que a maioria das pesquisas abordam elos do tipo satisfação e dependência e, com base nas diferentes necessidades dos envolvidos, os autores propõem um esquema para especificar elos. O modelo desses autores é baseado na pressuposição da existência de um conjunto mínimo de elos, de um conjunto de métricas e na definição de perfis para usuários. Entretanto, o trabalho não implementa nenhuma aplicação, não há estudos experimentais para avaliar a viabilidade do esquema e é apresentado, apenas, o esquema XML para a definição de elos, ou seja, os elos são criados e mantidos através de recursos de hipertexto.

Outro trabalho, o de [Sharif e Maletic \(2007\)](#), também faz uso de um modelo de elos baseado em XML, especificamente no elemento Xlink. Esse modelo é baseado em elos definidos em formato XML entre diferentes tipos de artefatos; todavia, o modelo não prevê a definição de atributos aos elos e, também, é subentendida a existência de um conjunto de elos já predefinidos. Além disso, os elos desse modelo são focados na evolução dos elos em um nível de granularidade fina. Assim como o trabalho anterior, essa pesquisa apresenta algumas limitações como: i) o modelo supõe que os elos sejam armazenados em arquivos texto em formato XML, ii) não

é apresentada nenhuma ferramenta ou experimentos para avaliar o modelo e iii) devido à quantidade de elos existentes em uma rastreabilidade executada em um nível de granularidade fina, a manutenção, a pesquisa e a integridade referencial em arquivos XML podem ser prejudicadas, o que não ocorre em modelos baseados em banco de dados.

A execução do estudo experimental foi uma boa oportunidade para avaliar o modelo, obtendo-se bons resultados. Foi observada uma melhoria no padrão de documentação da rastreabilidade e também foi possível verificar um estímulo por parte dos envolvidos em aperfeiçoar o padrão de rastreabilidade utilizado pelos grupos. Um dos motivos para esse estímulo foi o trabalho de identificação de informações que normalmente não são registradas, mas que levam a uma discussão sobre o projeto. Isso foi observado em um dos grupos que, após e durante a execução do experimento, passaram a rever a organização dos requisitos. O questionamento sobre o atributo *Rationale*, inserido no elo, conduziu a uma observação mais criteriosa sobre o projeto.

O desenvolvimento do experimento permitiu atingir alguns dos objetivos propostos como conhecer a execução do modelo e visualizar que sua aplicação é possível e viável, resultando em ganhos na qualidade das informações registradas na rastreabilidade.

Também foi positivo o grau de aceitação do modelo e do protótipo, visto que alguns dos laboratórios envolvidos na experimentação despertaram interesse em fazer uso dos recursos desenvolvidos. Além disso, todos os membros consideraram ótimo o uso do modelo.

Deve ser registrada a dificuldade para executar estudos de caso em ambientes reais. De forma geral, houve certa resistência por parte dos envolvidos em cooperar com o experimento. Acredita-se que isso se deve ou ao temor de uma possível comparação do trabalho executado pelo indivíduo/grupo ou a uma certa idéia de posse, pelo indivíduo, das informações do projeto em que ele participa.

Sobre a escolha dos projetos utilizados no estudo, vale algumas observações. Optou-se por utilizar projetos já concluídos, pois a execução de experimentos, em projetos reais novos, possui algumas complicações como:

- i) a demanda por tempo, no qual o avaliador necessita acompanhar o projeto ao longo de sua execução, o que aumenta a demanda de esforço e tempo;

- ii) problemas de descontinuidade ou decisões não-técnicas para os projetos de aplicações espaciais, que podem comprometer a observação experimental;
- iii) projetos que possuem informações sigilosas, nos quais somente decorrendo certo tempo, essas informações podem ser utilizadas para a execução de estudos experimentais.

Uma alternativa para a execução de estudos experimentais, em projetos já executados, seria o uso do ambiente acadêmico. Neste caso, o projeto é executado através de equipes de alunos (graduação ou pós-graduação) que formariam grupos-controle e grupos-estudo, e os resultados obtidos seriam comparados através de um modelo de qualidade. Esse tipo de experimento também possui inconvenientes como:

- i) por ser um ambiente acadêmico, o mesmo não condiz, em muitos aspectos, com os ambientes reais de desenvolvimento das empresas,
- ii) o caso de estudo deve ser mais restrito quanto à complexidade e ao tamanho do projeto, isso se deve ao nível de qualificação dos envolvidos e ao tempo disponível para a execução do estudo (calendário acadêmico).

Além dos resultados descritos, o trabalho permitiu a elaboração de artigos que já foram submetidos e publicados em revista e congressos, e de outros que estão em fase de submissão, [Tabela 6.1](#).

Tabela 6.1 - Artigos publicados

---

<b>Trabalhos já publicados</b>
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Adaptable Documentation and Traceability of Software Requirements. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 6. (WORCAP), 2006, São José dos Campos. Proceedings... São José dos Campos: INPE, 2006. On-line. Disponível em: < <a href="http://urlib.net/dpi.inpe.br/hermes2@1905/2006/10.20.14.48">http://urlib.net/dpi.inpe.br/hermes2@1905/2006/10.20.14.48</a> >. Acesso em: 18 dez. 2008.
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Uma Proposta de Modelo Para a Rastreabilidade de Requisitos. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 7. (WORCAP), 2007, São José dos Campos. Proceedings... São José dos Campos: INPE, 2007.
GENVIGIR, E. C.; SANT'ANNA, N. Comparação do impacto do uso de um processo de engenharia de requisitos entre grupos de desenvolvimento de software: Um estudo de caso. In: WORKSHOP IBEROAMERICANO DE INGENIERÍA DE REQUISITOS Y AMBIENTES DE SOFTWARE, X., 2007, Isla Margaita, Venezuela. Memorias... Caracas, Venezuela: IDEAS'07, 2007. p. 121–134. ISBN–978.980.323.3.70.
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Uma Análise Sobre as Práticas de Engenharia de Software Adotadas no Desenvolvimento de Software em Aplicações Espaciais no INPE. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 8. (WORCAP), 2008, São José dos Campos. Proceedings... São José dos Campos: INPE, 2008.
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Uma Proposta de Modelagem para a Generalização de Elos de Rastreabilidade. Revista de Informática Teórica e Aplicada. XV (2), p. 181–202, 2008. ISSN–01034308.
<b>Capítulo de livro aceito e em processo de publicação</b>
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Handbook of Software Engineering Research and Productivity Technologies: Implications of Globalisation, IGI Global. Chapter title: Requirements Traceability. Editor(s): Ramachandran, M. and Carvalho. R. A. Data prevista para publicação: jul. 2009.

---

## 6.2 Trabalhos futuros

Como contribuição para atividades futuras, esta pesquisa pode fornecer elementos de base para o desenvolvimento de outros trabalhos como:

- O desenvolvimento de técnicas para especificar um conjunto básico de elos para um novo projeto. Duas propostas são delineadas: i) utilizar recursos de inteligência artificial, como sistemas especialistas, a fim de identificar o perfil dos usuários — para tanto, a pesquisa de [Ramesh e Jarke \(2001\)](#) pode servir como base para definir o grau de conhecimento desses usuários, definindo o conjunto de elos a ser disponibilizado pelo modelo — e ii) Servir-se do perfil das atividades do processo e dos artefatos que serão rastreados ao longo do projeto.
- O uso do protótipo como base para a implementação de uma ferramenta que inclua um maior controle sobre o processo de rastreabilidade e mais

facilidades para os usuários. Essa proposta poderá ser implementada através do uso de protocolos de especificação e execução de processos, como o BPEL (OASIS, 2007) e os conceitos sobre BPM (WESKE, 2007) e Serviços Web. Tal idéia foi apresentada inicialmente como proposta para esta tese; no entanto, ela foi reduzida a fim de ajustar o trabalho ao tempo restante para a conclusão do doutorado. Uma arquitetura para essa proposta pode ser visualizada na Figura 6.1.

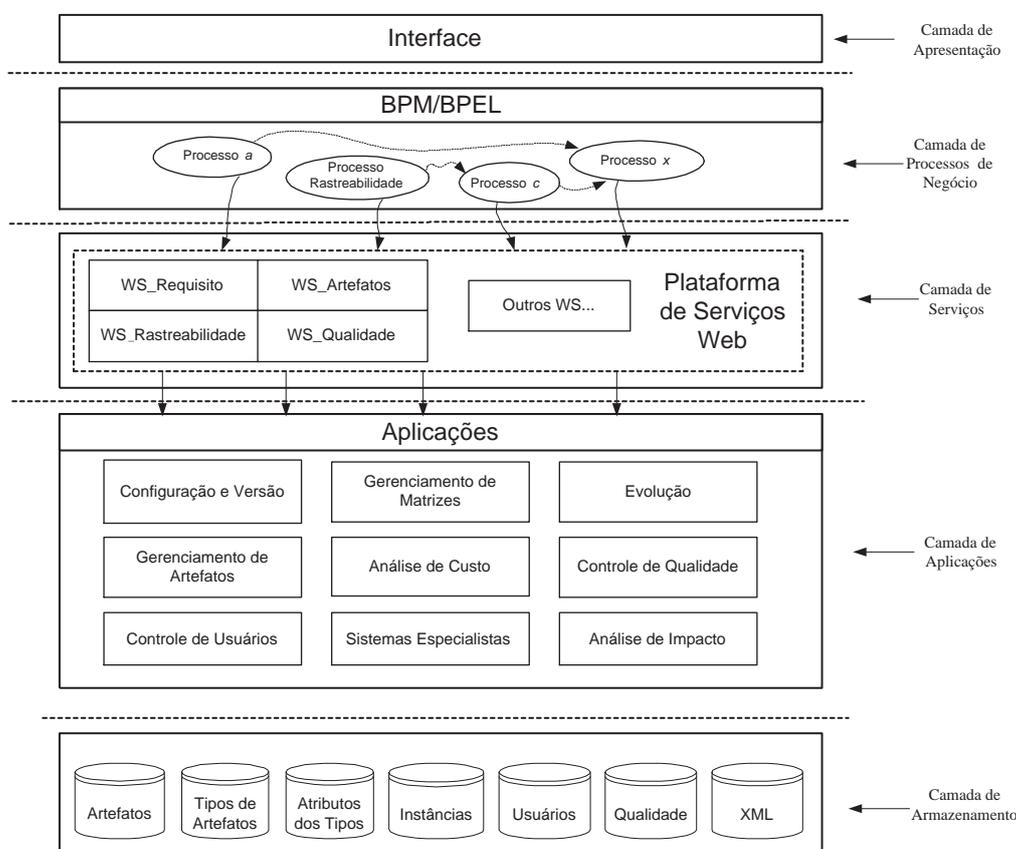


Figura 6.1 - Proposta de trabalho futuro para uma ferramenta para engenharia de requisitos.

- Execução de outros estudos experimentais a fim de avaliar o modelo com outros trabalhos.
- Uso de redes neurais artificiais para identificar padrões e gerar a rastreabilidade automaticamente.

Por fim, também merece atenção especial o aperfeiçoamento dos padrões já existentes e a definição de novas métricas para rastreabilidade. Esses elementos poderão contribuir para a garantia da consistência, da completude, da confiabilidade, da usabilidade e da eficiência da rastreabilidade, assuntos em aberto que merecem discussão e demonstração de viabilidade.



## REFERÊNCIAS BIBLIOGRÁFICAS

AIZENBUD-RESHEF, N.; NOLAN, B. T.; RUBIN, J.; SHAHAM-GAFNI, Y. Model traceability. **IBM Systems Journal**, v. 45, n. 3, p. 515–526, 2006. [42](#), [44](#), [46](#), [47](#), [56](#), [61](#)

ALEXANDER, I. Toward automatic traceability in industrial practice. In: INTERNATIONAL WORKSHOP ON TRACEABILITY IN EMERGING FORMS OF SOFTWARE ENGINEERING, 1., 2002, Edinburgh, Scotland. **Proceedings...** Edinburgh, Scotland: [S.n.], 2002. p. 26–31. [42](#)

\_\_\_\_\_. Semiautomatic tracing of requirement versions to use cases experiences & challenges. In: INTERNATIONAL WORKSHOP ON TRACEABILITY IN EMERGING FORMS OF SOFTWARE ENGINEERING, 2., 2003, Montreal, Quebec, Canadá. **Proceedings...** Montreal, Quebec, Canadá: [S.n.], 2003. Disponível em: <<http://www.soi.city.ac.uk/~gespan/paper6.pdf>>. Acesso em: 03 set. 2008. [51](#)

ALMEIDA, J. P.; ECK, P. V.; IACOB, M. Requirements traceability and transformation conformance in model-driven development. In: INTERNATIONAL ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE, 10., 2006, Hong Kong. **Proceedings...** Washington, USA: IEEE Computer Society, 2006. p. 355–366. ISBN 0-7695-2558-X. [44](#)

ANGELINA, E.; ALARCON, P. P.; J.GARBAJOSA. Analyzing and systematizing current traceability schemas. In: ANNUAL IEEE/NASA SOFTWARE ENGINEERING WORKSHOP, 30th., 2006, Columbia, MD, USA. **Proceedings...** Los Alamitos, Ca, USA: IEEE Computer Society Press, 2006. p. 21–32. ISBN 0-7695-2624-1. [51](#), [114](#)

ANTONIOL, G.; CANFORA, G.; CASAZZA, G.; LUCIA, A. D.; MERLO, E. Recovering traceability links between code and documentation. **IEEE Transactions on Software Engineering**, v. 28, n. 10, p. 970–983, 2002. [43](#)

ANTONIOL, G.; CASAZZA, G.; CIMITILE, A. Traceability recovery by modeling programmer behavior. In: WORKING CONFERENCE REVERSE ENGINEERING, 7., 2000, Brisbane, Qld., Australia. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2000. p. 240–247. ISBN 0-7695-0881-2. [42](#), [43](#)

ANTONIOU, G.; MACNISH, C.; FOO, N. Y. A note on the refinement of nonmonotonic knowledge bases. **Knowledge and Information Systems**, v. 2, p. 479 – 486, 2000. [34](#)

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – ABNT. **NBR ISO/IEC 12207**: tecnologia de informação – processos de ciclo de vida de software. Rio de Janeiro, 1998. [50](#)

BABAR, M. A.; KITCHENHAM, B. A.; ZHU, L.; GORTON, I.; JEFFERY, D. R. An empirical study of groupware support for distributed software architecture evaluation process. **Journal of Systems and Software**, v. 79, n. 7, p. 912 – 925, 2006. [111](#)

BASIL, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. **Encyclopedia of Software Engineering**, v. 1, n. 4, p. 528–532, 1994. [34](#), [91](#), [92](#)

BAUER, F. **Software Engineering**: report on a conference sponsored by the NATO science committee. Garmisch, Germany, Jan. 1969. 231 p. [33](#)

BELFORD, P. C.; BOND, A. F.; HENDERSON, D. G.; SELLERS, L. S. Specifications a key to effective software development. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2., San Francisco, Ca, USA. **Proceedings...** Los Alamitos, Ca, USA: IEEE Computer Society Press, 1976. p. 71–79. [40](#)

BERTOLIN, J. C. G. Uma análise da aplicação de técnicas da psicologia para a elicitación de requisitos de software. In: SEMANA ACADÊMICA DO CPGCC, 3., 1998, Porto Alegre. **Anais...** Porto Alegre: [S.n.], 1998. v. 1, p. 117–120.

Disponível em: <[http:](http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana98/bertolin.html)

[//www.inf.ufrgs.br/pos/SemanaAcademica/Semana98/bertolin.html](http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana98/bertolin.html)>. Acesso em: 20 fev. 2009. [34](#)

BØEGH, J. Software metrics and measurement principles. **IEEE Software Engineering**, v. 25, n. 2, p. 57–63, March/April 2008. [28](#)

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified software development process**. Boston, MA, USA: Addison Wesley Longman, 1999. 463 p. 2. ed. ISBN 0–201–57169–2. [73](#)

BOWEN, J.; O'GRADY, P.; SMITH, L. A constraint programming language for life-cycle engineering. **Artificial Intelligence in Engineering**, v. 5, n. 4, p. 206–220, 1990. [42](#)

BROOKS, F. P. J. **The mythical man-month**: essays on software engineering. Reading – MA: Addison Wesley, 1995. 2.ed. [27](#)

BURGE, J. E.; BROWN, D. C. Software engineering using rationale. **Journal of Systems and Software**, n. 81, p. 395 – 413, 2008. [105](#)

CHECHIK, M.; GANNON, J. Automatic analysis of consistency between requirements and designs. **IEEE Transactions on Software Engineering**, v. 27, n. 7, p. 651–672, 2001. [43](#)

CLELAND-HUANG, J.; CHANG, C. K.; HU, H.; JAVVAJI, K.; SETHI, G.; XIA, J. Automating speculative queries through event-based requirements traceability. In: IEEE JOINT CONFERENCE ON REQUIREMENTS ENGINEERING, Essen, Germany. **Proceedings...** Los Alamitos, Ca, USA: IEEE Computer Society, 2002. p. 289–296. ISBN 0-7695-1465-0. [43](#)

CLELAND-HUANG, J.; CHRISTENSEN, C. Event-based traceability for managing evolutionary change. **IEEE Transactions on Software Engineering**, v. 29, n. 9, p. 796–810, 2003. [29](#), [42](#), [43](#), [44](#), [45](#)

CLELAND-HUANG, J.; SCHMELZER, D. Dynamically tracing non-functional requirements through design pattern invariants. In: WORKSHOP ON TRACEABILITY IN EMERGING FORMS OF SOFTWARE ENGINEERING, 2., Montreal, Canada. **Proceedings...** Montreal, Canada: [S.n.], 2003. [43](#)

CLELAND-HUANG, J.; SETTIMI, R.; LUKASIK, W.; CHEN, Y. Dynamic retrieval of impacted software artifacts. In: MIDWEST SOFTWARE ENGINEERING CONFERENCE, 2003, DePaul University, Chicago, IL, USA. **Proceedings...** [S.l.]: [S.n.], 2003. [43](#)

COOKE, J.; STONE, R. A formal development framework and its use to manage software production. In: COLLOQUIUM BY THE INSTITUTION OF ELECTRICAL ENGINEERS PROFESSIONAL GROUP C1 (SOFTWARE ENGINEERING) – TOOLS AND TECHNIQUES FOR MAINTAINING TRACEABILITY DURING DESIGN, 1., 1991. **Proceedings...** London: IEE – The Institution of Electrical Engineers, 1991. p. 10/1. [42](#)

- COSTELLO, R.; LIU, D. Metrics for requirements engineering. **Journal of Systems and Software**, v. 1, n. 72, p. 39–63, 1995. [47](#), [48](#)
- CRNKOVIC, I.; FUNK, P.; LARSSON, M. Processing requirements by software configuration management. In: EUROMICRO CONFERENCE, 25, York, England, UK. **Proceedings...** Los Alamitos, CA, USA: IEEE Computer Society, 1999. p. 260–265. ISBN 0–7695–0240–7. [37](#)
- DAHLSTEDT, A. G.; PERSSON, A. Requirements interdependencies – moulding the state of research into a research agenda. In: INTERNATIONAL WORKSHOP ON REQUIREMENTS ENGINEERING FORUM ON SOFTWARE QUALITY, 9., 2003, Klagenfurt, Velden, Austria. **Proceedings...** [S.l.]: [S.n.], 2003. p. 71–80. [45](#)
- DAVIS, A. M. **Software requirements: objects, functions and states**. New Jersey: Prentice–Hall, 1993. [39](#), [40](#), [42](#)
- DAWES, J. Do data characteristics change according to the number of scale points used?: an experiment using 5–point, 7–point and 10–point scales. **International Journal of Market**, v. 50, n. 1, p. 61 – 104, 2008. [96](#)
- De LUCIA, A.; FASANO, F.; OLIVETO, R.; TORTORA, G. Enhancing an artifact management system with traceability recovery features. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, 20., 2004. **Proceedings...** [S.l.]: [S.n.], 2004. p. 306–315. [43](#)
- \_\_\_\_\_. ADAMS RE–trace: a traceability recovery tool. In: EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, 9., 2005. **Proceedings...** [S.l.]: [S.n.], 2005. p. 32–41. [46](#), [47](#)
- DEERWESTER, S.; DUMAIS, S. T.; FURNAS, G. W.; K., T.; HARSHMAN, R. Indexing by latent semantic analysis. **Journal of the American Society for Information Science**, v. 41, n. 1, p. 391–407, 1990. [43](#)
- DENG, M.; STIREWALT, R. E.; CHENG, B. H. Retrieval by construction: A traceability technique to support verification and validation of UML formalizations. **Journal of Software Engineering and Knowledge Engineering – IJSEKE**, v. 15, n. 5, p. 837–872, 2005. [43](#)
- DEVELLIS, R. F. **Scale development: theory and applications**. Thousand Oaks, Ca, USA: Sage Publications, 2003. 171 p. 2. ed. ISBN 0761926054. [96](#)

DICK, J. Rich traceability. In: INTERNATIONAL WORKSHOP ON TRACEABILITY IN EMERGING FORMS OF SOFTWARE ENGINEERING, 17., 2002. **Proceedings...** [S.l.]: [S.n.], 2002. p. 35–46. [29](#), [45](#), [58](#)

\_\_\_\_\_. Design traceability. **IEEE Software**, v. 22, n. 6, p. 14–16, Nov. Dec. 2005. [29](#), [45](#), [58](#)

DÖMGES, R.; POHL, K. Adapting traceability environments to project-specific needs. **Communications of the ACM**, v. 41, n. 12, p. 54–62, 1998. [39](#)

EGYED, A. Resolving uncertainties during trace analysis. In: SYMPOSIUM ON FOUNDATIONS OF SOFTWARE ENGINEERING, 12., 2004. **Proceedings...** New York, NY, USA: ACM, 2004. p. 3–12. [44](#)

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION – ECSS. **Space engineering: software part 1 principles and requirements ECSS–E–40 Part1B**. Noordwijk, The Netherlands, Nov. 2003. 112 p. [28](#)

\_\_\_\_\_. **Space engineering: software part 2 document requirements definitions (drds) – ECSSE–40 Part2B**. Noordwijk, The Netherlands, Mar. 2005. 88 p. [28](#)

EVANS, M. W. **The software factory: a fourth generation software engineering environment**. London: Wiley–Interscience, 1989. 332 p. ISBN 0471011924. [42](#)

FAISAL, M. H. **Toward automating the discovery of traceability links**. 118 p. Doctoral Thesis — University Of Colorado, 2005. [43](#)

FILHO, A. M. S. **Arquitetura de software: desenvolvimento orientado para arquitetura**. Rio de Janeiro, RJ: Campus, 2002. 178 p. 1. ed. ISBN 853521013X. [71](#)

GAUSE, D. C.; WEINBERG, G. M. **Exploring requirements: quality before design**. New York, New York, USA: Dorset House, 1989. 300 p. 1. ed. ISBN 0–932633–13–7. [72](#)

\_\_\_\_\_. **Are your lights on?: how to figure out what the problem really is**. New York, New York, USA: Dorset House, 1990. 157 p. 1. ed. ISBN 0932633161. [72](#)

GENVIGIR, E. C. **Um modelo de processo da engenharia de requisitos aplicáveis a ambientes de engenharia de software centrados em processos**. 258 p. (INPE–14630–TDI/1201). Dissertação (Mestrado em Computação Aplicada) — Instituto Nacional de Pesquisas Espaciais (INPE), São

José dos Campos, SP, 2004. Disponível em: <<http://mtc-m18.sid.inpe.br/col/sid.inpe.br/jeferson/2004/07.05.16.17/doc/paginadeacesso.html>>. Acesso em: 10 dez. 2008. 72

GENVIGIR, E. C.; SANT'ANNA, N. Comparação do impacto do uso de um processo de engenharia de requisitos entre grupos de desenvolvimento de software: um estudo de caso. In: WORKSHOP IBEROAMERICANO DE INGENIERÍA DE REQUISITOS Y AMBIENTES DE SOFTWARE, X., 2007, Isla Margarita, Venezuela. **Memorias...** Caracas, Venezuela: IDEAS'07, 2007. p. 121–134. ISBN 978–980–323–3. 72

GENVIGIR, E. C.; SANT'ANNA, N.; BORREGO FILHO, L. F. Modelagem de processos de software através do SPEM – software process engineering metamodel – conceitos e aplicação. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 3., 2003, São José dos Campos, SP. **Anais...** São José dos Campos, SP: INPE, 2003. p. 85–90. ISBN 85–17–00009–9. 66

GENVIGIR, E. C.; VIJAYKUMAR, N. L. Uma análise sobre as práticas de engenharia de software adotadas no desenvolvimento de software em aplicações espaciais no INPE. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA, 7., São José dos Campos, SP, Brasil. **Anais...** São José dos Campos, SP, Brasil: INPE, 2008. 29

GLINZ, M. A lightweight approach to consistency of scenarios and class models. In: INTERNATIONAL CONFERENCE REQUIREMENTS ENGINEERING, 2000, Schaumburg, IL, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2000. p. 49–58. 42

GOMES, N.; GOMES, A. S.; TEDESCO, P.; SERRANO, M. A. B. Aliando teoria da atividade e TROPOS na elicitação de requisitos de ambientes colaborativos de aprendizagem. In: WORKSHOP EM ENGENHARIA DE REQUISITOS, 3., 2003, Piracicaba. **Anais...** Fortaleza: [S.n.], 2003. v. 1, p. 13–29. 34

