Universidade do Estado do Rio Grande do Norte – UERN
Universidade Federal Rural do Semi-Árido – UFERSA
Mestrado em Ciência da Computação

Cognitio: Um Processo para Reuso de Requisitos

Ceres Germanna Braga Morais

Mossoró/RN
Fevereiro de 2010

Cognitio: Um Processo para Reuso de Requisitos

Ceres Germanna Braga Morais

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semi-Árido como parte dos requisitos necessários para obtenção do grau de Mestre em Sistemas Computacionais.

Área de concentração: Engenharia de Software

Lyrene Fernandes da Silva, Dra.

Orientadora

Mossoró, Rio Grande do Norte, Brasil

À minha mãe por tudo que sou
e por tudo que quero ser.

Agradecimentos

A Deus, pelo dom da vida, pelas bênçãos derramadas e por ser Guia de todos os meus passos e decisões.

Aos meus pais, Raimunda Braga e José Avelino, pelo amor e dedicação, incentivo e apoio em todas as etapas da minha vida. Tudo o que sou agradeço a vocês!

Ao meu noivo Joamar Neto, pela compreensão, amor, atenção e carinho sempre demonstrados. Obrigada por estar sempre presente, apesar da distância.

Aos meus irmãos Claudinho e Gabriela, pela compreensão dos momentos não compartilhados, em favor dos meus estudos. A Ia, pelo carinho, atenção e preocupação a mim destinados.

Às minhas mais que tias Aninha, Bebel, Dodora e Maguinha, por desempenharem sempre o papel de mãe. A madrinha por ter me inspirado a participar do mestrado. A vovô e aos meus tios pelo apoio. Aos meus primos pelos bons momentos. Agradeço a todos pela preocupação, atenção e carinho.

A madrinha Irani e sua família, e a dona Evaldite, Fafá e Jaianny, por sempre me receberem com carinho em suas casas, sempre que precisei resolver assuntos do mestrado em Natal ou Mossoró. Obrigada pelo apoio e por serem parte da minha família.

À minha orientadora Lyrene Fernandes, pelas palavras de apoio, pelo conhecimento compartilhado, por sempre me ajudar e incentivar durante o mestrado, em especial, durante a fase de desenvolvimento da dissertação. Obrigada pela disponibilidade e por acreditar no meu potencial.

Aos professores Pedro Fernandes e Roberta Coelho, pela aceitação e disponibilidade em participar da banca avaliadora desta dissertação.

A todos os professores do Programa de Pós-Graduação em Ciência da Computação UERN/UFERSA, em especial a Lyrene Fernandes, Marcelino Pereira, Pedro Fernandes, Milton Mendes, Iguatemi Fonseca, Cláudia Ribeiro e João de Deus, com os quais tive contato direto durante estes dois anos, pelos ensinamentos e contribuições para a minha formação!

Às secretárias do mestrado, Rosita e Emanuela, por sempre estarem disponíveis a ajudar nas questões burocráticas do mestrado, e pela atenção e afetividade. E a Fatinha pelo carinho.

Aos professores da graduação, Cecília Raquel, Max Lopes e Sebastião Alves, por me incentivarem a me inscrever no mestrado.

Aos amigos que sempre torceram por mim, em especial a Ceicinha, Edeilson, Izabella, Lenita, Libânia, Luana, Marília, Priscila, Raqueline, Regina, Mário, Paulo Cesar, Pierre e Rafael. Obrigada pela força!

Aos amigos da graduação, em especial Jisreelly, Daniel, Alípio, João Paulo, João Batista, Samuel, Karla, Clayton e Paulo Henrique.

Aos amigos do mestrado: Alexandre, Alexsandra, Carlos Evandro, Cassimiro, Diego, Edsongley, Luana, Luiz Carlos, Max, Perla, Ticiana e Sebastião, com os quais dividi angústias, desafios e conquistas, e os quais levarei por toda minha vida. Muito obrigada amigos pelos momentos compartilhados!

Aos amigos servidores do IFRN Campus Ipanguaçu pelo incentivo e apoio.

A Thalles Robson, Anselmo Couto, Cláudia Cappelli e Rildo Santos, pela ajuda incondicional durante o desenvolvimento desta dissertação. Obrigada pela disponibilidade, paciência e conhecimento compartilhado.

À UERN e à Fapern, pelo apoio financeiro, através da concessão de bolsas de pesquisa.

