UNIVERSIDADE METODISTA DE PIRACICABA
FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

UM PROCESSO DE VALIDAÇÃO DE REQUISITOS NÃO-FUNCIONAIS BASEADO NO *NFR-FRAMEWORK*

ANSELMO DE ARAÚJO COUTO

ORIENTADOR: PROF. DR. LUIZ EDUARDO GALVÃO MARTINS

PIRACICABA, SP
2009

UNIVERSIDADE METODISTA DE PIRACICABA
FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

UM PROCESSO DE VALIDAÇÃO DE REQUISITOS NÃO-FUNCIONAIS BASEADO NO *NFR-FRAMEWORK*

ANSELMO DE ARAÚJO COUTO

ORIENTADOR: PROF. DR. LUIZ EDUARDO GALVÃO MARTINS

Trabalho de dissertação apresentado ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP
2009

Couto, Anselmo de Araújo

Um Processo de Validação de Requisitos Não-Funcionais Baseado no NFR-Framework. Piracicaba, 2009

90p.

Orientador: Prof. Dr. Luiz Eduardo Galvão Martins
Dissertação (Mestrado) – Universidade Metodista de Piracicaba, Faculdade de Ciências Exatas e da Natureza, Programa de Pós-Graduação em Ciência da Computação.

1- Engenharia de Requisitos 2- Modelagem de Requisitos Não-Funcionais 3- NFR-Framework 4- Elicitação de Requisitos 5- Validação de Requisitos

PROCESSO DE VALIDAÇÃO DE REQUISITOS NÃO-FUNCIONAIS BASEADO NO *NFR-FRAMEWORK*

Autor: Anselmo de Araújo Couto

Orientador: Prof.Dr. Luiz Eduardo Galvão Martins

Dissertação de Mestrado apresentada em 20 de Fevereiro de 2009, à Banca Examinadora constituída dos Professores:

_____

Prof. Dr. Luiz Eduardo Galvão Martins

UNIMEP

_____

Prof. Dr. Luiz Camolesi Júnior

UNICAMP/CESET

_____

Profª. Dra. Miriam Sayão

PUC Rio Grande do Sul

À
minha querida esposa Roberta L. Sogayar
e aos meus filhos Catarina e Caetano.
A minha mãe Ancyla.

AGRADECIMENTOS

Agradeço a Deus pelas bênçãos que tenho recebido e pela condução da minha vida.

Agradeço à minha esposa Roberta L. Sogayar e aos meus filhos Catarina e Caetano pelo amor, carinho, dedicação e compreensão.

Agradeço a minha mãe e irmãos por sempre me estimularem a trilhar o caminho do aprender e apreender.

Agradeço ao meu orientador, Prof. Dr. Luiz Eduardo Galvão Martins, pelo incentivo, apoio e orientações durante a realização deste trabalho e por confiar no meu potencial.

Agradeço aos professores do programa de Mestrado em Ciência da Computação da UNIMEP pelo conhecimento compartilhado.

Agradecimentos finais ao Dr. Roberto Sogayar e à Dra. Maria Inês Terra L. Sogayar pelo apoio irrestrito nessa etapa tão importante da minha vida.

"Há um tempo em que é preciso abandonar as roupas usadas, que já têm a forma do nosso corpo, e esquecer os nossos caminhos, que nos levam sempre aos mesmos lugares. É o tempo da travessia: e, se não ousarmos fazê-la, teremos ficado, para sempre, à margem de nós mesmos".

Fernando Pessoa

RESUMO

O projeto de desenvolvimento de software é um processo complexo que envolve aspectos distintos para a realização. Um desses aspectos corresponde ao processo de validação de requisitos que deve ser realizado sobre uma coleção de documentos, que apresentam uma especificação produzida de acordo com as necessidades dos clientes. O objetivo é garantir uma solução mais adequada ao usuário, isto engloba especificação precisa, coerente e inequívoca. Muitos engenheiros de requisitos e a maioria das abordagens existentes incidem sobre os requisitos funcionais, isto é, sobre aquilo que o sistema deve fazer. Contudo, requisitos não-funcionais também são importantes para o desenvolvimento do *software*, pois questões como qualidade, segurança e desempenho são cruciais para o sucesso do sistema. Neste contexto, propomos um processo de validação de requisitos não-funcionais baseado no NFR-Framework. Nossa proposta oferece uma sistematização para validar os requisitos não-funcionais, analisando sua evolução e a qualidade do documento de requisitos produzido. Para isto testamos a sistematização em 3 (três) projetos de especificação de requisitos. Os resultados obtidos nos permitiram concluir que: a) as atividades propostas servem como instrumento de melhoria da qualidade no processo de validação de requisitos não-funcionais; b) as atividades propostas auxiliam os Engenheiros de Requisitos a efetuarem um refinamento dos requisitos não-funcionais, identificando propriedades do sistema que anteriormente estavam obscuras.

PALAVRAS-CHAVE: Engenharia de Requisitos, Elicitação de Requisitos, Modelagem de Requisitos Não-Funcionais, Validação de Requisitos, *NFR-Framework*.

ABSTRACT

The project of software development is a complex process which involves distinct aspects for its accomplishment. One of these aspects corresponds to the requirement validation process that should be performed under a collection of documents that present a specification elaborated according to the stakeholder's needs. The objective is to guarantee a more adequate solution to the end user and it comprises an accurate, coherent and inequivocous specification. Many requirement engineers and most of the existing approaches incise on functional requirements, that is, under what the system should do. However, non functional requirements are also important to the software development, because issues such as quality, safety and performance are crucial for the success of the system. In this context, we propose a process of non-functional requirements validation based on the *NFR-Framework*. Our proposal offers a systematization to validate non functional requirements analyzing their evolution and the quality of the rquiremment document produced. In order to do that, we tested the systematization of three projects of requirement specification. The results obtained enabled us to conclude that: a) the proposed activities serve as a tool of quality improvement in the process of validating non-functional requirements; b) the proposed activities assist Requirement Engineers to promote a refinement of non-functional requirements, identifying properties of the system that were obscures before.

KEY WORDS: Requirements Engineering, Requirements Elicitation, Modeling of Non-Functional Requirements, Validation Requirements, NFR-Framework.

Sumário

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

DOD - DEPARTAMENTO DE DEFESA DOS ESTADOS UNIDOS DA AMÉRICA

EUA - ESTADOS UNIDOS DA AMÉRICA

IEEE - INSTITUTO DE ENGENHEIROS ELETRICISTAS E ELETRÔNICOS

META - METODOLOGIA DE ELICITAÇÃO DE REQUISITOS BASEADA NA TEORIA DA ATIVIDADE

RNFS - REQUISITOS NÃO-FUNCIONAIS

SIG - SOFTGOAL INTERDEPENDENCY GRAPHS

LISTA DE TABELAS

1 INTRODUÇÃO

Sistemas complexos de *softwares* estão hoje espalhados pelo mundo e em diferentes áreas, desempenhando papel fundamental no dia a dia de cada ser humano. Isto pode ser evidenciado, por exemplo, pelo crescimento de áreas como comércio eletrônico, computação móvel, aviação, usinas elétricas e telecomunicações. Um comportamento incorreto do software pode levar a uma perda catastrófica em termos de custos, danos causados ao ambiente, ou até mesmo de vidas humanas.

Com o surgimento da Engenharia de Requisitos como disciplina de investigação, em meados da década de 1990, a preocupação inicial era com o sistema de *software*. A partir daí, evoluiu para uma perspectiva mais ampla em relação aos aspectos dos sistemas e das organizações, se partimos da premissa de que ao projetar sistemas de *software*, o engenheiro de requisitos tem como objetivo melhorar a situação vista como problemática das empresas.

Sendo o projeto de *software* uma atividade complexa, a validação de requisitos é de grande importância, sobretudo quando se lida com a concepção de aplicações de *software*. Por isso ela deve ser realizada sobre uma coleção de documentos que apresentam uma especificação produzida de acordo com as necessidades dos clientes. O objetivo é garantir uma solução mais adequada para o usuário: Isto engloba uma especificação precisa, coerente e inequívoca.

Muitos Engenheiros de Requisitos e a maioria das abordagens existentes enfocam os requisitos funcionais. Entretanto, alguns autores (CHUNG et. al., 2000; LAMSWEERDE, 2001; MYLOPOULOS et. al., 2001) apontam que requisitos não-funcionais também são importantes para o desenvovilmento, pois questões como qualidade, segurança e desempenho são cruciais para o sucesso do sistema de *software*.

Neste contexto, propomos um processo de Validação de Requisitos de *Software* baseado no *NFR-Framework*. Visamos, principalmente, oferecer um processo para

validar os requisitos não-funcionais, analisando sua evolução e a qualidade do documento de requisitos original.

1.1 OBJETIVO DO TRABALHO

Engenheiros de requisitos têm freqüentemente negligenciado ou esquecido os requisitos não-funcionais (RNFs) na concepção do *software*. Assim, o presente trabalho tem como objetivo:

a) Propor um processo de Validação de Requisitos Não-Funcionais, baseado no *NFR-Framework*;

Como consequências deste objetivo duas premissas para o processo de validação de requisitos não-funcionais foram elencadas. São elas: que as atividades propostas servem como instrumento de melhoria da qualidade no processo de validação de requisitos não-funcionais e auxiliem os engenheiros de requisitos a efetuar um refinamento dos requisitos não-funcionais, identificando propriedades do sistema que anteriormente estavam obscuras.

1.2 CONTEXTUALIZAÇÃO DA PESQUISA

O projeto tem como eixo condutor a proposta de abordagem de validação de requisitos de *software* orientado a metas, para requisitos não-funcionais. Para prover sustentação ao desenvolvimento da pesquisa, procuramos trabalhar com conceitos relacionados aos assuntos da Engenharia de Requisitos, Validação de Requisitos e e o NFR Framework para requisitos não-funcionais.

Foi dada atenção especial à Engenharia de Requisitos Orientada a Metas, salientando sua importância para o processo da Engenharia de Requisitos. A escolha de metas tem como fundamento a crescente investigação realizada pela Engenharia de Requisitos. Várias pesquisas indicam a necessidade dos modelos conceituais em lidar com metas (CYSNEIROS et. al., 2001). Cysneiros (2001) argumenta que uma das vantagens de termos tais modelos conceituais é a capacidade de se representar aspectos não-funcionais, como confidencialidade, desempenho, qualidade e precisão.

1.3 ESTRUTURA DO TRABALHO

Esta dissertação está composta por 7 capítulos, organizados da seguinte forma:

O capítulo 2 aborda conceitos gerais da Engenharia de Requisitos, incluindo uma seção sobre a Engenharia de Requisito Orientada a Metas.

O capítulo 3 dá visão geral sobre validação de requisitos, sendo abordados os aspectos relevantes aos métodos de validação de requisitos já propostos, como por exemplo: revisões, validação por protótipos, validação de modelos e teste.

O capítulo 4 apresenta um estudo sobre o *NFR-Framework*, abordando todos os aspectos propostos por Chung (2000).

O capítulo 5 refere-se ao trabalho da dissertação. Neste capítulo apresentamos as atividades para um processo de validação de requisitos não-funcionais, bem como os objetivos e as atividades propostas para o processo de validacão dos RNFs.

No capítulo 6 são apresentados três estudos de casos aplicando o processo de validação desenvolvido, incluindo uma discussão sobre.

No capítulo 7 expomos as conclusões finais e os trabalhos futuros.

2 ENGENHARIA DE REQUISITOS

Neste capítulo são abordados os principais conceitos sobre Engenharia de Requisitos, as principais atividades, assim como suas terminologias.

2.1 INTRODUÇÃO

A Engenharia de Requisitos de *software* é uma área relativamente nova, que tem se desenvolvido muito rapidamente. Em 1993, no primeiro Simpósio Internacional de Requisitos promovido pelo *Institute of Electrical and Electronic Engineers* (IEEE), foram definidas as fronteiras entre a Engenharia de Requisitos e as demais atividades de engenharia de *software*. A comunidade científica começou a reconhecer a elicitação[1] como tema de investigação, de modelagem e análise (LEITE e DOORN, 2004).

O objetivo desta divisão entre a Engenharia de Requisitos e as atividades da engenharia de *software* é ressaltar que o processo de definição dos requisitos de *software* é extremamente importante, pois requer fundamentação e processos próprios, os quais devem ser planejados e gerenciados ao longo de todo o ciclo de vida (BROOKS, 1987).

Para Martins (2001), a Engenharia de Requisitos vem crescendo no contexto de desenvolvimento de *software* porque projetistas de *software* perceberam que requisitos bem elicitados são fundamentais para o sucesso dos projetos.

Entretanto, há muitos problemas relacionados com a Engenharia de Requisitos, entre os quais se destacam (BROOKS, 1987):

- Definição do escopo;
- Entendimento entre as diferentes comunidades afetadas pelo desenvolvimento;
- Problemas em lidar com a volatilidade dos requisitos.

Christel e Kang (1992) indicam ainda que questões envolvidas nesta área são problemáticas, incluindo:

[1] A palavra elicitação vem do inglês *elicitation,* que significa descobrir algo obscuro ou obter informações.

- Dificuldade em atingir requisitos completos;
- Dificuldade de análise e validação.

Esses problemas podem levar a requisitos insuficientes, conseqüentemente comprometendo o desenvolvimento do sistema. Além disso, mais tarde, o sistema pode ser julgado como insatisfatório ou inaceitável. Assim, Boehm (1984) argumenta que muitos erros de requisitos passam despercebidos para fases posteriores do ciclo de vida, e que corrigir esses erros durante ou após a execução se revelou extremamente caro.

Estudos desenvolvidos dentro da Engenharia de Requisitos têm mostrado que o impacto dos requisitos pobres ou mal formulados é substancial. Os estudos realizados por Boehm (1987) sugerem que erros de requisitos, especificação e *design* são os mais caros em um sistema, em média 64%, em comparação com 36% para erros de codificação. A maioria destes erros não são encontrados durante a etapa do desenvolvimento, pelo contrário, nas fases de testes do sistema. Os custos resultantes para corrigir tais erros aumentam com o tempo de atraso em encontrá-los. O autor salienta que um erro de requisito encontrado na fase inicial custa apenas cerca de um quinto se encontrado na fase de teste, e de um quinze avos depois que o sistema estiver em uso (BOEHM, 1987).

Desta forma, Boehm (1987) acredita que reduzir erros de requisitos é possível através da melhoria das condições de elicitação, uma atividade frequentemente esquecida ou apenas parcialmente abordada pelas técnicas de engenharia.

As dificuldades mencionadas, agregadas às peculiaridades de cada projeto, mais as mudanças culturais existentes, posicionam a Engenharia de Requisitos como a chave para a complexa área de desenvolvimento de sistema (BROOKS, 1987).

Nas próximas seções são apresentadas definições a partir da literatura para clarificar conceitos relacionados à Engenharia de Requisitos.

2.2 REQUISITOS

Para Kotonia e Sommerville (1998), requisito é uma declaração de um serviço ou uma restrição que deve ser "implementada" em um sistema.

A norma IEEE Standard Glossary of Software Engineering Terminology (IEEE, 1990) define requisito como:

> **a)** condição ou capacidade necessária para um usuário resolver um problema ou atingir um objetivo;
>
> **b)** condição ou capacidade que deve ser satisfeita por um sistema ou componente para satisfazer contrato, padrão, especificação, ou outros documentos formalmente impostos;
>
> **c)** representação documentada de uma condição ou capacidade como em (a) ou (b).

Já Goguen (1993) define como:

> "propriedades que um sistema de *software* deve ter para conseguir êxito no ambiente onde será utilizado".

Sommerville e Sawyer (1997) por sua vez, afirmam que requisitos são:

> "descrições de como o sistema deve se comportar, com sua propriedade ou atributos".

Os autores ainda destacam os requisitos nos seguintes aspectos:

- facilidade no nível do usuário;
- propriedade geral do sistema;
- restrição específica sobre o sistema;
- especificação de um algoritmo particular, que deva ser empregado em cálculos particulares;
- restrição aplicável ao processo de desenvolvimento do sistema.

Analisando as definições acima, observa-se que requisito é uma declaração sobre o que o sistema proposto deverá fazer documentado e validado pelo cliente, para que o problema possa ser resolvido adequadamente e com qualidade.

Por esta ótica, há duas categorias de requisitos: aqueles responsáveis pela funcionalidade do sistema, e aqueles responsáveis por qualidades que devam estar presentes, tais como desempenho, integridade, disponibilidade e segurança. Os primeiros são denominados requisitos funcionais e os últimos requisitos não-funcionais.

Nuseibeh e Easterbrook (2000) afirmam que uma das primeiras medidas do sucesso de um sistema de *software* é verificar se ele atende às necessidades dos clientes. Entretanto, esse sucesso se deve essencialmente ao levantamento de requisitos. Adicionalmente, observam que, se os requisitos são inadequados, inconsistentes, incompletos e ambíguos, exercem um grande impacto na qualidade do *software* final.

Além de requisitos**,** outra definição necessária é a de Engenharia de Requisitos, amplo campo de estudos dentro da engenharia de *software*, e cada vez mais necessário para resolver os problemas encontrados nas organizações com relação à definição de sistemas. De uma maneira geral, elas estão percebendo que o tempo utilizado no entendimento do problema é um excelente investimento.

2.3 ENGENHARIA DE REQUISITOS

Segundo Sommerville e Sawyer (1997), Engenharia de Requisitos é um termo que abrange todas as atividades envolvidas na descoberta, documentação e manutenção de um conjunto de requisitos para um sistema computacional. O uso do termo “engenharia” implica que técnicas sistemáticas e repetíveis devem ser utilizadas para garantir que os requisitos do sistema sejam completos, consistentes e relevantes.

Já Letier e Lamsweerde (2004) salientam que a Engenharia de Requisitos está preocupada com a identificação dos objetivos a serem alcançados pelo sistema, as operacionalizações desses objetivos, especificações, limitações e atribuição de

responsabilidades dos serviços. Adicionalmente, fatores humanos, físicos e de *software* formam os componentes do sistema.

Nuseibeh e Easterbrook (2000) argumentam que:

> "a Engenharia de Requisitos é o processo de descobrir o propósito do sistema, através da identificação dos *stakeholders*[2] e as suas necessidades, e documentar de forma não ambígua para análise, comunicação e posterior implementação".

Leveson (2000) afirma que a Engenharia de Requisitos tem muito a ver com a psicologia cognitiva, teoria de sistemas e interação homem–máquina. Ela sugere que é importante especificar a intenção e finalidade subjacente aos requisitos identificados. Sua mensagem é a de que a tarefa de escrever especificação é multidisciplinar.

Estas definições são coerentes por uma série de razões. Primeiro, destacam a importância do estabelecimento de metas ou propósitos. Segundo, referem-se à especificação precisa, fornecendo base para analisar requisitos, validar o que o usuário precisa definindo como os projetistas devem construir e verificar se o fizeram corretamente. Por último, projetos de *software* dependem de pessoas para serem bem sucedidos, mas metodologias também são importantes, pois pessoas submetidas a uma estrutura de trabalho ruim geram resultados ruins.

Muitos autores, entre eles Kotonia (1998), dizem que o principal artefato produzido em Engenharia de Requisitos é o documento de requisitos. De acordo com Sommerville e Sawyer (1997), o documento de requisitos é uma declaração formal dos requisitos para os *stakeholders*, e estes podem ser clientes, usuários finais ou a equipe de desenvolvimento do *software*.

Dependendo da organização, o documento de requisitos pode receber diversos nomes, tais como "especificação funcional", "definição de requisitos", ou ainda

[2] Palavra da língua inglesa que significa todas as pessoas físicas ou jurídicas que são direta ou indiretamente afetadas por projeto.

"especificação dos requisitos do *software*". Algumas organizações, como o Departamento de Defesa dos EUA (DOD) e o IEEE, definiram seus próprios padrões para o documento de requisitos. Provavelmente, o mais conhecido destes padrões seja o IEEE Std 830-1998 (SOMMERVILLE e SAWYER, 1997).

Outro documento de requisito muito usado é o *Volere*. Desde a sua introdução, em 1995, sua abordagem tem sido adotada por milhares de organizações em todo o mundo (ROBERTSON e ROBERTSON, 2006). O modelo fornece seções para cada um dos tipos de requisitos para sistemas de *software*. Os autores afirmam que este modelo é simples e completo, pois foi produzido pelas experiências acumuladas em desenvolvimento de *software*.

2.4 PROCESSO DE ENGENHARIA DE REQUISITOS

A norma IEEE Standard Glossary of Software Engineering Terminology (IEEE, 1990) define processo como:

> "... uma seqüência de passos realizados para um determinado fim".

Kotonia e Sommerville (1998) argumentam que:

> "... processo é um conjunto organizado de atividades que transforma entradas em saídas".

Para Grande e Martins (2006), o processo de Engenharia de Requisitos objetiva principalmente a busca de conhecimentos das regras de negócios e verificação das necessidades do cliente, obtendo uma especificação não ambígua e completa dos requisitos de *software*, com o intuito de minimizar os erros, inadequações e falhas no produto final a ser entregue ao cliente.

Camacho (2005) determina que processo de Engenharia de Requisitos, como um todo, é muito mais complexo do que apenas uma atividade de transferência de conhecimento entre clientes e engenheiros de requisitos. Primeiro porque os clientes dificilmente têm uma idéia clara do que realmente querem, e segundo porque diferentes pessoas dentro da organização normalmente possuem requisitos

conflitantes, além de existir, na maioria dos casos, limitações tecnológicas influenciando os requisitos, entre outros. Assim, Processo de Engenharia de Requisitos é um conjunto estruturado de atividades dirigidas para criar, validar e manter um documento de requisitos. Poucas organizações têm um processo de Engenharia de Requisitos explicitamente definido e padronizado. Entretanto, Sommerville e Sawyer (1997) sugerem que cada organização deva desenvolver um processo adequado à sua realidade.

Nuseibeh e Easterbrook (2000) argumentam que a preparação de um trabalho inclui também a identificação de um processo apropriado para a Engenharia de Requisitos, bem como a seleção dos métodos e das técnicas para as suas atividades. O termo “processo” denota um exemplo de um modelo, que é uma descrição abstrata de como conduzir as atividades, descrevendo o comportamento de um ou mais agentes e a gerência dos recursos. Uma técnica prescreve como executar uma atividade particular e, se necessário, como descrever o produto dessa atividade em uma notação particular. Um método fornece uma prescrição de como executar uma coleção das atividades, focalizando como um conjunto relacionado das técnicas pode ser integrado, fornecendo a orientação em seu uso.

Para entender melhor o processo de Engenharia de Requisitos, segue abaixo a Figura 2.1 extraída de Kotonia e Sommerville (1997).