GOTEL, O. C. Z. **Contribution structures for requirements traceability.** 354 p. Submitted for the Diploma of Imperial College of Science, Technology and Medicine. Thesis (Doctor of Philosophy) — Faculty of Engineering of the University of London, London, 1995. 40, 41, 62

GOTEL, O. C. Z.; FINKELSTEIN, A. An analysis of the requirements traceability problem. In: FIRST INTERNATIONAL CONFERENCE REQUIREMENTS

- ENGINEERING, 1., 1994, Colorado Springs, CO, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society Press, 1994. p. 94–101. [41](#), [42](#), [44](#)
- GRAHAM, I. **Requirements engineering and rapid development**: an object oriented approach. New York, NY, USA: ACM Press/Addison–Wesley Publishing, 1998. 271 p. Harlow. ISBN 0–201–36047–0. [34](#)
- GROSS, D.; YU, E. From non–functional requirements to design through patterns. **Requirements Engineering Journal**, v. 6, n. 1, p. 18–36, 2001. [43](#)
- HAYES, J. H.; DEKHTYAR, A.; OSBORNE, E. Improving requirements tracing via information retrieval. In: IEEE INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING, 11th., 2003, Monterey Bay, California, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2003. p. 138–150. ISBN 1090–705X. [43](#)
- HULL, E.; JAKSON, K.; DICK, J. **Requirements engineering**. London: Spring Verlag, 2002. [49](#), [58](#)
- IEEE COMPUTER SOCIETY PRESS. **Guide to the software engineering body of knowledge – SWEBOK**: trial version (version 1.00). Los Alamitos, California, USA, May 2004. [27](#), [33](#)
- INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS – IEEE. **Standard glossary of software engineering terminology**: IEEE Std – 610.12–1990 IEEE. New York, NY, USA, Sept 1990. [33](#)
- \_\_\_\_\_. **Guide for developing system requirements specifications**: IEEE Std – 1233 IEEE. New York, NY, USA, 1998. 30 p. Item 3.15. [50](#)
- \_\_\_\_\_. **Recommended practice for software requirements specifications**: IEEE Std 830–1998. New York, NY, USA, 1998. [28](#), [50](#)
- INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING – INCONSE. **Tools survey**: requirements management (RM) tools. Seattle, WA, USA, 2008. Disponível em: <<http://www.paper-review.com/tools/rms/read.php>>. Acesso em: 9 set. 2008. [51](#), [52](#), [53](#)
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 15504–2:2003/Cor.1:2004(E)**: information technology – process assessment – part 2: performing an assessment. [S.l.], 2003. [28](#), [51](#)

JACKSON, J. A. Keyphrase based traceability scheme. In: COLLOQUIUM BY THE INSTITUTION OF ELECTRICAL ENGINEERS PROFESSIONAL GROUP C1 (SOFTWARE ENGINEERING) – TOOLS AND TECHNIQUES FOR MAINTAINING TRACEABILITY DURING DESIGN, 1., 1991. **Proceedings...** London: IEE – The Institution of Electrical Engineers, 1991. p. 2/1–2/4. [42](#)

JACKSON, M. **Software requirements and specifications**: a lexicon of practice, principles and prejudices. Massachusets: Addison–Wesley, 1995. 228 p. ISBN 0201877120. [72](#)

JARKE, M. Requirements traceability. **Communications of ACM**, v. 41, n. 12, p. 32–36, 1998. [44](#)

JARKE, M.; POHL, K. Requirements engineering in 2001: managing a changing reality. **IEE Software Engineering Journal**, v. 9, n. 6, p. 257 – 266, Nov. 1994. [34](#)

JUNG, J. J. Ontological framework based on contextual mediation for collaborative information retrieval. **Information Retrieval**, v. 10, n. 1, p. 85–109, 2007. [43](#)

KAINDL, K. The missing link in requirements engineering. **ACM SIGSOFT Software Engineering Notes**, v. 18, n. 2, p. 30–39, 1993. [42](#)

KITCHENHAM, B. A.; FRY, J.; LINKMAN, S. The case against cross-over designs in software engineering. In: INTERNATIONAL WORKSHOP ON SOFTWARE TECHNOLOGY AND ENGINEERING PRACTICE, 11., 2003, Amsterdam, The Netherlands. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2007. p. 65–67. ISBN 0–7695–2218–1. [111](#)

KOBAYASHI, A.; MAEKAWA, M. Need–based requirements change management. In: INTERNATIONAL CONFERENCE AND WORKSHOP ON THE ENGINEERING OF COMPUTER BASED SYSTEMS, 8th., Washington, DC, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2001. v. 2, p. 171–178. ISBN 0-7695-1086-8. [37](#)

KOTONYA, G.; SOMMERVILLE, I. **Requirements engineering**: process and techniques. Chichester, England: John Wiley & Sons, 2000. 294 p. ISBN 0–471–97208–8. [34](#), [36](#), [37](#), [40](#)

LAM, W.; LOOMES, M.; SHANKARARAMAN, V. Managing requirements change using metrics and action planning. In: EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, 3th., Amsterdam, The Netherlands. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 1999. p. 122–128. ISBN 0–7695–0090–0. [37](#)

LANCASTER, F. W. **Information retrieval systems**: characteristics, testing and evaluation. New York: Wiley, 1968. [43](#)

LEE, J. Extending the potts and bruns model for recording design rationale. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 13 th. **Proceedings...** [S.l.]: [S.n.], 1991. p. 114–125. [105](#)

\_\_\_\_\_. Design rationale systems: understanding the issues. **IEEE EXPERT**, n. May/June, p. 78 – 85, 1997. [105](#)

LEFERING, M. An incremental integration tool between requirements engineering and programming in the large. In: IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING, 3., 1993, San Diego, Ca, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 1993. p. 82–89. [42](#)

LEFFINGWELL, D.; WIDRIG, D. **Managing software requirements**: a unified approach. California,USA: Addison Wesley Longman, 2000. 491 p. 2. ed. [36](#), [37](#)

LETELIER, P. A framework for requirements traceability in UML–based projects. In: INTERNATIONAL WORKSHOP ON TRACEABILITY IN EMERGING FORMS OF SOFTWARE ENGINEERING, 2002. **Proceedings...** New York, NY, USA: [S.n.], 2002. p. 30–41. [29](#), [38](#), [44](#)

LIKERT, R. A technique for the measurement of attitudes. **Archives of Psychology**, v. 22, n. 140, p. 1 – 55, 1932. [96](#), [107](#)

LINDSAY, P.; YAOWEI, L.; TRAYNOR, O. A generic model for fine grained configuration management including version control and traceability. In: AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, Sydney, Australia. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 1997. p. 27–36. ISBN 0–8186–8081–4. [37](#)

MACLEAN, A.; YOUNG, R. M.; BELLOTTI, V. M. E.; MORAN, T. P. Questions, options, and criteria: elements of design space analysis.

**Human-Computer Interaction**, v. 6, n. 3, p. 201 – 250, 1991. [105](#)

MARCUS, A.; MALETIC, J. Recovering documentation-to-source-code traceability links using latent semantic indexing. In: IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 25th., 2003, Portland, Oregon, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2003. p. 125–132. [43](#)

MELLOR, S. J.; CLARK, A. N.; FUTAGAMI, T. Model-driven development. **IEEE Software**, v. 20, n. 5, p. 14–18, 2003. [53](#)

MORRIS, S. J.; GOTEL, O. C. Z. Model or mould?: a challenge for better traceability. In: INTERNATIONAL WORKSHOP ON MODELING IN SOFTWARE ENGINEERING, 2007. **Proceedings...** Washington, DC, USA: IEEE Computer Society Press, 2007. p. 56. [114](#)

MURPHY, G. C.; NOTKIN, D.; SULLIVAN, K. J. Software reflexion models: bridging the gap between design and implementation. **IEEE Transactions on Software Engineering**, v. 27, n. 4, p. 364–380, 2001. [43](#)

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION–NASA. **Manager’s Handbook for Software Development**: software engineering laboratory series – SEL–84–101 – revision 1. Greenbelt, Maryland, USA, NOVEMBER 1990. [36](#)

NORMAN, E. F.; MARTIN, N. Software metrics: roadmap. In: CONFERENCE ON THE FUTURE OF SOFTWARE ENGINEERING, 2000, Limerick, Ireland. **Proceedings...** New York, NY, USA: ACM, 2000. p. 357–370. ISBN 1–58113–253–0. [48](#)

NUSEIBEH, B.; EASTERBROOK, S. Requirements engineering: a roadmap. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2000, Limerick, Ireland. **Proceedings...** New York, NY, USA: ACM, 2000. p. 35–46. ISBN 1–58113–253–0. [27](#), [34](#), [38](#)

OBJECT MANAGEMENT GROUP – OMG. **Software process engineering metamodel specification – SPEM**: formal submission, OMG document number formal/05–01–06. [S.l.], Nov. 2005. [66](#)

ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS. **Web services business process execution language version 2.0**: OASIS standard. [S.l.], April 2007. Disponível em: <<http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf>>. Acesso em: 10 fev. 2009. 118

PALMER, J. D. Traceability. In: THAYER, R.; DORFMAN, M. (Ed.). **Software Requirements Engineering**. Los Alamitos, Ca, USA: IEEE Computer Society Press, 2000. p. 412–422. 38, 39, 44

PAPADOPOULOU, P. **Evaluation of a requirements traceability system**. MSc Thesis — Department of Computing – City University, 2002. 44

PINHEIRO, F. **Requirements Traceability**: perspectives on software requirements. In: . Norwell, MA, USA: Kluwer Academic Publishers, 2004. cap. 5, p. 91–115. ISBN 978–1–4020–7625–1. 39

PINHEIRO, F.; GOGUEN, J. An object–oriented tool for tracing requirements. **IEEE Software**, v. 13, n. 2, p. 796–810, 1996. 29, 42, 44

POHL, K. PRO–ART: enabling requirements pre–traceability. In: INTERNATIONAL CONFERENCE ON REQUIREMENT ENGINEERING, 2nd., 1996, Colorado Springs, Colorado, USA. **Proceedings...** Los Alamitos, Ca, USA: IEEE Computer Society, 1996. p. 76–84. ISBN 0–8186–7252–8. 45, 47

RAMESH, B.; EDWARDS, M. Issues in the development of a requirements traceability model. In: INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING, 1993, San Diego, Ca, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 1993. p. 256–259. ISBN 0–8186–3120–1. 40

RAMESH, B.; JARKE, M. Toward reference models for requirements traceability. **IEEE Transactions on Software Engineering**, v. 27, n. 1, p. 58–93, Jan. 2001. 19, 28, 29, 44, 45, 47, 54, 61, 111, 117

RICHARDSON, J.; GREEN, J. Automating traceability for generated software artifacts. In: IEEE INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING, 19., 2004, Linz, Austria. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2004. p. 24–33. ISBN 0–7695–2131–2. 43

RIEBISCH, M.; HUBNER, M. Traceability-driven model refinement for test case generation. In: IEEE INTERNATIONAL CONFERENCE AND WORKSHOPS ON THE ENGINEERING OF COMPUTER-BASED SYSTEMS, 12., 2005, Greenbelt, MD, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2005. p. 113–120. ISBN 0-7695-2308-0. [44](#)

RITTEL, H.; WEBBER, M. Dilemmas in a general theory of planning. **Policy Sciences**, v. 4, p. 155 – 169, 1974. [105](#)

ROCHE, J. M. Software metrics and measurement principles. **ACM SIGSOFT Software Engineering Notes**, v. 19, n. 1, p. 77–85, 1994. [48](#)

SABETZADEH, M.; EASTERBROOK, S. Traceability in viewpoint merging: a model management perspective. In: INTERNATIONAL WORKSHOP ON TRACEABILITY IN EMERGING FORMS OF SOFTWARE ENGINEERING, 3., 2005, Long Beach, Ca, USA. **Proceedings...** New York, NY, USA: ACM, 2005. p. 44–49. [44](#)

SAWYER, P.; SOMMERVILLE, I.; VILLER, S. **Requirements process improvement through the phased introduction of good practice, software process – improvement and practice: requirements engineering adaptation and improvement for safety and dependability – REAIMS (Esprit Project 8649).** LANCASTER, UK, 1998. 19–34 p. Disponível em: <<http://www.comp.lancs.ac.uk/computing/research/cseg/projects/reaims/publications.html>>. [34](#)

SAYÃO, M.; LEITE, R. J. Rastreabilidade de requisitos. **Revista de Informática Teórica e Aplicada**, v. 13, n. 1, p. 57–86, 2004. ISSN 01034308. [39](#)

SERRANO, M. A. B. Elicitação dos requisitos de meta-ambientes de engenharia de software através da análise de negócios. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES, 1997, Fortaleza. **Anais...** Fortaleza: SBC, 1997. p. 33–48. [34](#)

SHARIF, B.; MALETIC, J. L. Using fine-grained differencing to evolve traceability links. In: INTERNATIONAL WORKSHOP ON TRACEABILITY IN EMERGING FORMS OF SOFTWARE ENGINEERING, 7., 2007, Lexington, KY, USA. **Proceedings...** New York, NY, USA: ACM, 2007. p. 30–41. ISBN 1-59593-6017. [114](#)

SHAW, M. The coming-of-age of software architecture research. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 8., 2001, Toronto, Ontario, Canada. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2001. p. 656–664. ISSN 0270–5257. [71](#)

SMITHERS, T.; TANG, M. X.; TOMES, N. The maintenance of design history in AI-based design. In: COLLOQUIUM BY THE INSTITUTION OF ELECTRICAL ENGINEERS PROFESSIONAL GROUP C1 – TOOLS AND TECHNIQUES FOR MAINTAINING TRACEABILITY DURING DESIGN, 1., 1991, London. **Proceedings...** London: IEE – The Institution of Electrical Engineers, 1991. p. 8/1. [42](#)

SOFTWARE ENGINEERING INSTITUTE – SEI – CARNEGIE MELLON UNIVERSITY. **Software Metrics**: SEI curriculum module SEI-CM-12-1.1. Pittsburgh, Pennsylvania, USA, December 1988. [48](#)

\_\_\_\_\_. **CMMI for development**: technical report (CMMI-DEV) CMU/SEI-2006-TR-008. Pittsburgh, Pennsylvania, USA, Feb. 2006. Version 1.2. [28](#), [51](#)

SOLINGEN, R. V.; BERGHOUT, E. **The goal/question/metric method**: a practical guide for quality improvement of software development. Maidenhead, England: McGraw-Hill, 1999. 1. ed. [34](#), [92](#)

SOMMERVILLE, I. **Software engineering**. Harlow: Addison Weley, 1995. 564 p. 5. ed. [34](#)

SOMMERVILLE, I.; SAWYER, P. **Requirements engineering**: a good practice guide. Lancaster, UK: John Wiley & Sons, 1998. [27](#), [36](#)

SPANOUDAKIS, G. Plausible and adaptive requirements traceability structures. In: ACM INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 14., 2002, Ischia, Italy. **Proceedings...** New York, NY, USA: ACM, 2002. p. 135–142. ISBN 1–58113–556–4. [43](#), [51](#)

SPANOUDAKIS, G.; ZISMAN, A.; NANA, E. P.-M.; KRAUSE, P. Rule-based generation of requirements traceability relations. **Journal of Systems and Software**, n. 72, p. 105–127, 2004. [39](#), [44](#), [46](#), [47](#)

- TANG, A.; YAN, J.; JUN, H. A rationale–base architecture model for design traceability and reasoning. **Journal of Systems and Software**, v. 80, n. 6, p. 918 – 934, June 2007. [39](#), [109](#)
- TANG, M. X. A knowledge–based architecture for intelligent design support. **The Knowledge Engineering Review**, v. 12, n. 4, p. 387–406, 1997. [42](#)
- TORANZO, M.; CASTRO, J.; MELLO, E. Uma proposta para melhorar o rastreamento de requisitos. In: WORKSHOP DE ENGENHARIA DE REQUISITOS, 2002, Valência, Espanha. **Anais...** [S.l.]: [S.n.], 2002. p. 194–209. ISBN 84–96023–01–X. [19](#), [29](#), [44](#), [45](#), [47](#), [51](#), [55](#), [61](#)
- TRYGGESETH, E.; NYTRØ, O. Dynamic traceability links supported by a system architecture description. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, 1997, Bari, Italy. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 1997. p. 180–187. [42](#)
- U.S. DEPARTMENT OF DEFENSE. **Military standard 2167A**: defense system software development. Washington, D.C., USA, 1988. [28](#)
- \_\_\_\_\_. **Military standard MIL–STD–498**: software development and documentation. Washington, D.C., USA, Dec. 1994. [28](#)
- \_\_\_\_\_. **National defense budget estimates for fy 2008**: office of the ubder secretary of defense (coomptroller). USA, Mar. 2007. 226 p. Disponível em: <http://www.defenselink.mil/comptroller/defbudget/fy2008/>>. Acesso em: 04/11/2008. [28](#)
- Van RIJSBERGEN, C. J. **Information retrieval**. London: University of Glasgow, 1979. 2nd edition. [43](#)
- WESKE, M. **Business process management**: concepts, languages, architectures. New York, NY, USA: Springer, 2007. 368 p. 1. ed. ISBN 978–3–540–73521–2. [118](#)
- WEST, M. Quality function deployment in software development. In: COLLOQUIUM BY THE INSTITUTION OF ELECTRICAL ENGINEERS PROFESSIONAL GROUP C1 (SOFTWARE ENGINEERING) – TOOLS AND TECHNIQUES FOR MAINTAINING TRACEABILITY DURING DESIGN, 1., 1991, London. **Proceedings...** London: IEE – The Institution of Electrical Engineers, 1991. p. 5/1–5/7. [42](#)

WOHLIN, C.; RUNESON, P.; HÖST, M.; REGNELL, B.; WESSLÉN, A.  
**Experimentation in software engineering**: an introduction. USA: Kluwer  
Academic Publishers, 2000. 1. ed. [97](#)

ZAVE, P. Classification of research efforts in requirements engineering. **ACM  
Computing Surveys Transactions on Software Engineering and  
Methodology**, v. 29, n. 4, p. 315 – 321, 1997. ISSN 0360–0300. [72](#)

ZELKOWITZ, M.; WALLACE, D. Experimental models for validating technology.  
**IEEE Computer**, v. 5, n. 31, p. 23 – 31, 1998. [91](#)

ZISMAN, A.; SPANOUDAKIS, G.; PÉRES-MIÑANA, E.; KRAUSE, P. Tracing  
software engineering artifacts. In: INTERNATIONAL CONFERENCE ON  
SOFTWARE ENGINEERING RESEARCH AND PRACTICE, 3., 2003, Las  
Vegas, Nevada, USA. **Proceedings...** [S.l.]: CSREA Press, 2003. p. 448–455. [43](#)



## **A APÊNDICE A – Questionários utilizados para execução do experimento**

### **A.1 Questionário sobre o perfil dos participantes do experimento**

- 1) Qual a sua formação profissional (graduação)?
- 2) Em que ano você concluiu a graduação?
- 3) Qual a sua idade?
- 4) Qual a sua área de especialização?
- 5) Há quanto tempo você atua na área de Informática?
- 6) Há quanto tempo você trabalha com desenvolvimento de sistemas?
- 7) Há quanto tempo você conhece metodologias sobre qualidade de software?
- 8) Há quanto tempo você conhece metodologias para engenharia de requisitos (levantamento, análise, gerenciamento, validação, matrizes de rastreabilidade, etc...)?
- 9) Há quanto tempo você tem conhecimento específico sobre matrizes de rastreabilidade?
- 10) Em quantos projetos envolvendo até 20 pessoas você trabalhou?
- 11) Em quantos projetos envolvendo de 20 até 50 você trabalhou?
- 12) Em quantos projetos com mais de 50 pessoas você trabalhou?
- 13) Em quantos projetos usando técnicas de engenharia de requisitos você trabalhou?
- 14) Em quantos projetos usando matrizes de rastreabilidade você trabalhou?

### **A.2 Questionário sobre as práticas de engenharia de requisitos e de engenharia de software adotadas pelo grupo**

- 1) A organização em que você trabalha possui práticas estabelecidas para engenharia de software?

- 2) O grupo em que você trabalha possui práticas estabelecidas para engenharia de software?
- 3) A organização em que você trabalha apóia a adoção de práticas para qualidade de software?
- 4) Seu grupo faz uso de alguma norma para Engenharia de Software?  Sim  
 Não  ISO/IEC-12207  ISO/IEC-15504  ISO-9000  CMMI  
 MPSBR  ECSS  Outras
- 5) Em seu grupo é obrigatório o uso de práticas para engenharia de software?  
 Sim  Não  Somente em alguns casos
- 6) Sobre as práticas para engenharia de software a organização ou grupo em que você trabalha **faz uso de práticas definidas para:**
- a) Engenharia de requisitos?  Sim  Não  
 Elicitação  Análise  Validação  Documentação  Gerenciamento  
 Outros
- b) Projeto de Software?  Sim  Não  
 Projeto Arquitetural  Projeto Detalhado  Outros
- c) Desenvolvimento de Software?  Sim  Não  
 Minimização de Complexidade  Antecipação de Mudanças  
 Padronização no desenvolvimento  Mensuração  Outros
- d) Teste de Software?  Sim  Não  
 Técnicas de teste  Níveis de teste (unidade, integração, sistema)  
 Mensuração
- e) Manutenção de Software?  Sim  Não  
 Classificação e identificação  Análise de Manutenção  Testes  
 Estimativa de custo  Análise de Impacto  Outros
- f) Configuração e Versão de Software?  Sim  Não  
 Gestão de configuração  Identificação de itens de configuração  
 Controle de acesso e de mudanças  Registro da situação, armazenamento e manuseio de forma controlada de todos os itens.

- g) Gerencia de Projeto? Sim Não  
Definição de escopo Planejamento de projeto Mensuração  
Estimativa Controle de esforço/tempo Outros
- h) Qualidade de Software? Sim Não  
Garantia da qualidade Verificação Validação Revisão conjunta  
Auditoria Resolução de problemas Outros
- 7) Cite quais as práticas de engenharia de requisitos adotadas pela sua organização ou grupo que você considera ideais?
- 8) Quais práticas de engenharia de requisitos adotadas pela organização que você considerada desnecessárias?
- 9) Você utiliza alguma métrica sobre rastreabilidade? Quais?
- 10) E sobre a qualidade dos elos(links) é utilizado algum padrão ou métrica para avaliar a qualidade desses elementos?
- 11) Quais práticas de engenharia de requisitos que não são adotadas que você considera ideais?

### **A.3 Questionário e itens para avaliação de matriz de rastreabilidade**

- 1) Determine o tempo empregado para construir cada um dos elos existentes na matriz e a razão ou porque da existência de cada elo.

Origem Destino	Tempo	Qual o grau de confiança da sua resposta?	Qual a Razão ou Porque da existência do elo?	Qual o grau de confiança da sua resposta?
REQ_040 ↔ REQ_033		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa
REQ_040 ↔ REQ_025		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa
REQ_040 ↔ REQ_039		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa
REQ_040 ↔ REQ_107		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa
REQ_040 ↔ REQ_108		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa
:	:	:	:	:
REQ_n ↔ REQ_n		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa		<input type="checkbox"/> Nenhum <input type="checkbox"/> Pouco <input type="checkbox"/> Mais ou menos <input type="checkbox"/> Muita <input type="checkbox"/> Completa

- 2) Determine qual o esforço total gasto nos elos inseridos na matriz? \_\_\_\_\_

Qual o grau de confiança da sua resposta?  Nenhum  Pouco  Mais ou menos  Muita  Completa

- 3) Você encontrou problemas para determinar o esforço e as razões/porquês dos elos na matriz apresentada? Caso a sua resposta seja sim cite quais foram esses problemas.

- 4) A maioria dos problemas estava relacionado à quais fatores?

## A.4 Documentação auxiliar para o líder responder o questionário 3

1) Exemplo de matriz de rastreabilidade utilizada para o experimento

Tabela A.1 - Exemplo de matriz.

Destino \ Origem	GSEG1	GSEG2	GSEG3	GSEG4	GSEG5	GSEG6	GSEG8
SYS1	x	x					
SYS4:7.4.1		x					
SYS5:6		x			x		
SYS...							

2) Exemplo da descrição dos elementos que compõem a matriz.

SYS1 - The maximum total storage capacity on-board (about 800 Mbit -TBC- for PLTM) imply the need of downloading PLTM at each pass. Data for a maximum of three (3) successive orbits can be stored, in case of anomaly	GSEG1 - Satellite control center requirements
	GSEG2 - Mission center requirements
SYS4: 7.4.1 - Prepare and monitor the execution of Payload Operation Plan (FI).	GSEG2 - Mission center requirements
SYS5.6 - Data retrieval from Satellite Control Center shall be accomplished by the Mission Center, taking into account the time limited archiving capability of the SCC	GSEG2 - Mission center requirements
	GSEG6 - Communication network requirements
⋮	⋮

3) Exemplo da descrição detalhada dos elementos que compõem a Matriz

**SYS1:** The maximum total storage capacity on-board (about 800 Mbit -TBC- for PLTM) imply the need of downloading PLTM at each pass.

Data for a maximum of three (3) successive orbits can be stored, in case of anomaly.

**SYS4: 7.4.1** Prepare and monitor the execution of Payload Operation Plan (FI).

**SYS5.6:** Data retrieval from Satellite Control Center shall be accomplished by the Mission Center, taking into account the time limited archiving capability of the SCC (specified in item 8.3.2.3).