Resumo

Devido à complexidade de se desenvolver softwares que atendam às necessidades dos *stakeholders*, torna-se cada vez maior a busca pelo desenvolvimento de métodos que auxiliem a Engenharia de Requisitos (ER), uma vez que esta é fator fundamental para o alcance da qualidade do software. Este trabalho está inserido no processo da ER com o intuito de possibilitar a elicitação de requisitos, através do reuso destes. Para tanto, foi desenvolvido um processo para reuso de requisitos, denominado *Cognitio*. Esse processo apóia o desenvolvimento de software com reuso e para reuso, visando conquistar ganhos na produtividade e qualidade dos softwares gerados. Através do processo *Cognitio*, pretende-se motivar os *stakeholders* tanto para desenvolver software com reuso quanto para o reuso. Para tanto, desenvolveu-se um repositório de requisitos, organizando-os com a linguagem Q7, e representando-os com o *framework* NFR, como o processo pode ser aplicado. Para dar apoio ao *Cognitio*, gerou-se a ferramenta *Cognitio Tool*.

Palavras-Chave:

Engenharia de requisitos; Reuso de requisitos; *Framework* NFR; Linguagem Q7; repositório de requisitos; processo de reuso de requisitos; ferramenta *Cognitio Tool*.

Abstract

Due to the complexity of developing software which match stakeholders' demands, the development of methods to support Requirement Engineering (RE) has been increasing more and more. It is, therefore, a fundamental factor in reaching software quality. This study deals with RE processing in order to promote the elicitation of requirements by reusing them. A requirement reusing process, named *Cognitio*, was created to support software development by and to reuse requirements. It aims at enhancing software productivity and quality, so that stakeholders feel motivated to develop software by reusing and to reuse them. A repository of requirements was done by organizing information about the requirements using Q7 language and to present, via NFR framework, how the process can be applied. In order to support *Cognitio*, the tool *Cognitio Tool* was created.

Keywords:

Requirements engineering; Requirements reuse; NFR Framework; Language Q7; requirements repository; requirements reuse process; *Cognitio Tool*.

Conteúdo

Lista de Símbolos e Abreviaturas

Abreviatura	Descrição
BPMN	Business Process Modeling Notation
ER	Engenharia de Requisitos
FAPERN	Fundação de Apoio à Pesquisa do Estado do Rio Grande do Norte
GBRAM	Goal-Based Requirements Analysis Method
GORE	Goal-Oriented Requirements Engineering
I*	Intentional Strategic Actor Relationships modelling
MER	Modelo Entidade-Relacionamento
NFR	Non-Functional Requirements
PHP	Hypertext Preprocessor
PIN	Personal Identification Number
RF	Requisito Funcional
RNF	Requisito não funcional
SGBD	Sistema Gerenciador de Banco de Dados
SIG	Softgoal Interdependency Graph
UdI	Universo de Informações
UML	Unified Modeling Language
XML	Extensible Markup Language

Lista de Tabelas

Lista de Figuras

Lista de Quadros

Capítulo 1

Introdução

1.1 Contextualização

A qualidade de um produto de software está intimamente ligada à qualidade de sua especificação (NUSEIBEH, EASTERBROOK, 2000). Tendo em vista a necessidade de se desenvolver sistemas de qualidade, entra em foco a Engenharia de Requisitos (ER), cujo papel fundamenta-se em descobrir o que o software a ser gerado deverá proporcionar aos usuários, através da análise e documentação dos requisitos (NUSEIBEH, EASTERBROOK, 2000).

No entanto, nem sempre a atividade de ER é valorizada no processo de desenvolvimento de software, causando, na maioria das vezes, a criação de um sistema que não atende à especificação dos usuários, elevando os custos durante e após a sua implementação.

Todavia, sendo a ER um processo complexo, busca-se a realização do reuso de requisitos, o qual consiste em usar, de forma sistemática, documentos de requisitos existentes com o intuito de diminuir o esforço e aumentar a qualidade dentro do ciclo de vida do software (VILLEGAS, LAGUNA, 2001).

Dessa forma, o desenvolvimento de software baseado no reuso de requisitos permite que a aplicação seja construída a partir de artefatos que já foram previamente especificados e validados, o que resulta no ganho da produtividade e na qualidade do software criado. O ganho na produtividade é conseguido tendo em vista que o esforço em desenvolver algumas atividades da ER, tais como especificação e análise de requisitos em um dado domínio, é diminuído. Já o ganho na qualidade é adquirido uma vez que os requisitos já foram utilizados, modificados e implementados em outros sistemas, o que elimina ou diminui possíveis erros de sua especificação.

Assim, para prover os benefícios do reuso de requisitos, o processo aqui apresentado busca aumentar a confiabilidade do sistema gerado e diminuir gastos e custos de tempo demandados devido ao desenvolvimento das atividades que compõem a ER, buscando a qualidade dos requisitos especificados.

1.2 Motivação e Problema

O reuso de requisitos é responsável pela diminuição do retrabalho demandado pelos engenheiros de requisitos durante a especificação destes, proporcionando a aquisição de maior confiabilidade e consistência dos produtos desenvolvidos.

Existem na literatura alguns trabalhos que apresentam processos de reuso de requisitos tanto para linhas de produtos, isto é, reuso em larga escala (MONZÓN, DUEÑAS, 2004), quanto para aplicações diversas (HOMOD, RINE, 1999) (TOVAL et al, 2002) (LOPEZ, LAGUNA, GARCIA, 2002) (LAM et al, 1997).