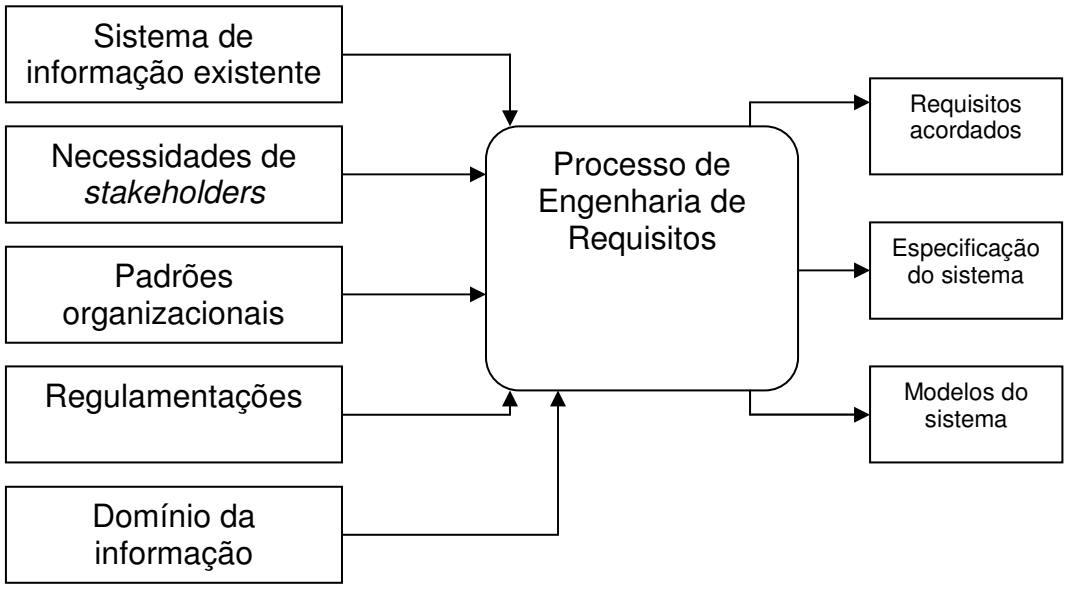


Figura 2.1. Entradas e Saídas do Processo de Engenharia de Requisitos (KOTONYA e SOMMERVILLE, 1997)

Pode-se perceber que as entradas para o processo de Engenharia de Requisitos incluem: informações sobre sistemas já existentes, necessidades dos *stakeholder*; padrões da organização, regulamentações, e informações do domínio da aplicação. Todas essas informações são utilizadas para a realização das atividades do processo. Como resultados, surgem os requisitos acordados, uma especificação de requisitos e modelos do sistema.

Esta visão ressalta que, para a realização das atividades do processo de Engenharia de Requisitos, fatores humanos e técnicos têm de ser adequadamente tratados, objetivando, desta forma, que cada resultado do processo seja o mais completo e consistente possível.

Embora diferentes projetos exijam processos com características específicas para contemplar as suas peculiaridades, é possível criar um grupo de atividades básicas que devem ser considerados na definição de qualquer processo de Engenharia de Requisitos. São elas (LEITE e DOORN, 2004):

- Elicitação;
- Análise;
- Modelagem.

2.5 ATIVIDADE DA ENGENHARIA DE REQUISITOS

Nuseibeh e Easterbrook (2000) e Brackett (1990) demonstram que há vários grupos de atividades propostos para o Processo de Engenharia de Requisitos. Entretanto, neste trabalho abordaremos a proposta de Sommerville, conforme a Figura 2.2, cujo processo de atividades envolve:

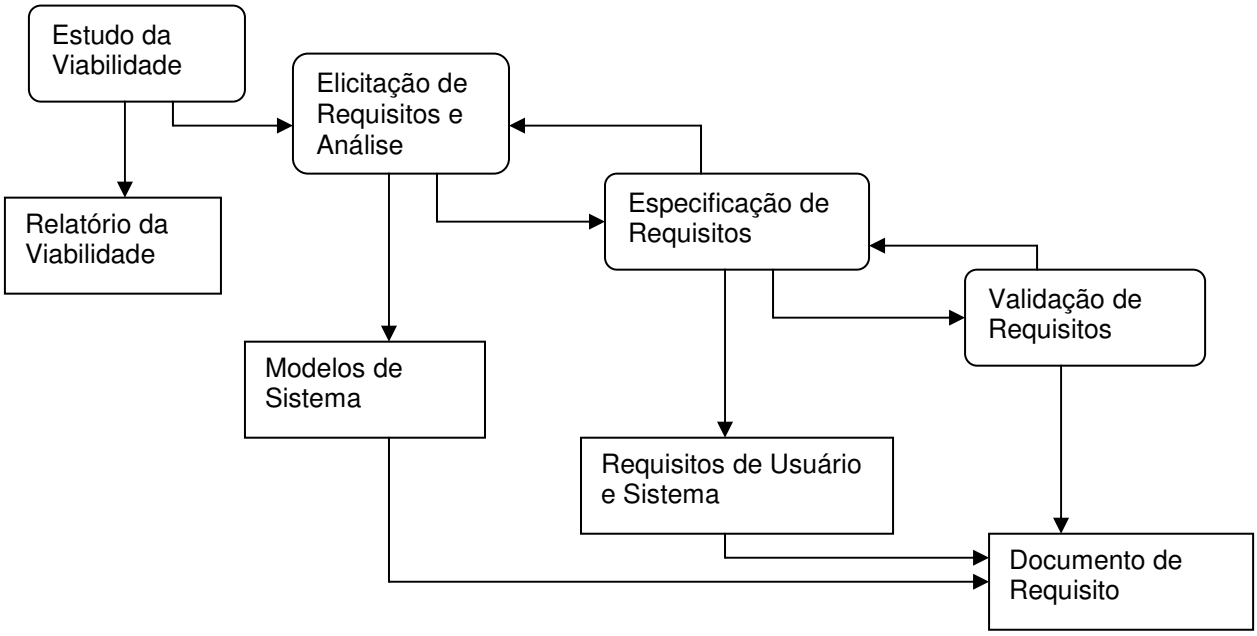


Figura 2.2 – Processo de Engenharia de Requisitos (SOMMERVILLE, 2001)

Conforme a Figura 2.2 o processo divide as tarefas em cinco atividades principais:

- **Estudo da Viabilidade** – O objetivo nesta fase é avaliar, de um ponto de vista tecnológico e organizacional, a viabilidade do projeto (SOMMERVILLE, 2001).
- **Elicitação de requisitos e Análise** – Elicitar é um processo iterativo, e a todo o momento as circunstâncias fazem com que o analista execute uma etapa usando uma técnica específica da elicitação. O resultado é uma lista de requisitos, ou algum tipo do modelo do sistema, ou de ambos (HICKEY e DAVIS, 2003). Para Sommerville e Sawyer (1997), elicitação de requisitos é o meio pelo qual analistas determinam os problemas e as necessidades dos clientes, para que o pessoal do desenvolvimento possa construir um sistema que realmente resolva os problemas dos clientes.

  Sommervile (2001) salienta que elicitação e análise são processos difíceis por vários razões:

  a) *Stakeholders* sabem o que eles querem apenas em termos gerais; a maioria tem dificuldade em articular o que necessitam do sistema, desta forma, podem fazer exigências irrealistas e

muitas vezes não têm conhecimento do custo dos pedidos levantados;

b) *Stakeholders* naturalmente expressam requisitos com conhecimento implícito de seu próprio trabalho;

c) Diferentes *stakeholders* têm necessidades diferentes e podem expressá-los de maneiras diferentes. Engenheiros de requisitos têm de descobrir todas as fontes comuns de requisitos e gerenciar os possíveis conflitos;

d) Fatores políticos podem influenciar os requisitos do sistema. Estes podem ser provenientes de gestores que exigem requisitos específicos, porque lhes permitem aumentar sua influência na organização;

e) O ambiente empresarial e econômico em que se realiza a análise é dinâmica. Inevitavelmente, mudanças durante a análise do processo ocorrerão. Novos requisitos podem surgir de novos *stakeholders* que não foram consultados nas fases iniciais do levantamento.

- **Especificação de Requisitos** – Etapa na qual resultados da elicitação e análise de requisitos serão transformados em documentos que organizam os requisitos do sistema, servindo de base entre o desenvolvedor e o cliente (MARTINS, 2001).
- **Validação de requisitos** – Nesta etapa, a preocupação é demonstrar se os requisitos realmente definem o sistema que o cliente deseja. Ela tem muito em comum com a análise, uma vez que está preocupada em encontrar problemas com os requisitos. No entanto, são processos distintos, pois a validação deve se preocupar com o projeto completo de requisitos, enquanto a análise envolve trabalho com requisitos incompletos nas fases iniciais (SOMMERVILLE, 2001). Ainda segundo o autor, validação de requisitos é importante porque os erros de requisitos podem levar a um extenso retrabalho; além disso, o custo de fazer uma mudança no sistema é muito maior do que reparar o erro na sua concepção.
- **Documentação de requisitos** - Para Power e Moynihan (2003), o documento de requisitos é uma parte integrante e essencial do processo de

requisitos de *software*. O termo "documentação" refere-se tanto ao processo de documentar requisito quanto ao produto resultante do trabalho. Além disso, o documento de requisito é usado para comunicar requisito do sistema aos clientes, aos usuários, aos gerentes e aos colaboradores do sistema (SOMMERVILLE e SAWYER, 1997).

- **Gerenciamento dos requisitos** – É o planejamento e controle dos requisitos nas atividades de elicitação e análise, especificação, validação de requisitos (THAYER e DORFMAN, 1997).

Segundo Kotonia e Sommerville (1998), problemas surgem na execução destas atividades. Diversos fatores dificultam o desenvolvimento de documentos de requisitos que realmente satisfaçam os usuários. Alguns problemas com requisitos encontrados durante o processo de Engenharia de Requisitos são relatados por:

- Os requisitos não refletem as reais necessidades do cliente em relação ao sistema a ser desenvolvido;
- Os requisitos são inconsistentes e/ou incompletos;
- É dispendioso fazer mudanças após os requisitos terem sido acordados entre as partes (cliente e equipe de desenvolvimento);
- É comum ocorrer interpretação errônea entre clientes e equipe de desenvolvimento.

Para atender aos requisitos dos usuários e clientes, é necessário que as atividades do processo de Engenharia de Requisitos sejam realizadas de forma mais sistemática e com um suporte por computador. Isto acarretará processos com maior maturidade, o que permite que uma organização obtenha *softwares* com qualidade, não por dependência de esforços individuais, mas por uma própria evolução do seu processo, o qual utiliza boas práticas de Engenharia de Requisitos.

Não existem limites bem definidos entre as atividades propostas. De fato, elas são intercaladas e existe um elevado grau de iteração e *feedback* entre elas. Em geral, o processo é executado até quando todos os usuários estiverem satisfeitos e de acordo com os requisitos (SOMMERVILLE, 2001).

2.6 ENGENHARIA DE REQUISITOS ORIENTADA A METAS

Há uma forte evidência de que a Engenharia de Requisitos necessita de métodos e instrumentos adequados que ajudem os engenheiros de requisitos a buscarem, por exemplo, qualidade nos requisitos, diminuição do retrabalho sobre os requisitos, e redução dos recursos, tais como o tamanho das equipes (AL-SUBAIE, MAIBAUM, 2006).

Kavakli e Loucopoulos (2003) argumentam que projetos de Engenharia de Requisitos exigem o envolvimento de vários *stakeholders* (o patrocinador, agentes externo, desenvolvedores e usuários do sistema). O desafio é fazer as partes interessadas coordenarem suas ações, para proporcionar um resultado comum. Por estes motivos, há grande interesse pela abordagem que se baseia no sistema de identificação de metas, através de questões do porquê uma determinada meta é exigida, como pode ser atingida, quem é responsável por ela no sistema e no ambiente (YU e MYLOPOULOS, 1998).

Desta forma, a Engenharia de Requisitos Orientada a Metas é um paradigma cada vez mais reconhecido para elicitar, elaborar, analisar, negociar, documentar e modificar requisitos de *software* (LAMSWEERDE, 2001; KAVAKLI e LOUCOPOULOS, 2003).

Para Zave e Jackson (1997), uma meta é um objetivo que um sistema deve alcançar. Lamsweerde (2001) define metas como propriedades requeridas do sistema, expressa pelos *stakeholders.* As metas são formuladas em diferentes níveis de abstração, abrangendo o alto nível, preocupações de estratégia e baixo nível, preocupações técnicas.

As metas cobrem tipos diferentes de interesses: os funcionais, associados aos serviços a serem fornecidos; e os não-funcionais, associados à qualidade do serviço, tal como à segurança, desempenho, e confiabilidade (LAMSWEERDE, 2001).

2.6.1 NECESSIDADE DAS METAS

Existem muitas razões pelas quais metas são importantes para o processo da Engenharia de Requisitos, por exemplo, (LAMSWEERDE, 2001):

- Atingir a perfeição é uma das principais preocupações. Metas fornecem um critério suficiente para a completude de uma especificação de requisitos: a especificação está completa, se todas as metas podem ser comprovadas e alcançadas;

- Evitar requisitos irrelevantes é outra grande preocupação. Metas fornecem um critério de requisitos pertinentes; é pertinente quando um conjunto de metas está em conformidade com o domínio da aplicação, ou quando a especificação é utilizada na prova de pelo menos uma meta;

- Explicar requisitos às partes interessadas é outra questão importante. As metas fornecem a fundamentação para as necessidades dos *stakeholders*. De forma semelhante, há o alinhamento das metas do projeto com a concepção dos processos. Para as metas do projeto, uma árvore de refinamento fornece ligações de rastreabilidade dos objetivos estratégicos (alto nível) aos requisitos técnicos de baixo nível. Para sistemas da aplicação de negócio, as metas podem ser usadas para relacionar *software* aos contextos organizacional e do negócio;

- O refinamento das metas oferece um mecanismo natural para a estruturação do documento de requisito, aumentando a legibilidade do documento;

- Os engenheiros de requisitos podem considerar várias alternativas durante o processo de levantamento dos requisitos: os refinamentos alternativos das metas, por exemplo, fornecem um nível de abstração, em que os responsáveis pelas decisões podem ser envolvidos para validar as escolhas que são feitas, ou sugerir alternativas negligenciadas durante o processo. Os refinamentos alternativos das metas permitem que as propostas alternativas do sistema sejam exploradas;

- Controlar conflitos entre vários pontos de vista é um dos principais problemas. As metas foram reconhecidas para fornecer as bases para detectar conflitos entre requisitos e resolvê-los;

- Evolução e volatilidade dos requisitos. Um requisito representa uma maneira particular de se atingir uma meta específica; conseqüentemente, é mais provável que um requisito evolua mais do que a própria meta. Desta forma, é reconhecido que as metas são mais estáveis, devido ao seu nível de abstração; isto possibilita, caso os requisitos evoluam, integração fácil com os modelos das metas estabelecidos;

- Por último, mas não menos importante, as metas conduzem à identificação das necessidades dos *stakeholders*, sistematicamente usada em conjunto com cenários, o que tem se mostrado muito eficiente (Anwer e Ikram; 2006 LAMSWEERDE e LETIER, 2002; MYLOPOULOS et al., 2001 ; YU e MYLOPOULOS, 1998 ; DARDENNE et al, 1993).

2.6.2 IDENTIFICAÇÃO DAS METAS

A identificação das metas não é necessariamente uma tarefa fácil (LAMSWEERDE, 2001). Para o autor, as metas são explicitamente indicadas pelos *stakehokders* ou em algum material à disposição do engenheiro de requisito. Na maior parte das vezes, as metas estão implícitas, sendo assim necessário realizar a sua elicitação.

Outro fator importante para a identificação das metas é a análise preliminar do sistema, a qual normalmente resulta em uma lista de problemas, que podem ser depois resolvidos.

Por isso, Engenharia de Requisitos Orientada a Metas fornece métodos e uma série de técnicas para a identificação. A lista a seguir apresenta um resumo destas técnicas (LAMSWEERDE, 2001; ANTON, 1997):

- Compreensão dos problemas relatados pelos *stakeholders*;
- Extrair declarações intencionais:

- transcrições de entrevista;
- políticas empresariais;
- missão do empreendimento;
- metas do empreendimento;
- *workflow*, diagramas;
- cenários escritos com os *stakeholders*.

- As metas são identificadas de forma incremental. As perguntas “por que?”, “como?” e “de que outra forma?” ajudam a detectar as metas (LAMSWEERDE, 2001).

O porquê ajuda a descobrir as metas e a razão dos objetivos, que, de fato, identificam os objetivos mais elevados. Este mecanismo fornece uma razão contínua para cada requisito; Yu e Mylopoulos (1998) argumentam que esta é a maneira natural de se fazer o processo da engenharia de requisitos: um requisito não será considerado se não contribuir para um objetivo de nível mais alto na hierarquia.

Metas necessitam ser refinadas até que as submetas estejam alcançadas (LAMSWEERDE, 2001). A pergunta "como?” ajuda a derivar as metas em níveis inferiores (ANWER e IKRAM, 2006).

Por fim, "de que outra forma?” ajuda a identificar as alternativas para satisfazer os níveis superiores das metas. Este acordo hierárquico provê vantagens implícitas, como rastreabilidade de nível superior, integralidade e identificação de conflitos.

2.6.3 TIPOS DE METAS

Tipos de meta podem ser baseados em requisitos funcionais e não-funcionais (LAMSWEERDE, 2001). Três tipos de metas são identificados e definidos por Regev e Wegmann (2005): realização, manutenção e *softgoal*.

- Realização das Metas - O objetivo da realização é cumprido quando as metas são atingidas. Lamsweerde (2001) argumenta que ela está relacionada com a ação reguladora, que tenta manter uma interpretação mais perto de uma

norma, ou um aprendizado que pode até mesmo alterar o sistema de normas, interpretações e ações;

- Manutenção de metas – é definida com uma propriedade que detem as metas satisfeitas, enquanto sua condição permanece constante ou verdadeira e todos os futuros estados. Em geral, a manutenção de metas mapeia os requisitos não-funcionais (REGEV e WEGMANN, 2005)

- *Softgoal* - meta que não tem critérios bem definidos para a realização (LAMSWEERDE, 2001). A *softgoal* está relacionada com os requisitos não-funcionais (BAIXAULI et. al., 2004).

2.6.4 NÍVEIS DE ABSTRAÇÃO DE METAS

Três níveis de abstração são descritos na Figura 2.3, conforme (ANWER e IKRAM 2006).

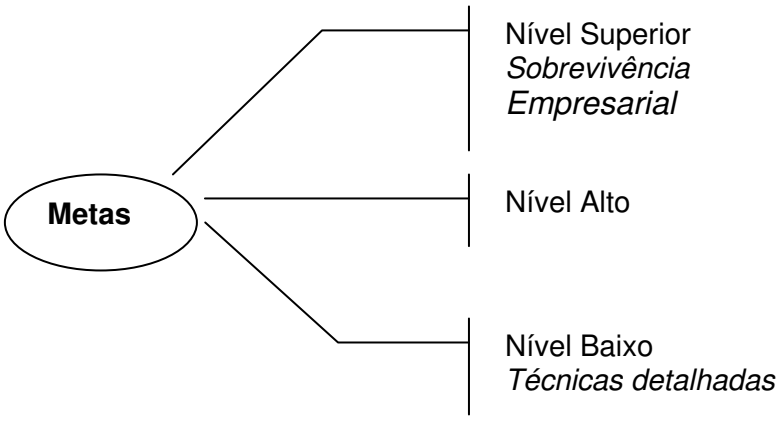


Figura 2.3 – Níveis de Abstração das Metas

A Figura 2.3 demonstra os níveis de abstração das metas. Metas de nível alto (preocupações estratégicas) referem-se a ambos os sistemas, englobam tanto o *software* e seu ambiente, como por exemplo, seres humanos, equipamentos e *software* (LAMSWEERDE, 2001). Metas de nível baixo são os refinamentos gerados pelas metas de nível alto; por exemplo, a meta de nível alto Usabilidade pode ser decomposta, gerando as mestas de nível baixo Facilidade de uso e *Help-Online.* A meta de nível superior, proposta por Regev e Wegmann (2005), é atribuída a uma empresa em relação à sua sobrevivência. Isto quer dizer que a sobrevivência está

atrelada a normas. Exemplos de tais metas dependem da natureza da empresa e do seu ambiente. Para uma empresa comercial, que busca tipicamente manter suas finanças em boas condições, uma norma poderia ser sua relação com outras empresas. Isto sugere uma meta de manutenção que poderia ser refinada alcançando submetas como: manter relacionamentos com clientes, fornecedores, funcionários e investidores.

3 VALIDAÇÃO DE REQUISITOS

Neste capítulo, abordaremos as principais definições, bem como as diretrizes para validação requisitos.

3.1 INTRODUÇÃO

Kotonia e Sommerville (1998) analisam que a validação é a última etapa da Engenharia de Requisitos.

Depois que o documento de requisitos for produzido, os requisitos devem ser validados formalmente. Este processo de validação está relacionado a conferir os requisitos, corrigindo conflitos e ambigüidades, visando assegurar que sigam normas de qualidade (SOMMERVILLE e SAWYER, 1997).

O objetivo da validação de requisitos é conferir os que estiveram elicitados e descobrir possíveis problemas com eles. O processo deve envolver *stakeholders* de sistema, engenheiros de requisitos e projetistas de sistemas (SOMMERVILLE e SAWYER, 1997).

Problemas de requisitos poderiam ser, segundo Sommerville e Sawyer (1997):

- Falta de conformidade para normas de qualidade;
- Requisitos pobremente formulados e que são ambíguos;
- Conflitos de requisitos que não foram descobertos durante o processo de análise.

Para Martins (2001), estes problemas devem ser resolvidos antes que o documento de especificação seja aprovado e utilizado para o desenvolvimento do sistema. Segundo Sommerville e Sawyer (1997), alguns problemas podem ser resolvidos e refeitos no documento de requisitos, entretanto, outras vezes será preciso voltar às fases iniciais da Engenharia de Requisitos.

Sommerville e Sawyer (1997) fazem uma distinção muito apropriada entre análise e validação de requisitos. Estas atividades têm muito em comum; elas envolvem

encontrar conflitos entre os requisitos de sistema. Porém, há uma diferença importante entre elas:

- Análise está preocupada com requisitos quando são extraídos de *stakeholders* de sistema. Os requisitos estão normalmente incompletos e são expressos dentro um modelo informal e não estruturado. Haverá uma mistura de anotações provavelmente para descrevê-los.

- Validação está relacionada a conferir o desenho final de um documento de requisitos que inclui: requisitos de sistema e a remoção de requisitos incompletos e inconsistentes. O documento e os requisitos devem seguir normas de qualidade definidas.

Ainda, segundo os autores, há dificuldade para a aplicação da validação, pois não existe um documento que sirva de base para a validação, nem uma maneira de demonstrar que uma especificação esteja correta com respeito a alguma outra representação do sistema (SOMMERVILLE e SAWYER, 1997).

Nuseibeh e Easterbrook (2000) afirmam que é uma atividade complexa, por dois motivos: o primeiro está na dificuldade de levantar todos os requisitos com o cliente; e o segundo, por razões sociais: existe uma dificuldade de consenso entre os diferentes responsáveis pelo *software*. Entretanto, é sabido que o processo de validação pode aumentar a confiança na especificação do sistema, que satisfará as reais necessidades do cliente. A validação da especificação assegura que o documento de requisitos representa descrição clara do sistema para *design* e implementação. É a última chance antes de entrar no processo de desenvolvimento, para considerar que os requisitos são aceitáveis a todos os proprietários dos sistemas.

Sempre há uma tendência natural para apressar o processo de validação de forma que o desenvolvimento de sistemas possa começar. Porém, não permitir tempo suficiente para validação implicará um desenvolvimento de sistemas caóticos e, quando estes problemas de requisitos emergirem, será preciso refazer todo o projeto (SOMMERVILLE e SAWYER, 1997).

Durante o processo de validação de requisitos, tipos diferentes de diretrizes são bem vindas para se obter um resultado satisfatório no documento de requisitos. Estas diretrizes incluem (SOMMERVILLE e SAWYER, 1997):

3.2 OS REQUISITOS SATISFAZEM PADRÕES ORGANIZACIONAIS

Antes de distribuir o documento de requisitos, este deve passar por uma revisão geral. Uma pessoa deve examinar os padrões para assegurar que a estrutura de documento e os requisitos definidos são consistentes.