**SYS8.3: FV/03:** Display, at Control Center operator command, the required parameters (synoptic parameter displays, parameter curves, replay of HKTM, on-board log-book,...) at either a local or a remote terminal (which can be in France - not in real time - for FBM functional chains monitoring on a DRPPC station).

## A ANEXO A - ARTIGOS E OUTROS TRABALHOS CIENTÍFICOS PUBLICADOS DURANTE O DOUTORADO

Tabela A.1 - Artigos publicados

---

<b>Trabalhos já publicados</b>
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Adaptable Documentation and Traceability of Software Requirements. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 6. (WORCAP), 2006, São José dos Campos. Proceedings... São José dos Campos: INPE, 2006. On-line. Disponível em: < <a href="http://urlib.net/dpi.inpe.br/hermes2@1905/2006/10.20.14.48">http://urlib.net/dpi.inpe.br/hermes2@1905/2006/10.20.14.48</a> >. Acesso em: 18 dez. 2008.
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Uma Proposta de Modelo Para a Rastreabilidade de Requisitos. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 7. (WORCAP), 2007, São José dos Campos. Proceedings... São José dos Campos: INPE, 2007.
GENVIGIR, E. C.; SANT'ANNA, N. Comparação do impacto do uso de um processo de engenharia de requisitos entre grupos de desenvolvimento de software: Um estudo de caso. In: WORKSHOP IBEROAMERICANO DE INGENIERÍA DE REQUISITOS Y AMBIENTES DE SOFTWARE, X., 2007, Isla Margaita, Venezuela. Memorias... Caracas, Venezuela: IDEAS'07, 2007. p. 121–134. ISBN–978.980.323.3.70.
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Uma Análise Sobre as Práticas de Engenharia de Software Adotadas no Desenvolvimento de Software em Aplicações Espaciais no INPE. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 8. (WORCAP), 2008, São José dos Campos. Proceedings... São José dos Campos: INPE, 2008.
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Uma Proposta de Modelagem para a Generalização de Elos de Rastreabilidade. Revista de Informática Teórica e Aplicada. XV (2), p. 181–202, 2008. ISSN–01034308.
<b>Capítulo de livro aceito e em processo de publicação</b>
GENVIGIR, E. C.; VIJAYKUMAR, N. L. Handbook of Software Engineering Research and Productivity Technologies: Implications of Globalisation, IGI Global. Chapter title: Requirements Traceability. Editor(s): Ramachandran, M. and Carvalho. R. A. Data prevista para publicação: jul. 2009.

---

## Comparação do Impacto do Uso de Um Processo de Engenharia de Requisitos Entre Grupos de Desenvolvimento de Software – Um Estudo de Caso

Elias Canhadas Genvigir<sup>1,2</sup>, Nilson Sant'Anna<sup>1</sup>

<sup>1</sup>Universidade Tecnológica Federal do Paraná (UTFPR)  
Av. Alberto Carazzai, 1.640 – 86300-000 – Cornélio Procópio – PR – Brasil

<sup>2</sup>Instituto Nacional de Pesquisas Espaciais (INPE)  
Av. dos Astronautas, 1.178 – 12227-010 – São José dos Campos – SP – Brasil  
{elias, nilson}@lac.inpe.br

**Resumo.** O uso de processos no desenvolvimento de um projeto de software constitui uma das fontes de esforços na pesquisa em Engenharia de Software. Experimentos são realizados usando dados de diferentes projetos, mas poucos trabalhos usam a comparação entre grupos que desenvolvem um mesmo projeto. Este trabalho faz uso da execução de um experimento, realizado sobre um mesmo projeto, para avaliar os requisitos de grupos que utilizam ou não um determinado processo para Engenharia de Requisitos. O quanto um processo de Engenharia de Requisitos impacta nos requisitos levantados é o foco principal deste trabalho. Análises estatísticas foram realizadas e seus resultados foram usados para explorar três perguntas sobre os requisitos além de reportar os resultados mais evidentes sobre o efeito do uso de um processo sobre a qualidade dos requisitos levantados.

**Palavras Chaves:** Engenharia de Requisitos, Processos de Software, Experimentação de Software.

### 1 Introdução

É de vital importância para o sucesso do projeto de software a gerência de seus requisitos. Este sucesso, por sua vez, está baseado na expectativa do usuário e na forma pela qual suas necessidades serão atendidas por esse software.

A engenharia de requisitos concentra-se nas necessidades e expectativas sobre o software sendo rica em complexidade e várias são as técnicas que podem ser utilizadas para o gerenciamento de requisitos. Entre essas complexidades pode-se citar o grande volume de requisitos levantados nos projetos modernos de software, a forma de representar um processo e a maneira na qual os requisitos podem ser modelados.

É papel da engenharia de requisitos criar mecanismos para manter o equilíbrio das necessidades dos usuários em relação ao que o software deve realizar, ou seja, ela deve assegurar que o produto de software atenda seus objetivos, tanto do ponto de vista do usuário como do ponto de vista técnico e do negócio em questão. Para tanto

uma metodologia de trabalho deve ser adotada para a especificação de requisitos do software, mas qual o impacto que um processo modelado pode causar em grupos de desenvolvimento que aplicam pela primeira vez um determinado processo?

O artigo apresenta os resultados obtidos na execução de um experimento sobre um processo de engenharia de requisitos. Para tanto foi utilizado um estudo de caso para observar questões relacionadas ao uso ou não de um processo modelado e qual o grau de impacto na qualidade dos requisitos levantados durante o experimento. Um processo modelado é compreendido, neste estudo, como a representação de atividades do mundo real para um processo de produção de software sendo desenvolvido e modelado por meio de um meta-processo e representado por uma linguagem de modelagem de processos [1].

O experimento foi realizado através do uso de grupos de trabalho, os quais tiveram que levantar os requisitos de um dado sistema fazendo uso ou não do processo modelado de engenharia de requisitos.

## 2 Questões Pesquisadas

O estudo reporta as contribuições mais evidentes realizadas no experimento, focando três perguntas que orientaram o trabalho:

1. É significativa a diferença na qualidade dos requisitos levantados entre equipes de desenvolvimento que usam um processo modelado de engenharia de requisitos e equipes que não usam um processo modelado?
2. O uso pela primeira vez de um processo modelado de requisitos por uma equipe de desenvolvimento impacta significativamente a qualidade dos requisitos?
3. A representação de requisitos através de casos de uso facilita a elicitação dos requisitos de um sistema?

A necessidade de obter informações sobre a execução e a qualidade de artefatos gerados por um processo modelado de requisitos orientou a investigação sobre o quanto esse processo pode inferir positivamente ou não no levantamento dos requisitos de um sistema. Mas a avaliação de qual deveria ser o processo de requisitos a ser utilizado deixava lacunas a respeito do conhecimento prévio dos membros dos grupos sobre processos já existentes o que poderia interferir nos resultados. Para solucionar este problema um novo processo de requisitos baseado nos principais processos existentes e modelado em uma linguagem de definição de processos foi definido para o experimento.

## 3 Modelo Para o Processo de Engenharia de Requisitos

A definição do processo levou a pesquisa sobre os trabalhos desenvolvidos na área Davis [2], Jarke e Pohl [3], Sommerville [4], Macaulay[5], Jalote [6], Sommerville e Sawyer [7], Graham [8], Pohl [9], Kontonya e Sommerville [10]. Foi observado que os autores especificam de forma abrangente o processo de engenharia de requisitos e que é grande a preocupação com sua divisão em grandes atividades e na concentração

de esforços pela determinação de seus campos de conhecimento. Por outro lado as normas e os modelos investigados: ISO 15504 [11], ISO/IEC 12207 - Software Life Cycle Processes [12], SW-CMM- The Capability Maturity Model for Software [13], RUP - Rational Unified Process [14], SWEBOK – Guide to the Software Engineering Body of Knowledge [15], MPS.BR - Modelo de Melhoria do Processo de Software Brasileiro [16]; destacam com maior ênfase as tarefas, ou seja, como e o que deve ser feito pelo processo. De modo geral, tanto os modelos e normas quanto os autores não especificam de forma totalmente ampla as atividades que compõem o processo e o campo de conhecimento as quais elas compreendem.

O modelo proposto neste trabalho levou em consideração as observações dos autores e as melhores práticas descritas pelas normas e modelos, abordando de maneira abrangente e ao mesmo tempo precisa o processo de engenharia de requisitos.

### 3.1 Modelando o Processo Através do Software Process Engineering Metamodel Specification

Após a pesquisa sobre Engenharia de Requisitos, foi necessário modelar o processo para uso durante o experimento. Para tanto foi utilizado, para representar os elementos do processo, a linguagem de modelagem SPEM - Software Process Engineering Metamodel Specification [17]. O SPEM surgiu da necessidade de unificação entre várias linguagens de modelagem de processos, sendo sua publicação inicial chamada de UPM - Unified Process Model [18].

A modelagem consiste em utilizar diagramas da UML (Pacote, Caso de Uso, Classe e de Atividades) associados aos estereótipos propostos pelo SPEM para representar tanto os elementos estáticos quanto os dinâmicos de um processo.

O processo proposto para a Engenharia de Requisitos é composto por quatro elementos do SPEM chamados Disciplines: Elicitação, Análise e Negociação, Validação e Documentação, que são executados de forma iterativa, alocados dentro de um diagrama de pacotes representando o nível mais alto do processo (Fig. 1).

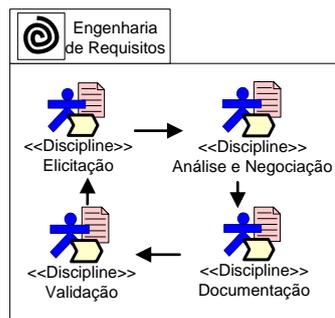
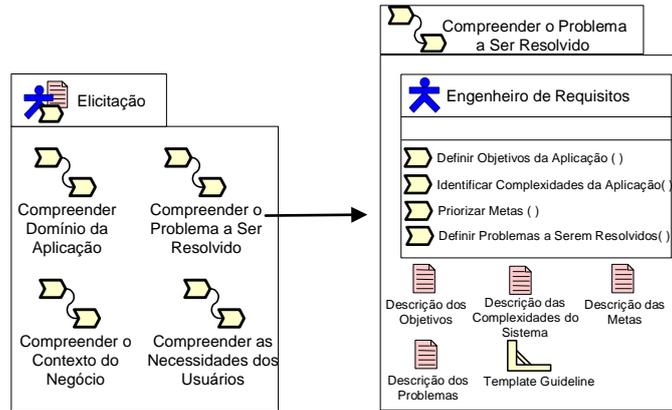


Fig. 1. Diagrama de Pacote do Processo de Engenharia de Requisitos

Cada uma das Disciplines é refinada por meio de diagrama de pacotes que permitem visualizar, em um nível alto de abstração, as principais atividades que compõem o processo.

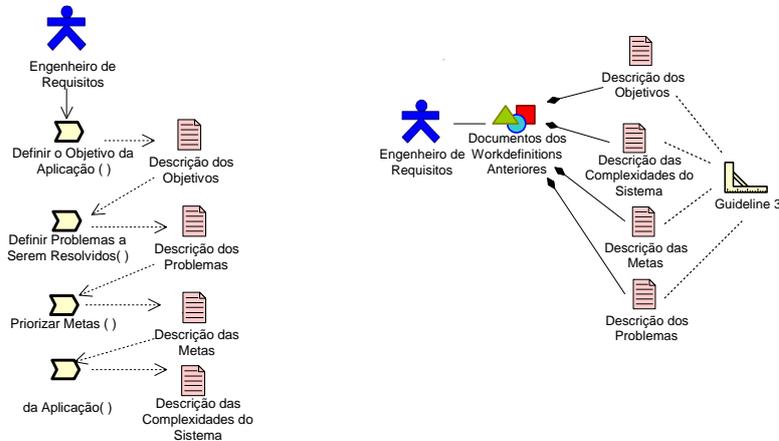
Os diagramas de pacotes são utilizados durante toda a modelagem do processo para apresentar o conjunto de elementos que compõem uma determinada Discipline,

como apresentado na Figura 2, esta figura apresenta o refinamento através de diagramas de pacotes da Disciplina Elicitação que por sua vez é composta de quatro WorkDefinition sendo que um deles é decomposto em outro diagrama de pacotes o qual contém os elementos em nível atômico.



**Fig. 2.** Diagrama de Pacotes da Disciplina Elicitação e a decomposição em outro diagrama de Pacotes do elemento WorkDefinition Compreender o Problema a ser Resolvido.

A seqüência da execução das Disciplinas do Processo é demonstrada através do uso de um diagrama de atividades que permitem a visualização dos WorkProduct gerados pelo WorkDefinition ou por uma Disciplina e quais os participantes do processo que são responsáveis por cada um dos artefatos gerados (Fig. 3).



**Fig. 3.** Diagrama de Atividades e de Classes do WorkDefinition Compreender o Problema a ser Resolvido.

Ao todo o processo modelado é composto por 31 atividades (Tabela 1), representadas no processo através dos diagramas, em diferentes níveis de abstração. A documentação do processo consiste, além dos diagramas, de uma descrição textual detalhada dos objetivos, das tarefas e dos artefatos consumidos e gerados por cada uma das atividades.

**Tabela 1.** Disciplines e Atividades que compõem o processo modelado de engenharia de Requisitos utilizado no experimento.

<b>Discipline</b>	<b>Atividades de cada Discipline</b>
<b>Elicitação</b>	
<b>WorkDefinition:</b> <b>Compreender o domínio da aplicação</b>	<ul style="list-style-type: none"> <li>• Identificar o domínio do negócio</li> <li>• Identificar características do negócio</li> <li>• Identificar as atividades do negócio</li> </ul>
<b>WorkDefinition:</b> <b>Compreender o contexto do negócio</b>	<ul style="list-style-type: none"> <li>• Identificar a estrutura organizacional</li> <li>• Identificar sistemas existentes</li> </ul>
<b>WorkDefinition:</b> <b>Compreender o problema a ser resolvido</b>	<ul style="list-style-type: none"> <li>• Definir objetivos da aplicação</li> <li>• Identificar complexidades da aplicação</li> <li>• Definir problemas a serem resolvidos</li> <li>• Priorizar metas</li> </ul>
<b>WorkDefinition:</b> <b>Compreender as necessidades dos usuários</b>	<ul style="list-style-type: none"> <li>• Definir funcionalidades do sistema</li> <li>• Identificar atores e funções</li> <li>• Definir casos de uso</li> <li>• Definir o diagrama de caso de uso de contexto</li> <li>• Descrever casos de uso</li> <li>• Definir os protótipos de interface</li> </ul>
<b>Análise e Negociação</b>	<ul style="list-style-type: none"> <li>• Definir e executar checklist</li> <li>• Estabelecer rastreabilidade</li> <li>• Executar matriz de interação</li> <li>• Documentar problemas</li> <li>• Negociar Requisitos</li> </ul>
<b>Documentação</b>	<ul style="list-style-type: none"> <li>• Elaborar dicionário de termos</li> <li>• Definir padrões para a documentação</li> <li>• Gerar especificação de requisitos de software</li> </ul>
<b>Validação</b>	<ul style="list-style-type: none"> <li>• Definir ações</li> <li>• Definir plano de revisão</li> <li>• Distribuir documentação</li> <li>• Corrigir especificação de requisitos</li> <li>• Listar problemas</li> <li>• Validar artefatos versus casos de uso</li> <li>• Validar Interfaces versus casos de uso</li> <li>• Executar checklists</li> </ul>

As quatro Disciplines do processo, listadas na Tabela 1, cobrem tanto as normas quanto os modelos propostos pelos autores analisados, além de estabelecer a rastreabilidade através da execução de um conjunto de atividades definidas internamente nas Disciplines e na execução iterativa do processo.

#### 4 Métodos de Pesquisa

O experimento realizado orienta-se sobre o modelo conceitual e o modelo GQM - *Goal Question Metric* propostos por Basili [19][20]. Esses modelos dividem o trabalho de experimentação em fases (definição, planejamento, operação e interpretação) e são orientados a metas. Neste trabalho, essas fases são apresentadas como: escopo do trabalho; definição do estudo métodos e práticas; análise e interpretação.

#### 4.1 Escopo do Trabalho

O estudo de caso foi desenvolvido no ano de 2005 com alunos de pós-graduação da área de Engenharia de Software do curso de Pós-graduação em Computação Aplicada do Instituto Nacional de Pesquisas Espaciais – INPE e teve duração, de experimentação, de três meses. Seu principal propósito foi verificar as vantagens e desvantagens do uso de um processo modelado em relação à outra ou nenhuma técnica específica de Engenharia de Requisitos, avaliando a qualidade dos requisitos levantados entre os grupos de trabalho.

#### 4.2 Definição do Estudo Métodos e Práticas

O experimento consistiu em formar oito grupos, contendo cada um cinco indivíduos, entre eles um gerente escolhido pelos membros do grupo, que simularam oito empresas distintas.

Todos os grupos, representando empresas, competiram entre si para apresentar, em um mesmo prazo estipulado, um sistema de informação que automatizasse uma fábrica de artefatos metálicos. Com a intenção de estimular a competição entre os grupos, a empresa que apresentasse um produto de software adequado às necessidades estabelecidas pelo cliente teria um contrato de prestação de serviços estabelecido. É interessante ressaltar que o cliente utilizado no experimento consistiu de uma empresa real de produção de artefatos metálicos, na qual um indivíduo externo aos procedimentos metodológicos do experimento (o diretor da empresa) se dispôs a atender, em tempo integral, todos os grupos que participaram do experimento, assim que estes necessitassem de alguma visita, reunião ou informações para o levantamento de requisitos junto ao cliente.

Foram criadas duas categorias de grupos, uma que trabalhou com o processo modelado e outra que não teve acesso ao modelo do processo, ou seja, dos oito grupos formados quatro fizeram uso do processo modelado de Engenharia de Requisitos e quatro trabalharam sem o processo modelado. Tanto a composição dos membros que fariam parte de cada grupo quanto a escolha dos grupos que trabalhariam com e sem o processo foi feita aleatoriamente. Os grupos foram divididos em pares e ímpares, também de forma aleatória, assim como a definição de qual deles utilizaria o processo modelado.

Para todos os grupos foi fornecida uma pequena descrição textual do caso de estudo, com o objetivo de permitir uma visão geral do sistema a ser desenvolvido e os artefatos que deveriam ser entregues após a conclusão do trabalho. Para este experimento deveria ser entregue, 90 dias após a apresentação dos objetivos, os requisitos modelados por meio de casos de uso e um sistema de informação que atendesse a descrição fornecida.

Detalhes sobre a aplicação a ser desenvolvida foram omitidos. Isso foi feito para que os grupos necessitassem elicitar junto ao cliente os requisitos do sistema a ser desenvolvido evitando-se, assim, que a atividade de elicitação fosse descaracterizada, mal utilizada ou diminuísse o grau de realismo do experimento.

Para todos os grupos foram apresentados de forma expositiva, os objetivos que deveriam ser atingidos: *"Um sistema computacional deve ser desenvolvido, em 90 dias, com os objetivos de: melhorar os trabalhos dos setores de venda e de compra, melhorar a monitoração das diversas unidades de produção, e manter o diretor informado sobre todos os setores de uma fábrica de artefatos metálicos"*.

Como atividades específicas os grupos deveriam: a) Desenvolver o levantamento dos requisitos modelando-os através de casos de uso; b) Desenvolver o sistema baseado nos requisitos levantados.

Os grupos sem o processo modelado puderam utilizar-se de qualquer outro modelo, metodologia, ciclo de vida ou processo para desenvolver o sistema, se assim o desejassem. Os grupos com processo utilizaram, especificamente, o processo para a engenharia de requisitos, fornecido no momento da apresentação dos objetivos que consistia de um manual contendo a documentação do processo e sua descrição.

Para estabelecer conclusões sobre o experimento três categorias de informações foram coletadas: 1) Informações sobre os grupos e seus membros; 2) Informações sobre a execução do processo, e 3) Informações sobre os requisitos: número de requisitos levantados; incompletos; sobrepostos; redundantes; negociados; e corretos.

Na primeira categoria as informações foram coletadas através da aplicação de um questionário contendo quarenta perguntas dirigidas aos membros e aos gerentes dos grupos. Foram apresentadas questões quanto à experiência dos indivíduos, sobre o esforço despendido e sobre as experiências em relação ao trabalho executado. A segunda categoria também fez uso de questionários. Por fim, a terceira categoria utilizou um avaliador externo ao experimento que fez uso da aplicação das métricas estabelecidas no experimento para avaliar os requisitos.

Sobre métricas que poderiam ser utilizadas para medir um requisito, considerou-se o grau de qualidade, visto que a qualidade dos requisitos está diretamente relacionada à qualidade do produto de software. Tornou-se então necessário estabelecer o que era um requisito com qualidade.

A norma IEEE-830 [21] define que requisitos de alta qualidade são claros, completos, sem ambigüidade, implementáveis, consistentes e testáveis. Ainda, segundo a norma, os requisitos que não apresentem estas qualidades são problemáticos e devem ser revistos e renegociados com os clientes e usuários.

Por falta de padrões para avaliar especificamente casos de uso, foi utilizada a norma IEEE-830, com algumas modificações, para avaliar os casos de uso e suas descrições, como os cenários. Foi estabelecido então, que um caso de uso deveria ser:

- Correto: um caso de uso é correto somente se pode ser identificado no software;
- Não ambíguo: um caso de uso é não ambíguo se tem somente uma interpretação;
- Completo: um caso de uso é completo se: 1) Sua descrição é significativa e está relacionada à funcionalidade, desempenho, atributos, restrições de projeto e interfaces externas; 2) Sua descrição define as respostas do software para todas as classes quanto aos dados de entradas, em todas as classes de situações realizáveis (cenários); 3) Todos os termos e referências a atores, a figuras, tabelas e definições estão relacionados; 4) Possui grau de importância e risco: um caso de uso é estável se possui um identificador para indicar a importância do requisito e o grau de risco;
- Consistente: um caso de uso é consistente se não está relacionado a nenhum conflito, ou seja, se é compatível com os objetivos do sistema e se não possui problemas de acordo técnico entre os envolvidos.
- Verificável: um caso de uso é verificável somente se sua declaração e descrição são verificáveis. Um caso de uso é verificável somente se, existir algum processo de custo finito, no qual uma pessoa ou máquina possa verificar se este está sendo atendido pelo produto do software. Em geral, um caso de uso ambíguo é não verificável;

- Modificável: um caso de uso é modificável somente se a sua estrutura e estilo permitem que qualquer mudança possa ser feita facilmente, de forma completa e consistente, mantendo a estrutura e o estilo;
- Rastreável: um caso de uso é rastreável se a origem de cada um de seus requisitos está clara e se este é facilmente referenciado e rastreado tanto nos seus desdobramentos (cenários) quanto em relação a sua origem.

Após determinar as métricas foi avaliada a técnica para aplicá-las nos casos de uso.

A solução inicial para aplicar as métricas estabelecidas dar-se-ia pelo uso de um modelo de casos de uso do problema, definido previamente por especialistas independentes, contra o modelo elicitado e elaborado por cada grupo. Mas essa técnica traz implicações sobre o alto grau de abstração existente na definição desse diagrama UML, ou seja, um mesmo problema pode ser modelado de diferentes formas e encontrar uma mesma solução. Devido a esse problema optou-se por uma avaliação sem interferências ou comparações com um modelo de casos de uso pré-definido. Para tanto, um avaliador externo, com experiência comprovada em engenharia de requisitos e técnicas orientadas a objetos, teve acesso às especificações dos requisitos de software e também aos produtos de software desenvolvidos por cada um dos oito grupos. Este avaliador levantou os requisitos, grupo a grupo, e aplicou as métricas definidas para avaliação dos casos de uso, utilizando uma lista de checagem que continha os sete critérios baseados na norma IEEE-830, citados anteriormente.

Ao avaliar o conjunto de casos de uso de um dado grupo o avaliador verificava, por exemplo, se um caso de uso era correto conferindo se o mesmo poderia ser identificado no produto de software. Como o avaliador possuía acesso tanto às especificações de requisitos quanto ao produto de software ele verificava a informação e a reportava para a lista de checagem dos casos de uso do grupo que estava sendo avaliado (Tabela 2). Caso o item avaliado existisse e estivesse completo, o item recebia o valor 1, caso contrário, possuísse problemas, estivesse errado ou parcialmente preenchido, recebia o valor 0.

**Tabela 2.** Disciplines e Atividades que compõem o processo modelado de engenharia de Requisitos utilizado no experimento.

Itens	Casos de Uso						
	a	b	c	....	....	y	z
Correto	0	1	1	0	0	0	0
Não ambíguo	1	1	0	1	1	1	1
Completo	1	0	1	0	1	0	0
Grau de Importância	1	1	1	1	1	1	1
Consistente	1	0	1	0	1	0	0
Verificável	1	0	1	0	1	0	0
Modificável	1	0	1	0	1	0	0
Rastreável	1	0	1	0	1	0	0

A verificação estatística das questões definidas no início do experimento foi feita pela definição das hipóteses e pela constatação ou não da veracidade destas. Para cada um dos itens avaliados foram definidas as seguintes hipóteses:

- 1)  $H_0$  (hipótese nula): Os requisitos levantados pelos grupos, com processo e sem uso do processo, possuem o mesmo grau de qualidade do item avaliado. Assim, considera-se a hipótese nula por  $H_0: \delta = 0$  (sem diferença significativa).
- 2)  $H_A$  (hipótese alternativa): Os requisitos levantados pelos grupos, com processo e sem uso do processo, não possuem o mesmo grau de qualidade do item

avaliado. Assim, considera-se a hipótese alternativa por  $H_A: \delta \neq 0$  (existência de diferença significativa).