Logo, para se haver o reuso efetivo de requisitos, é extremamente importante que os engenheiros de requisitos e os desenvolvedores de software tenham acesso aos requisitos que podem ser reusados. Nesse caso, é

interessante que haja ferramentas de armazenamento que possam sistematizar o reuso, de forma que a busca e recuperação dos requisitos seja facilitada. Além disso, é interessante também que exista a definição de processos que conduzam os engenheiros de requisitos a definir seus requisitos permitindo o seu posterior reuso.

Dessa forma, este trabalho propõe o desenvolvimento de um processo de reuso e sua utilização como auxílio à elicitação de requisitos, uma vez que o mesmo dá suporte à construção de um repositório de requisitos, o qual serve como fonte de conhecimento capaz de guiar os engenheiros de requisitos durante a especificação dos requisitos necessários para sua aplicação.

Tendo em vista a busca pelo incentivo à atividade do reuso, foi especificado um processo de reuso de requisitos denominado *Cognitio* e uma ferramenta, *Cognitio Tool*. Do Latim, *Cognitio* significa Conhecimento. Este termo foi escolhido para denominar o processo de reuso especificado tendo em vista que, mais do que requisitos, o processo *Cognitio* armazena e reusa conhecimento.

Logo, tanto o processo *Cognitio*, quanto a ferramenta *Cognitio Tool*, foram especificados a fim de se estabelecer o reuso de requisitos de software, tanto no que diz respeito ao i) desenvolvimento de software com reuso; quanto ao ii) desenvolvimento de software para o reuso.

O desenvolvimento de software com o reuso é promovido uma vez que os envolvidos podem, através do *Cognitio* e da *Cognitio Tool* buscar e encontrar requisitos necessários para o software que está sendo desenvolvido, e assim recuperá-los para reuso. Por outro lado, o desenvolvimento de software para o reuso, se dá uma vez que o *Cognitio* especifica como os requisitos devem ser preparados, de forma que apresentem qualidade e consistência necessárias para o seu posterior reuso.

Assim, a principal motivação do desenvolvimento tanto do processo de reuso de requisitos quanto da ferramenta é o fornecimento de um mecanismo

que possa apoiar os engenheiros de requisitos, e demais interessados, durante a elicitação dos requisitos necessários para a sua aplicação, e incentivá-los à prática da especificação dos requisitos, objetivando o fornecimento dos mesmos para o reuso e para o desenvolvimento com reuso.

1.3 Limitações dos Trabalhos Relacionados

O reuso de requisitos é apresentado como uma abordagem importante para que softwares desenvolvidos a partir da prática do reuso apresentem atributos de qualidade. Para o desenvolvimento dessa dissertação, alguns trabalhos (KANG, 1990), (TOVAL et al, 2002), (LÓPEZ, LAGUNA, GARCIA, 2002), (MOROS, CHICOTE, TOVAl, 2008) e (EBLING, AUDY, PRIKLADNICK, 2009) que apresentam processos de reuso de requisitos foram usados como referência. Estes trabalhos relacionam práticas e ferramentas para promover o reuso de requisitos.

Entretanto, alguns destes trabalhos apresentam limitações, como por exemplo, em relação à organização das informações inerentes à modelagem de requisitos, tais como pré-rastreabilidade e contexto dos requisitos.

Além disso, nem todas as abordagens evidenciam em que momento do processo de Engenharia de Requisitos podem ser aplicadas, e nem como os requisitos devem ser classificados ou armazenados. O Capítulo 6 desta dissertação apresenta detalhadamente um comparativo entre os trabalhos relacionados e o processo *Cognitio*, mostrando as vantagens e desvantagens de cada um.

1.4 Solução Proposta

Para abordar o problema de como prover o reuso de requisitos com foco na qualidade do software desenvolvido, é proposto o processo *Cognitio* e a ferramenta *Cognitio Tool.*

Para apresentar como o processo de reuso é realizado, foi escolhido o *framework* NFR (CHUNG et al, 2000) para modelagem dos requisitos a serem reusados. Esta escolha foi realizada levando-se em consideração que o *framework* NFR expressa tanto requisitos funcionais quanto não funcionais, através dos conceitos de metas e softmetas. Além disso, essa abordagem vem sendo utilizada para catalogar requisitos estabelecendo o relacionamento e conflitos entre eles (SANTOS et al, 2000) (COUTO, 2009) (ANDREOPOULOS, 2004).

Uma vez que o processo *Cognitio* foi definido, verificou-se a necessidade de se classificar os requisitos armazenados, buscando auxiliar a compreensão por parte dos *stakeholders* durante o processo de aquisição dos ativos. Para tanto, a linguagem Q7 (LEITE et al, 2005), a qual consiste em uma linguagem de organização de qualidades, foi utilizada. Esta linguagem foi escolhida, uma vez que além de definir parâmetros importantes para a atividade de reuso de requisitos, pode ser adaptada para abranger outros modelos de representação de requisitos.