Existem dois benefícios neste processo: o primeiro, custo baixo de trabalho, pois apenas uma pessoa examinará o documento. O segundo é que, conferindo-se o documento contra um padrão, pode-se revelar problemas de requisitos mais rapidamente. Se um documento de requisitos não estiver nos padrões definidos, pode indicar problemas com a especificação de requisitos, requerendo a intervenção de administração para a solução.

O processo deve ser feito por um analista, ou um engenheiro que está familiarizado com os padrões de requisitos, mas que não foi envolvido na especificação de requisitos do sistema. Não é necessário que o inspetor de documento entenda dos detalhes do requisito.

Se existe um padrão para requisitos, o analista ou o engenheiro deve confrontar o documento de requisitos com o padrão estabelecido. Esta verificação inicial também inclui conferir se todas as páginas no documento estão numeradas, todos os diagramas e figuras rotulados, e que foram completados todos os apêndices exigidos no documento.

Os custos de introdução e aplicação dessa diretriz são baixos. Uma vez definidos os padrões, o custo só aumentará quando o inspetor ou examinador não está familiarizado com o padrão, e tem que gastar algum tempo para entendê-lo. Contudo, é importante ressaltar que, se o documento de requisito for muito grande, esta verificação inicial não pode ultrapassar um dia.

Conforme ressalta Sommerville e Sawyer (1997), tendo os padrões definidos, é improvável que haja qualquer problema em introduzir esta diretriz, entretanto, pode

haver alguns problemas que se aplicam à diretriz, particularmente em organizações pequenas. Por dois motivos: dificuldade em encontrar um inspetor ou examinador para revisar o documento (independência é importante porque as pessoas que conhecem o documento leram e acham que o que está escrito está correto); e algumas pessoas podem não ser convencidas do valor deste tipo de verificação.

3.3 ORGANIZAR INSPEÇÕES FORMAIS DE REQUISITOS

Os requisitos devem ser validados por um grupo de pessoas que sistematicamente conferem, conhecem e discutem os problemas de requisitos, e concordam que eles possam ser resolvidos.

Inspeções formais são um modo efetivo de descobrir problemas em documentos de requisitos, documentação de usuário, projeto de *software* e programas.

A inspeção é uma reunião neutra para resolução de problema, na qual não deve haver nenhuma culpa por qualquer problema descoberto. Podem ser solucionados conflitos durante a inspeção, sem confrontações entre *stakeholders*.

Desta forma, deve ser presidida por alguém não envolvido na elicitação dos requisitos que estão sendo validados. Durante a reunião, um engenheiro de requisitos apresenta cada requisito e aguarda comentários e problemas levantados para discussão posterior. Uma pessoa do grupo deve ser nomeada para o papel de escriturário para anotar os problemas de requisitos identificados.

É importante ressaltar que uma inspeção de requisitos formal deve envolver uma equipe de inspetores de diferentes equipes, que leram o documento de requisitos, o registro de problemas levantados e os requisitos de sistema. Neste processo, os revisores podem usar uma lista de especificação para ajudar a focalizar a atenção deles em aspectos particulares do documento de requisitos.

Ao contrário de inspeções de programa, em que os erros são simplesmente comunicados ao autor do programa para correção, inspeções de requisitos envolvem um grupo, tomando algumas decisões sobre as ações que devem ser tomadas para corrigir os problemas identificados. Ações que podem ser decididas para cada problema são as seguintes:

- Clarificação de requisitos. O requisito pode ter sido mal expressado ou omitido acidentalmente durante a elicitação de requisitos. O autor deve melhorar o requisito reescrevendo-o;

- Falta de informações. Alguma informação está faltando no documento de requisitos, e é responsabilidade dos engenheiros de requisitos que estão revisando o documento descobrir esta informação de *stakeholders* de sistema, ou de outras fontes de requisitos;

- Requisitos conflitantes. Se há um conflito entre requisitos elicitados e os *stakeholders* envolvidos, é preciso negociar para solucionar o conflito;

- Requisito irreal. O requisito não parece ser implementável com as tecnologias disponíveis. As partes interessadas devem ser consultadas para decidir se deveria ser suprimido ou modificado para torná-lo mais realista.

Introduzir inspeções de requisitos envolve estabelecer ações que possam conduzir a inspeção e a formação das pessoas técnicas para o trabalho. Experiência anterior com inspeções é um fator essencial no processo de validação, entretanto, caso não exista uma equipe técnica com experiência, deve-se introduzir a inspeção de requisitos em um projeto piloto. Isto deverá minimizar os custos do processo de inspeção. O projeto piloto irá ajudar a determinar se este método é eficaz em termos de custos para a organização, e ajudar a julgar o melhor modo para estas revisões formais.

Este processo envolve custos moderados, mas resulta num significativo ganho de tempo no desenvolvimento do *software*. Inspeções requerem um grupo de pessoas com habilidades diferentes e responsabilidade para ler documento. Como as pessoas envolvidas podem trabalhar para organizações diferentes ou partes diferentes da mesma organização, este processo pode ser um pouco difícil, pois é praticamente impossível definir participação em inspeções de requisitos como parte do trabalho. Inicialmente, pelo menos, com o apoio dos gestores, será preciso invocar boa vontade de todos para participação na inspeção de requisito. Entretanto,

boa maneira de provar o valor da inspeção de requisito é realizar treinamento adequado e sensibilização das pessoas envolvidas no desenvolvimento do *software*.

3.4 USAR EQUIPE MULTIDISCIPLINAR

Os documentos de requisitos devem ser revisados por uma equipe multidisciplinar, envolvendo pessoas com diferentes conhecimentos. Devem incluir um usuário final de sistema, representante do cliente, um ou mais conhecedores do domínio, um ou mais engenheiros responsáveis pelo projeto e implementação do sistema, e um ou mais engenheiros de requisitos. Isto se aplica quando usamos inspeções de requisitos formais ou de um processo menos estruturado de revisão.

Os benefícios desta diretriz são:

- Pessoas com conhecimentos diferentes trazem habilidades e experiência à revisão;
- Se *stakeholders* do sistema forem envolvidos no processo de revisão dos requisitos, sentem-se prestigiados e, portanto, mais propensos a serem solidários com as alterações que lhes dizem respeito, porque irão saber das reais necessidades.

Não deve haver custos adicionais envolvidos na introdução e aplicação desta orientação, a menos que usem consultores externos para fazer a revisão. O grande problema é garantir um amplo espectro de envolvimento, pois muitas pessoas não estão interessadas em dispor do tempo de seus outros trabalhos para participar da revisão dos requisitos. Elas já podem ter sido movidas para outros projetos. Isto é particularmente provável sobretudo para os especialistas de negócio.

3.5 DEFINIR *CHEKLISTS* DE VALIDAÇÃO

Nesta diretriz, podemos definir uma lista de verificação que ajuda a focalizar a atenção dos examinadores de requisitos em atributos críticos do documento de requisitos. Estas listas identificam o que os examinadores devem procurar quando estiverem validando os requisitos de sistema.

Perguntas que poderão ser incluídas em um *checklist* de verificação:

- Os requisitos são completos, ou seja, o revisor tem conhecimento de algum requisito ou informação faltante na descrição do requisito?;

- Os requisitos são consistentes? As descrições de diferentes requisitos são contraditórias? Há contradições entre requisitos individuais ou nos requisitos gerais do sistema?;

- Os requisitos são compreensíveis, isto é, os leitores do documento podem entender o que significam? Esse é o mais importante atributo do documento de requisitos, se ele não pode ser compreendido, os requisitos não podem ser validados;

- Os requisitos são ambíguos, ou seja, existem diferenças nas possíveis interpretações? Leitores com diferentes conhecimentos podem ter diferentes interpretações dos requisitos;

- O documento de requisitos está estruturado, isto é, as descrições dos requisitos estão organizadas de modo que requisitos relacionados estejam agrupados? Seria uma estrutura alternativa para uma melhor compreensão;

- Os requisitos são rastreáveis, isto é, são claramente identificados, incluindo ligações com requisitos relacionados?.

O processo de validação por *checklist* deve ser expresso numa forma geral e compreensível para pessoas como o usuário final, que não é especialista de sistema. Regra geral, os *cheklist* de requisito não devem ser tão longos.

O perigo, naturalmente, é que o *checklist* de verificação se transforme em algo muito vago, por meio do qual seja impossível responder às questões de maneira útil. É preciso encontrar um equilíbrio direto entre a generalidade e o detalhe. Ao contrário das inspeções de programa, o *cheklist* de verificação está interessado em falhas

muito específicas; sendo assim, não é bom para inspeções dos requisitos por causa das diferenças entre os tipos de sistema.

O uso de um *checklist* bastante geral, com questões como as acima mecionadas, não é uma diretriz cara para se implementar. É necessário elaborar uma lista inicial com base na experiência das pessoas que tenham estado envolvidas na validação de requisitos.

3.6 PROTÓTIPOS PARA VALIDAR REQUISITOS

Protótipos são desenvolvidos para permitir uma melhor representação dos requisitos de um sistema. Normalmente, o usuário tem de esperar até etapas finais do processo de desenvolvimento para visualizar uma versão executável do sistema. Com o desenvolvimento de protótipos, os usuários podem ter uma idéia antecipada de como o sistema funcionará. Observa-se que a técnica de prototipação geralmente é usada para ajudar nas atividades de elicitação e análise de requisitos. Contudo, também pode ser usada para validá-los.

Após o documento de requisitos já ter sido definido e acordado, pode-se aperfeiçoar o protótipo desenvolvido na elicitação e análise, e utilizá-lo para validar os requisitos, com auxílio mais efetivo dos usuários e clientes.

Seguem abaixo as vantagens da construção de um protótipo:

- Extração dos requisitos;
- Validação dos requisitos;
- Ponto de auxílio na comunicação entre desenvolvedores e *stakeholders;*
- Treinamento de usuários;
- Demonstração do sistema operando (acompanhamento do trabalho frente ao usuário).

Há altos custos envolvidos na implantação desta diretriz, principalmente quando não se tem o *software* de prototipagem e experiência para utilizá-lo. Esses custos incluem desenvolvimento dos protótipos do sistema e de formação dos

desenvolvedores. Além disso, é muito difícil encontrar especialista em prototipagem; faltam pessoas qualificadas, com experiência nesta área.

3.7 CASOS DE TESTES PARA OS REQUISITOS

Propor testes possíveis é uma maneira eficaz de revelar problemas de requisitos, tais como incompletude e ambiguidade. Se houver dificuldades em derivar casos do teste para um requisito, há algum tipo do problema no requisito. Pode haver uma informação faltante ou a descrição do requisito não está claramente escrita. O conjunto proposto de casos do teste pode ser usado como base para o planejamento e a derivação dos casos reais do teste a ser aplicado no sistema final.

Para cada requisito apresentado no documento, é preciso analisá-lo e definir um teste que possa objetivamente verificar se o sistema o satisfaz. O objetivo de propor casos de teste para validar os requisitos está diretamente relacionado com os requisitos elicitados, e não com os do sistema. A preocupação não é com os aspectos práticos, tais como teste de custos, evitar testes redundantes, definição detalhada dos testes de dados, portanto, não é propor testes reais a serem aplicados no sistema definitivo.

Para definir os casos de teste, é preciso esclarecer algumas dúvidas sobre os requisitos:

- Qual cenário pode ser utilizado para verificar o requisito? Esta pergunta define o contexto em que o teste deve ser aplicado;

- O requisito, por si próprio, inclui bastante informação para permitir que um teste seja definido? Se não, que outros requisitos devem ser examinados para encontrar essa informação?;

- É possível verificar o requisito utilizando um único teste ou vários casos de teste? Para vários testes, pode significar que existe mais de um requisito embutido em uma única descrição.

Os custos iniciais para introduzir esta diretriz são baixos porque ela não requer nenhum investimento em tecnologia, nem em novos métodos. Porém, precisará de uma sessão de treinamento para demonstrar o seu valor.

Nesta diretriz, podemos encontrar alguma resistência porque os engenheiros de requisitos veem a derivação de casos do teste como responsabilidade de outra pessoa. É preciso deixar claro que não estão definindo os testes reais que serão aplicados ao sistema. Deve-se enfatizar que erros encontrados nos testes ajudam o processo da validação dos requisitos.

4 NFR-FRAMEWORK

4.1 INTRODUÇÃO

Não há como desconsiderar a importância dos requisitos no processo de desenvolvimento de *software*. Entretanto, muitas empresas não têm dado a devida atenção a este fato. Por outro lado, não há mais espaço para que o desenvolvimento continue sendo feito de forma artesanal, ou seja, confiando apenas na experiência e conhecimento técnico do desenvolvedor.

Além disso, engenheiros de requisitos e a maioria das linguagens existentes incidem sobre os requisitos funcionais, negligenciado ou esquecendo os requisitos não-funcionais na concepção do *software*. Os atributos de qualidade de *software* (requisitos não-funcionais), por exemplo, são vistos como conseqüência destas decisões, e não como algo que foi pensado. Para mudar este quadro e tratá-los coerentemente, são necessários não apenas métodos e representações, mas também sua reutilização em projetos de desenvolvimento de *software*.

Pesquisas indicam que técnicas baseadas em metas são mais adequadas para elicitar requisitos não-funcionais, pois possibilitam (CHUNG et al. , 2000; KAVAKLI e LOUCOPOULOS, 2003; CHUNG, 1991):

- Examinar a utilidade dos requisitos;
- Que o grafo SIG (*Softgoal Interdependency Graphs*) permita uma visão vertical desde a estratégia de alto nível até o detalhe;
- Que os operadores lógicos E e OU permitam uma melhor tomada de decisão;
- Que a formalização permita provar a correção e completude dos requisitos não-funcionais.

Chung (2000) propôs o *NFR-Framework*. É uma abordagem aceita, pois se ajusta a todos os tipos de requisitos não-funcionais, desde as primeiras etapas do processo de desenvolvimento de *software*.

4.2 VISÃO GERAL DO *NFR-FRAMEWORK*

O *NFR-Framework* contribui para que os desenvolvedores tratem os requisitos não-funcionais de modo a expressá-los sistematicamente, e usá-los para guiar o processo de desenvolvimento de *software* racionalmente.

O *NFR-Framework* utiliza uma abordagem voltada a procedimentos para requisitos não-funcionais, nos quais os RNFs possam ser representados como metas a serem atingidas. Uma característica importante é que as metas contribuirão positiva ou negativamente, e muitas vezes apenas parcialmente, no sentido de uma determinada meta. Portanto, metas de requisitos não-funcionais podem não ser satisfeitas. Assim, o termo "satisfação da meta" sugere o que é esperado do *software*, dentro de limites aceitáveis para os requisitos não-funcionais. Essa abordagem permite que os aspectos de subjetividade, relatividade e interatividade inerentes aos requisitos não-funcionais sejam mais compreensíveis. Desta forma, CHUNG (2000) propõe uma noção de *Softgoal* para descrever as metas que precisam ser satisfeitas (CHUNG et. al. 2000).

O *NFR-Framework* consiste em cinco componentes principais: S*oftgoals*, interdependências, um procedimento de avaliação, métodos de refinamento e regras de correlação (LAMESWEERD et. al., 2001).

4.2.1 *SOFTGOALS*

São as unidades básicas para representar requisitos não-funcionais. S*oftgoals* ajudam os desenvolvedores a lidar com RNFs, que podem ter natureza subjetiva, relativa ou interativa. São três os tipos de s*oftgoals*:

- *Softgoal*s de RNFs representam requisitos não-funcionais a serem satisfeitos. Elas agem como restrições gerais no sistema;

- *Softgoals* de Operacionalização ajudam a satisfazer RNFs, fornecendo mecanismos mais concretos no sistema. Elas fornecem operações, processos, representação de dados, estruturação, limitações e agentes no sistema para satisfazer as necessidades declaradas nas s*oftgoals* de RNFs;

- *Softgoals* Declarativas ajudam a justificar decisões. Justifica e explica a lógica do contexto de uma *softgoal* ou a ligação de interdependência.

Figura 4.1 - Representação Gráfica da Softgoal

4.2.2 INTERDEPENDÊNCIAS

São interrelações entre os refinamentos de *softgoals* no sentido da satisfação de *softgoals* relacionadas.

Existem três tipos de refinamentos:

- Decomposição - Uma *softgoal* é refinada em outra da mesma natureza;
  Exemplo de Decomposição: vamos considerar como requisito “manter as contas com boa segurança”.

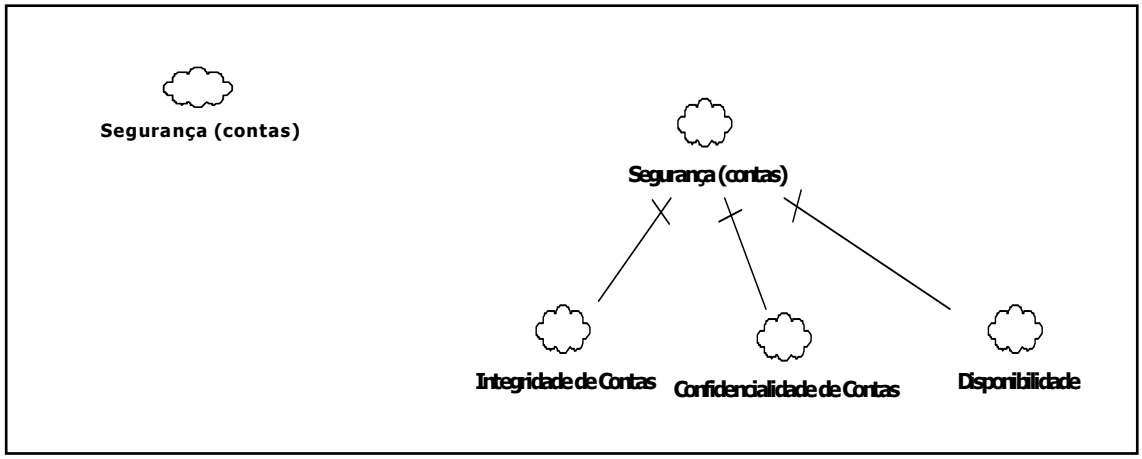


Figura 4.2 – Decomposição de *Softgoal* Adaptado de Chung (2000)

A Figura 4.2 demonstra um exemplo de decomposição, a *Softgoal* Segurança de Contas foi refinada em outras como: Integridade de Contas, Confidencialidade de Contas e Disponibilidade;

- Operacionalização - O RNF é refinado em s*oftgoals* de operacionalização; isto corresponde a uma transição crucial, na qual os RNFs que serão alcançados (confidencialidade de contas) são decompostos em *softgoals* operacionalizantes que podem alcançar o RNFs.
  Exemplo de Operacionalização: vamos considerar o requisito “manter as contas com boa segurança”.

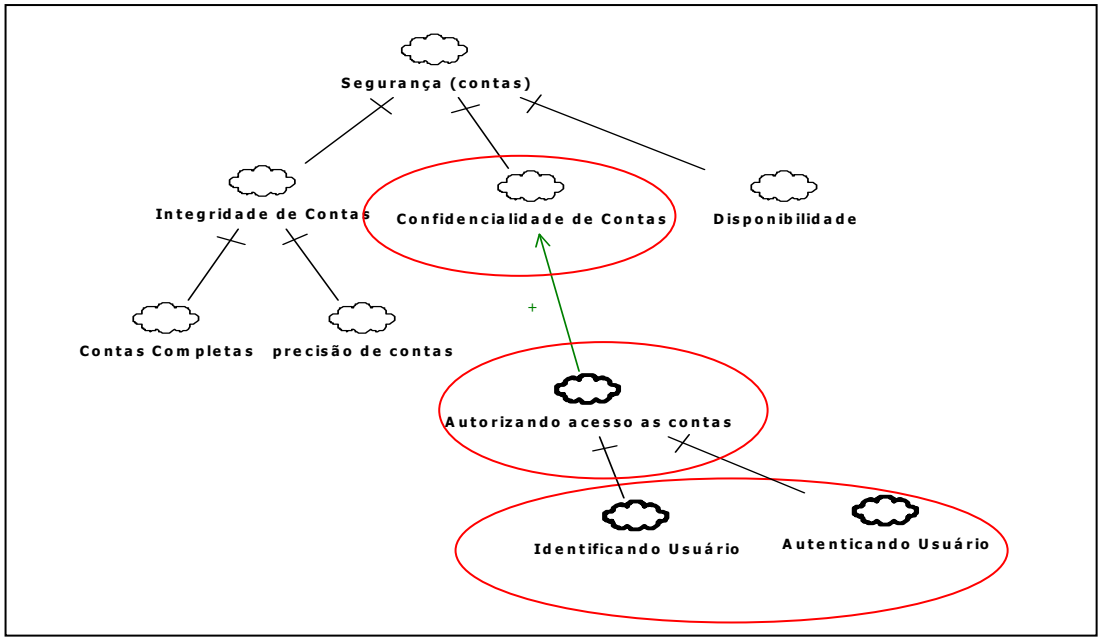


Figura 4.3 – Operacionalização de *Softgoal* adaptado de Chung (2000)

A Figura 4.3 demonstra um exemplo de operacionalização, a *Softgoal* Segurança de Contas foi refinada em outras como: Integridade de Contas, Confidencialidade de Contas e Disponibilidade. Confidencialidade de contas foi operacionalizada em “Autorizando acesso às contas numa interdependência positiva”, que por sua vez foi operacionalizada em “Identificando e Autenticando usuários”;

- Argumentação - Razões de projeto são anotadas através de argumentação. Elas são rastreadas por refinamento, geralmente envolvendo *softgoals*.
  Exemplo de Argumentação: consideraremos o requisito “manter as contas com boa segurança”.

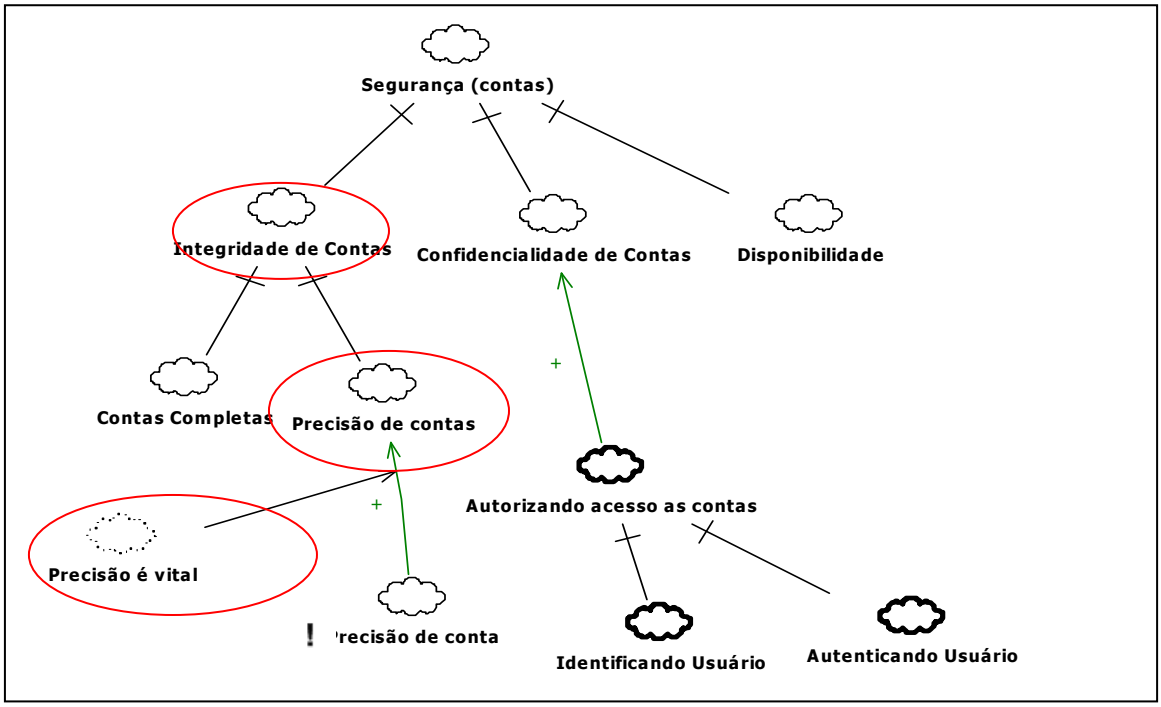


Figura 4.4 – Argumentação de *Softgoal* Adaptado de Chung (2000)

A Figura 4.4 demonstra um exemplo de Argumentação, a *Softgoal* Segurança de Contas foi refinada em outras como: Integridade de Contas, Confidencialidade de Contas e Disponibilidade. Integridade de contas foi refinada em Contas completas e Precisão de contas. Precisão de contas foi refinada em outra *Softgoal* de mesmo nome, indicando priorização, numa relação positiva; nesta relação é usada a argumentação “Precisão é vital”, alertando o desenvolvedor para este item.