### 4.3 Análise

Foram determinados os valores das variáveis necessárias para a resposta à hipótese  $H_0$  e à hipótese  $H_A$ . Os valores, tanto os com processo como os que não usaram o processo, foram comparados entre os dois tipos de grupos.

A Tabela 3 apresenta o número de acertos obtidos após a análise dos casos de uso dos grupos e a Tabela 4 apresenta o número de erros.

**Tabela 3.** Número de acertos obtidos nos casos de uso levantados pelos grupos.

Grupos usando Processo	Número de Casos de Uso	Corretos	Não ambíguos	Completo	Grau de Importância	Consistente	Verificável	Modificável	Rastreável
1	16	15	11	15	16	10	12	16	16
3	14	14	14	14	14	10	14	14	14
5	10	10	10	10	10	9	10	10	10
7	17	17	10	17	17	10	9	17	17
Sem uso Processo									
2	14	10	4	0	0	4	0	0	0
4	19	8	8	0	0	8	0	0	0
6	27	9	7	0	0	7	0	0	0
8	103	10	10	0	0	9	0	0	0

**Tabela 4.** Número de erros obtidos dos casos de uso levantados pelos grupos.

Grupos usando Processo	Número de Casos de Uso	Não Corretos	Ambíguos	Incompletos	Sem Grau de Importância	Inconsistente	Não Verificável	Não Modificável	Não Rastreável
1	16	1	5	1	0	6	4	0	0
3	14	0	0	0	0	4	0	0	0
5	10	0	0	0	0	1	0	0	0
7	17	0	7	0	0	7	8	0	0
Sem Uso Processo									
2	14	4	10	14	14	10	14	14	14
4	19	11	11	19	19	11	19	19	19
6	27	18	20	27	27	20	27	27	27
8	103	93	93	103	103	94	103	103	103

Para observar a diferença entre as médias de cada um dos itens dos dois tipos de grupos, utilizou-se o teste estatístico não-paramétrica de Mann-Whitney.

Desde que atingido um grau de mensuração pelo menos ordinal, pode-se aplicar a prova U de Mann-Whitney [22], para comprovar se dois grupos independentes foram ou não extraídos da mesma população. Segundo Siegel [22] trata-se de uma das mais poderosas provas não-paramétricas, e constitui uma alternativa extremamente útil para a prova paramétrica t ou teste t de Student, quando se deseja evitar as suposições exigidas por este último, ou quando a mensuração atingida é inferior a escala de intervalos. Com esse teste, pode-se testar se duas amostras independentes provêm de populações idênticas. Em particular podemos testar a hipótese nula  $\mu_1 = \mu_2$  sem

precisar supor que as populações originárias tenham a forma aproximada de distribuições normais.

A decisão do teste pode se basear em R1 ou R2 que são, respectivamente, a soma dos postos dos valores da primeira amostra e da segunda amostra, não importando qual seja a amostra 1 e qual seja a amostra 2. A decisão do teste de hipótese pode basear-se em R1, ou em R2 ou pela estatística U apresentada pelas fórmulas:

$$U = n1.n2 + \frac{n1(n1 + 1)}{2} - R1 \quad \text{e} \quad U = n1.n2 + \frac{n2(n2 + 1)}{2} - R2 \quad (1)$$

Segundo Mood [23] se aplicado a prova do teste U a dados que possam ser adequadamente analisados pela prova paramétrica t, seu poder-eficiência tende para  $3/\pi = 95,5$  por cento quando o número de elementos aumenta, e está próximo de 95 por cento para amostras de tamanho moderado. Trata-se, portanto, de uma excelente alternativa para a prova t, e que dispensa as suposições restritivas e as exigências inerentes à prova t.

Após a tabulação dos dados considerou-se a hipótese de teste ao nível de 95% de confiança (estatisticamente discernível no nível de 5%), observaram-se os resultados, extraídos da análise efetuada pelos softwares SPSS e Excel para o teste U, admitindo-se para todos os testes de hipótese que se seguiram que:  $H_0: \delta=0$  e  $H_A: \delta \neq 0$  com  $\alpha \leq 0,05$  para U e n1 e n2 dados.

Segundo os resultados obtidos, foi rejeitada a hipótese nula para todos os itens avaliados, ou seja, estatisticamente não é possível afirmar que os dois grupos possuem um número igual ou aproximado de requisitos com os mesmos padrões de acerto, observando-se que os grupos que fizeram uso do processo obtiveram em todos os itens diferença significativa positivamente sobre a qualidade dos requisitos levantados.

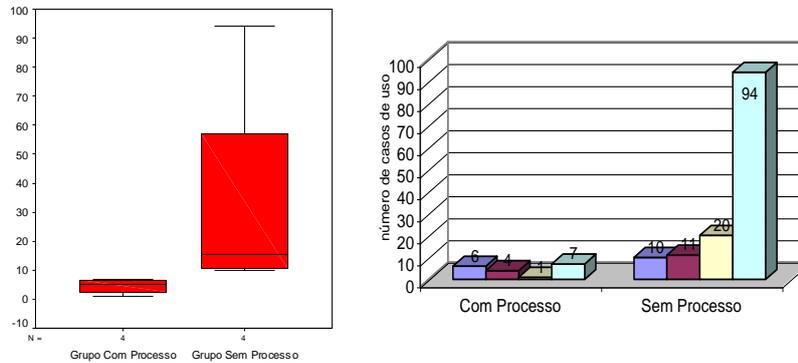
A Tabela 5 apresenta alguns valores estatísticos levantados sobre o número de erros observados sobre os itens avaliados nos casos de uso levantados pelos grupos.

**Tabela 5.** Resultados da comparação entre grupos.

Elementos Analisados	Tipo de grupo	N	Média	Desvio Padrão	Erro Médio
Número de Casos de Uso não corretos	1	4	0,25	0,500	0,250
	2	4	31,50	41,396	20,698
Número de Casos de Uso Ambíguos	1	4	3,00	3,559	1,780
	2	4	33,50	39,921	19,960
Número de Casos de Uso incompletos	1	4	0,25	,500	0,250
	2	4	40,75	41,844	20,922
Número de Casos de Uso sem grau de risco	1	4	0,00	,000	0,000
	2	4	40,75	41,844	20,922
Número de Casos de Uso Inconsistentes	1	4	4,50	2,646	1,323
	2	4	33,75	40,418	20,209
Número de Casos de Uso não Verificáveis	1	4	3,00	3,830	1,915
	2	4	40,75	41,844	20,922
Número de Casos de Uso não Modificáveis	1	4	0,00	,000	0,000
	2	4	40,75	41,844	20,922
Número de Casos de Uso não Rastreável	1	4	0,00	,000	0,000
	2	4	40,75	41,844	20,922

A tabela apresenta valores a respeito do número de casos de uso levantados pelos grupos e analisados. Onde: N = número de grupos, Tipo de grupo = caracteriza-se por 1 = grupos com processo; 2= grupos sem uso do processo

A Figura 4 apresenta dois gráficos, o primeiro um Box Plot e o segundo um gráfico de barras que demonstram os valores apresentados pelos grupos nos itens: número de casos de uso inconsistentes.



**Fig. 4.** Gráficos comparativos entre os grupos: número de casos de uso inconsistentes.

Ao observar os gráficos é possível notar que o agrupamento dos dados dos grupos que usaram o processo é mais conciso enquanto os grupos que não usaram o processo possuem grande discrepância, fator que foi considerado na análise estatística por apontar a relevância de quanto à falta de um processo para o desenvolvimento dos requisitos pode refletir em incoerências notadas no alto número de requisitos inconsistentes e dos erros apresentados nos demais itens avaliados. Ainda sobre a análise dos dados observa-se grande variação do grupo número oito (sem processo). Mesmo retirando este último grupo da análise não é possível aceitar  $H_0$  para nenhum dos itens avaliados, considerando  $p \leq 0,05$  e  $n1=3$  e  $n2=4$ .

Também foi realizada análise estatística sobre os dados do perfil dos membros dos grupos. Através de questionários três categorias de perguntas foram feitas:

- 1) Sobre experiência, por exemplo: Quanto tempo de experiência na área? Quanto tempo de experiência em desenvolvimento de sistemas? Quanto tempo de conhecimento em metodologias sobre qualidade? Quanto tempo de conhecimento em metodologias sobre engenharia de requisitos? Participação em quantos projetos de pequeno, médio e grande porte?
- 2) Dedicção e expectativas sobre experimento, por exemplo: Quantas horas de estudos adicionais? Quantas horas de estudo sobre o problema abordado no experimento? Quantas reuniões do grupo participou? Quais as categorias de problemas encontrados?
- 3) Comportamento do grupo, como exemplo: Quantas reuniões o grupo realizou? Tempo médio das reuniões do grupo? Participação do gerente do grupo? Modelo de comunicação usada pelo gerente do grupo?

O questionário totalizou 40 perguntas e os membros dos grupos gastaram em média 30 minutos para respondê-las.

Para análise dos dados coletados nos questionários foi utilizado o teste exato de Fisher e o teste  $\chi^2$  (Qui-quadrado), que permitem avaliar se a diferença nas amostras constituem evidências convincentes de associação entre as variáveis analisadas [22].

Todos os teste foram realizados considerando as hipóteses  $H_0$ : a variável analisada é independente entre as duas categorias de grupo e;  $H_A$ : a variável analisada não é independente existindo associação entre as duas categorias de grupos. Foi considerado nível de confiança de 95% e significância de 5%.

Nas duas primeiras categorias de perguntas aceitou-se a Hipótese  $H_0$  para todos os casos, o que permitiu considerar a inexistência de diferença, estatisticamente discernível, entre os itens avaliados sobre experiência, dedicação e expectativas dos membros que compunham as duas categorias independentes de grupos.

Para a terceira categoria de perguntas três itens rejeitaram a hipótese  $H_0$ : 1) o número de reuniões realizadas entre os membros dos grupos, grupos com processo tiveram em média nove reuniões, enquanto os grupos sem uso do processo tiveram três reuniões; 2) a frequência de participação nas reuniões, membros dos grupos com uso do processo participaram em média de 89% das reuniões enquanto membros dos grupos sem uso do processo participaram 60%; 3) o número de reuniões com o cliente, grupos usando o processo tiveram em média sete reuniões com o cliente enquanto grupos sem uso do processo tiveram duas reuniões. Para estes três casos foi aceita a hipótese alternativa  $H_A$  o que permitiu considerar a existência de associação para esses três itens entre as duas categorias de grupos analisados sendo que para os demais casos foi aceita a hipótese  $H_0$ , ou seja, não houve diferença entre os grupos.

## 5. Interpretação

No contexto estatístico os resultados da análise, para os dados coletados, foram obtidos ao nível de certeza de 95%. Adotando-se uma metodologia mais precisa sobre o ponto de vista estatístico, o experimento pode ser executado até que se possa extrair um número de amostra que seja significativa estatisticamente para que outros tipos de testes possam ser aplicados, o que poderia apresentar o comportamento populacional.

Com relação ao propósito do experimento, no qual foi aplicado o processo para engenharia de requisitos estabelecido, houve bons resultados. Pode ser verificado o sucesso da aplicação de um processo como também algumas vantagens nítidas na aplicação de um processo modelado contrapondo o uso do empirismo no desenvolvimento dos requisitos de um sistema, visto que nenhum dos grupos, que não utilizaram o processo, fizeram uso de técnicas específicas para a Engenharia de Requisitos em seus projetos.

Sobre os requisitos levantados pelos grupos destaca-se o grupo número oito - sem uso do processo. Este grupo apresentou um alto número de requisitos levantados, cento e três. Esse dado enfatiza que a falta de uma metodologia para conduzir o trabalho da engenharia de requisitos, em um grupo de desenvolvimento, pode interferir negativamente na qualidade dos requisitos, mesmo quando o grupo não difere em experiência, dedicação e comportamento dos demais grupos.

Para este experimento, com a amostra aqui apresentada, sobre os aspectos de aplicabilidade, pode-se verificar a validade e o aumento do grau de qualidade dos requisitos. Vários fatores também devem ser considerados, como a falta da parte experimental de um processo de gerência de projeto que poderia mostrar o tempo gasto na execução de cada atividade o que poderia sugerir outras análises.

Um item que deve ser observado é que a determinação de um processo já estabelece um padrão de qualidade entre os grupos que o utilizam, pois nos testes estatísticos aplicados não foi possível observar diferença entre a experiência dos membros dos dois tipos de grupos.

Para os requisitos que foram representados em forma de casos de uso não é possível afirmar que os mesmos melhoram a interpretação dos requisitos como afirmado por Maiden e Robertson [24], pois a análise é feita por grupos que utilizam a mesma forma de representação.

Uma das explicações sobre a qualidade dos requisitos pode ser dada pelo fato de que a as reuniões solicitadas pelos grupos que usaram o processo junto ao cliente para o levantamento dos requisitos foi 72% maior do que os grupos que não utilizam o processo. O que permite sugerir que o formalismo para o desenvolvimento das atividades do processo força o analista a envolver-se com o problema do cliente compreendendo de forma ampla o problema a ser resolvido.

Ainda sobre o experimento também foi positivo o grau de aceitação do uso do processo, visto que:

- 52% dos membros dos grupos com processo consideraram o processo de média facilidade de uso;
- 32% dos membros dos grupos com processo consideraram o processo de alta facilidade de uso;
- 16% dos membros dos grupos com processo consideraram o processo de uso altamente recomendado.

Sobre o grau de dificuldade associado à execução do processo no experimento, observou-se que:

- 11% dos membros dos grupos com processo consideraram como maior dificuldade a compreensão da linguagem utilizada para modelar o processo;
- 5% encontrou dificuldades em executar as atividades;
- 21% atribuiu as falhas à gerência do grupo;
- 47% à falta de conhecimento técnico para executar as atividades;
- 16% à falta de tempo para compreender e executar o modelo proposto.

Também é observado que a dificuldade na interpretação e na aplicação de um processo de requisitos não possui grande relevância comparada a outros problemas.

Ainda sobre os requisitos foi observada grande diferença entre os requisitos dos grupos que utilizaram o processo e os dos que não o utilizaram, havendo assim a tendência de validar em todos os casos a hipótese alternativa que supõe diferença entre as médias das duas categorias de grupos no item qualidade dos requisitos. Mas com relação a essa interpretação e ao fato de ser pequena a representatividade da amostra, motiva-se a continuação deste experimento em trabalhos futuros, o que possibilitaria estabelecer conclusões mais precisas, bem como reconsiderar o mecanismo de avaliação dos casos de uso, visto que o uso de avaliadores independentes pode inferir variabilidade nos resultados obtidos.

## 6. Conclusão

A execução de experimentos em Engenharia de Software, especificamente na Engenharia de Requisitos, compõem uma tarefa complexa, dada as características do software. O controle das diversas variáveis inerentes ao ambiente de desenvolvimento torna quase inviável esse tipo de experimento. Devido a esses problemas muitos trabalhos optam por utilizar dados de diferentes projetos, mas ao lidar com a Engenharia de Requisitos diferentes projetos apresentam diferentes e complexas variações sobre suas variáveis.

Este caso de estudo apresentou como um processo modelado para engenharia de requisitos pode ser aplicado em um caso prático e que o uso de processos apresentam uma alternativa real para a melhoria dos requisitos do software.

Estudos futuros podem fazer uso de outros caminhos para a execução de experimentos como o uso de várias formas para representar requisitos, o aumento da população analisada e o acréscimo de métricas gerenciais sobre os dados levantados.

Outro ponto que pode ser considerado é o refinamento dos mecanismos de avaliação de requisitos.

Este artigo apresentou que o uso de um processo modelado aplicado pela primeira vez por grupos de desenvolvimento apresenta resultados positivos sobre a qualidade dos requisitos elicitados.

## Referências

1. Derniame, J., Ali Kaba, B., Wastell, D. G.: *Software Process: Principles, Methodology, Technology*. Lecture Notes in Computer Science 1500 Springer (1999).
2. Davis, A. M. "Software requirements: objects, functions and states". 1 ed., Prentice-Hall Inc. (1993).
3. Jarke, M., Pohl, K. "Requirements engineering in 2001: managing a changing reality." *IEEE Software Engineering Journal*, v. 9, n. 6, p. 257-266, November, (1994).
4. Sommerville, I. "Software engineering". 5. ed. Harlow, Addison Wesley, (1995).
5. Macaulay, L. "Requirements engineering". 1. ed. Springer, (1996).
6. Jalote, P. "An integrate approach to software engineering". 2. ed., Springer, (1997).
7. Sommerville, I., Sawyer, P. "Requirements engineering: a good practice guide". Lancaster University, John Wiley & Sons LTD, July (1998).
8. Graham, I. "Requirements engineering and rapid development: an object oriented approach". 1. ed. Harlow, Addison Wesley, (1998).
9. Pohl, K. "The Three dimensions of requirements engineering". *Conference on Advanced Information Systems Engineering, A Framework and its Applications*. Information Systems. Proceedings.v. 19, n. 3, p. 243-258,(1993).
10. Kontonya, G., Sommerville, I. "Requirements engineering : process and techiques". 1. ed. Chichester England, John Wiley & Sons, April. ISBN 0-471-97208-8 (2000).
11. International Standard Organization (ISO) SPICE – "Software Process Assessment". Part1: concepts and introductory guide Version 1.00, ISO/IEC (1995).
12. International Standard Organization ISO/IEC 12207, *Software Life Cycle Processes*. (1997).
13. Ministério da Ciência e Tecnologia (MCT). Tradução não oficial do CMU/SEI-93-TR-24-CMM V1.1, *Modelo de Maturidade de Capabilidade de Software (SW-CMM)*, Tradução de: José Marcos Gonçalves André Villas Boas, Versão 1.2, 11/10/2001.
14. Rational Software Corporation - Rational Unified Process - RUP, (2000).
15. Institute of Electrical and Electronics Engineers – IEEE - SWEBOK – "Guide to the software engineering body of knowledge : trial version (version 1.00)", editors, Bourque, P.; Dupuis, R., Tripp, L. L. IEEE Computer Society Press, May, (2001).
16. Kival, C. W., Eratóstenes, E. R. A., Ana Regina, C. R., Cristina, A. F. M., Danilo, S., Clênio, F. S. "Brazilian Software Process Reference Model and Assessment Method", *ISCIS 2005, LNCS 3733*, pp. 402-411, (2005).
17. Object Management Group. "Software process engineering metamodel specification - SPEM". Formal Submission, OMG document number formal/02-11-14, November (2002).
18. Genvigir, E. C., Sant'Anna, N., Borrego, L. F. F., Cereja, M. G. J., Casillo, B. H. "Modelando Processos de Software através do UPM - Modelo de Processo Unificado", XIV Congresso Internacional de Tecnologia de Software Qualidade de Software, CITS, (2003)
19. Basili, V. R., Selby, R. W., Hutchens, D. H. "Experimentation in software engineering" *IEEE Transactions on Software Engineering, USA*, v. 12, n.7, p. 733-743, (1986).
20. Basili, V. R., Caldiera I. G., Rombach H. D. "Goal Question Metric Paradigm. *Encyclopedia of Software Engineering*", v. 1, John Wiley & Sons, (1994).
21. Institute of Electrical and Electronics Engineers – "IEEE Std 830-1998 IEEE Recommended practice for software requirements specifications". New York, IEEE, (1998).
22. Siegel, S. "Estatística não-paramétrica". 1. ed. Rio de Janeiro, MacGraw – Hill, (1954).
23. Mood, A. M. "On the asymptotic efficiency of certain non-parametric two-sample tests". 1. ed. *Ann. Math. Statist.*, 25, p. 514-522, (1954).
24. Maiden, N., Robertson, S. "Developing use cases and scenarios in the requirements process" - ICSE - International Conference on Software Engineering. ACM Digital Library, Minneapolis, (2005).

# Uma Proposta de Modelagem para a Generalização de Elos de Rastreabilidade

Elias Canhadas Genvigir<sup>1 2</sup>  
Nandamudi Lankalapalli Vijaykumar<sup>1</sup>

**Resumo:** Diversos modelos propõem tipos pré-definidos de elos para a rastreabilidade de requisitos. Tais modelos fazem extensivas observações sobre as práticas da rastreabilidade, mas são limitados tanto pelos tipos de elos pré-definidos quanto pela capacidade de incluir atributos para os elos. Este trabalho propõe um modelo para rastreabilidade de requisitos que generaliza os tipos de elos já definidos, permitindo a adição de novos padrões e a inclusão de atributos para os elos que serão utilizados em um determinado processo de rastreabilidade.

**Abstract:** Several models proposed traceability links that provide pre-defined groups of links for requirements traceability. These models are limited to pre-defined links without the ability to add new attributes to the existing links. This work proposes a model for requirements traceability that generalizes the types of links already established in the literature and enables addition of new standards allowing the inclusion of attributes to the links that will be used in a specific traceability process.

## 1 Introdução

Um software é dirigido à realização de tarefas, que estão associadas à solução computacional de um problema do domínio da natureza humana. Dada a amplitude dos problemas aos quais o software pode ser dirigido é necessário que sua produção atinja padrões básicos de qualidade e produtividade.

Com a evolução da pesquisa e a necessidade de melhoria na produção de software a Engenharia de Software foi se especializando em várias subáreas. Uma classificação para essas subáreas foi realizada pelo projeto *Guide to the Software Engineering Body of Knowledge* – SWEBOK [1] que definiu dez áreas de conhecimento para a Engenharia de Software, sendo que a primeira área de conhecimento definida trata sobre os requisitos de software.

Os requisitos são de extrema importância, pois são definidos durante os estágios iniciais do desenvolvimento, como uma especificação do que deve ser implementado descrevendo

---

<sup>1</sup>Universidade Tecnológica Federal do Paraná – UTFPR. Av. Alberto Carazzai, 1640, CEP: 86300–000, Cornélio Procopio, PR, Brasil –{elias@utfpr.edu.br}

<sup>2</sup>Instituto Nacional de Pesquisas Espaciais – INPE. Av. dos Astronautas, 1.758, CEP: 12227–010, São José dos Campos, SP, Brasil –{elias,vijay@lac.inpe.br}

como o sistema deve comportar-se, detalhando os atributos ou propriedades do sistema, ou ainda, podendo estabelecer restrições sobre o processo de desenvolvimento [2]. A importância dos requisitos para o desenvolvimento de sistemas é tão crítica que nenhuma outra parte do trabalho com software incapacita e prejudica tanto o sistema e, depois de concluído, nenhuma outra parte é mais difícil de corrigir do que os requisitos [3].

A área da Engenharia de Software que trata sobre os requisitos de software é conhecida como Engenharia de Requisitos, que é o processo de descoberta dos requisitos, de identificação dos envolvidos e suas necessidades e de documentação de forma que seja útil para a análise, comunicação, e a subsequente implementação [4].

Entre as várias atividades que compõem a Engenharia de Requisitos destaca-se a Rastreabilidade, que é comumente usada para descrever a referência para um grupo coletivo de requisitos baseados em seus relacionamentos.

Os relacionamentos são estabelecidos entre requisitos e artefatos de software usando um elo. Elos são elementos necessários para estabelecer a rastreabilidade, enquanto que artefatos são considerados informações produzidas ou modificadas como parte de um processo de Engenharia de Software [5]. Artefatos podem ser modelos, documentos, código fonte, seqüências de testes, requisitos ou executáveis. Esses são os elementos de um sistema que podem ser rastreados [6].

Existe uma série de propostas para modelos de elos de rastreabilidade, também definidos como modelos de referência[5], que provêm padrões pré-definidos de grupos de elos [5, 7, 8, 9]. Tais pesquisas fazem extensivas observações das práticas da rastreabilidade, mas seus modelos são limitados na fixação dos tipos de elos, ou seja, os grupos de elos são definidos para atingir uma determinada solução para o domínio do problema para o qual o modelo é proposto, o que pode limitar as práticas da rastreabilidade. Em adição, esses modelos não permitem a inclusão de atributos ou semânticas ricas[10, 11] o que restringe a descrição dos elos.

Modelos de referência podem ser muito úteis, visto que grande é o esforço utilizado para analisar o domínio para um novo sistema. Nos casos de domínios padronizados é reportado que o uso de modelos de referência podem reduzir o esforço em até 80% [12]. Entretanto em domínios, não suficientemente padronizados, os modelos de referência podem limitar o desenvolvimento da aplicação.

Este trabalho aborda modelos de referência para rastreabilidade e suas limitações, focando na definição de um modelo que generaliza a definição dos elos para diferentes processos de rastreabilidade. São abordados modelos que representam elos de rastreabilidade e com base nas limitações já mencionadas é proposto um modelo que permite: (i) representar vários tipos de elos já suportados pelos modelos existentes; e (ii) criar novos tipos de elos para servir a uma necessidade específica de um projeto. O modelo apresentado aqui pode

também adicionar atributos para melhorar a semântica dos elos.

O modelo proposto tem por objetivo permitir a criação de diferentes tipos de elos pelos desenvolvedores, além da inserção de atributos a esses elementos, possibilitando que os desenvolvedores possam definir padrões de elos, conforme as necessidades específicas de seus projetos, viabilizando um maior nível de detalhes sobre a rastreabilidade.