1.5 Contribuições da Dissertação

As principais contribuições desta dissertação são:

- A criação de um processo de reuso de requisitos de software que contempla o desenvolvimento de sistemas com reuso e para o reuso, bem como sua definição para uso genérico, independente da abordagem de

modelagem de requisitos escolhida, cabendo ao usuário adaptar o processo de acordo com as características do modelo;

- A adaptação da linguagem Q7 para a classificação das informações do *framework* NFR;
- A apresentação de como o *framework* NFR pode contribuir para a modelagem de requisitos que possam ser reusados posteriormente;
- O desenvolvimento da ferramenta *Cognitio Tool* para dar suporte ao processo de reuso, facilitando a realização das atividades que o compõem.

Assim, através dessas contribuições, busca-se que o reuso de requisitos seja uma atividade facilitada e incentivada, uma vez que foram definidos passos e mecanismos que além de permitir o acesso às informações reusáveis, estabelece que requisitos sejam preparados para o reuso.

1.6 Estrutura do Trabalho

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

O Capítulo 2 dá uma visão geral a cerca dos conceitos principais sobre Engenharia de Requisitos, reuso de requisitos, *framework* NFR e linguagem Q7.

O Capítulo 3 apresenta o processo de reuso de requisitos *Cognitio*, mostrando cada uma de suas atividades de forma detalhada.

O Capítulo 4 apresenta a ferramenta *Cognitio Tool*, desenvolvida para prover o apoio ao processo de reuso definido, através da realização das atividades de armazenamento, busca, recuperação dos requisitos, entre outras.

O Capítulo 5 refere-se ao estudo de caso, a fim de demonstrar o processo e a ferramenta de reuso de requisitos.

O Capítulo 6 traz alguns trabalhos relacionados, os quais apresentam processos de reuso de requisitos existentes na literatura.

O Capítulo 7 apresenta as contribuições deste trabalho, bem como são listados os trabalhos futuros.

Capítulo 2

Fundamentação teórica

Neste capítulo é apresentada uma visão geral acerca de alguns assuntos que estão relacionados com o processo de reuso de requisitos proposto nesta dissertação. Na Seção 2.1 são abordados conceitos de Engenharia de Requisitos. A Seção 2.2 apresenta a importância do reuso de requisitos para o processo de desenvolvimento de software. A Seção 2.3 aborda a Engenharia de Requisitos Orientada a Metas. E por fim, a Seção 2.4 apresenta as considerações finais do Capítulo.

2.1 Engenharia de Requisitos

Requisitos são entendidos como condição ou capacidade de que um usuário necessita para solucionar um problema específico, ou para atingir um objetivo, podendo ser divididos em funcionais, os quais descrevem uma determinada ação que o sistema deve dar suporte; e não-funcionais, os quais descrevem qualidades que o sistema deve satisfazer (RAMOS, 1999).

Para tanto, a ER é uma área que se preocupa com a elicitação de requisitos, sua documentação e respectivo gerenciamento (RAMOS, 1999),

auxiliando a modelagem e análise de sistemas, de forma que suas características sejam bem entendidas antes de serem implementadas (ROBINSON, PAWLOWSKI, VOLKOV, 2003). Esta atividade está relacionada com a identificação e operacionalização de metas a serem atingidas pelo sistema a ser desenvolvido.

De acordo com Sommerville, Sawyer e Viller (1998), a ER consiste na atividade realizada para cobrir todas as atividades envolvidas na descoberta, documentação, e manutenção de um conjunto de requisitos, de um determinado sistema computacional.

Logo, a postura da área de ER é a de "prover, ao engenheiro de software, os métodos, técnicas e ferramentas que auxiliam o processo de compreensão e registro dos requisitos que o software deve atender" (LEITE, 2006), cabendo a ela aperfeiçoar processos de descobrimento, análise, negociação, validação e documentação dos requisitos, para que os mesmos possam ser gerenciados com mais precisão (CHAVES, 2005).

Como produto, a ER provê modelos a partir do qual o documento de requisitos é produzido. Para isso, há o Universo de Informações (UdI), que consiste em um contexto previamente definido, do qual são retirados os requisitos necessários para o desenvolvimento de uma aplicação (LEITE, 1994).

Para melhor entendimento desses conceitos, a Seção 2.1.1 apresenta o Processo de Engenharia de Requisitos; a Seção 2.1.2 aborda os requisitos funcionais (RF); a Seção 2.1.3 mostra conceitos relacionados aos requisitos não-funcionais (RNF).

2.1.1 Processo de Engenharia de Requisitos

De acordo com Sommerville e Sawer (1997) e Kotonya e Sommerville (1998), o processo de ER consiste em um conjunto estruturado de atividades para conhecer os requisitos, validá-los e mantê-los em um documento de requisitos.