4.2.3 TIPOS DE CONTRIBUIÇÕES

Um grupo de metas pode, através do E ou OU, contribuir para suas correlatas.

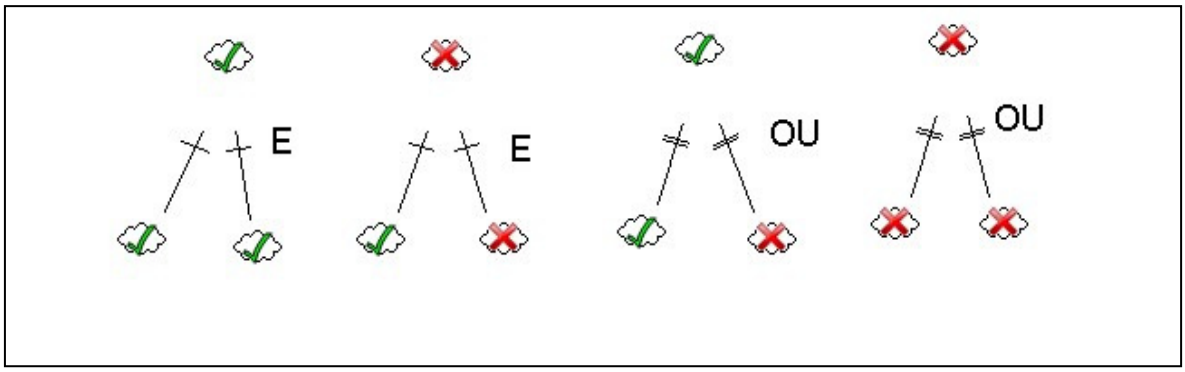


Figura 4.5 – Tipos de Contribuições E e OU Adaptado de Chung (2000)

- Um único descendente pode contribuir para o seu “pai” nos seguintes níveis: totalmente positiva (“++” ou Make), parcialmente positiva (“+” ou Help), parcialmente negativa (“ - “ ou Hurt), totalmente negativa (“- - “ ou Break).

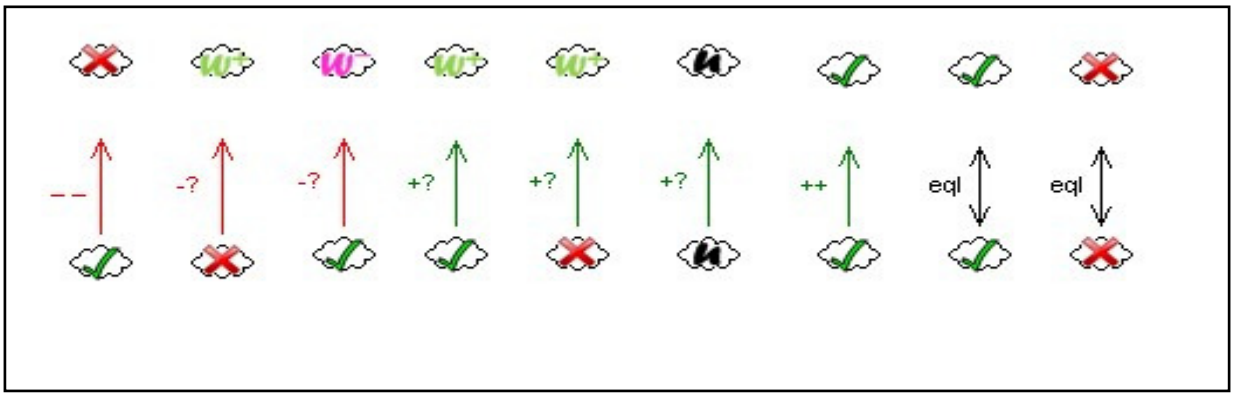


Figura 4.6 - Grupo de Metas Make, Help, Hurt e Break Adaptado de Chung (2000)

4.2.4 PROCEDIMENTO DE AVALIAÇÃO

Este procedimento determina o grau em que determinado requisito não-funcional é satisfeito por um conjunto de decisões de projeto. Os nós da folha do gráfico de interdependência de S*ofgoals* são rotulados de acordo com seu grau de satisfação: satisfeito, negado, conflitante, fracamente negado, fracamente satisfeito ou pendente. Os rótulos são propagados até o topo de S*oftgoals* em duas etapas.

Em primeiro lugar, o impacto individual de cada contribuição em direção ao “pai” é avaliado. Posteriormente, todos os contribuintes para certo “pai” são combinados em um único rótulo. Por isso, o processo de avaliação é útil na seleção entre os refinamentos alternativos. Na presença de alternativas competitivas, os desenvolvedores podem usar o procedimento para ver qual o impacto de uma seleção particular nas S*oftgoals*. Se uma seleção influencia negativamente uma *softgoal* importante, os desenvolvedores podem simplesmente rejeitá-la, escolher outra e novamente usar o procedimento para ver o impacto da nova escolha. Ao repetir esse processo, os desenvolvedores podem ajudar a fazer uma escolha assinalando o maior benefício com as trocas aceitáveis. Chung (2000) propõe usar catálogos de avalição para ajudar nesse processo, conforme a Figura 4.7.

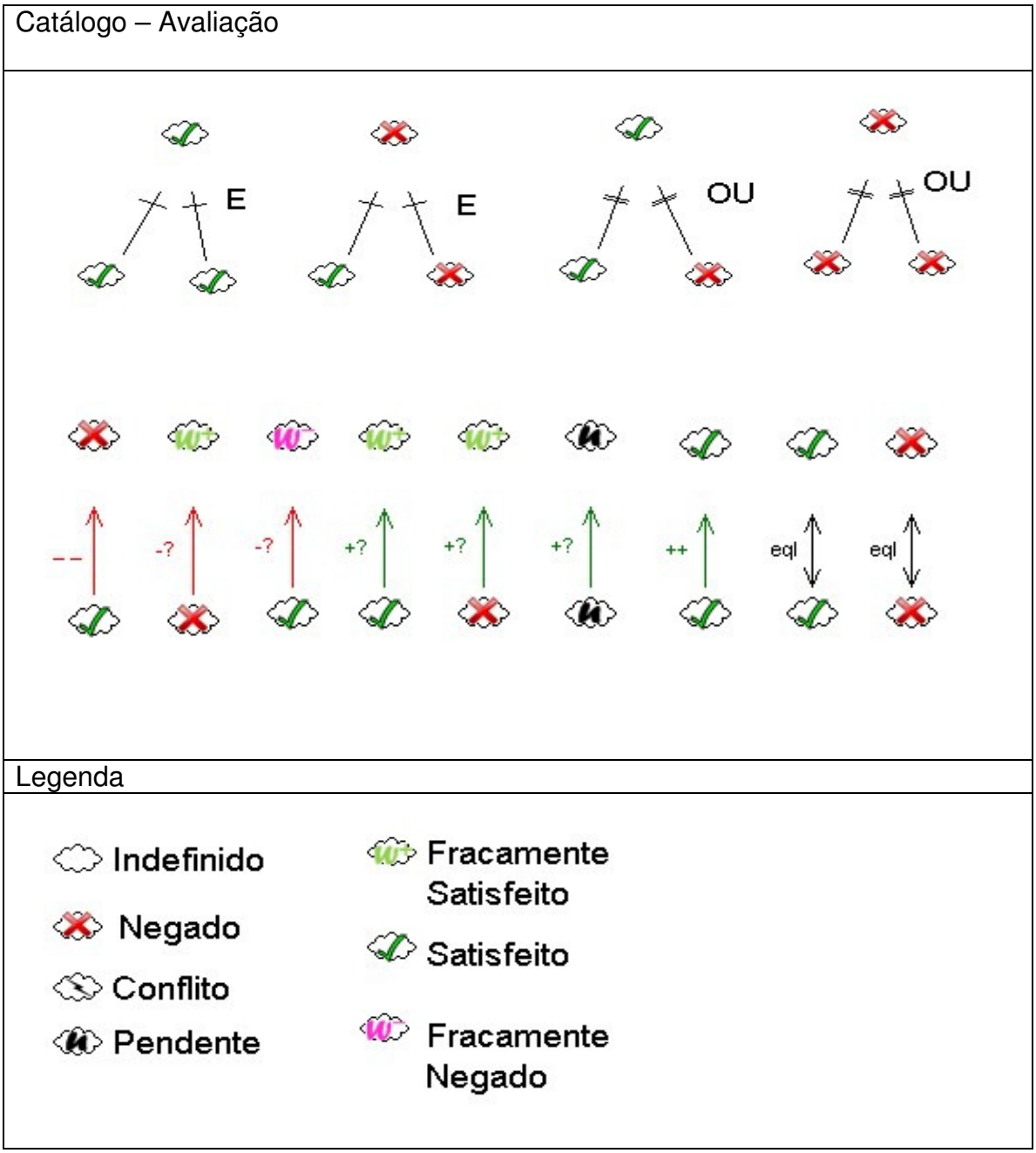


Figura 4.7 - Catálogo de Avaliação Completo Adaptado de Chung (2000)

4.2.5 MÉTODOS DE REFINAMENTO

Os métodos de refinamento oferecem catálogos de técnicas de desenvolvimento para refinamento de *softgoals*. Tais métodos correspondem a "padrões de refinamento". Durante o processo de desenvolvimento, *softgoals* podem ser refinadas pelo desenvolvedor de uma forma *ad hoc*. Esta tarefa é demorada e torna difícil o conhecimento do domínio compartilhado, prorrogado, adaptado, ou reutilizado. Métodos fornecem refinamentos genéricos que podem ser

exemplificados em uma S*oftgoal*. Eles podem ser retirados de um catálogo que é parte do *NFR-Framework*.

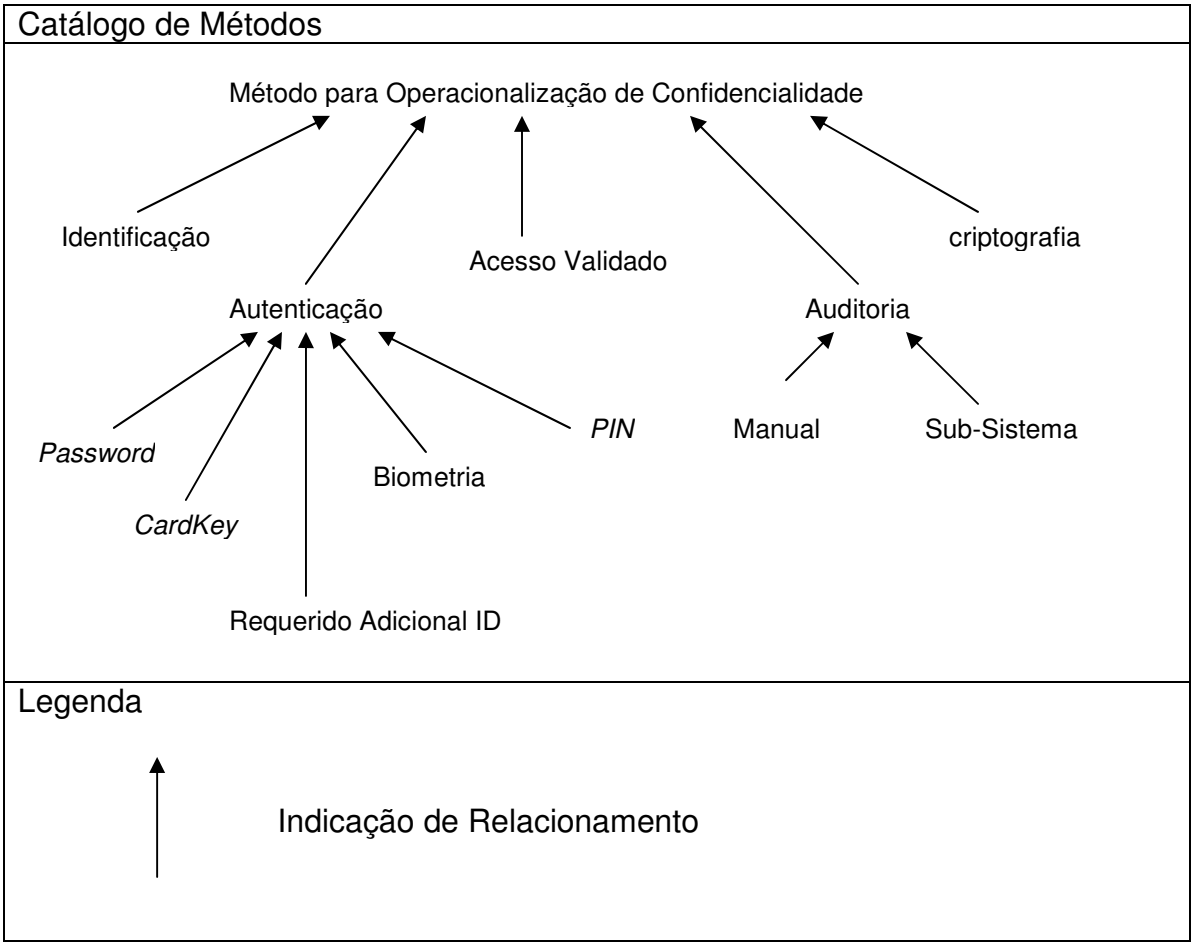


Figura 4.8 - Catálogo de Operacionalização de Métodos para Alcançar Confidencialidade Adaptado de Chung (2000)

4.2.6 REGRAS DE CORRELAÇÃO

Elas fornecem catálogos que permitem inferir possíveis interações, tanto positivas quanto negativas, entre *softgoals*. Os requisitos não-funcionais estabelecidos para um sistema podem estar em conflito ou em harmonia um com o outro. O conflito é representado por correlações negativas, e a harmonia, pelas correlações positivas. Assim como os métodos de refinamentos, as correlações permitem capturar conhecimento sobre interações genéricas entre S*oftgoals*, para codificar tal conhecimento, e compilá-lo em um catálogo. Inicialmente coletados da literatura e da experiência da indústria, as regras de correlação podem ser partilhadas, extendidas, ajustadas e reutilizadas. Detectar correlações negativas pode ser visto como uma troca reveladora, pois pode ser boa para uma *softgoal* ou ao mesmo tempo ruim

para outra. A tabela 4.1 exemplifica o catálogo de correlação adapatdo do grafo SIG da Figura 4.9 (CHUNG2000).

Tabela 4.1 - Catálogo do Impacto das *Softgoals* de Operacionalização sobre as *Softgoal* de RNFs

Catálogo de Correlação					
Softgoals de operacionalização Filhos	Softgoal de RNFs				
	Precisão	Confidencialidade	Equipe Responsável	Espaço	Usabilidade
Validação	+	+	-		
Compressão			-	+	
Indexação			+		
Autorização		+			
ID adicional		+			-

Uma questão importante dessa abordagem é que as *softgoals* possuem a propriedade de interagir entre si, em conflito ou sinergia (CHUNG, 2000). As *softgoals* são decompostas em outras, e representadas em uma estrutura gráfica inspirada nas árvores E ou OU para solução de problemas; esses grafos são chamados SIG (*Softgoal Interdependency Graphs*). A Figura 4.9 apresenta um exemplo do SIG adaptado de Chung (2000).

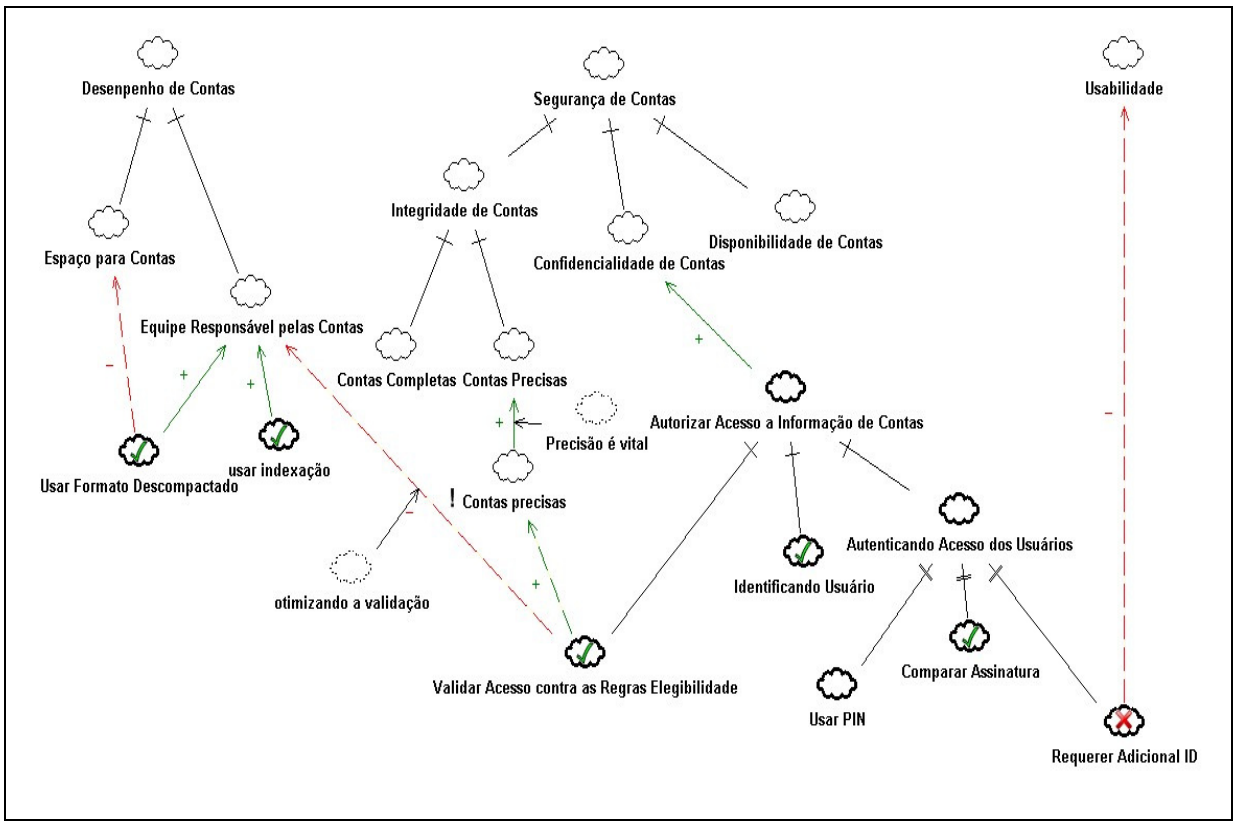


Figura 4.9 – Exemplo de Grafo SIG Adaptado de Chung (2000)

5 PROCESSO DE VALIDAÇÃO DE REQUISITOS NÃO-FUNCIONAIS

A Engenharia de Requisitos é a base do processo de desenvolvimento de *software*, entretanto, apesar de todas as ferramentas e metodologias disponíveis para suporte, sente-se a falta de métodos adequados para a validação dos requisitos. Assim, procura-se inserir nesse processo atividades para a validação de requisitos, com o intuito de tornar mais sistemática esta etapa. Para isso, é proposto um conjunto de atividades baseadas no *NFR-Framework* a fim de auxiliar os engenheiros de requisitos a efetuar um refinamento dos requisitos não-funcionais do sistema, identificando propriedades que anteriormente estavam obscuras.

5.1 ATIVIDADES DO PROCESSO DE VALIDAÇÃO DOS REQUISITOS NÃO- FUNCIONAIS

As atividades do processo de validação dos requisitos não-funcionais são apresentadas na Figura 5.1. Estas 6 (seis) atividades se complementam para o processo de validação de RNFs. As descrições de cada atividade proposta e seus respectivos produtos serão apresentadas nas próximas seções.

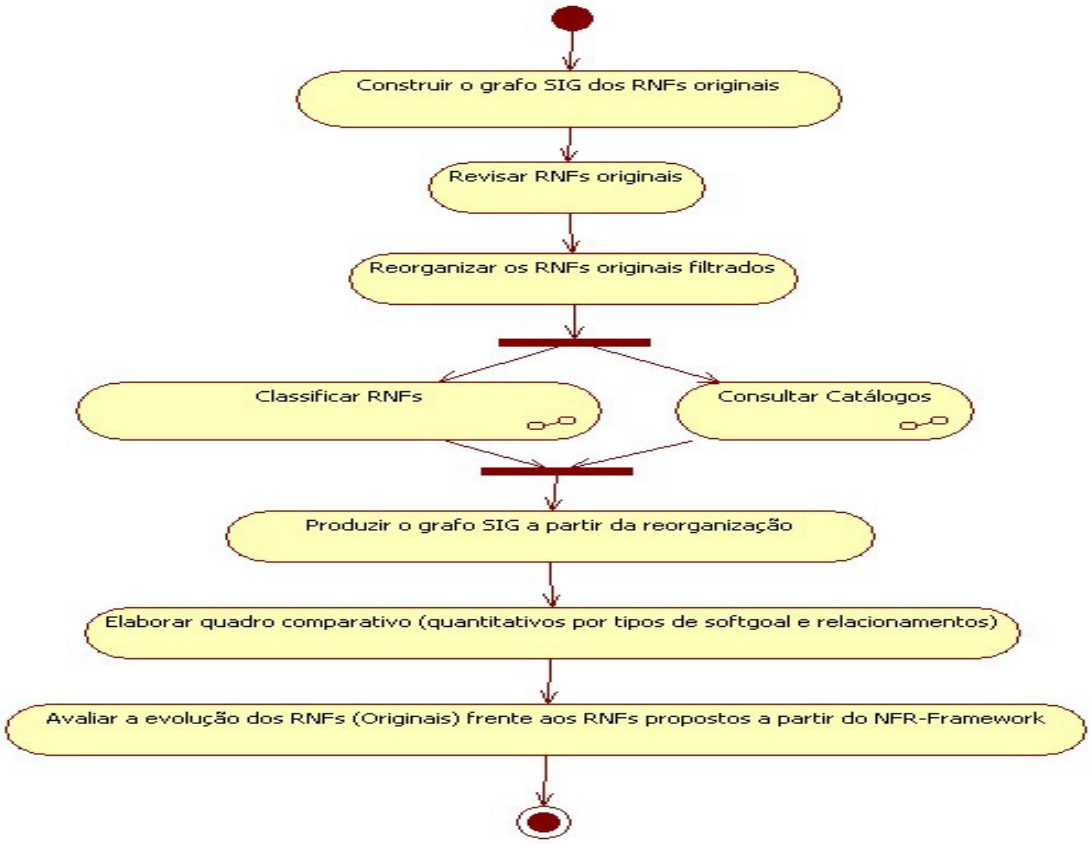


Figura 5.1 - Atividades do Processo de Validação de Requisitos Não-Funcionais

5.1.1 CONSTRUIR O GRAFO SIG DOS RNFS ORIGINAIS

Nesta atividade, o engenheiro de requisito deve identificar a lista dos requisitos não-funcionais e, a partir deles, construir o grafo SIG, usando como sugestão a ferramenta STARUML, pois é de fácil aprendizado e, consequentemente, fácil utilização. Isto permite que o engenheiro de requisito possua uma estrutura hierárquica que permita a visualização completa de tais requisitos.