A pesquisa é organizada da seguinte forma: a seção 2 apresenta os conceitos sobre rastreabilidade, seus aspectos e importância e sobre elos e seus tipos. Na sequência é explorado, brevemente, o conceito de modelos, necessário para compreender o uso de modelos para rastreabilidade e então é apresentado o modelo proposto, seguido de sua modelagem e demonstração.

## 2 Rastreabilidade

A rastreabilidade está intimamente associada ao processo de produção de artefatos de software, especificamente aos requisitos e a capacidade de estabelecer vínculos entre esses requisitos e outros artefatos que os satisfaçam. Essa característica é observada por Letelier [7] que afirma que a rastreabilidade de requisitos ajuda a garantir uma contínua concordância entre os requisitos dos interessados no sistema e os artefatos produzidos ao longo do processo de desenvolvimento de software. Na mesma linha Palmer [13] aponta que a rastreabilidade dá a assistência essencial na compreensão do relacionamento que existe com e entre requisitos de software, projeto e implementação sendo que estes relacionamentos auxiliam o projetista a mostrar quais elementos do projeto satisfazem os requisitos.

Sobre as vantagens da rastreabilidade é observado que seu uso ajuda a estimar variações em cronogramas e em custos do projeto [13, 14, 15, 16]. Além disso, Sayão e Leite [17] apontam que a rastreabilidade pode auxiliar gerentes de projeto a: verificar a alocação de requisitos a componentes de software; resolver conflitos entre requisitos; verificar requisitos nos processos de testes; corrigir defeitos através da identificação do componente de origem do erro; validar o sistema junto aos clientes; analisar o impacto na evolução dos sistemas; prever custos e prazos; e gerenciar riscos e reuso de componentes.

Dependendo de sua semântica, a rastreabilidade pode ser usada para (a) assistir o processo de verificação dos requisitos para um sistema, (b) estabelecer o impacto de mudanças na especificação de requisitos através de seus artefatos ou da documentação (Ex. projeto, teste e implementação de artefatos), (c) compreender a evolução de um artefato, e (d) compreender os aspectos do projeto e o suporte de *Rationales*. Assim, a geração e a manutenção de tais relações podem fornecer uma base mais eficaz para a garantia da qualidade do sistema, a gerência das mudanças, e a manutenção do software [18].

Apesar de ser usada por vários processos ainda é evidente a falta de padrões para a

qualidade da rastreabilidade, isso pode ser observado nas normas e modelos como o CMMI-DEV [19] que aponta a prática específica: 1.4 Manter a Rastreabilidade Bidirecional para os Requisitos, ou a norma ISO/IEC 15504 -2 [20] que possui as práticas base: ENG.3.4 – Estabelecer a rastreabilidade e PRO.4.5 – Manter a rastreabilidade, em ambos casos propõem-se apenas que a rastreabilidade seja executada mas nenhum padrão para a qualidade da rastreabilidade é estabelecido.

Um ponto fundamental para a qualidade trata sobre as métricas, mas existem poucas pesquisas sobre esse tema aplicado à rastreabilidade. Algumas métricas são propostas por Costello e Liu [21] que definem cinco tipos: cobertura de próximo nível (COV), profundidade plena e alta cobertura (DHCOV), vinculação estatística, rastreabilidade inconsistente (ITM), e rastreabilidade indefinida (UTM); e por Hull et al. [22] que definem cinco elementos para a mensuração da rastreabilidade: amplitude, profundidade, crescimento, balanço e mudança latente.

Sobre os tipos de rastreabilidade basicamente existem dois tipos, a pré-rastreabilidade que está concentrada no ciclo de vida dos requisitos antes de serem incluídos na especificação de requisitos, e a pós-rastreabilidade, que está concentrada no ciclo de vida dos requisitos após serem incluídos na especificação [23].

A capacidade de rastrear um requisito até seus refinamentos é definida como rastrear para frente (*Forwards*), e a capacidade de rastrear um refinamento até sua origem é definida como rastrear para trás (*Backwards*) [14]. Os dois tipos são essenciais e fazem parte tanto da Pós quanto da pré-rastreabilidade.

## 2.1 Elos de Rastreabilidade

Como já afirmado a rastreabilidade é realizada entre requisitos e demais artefatos fazendo uso de elos, esses elementos podem ser formalizados como:  $A = \{a_1, a_2, a_3, \dots, a_n\}$  representa todos os artefatos produzidos e identificados no processo de desenvolvimento, então  $E = \{e_1, e_2, e_3, \dots, e_n\} \subset A$  é um subgrupo dos artefatos que representam elos, e  $R = \{r_1, r_2, r_3, \dots, r_n\} \subset A$  é um subgrupo de  $A$  que representam os requisitos.

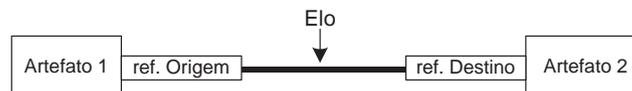
Várias propostas têm sido feitas para elos de rastreabilidade assim como para modelos que suportem: (i) algum padrão focado na manutenção do processo e (ii) a representação desses elos.

Basicamente os elos estão associados à uma metodologia ou à alguma informação do domínio do processo de desenvolvimento. Neste aspecto, elos foram criados para orientação a objetos [18], orientação a aspectos [24], desenvolvimento centrado em visão [25], e desenvolvimento dirigido a modelo [26, 27]. Além disso elos podem ser criados por aspectos ambientais e organizacionais [5]; no tipo de informação a ser rastreada [9]; ou ainda por con-

figurações que integram especificações textuais [7]. O principal problema desses elos é que eles são criados e usados somente para um propósito específico.

No caso do processo de engenharia de requisitos os elos são usados nas atividades de: validação, análise, evolução e referência cruzada entre requisitos e artefatos [13]. Também fazem uso desse recurso os processos de gerência de projeto (auxiliar a predição de custo, restrições ou impactos); de manutenção; de teste (geração de casos de teste baseados em cenários ou modelos); e no processo da garantia da qualidade (validação do software)[6, 8, 13, 23, 28, 29, 30].

Um elo representa explicitamente o relacionamento definido entre dois artefatos  $a_1$  e  $a_2$ .  $a_1 \rightarrow a_2$  são considerados como diretamente ligados enquanto um elo indireto usa um artefato intermediário como em  $a_1 \rightarrow a_2 \rightarrow a_3$  [6]. Um elo é estabelecido entre um artefato origem e um artefato destino, Figura 1.



**Figura 1.** Elementos básicos de um elo.

Informações sobre a origem e o destino são suficientes para suportar a rastreabilidade para frente e para trás, mas outras atividades da engenharia de requisitos, como o auxílio à previsão de custos e prazos e a resolução de conflitos, necessitam de outros elementos para sua execução [10, 11].

Várias categorias de elos podem ser determinadas usando como base os atributos e propriedades ou a aplicação desses elos no processo de desenvolvimento. Devido a essas características existem muitos tipos de elos descritos na literatura. A Tabela 1 mostra alguns desses elos.

**Tabela 1.** Exemplos de elos descritos na literatura classificados por seus respectivos autores e subgrupos.

<b>Autor</b>	<b>Grupos</b>	<b>Tipos de elo</b>	
Ramesh e Jarke[5]	Relacionado ao produto	Satisfação Dependência	
	Relacionado ao processo	Evolução <i>Rationales</i>	
Toranzo e Castro [9]		Satisfação Recurso Responsabilidade Representação Alocação Agregação	
	Pohl [31]	Condição	Restrições Pré-condições
		Documentos	Exemplos Propósito Caso de teste Comentários Segundo Plano
		Abstração	Refinado Generalizado
		Evolução	Elaborado Formalizado Baseado em Satisfação Substituído
	Conteúdo	Similar Comparação Contradição Conflito	
De Lucia et al. [32]		Dependência Composição Indireto	
	Spanoudakis et al. [18]		Sobreposição Execução_Requerida Característica_Requerida Parcialmente_Realizável
		Aizenbud-Reshef et al.[26]	
Computacional			Derivação Análise

## 2.2 Técnicas de Rastreabilidade

O desenvolvimento e o uso de técnicas de rastreabilidade tiveram início na década de 1970 [26] e o primeiro método usado para expressar e manter a rastreabilidade foi a referência

cruzada [33].

Outras técnicas podem ser usadas tanto para representar como para estabelecer relacionamentos incluindo matrizes [14, 34]; dependência de frases chaves [35]; integração de documentos [36]; hipertexto [37, 38, 39], grafos [8]; métodos formais [40]; esquemas dinâmicos [6, 41, 42]; sistemas baseados em suposição da manutenção de verdade [43, 44]; e redes de confiança [45].

Técnicas automatizadas ou semi automatizadas de rastreabilidade também foram desenvolvidas, como a detecção de existência de elos entre documentos de requisitos, documentos de projeto, e código fonte usando uma máquina de aprendizagem [46]; técnicas de análise altamente escaláveis para análise automática de consistência entre requisitos e projetos [47]; técnicas para recuperação de informação – RI (*Information Retrieval*) [48, 49, 50] como a indexação por análise da semântica latente [51, 52] que são usadas para recuperar elos de rastreabilidade ou na recuperação por construção de elos sobre a formalização de semânticas [53].

Pesquisas mais recentes vêm mostrando que as técnicas baseadas em RI podem ser usadas para descobrir dinamicamente elos [41, 54, 55, 56], além de permitir o relacionamento entre vários tipos de artefatos [53] como código e documentação [55, 57], requisitos e projeto [54], artefatos e requisitos [58]. Outra característica dessas técnicas é que elas podem prover a rastreabilidade sem a necessidade de manter armazenados os elos.

A técnica conhecida como modelo de reflexão de software [59] 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 nesta mesma linha é o de Richardson e Green [60], que usa uma síntese automatizada para inferir elos. Perturbações no artefato de origem são analisadas e usadas para identificar os elos entre os artefatos de origem e destino.

Além da rastreabilidade entre artefatos algumas técnicas como a rastreabilidade baseada em eventos [6] podem ser usadas para rastrear requisitos associados a desempenho em modelos de desempenho executáveis [6, 61, 62], e também para rastrear a qualidade de requisitos implementados através de padrões de projeto já conhecidos [63], que podem ser ligados usando seus invariantes [64].

### **2.3 Modelos para rastreabilidade**

Segundo Mellor et al. [65] 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 uma linguagem (tanto textual ou gráfica) que por sua vez possui certo grau de abstração.

No caso da rastreabilidade, os modelos são desenvolvidos com base nas informações sobre um determinado domínio da rastreabilidade (usuários, práticas, metodologias, normas, etc.). Assim a definição dos elos está associada com os conceitos modelados.

Ramesh e Jarke [5] criaram um meta-modelo para rastreabilidade utilizando como base uma extensa pesquisa empírica sobre as conseqüências dos diferentes usos, perspectivas e necessidades de informação dos envolvidos no processo de desenvolvimento.

Os autores encontraram três categorias de informações associadas à rastreabilidade: interessados; fontes (padrões, normas, regras institucionais, etc); e objetos (objetos conceituais, modelos, ou artefatos), que foram utilizadas para estabelecer o meta-modelo.

Além das três classes de informação a pesquisa permitiu aos autores criar uma classificação dos usuários da rastreabilidade. Esses foram divididos, quanto suas práticas, em dois grupos distintos: usuários sofisticados e normais (*high-end e low-end users*). Com base nessa classificação dois modelos foram criados para representar seus processos de rastreabilidade.

Adicionalmente à criação do meta-modelo, conceitualmente simples mas muito amplo em seus elementos e elos, o trabalho de Ramesh e Jarke [5] apresenta uma classificação muito interessante sobre os usuários da rastreabilidade, dando grande valor às práticas utilizadas por estes nas organizações. Entretanto, esse modelo também pré-estabelece os tipos de elos a serem utilizados, permitindo apenas a divisão das classes pré-estabelecidas. Outro problema encontrado no modelo é que este não possibilita a adição de atributos aos elos, o que permitiria o enriquecimento da semântica desses artefatos, apesar da existência do tipo de elo *Rationale* que é importante para determinar as razões da criação ou alterações de artefatos.

O modelo de Toranzo e Castro [9] faz o uso de representação gráfica através de diagramas de classes da UML. Esse modelo tem como conceito a classificação das informações a serem rastreadas, que são divididas em quatro classes: Ambientais, Organizacionais, Gerenciais e de Desenvolvimento. Com base na classificação os autores desenvolveram um meta-modelo; um modelo intermediário e um processo para aplicação das estratégias anteriores sobre a rastreabilidade. É dada atenção a aspectos gerenciais, como a definição do elo do tipo responsabilidade. Esse aspecto é notado pela influência dos contextos da classificação definida (Ambientais, Organizacionais, Gerenciais e de Desenvolvimento). Entretanto, os tipos de elos também são pré-definidos sendo focados apenas nos contextos utilizados no modelo, devido a essa característica outros padrões de gerência ou de rastreabilidade, como dirigida à visão ou a modelos poderão não ser completamente expressos.

Uma discussão sobre o problema da pré-especificação de elos nos modelos de rastreabilidade é apresentada por Aizenbud-Reshef et al. [26]. Os autores propõem uma solução, para este problema, usando o conceito de *Model Driven Architecture* – MDA. Esta proposta considera que a criação de modelos para sistemas deve incluir um meta-modelo de requisi-

tos, sendo que os modelos resultantes, que incluem um modelo explícito de requisitos, podem prover a rastreabilidade entre requisitos e outros artefatos. A existência de tais modelos não provê mecanismos de como produzir, automaticamente, os elos entre requisitos e suas dependências, ou seja, segundo os autores o fato de se possuir uma semântica para produzir elos significativos não garante uma efetiva estratégia de rastreabilidade. Assim os elos necessitam ser construídos e mantidos manualmente.

Segundo Aizenbud–Reshef et al. [26] o ideal seria gerar os artefatos, ou suas estruturas básicas, e produzir os elos entre eles, e se necessário os elos seriam detalhados. Para tanto a MDA poderia ser uma alternativa para realizar essas tarefas.

Apesar da crítica sobre os modelos de rastreabilidade, que pré–definem os tipos de elos, a proposta de Aizenbud–Reshef et al. está vinculada a uma solução baseada em arquitetura, o que pode ser um problema para processos de desenvolvimento baseados em outros conceitos. Entretanto, esta proposta pode ser muito relevante para os envolvidos com a MDA.

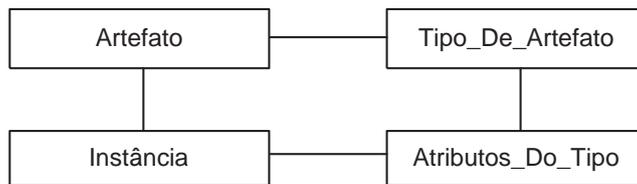
Os modelos existentes provêm padrões pré–definidos, de grupos de elos, que podem ser suportados pelas soluções conceituais propostas por esses modelos. Esses trabalhos fazem extensivas observações das práticas da rastreabilidade, mas seus modelos são limitados na fixação dos tipos de elos, ou seja, são definidos para cobrir uma determinada abordagem de solução para o domínio do problema ao qual seus modelos são propostos [26].

Outro problema vigente na área abrange a restrição da descrição dos elos, sendo que estes, na maioria dos casos, representam apenas a ligação entre dois elementos do processo de desenvolvimento, ou seja, esses modelos não permitem a inclusão de atributos ou semânticas ricas, assim informações necessárias para estabelecer análises, como avaliação de impacto, derivação ou cobertura [10, 11, 22], sobre a rastreabilidade, não são apresentadas, representadas ou mesmo suportadas por esses modelos.

### **3 Modelo Proposto**

Os modelos apresentados são limitados no que diz respeito aos tipos de elos, ou seja, são fixados, e conseqüentemente são definidos para lidar apenas com soluções específicas para um domínio fixo de problemas.

Com base nas restrições dos modelos já apresentados é proposto um modelo baseado em quatro elementos que podem representar vários tipos de artefatos. Diferente das outras abordagens o modelo proposto permite que sejam definidos tanto os padrões dos requisitos, dos artefatos ou dos elos a serem utilizados ao longo do projeto. A Figura 2 apresenta os quatro elementos que compõem o modelo.



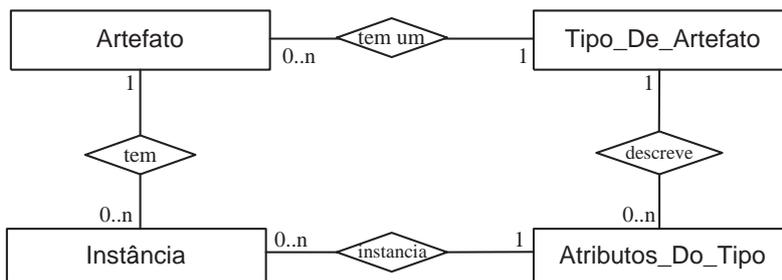
**Figura 2.** Elementos do modelo proposto para a rastreabilidade de requisitos.

O modelo se baseia na generalização de todos os tipos de artefatos que possam fazer parte do processo de rastreabilidade, focando a representação das informações. Essa característica permite tanto a criação de novos tipos de elos quanto de outros artefatos, conforme a necessidade dos envolvidos na rastreabilidade. Dois casos particulares justificam essa generalização. O primeiro trata sobre a agregação de novos campos para um dado tipo de elo ou artefato. O segundo trata sobre a inserção de novos tipos de artefatos.

Para o primeiro caso os elos constituem, na maioria das aplicações existentes, apenas a existência da ligação entre dois artefatos, sendo que informações importantes não são agregadas a essa ligação. Ao constituir um elo de rastreabilidade deve-se permitir que sejam inseridas informações do processo ou adjacentes a esse, como informações sobre qualidade e *Rationale*.

O segundo caso consiste da necessidade de inclusão de novos tipos de elos ou outros artefatos ao longo do projeto ou para um novo projeto. Permitir a definição dos tipos de elos somente no início do projeto não condiz com a necessidade de evolução e adaptação do projeto frente às alterações que possam vir a ser realizadas.

A relação entre os elementos do modelo pode ser visualizada através de um diagrama entidade relacionamento, Figura 3.



**Figura 3.** Diagrama Entidade Relacionamento do modelo proposto.

No modelo relacional um esquema de relação é dado por  $P(A, K)$ , onde  $P$  é o nome dado à relação e  $A = \{A_1, A_2, \dots, A_n\}$  é um finito grupo de atributos. Cada atributo  $A_i$  representa um papel desempenhado por algum domínio  $D$ ,  $\text{dom}(A_i)$ . O número  $n$  de atributos

define o grau da relação. Enquanto que  $K$  consiste de um grupo finito de atributos chaves  $K = \{K_1, K_2, \dots, K_n\}$ , onde  $K \subseteq A$ . Dizer que  $K = \{K_1, K_2, \dots, K_n\}$  é a chave de um esquema de relação é dizer que qualquer relação válida  $p$  em  $P$  tem a propriedade que para qualquer tupla distinta  $t_1$  e  $t_2$  em  $p$ ,  $t_1(K) \neq t_2(K)$ , e que nenhum outro subgrupo de  $K$  tem essa propriedade.

Assim, como no modelo relacional, o modelo proposto é definido como um grupo  $S$  formado por quatro esquemas de relação  $S=(Tipo\_De\_Artefato (TF), Atributos\_Do\_Tipo (AT), Instancias (I), Artefatos (A))$ . A Tabela 2 mostra as relações e seus atributos.

**Tabela 2.** Descrição dos elementos do Modelos.

Nome da Relação	Nomes dos Atributos	Domínio
Tipo_De_Artefato	TipoDeArtefatoID	chave primária
	Nome	nomes de artefatos
	Descrição	descrição dos artefatos
Atributos_Do_Tipo	AtributosDoTipoID	chave primária
	TipoDeArtefatoFK	chave estrangeira para $TF$
	Name	nomes dos atributos da relação $TF$
	Descrição	descrição dos atributos
Instância	InstânciaID	chave primária
	ArtefatoFK	chave estrangeira da relação $A$
	AtributosDoTipoFK	chave estrangeira da $AT$
	Valor	valores para os atributos do artefato
Artefato	ArtefatoID	chave primária
	TipoDeArtefatoFK	chave estrangeira de $TF$
	Data	data da criação do artefato
	Versão	versão do artefato
	Descrição	descrição do artefato

Cada elemento de  $S$  pode ser instanciado. Por exemplo, *Artefato* pode ser instanciado como  $a_i(A)$  tendo grau 5 (domínios: ArtefatoID, TipoDeArtefatoFK, Data, Versão, e Descrição), exemplo:

$$a_i(A) \subseteq (dom(ArtefatoID) \times dom(TipoDeArtefatoFK) \times dom(Data) \times dom(Versão) \times dom(Descrição))$$

O mesmo conceito se aplica aos outros elementos de  $S$ .

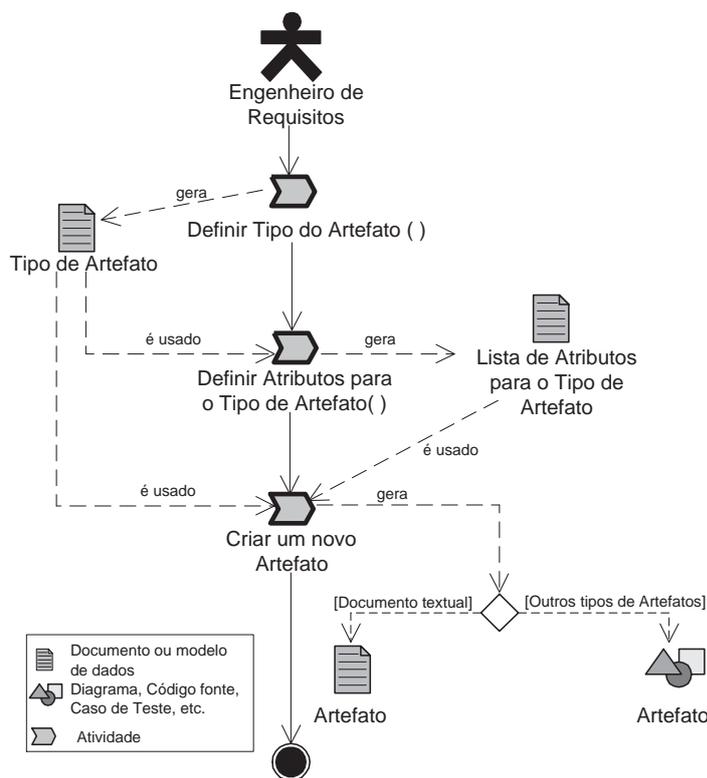
Para criar um artefato  $a_i$ , no modelo proposto, são necessários os seguintes passos:

1. Definir o tipo de artefato – Especificar o nome e a descrição de uma categoria na qual uma instância de artefato fará parte, como: código fonte, documento, requisito, elo,

- caso de uso, etc.
2. Determinar os atributos do tipo do artefato – Os atributos são usados para detalhar o tipo de artefato. Por exemplo, um tipo de artefato caso de uso pode possuir os seguintes atributos: nome, ator, pré-condições, fluxo principal, etc.
  3. Criar um artefato – A criação de um artefato depende do tipo ao qual o novo artefato será instanciado e do domínio de valores que serão atribuídos a ele. Suas características são definidas pela relação *Atributos\_Do\_Tipo*, assim através das chaves dessa relação é possível associar os atributos.

Os valores dos atributos de  $a_i$  são designados na relação *Instância*, que através de suas restrições estabelece que esses valores pertencem exclusivamente ao artefato  $a_i$ .

O processo de criação de um artefato pode ser visualizado através de um diagrama de atividades usando a notação do *Software Process Engineering Metamodel Specification–SPEM* [66, 67] proposto pelo Object Management Group – OMG, Figura 4.



**Figura 4.** Diagrama de atividades do processo de criação de um artefato usando o modelo.

O artefato  $a_i$  é definido pelos atributos da relação *Artefato*, pelo tipo definido na rela-

ção *Tipo\_Do\_Artefato*, e pelos valores armazenados na relação *Instância*.

Na Figura 5 um artefato  $a_i$  é definido como sendo do tipo Requisito Funcional e tendo dois atributos (Descrição e Fator de Risco). Cada um desses atributos tem suas instâncias (“O sistema deve...” e “Risco1”) que estão associados ao artefato  $a_i$ .

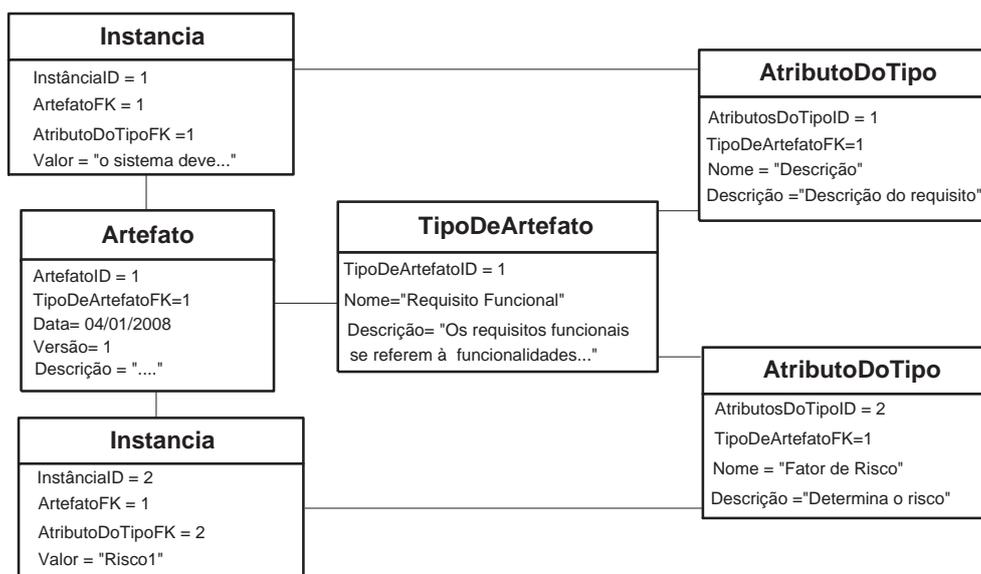


Figura 5. Valores do artefato  $a_1$ .