Essas atividades, de acordo com Sommerville, Sawyer e Viller (1998) e Nuseibeh e Easterbrook (2000), dividem-se em:

i) **Elicitação de requisitos**: nesta fase, os requisitos são descobertos através de métodos de elicitação de requisitos, sejam eles questionários, entrevistas, reuniões, *brainstorm*, entre outros. Esses métodos são previamente escolhidos pela equipe de ER e realizados com a participação dos *stakeholders*.

ii) **Análise e negociação dos requisitos**: aqui, os requisitos são analisados em relação a sua completude, necessidade, consistência e praticabilidade no contexto de custos e tempo disponível.

iii) **Documento de requisitos**: nesta fase, um documento dos requisitos elicitados e analisados é desenvolvido, descrevendo o domínio da aplicação e o sistema a ser implementado.

iv) **Verificação e validação de requisitos**: nesta fase, a consistência do documento de requisitos é checada. Isso pode ser realizado através de uma observação detalhada do documento de requisitos, por meio da execução de testes e inspeções, entre outros métodos.

v) **Gerenciamento de requisitos**: esta fase envolve a coleta, armazenamento e manutenção de informações que referem-se aos impactos que mudanças, inserções ou exclusões de requisitos podem exercer sobre o sistema e sobre outros requisitos.

A Figura 2.1 apresenta um modelo genérico do processo de ER, adaptado de (KOTONYA, SOMMERVILLE, 1998).

As atividades especificadas na Figura 2.1 ocorrem de forma iterativa, iniciando com a **elicitação de requisitos**, a qual consiste da observação das necessidades dos usuários, da busca por sistemas existentes, da consulta de regulamentos e padrões que podem influenciar no desenvolvimento do sistema.

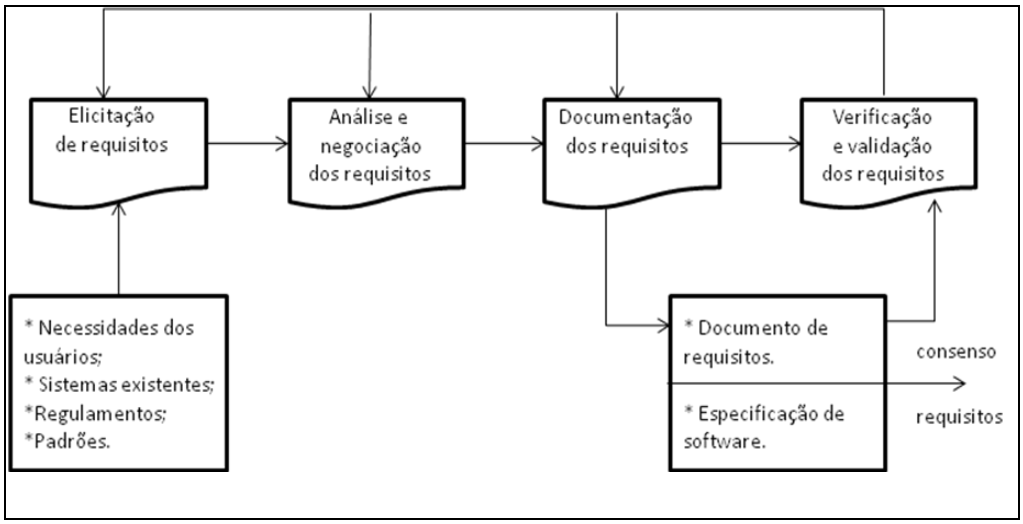

Figura 2.1: Processo de ER, adaptado de (KOTONYA, SOMMERVILLE, 1998)

Realizada esta fase, ocorre a **análise e negociação dos requisitos**, a **documentação dos requisitos** elicitados e analisados, e a partir da documentação, é realizada a **verificação e validação dos requisitos**. No entanto, a **verificação e validação** dependem também do consenso dos *stakeholders* em relação aos requisitos obtidos.

Embora diferentes projetos exijam processos com características específicas para contemplar as suas particularidades, Leite e Doorn (2004) definiram um grupo de atividades básicas que devem ser considerados na definição de qualquer processo de ER, composto da elicitação, análise e modelagem dos requisitos.

Dessa forma é através da ER que os requisitos do sistema são encontrados e definidos. Na literatura existem algumas formas de classificação de requisitos de software como, por exemplo, requisitos do usuário, do sistema, funcionais e não funcionais, inversos. Para o escopo deste trabalho é utilizada a classificação dos requisitos em funcionais e não funcionais, os quais estão apresentados nas Seções 2.1.2 e 2.1.3 a seguir.