Esta atividade servirá de base para a identificação dos requisitos elicitados, relacionados e possíveis dependências. A construção do grafo SIG deve ser feita com base no *NFR-Framework*, que consiste em cinco componentes principais: *softgoals*, interdependências, procedimento de avaliação, métodos de refinamento e regras de correlações. *Softgoals* são usadas para representar requisitos não-funcionais, e compostas por três tipos de *softgoals*: *softgoals* de requisitos não-funcionais, *softgoals* de operacionalização, e *softgoals* declarativas. Esta atividade é importante porque através da operacionalização de *softgoals*, se estabelece a representação concreta da concepção do *software,* ou execução das soluções (por exemplo, operações, processos, dados e representações) obtidas como resultado de decisões tomadas para a *softgoal* de requisitos não-funcionais. O capítulo 3 desta dissertação fundamenta em detalhes o *NFR-Framework* para execução desta atividade. Abaixo segue entrada, processo e saída desta atividade conforme Figura 5.2.

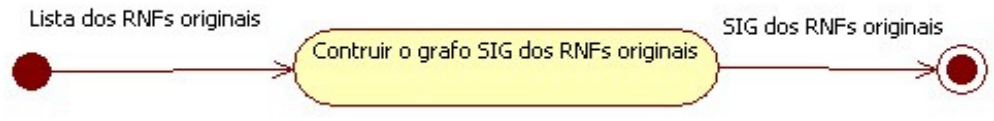


Figura 5.2 - Atividade para Construir o Grafo SIG Original

5.1.2 REVISAR RNFS ORIGINAIS

Esta é uma fase de inspeção do documento de requisito. Nesta atividade, o engenheiro de requisito deve avaliar se os RNFs elecitados são efetivamente requisitos não-funcionais. É extremamente importante, devido à diversidade e à divergência de conceitos utilizados nas definições dos requisitos não-funcionais, as quais conduzem a definições que contêm elementos errôneos, como por exemplo:

"Internacionalização" foi definida em estudo de caso como um requisito não-funcional, com o objetivo de expor a necessidade de o sistema ser em português e inglês. Entretanto, na literatura, a nomenclatura correta é "Usabilidade". A Figura 5.3 demonstra as entradas, o processo cujo objetivo principal é remover da lista os requisitos não-funcionais identificados erroneamente e, por fim, a saída com a lista dos RNFs revisados.

Figura 5.3 - Atividade de Revisão dos RNFs Originais

5.1.3 REORGANIZAR OS RNF ORIGINAIS REVISADOS

Propomos nesta atividade que a reorganização dos requisitos não-funcionais siga a classificação descrita por Sommerville (2001), que os classifica em Requisitos do Processo, Requisitos de Produto e Requisitos Externos. Conforme Figura 5.4 abaixo.

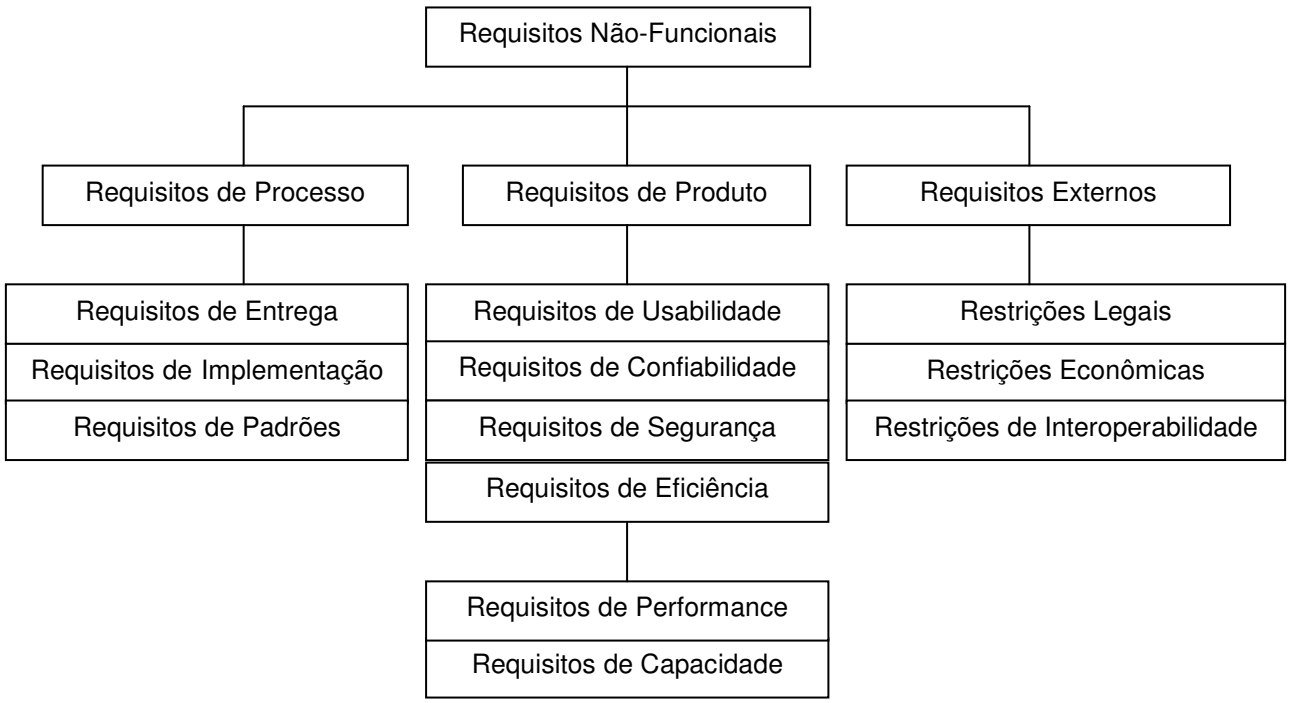


Figura 5.4 - Clasificação dos RNFs Segundo Sommerville (2001)

A reorganização dos RNFs deve ser feita paralelamente, usando também os catálogos de RNF (Anexos A, B, C e D), para um melhor entendimento dos termos relacionados, conseqüentemente, ter maior consistência na nova organização dos

mesmos. Esta atividade está clarificada na Figura 5.5, na qual entradas são os RNFs revisados, utilizando como apoio os catálogos de RNFs no processo, e como saída a geração de uma nova lista de RNFs.

Figura 5.5 - Atividade de Reorganização dos RNFs Originais Revisados

5.1.4 PRODUZIR O GRAFO SIG A PARTIR DA REORGANIZAÇÃO

Após o refinamento dos RNFs, o engenheiro de requisito deve construir um novo grafo SIG, cujo principal objetivo é demonstrar que os requisitos não-funcionais reorganizados proporcionam uma visão mais realista do sistema. Nesta atividade, podemos também armazenar as considerações do desenvolvedor sobre os requisitos não-funcionais, e mostrar a interdependência entre eles. Um outro fator relevante está relacionado aos conflitos entre *softgoals*. Quando se detecta um desses conflitos, é preciso que se tenha uma negociação para a resolução, que implica em selecionar alternativas, reavaliar outras prioridades, ou até mesmo revisar os requisitos. Embora apenas as *softgoals* de operacionalização sejam refletidas na concepção e na execução de um desenvolvimento de *software*, todas as *softgoals* que participam da construção do SIG podem aparecer na documentação de apoio para posterior desenvolvimento. Conforme a Figura 5.6, a entrada desta atividade são os RNFs reorganizados, tendo como processo o grafo SIG, e como saída o grafo SIG dos RNFs propostos.

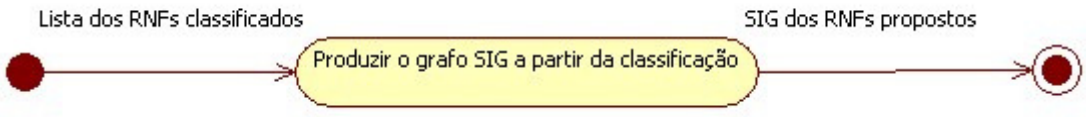


Figura 5.6 - Atividade para Produzir do Grafo SIG

5.1.5 ELABORAR QUADRO COMPARATIVO (QUANTITATIVOS POR TIPOS DE *SOFTGOAL* E RELACIONAMENTOS)

Nesta atividade, o engenheiro de requisito demonstra, através de quadro comparativo (Tabela 5.1), que as *softgoals* da proposta ultrapassam em números os

originais. Adicionalmente, poderá ser gerado gráfico de colunas que demonstra o quanto as *softgoals* da proposta ultrapassam os originais. O quadro comparativo de elementos conceituais do *NFR-Framework* é composto por: S*oftgoal* de Requisitos Não-Funcionais, Sub-*Softgoal, Softgoal* de Operacionalização, *Softgoal* Declarativa, Relacionamento de Correlações Positivas e Relacionamento de Correlações Negativas. A Figura 5.7, podemos observar a entrada dessa atividade, produzindo um quadro comparativo apontando diferenças entre os SIG originais, elaborados na primeira atividade, e os propostos produzidos na quinta atividade.

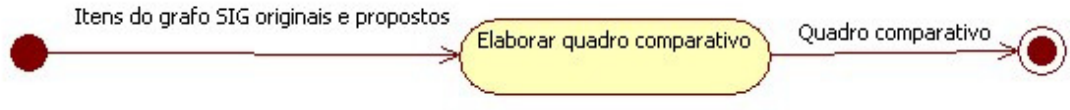


Figura 5.7 - Atividade para Elaborar Quadro Comparativo

Tabela 5.1 - Quadro Comparativo por Tipos de *Softgoal* e Relacionamentos

	Softgoal de Requisitos não-funcionais	Sub-Softgoal	*Softgoal* de Operacion-alização	*Softgoal* Declarati-va	Relacionamento de Correlações Positivas	Relacioname-nto de Correlações Negativas	Totais
Original							
Proposta							

5.1.6 AVALIAR A EVOLUÇÃO DOS RNFS (ORIGINAIS) FRENTE AOS RNFS PROPOSTOS A PARTIR DO *NFR-FRAMEWORK*

Um indicador é um dado numérico, expresso em uma unidade de medida, ao qual se atribui um valor percentual, com a finalidade de apoiar a avaliação. Nesta etapa, o engenheiro de requisito deve utilizar a fórmula proposta para averiguar a Evolução do RNFs (E_{RNF}) originais frente aos propostos. Esta atividade se dará por inspeção, pois o engenheiro contará os itens dos grafos originais e dos propostos. Fundamentalmente, procura-se identificar os itens que aparecem nos propostos, mas não nos originais.

Para se calcular a Evolução do RNFs (E_{RNF}), o engenheiro de requisito deverá levantar o número Total de Elementos Conceituais Propostos Novos (TECPNO) e dividi-lo pelo Total de Elementos Conceituais Originais Revisados (TECOR) somados ao Total de Elementos Conceituais Propostos Novos (TECPNO). Para a elaboração desta atividade, o engenheiro de requisito deve usar a fórmula (Figura 5.8) para se obter o valor percentual. Após esta etapa, o engenheiro deve usar a tabela 5.2 para analisar a evolução dos RNFs originais frente aos propostos, mas também avaliar a qualidade da especificação original. Por exemplo, se a Evolução de RNF (E_{RNF}) for muito baixa, significa que não houve uma evolução dos RNFs originais frente aos propostos; por outro lado, demonstrará que a Qualidade da Especificação Original (QEO) é muito alta.

Fórmula

$$E_{RNF} = \frac{TECPNO}{TECOR + TECPNO}$$

Figura 5.8 - Fórmula de Evolução dos RNFs.

Legenda
E_{RNF} – Evolução dos Requisitos Não-Funcionais
TECPNO - Totais de Elementos Conceituais Propostos Novos
TECOR - Totais de Elementos Conceituais Originais Revisados

Tabela 5.2 - Indicadores de Evolução dos RNFs Proposto e a Qualidade da Especificação Original

E_{RNF}	**VP**	**QEO**
Muito Baixa	**VP ≤ 20%**	Muito Alta
Baixa	**20% < VP ≤ 40%**	Alta
Média	**40% < VP ≤ 60%**	Média
Alta	**60% < VP ≤ 80%**	Baixa
Muito Alta	**VP > 80%**	Muito Baixa

Legenda
E_{RNF} – Evolução dos Requisitos Não-Funcionais
VP – Valor Percentual
QEO – Qualidade da Especificação Original

6 ESTUDOS DE CASO

Este capítulo apresenta três estudos de caso, com o objetivo de utilizar as seis atividades propostas no processo de validação de requisitos. Neles foi avaliada a qualidade da especificação dos requisitos não-funcionais. Os estudos seguiram passo a passo as atividades propostas. O primeiro foi realizado em um documento de requisitos especificado para um *software* de gerenciamento de requisitos; o segundo refere-se a uma especificação para elicitação de requisitos utilizando a META e, por último, a documentação dos requisitos do sistema de gestão de *ranking* e chaves de torneios de tênis.

Antes de iniciar os estudos de caso foram elencadas duas questões a serem perseguidas:

1. As atividades propostas servem como instrumento de melhoria de qualidade no processo de validação de requisitos não-funcionais;
2. As atividades propostas auxiliam os Engenheiros de Requisitos a efetuar um refinamento dos requisitos não-funcionais, identificando propriedades do sistema que anteriormente estavam obscuras.

Para a construção do grafo SIG foi utilizada uma ferramenta que apóia a notação do *NFR-Framework*. A ferramenta foi escolhida porque é *open-source*, e chamada *StarUML*. A ferramenta funciona apenas em plataformas *Windows*. A atual *StarUML* foi usada a versão 5.0 com o sistema operacional *Windows XP*. Nas próximas seções, descreveremos estas especificações e os resultados obtidos com as atividades propostas.

6.1 FERRAMENTA AUTOMATIZADA PARA GERENCIAMENTO DE REQUISITOS

A seguir abordaremos o primeiro estudo de caso dando uma visão geral da Ferramenta SIGERAR, em seguida utilizaremos as seis atividades proposta no processo de validação dos requisitos não-funcionais.

6.1.1 VISÃO GERAL DA FERRAMENTA SIGERAR

A ferramenta automatizada para gerenciamento de requisitos foi desenvolvida na UNIMEP com o objetivo de coletar, armazenar e manter os requisitos acordados, durante todo o ciclo de vida do *software*, gerenciando as mudanças ocorridas nos requisitos, permitindo rastrear os relacionamentos entre os requisitos, e entre os requisitos e documentos produzidos no processo de desenvolvimento do sistema. A ferramenta organiza o controle das mudanças, permitindo subsídios para a análise de impacto e custos, em tempo e dinheiro, que estas mudanças trarão para a organização (GRANDE e MARTINS, 2006).

A elicitação, análise e validação dos requisitos para o desenvolvimento da ferramenta foram efetuadas junto a uma equipe formada por analistas de sistemas, coordenadores e gerentes de projetos, oriundos de empresa prestadora de serviços médicos e por fornecedora de *software* de gestão ligada à área de saúde, que prestaram apoio integral ao desenvolvimento da ferramenta (GRANDE e MARTINS, 2006). A seguir, será apresentada a lista de requisitos não-fucionais especificada na documentação da ferramenta.

6.1.2 LISTA DOS REQUISITOS NÃO-FUNCIONAIS

Ao inspecionarmos o documento de requisitos, foram identificados cinco RNFs.

R1 – O sistema deverá prever mecanismos para controlar o acesso ao sistema (*login* e senha).

R2 – O sistema deverá ter a figura de um Administrador responsável por todo o controle e gerência do sistema, promovendo a manutenção de todos os cadastros básicos.

R3 – Deverá haver controle interno no sistema para que o mesmo seja abandonado automaticamente, caso o usuário deixe de usá-lo por dez minutos. A tela inicial de *login* deverá ser apresentada para continuidade do trabalho suspenso.

R4 – O sistema deverá ter portabilidade. Assim, deverá ser desenvolvido em linguagem e plataforma adequada (ex: Java).

R5 – O sistema deverá prever interoperabilidade, de forma a promover intercâmbio com outros sistemas ou *software* de apoio.

6.1.3 MODELAGEM DOS REQUISITOS NÃO-FUNCIONAIS (ORIGINAL)

O grafo SIG da Figura 6.1 demonstra que os requisitos não-funcionais precisam ser melhor elaborados.

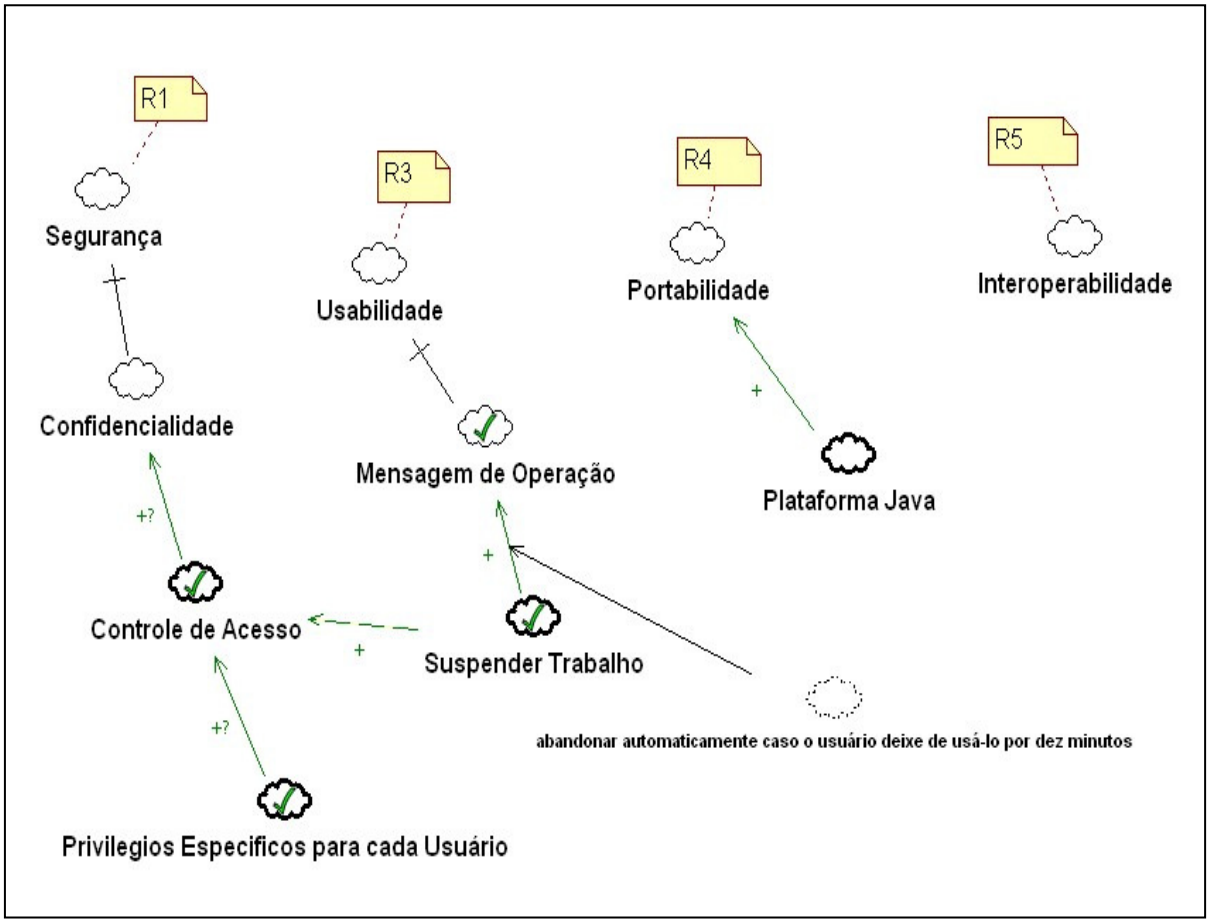


Figura 6.1 - Grafo SIG com os RNFs Originais (Estudo de Caso 1)

Neste estudo de caso existe pouca operacionalização e interdependência, conseqüência do pequeno número de RNFs e de uma descrição pobre dos mesmos. Para resolver estas questões, há a necessidade de se pesquisar conceitos e catálogos (consultar Anexo A, B, C e D) para um levantamento mais adequado ao projeto de software. Por exemplo, RNFs como usabilidade, segurança, performance, consistência, e disponibilidade são extremamente importantes para aplicações WEB; entretanto, muitos não foram mencionados na lista de requisitos não-funcionais. A idéia principal é demonstrar que não houve uma atenção especial aos requisitos não-funcionais. Sendo assim, para atingirmos todos os objetivos deste estudo de caso (que são: a construção do grafo SIG, organização dos RNFs revisados, comparação dos grafos SIG originais e propostos, construção do quadro comparativo e análise da evolução dos RNFs originais frente aos propostos), foi

preciso analisar os RNFs de forma mais precisa, acarretando um acréscimo na lista de requisitos não-funcionais do sistema proposto.

6.1.4 NOVA ORGANIZAÇÃO DOS REQUISITOS NÃO-FUNCIONAIS (PROPOSTA)

A nova organização dos requisitos não-funcionais seguiu a classificação, descrita por Sommerville (2001), em Requisitos do Processo, Requisitos do Produto e Requisitos Externos. Além disso, o engenheiro de requisito pode usar os catálogos para organizar e entender os requisitos não-funcionais.

Classificação dos Requisitos Não-Funcionais

Requisitos do Processo

RNF001	**Desenvolvimento do sistema** - em linguagem *Java* com *JSP (Java Server Pages)*

Requisitos do Produto

RNF002	**Segurança – Confidencialidade** O sistema deverá prever mecanismos para controlar o acesso ao sistema (*login* e senha).
RNF003	**Segurança – Disponibilidade** O sistema deverá estar disponível 24 horas por dia, 7 dias por semana.
RNF004	**Segurança – Integridade** Os dados inseridos e consultados devem ser consistentes e sincronizados. Realizar *backup* de segurança.
RNF005	**Usabilidade** Deverá ter facilidade de uso, *help-online* e mensagens de operações Deverá haver controle interno no sistema para que o mesmo seja abandonado automaticamente, caso o usuário deixe de usá-lo por dez minutos. A tela inicial de *login* deverá ser apresentada para continuidade do trabalho suspenso.
RNF006	**Confiabilidade** Sistema sempre disponível.
RNF007	**Performance** Deverá manter compacto o armazenamento de dados no servidor dedicado, tempo de resposta satisfatório para as transações(regras de otimização).
RNF008	**Banco de Dados - SGBD (Sistema Gerenciador de Banco de Dados) Firebird 1.5.**
RNF009	**Utilizar UML para modelagem**
RNF010	**Servidor Web (Apache Server) que gerencia as requisições vindas da Internet.**
RNF011	**Servidor de regras de negócio (Jakarta Tomcat) que gerencia o acesso às informações.**

Requisitos Externos

RNF012	**Interoperabilidade** O sistema deverá prever interoperabilidade, de forma a promover intercâmbio com outros sistemas ou *software* de apoio. Podemos citar como sub-goal da interoperabilidade a portabilidade, flexibilidade e customização de sistema. Plataforma java, acesso por vários *browsers*

6.1.5 NOVA MODELAGEM DOS REQUISITOS NÃO-FUNCIONAIS (PROPOSTA)

Após a reorganização dos requisitos não-funcionais, um novo grafo SIG foi gerado, conforme a Figura 6.2, cujo objetivo é demonstrar que os requisitos não-funcionais proporcionam para o analista uma visão mais realista do sistema. No grafo é possível observar que, depois que os requisitos não-funcionais foram reorganizados, os relacionamentos foram identificados. Por exemplo, a subgoal *Integridade* foi decomposta em *Precisão*, que em seguida foi operacionalizada em *Sincronizar dados*. Neste exemplo existem dois impactos relevantes: um positivo outro negativo. *Sincronizar dados* tem uma interdependência positiva com a operacionalização *Tempo de resposta*, e negativa com a subgoal *Custo de desenvolvimento*. Entretanto, é importante salientar que apesar desta interdependência negativa, isto é, relação de conflito entre os requisitos, a subgoal *Custo de desenvolvimento* possui o rótulo fracamente negada, pois cabe ao desenvolvedor decidir e negociar com o *stakeholders* uma possível solução.