A visualização do artefato  $a_1$  pode ser feita através de três operações conhecidas na álgebra relacional: projeção ( $\pi$ ), seleção ( $\sigma$ ) e junção natural ( $\bowtie$ ):

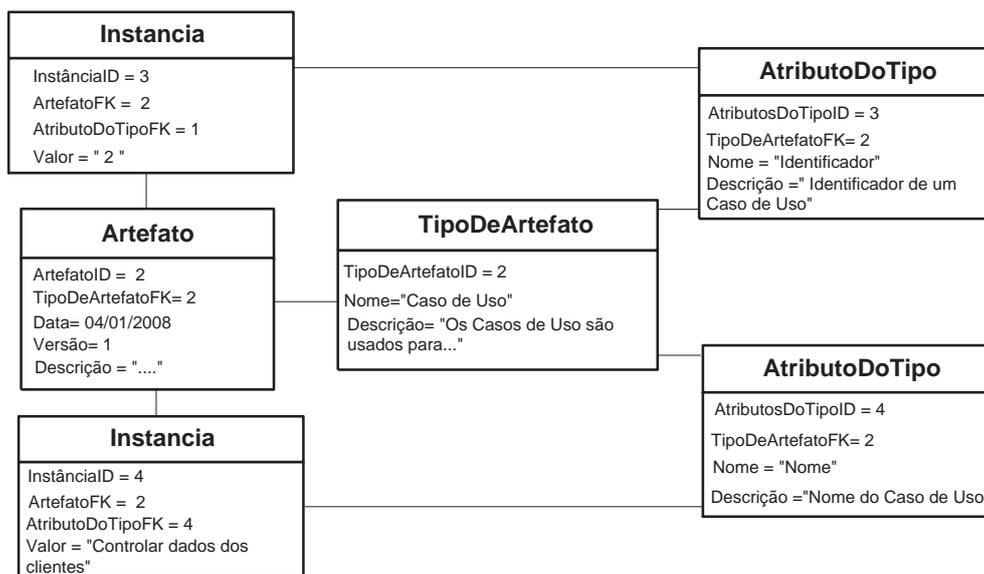
$$\pi_{a.ArtefatoID,tf.Nome,at.Nome,i.Valor}(\sigma_{a.ArtefatoID=1}(a \bowtie i \bowtie tf \bowtie at)). \quad (1)$$

A Tabela 3 mostra o resultado da expressão 1.

Tabela 3. Resultado da expressão 1.

a.ArtefatoID	tf.Nome	at.Nome	i.Valor
1	Requisito Funcional	Descrição	O sistema deve...
1	Requisito Funcional	Fator de Risco	Risco1

Todos os atributos necessários para documentar um determinado artefato podem ser criados através da relação *at*. Para estabelecer um elo é necessário considerar a criação de pelo menos mais um artefato, dito  $a_2$ . Aqui  $a_2$  é do tipo *tf* = “Caso de Uso”, que possui dois atributos *at* = “Identificador” e *at* = “Nome”, que têm valores instanciados *i.valor* = “2” e *i.valor* = “Controlar dados do cliente”. O artefato  $a_2$  é apresentado na Figura 6.



**Figura 6.** Valores do artefato  $a_2$ .

A partir da criação de  $a_1$  e  $a_2$  um elo pode ser estabelecido entre eles. Para criar um elo o procedimento é similar ao aplicado aos outros artefatos. O elo a ser criado é um artefato, definido como  $a_3$ , do tipo  $tf$ ="Elo", que consiste de três atributos  $at.nome$ ="Identificador",  $at_1.nome$ ="Código da Origem",  $at_2.nome$ ="Código do Destino" que tem como valores de instâncias  $i_1.valor=1$ ,  $i_2.valor=1$ , e  $i_3.valor=2$ . A Figura 7, ilustra  $a_1 \rightarrow a_2$  que é executado por  $a_3$ .

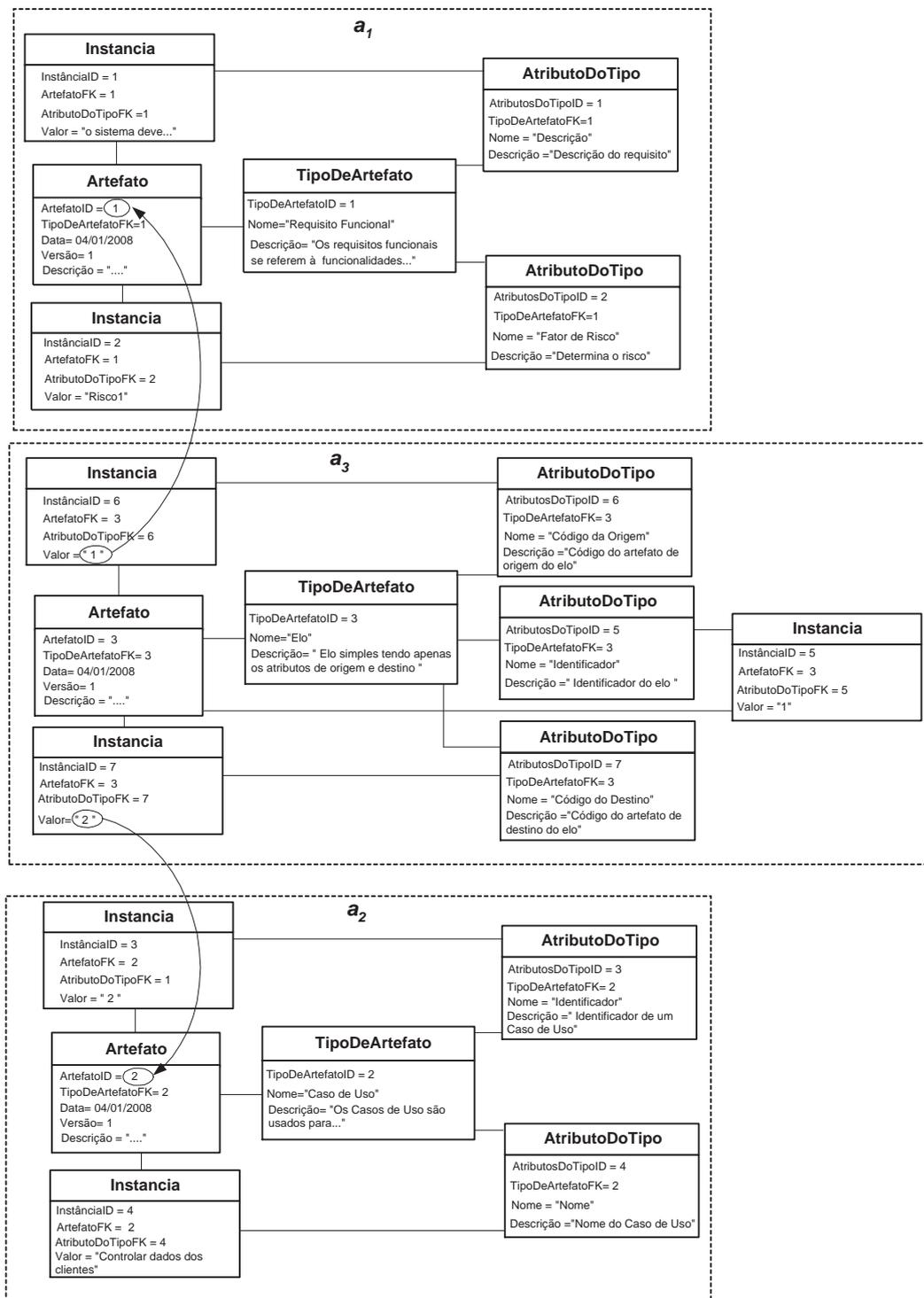


Figura 7. Valores do artefato  $a_3$ . O artefato  $a_3$  é do tipo elo usado para relacionar os artefatos  $a_1$  e  $a_2$ .

O elo apresentado por  $a_3$  pode ser visualizado usando as operações de projeção ( $\pi$ ),

seleção ( $\sigma$ ) e junção natural ( $\bowtie$ ):

$$\pi_{a.ArtefatoID,at.AtributoID,tf.Nome,i.Valor}(\sigma_{a.ArtefatoID=3}(a \bowtie i \bowtie tf \bowtie at)). \quad (2)$$

O resultado da Expressão 2 é apresentado na Tabela 4.

**Tabela 4.** Resultado da Expressão 2.

<b>a.ArtefatoID</b>	<b>at.AtributoID</b>	<b>ak.Nome</b>	<b>i.Valor</b>
3	5	Identificador	1
3	6	Código da Origem	1
3	7	código do Destino	2

Através do resultado da Expressão 2, é possível verificar que *i.Valor* apresenta os valores que correspondem ao código do artefato de origem e o código do artefato de destino referentes a um elo. Além dos atributos de identificação (origem e destino) outros atributos podem ser adicionados ao elo, o que permite aumentar a capacidade desse elemento dentro do processo de rastreabilidade. O modelo permite que *n* novos atributos sejam inseridos em um determinado tipo de elo, desta forma atributos que podem ser utilizados para a elaboração de métricas como tempo, esforço, custo, padrões para *Rationales* ou risco podem ser inseridos ao tipo de elo, tal propriedade amplia a capacidade de elaboração métricas para uso, tanto do processo de rastreabilidade quanto para outros processos do projeto.

## 4 Conclusão

A maior desvantagem dos modelos existentes é que estes estão vinculados a tipos de elos pré-definidos. Nestes modelos quando existe a possibilidade de criação de um novo tipo de elo este se torna um subtipo do elo já pré-definido, o que pode ocasionar a herança de comportamento, quando estas são definidas, o que nem sempre é desejado.

A vantagem em permitir a definição de novos tipos de elos e de novos atributos é a criação de modelos intermediários o que aumenta o domínio dos modelos existentes. Como exemplo, o elo Agregação é definido em alguns modelos enquanto o elo Evolução é definido em outros. O modelo proposto pode incluir ambos e, adicionalmente, permitir a criação de tipos de subtipos através de atributos que determinem um dado artefato como um subelemento.

A principal vantagem do modelo proposto é a generalização sobre o domínio dos tipos de elos de rastreabilidade e a agregação de atributos descritivos para esses elos.

A generalização permite a criação de novos tipos, mas ela aumenta a complexidade para implementar sistemas de rastreabilidade. Isso ocorre devido ao fato de que a criação de

novos atributos, para uma dada necessidade, pode não pertencer ao grupo de restrições do modelo. A definição desses atributos implica um maior controle para que a integridade dos dados não seja afetada.

Uma solução para a criação de novos tipos de elos é mostrar aos usuários, quando iniciar um novo processo de rastreabilidade, um grupo pré-definido de elos que possibilitam a implementação de novos atributos, tanto para os novos elos quanto para os pré-definidos.

Além da criação de tipos de elos, a definição de atributos para um novo tipo pode permitir a redução do número de matrizes de rastreabilidade, pois informações como *Rationales* não precisam ser expressas em uma matriz específica. Desta forma, uma única matriz entre requisitos e componentes pode possuir informações sobre satisfação, *Rationales*, dependência ou custo.

Outro importante ponto que envolve a inclusão de atributos a elos trata sobre definição de métricas para a rastreabilidade. Como observado na seção 2, poucos são os trabalhos que exploram a qualidade da rastreabilidade assim como a definição de métricas para esse processo. O modelo habilita a incorporação de novos atributos para os elos permitindo que sejam adicionadas as métricas já apresentadas ou novos padrões associados a tempo, custo e esforço, tanto para o processo quanto para os artefatos, o que pode facilitar outras atividades do processo de desenvolvimento.

Trabalhos futuros devem explorar alternativas que conduzam a um conjunto de elos que sejam adequados as necessidades da aplicação, o que envolve tanto as necessidades de qualidade estabelecidas por usuários, desenvolvedores, clientes e fornecedores. Entre essas alternativas o uso de mecanismos de inteligência artificial, como sistemas especialistas ou agentes, poderiam ser usados para monitorar e analisar as alterações nos elos e seus resultados aplicados para explorar as possibilidades sobre métricas para a rastreabilidade.

## Referências

- [1] SWEBOK – Guide to the software engineering body of knowledge: trial version(version 1.00), IEEE Computer Society Press, May 2004.
- [2] Sommerville, I., Sawyer, P., Requirements engineering: a good practice guide, 1nd edition, John Wiley & Sons, July 1998.
- [3] Brooks, F.P.Jr., The mythical man–month: essays on software engineering, 2nd edition, Addison Wesley,(Reading, Massachusetts, 1995).
- [4] Nuseibeh, B., Easterbrook, S. Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering (ICSE'00), pp. 35–46, 2000.

- [5] Ramesh, B. Jarke, M. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 27 (1), 58–93, 2001.
- [6] Cleland–Huang, J. Chang, C.K., Christensen, M. Event–Based Traceability for Managing Evolutionary Change. *IEEE Transactions on Software Engineering*, 29 (9), 796–810, 2003.
- [7] Letelier, P. A Framework for Requirements Traceability in UML–based Projects. In: *Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering*, pp. 30–41, 2002.
- [8] Pinheiro, F.A.C., Goguen, J.A. An Object–Oriented Tool for Tracing Requirements. *IEEE Software*, 13 (2), 52–64, 1996.
- [9] Toranzo, M., Castro, J., E. Mello. Uma proposta para melhorar o rastreamento de requisitos. In: *Proceedings of the Workshop de Engenharia de Requisitos*, pp. 194–209, 2002.
- [10] Dick, J. Rich Traceability. In: *Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering*, at the 17th IEEE Conference on Automated Software Engineering (ASE), 2002.
- [11] Dick, J. Design Traceability. *IEEE Software* November/December, 14–16, 2005.
- [12] Scheer, A. W. *Business Process Engineering: Reference Models for Industrial Enterprises*. Springer-Verlag, 1998.
- [13] Palmer, J.D. Traceability. (in R.H. Thayer and M. Dorfman (eds.). *Software Requirements Engineering*. 2nd, IEEE Computer Society Press, p. 412–422), 2000.
- [14] Davis, A. M. *Software requirements: objects, functions and states*, Prentice–Hall, New Jersey, 1993.
- [15] Dömges, R. Pohl, K. Adapting Traceability Environments to Project–Specific Needs. *Communications of the ACM*, 41(12), 54–62, 1998.
- [16] Pinheiro, F. A. C. *Requirements Traceability*, Chapter of the Book *Perspectives On Software Requirements*. Kluwer Academic Publishers, 2003.
- [17] Sayão, M. Leite, J. C. S. P. Rastreabilidade de Requisitos. *Revista de Informática Teórica e Aplicada*, 13 (1), 57–86, 2006.
- [18] Spanoudakis, G., Zisman, A., Péres–Miñana, E., Krause, P. Rule–based generation of requirements traceability relations, *The Journal of Systems and Software*, 72, 105–127, 2004.

- [19] Software Engineering Institute. CMMI for Development (CMMI-DEV, V1.2). CMU/SEI-2006-TR-008, Software Engineering Institute, Carnegie Mellon University, 2006.
- [20] ISO/IEC 15504-2:2003/Cor.1:2004(E). Information Technology – Process Assessment – Part 2: Performing an Assessment. International Organization for Standardization: Geneva, 2003.
- [21] Costello, R., J., Liu, D. Metrics for requirements engineering. *Journal of System and Software*, 29(1), 39–63, 1995.
- [22] Hull, E., Jakson, K., Dick, J. *Requeriments Engineering*. Spring Verlag, London, September, 2002.
- [23] Gotel, O.C.Z. Finkelstein, A. An Analysis of the Requirements Traceability Problem. In: Proceedings of the First Int'l Conf. Requirements Eng., pp. 94–101, 1994.
- [24] Egyed, A. Resolving Uncertainties during Trace Analysis. In: Proceedings of the 12th ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE), pp. 3–12, 2004.
- [25] Sabetzadeh, M. Easterbrook, S. Traceability in Viewpoint Merging: A Model Management Perspective. In: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering – TEFSE, pp. 44–49, 2005.
- [26] Aizenbud-Reshef, N., Nolan, B.T., Rubin, J., Shaham-Gafni, J. Model traceability, *IBM Systems Journal*, 45 (3), 515–526, 2006.
- [27] Almeida, J. P., Eck, P. V., Iacob, M. Requirements Traceability and Transformation Conformance in Model-Driven Development. In: Proceedings of the 10th IEEE international Enterprise Distributed Object Computing Conference –Edoc'06, pp. 355–366, 2006.
- [28] Jarke, M. Requirements Traceability, *Communications of ACM*, 41 (12), 32–36, 1998.
- [29] Papadopoulou, P. Evaluation of a requirements traceability system, MSc Thesis, Department of Computing, City University, 2002.
- [30] Riebisch, M., Hubner, M. Traceability-Driven Model Refinement for Test Case Generation. In: Proceedings of the 12th IEEE international Conference and Workshops on the Engineering of Computer-Based Systems –(Ecsb'05), pp. 113–120, 2005.
- [31] Pohl, K. PRO-ART: Enabling Requirements Pre-Traceability. In: Proceedings of the 2nd International Conference on Requirement Engineering, pp. 76–84, 1996.

- [32] De Lucia, A., Fasano, F., Oliveto, R., Tortora, G. ADAMS Re-Trace: a Traceability Recovery Tool. In: Proceedings of the 9th European Conference on Software Maintenance and Reengineering –CSMR’05, pp. 32–41, 2005.
- [33] Evans, M.W. *The Software Factory*, John Wiley and Sons, 1989.
- [34] West, M. Quality Function Deployment in Software Development. In: Proceedings of the IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, pp. 5/1–5/7, 1991.
- [35] Jackson, J.A Keyphrase Based Traceability Scheme. In: Proceedings of the IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, pp. 2/1–2/4, 1991.
- [36] Lefering, M. An Incremental Integration Tool Between Requirements Engineering and Programming in the Large. In: Proceedings of the IEEE International Symposium on Requirements Engineering, pp. 82–89, 1993.
- [37] Alexander, I. Toward Automatic Traceability in Industrial Practice. In: Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 26–31, 2002.
- [38] Glinz, M. A Lightweight Approach to Consistency of Scenarios and Class Models. In: Proceedings of the Fourth Int’l Conf. Requirements Eng., 2000.
- [39] Kaindl, H. The Missing Link in Requirements Engineering. *ACM SIGSOFT Software Engineering Notes*, 18 (2), pp.30–39, 1993.
- [40] Cooke, J., Stone, R.A Formal Development Framework and Its Use to Manage Software Production, Tools and Techniques for Maintaining Traceability During Design. In: Proceedings of the IEEE Colloquium Computing and Control Division, pp. 10/1,1991.
- [41] Antoniol, G. Casazza, G. Cimitile, A. Traceability Recovery by Modeling Programmer Behavior. In: Proceedings of the Seventh IEEE Working Conf. Reverse Eng., pp. 240–247, 2000.
- [42] Tryggeseth, E., Nytrø, O. Dynamic Traceability Links Supported by a System Architecture Description. In: Proceedings of the IEEE International Conference on Software Maintenance, pp. 180–187, 1997.
- [43] Tang, M. X. A knowledge-based architecture for intelligent design support. *The Knowledge Engineering Review*. 12(4), 387–406, 1997.

- [44] Smithers, T., Tang, M.X., Tomes, N. The Maintenance of Design History in AI-Based Design. In: Proceedings of the IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, pp. 8/1, 1991.
- [45] Bowen, J., O'Grady, P., Smith, L. A Constraint Programming Language for Life-Cycle Engineering. *Artificial Intelligence in Engineering*, 5 (4), 206–220, 1990.
- [46] Faisal, M.H. Toward automating the discovery of traceability links. Doctoral Theses University Of Colorado, p. 118, 2005.
- [47] Chechik, M., Gannon, J. Automatic Analysis of consistency between Requirements and Designs. *IEEE Transactions on Software Engineering*, 27 (7), 651–672, 2001.
- [48] Jung, J. J. Ontological framework based on contextual mediation for collaborative information retrieval. *Information Retrieval*, 10 (1), 85–109, 2007.
- [49] Lancaster, F. W. *Information Retrieval Systems: Characteristics, Testing and Evaluation*, Wiley, New York, 1968.
- [50] van Rijsbergen, C. J. *Information Retrieval*, 2nd edition, University of Glasgow, London, Butterworths, 1979.
- [51] De Lucia, A., Fasano, F., Oliveto, R., Tortora, G. Enhancing an Artifact Management System with Traceability Recovery Features. In: Proceedings of the 20th International Conference on Software Maintenance, pp. 306–315, 2004.
- [52] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41, 391–407, 1990.
- [53] Deng, M., Stirewalt, R.E., Cheng, B.H. Retrieval By Construction: A Traceability Technique to Support Verification and Validation of UML Formalizations. *International Journal of Software Engineering and Knowledge Engineering*–(IJSEKE), 15 (5), 837–872, 2005.
- [54] Hayes, J. H., Dekhtyar, A., Osborne, E. Improving Requirements Tracing via Information Retrieval. In: Proceedings of the IEEE Requirements Engineering Conference, pp. 138–150, 2003.
- [55] Marcus, A., Maletic, J. Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing. In: Proceedings of the IEEE International Conference on Software Engineering, pp. 125–132, 2003.

- [56] Spanoudakis, G. Plausible and Adaptive Requirements Traceability Structures. In: Proceedings of the 14th ACM International Conference on Software Engineering and Knowledge Engineering, pp. 135–142, 2002.
- [57] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, 28 (10), 970–983, 2002.
- [58] Zisman, A., Spanoudakis, G., Péres-Miñana, E., Krause, P. Tracing software engineering artifacts. In: Proceedings of the 2003 International Conference on Software Engineering Research and Practice (SERP'03), pp. 448–455, 2003.
- [59] Murphy, G. C., Notkin, D. Sullivan, K. J. Software reflexion models: Bridging the gap between design and implementation. *IEEE Transactions on Software Engineering*, 27(4), 364–380, 2001.
- [60] Richardson, J., Green, J., 2004. Automating traceability for generated software artifacts. In: Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE'04), pp. 24–33.
- [61] Cleland-Huang, J., Chang, C.K., Hu, H., Javvaji, K., Sethi, G., Xia, J., 2002. Requirements Driven Impact Analysis of System Performance. In: Proceedings of the IEEE Joint Conference on Requeriments Engineering, pp. 289–296.
- [62] Cleland-Huang, J., Settimi, R., Lukasik, W., Chen, Y. Dynamic Retrieval of Impacted Software Artifacts. In: Proceedings of the Midwest Software Engineering Conference, 2003.
- [63] Gross, D., Yu, E. From Non-Functional Requirements to Design through Patterns. *Requirements Engineering Journal*, 6 (1), 18–36, 2001.
- [64] Cleland-Huang, J., Schmelzer, D. Dynamically Tracing Non-Functional Requirements through Design Pattern Invariants. In: Proceedings of the Workshop on Traceability in Emerging Forms of Software Engineering, 2003.
- [65] Mellor, S. J., Clark, A. N., Futagami, T. Model-Driven Development. *IEEE SOFTWARE*, 20(5), 14–18, 2003.
- [66] Genvigir, E. C.; Sant'anna, N.; Borrego, F. Modelagem de processos de software através do SPEM – Software Process Engineering Metamodel Specification – Conceitos e Aplicação. In: Workshop dos Cursos de Computação Aplicada do INPE – Instituto Nacional de Pesquisas Espaciais, pp. 85–90, 2003.
- [67] Object Management Group, Software Process Engineering Metamodel Specification (SPEM), Formal Submission, OMG document number formal/05–01–06, January 2005.

# Requirements Traceability

Elias Canhadas Genvigir<sup>a b</sup>  
Nandamudi Lankalapalli Vijaykumar<sup>b</sup>

<sup>a</sup> Federal University of Technology – Paraná – UTFPR. Av. Alberto Carazzai, 1640, CEP: 86300–000, Cornélio Procópio, PR, Brazil.

<sup>b</sup> National Institute for Space Research – INPE. Av. dos Astronautas, 1.758, CEP: 12227–010, São José dos Campos, SP, Brazil.

## Abstract

This chapter presents a research about the Software Requirements Traceability. The main elements of traceability, definitions, problems and prospects are presented. The chapter is organized by topics and its beginning is a review about requirements engineering, its categories (Elicitation, Analysis and Negotiation, Documentation, Validation, and Management) and its role in software development. Afterwards, the requirements management and its elements (Identification, Change Management and Traceability) are described. Traceability is discussed; its aspects and problems are exploited as well as its classifications, techniques, links, metrics and models. Finally the Conclusion presents the main points that can be explored in future researches.

## 1 – Introduction

Software production activity, also known as software development process, has been improving due to the demands in quality increase of those products.