2.1.2 Requisitos Funcionais

De acordo com Cysneiros (2001), requisitos funcionais (RF) expressam funções ou serviços que um software deve ou pode ser capaz de executar ou fornecer, definem o comportamento do sistema, especificando o que seus componentes deverão ser capazes de implementar, usando as entradas dos processos para obter as saídas esperadas (MACEDO, LEITE, 1999). A Figura 2.2 apresenta a taxonomia dos RFs.

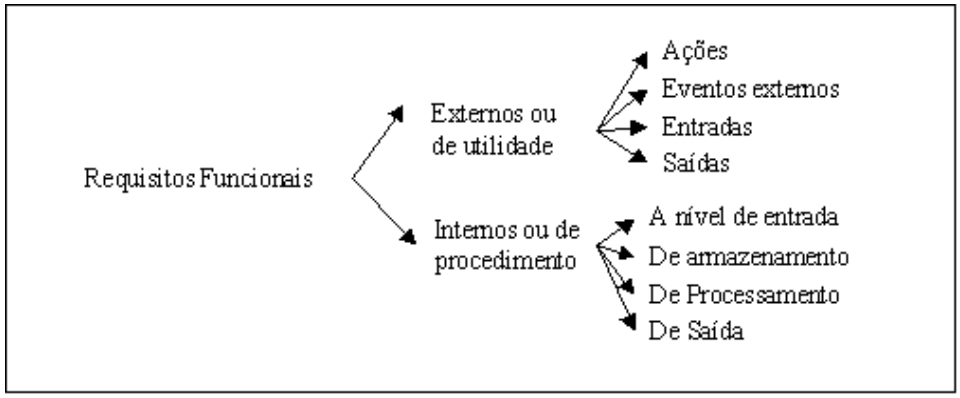

Figura 2.2: Taxonomia dos requisitos funcionais (MACEDO, LEITE, 1999)

Assim, como mostra a Figura 2.2, os RFs podem ser externos ou internos. Os externos compreendem as interações com os usuários e com outros sistemas, enquanto que os RFs internos compreendem os requisitos especificados para realizações inerentes ao sistema. Portanto, a definição correta dos RFs é de suma importância para que o sistema desenvolvido realize suas funções de forma a atender a todas as especificações que a ele foram atribuídas.

2.1.3 Requisitos Não Funcionais

Os requisitos não funcionais (RNF) são entendidos como atributos de qualidades (BOEHM, 1978) (KELLER, 1990), objetivos (MOSTOW, 1985), restrições (ROMAN, 1985), confiabilidade, entre outros. Esses requisitos

desempenham um papel crítico durante o desenvolvimento de software, e problemas ocorridos por sua má elicitação, tornam-se bastante dispendiosos, se o desenvolvimento do sistema já tiver sido finalizado (BROOKS, 1987) (DAVIS, 1993). A Figura 2.3 apresenta a taxonomia dos RNFs.

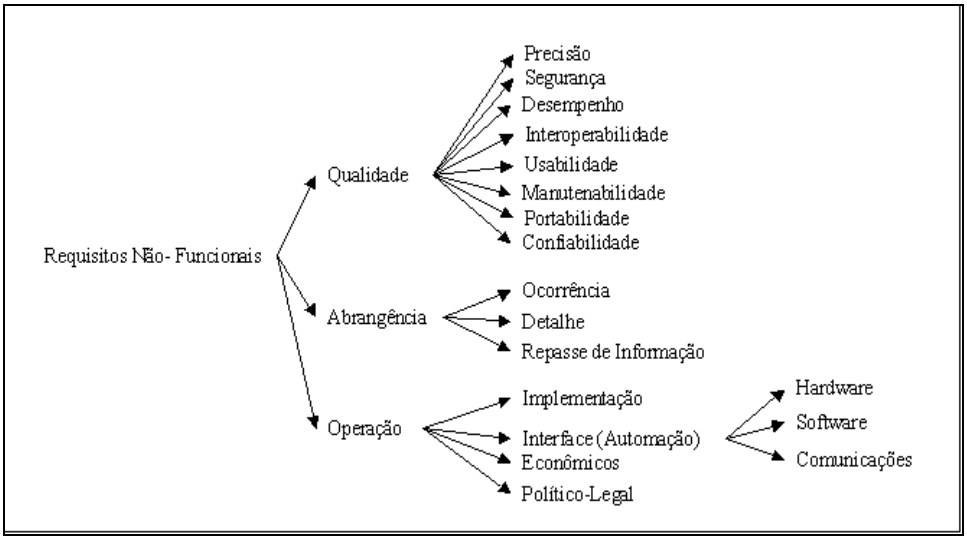

Figura 2.3: Requisitos Não-Funcionais (MACEDO, LEITE, 1999)

Como apresenta a Figura 2.3, RNFs abordam importantes aspectos relacionados à qualidade de softwares. Eles são essenciais para que softwares apresentem as características para as quais foi especificado (CYSNEIROS, 2001). Esse tipo de requisito é geralmente subjetivo, uma vez que pode ser visto, interpretado e conceituado de forma diferente por diferentes pessoas (CHUNG et al, 2000).