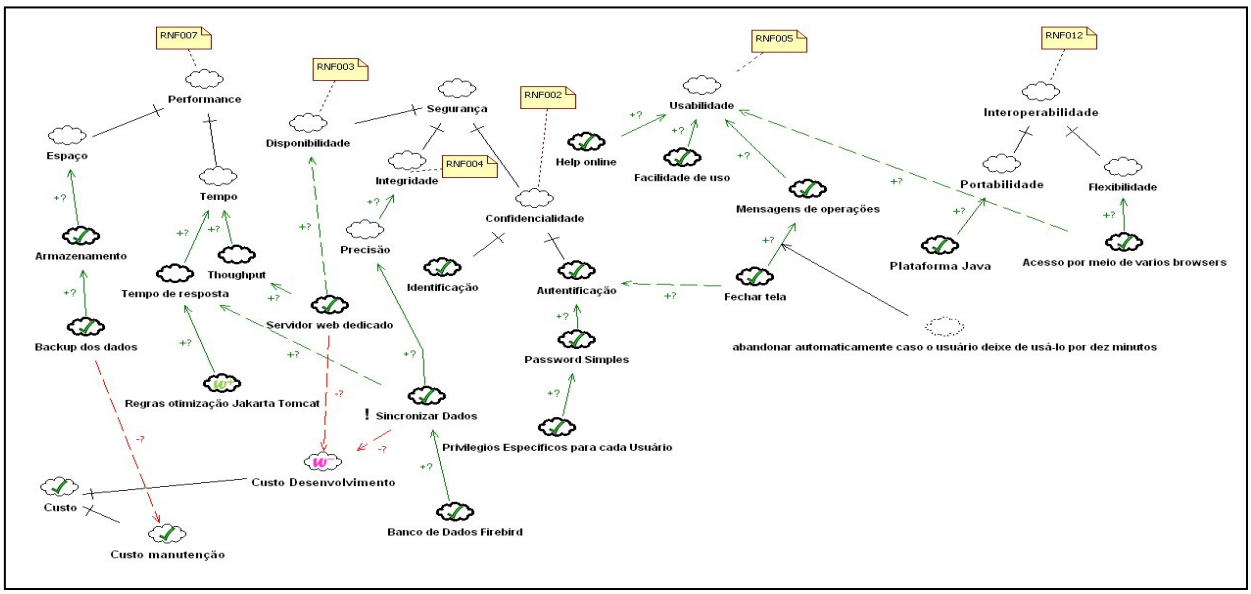


Figura 6.2 - Grafo SIG com os Novos RNFs Propostos (Estudo de Caso 1)

6.1.6 QUADRO COMPARATIVO POR TIPOS DE *SOFTGOAL* E RELACIONAMENTOS

Após a atividade de construção do grafo SIG dos RNFs propostos, foi gerado uma Tabela 6.1 com todos os tipos de elementos do grafo SIG.

Tabela 6.1 - Quadro Comparativo por Tipos de *Softgoal* e Relacionamentos (Estudo de Caso 1)

	Softgoal de Requisitos não-Funcionais	Sub-*Softgoal*	*Softgoal* de Operacionalização	*Softgoal* Declarativa	Relacionamento de Correlações Positivas	Relacionamento de Correlações Negativas	Totais
Original	4	2	4	1	1	0	12
Proposta	5	10	18	1	4	3	41

A tabela acima separa os itens que compõem os grafos SIGs original e proposto. O objetivo é demonstrar numericamente a diferença entre eles, para cada item. Como se pode observar, as *Softgoals* de requisitos não-funcionais da proposta, que representam os requisitos de nível mais abstrato, são superiores a quantidade de *Softgoals* de requisitos não-funcionais originais. Este fator pode ser evidenciado em quase todos os itens da tabela, com exceção das *Softgoals* Declarativas. Adicionalmente, baseado na Tabela 6.1 pode-se gerar gráficos de colunas (Gráfico 6.1) para que o engenheiro de requisito possa visualizar os itens compostos pelo SIG de forma mais clara.

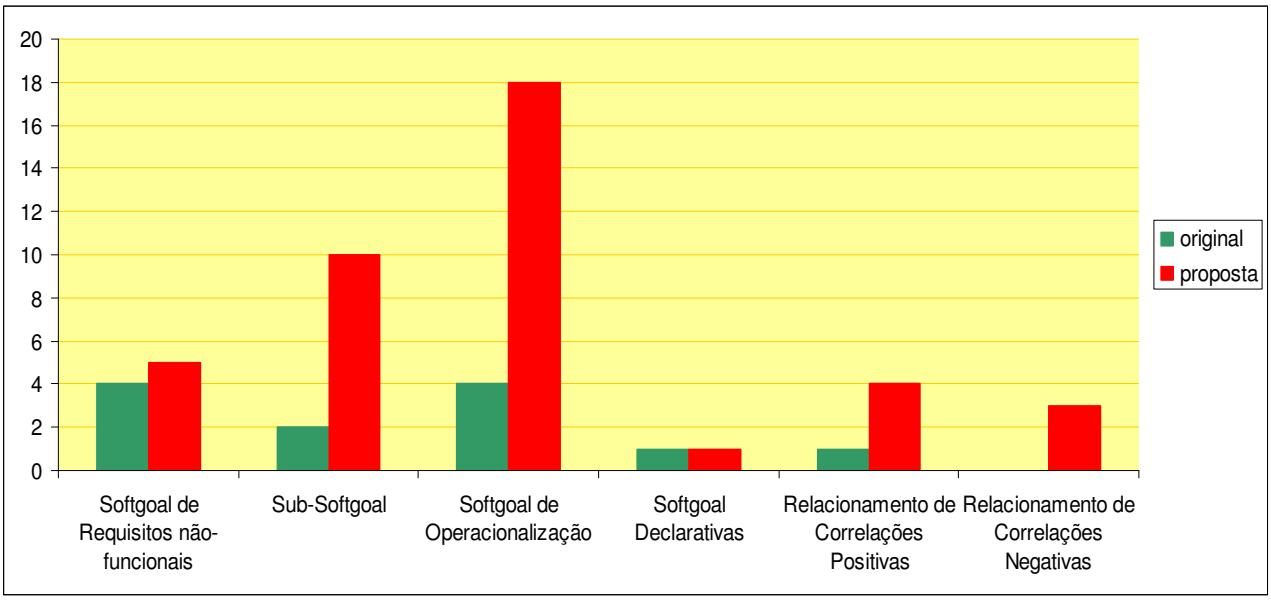


Gráfico 6.1 - Gráfico Comparativo dos Elementos do SIG Oriundos dos RNFs Originais e Propostos (Estudo de Caso 1)

6.1.7 Indicadores da Evolução dos RNFs (Original) Frente aos RNFs Propostos a partir do *NFR-Framework*.

A partir dos dados obtidos, foi calculada a Evolução dos RNFs (E_{RNF}), indicando a evolução dos RNFs originais frente aos propostos. O cálculo demonstra que a porcentagem apresentada neste estudo de caso apresenta taxa superior a 60 %, indicando a utilidade das atividades no processo de validação de requisitos não-funcionais.

$$\mathbf{VP} = \frac{25}{12 + 25} = 67\ \%$$

Após a aplicação da fórmula, o $\mathbf{E_{RNF}}$ foi considerado alto e o **QEO** baixo, conforme Tabela 6.2 . Isto significa que enquanto houve uma evolução dos RNFs originais frente aos propostos, não houve boa qualidade na especificação original dos RNFs.

Tabela 6.2 - Indicadores de Evolução dos RNFs Proposto e a Qualidade da Especificação Original (Estudo de Caso 1)

E_{RNF}	VP	QEO
Muito Baixa	**VP ≤ 20%**	Muito Alta
Baixa	**20% < VP ≤ 40%**	Alta
Média	**40% < VP ≤ 60%**	Média
Alta	**60% < VP ≤ 80%**	Baixa
Muito Alta	**VP > 80%**	Muito Baixa

Legenda
E_{RNF} – Evolução dos Requisitos Não-Funcionais
VP – Valor Percentual
QEO – Qualidade da Especificação Original

6.2 Ferramenta de Apoio à Elicitação de Requisitos Utilizando a META

6.2.1 Visão Geral da Ferramenta

Foi realizada a elicitação de requisitos, como processo da Engenharia de Requsitos com a própria META, na compreensão do ambiente que a ferramenta auxiliará. A META apresenta uma seqüência de processos a serem desenvolvidos com a finalidade de elicitar os requisitos de forma orientada e seqüencial. O processo foi

realizado em três etapas: a) Divisão do Problema em Atividades; b) Delineamento do Contexto em Atividades; e c) Descrição da Estrutura das Atividades (FRANCETO, 2005).

A versão da ferramenta de apoio à META, para a qual foi realizada a elicitação de requisitos, foi um ambiente mono-usuário. Apenas o analista de requisitos está envolvido no contexto da utilização da ferramenta. Alguns pontos observados durante a elicitação de requisitos auxiliaram no desenvolvimento do projeto da ferramenta (FRANCETO, 2005). As principais funções do sistema são: levantar atividades candidatas; selecionar atividades; descrever histórico das atividades selecionadas; identificar os motivos e resultados da atividade; identificar os elementos no nível individual; identificar os elementos no nível social e modelar a atividade através do diagrama de Engeström. A seguir, a lista de requisitos documentados na especificação dos RNFs.

6.2.2 Lista dos Requisitos não-Funcionais

O documento inspecionado apresentou apenas dois RNFs, na lista abaixo identificados com R1 e R2.

R1 - Interface amigável
R2 - Criação de *Help*

6.2.3 Modelagem dos Requisitos não-Funcionais (Original)

O grafo SIG apresentado na Figura 6.3 ilustra apenas um requisito não-funcional e duas operacionalizações. Isto sugere que os requisitos não-funcionais não receberam a devida atenção do engenheiro de requisito. Assim, este exemplo possui apenas duas operacionalizações, consequentemente nenhuma correlação positiva ou negativa. As próximas atividades serão usadas para ajudar o engenheiro de requisito a descobrir no documento de especificação os RNFs que fazem parte do *software*, e que, por falha no projeto ou desconhecimento dos RNFs, não estão na especificação original.

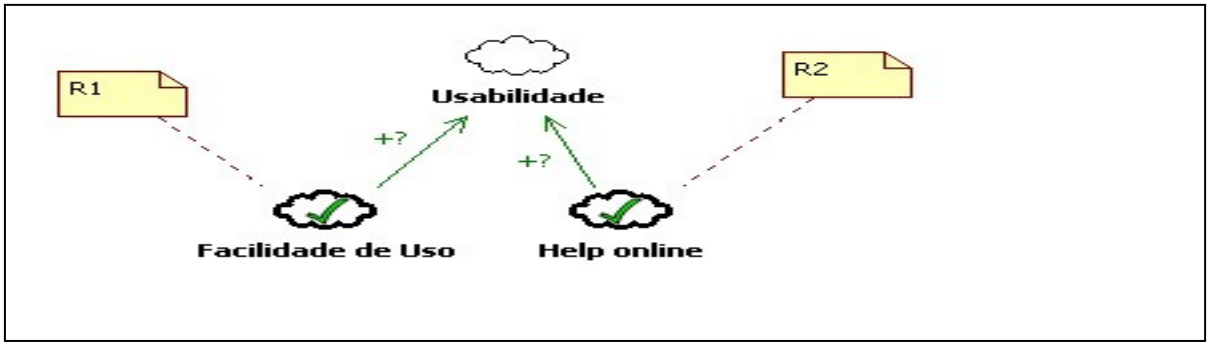


Figura 6.3 - Grafo SIG com os RNFs Originais (Estudo de Caso 2)

6.2.4 NOVA ORGANIZAÇÃO DOS REQUISITOS NÃO-FUNCIONAIS (PROPOSTA)

Para ajudar no processo de identificação dos novos requisitos foram utilizados os catálogos anexos (A, B, C e D) e a classificação descrita por Sommerville (2001). Catálogos são úteis para refinamento de s*oftgoals*. Eles fazem parte do *NFR-Framework* e permitem armazenar conhecimentos sobre requisitos não-funcionais. Especificamente neste estudo de caso, os catálogos foram muito utilizados, devido ao fato de que em toda especificação não há detalhes sobre RNFs.

Classificação dos Requisitos Não-Funcionais

Requisitos do Processo

RNF001	**Desenvolvimento do sistema** - em linguagem *Java*

Requisito do Produto

RNF002	**Segurança – Confidencialidade** O sistema deverá prever mecanismos para controlar o acesso ao sistema (*login* e senha).
RNF003	**Segurança – Integridade** Os dados inseridos e consultados devem ser consistentes e sincronizados. Realizar *backup* de segurança
RNF004	**Usabilidade** Deverá ter facilidade de uso, *help-online* e mensagens de operações. Deverá haver controle interno no sistema para que o mesmo seja abandonado automaticamente, caso o usuário deixe de usá-lo por dez minutos. A tela inicial de *login* deverá ser apresentada para continuidade do trabalho suspenso.
RNF005	**Confiabilidade** Sistema sempre disponível.
RNF006	**Banco de Dados - SGBD (Sistema Gerenciador de Banco de Dados) MySQL**
RNF007	**Utilizar UML para modelagem**

Requisitos Externos

RNF008	**Interoperabilidade** Diagrama de Engeström

6.2.5 NOVA MODELAGEM DOS REQUISITOS NÃO-FUNCIONAIS (PROPOSTA)

A nova lista dos RNFs gerada permite a construção do grafo SIG, conforme a Figura 6.4. Por exemplo, a S*oftgoal Segurança* foi decomposta em *Disponibilidade*, *Integridade* e *Confidencialidade*, que em seguida foi operacionalizada em *Identificação e Autenticação*. A S*oftgoal Usabilidade* foi operacionalizada em *Facilidade de uso, Help online e Mensagens de operação*. Neste grafo SIG existem impactos relevantes como, por exemplo: a *Softgoal Custo* ajuda os engenheiros de requisitos nas tomadas de decisões, assim como o aumento significativo dos RNFs possibilita uma visão mais clara sobre os mesmos e também uma compreensão sobre eles.

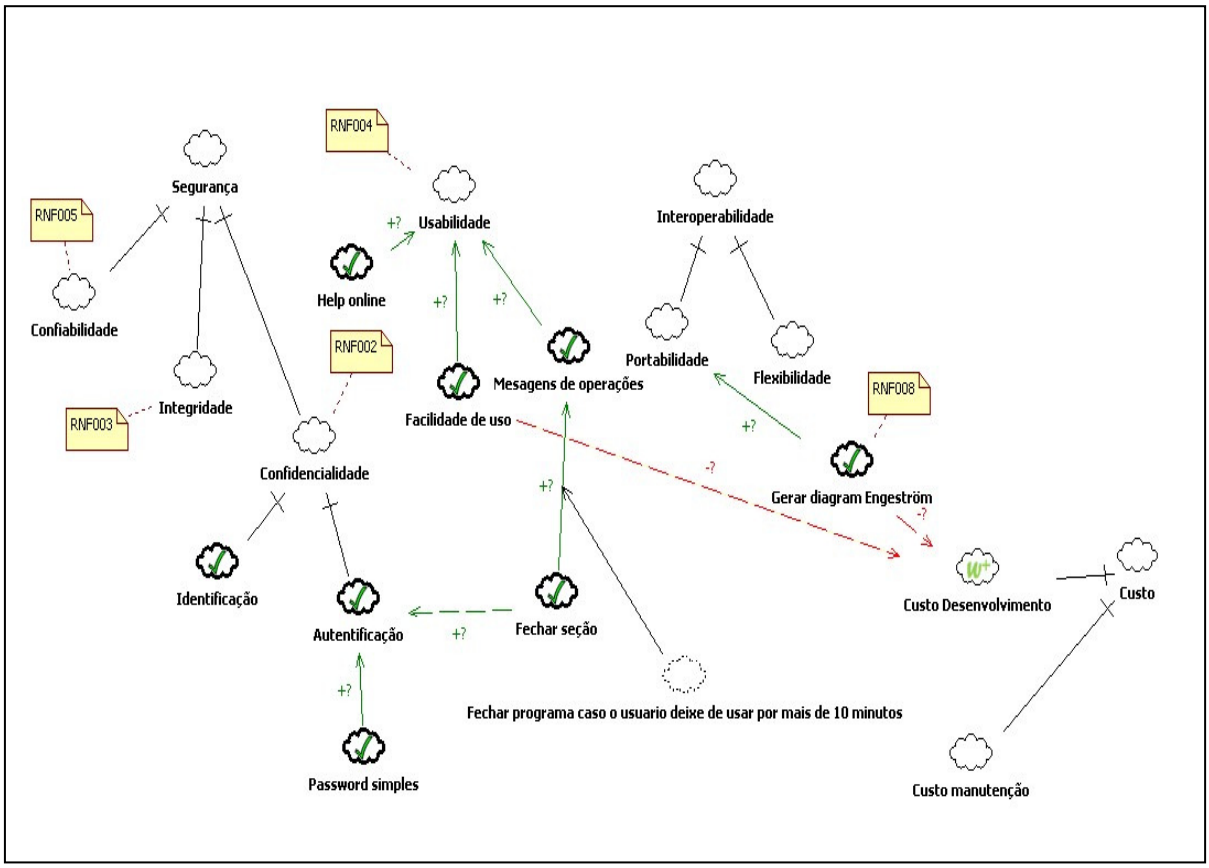


Figura 6.4 - Grafo SIG com os Novos RNFs Propostos (Estudo de Caso 2)

6.2.6 Quadro Comparativo por Tipos de *Softgoal* e Relacionamentos

A tabela 6.3 estabelece uma divisão dos itens que compõem os grafos SIGs originais e propostos. Conforme podemos observar, a proposta apresenta aumento significativo de itens em relação ao original, representando um aumento de mais de 700%. Para que se possa visualizar essa superioridade, foi gerado o Gráfico 6.2 dos itens compostos pelo SIG, evidenciando a utilidade das atividades propostas.

Tabela 6.3 - Quadro Comparativo por Tipos de *Softgoal* e Relacionamentos (Estudo de Caso 2)

	Softgoal de Requisitos não-Funcionais	Sub-*Softgoal*	*Softgoal* de Operacionalização	*Softgoal* Declarativa	Relacionamento de Correlações Positivas	Relacionamento de Correlações Negativas	Totais
Original	1	0	2	0	0	0	3
Proposta	4	7	8	1	1	2	23

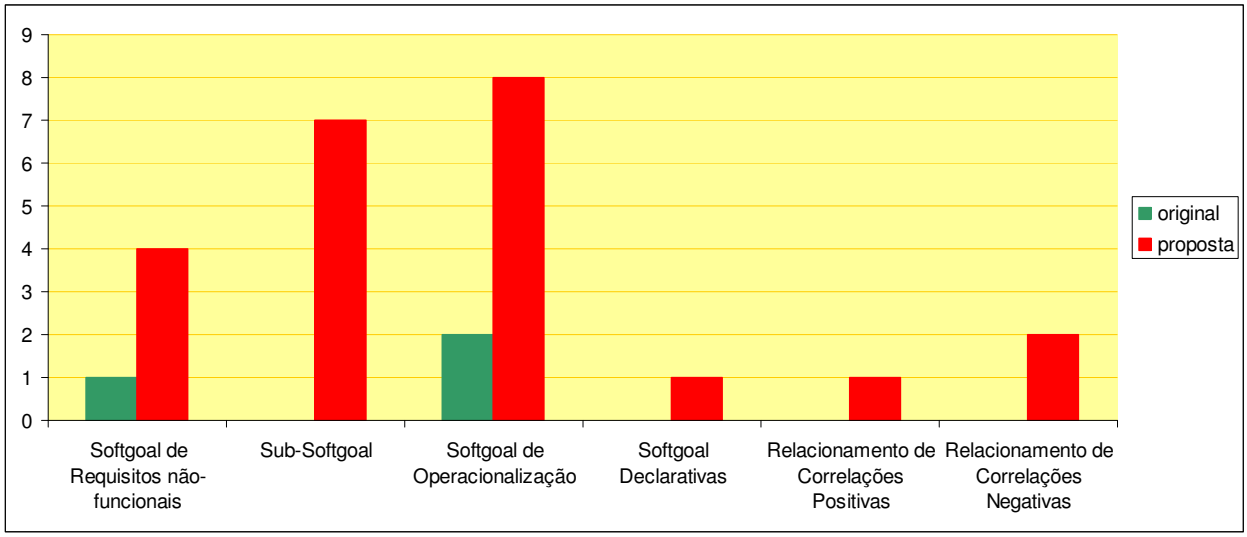


Gráfico 6.2 - Gráfico Comparativo dos Elementos do SIG Oriundos dos RNFs Originais e Propostos. (Estudo de Caso 2)

6.2.7 Indicadores da Evolução dos RNFs (Original) Frente aos RNFs Propostos a partir do *NFR-Framework*.

Após o quadro comparativo montado, foi calculada a Evolução dos RNFs (E_{RNF}). O cálculo demonstra que a porcentagem apresentada neste estudo de caso é superior a 80%, o que sugere avanços significativos dos RNFs originais frente aos propostos.

$$\mathbf{VP} = \frac{17}{3+17} = 85\ \%$$

Assim, podemos concluir que o **E_{RNF}** é muito alto e o **QEO** muito baixo, conforme a Tabela 6.4. Demonstramos que, apesar da evolução dos RNFs originais frente aos RNFs propostos, por outro lado, a qualidade na especificação original dos RNFs está muito abaixo da ideal.

Tabela 6.4 - Indicadores de Evolução dos RNFs Proposto e a Qualidade da Especificação Original (Estudo de Caso 2).

E_{RNF}	VP	QEO
Muito Baixa	**VP ≤ 20%**	Muito Alta
Baixa	**20% < VP ≤ 40%**	Alta
Média	**40% < VP ≤ 59%**	Média
Alta	**60% < VP ≤ 79%**	Baixa
Muito Alta	**VP > 80%**	Muito Baixa

Legenda
E_{RNF} – Evolução dos Requisitos Não-Funcionais
VP – Valor Percentual
QEO – Qualidade da Especificação Original

6.3 DOCUMENTAÇÃO DOS REQUISITOS DO SISTEMA DE GESTÃO DE *RANKING* E CHAVES DE TORNEIOS DO TÊNIS CLUBE

6.3.1 VISÃO GERAL DO DOCUMENTO DE ESPECIFICAÇÃO

O Tênis Clube de Piracicaba (TCP) regularmente promove torneios de tênis, divididos em categorias, sendo que os atletas participantes, possuem uma classificação quanto à categoria e uma pontuação para a sua participação no *Ranking*. O TCP possui quatro quadras de tênis, as quais são utilizadas para os torneios e aulas. Os torneios são divididos em etapas, e cada uma possui as suas chaves com as partidas entre os atletas. Para a realização concreta do torneio, é preciso dividir os atletas inscritos por categoria (avançado, intermediário, principiante e infantil), montar as chaves, as etapas e os horários; para tanto, são necessários os dados do atleta, a agenda das quadras, a agenda dos atletas e outras informações pertinentes à partida. Após a realização do torneio, os participantes ganham pontos, os quais são somados aos seus pontos anteriores e uma nova atualização do *ranking* é publicada. A realização de partidas e contagem de vitórias e derrotas é

considerada para a mudança de categoria dos atletas (TORRES, RODRIGUES e GUASSI, 2007).