The demand for more quality, cost decrease added to introduction of new technologies naturally forces to process improvement. Those facts are not new and this occurs in several other production areas that employ different techniques to increase their development processes. In particular for software the research area involved with the process improvement is the Software Engineering (Bauer, 1969; IEEE, 1990; IEEE, 2004).

The very first model ever employed in the software development process, that proposed work segmentation, is the waterfall model. It was able to allow a structured overview about the software development and its structure was the foundation stone to propose new processes as new necessities were required.

There are other development models such as prototyping, incremental, evolutionary, and spiral.

It is interesting to observe a common point in all these models: all of them start with Requirements Engineering. Generally speaking, Requirements Engineering is the process to identify stakeholders and their needs, and documenting these in such a way that makes analysis, communication, and subsequent implementation feasible and at the same time leading to a product of quality and reliability (Nuseibeh & Easterbrook, 2000).

The fact that Requirements Engineering being the first activity within the software production process can be better explored by analyzing the following statements:

*“The main goal of Requirements Engineering is to define the purpose of the system proposed and to define its external behavior”* (Antoniou et al., 2000).

*“The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended”* (Nuseibeh & Easterbrook, 2000).

The first statement refers to the definition of purpose, which is an inherent characteristic during the initial activities both in projects and research methodologies (Basili et al., 1994; Solingen & Berghout, 1999).

The second refers to meet the intended purpose. If a software development's basis is to come up with a solution to a given problem, then it is of a fundamental importance the knowledge of the exact dimension of the problem to be solved. Requirements are initially discovered in projects and to understand them is a basic condition for the next phases.

Thus, one can get an idea of the importance with respect to requirements and why the Requirements Engineering occurs in the initial phases within the development process.

Requirements engineering activities can be divided into five categories: Elicitation, Analysis and Negotiation, Documentation, Validation, and Management. These definitions alternate among different authors (Jarke & Pohl, 1994; Sawyer et al., 1998; Graham, 1997; Kotonya & Sommerville, 2000; Sommerville, 1995) but the concepts are similar:

- Elicitation – It is the activity that involves the discovery of the requirements of the system. In this activity the system developers work together with customers and users to define the problem to be solved, the services that the system should provide, the system performance requirements, the characteristics of hardware and other resources inherent in the system. There are three levels of requirements elicitation: business, user, and functional.
- Analysis and Negotiation – The customers and users needs must be in compliance with the definitions of software requirements. This activity is used to analyze in detail the requirements and resolve possible conflicts between those involved with the system.
- Documentation – Requirements need to be documented to serve as a basis for the remainder of the development process. This documentation must be made in a consistent way and must follow some standards to demonstrate, in several levels of detail, the requirements specification.
- Validation – This activity ensures that the software requirements specification is in compliance with the system requirements. This means that the requirements will be checked to assess whether they have acceptable representation and description beyond the analysis of properties as correctness, completeness, consistency and feasibility.
- Management – The Requirements management activity should assist the requirements evolution maintenance through the process development covering all processes involved in changes in the system requirements

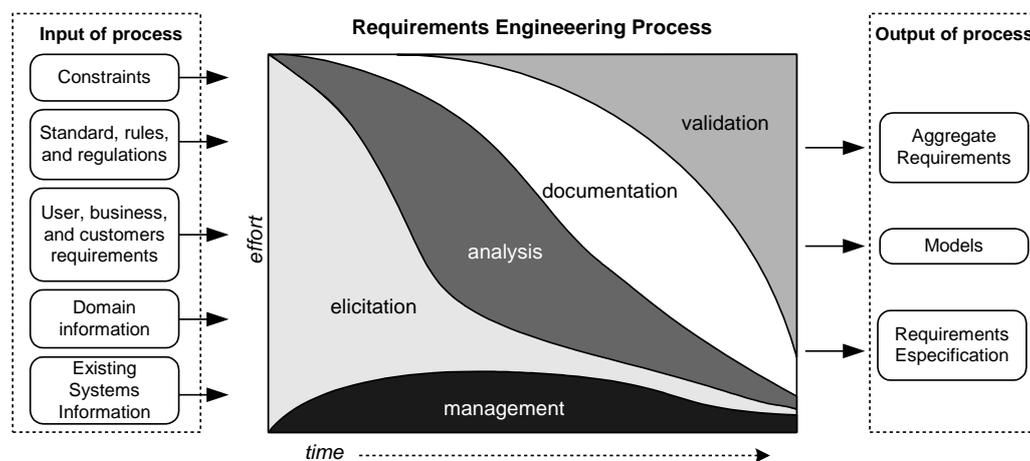
## 2 Requirements Management

One of the major difficulties for the requirements engineering is the control and the aggregation of new system requirements. This occurs due to the impact of proposed changes, the process inflexibility and the difficulty in providing support to apply these changes.

Requirements Engineering aims to solve such problems. Its main concerns are managing changes to agreed requirements, managing the relationships between related requirements, and managing dependencies between the requirement document and other documents produced during other software engineering processes (Sommerville & Sawyer, 1998).

Leffngwell and Widrig (2000) define the requirements management as a systematic approach to elicitation, organization, and documentation of a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system. Though a difficult task, the requirements change is seen as something normal and inevitable, an inherent fact of software.

So, requirements management can be seen as a parallel process to support other requirements engineering activities (Sommerville & Sawyer, 1998; Kotonya & Sommerville, 2000) taking place even after the specification phase. Requirements engineering process and the requirements management activity, occurring during the process, can be seen in Figure 1.



**Figure 1**– Requirements Engineering Process (Based in: NASA, 1990; Kotonya & Sommerville, 2000).

Requirements management can be divided into three specific areas:

- a) Identification – Activities related to identification and storage requirements. Identification practices focus on the assignment of a unique identifier for each requirement and basically there are 4 techniques for its implementation: Natural identification scheme for hierarchical requirements, that identifies offspring requirements following parent requirement's identification (Leffingwell & Widrig, 2000); Dynamic renumbering, which allows automatic interdocument renumbering of paragraphs when inserting information using word processing systems and cross references (Kotonya & Sommerville, 2000); Database record identification, in which each requirement added into a

database is considered as an entity so that it can be uniquely identified, versioned, referenced and managed (Kotonya & Sommerville, 2000); Symbolic identification, requirements are identified using symbolic names related to the contents of the requirement for example, EFF-1, EFF-2.

- b) Change management – Activities related to requirements changes as impact analysis, communication, stability measurement and incorporation of elements in the system.

System requirements are changeable during the entire life cycle and system developers must continuously and carefully manage the changes of system requirements to provide a useful and satisfying system for their customers at a reasonable cost and time (Kobayashi & Maekawa, 2001). Leffingwell and Widrig (2000) introduce five guidelines to manage changes: 1) Recognize that changes are inevitable, and plan for their implementation, 2) Baseline the requirements, 3) Establish a single channel to control changes, 4) Use a change control system to capture changes, 5) Manage changes hierarchically. In addition to general guidelines there are several authors that define methods to managing changes in requirements as Crnkovic et al. (1999) proposing support to changes management during the product life cycle; Lindsay et al. (1997) emphasize the fine granularity; Lam et al. (1999) introduce metrics for use in support, in this case the main management activity is the measurement; and Kobayashi and Maekawa (2001) who created the model NRM (Need-based Requirements Management) that consisted of objects concerned with specifying system requirements, e.g., needs, goals, actions, stakeholders, systems, behaviors, etc., and relationships among them analyzing these feature based on the following four aspects (Where, Who, Why, What).

- c) Traceability – Traceability activities as definition of links between requirements and other artifacts produced in the development process. Traceability is the main goal of this work and it will be thoroughly explored below.

### **3 Traceability**

Requirements traceability begins at elicitation advancing through the system development until the maintenance and it is the main activity of the requirements engineering (Nuseibeh & Easterbrook, 2000).

Traceability is intimately associated with producing software artifacts related to requirements, and relationships among these requirements. In fact, requirements traceability ensures a continuous agreement between the stakeholders and the artifacts produced during the development process of the software Letelier (2002). Besides, it enables a clear understanding of the relationships among software requirements, design and implementation. Software designer can be guided to identify which elements satisfy the requirements and those that do not satisfy (Palmer, 2000).

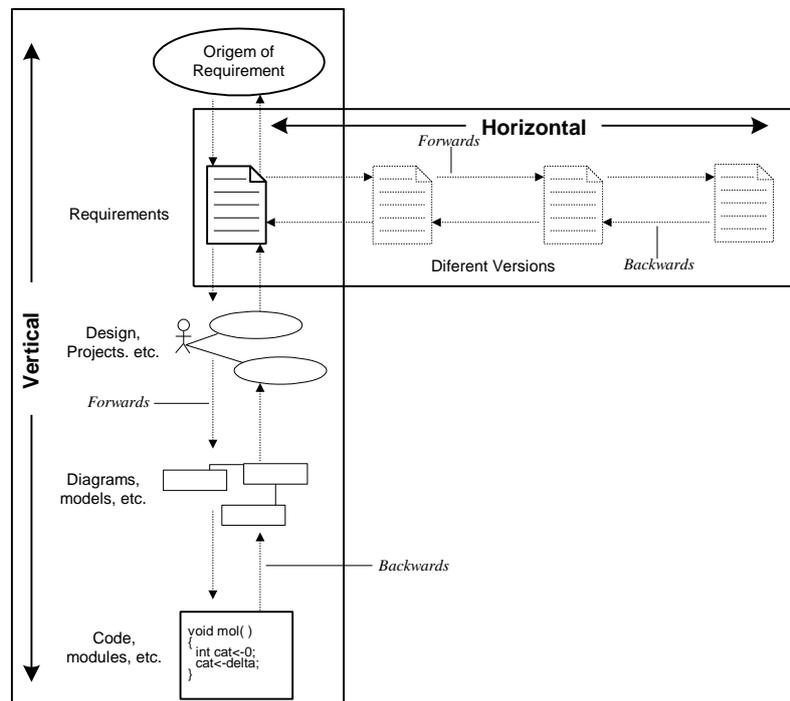
Traceability can also aid in identifying variations in schedules as well as project costs (Palmer, 2000; Davis, 1993; Dömges & Pohl, 1998; Pinheiro, 2003). It is also very useful to project managers to verify whether requirements were observed within the software components, resolving conflicts among requirements, verify requirements in the test process and other issues (Sayão & Leite, 2004).

Depending on its semantics, traceability may be used to: (a) assist in verifying requirements; (b) establish the effect of changes in the requirement specification by means of artifacts or documentation (e.g. design, test and implementation); (c) understand the evolution of an artifact; and (d) understand the aspects of design and support of Rationales (Tang et al., 2007). Thus, creating and maintaining such relationships may provide an efficient basis to guarantee software quality, dealing with changes and software maintenance (Spanoudakis et al., 2004).

### 3.1 Classifications of Traceability

The capacity to trace a requirement from its origin up to all its refinements is defined as Forward Tracing, and the capacity to trace refinements from requirements is defined as Backward Tracing (Davis, 1993). These two capacities should be inserted in every type of traceability, and it is a basic property to conduct a complete traceability. A traceability process is considered as a failure if it does not execute both capacities. Traceability can be: a) horizontal and vertical; and b) pre and post-traceability.

Horizontal traceability is a traceability between versions and variants of a requirement, or other artifacts, within a particular phase of its life. Vertical traceability illustrates traceability between requirements and artifacts produced during the process development throughout the project life cycle (Belford et al., 1976; Ramesh & Edwards, 1993; Gotel, 1995). A simplified distinction between horizontal, vertical, forward, and backward traceability is shown in Figure 2.



**Figure2** – Horizontal and Vertical Traceability (Adapted from: Gotel, 1995).

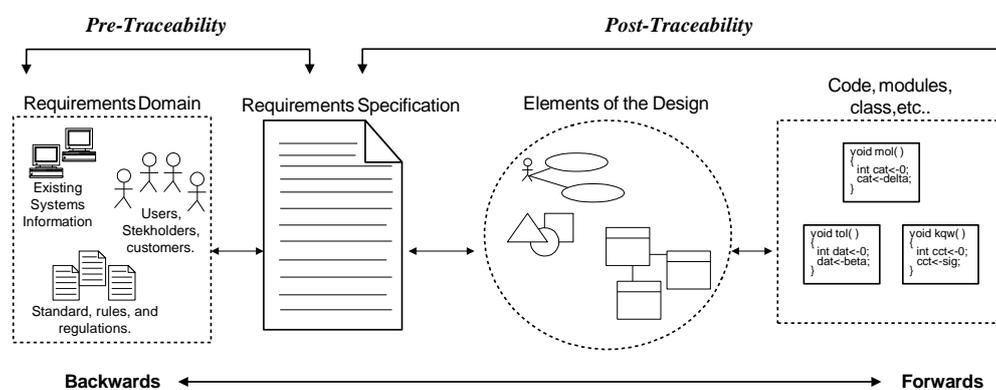
Kotonya and Sommerville (2000) emphasize the direction of traceability and further extend vertical traceability with 'to' and 'from' attributes as well as with traceability between requirements and other artifacts. This classification describes bi-directional

nature of traceability (extended by (Kotonya & Sommerville, 2000) base on (Davis, 1993)):

- Forward-to traceability: traceability of sources (customer requirements, system level requirements, etc) to requirements.
- Forward-from traceability: traceability of requirements to design specifications.
- Backward-to traceability: traceability of design specifications to requirements.
- Backward-from traceability: traceability of requirements to their sources (customer requirements, system level requirements, etc)

The second type of traceability deals with pre-traceability, which is concentrated in the life cycle of requirements before they are included in the requirements specification, and post-traceability, concentrates in the life cycle of requirements after being included in the requirements specification (Gotel & Finkelstein, 1994).

Figure 3 shows the pre and post-traceability, it is possible to observe that requirements knowledge is distributed and merged in successive representations.



**Figure 3** – Pre and Post-traceability (Adapted from Gotel, 1995).

Gotel and Finkelstein (1994) distinguish Pre and Post-Traceability. Although both these types of traceability are needed it is crucial to understand their subtle differences, as each type imposes its own distinct requirements on potential support.

The main differences, between Pre and Post-Traceability, involve the information they deal with and the problems they can assist.

Post-traceability depends on the ability to trace requirements from, and back to, a baseline (requirements specification) (Figure 3), through a succession of artifacts in which they are distributed. Changes to the baseline need to be re-propagated through this chain (Gotel & Finkelstein, 1994).

Pre-Traceability depends on the ability to trace requirements from, and back to, their originating statement(s), through the process of requirements production and refinement, in which statements from diverse sources are eventually integrated into a single requirement in the requirements specification. Changes in the process need to be re-worked into the requirements specification. Changes to the requirements specification need to be carried out with reference to this process, so they can be instigated and propagated from their source. This requires visibility of the subtle interrelationships that exist between initial requirements (Gotel & Finkelstein, 1994).

### **3.2 Techniques of Traceability**

The establishing, the maintenance and the relationships representation, that are present in different traceability types, are conducted through some traceability technique. The use and development of these techniques originated in the early 70s (Aizenbud–Reshef et al., 2006), and the first method used to express and maintain traceability was cross–referencing (Evans, 1989).

Other techniques can be used both to represent and to establish relationships including standard approaches such as matrices (Davis, 1993; West, 1991); keyphrase based scheme (Jackson, 1991); document integration (Lefering, 1991); hypertext links (Alexander, 2002; Glinz, 2000; Kaindle, 1993); graphs (Pinheiro & Goguen, 1996); formal methods (Cleland–Huang & Christensen, 2003; Antoniol et al., 2000; Tryggeseth & Nytrø, 1997); dynamic schemes (Cleland–Huang & Christensen, 2003; Antoniol et al., 2000; Tryggeseth & Nytrø, 1997); assumption based truth maintenance system (Tang, 1997; Smithers et al., 1991); and constraint networks (Bowen et al., 1990).

Automated or semi–automated traceability techniques had also been developed, such as the detection of existence of links using machine learning (Faisal, 2005); automatic analysis of consistency in low–degree polynomial time (Chechik & Gannon, 2001); techniques for Information Retrieval (Jung, 2007; Lancaster, 1968; van Rijsbergen, 1979) as the Latent Semantic Indexing – LSI (De Lucia et al., 2004; Deerwester et al., 1990) that may be used to recover relationship links by means of formalizing semantics (Deng et al., 2005).

Recent research show that techniques based on Information Retrieval can be used to dynamically discover links (Antoniol et al., 2000; Hayes et al., 2003; Marcus & Maletic, 2003; Spanoudakis, 2002), besides allowing the relationship between several types of artifacts (Deng et al., 2005) such as source code and documentation (Marcus & Maletic, 2003; Antoniol et al., 2002), design and requirements (Hayes et al., 2003), and requirements and artifacts (Zisman et al., 2003).

A popular technique, known as Software Reflexion Model (Murphy et al., 2001) checks whether the implemented code conforms with the project model using mapping rules between the project model and the source model extracted from the implemented code. Another research in this same line belongs to Richardson and Green (2004) that uses an automated synthesis to infer links.

Besides traceability between artifacts, some other techniques such as Event–based traceability – EBT (Cleland–Huang & Christensen, 2003) can be used to associate performance requirements (Cleland–Huang & Christensen, 2003; Cleland–Huang et al., 2002; Cleland–Huang et al., 2003), as well as to trace the quality of requirements based on well known design patterns (Gross & Yu, 2001), that can be related and linked using their invariants (Cleland–Huang & Schmelzer, 2003).

### **3.3 Traceability links**

The main resource used to maintain and represent the traceability relationships is the link.

Several proposals have been made for traceability links as well as for models that provide support: (i) to some standardization focusing on maintenance process and (ii) to represent such links.

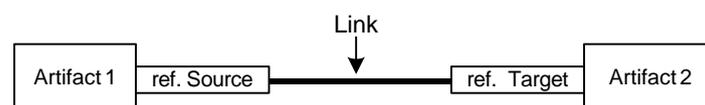
Basically, links are associated towards some methodological information of the development process. In that aspect, links were created for object-oriented (Spanoudakis et al., 2004), aspect-oriented (Egyed, 2004), viewpoints-based development (Sabetzadeh & Easterbrook, 2005), or model-driven development (Aizenbud-Reshef et al., 2006; Almeida et al., 2006). Besides of links can be created for environmental and organizational aspects (Ramesh & Jarke, 2001); on the type of information to be traced (Toranzo et al., 2002); or on the configuration integrating textual specifications (Letelier, 2002). The drawback in these is that links are created and used for specific purposes only.

In the case of Requirements Engineering, links were used in the following activities: validation, analysis, evolution, and cross-referencing between requirements and artifacts (Palmer, 2000). Also the following processes make use of this resource: project management (to assist the prediction of cost, restrictions or impacts); maintenance; test (generation of test cases based on scenarios or models); and quality assurance process (software validation) (Cleland-Huang & Christensen, 2003; Pinheiro & Goguen, 1996; Palmer, 2000; Gotel & Finkelstein, 1994; Jarke, 1998; Papadopoulou, 2002; Riebisch & Hubner, 2005).

Links can be formalized as follows:  $A=\{a_1,a_2,a_3,\dots,a_n\}$  represents all the artifacts produced and identified by the development process;  $E=\{e_1,e_2,e_3,\dots,e_n\}\subset A$  represents links; and  $R=\{r_1,r_2,r_3,\dots,r_n\}\subset A$  represents requirements.

A link establishes an explicit relationship between two artifacts  $a_1$  and  $a_2$ .  $a_1 \rightarrow a_2$  are considered as directly linked while an indirect link uses an intermediate artifact such as in  $a_1 \rightarrow a_2 \rightarrow a_3$  (Cleland-Huang et al., 2003).

Information about source and target artifacts is enough to support backward and forward traceability, Figure 4, but other requirements engineering activities such as schedules and conflict resolution may require other attributes for their execution (Dick, 2002; Dick, 2005).



**Figure 4** – Basic elements of a link.

Several categories of links may be determined by means of attributes and properties or even through application of these links within the development process. Literature has already suggested some types of links to be used in traceability.

Ramesh and Jarke (2001) define four types of links: Satisfies, Depends-on, Involves-to and Rationales. Satisfies and Depends-on are a group called product related, this set describe properties and relationships of design objects independent of how they were created, while Involves-to and Rationales are a group called process related because it can be captured only by looking at the history of actions taken in the process itself and cannot be recovered from the product relationships alone.

Toranzo et al. (2002) consider six type of links: Satisfaction – it indicates that the origin element has dependence on satisfaction with a target class; Resource – the origin class has dependence on the resource with the target class; Responsibility – it determines participation, responsibility or action of participants on generated elements; Representation – it registers the representation or modeling of requirements in other languages; Allocation – it represents the relationship between the origin and

target class, being this last class a subsystem; and Aggregation – it represents the composition between elements.

Pohl (1996), in his dependence model, defines eighteen different links of dependence (Dahlstedt & Persson, 2003) they are categorized in five classes: Condition; Content, Documents, Evolutionary and Abstraction.

De Lucia et al. (2005) present three types of links: Dependence – a directed link between a master and a slave artifact, the slave artifact depends on or is impacted by changes to the master artifact, Undirected – artifacts impact on each other, and Composition – the master artifact is part of the slave artifact.

Spanoudakis et al. (2004) based on studies of the software systems documentation, identified four types of relations between standards of requirements, use cases and object-oriented analysis models: Overlap relations – it denotes that the connected elements refer to a common feature of the underlying system or its domain. Thus, a relation of this type expresses the potential of a dependency between these elements; Requires\_Execution\_Of – it denotes that the sequence of terms requires the execution of the operation to which it is related; Requires\_Feature\_In – It denotes that the relevant part of the use case cannot be conducted without the existence of the structural or functional feature required by the requirement statement or the relation denotes that one of the requirement statements refers to the existence of a feature of a system that is required by the other; Can\_Partially\_Realize relations – The meaning of the relation is that the execution of the use case can execute part of the requirement statement.

Aizenbud-Reshef et al. (2006) define four types of relationships: Imposed – is a relationship between artifacts that exists due to a violation of the relationship creator; Inferred – is a relationship that exists because the artifacts satisfy a rule that describes the relationship; Manual – is a relationship that is created and maintained by a human user, this relationship can be either imposed or inferred; and Computed relationship – is created and maintained automatically by a computer. There are two basic types of computed relationships: Derivation – this relationship denotes that, given the content of an artifact, it is possible to compute valid content for the related artifact; and Analysis – it is a type of inferred relationship created by analysis programs that examine code or models against a set of rules.

A summary of the link types is presented in Table 1.

**Table 1** – Examples of links described in the literature, they are classified by author and link groups.

Author	Groups	Link type
Ramesh and Jarke (2001)	Product Related	Satisfies
		Depends-on
	Process Related	Envolves-to
		Rationales
Toranzo et al. (2002)		Satisfaction Resource Responsibility Representation
Pohl (1996)	Condition	Constraints
		Precondition
	Documents	Example_for
		Purpose
		Test_Case_for
		Comments
		Background
	Abstraction	Refines
		Generalizes
	Evolutionary	Elaborates
		Formalizes
		Based_on
		Satisfies
		Replaces
	Content	Similar
		Compares
Contradicts		
Conflicts		
De Lucia et al. (2005)		Dependence
		Undirected
		Composition
Spanoudakis et al. (2004)		Overlap
		Requires_Execution_Of
		Requires_Feature_In
		Can Partially Realise
Aizenbud-Reshef et al. (2006)		Imposed
		Inferred
		Manual
	Computed	Derivation
		Analysis

### 3.4 Traceability metrics

Software metrics can provide objective information necessary for technical and managerial insight into, control, and improvement of the development effort (Costello

& Liu, 1995). They should provide quantitative information to support those tasks (Norman & Martin, 2000) and they can be used to improve the productivity and the quality of both product and process (SEI, 1988).

According to (Roche, 1994) software metrics are inserted in the software measurement context and they cover topics such as:

- Performance evaluation and models
- Algorithmic/computational complexity
- Reliability models
- Cost and effort estimation
- Quality models and measures
- Structural and complexity measures
- Design measures
- Productivity measures and models

Metrics for traceability use data on links and other techniques of representation to show how an organization maintains the measurement on the requirement relationships. The information obtained can determine whether all the relationships required by a dependency are met, setting specifications incomplete or too complex for the system, and can also be used to prevent incorrect interpretations of other metrics. Moreover, they can expose requirements leakage that is, the appearance of lower level requirements with no valid origin at the system level (Costello & Liu, 1995).

Costello and Liu (1995) define five types of traceability metrics: next-level coverage (COV) metrics, full depth and height coverage (DHCOV) metrics, linkage statistics, inconsistent traceability metrics (ITM), and undefined traceability metrics (UTM).

COV metrics include the number of requirements that trace consistently to the next level up (COVup), the number of requirements that trace consistently to the next level down (COVdown), and the number of requirements that trace consistently to the next level in both directions (COV). DHCOV metrics involve a similar analysis, but evaluate traceability to the highest (HCOV) and lowest (DCOV) levels of specification. Note that for systems with three or fewer levels of specification, COV and DHCOV metrics are identical. Linkage statistics measure the complexity of traceability data by providing amounts of the number of higher/lower level requirements to which each requirement in a specification is traced. ITM include numbers of requirements in a specification that have at least one inconsistent traceability link upward (ITMup) and downward (ITMdown). Finally, UTM include the number of requirements in a specification with no traceability links upward (UTMup) and downward (UTMdown) (Costello & Liu, 1995).

Hull et al. (2002) also propose a discussion about traceability metrics. Based on the satisfaction relationship type, and moving down through the requirements, the authors define three dimensions and two other elements for the measurement of traceability.