Como vários RNFs possuem efeitos de natureza global nos softwares, uma solução localizada pode não ser adequada (CYSNEIROS, 2001). Dessa forma, estes requisitos são considerados difíceis de lidar e vitais de serem tratados para que se possa obter softwares de qualidade.

Tendo em vista os conceitos e as peculiaridades tanto dos RFs quanto dos RNFs, vê-se a importância de se reutilizar requisitos que já foram especificados e implementados anteriormente, a fim de prover a qualidade dos sistemas

desenvolvidos. Dessa forma, a Seção 2.2 a seguir apresenta os conceitos de reuso de requisito, suas vantagens, e como o mesmo pode ser alcançado.

2.2 Reuso de Requisitos

O reuso de requisitos consiste em uma abordagem para sistematizar o uso de documentos de requisitos existentes com o objetivo de reduzir o esforço desprendido nas fases do ciclo de vida do software (VILLEGAS, LAGUNA, 2001), para o desenvolvimento de uma nova aplicação. Essa abordagem é útil para dar assistência e orientação ao engenheiro de requisitos, durante todo o processo de elicitação de requisitos (ROLLAND, PRAKASH, 1999).

A redução do esforço através do reuso de requisitos se dá de duas formas: melhorando o processo de ER (CYBULSKI, 1998) e dando suporte ao processo de desenvolvimento de software através do reuso de informações (BELLIZONA, FUGINI, PERCINI, 1992).

Com o reuso, requisitos similares já implementados podem ser apresentados a novos *stakeholders* e erros na especificação de novos requisitos podem ser diminuídos ou evitados (DAY, 2004). Além disso, ocorre o aumento da produtividade, aumento da qualidade, da padronização, da interoperabilidade, previsibilidade, confiabilidade, redução dos custos, no tempo de entrega do sistema e dos riscos (KOSCIANSKI, SOARES, 2006).

Desta forma, o reuso de requisitos é visto como uma das necessidades mais prementes e um dos maiores desafios no âmbito da ER, de forma que seu uso gera soluções que podem proporcionar grande impacto nos estudos e na prática da Engenharia de Software (MOROS, CHICOTE, TOVAL, 2008).

Em seu trabalho, Lam et al (1997) apresenta alguns passos para se alcançar o reuso de requisitos, tais como: ter cuidado com requisitos que apresentam conceitos generalizados; buscar identificar sistemas de uma mesma linha/família para maximizar o reuso; desenvolver um modelo de documento

de requisitos com base em conceitos abstraídos durante a elicitação destes; tornar explícito o que deverá/poderá ser reutilizado para evitar desvios durante a atividade de reuso; e compreender que partes do processo de ER também são reutilizáveis.

De acordo com Silva (2006), o reuso de requisitos ocorre especialmente quando se tratam de RNFs, haja vista que geralmente são semelhantes mesmo em diferentes aplicações. Nos trabalhos (CYSNEIROS, 2007) (SOMMERVILLE, 2000) é proposta a catalogação dos RNFs para que eles possam ser reusados. Além disso, a utilização de artefatos como cenários e discriminantes são utilizados em diversas abordagens para se obter um maior aproveitamento do reuso de requisitos (FOOK, ABDELOUAHAB, 2001).

Outra forma de prover o reuso de requisitos é através do encapsulamento de requisitos duplicados em uma estrutura única, facilitando o reuso destes por outras estruturas do sistema, ou de sistemas diferentes (RAMOS et al, 2007).

Alguns aspectos como não-conformidades, erros lógicos, erros conceituais e omissões durante a especificação de requisitos podem trazer baixa qualidade ao software, dificultando, portanto, o seu reuso (KOSCIANSKI, SOARES, 2006). Ao contrário, se os requisitos apresentam corretude, precisão, completude, consistência, verificabilidade, modificabilidade e rastreabilidade, pode-se dizer que os mesmos favorecem à qualidade do software e, dessa forma, o reuso pode ajudar o desenvolvimento de software (KOSCIANSKI, SOARES, 2006).

No entanto, para que essa atividade seja exercida de forma a beneficiar o processo de desenvolvimento, os requisitos devem estar bem documentados, permitindo que os desenvolvedores tenham facilidade em compreendê-los, adaptá-los e integrá-los (ALMEIDA, 2004).

Assim, tendo em vista os conceitos apresentados, observa-se a necessidade da existência de um processo para o reuso de requisitos, a fim de

proporcionar aos *stakeholders* um local de fácil acesso e captura de requisitos que sejam desejáveis para o desenvolvimento de sua aplicação.