O Sistema de Gestão de *Ranking* e Chaves de Torneios tem como um dos propósitos principais gerar um fluxo de trabalho automatizado para a composição e atualização da posição no *Ranking* dos atletas cadastrados no Tênis Clube de Piracicaba. Para tal, o sistema usa o Regulamento de Pontuação de *Ranking* do TCP, o qual concede ao atleta uma pontuação, dependendo do seu desempenho (classificação) nos torneios realizados, bonificação por número de inscritos no torneio e dos resultados dos jogos de desafios, assim como, a classificação dos atletas por categoria (avançado, intermediário, principiante e infantil) (TORRES, RODRIGUES e GUASSI, 2007).

6.3.2 LISTA DOS REQUISITOS NÃO-FUNCIONAIS

Nesta atividade, foi inspecionado o documento, elencando os requisitos não-funcionais, acrescentando os códigos denominados R1, R2, R3, R4, R5 e R6, para rastreá-los dentro do grafo SIG. Analisando os requisitos não-funcionais elicitados, apenas o RNF *Segurança* foi coerentemente levantado. O requisito não-funcional *Imunidade,* por exemplo, não consta na classificação descrita por Sommerville (2001). Este erro é muito comum, devido à diversidade e à divergência de conceitos sobre RNFs. Neste caso específico, a descrição sugere que a utilização de um antivírus e de um *firewall*, além de um sistema operacional, será de responsabilidade do cliente. Assim, a literatura sugere que o termo mais apropriado para este requisito seria *Interoperabilidade*.

R1 - Requisitos de Segurança

R2- Requisitos de Acesso

Atualmente, o sistema não prevê restrição de acesso, tanto ao usuário-administrador quanto ao usuário-operador. Futuramente, com a expansão do sistema, prevemos novos tipos de usuários, e cada um terá um nível de acesso ao sistema, o qual começará a contar com identificação e senha de entrada. O usuário-administrador terá acesso total e irrestrito a todas as áreas do sistema; quanto ao usuário-

operador, terá acesso às áreas operacionais do sistema, e os usuários de consulta, por exemplo, terão acesso a telas de consultas de informações e relatórios.

R3- Requisitos de Integridade

Os principais requisitos de integridade identificados para o sistema foram:

- O sistema deve estar prevenido contra a entrada incorreta de dados, desde o cadastramento dos atletas até o fechamento do torneio;
- O sistema deve estar preparado para recuperar dados que acidentalmente foram modificados ou excluídos;
- A integridade do Banco de Dados ficará a cargo do Sistema Gerenciador de Banco de Dados.

R4- Requisitos de Privacidade

O sistema aqui considerado não possui qualquer requisito de privacidade, uma vez que o escopo do negócio a ser tratado pelo sistema não exige informações confidenciais tanto dos indivíduos cadastrados, dos usuários envolvidos, ou mesmo informações sigilosas da organização em que o sistema atua. Enfim, as informações contidas no sistema e a sua visualização por seus usuários não acarretará qualquer dano financeiro ou moral aos envolvidos.

R5 - Requisitos de Auditoria

Este sistema não possui qualquer implicação Legal, Tributária, Fiscal ou Contábil, portanto, não foram identificados requisitos de auditoria necessários a este tipo de projeto.

R6 - Requisitos de Imunidade

Devido ao tipo e ao tamanho de sistema, do local de trabalho e do modo de operação, atualmente fica a cargo do cliente dar suporte aos requisitos de imunidade (vírus, “ataques”, “invasões” via rede ou internet), para tanto, basta o cliente lançar mão de um antivírus e de um *firewall*, além de um sistema operacional seguro.

6.3.3 Modelagem dos Requisitos Não-Funcionais (Original)

Após listar os requisitos não-funcionais, foi construído o grafo SIG conforme Figura a 6.5. A *softgoal Segurança* foi decomposta em *Integridade* e *Confidencialidade*. *Integridade* foi decomposta em duas sub-goals *Banco de dados* e *Recuperar dados*, que por sua vez foi operacionalizada em *Backup de dados* com rótulo satisfeito. A sub-goal *Confidencialidade* foi operacionalizada em *Autenticação*, *Password simples*

e *Privilégios específicos para cada usuário* com todos com rótulos negados, evidenciando que a sub-goal *Confidencialidade* não é um RNF importante nesta fase do desenvolvimento do projeto, o que na nossa visão é um erro e deveria ser renegociado com os *stakeholders*.

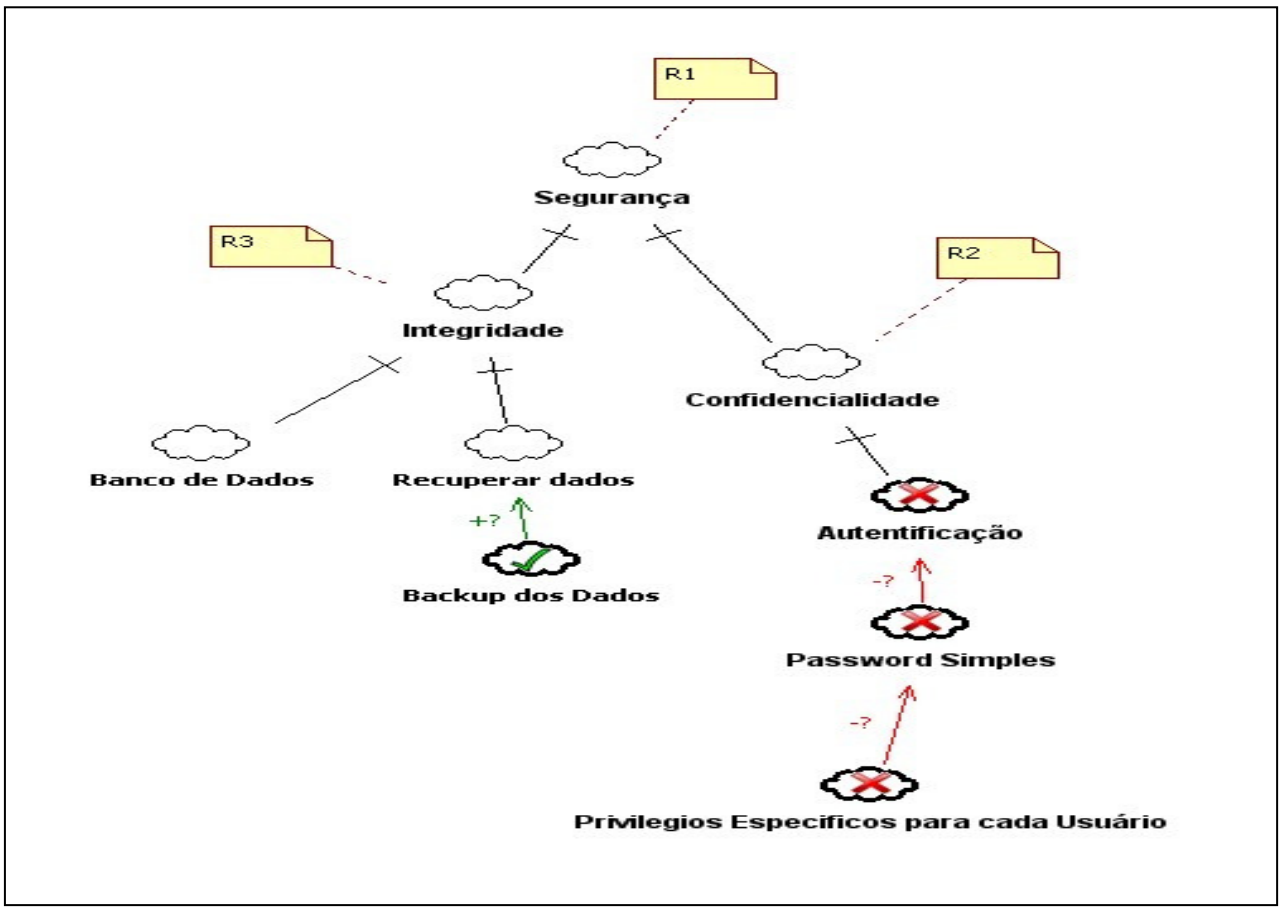


Figura 6.5 - Grafo SIG com os RNFs Originais (Estudo de Caso 3)

6.3.4 NOVA ORGANIZAÇÃO DOS REQUISITOS NÃO-FUNCIONAIS (PROPOSTA)

Esta atividade proporciona para o analista uma visão mais realista dos RNFs, classificados em Processo, Produto e Requisitos Externos segundo Sommerville (2001). Além disso, foram utilizados catálogos para organizar e decompor requisitos não-funcionais de forma mais racional.

Classificação dos Requisitos Não-Funcionais

Requisitos do Processo

RNF001	**Desenvolvimento do sistema** - em linguagem *Java*

Requisito do Produto

RNF002	**Segurança – Confidencialidade** O sistema deverá prever mecanismos para controlar o acesso ao sistema (*login* e senha).
RNF003	**Segurança – Disponibilidade** O sistema deverá estar disponível 24 horas por dia, 7 dias por semana.
RNF004	**Segurança – Integridade** Os dados inseridos e consultados devem ser consistentes e sincronizados.
RNF005	**Usabilidade** Deverá ter facilidade de uso, *help-online* e mensagens de operações. Deverá haver controle interno no sistema para que o mesmo seja abandonado automaticamente, caso o usuário deixe de usá-lo por dez minutos. A tela inicial de *login* deverá ser apresentada para continuidade do trabalho suspenso.
RNF006	**Confiabilidade** Sistema sempre disponível.
RNF007	**Performance** Deverá manter compacto o armazenamento, tempo de resposta satisfatório para as transações (regras de otimização SGDB). Realizar *backup* de segurança.
RNF008	**Banco de Dados - SGBD (Sistema Gerenciador de Banco de Dados) Firebird 1.5.**

Requisitos Externos

RNF09	**Interoperabilidade** O sistema deverá prever interoperabilidade, de forma a promover intercâmbio com outros sistemas ou *software* de apoio. Podemos citar como sub-goal da interoperabilidade a portabilidade, flexibilidade e customização de sistema.

6.3.5 NOVA MODELAGEM DOS REQUISITOS NÃO-FUNCIONAIS (PROPOSTA)

Após classificar os RNFs, foi construído o grafo SIG, conforme a Figura 6.6, que ilustra, por exemplo, as S*oftgoals Segurança*, *Usabilidade*, *Performance* e *Intereperabilidade*, decompostas com as respectivas sub-softgoals e operacionalizações. Neste grafo SIG, existem impactos relevantes como, por exemplo, a sub-softgoal *Imunidade* cujas atualizações do antivírus e do *firewall* descritas como operacionalizações, são de responsabilidade do cliente. Isto é muito interessante, pois o grafo SIG permite armazenar conhecimento sobre o projeto de forma clara e precisa.

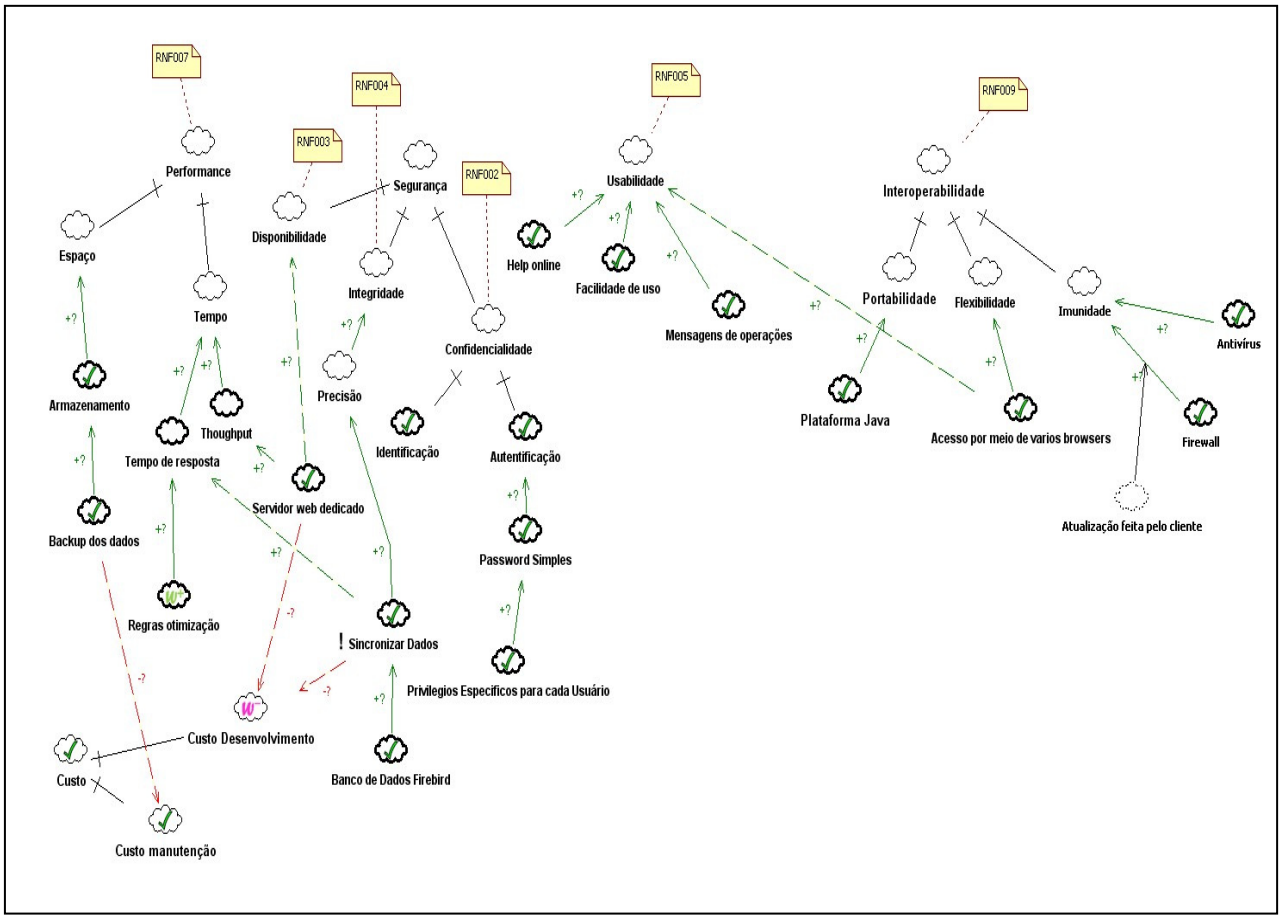


Figura 6.6 - Grafo SIG com os Novos RNFs Propostos (Estudo de Caso 3)

6.3.6 QUADRO COMPARATIVO POR TIPOS DE *SOFTGOAL* E RELACIONAMENTOS

A Tabela 6.5 demonstra os itens que compõem os grafos SIGs originais e propostos. Ao observar a tabela, fica claro que o número de itens da proposta é muito superior ao original. Por exemplo, a *softgoal* de operacionalização da proposta tem dezenove itens e a original possui quatro. Assim, os dados obtidos revelam um acréscimo de quinze itens da proposta em relação à original. Esta superioridade fica evidenciada no Gráfico 6.3.

Tabela 6.5 - Quadro Comparativo por Tipos de *Softgoal* e Relacionamentos (Estudo de Caso 3).

	Softgoal de Requisitos não-Funcionais	Sub-*Softgoal*	*Softgoal* de Operacionalização	*Softgoal* Declarativa	Relacionamento de Correlações Positivas	Relacionamento de Correlações Negativas	Totais
Original	1	4	4	0	0	0	9
Proposta	5	11	19	1	3	3	42

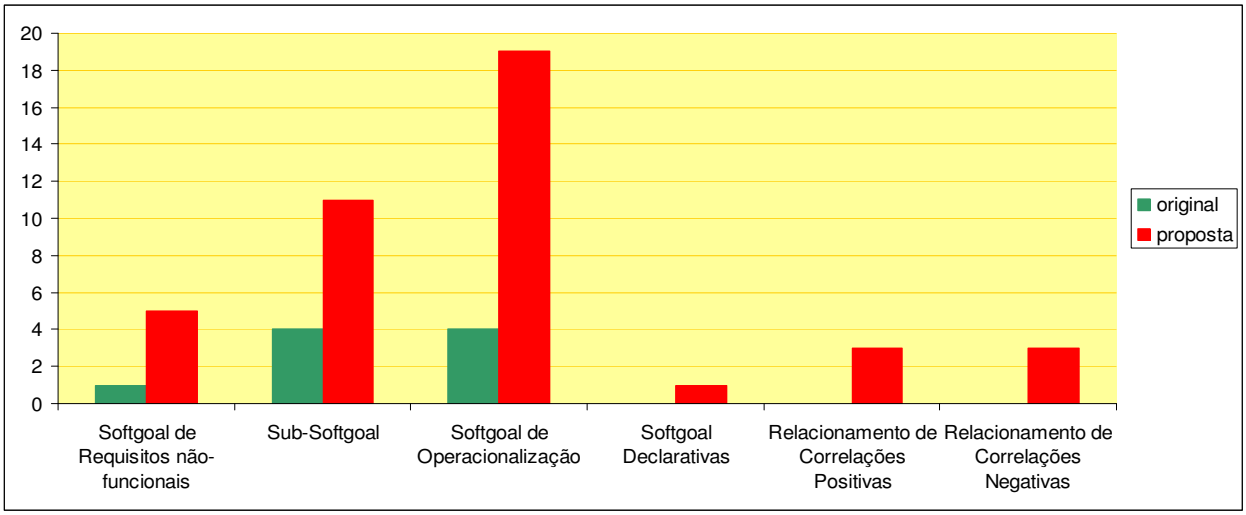


Gráfico 6.3 - Gráfico Comparativo dos Elementos do SIG Oriundos dos RNFs Originais e Propostos (Estudo de Caso 3)

6.3.7 INDICADORES DA EVOLUÇÃO DOS RNFS (ORIGINAL) FRENTE AOS RNFS PROPOSTOS A PARTIR DO *NFR-FRAMEWORK*.

Esta atividade refere-se ao cálculo da Evolução dos RNFs (E_{RNF}), o qual demonstra que a porcentagem apresentada neste estudo de caso apresenta taxa superior a 70 %.

$$\mathbf{VP} = \frac{28}{9 + 28} = 75\ \%$$

Assim, podemos concluir que o $\mathbf{E_{RNF}}$ é alto e o **QEO** é baixo, conforme a Tabela 6.6. Isto sugere que, apesar da evolução dos RNFs originais frente aos RNFs propostos, a qualidade na especificação original dos RNFs está baixa.

Tabela 6.6 - Indicadores de Evolução dos RNFs Propostos e a Qualidade da Especificação Original (Estudo de Caso 3)

$\mathbf{E_{RNF}}$	VP	QEO
Muito Baixa	**VP ≤ 20%**	Muito Alta
Baixa	**20% < VP ≤ 40%**	Alta
Média	**40% < VP ≤ 60%**	Média
Alta	**60% < VP ≤ 80%**	Baixa
Muito Alta	**VP > 80%**	Muito Baixa

Legenda
E_{RNF} – Evolução dos Requisitos Não-Funcionais
VP – Valor Percentual
QEO – Qualidade da Especificação Original

6.4 ANÁLISE DE RESULTADOS DAS TENDÊNCIAS

Nos três estudos apresentados, fica claro que não existe consenso para os requisitos não-funcionais, mas sim grande discrepância conceitual. Houve casos em que o significado não era claro, pois foram utilizados termos sem uma definição ou um exemplo claro. Isto sugere que nos três casos estudados, os engenheiros de requisitos não conhecem as definições sobre requisitos não-funcionais.

Podemos observar ainda que os itens sub-*softgoal* e *softgoal* de operacionalizações da proposta pós-validação apresentam os maiores valores. Isto ocorre porque as *softgoals* de requistos não-funcionais originais, que são os requisitos não-funcionais mais abstratos, não foram apropriadamente decompostas. Isto sugere que as possíveis metas dos RNFs originais não incluem soluções ou alternativas para alcançá-las. Por outro lado, os mesmos itens da proposta oferecem um refinamento das metas que possibilita maior legibilidade dos RNFs, e consequentemente do documento de especificação.

É sabido que conflitos entre requisitos acontecem, e durante o processo de engenharia de requisitos isto é muito comum; entretanto, devem ser identificados e negociados. Dito isto, um fator importante nos três estudos apresentados é que não ocorreu nenhum relacionamento de correlações negativas, ou seja, conflitos de requisitos. A ocorrência deste item no documento de especificação não necessarimente é um fator ruim, ao contrário, pode ser esclarecdor, pois fornece informações básicas para a detecção e resolução dos conflitos que surgem a partir de várias posições entre os *stakeholders*. Desta forma, os itens dos grafos SIGs propostos possibilitam que os engenheiros de requisitos tomem decisões mais realistas durante todo o processo de desenvolvimento do *software*.

Outro fator importante nestes 3 (três) estudos apresentados é que eles contribuíram de forma unificada para a avaliação e o amadurecimento da sistematização do processo de validação de requisitos não-funcionais. Independente das características individuais de cada estudo, foi possível identificar um padrão quantitativo nos elementos do grafo SIG *softgoal* de requisitos não-funcionais, sub-*softgoal* e *softgoal* de operacionalização. Estes padrões sugerem que, à medida

que o número de *softgoal* de requisitos não-funcionais cresce, a sub-*softgoal* também aumenta; consequentemente, a *softgoal* de operacionalização atinge os maiores valores. Isto demonstra que nos três estudos apresentados, os RNFs cruciais para o desenvolvimento do *software* não foram priorizados, impossibilitando que os RNFs possam se relacionar com as decisões de projetos.

7 CONCLUSÕES

7.1 CONSIDERAÇÕES INICIAIS

Esta dissertação concentrou-se em uma das fases da Engenharia de Requisitos tendo como eixo condutor o processo de validação de requisitos não-funcionais. A motivação está no fato de que muitos engenheiros de requisitos e a maioria das ferramentas de desenvolvimento incidem sobre os requisitos funcionais, isto é, sobre aquilo que o sistema deve fazer. Entretanto, estudos apontam que requisitos não-funcionais são importantes para o desenvolvimento, pois questões como qualidade, segurança e desempenho são cruciais para o sucesso de um sistema de *software*. Isto possibilitou elencar como ponto de partida duas premissas para o processo de validação de requisitos não-funcionais. São elas:

- As atividades propostas servem como um instrumento de melhoria de qualidade no processo de validação de requisitos não-funcionais;
- As atividades propostas auxiliam os Engenheiros de Requisitos a efetuar um refinamento dos requisitos não-funcionais, identificando propriedades do sistema que anteriormente estavam obscuras.

Em virtude do que foi mencionado, propusemos seis atividades para o processo de validação de requisitos não-funcionais. Elas fornecem, sistematicamente, subsídios para que RNFs sejam avaliados na fase de concepção dos projetos. As atividades propostas são baseadas no *NFR-Framework*, e uma das vantagens é que essa abordagem ajuda os engenheiros de requisitos a lidar com RNFs. Isto é muito importante, e foi observado, na utilização das atividades nos três estudos de caso que os engenheiros de requisitos possuem uma baixa compreensão dos RNFs, desconhecendo-os e, conseqüentemente, anulando sua utilização no processo de elicitação. Além disso, o *NFR-Framework* fornece a vantagem de extrair o Grafo de Interdependência da *Softgoal* (SIG), possibilitando armazenar conhecimentos sobre os RNFs.