To determine which aspects of these dimensions are useful in terms of measuring the requirements engineering process, it is necessary to distinguish between two types of indicators: i) Layer metrics: measurements relating to a single stage of development, e.g. just to the systems requirements layer; ii) Global metrics: measurements spanning several stages of development (Hull et al., 2002).

The three dimensions and the other two elements are presented below:

- Dimensions:
  - a) Breadth – It relates to coverage, and as such it is a layer metric. The coverage can be used to measure progress of process that creates traceability at a single layer. It measures the extent to which requirements are covered by the adjacent layer above or below.
  - b) Depth – It looks at the number of layers that traceability extends upwards or downwards from a given layer, making it a global metric.
  - c) Growth – It is related to potential change impact. How many requirements at lower levels are related to a single requirement at the top level?
- Other elements:
  - a) Balance – Its idea is to look at the distribution of Growth factors for individual requirements between two given layers, and examine those that lie in the outer quartiles of the distribution. The goal is to identify requirements that have an abnormal high or low Growth factor, and subject them to special scrutiny.
  - b) Latent Change – A single change request, may suddenly introduce a chain of potential latent change into the system. In such circumstances, it would be highly desirable to track progress and estimate the consequential work.

Traceability metrics presented aims to facilitate the improvement of the quality of the traced requirements, which is very appropriate. But a more critical eye on the metrics in place, can point to lack of concern about the quality of links, which may be useful for other activities of the requirements engineering.

All proposals of traceability metrics gather information to determine the requirements quality, while the traceability quality itself is not evaluated.

The standards for software quality follow a pattern of refinement where the higher level elements of the development process are refined into smaller elements. An example of this can be seen in the standards ISO/IEC – 12207 (ISO/IEC, 1995), which is about the life cycle of the development process, in IEEE – 830 (IEEE, 1998), dealing with specification of requirements of software, and the IEEE – 1233 (IEEE, 1998b), addressing the development of the specification of requirements for systems.

The same pattern of refinement does not occur with the traceability, i.e., link which is the main element of maintenance and representation of relationships, used in the lowest level of the process of requirements management, is not explored in quality factor but only used as a tool to assess the quality of elements in higher levels.

Despite being used by several software engineering processes there is still a lack for clear standards for the traceability quality. That characteristic may be observed in the standards CMMI–DEV (SEI, 2006) that points to a specific practice: 1.4 Maintain Bidirectional Traceability of Requirements, or ISO/IEC 15504–2 (ISO/IEC, 2003) which issues basic practices: ENG.3.4 – Establish traceability and PRO.4.5 – Maintain traceability. As stated by Angelina et al. (2006) the standards and norms do not include traceability metric to ensure the consistency, completeness, reliability, usability and efficiency of the implementation of traceability.

### **3.5 Models for Traceability**

Another important research area for the traceability are models. According to Mellor et al. (2003) a model is a coherent set of formal elements describing something built for some purpose that is subject to a particular form of analysis. It may be expressed

by a language (either textual or graphical) in which a certain degree of abstraction can be achieved.

In the case of traceability, models are developed based on information with respect to traceability domain (users, practices, methodologies, norms, etc.).

Several authors use models to represent concepts related to traceability such as the definition of different types of links.

Ramesh and Jarke (2001) proposed a metamodel for traceability using as a basis extensive empirical research about the consequences of different uses, perspectives and information needs of those involved in the development process.

They found three categories of information associated with traceability: stakeholders; sources (standards, norms, rules, etc.); and objects (conceptual objects, models, or artifacts), which were used to establish the metamodel.

Besides these three categories the research enabled the authors to classify users of traceability. The users were classified with respect to their practices, in two separate groups: high-end and low-end users. Based on this classification two models were created to represent their traceability processes.

In addition to creating a conceptually simple but very broad metamodel in terms of elements and links, research by Ramesh and Jarke (2001) presents a very interesting classification on users of traceability, assigning a significant figure to the practices employed by the users within their organizations. But, this model also pre-establishes the types of links to be used, even when these pre-established types of links can be divided and used as reference. Another problem found in the model, proposed by Ramesh and Jarke (2001), is that it does not have a description of attributes that can be added to enhance the semantics of the link, despite the existence of the important type of link Rationale.

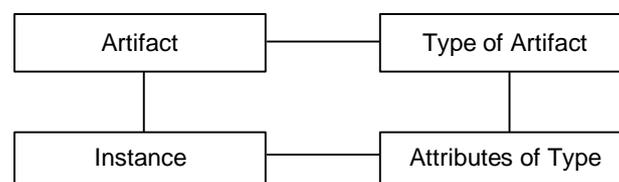
Model proposed by Toranzo et al. (2002) uses graphical representation based on UML. This model classifies information to be traced into four classes: Environmental, Organizational, Managerial, and Development. Based on this classification, the authors developed the following: a meta-model; an intermediate model; and a process to apply requirements traceability strategies. The Toranzo et al. (2002) model addresses management issues such as the link that belongs to the category of responsibility. This is noted by the influence of the defined classification context (Environmental, Organizational, Managerial and Development). However, the types of links are also pre-defined and addressed in the context used in the model. Other management and traceability standards, as addressed to the vision may not be completely addressed to express their links.

A discussion on the problem of pre-specification of links in the traceability models is found in (Aizenbud-Reshef et al., 2006). The authors propose a solution to this problem, using the concept of Model-Driven Architecture MDA. This proposal considers that creation of models for systems must include requirements metamodel, and the resulting models, which include a requirements explicit model, can provide traceability between requirements and other artifacts. The existence of these models does not provide mechanisms to produce, automatically, the links between requirements and its dependencies, i.e., due to the fact of possessing semantics to produce significant links does not guarantee an effective strategy for traceability. Thus the links need to be constructed and maintained manually.

According to Aizenbud–Reshef et al. (2006) the idea would generate the artifacts, and its basic structures, and produce the links between them, and if necessary the links would be detailed. In this case MDA could be an alternative to accomplish these tasks.

Despite the observation about the traceability models which have pre–defined types of links Aizenbud–Reshef et al. (2006) proposed an architecture–based solution, which may turn into a problem for developing processes based on other concepts. However this proposal can be very relevant to those involved with the MDA.

An alternative to the problem of pre–definition of the types of links can be a model based on the generalization of all kinds of artifacts that may be part of the process of traceability, focusing on the representation of information. This feature allows both the creation of new types of bonds and other artifacts, as the need of those involved in traceability. This model is represented in Figure 5.



**Figure 5** – Model for link generalization (Genvigir and Vijaykumar, 2008)

The idea is to include all types of artifacts into the traceability process focusing mostly on representing information. This feature promotes creation of both new types of links and artifacts according to the necessity of the involved elements of traceability. Two particular cases justify this generalization. The first one is with respect to aggregation of new fields to be incorporated into a certain type of link or artifact. The second enables insertion of new types of artifacts.

For the first case, the links, for most of the existing applications, are just a connection between two artifacts where information is not aggregated to this connection. Therefore, it is essential to enable the association of information of the process (quality, rationale) to the link. These information should be represented in the model.

The second case consists in the necessity of including new types of links or other artifacts during an ongoing project or for a new project. Definition of types of links right in the beginning of a project does not reflect the reality in which a project might face changes or adjustments during its development.

#### **4 Conclusion**

The correct traceability use can help a lot of activities within the development process. It can improve the quality of the product as well, as the software process.

In spite of the traceability importance some of its aspects can be explored in future research.

With respect to traceability models, presented in this work, it is possible to see that they provide pre–defined standards of groups of links and those groups can be supported by their models. Research does extensive observations about the traceability practices, but their models are limited as the types of links are fixed. Then, the models are defined to cover a specific approach for solving their problem domain (Aizenbud–Reshef et al., 2006).

There is a perspective of a better understanding of the explicit relationships between models, modelers and what is modeled will be the basis for further work in the context of current requirements engineering research and practice. This concept will be able to lead to a proper understanding of how traceability properties may be identified and integrated (Morris & Gotel, 2007).

The link description can be better exploited. The link types, in most cases, only represent the connection between two elements of the development process. These models do not enable inclusion of entities that define properties of the links, called attributes, or yet of the richer semantics. This means that essential information to establish analysis, impact evaluation, derivation or coverage about the traceability are not at all supported or even represented by such models (Dick, 2002; Dick, 2005; Hull et al., 2002).

Finally, it is essential to improve the existing standards and to develop new metrics for the traceability. These elements can contribute to ensuring consistency, completeness, reliability, usability and efficiency for the traceability. These topics remain open and deserve a better discussion and demonstration of feasibility.

## 5 References

- Aizenbud–Reshef, N., Nolan, B. T., Rubin, J., & Shaham–Gafni, Y. (2006). Model traceability. *IBM Systems Journal*, 45 (3), 515–526.
- Alexander, I. (2002). Toward automatic traceability in industrial practice. In *International Workshop on Traceability in Emerging Forms of Software Engineering* (pp. 26–31). Edinburgh, Scotland.
- Almeida, J. P., Eck, P. V., & Iacob, M. (2006). Requirements traceability and transformation conformance in model–driven development. In *International Enterprise Distributed Object Computing Conference* (pp.355–366). Washington, USA: IEEE Computer Society Press.
- Angelina, E., Alarc’on, P. P., & Garbajosa J. (2006). Analyzing and systematizing current traceability schemas. In *Annual IEEE/NASA Software Engineering Workshop* (pp. 21–32). Los Alamitos, CA, USA: IEEE Computer Society Press.
- Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., & Merlo, E. (2002). Recovering traceability links between code and documentation. *IEEE Transactions on Software Engineering*, 28 (10), 970–983.
- Antoniol, G., Casazza, G., & Cimitile, A. (2000). Traceability recovery by modeling programmer behavior. In *Working Conference Reverse Engineering* (pp. 240–247). Washington, DC, USA: IEEE Computer Society Press.
- Antoniou, G., Macnish, C., & Foo, N. Y. (2000). A note on the refinement of nonmonotonic knowledge bases. *Knowledge and Information Systems*, 2, 479–486.
- Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of Software Engineering*, 1(4), pp. 528–532.
- Bauer, F. *Software Engineering: Report on a conference sponsored by the NATO Science committee*. Garmisch, Germany, Jan. 1969, pp. 231.
- Belford, P. C., Bond, A. F., Henderson, D. G., & Sellers, L. S. (1976). Specifications a key to effective software development. In *International Conference on Software Engineering* (pp. 71–79). Los Alamitos, CA, USA: IEEE Computer Society Press.

- 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. *IEEE Transactions on Software Engineering*, 27 (7), 651–672.
- Cleland-Huang, J., Chang, C. K., Hu, H., Javvaji, K., Sethi, G., & Xia, J. (2002). Automating speculative queries through event-based requirements traceability. In *IEEE Joint Conference on Requirements Engineering* (pp. 289–296). Los Alamitos, CA, USA: IEEE Computer Society.
- Cleland-Huang, J., & Christensen, C. (2003). Event-based traceability for managing evolutionary change. *IEEE Transactions on Software Engineering*, 29 (9), 796–810.
- 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*. Montreal, Canada.
- Cleland-Huang, J., Settimi, R., Lukasik, W., & Chen, Y. (2003). Dynamic retrieval of impacted software artifacts. In *Midwest Software Engineering Conference*.
- Costello, R., & Liu, D. (1995). Metrics for requirements engineering. *Journal of Systems and Software*, 1(72), 39–63.
- Crnkovic, I., Funk, P., & Larsson, M. (1999). Processing requirements by software configuration management. In *25th Euromicro Conference* (pp. 260–265).
- Dahlstedt, Å. G., & Persson, A. (2003). Requirements interdependencies – moulding the state of research into a research agenda. In *International Workshop on Requirements Engineering: Foundation for Software Quality* (pp. 71–80).
- Davis, A. M. (1993). *Software requirements: objects, functions and states*. New Jersey: Prentice-Hall.
- De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2004). Enhancing an artifact management system with traceability recovery features. In *International Conference on Software Maintenance* (pp. 306–315).
- De Lucia, A., Fasano, F., Oliveto, R., & Tortora, G. (2005). ADAMS re-trace: a traceability recovery tool. In *European Conference on Software Maintenance and Reengineering* (pp. 32–41).
- Deerwester, S., Dumais, S. T., Furnas, G. W., K., T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(1), 391–407.
- Deng, M., Stirewalt, R. E., & Cheng, B. H. (2005). Retrieval by construction: A traceability technique to support verification and validation of UML formalizations. *Journal of Software Engineering and Knowledge Engineering – IJSEKE*, 15 (5), 837–872.
- Dick, J. (2002). Rich traceability. In *International Workshop on Traceability In Emerging Forms of Software Engineering* (pp. 35–46).
- Dick, J. (2005). Design traceability. *IEEE Software*, 22(6), 14–16.

- Dömges, R., Pohl, K. (1998). Adapting traceability environments to project-specific needs. *Communications of the ACM*, 41(12), 54–62.
- Egyed, A. (2004). Resolving uncertainties during trace analysis. In *Symposium on Foundations of Software Engineering* (pp. 3–12).
- Evans, M. W. (1989). *The Software Factory: A Fourth Generation Software Engineering Environment*. Wiley-Interscience.
- Faisal, M. H. (2005). *Toward automating the discovery of traceability links*. Doctoral Thesis. University of Colorado.
- Genvigir, E. C., & Vijaykumar, N. L. (2008). A Modeling Proposed for Generalization of Traceability Links. *Revista de Informática Teórica e Aplicada*, XV(2), 181–202 (in Portuguese).
- Glinz, M. (2000). A lightweight approach to consistency of scenarios and class models. In *International Conference on Requirements Engineering* (pp. 49–58). Washington, DC, USA: IEEE Computer Society.
- Gotel, O. C. Z. (1995). *Contribution Structures for Requirements Traceability*. Thesis (Doctor of Philosophy) Faculty of Engineering of the University of London, London.
- Gotel, O. C. Z., & Finkelstein, A. (1994). An analysis of the requirements traceability problem. In *First International Conference Requirements Engineering* (pp. 94–101). Washington, DC, USA: IEEE Computer Society Press.
- Graham, I. (1997). *Requirements engineering and rapid development: an object oriented approach*. Addison Wesley Professional.
- Gross, D., & Yu, E. (2001). From non-functional requirements to design through patterns. *Requirements Engineering Journal*, 6 (1), 18–36.
- Hayes, J. H., Dekhtyar, A., & Osborne, E. (2003). Improving requirements tracing via information retrieval. In *IEEE Requirements Engineering Conference*. (pp. 138–150).
- Hull, E., Jakson, K., & Dick, J. (2002). *Requirements Engineering*. London: Spring Verlag.
- IEEE Computer Society Press (2004). *Guide to the software engineering body of knowledge – SWEBOK: trial version (version 1.00)*.
- Institute of Electrical and Electronics Engineers (1990). *IEEE Standard Glossary of Software Engineering Terminology: IEEE Std – 610.12–1990*. New York, NY, EUA, 1990.
- Institute of Electrical and Electronics Engineers (1998). *IEEE Guide for developing system requirements specifications: IEEE Std – 1233*. New York, NY, EUA.
- Institute of Electrical and Electronics Engineers (1998). *Recommended practice for software requirements specifications: IEEE Std 830–1998*. New York, NY, EUA.
- Institute of Electrical and Electronics Engineers (1990). *Standard Glossary of Software Engineering Terminology: IEEE Std 610.12*. New York, NY, EUA.
- International Organization for Standardization (2003). *ISO/IEC 15504–2:2003/Cor.1:2004(E): Information technology – process assessment – part 2: Performing an assessment*.

- International Organization for Standardization (1995). *ISO/IEC 12207:1995 Information technology - Software life cycle processes*.
- Jackson, J. A. (1991). Keyphrase based traceability scheme. In *Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering) – Tools and Techniques for Maintaining - Traceability During Design* (pp. 2/1–2/4). London: IEE – The Institution of Electrical Engineers.
- Jarke, M. (1998). Requirements traceability. *Communications of ACM*, 41 (12), 32–36.
- Jarke, M., Pohl, K. (1994). Requirements engineering in 2001: managing a changing reality. *Software Engineering Journal*, 9 (6), 257–266.
- Jung, J. J. (2007). Ontological framework based on contextual mediation for collaborative information retrieval. *Information Retrieval*, 10 (1), 85–109.
- Kaindl, K. (1993). The missing link in requirements engineering. *ACM SIGSOFT Software Engineering Notes*, 18 (2), 30–39.
- Kobayashi, A., & Maekawa, M. (2001). Need-based requirements change management. In *Conference on the Future of Software Engineering* (pp. 171–178). Washington, DC, USA: IEEE.
- Kotonya, G., & Sommerville, I. (2000). *Requirements engineering: process and techniques*. Chichester, England: John Wiley & Sons.
- Lam, W., Loomes, M., & Shankararaman, V. (1999). Managing requirements change using metrics and action planning. In *European Conference on Software Maintenance and Reengineering* (pp. 122–128).
- Lancaster, F. W. (1968). *Information Retrieval Systems: Characteristics, Testing and Evaluation*. New York: Wiley.
- Lefering, M. (1993). An incremental integration tool between requirements engineering and programming in the large. In *IEEE International Symposium on Requirements Engineering* (pp. 82–89).
- Leffingwell, D., & Widrig, D. (2000). *Managing Software Requirements: A unified approach*. California, USA: Addison Wesley Longman.
- Letelier, P. (2002). A framework for requirements traceability in UML-based projects. In *International Workshop on Traceability In Emerging Forms of Software Engineering* (pp. 30–41). New York, NY, USA.
- Lindsay, P., Yaowei, L., & Traynor, O. (1997). A generic model for fine grained configuration management including version control and traceability. In *Australian Software Engineering Conference* (pp. 27–36).
- Marcus, A., & Maletic, J. (2003). Recovering documentation-to-source-code traceability links using latent semantic indexing. In *IEEE International Conference on Software Engineering* (pp.125–132). Washington, USA: IEEE Computer Society Press.
- Mellor, S. J., Clark, A. N., & Futagami, T. (2003). Model-driven development. *IEEE Software*, 20 (5), 14–18.

- Morris, S. J., & GoteL, O. C. Z. (2007). Model or mould? A challenge for better traceability. In *International Workshop on Modeling in Software Engineering*. Los Alamitos, CA, USA: IEEE Computer Society Press.
- Murphy, G. C., Notkin, D., & Sullivan, K. J. (2001). Software reflexion models: Bridging the gap between design and implementation. *IEEE Transactions on Software Engineering*, 27 (4), 364–380.
- National Aeronautics and Space Administration. (1990). *Manager's Handbook for Software Development: Software engineering laboratory series –SEL–84–101 – revision 1*. Greenbelt, Maryland, EUA.
- Norman, E. F., & Martin, N. (2000). Software metrics: roadmap. In *Conference on the Future of Software Engineering* (pp. 357–370). New York, NY, USA: ACM Press.
- Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: a roadmap. In *International Conference on Software Engineering* (pp. 35–46). New York, NY, USA: ACM.
- Palmer, J. D. (2000). Traceability. In Thayer, R.H. & Dorfman, M. (Ed.), *Software Requirements Engineering* (pp. 412–422). Los Alamitos, CA, USA: IEEE Computer Society Press.
- Papadopoulou, P. (2002). Evaluation of a Requirements Traceability System. MSc Thesis, Department of Computing, City University.
- Pinheiro, F. (2003). Requirements Traceability. In J. C. S. P. Leite & J. H. Doorn (Eds.), *Perspectives on software requirements* (pp. 91–115). Norwell, MA, USA: Kluwer Academic Publishers.
- Pinheiro, F., & Goguen, J. (1996). An object-oriented tool for tracing requirements. *IEEE Software*, 13 (2), 796–810.
- Pohl, K. (1996). PRO-ART: Enabling requirements pre-traceability. In *International Conference on Requirement Engineering* (pp. 76–84).
- Ramesh, B., & Edwards, M.(1993). Issues in the development of a requirements traceability model. In *International Symposium on Requirements Engineering* (pp. 256–259).
- Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 27 (1), 58–93.
- Richardson, J., & Green, J. (2004). Automating traceability for generated software artifacts. In *IEEE International Conference on Automated Software Engineering* (pp. 24–33). Washington, DC, USA: IEEE Computer Society.
- Riebisch, M., & Hubner, M. (2005). Traceability-driven model refinement for test case generation. In *IEEE International Conference and Workshops on the Engineering of Computer-Based Systems* (pp. 113–120). Washington, DC, USA: IEEE Computer Society.
- Roche, J. M. (1994). Software metrics and measurement principles. *ACM SIGSOFT Software Engineering Notes*, 19 (1), 77–85.
- Sabetzadeh, M., & Easterbrook, S. (2005). Traceability in viewpoint merging: A model management perspective. In *International Workshop on Traceability in Emerging Forms of Software Engineering* (pp. 44–49). New York, NY, USA: ACM.

- Sawyer, P., Sommerville, I., & Viller, S. (1998). Requirements process improvement through the phased introduction of good practice, *Software Process – Improvement and Practice: Requirements engineering adaptation and improvement for safety and dependability – REAIMS (Esprit Project 8649)*. pp. 19–34.
- Sayão, M., & Leite, R. J. (2004). Rastreabilidade de Requisitos. *Revista de Informática Teórica e Aplicada*. 13 (1), 57–86 (in Portuguese).
- Smithers, T., Tang, M. X., & Tomes, N. (1991). The maintenance of design history in AI-based design. In *Colloquium by The Institution of Electrical Engineers Professional Group C1 - Tools and Techniques for Maintaining Traceability During Design* (pp 8/1). London: IEE – The Institution of Electrical Engineers.
- Software Engineering Institute – SEI – Carnegie Mellon University. (2006). *CMMI for Development: Technical report (CMMI–DEV) CMU/SEI–2006–TR–008 Version 1.2*. Pittsburgh, PA, USA.
- Software Engineering Institute – SEI – Carnegie Mellon University. (1988). *Software Metrics: SEI curriculum module SEI–CM–12–1.1*. Pittsburgh, Pennsylvania, EUA, December.
- Solingen, R. V., & Berghout, E. (1999). *The Goal/Question/Metric Method: A practical guide for quality improvement of software development*. Maidenhead, England: McGraw–Hill.
- Sommerville, I. (1995). *Software engineering*. Harlow: Addison Wesley.
- Sommerville, I., & Sawyer, P. (1998). *Requirements engineering: a good practice guide*. Lancaster, UK: John Wiley & Sons.
- Spanoudakis, G. (2002). Plausible and adaptive requirements traceability structures. In *ACM International Conference on Software Engineering and Knowledge Engineering* (pp. 135–142). New York, NY, USA: ACM.
- Spanoudakis, G., Zisman, A., Péres, E. & M., Krause, P. (2004). Rule-based generation of requirements traceability relations. *Journal of Systems and Software*, (72), 105–127.
- Tang, A., Yan, J., & Jun, H. (2007). A rationale-base architecture model for design traceability. *Journal of Systems and Software*, 80 (6), 918–934.
- Tang, M. X. (1997). A knowledge-based architecture for intelligent design support. *The Knowledge Engineering Review*. 12 (4), 387–406.
- Toranzo, M., Castro, J., & Mello, E. (2002). An approach to improve requirements traceability. In *Workshop of Requirements Engineering, (in portuguese)* (pp. 194–209).
- Tryggeseth, E., & Nytrø, O. (1997). Dynamic traceability links supported by a system architecture description. In *International Conference on Software Maintenance* (pp. 180–187). Washington, DC, USA: IEEE.
- van Rijsbergen, C. J. (1979). *Information Retrieval*. London: University of Glasgow.
- West, M. (1991). Quality function deployment in software development. In *Colloquium by The Institution of Electrical Engineers Professional Group C1 (Software Engineering) – Tools and Techniques for Maintaining Traceability*

*During Design* (pp. 5/1–5/7). London: IEE – The Institution of Electrical Engineers.

Zisman, A., Spanoudakis, G., Péres, E.M., & Krause, P. (2003). Tracing software engineering artifacts. In *International Conference on Software Engineering Research and Practice* (pp. 448–455).

## Key Terms

**Requirements Engineering** is the process of identification, analysis, documentation and management of requirements so that they are useful for implementation, consistent with the needs of stakeholders and to provide the software product quality.

**Requirements Management** is a parallel process to support other requirements engineering activities such as identification, change management and requirements traceability throughout the development process.

**Requirements Traceability** is the ability to define traces between requirements and other artifacts throughout the development process; and to follow those traces in any direction.

**Traceability Links** are the main resource used to maintain and represent the traceability relationships. A link establishes an explicit relationship between two artifacts.

**Models** is a set of elements describing something built for some purpose that is subject to a particular form of analysis.

**Techniques** are technical and managerial procedures that aid in the evaluation and improvement of the software development process.

**Stakeholders** are individuals or organizations involved or affected by the system and who have an influence on the requirements. They could be end-users, managers, suppliers and customers.

## **PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE**

### **Teses e Dissertações (TDI)**

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

### **Manuais Técnicos (MAN)**

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

### **Notas Técnico-Científicas (NTC)**

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

### **Relatórios de Pesquisa (RPQ)**

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

### **Propostas e Relatórios de Projetos (PRP)**

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

### **Publicações Didáticas (PUD)**

Incluem apostilas, notas de aula e manuais didáticos.

### **Publicações Seriadas**

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

### **Programas de Computador (PDC)**

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

### **Pré-publicações (PRE)**

Todos os artigos publicados em periódicos, anais e como capítulos de livros.