7.2 CONTRIBUIÇÕES

Este trabalho contribui para que os engenheiros de requisitos possam representar conceitos para lidar com RNFs durante o processo de desenvolvimento de *software*. Adicionalmente, a utilização das atividades do processo de validação revelou alguns aspectos:

- Oferece um modo sistemático para avaliar a evolução dos RNFs;
- Tabela com os indicadores de Evolução dos RNFs Propostos e a Qualidade da Especificação Original dos RNFs;
- Permite que os RNFs possam se relacionar com as decisões de projetos;
- Permite detectar defeitos nos RNFs;
- Detectar omissões no projeto;
- Avaliar a qualidade do documento de especificação de requisitos não-funcionais.

Nos estudos, as atividades também proporcionaram suporte para detectar e lidar com conflitos. Outro fator relevante é que ambigüidades foram detectadas e reduzidas, clarificando as especificações dos RNFs. Por todos estes aspectos as atividades melhoram o processo de validação de requisitos não-funcionais.

7.3 REVISÃO CRÍTICA

A nossa abordagem tem algumas limitações:

- Deve-se ampliar os catálogos dos RNFs: Há a necessidade de se buscar no setor empresarial de desenvolvimento de *software* conhecimento sobre requisitos não-funcionais para ampliação dos catálogos;

- Nossas atividades estão fortemente dependentes do *NFR-Framework*. Se ocorrerem mudanças no *NFR-Framework,* as atividades precisam ser revistas.

7.4 TRABALHOS FUTUROS

A nossa investigação baseou-se no processo de validação de requisitos não-funcionais, através de 6 (seis) atividades, cujo objetivo final é saber se houve evolução dos requisitos propostos frente aos originais, como também aferir qualidade no documento de especificação original.

Contudo, algumas análises dos RNFs propostos poderiam ser feitas no futuro. Por exemplo, investigar o impacto das alterações dos requisitos não-funcionais sobre os requisitos funcionais, avaliar o quanto a baixa qualidade da especificação pode tornar a implementação do projeto inviável, realizar experimentação das atividades com outras pessoas especialistas e inserir uma análise de incremento da qualidade. Além disso, o desenvolvimento de uma ferramenta de apoio do processo de validação proposto auxiliará os engenheiros de requisitos a identificar os requisitos não-funcionais e avaliar a qualidade do documento de especificação original. Estas são questões que precisam ser consideradas no futuro.

7.5 CONSIDERAÇÕES FINAIS

Acreditamos que a utilização do processo proposto dentro do contexto da validação de requisitos contribuirá para uma melhoria na qualidade da especificação, modelagem, análise e decisões tomadas ao longo do processo de Engenharia de Requisitos. As atividades propostas tornam explícitas as ligações e as soluções entre a elicitação dos RNFs e o processo de validação coerente em cada projeto. Isto permite uma visualização clara dos objetivos e, consequentemente, analisar sistematicamente quanto o projeto está alinhado com as especificações do usuário. A sua utilização irá ajudar os engenheiros de requisitos a encontrar, com mais facilidade, eventuais falhas no documento inicial de requisitos.

REFERÊNCIAS BIBLIOGRÁFICAS

AL-SUBAIE, H. S. F., MAIBAUM, T. S. E.; "***Evaluating the Effectiveness of a Goal-Oriented Requirements Engineering Method"***; Fourth International Workshops on Comparative Evaluation in Requirements Engineering, 2006.

ANTON***,*** A.I.; "***Goal Identification and Refinement in the Specification of Software Based Information Systems"***; Dissertação de Phd, Georgia Institute of Technology, Atlanta, 1997.

ANWER ,S. , IKRAM ,N. ;**"Goal Oriented Requirement Engineering: A Critical Study of Techniques"**; xiii Asia Pacific Software Engineering Conference, 2006.

BAIXAULI***,*** B. G., LEITE, J. C. S., Mylopoulos, J***.*** ; ***"Visual Variability Analysis for Goal Models"***; 12th IEEE International Requirements Engineering Conference (RE'04), Kyoto,Japão, 2004.

BOEHM, B.; ***"Model and metrics for software management and engineering"***; IEEE Computer Society Press, 1984.

BOEHM, B. ; ***"Industrial software metrics top 10 list"*** ; IEEE, 1987***.***

BRACKETT, J. W***.; "Software requirements. Pittsburgh, Pennsylvania:"***; SEI, Janeiro. 1990.CMU/SEI-CM.

BROOKS, F.P. Jr.; ***"No Silver Bullet: Essence and Accidents of Software Engineering"***; IEEE Computer, Abril 1987***.***

CAMACHO, C.; ***"Gerenciando Conflitos em Reuniões: Uma Estratégia para a Elicitação de Requisitos de Software"***; Dissertação de Mestrado, PUC - Rio de Janeiro, 2005.

CHUNG, L. NIXON, B. YU, E., MULOPOULOS, J.; "**Non-Functional Requirements in Software Engineering"**; Kluwer Academic Publisher, 2000.

CHUNG L.; "**Representation and Utilization of Non-Functional Requirements for Information System Design"** ; University of Toronto, 1991.

CHRISTEL, M. G. KANG, K. C.; **"Issues in Requirements Elicitation, Technical Report"**; CMU/SEI-92-TR-012 ESC-TR-92-012, September 1992.

CYSNEIROS, L. M. LEITE, J. C. S. P. SABAT, J. M. N.; ***"A Framework for Integrating Non-Functional Requirements into Conceptual Models"***; Requirements Eng. Springer-Verlag, Londres, 2001.

DARDENNE, A. LAMSWEERDE, A. FICKAS, S.; **"Goal-directed Requirements Acquisition"**; Science of Computer Programming, 1993.

GOGUEM, J. A. LINDE, C. ;**"Techeniques Requirements Elicitation"**; IEEE Computer Society, 1993.

GRANDE, J. I. MARTINS, L. E. G. ;***"Uma Ferramenta para Gerenciamento de Requisitos"***, WER, Rio de Janeiro, 2006.

HICKEY, A. M., DAVIS, A. M.; "***Elicitation Technique Selection: How Do Experts Do It"***; Proceedings of the 11th IEEE International Requirements Engineering Conference, 2003

IEEE ; "***IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries"***; New York, NY: Institute of Electrical and Eletronic Engineers 1990.

IEEE ; "***IEEE Std 1219-1998. IEEE Standard for Software Maintenance"***; New York: Institute of Electrical and Eletronic Engineers. 1998.

IEEE ; "***IEEE STD-830, IEEE Recommended Practice for Software Requirements Specifications"***; New York: Institute of Electrical and Electronic Engineers, 1998.

KAVAKLI ,E. LOUCOPOULOS, P. ; **"Goal Driven Requirements Engineering: Evaluation of Current Methods"**; EMMSAD , 2003.

KOTONYA, G. SOMMERVILLE, I. ; "***Requirements Engineering: Processes and Techniques"***; Wiley, Chichester, England 1998.

LAMSWEERDE, A. V. ; ***"Goal-Oriented Requirements Engineering: A Guided Tour"***; Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering, Toronto, Agosto 2001.

LAMSWEERDE, A., LETIER, E.; "**Deriving Operational Software Specifications from System Goals"**; SIGSOFT 2002/FSE-10, ACM Press, Charleston, SC, USA, 2002.

LEITE, J. C. S. P.; DOORN, J. H.; ***"Perspectives on Software Requirements"***; Kluwer Academic Press, 2004.

LETIER, E LAMSWEERDE A. V.; ***"Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering Proceedings"*** ; ACM International Symp. on the Foundations of Software Engineering, Newport Beach (CA), Novembro. 2004.

LEVESON, N. G.;**"Intent Specifications: An Approach to Buildingn Human Centered Specifications"**; IEEE Transactions on Software Engineering, vol. 26, no. 1, janeiro de 2000.

MARTINS, L. E. G.; ***"Uma Metodologia de Elicitação de Requisitos de Software Baseada na Teoria da Atividade"***;Tese de Doutorado, Campinas, SP, agosto de 2001.

MYLOPOULOS, J.; CHUNG, L.; LIAO, S.S.Y.; WANG, H., YU, E. ; **"Exploring Alternatives During Requirements Analysis "**, IEEE , Janeiro 2001.

NUSEIBEH ,B. e EASTERBROOK, S. ;***"Requirements Engineering: A Roadmap"***; In ICSE – Future of SE Track, 2000.

POWER, N. e MOYNIHAN,T. ;***"A Theory of Requirements Documentation Situated in Practice"*** ; ACM ;San Francisco, California, USA. 2003.

REGEV, G. E WEGMANN, A. ;**"Where do Goals Come from: the Underlying Principles of Goal-Oriented Requirements Engineering"**; 13th IEEE International requirements Engineering Conference, Paris, Agosto, 2005.

ROBERTSON, J. ROBERTSON, S.; ***"Volere Requirements: How to Get Started."***; <http://www.volere.co.uk/gettingstarted.htm> Acesso em: 09/04/2008.

SOMMERVILLE, I.; "***Software Engineering***"; Addison Wesley. 2001.

SOMMERVILLE, I. SAWYER, P.; ***"Requirements engineering: A good practice guide"***; Wiley, Chichester, England 2000.

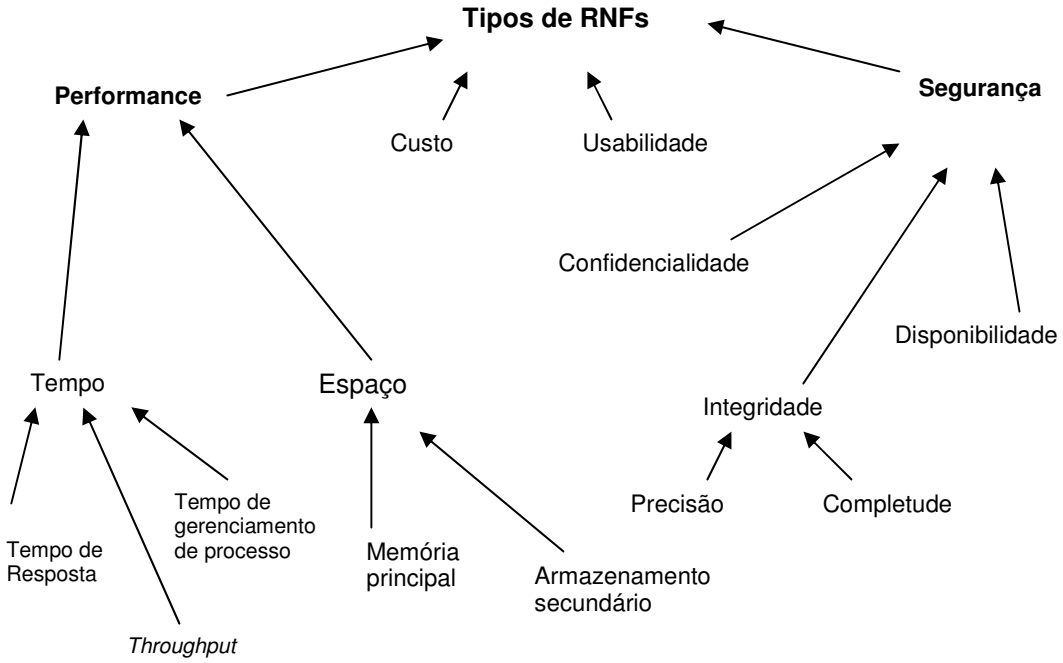
THAYER, R. H. DORFMAN, M.; ***"Software Requirements Engineering"***; IEEE Computer Society Press. 1997.

YU, E. MYLOPOULOS, J.; **"Why Goal-Oriented Requirements Engineering"**; http://www.cs.toronto.edu/pub/eric/REFSQ98.html Acesso em 21/04/2008.

ZAVE, P. JACKSON M. ;**"Four Dark Corners of Requirements Engineering"**; ACM Transactions on Software Engineering and Methodology,1997.

ANEXO A

Catálogo - Tipos de RNFs



Legenda

Relacionamento de sub-tipos

Figura A - Catálogo de Tipos de Requisitos Não-Funcionais (CHUNG et. al., 2000)

ANEXO B

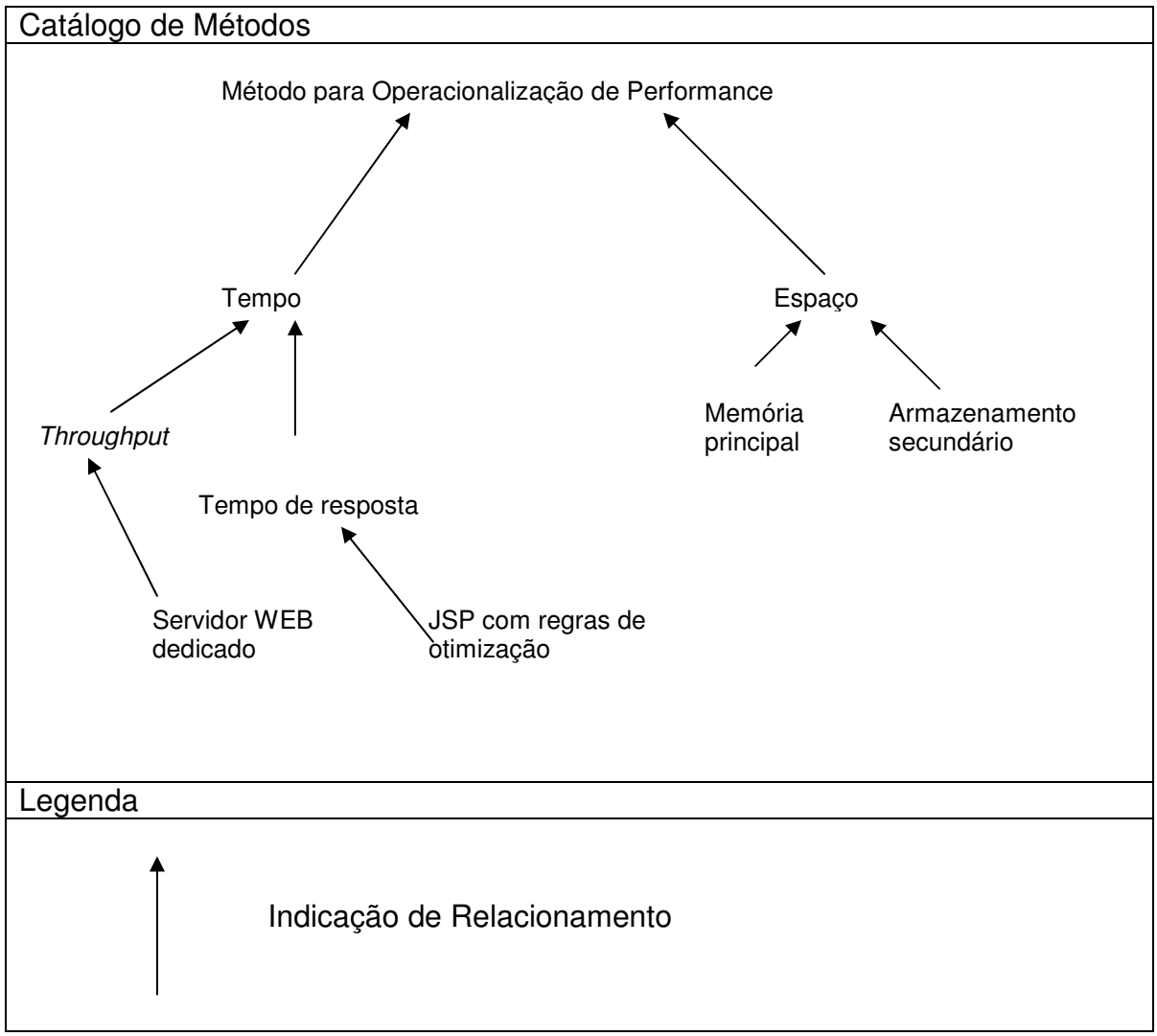


Figura B - Catálogo de Operacionalização de Métodos para Alcançar Performance (CHUNG et. al., 2000)

ANEXO C

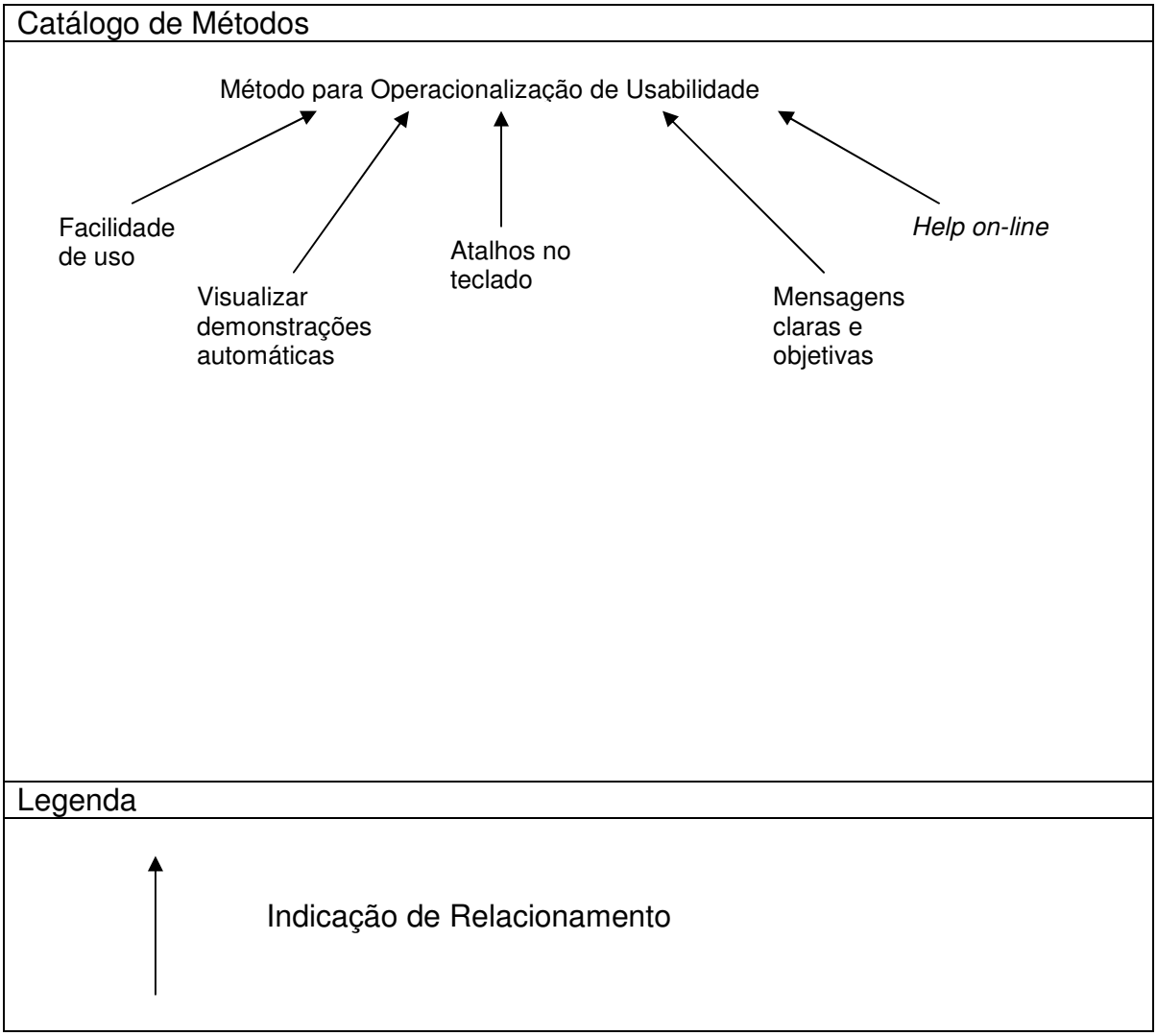


Figura C - Catálogo de Operacionalização de Métodos para Alcançar Usabilidade

ANEXO D

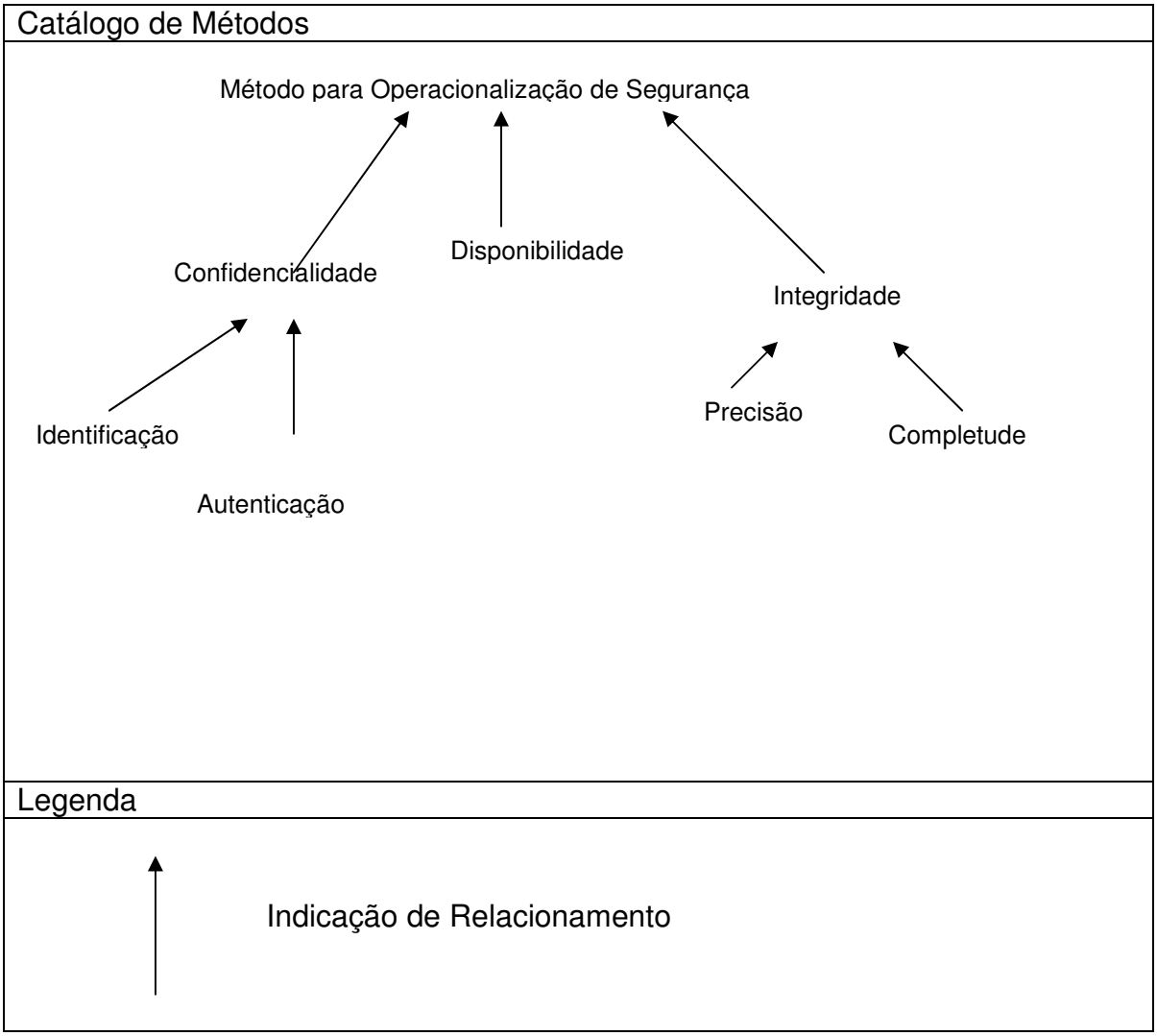


Figura D - Catálogo de Operacionalização de métodos para Alcançar Segurança (CHUNG et. al., 2000)