Processos de Reuso de Requisitos

Para estabelecer o desenvolvimento do *Cognitio*, realizou-se um estudo detalhado de outros processos de reuso de requisitos existentes na literatura. Alguns destes trabalhos (KANG, 1990), (TOVAL et al, 2002), (LÓPEZ, LAGUNA e GARCIA, 2002), (MOROS et al, 2008) e (EBLING et al, 2009), são listados a seguir, e são comparados com o *Cognitio* no Capítulo 6 desta dissertação.

i) Método FODA

O método FODA (KANG, 1990) consiste em um método de análise de domínio que visa à construção de artefatos genéricos que sejam largamente reutilizáveis dentro do domínio. O objetivo principal da análise de domínio é obter e representar informações sobre sistemas de software que compartilham características comuns. O FODA consiste nas atividades de Análise de Contexto e Modelagem de Domínio.

A atividade de modelagem do domínio é subdividida nas atividades: Análise de Características, Modelagem Entidade-Relacionamento e Análise Funcional. Dentre os métodos de análise de domínio, o método FODA é considerado o mais maduro e documentado. No entanto, o mesmo apresenta limitações tais como a falta de orientação quanto à modelagem da arquitetura do domínio. As atividades deste método são apresentadas a seguir.

A Análise de Contexto define o escopo do domínio que será explorado para obter artefatos do domínio. Nesta fase, as relações existentes entre os domínios candidatos e os elementos externos ao domínio são analisadas, e a variabilidade destas relações são analisadas. O resultado final da análise de

contexto deve ser documentado em um modelo de contexto, o qual define os limites do domínio que será modelado na próxima fase.

A Modelagem de Domínio analisa as características comuns e variáveis entre as aplicações, e vários modelos cobrindo diferentes aspectos do domínio são produzidos. Esta fase consiste em análise de características, modelagem entidade-relacionamento e análise funcional.

ii) Método SIREN de Reuso de Requisitos

O método SIREN (TOVAL et al, 2002) se baseia na utilização de um modelo de processo em espiral, umas planilhas de documentos de requisitos e um repositório de requisitos reutilizáveis, que se encontra organizado por catálogos.

Os catálogos de requisitos de SIREN se organizam de acordo com *perfis* e *domínios*. Estes catálogos se compõem de uma hierarquia de documentos de especificação, estruturados de acordo com padrões do IEEE.

Todo requisito em SIREN tem um conjunto mínimo de atributos, mesmo que dependendo do catálogo pode ser definidos atributos adicionais. Um dos atributos dos requisitos são as relações de *rastreabilidade exclusiva*, que implicam que só um de os requisitos que mantenham esta relação poderá ser incluído em um mesmo projeto.

iii) Metamodelo para Reuso de Requisitos

Desenvolvido por López, Laguna e Garcia (2002), esse metamodelo surgiu tendo em vista a existência de diversas técnicas de modelagem para diferentes paradigmas de desenvolvimento. Dessa forma, os autores observaram a necessidade de haver um esquema conceitual para descrever os requisitos em um *framework* de reuso, denominado Metamodelo.

Os elementos centrais do metamodelo são: modelo de representação de requisitos (representa diferentes diagramas de requisitos); Unidade de

modelagem (descreve unidades que estão inseridas nos diagramas de requisitos); e o Objetivo de domínio (representa as características do modelo de representação dos requisitos).

O modelo de representação de requisitos está relacionado a apenas um projeto de requisitos, de forma que este é caracterizado pelo objetivo do domínio, isto é, o conhecimento do domínio.

Em relação às unidades de modelagem, estas podem representar uma dependência, generalização ou associação entre si. Todos esses tipos de relacionamentos mostram a direção estabelecida entre os elementos de ação como fonte enquanto outros elementos de ação são atingidos.

Para prover o metamodelo, os autores desenvolveram a ferramenta para reuso de requisitos (R^2), composto por oito elementos principais, os quais são: Interface com o usuário, Editor de requisitos, Gerenciamento de dados, Repositório, Analisador de consistência, Gerenciador de léxico, Tradutor de diagramas, e Gerenciamento de repositório.

iv) Metamodelo de Variabilidade para Reuso de Requisitos

Definido por Moros et al (2008), esse metamodelo consiste em uma abordagem de reuso de requisitos denominado Requirement Engineering MetaModel (REMM), o qual provê a modelagem de variabilidade de forma direta, através de artefatos de requisitos.

Assim, esse metamodelo destina-se a modelagem tanto de catálogos de requisitos para o reuso quanto de produtos de requisitos gerados por meio do reuso. O mesmo consiste em um primeiro passo para integrar requisitos reusáveis na abordagem Model-Driven Software Development (MDSD). Além disso, o REMM provê de uma ferramenta denominada REMM-Studio a qual foi implementada para dar suporte a modelagem de requisitos, reuso e validação.

O conceito central incluído no REMM é o repositório que contém o catálogo de requisitos reusáveis, o qual inclui um conjunto de requisitos

reusáveis que pertencem a um mesmo catálogo de tipos de requisitos, ou seja, a um mesmo perfil ou domínio. Esses elementos pertencem à faceta "modelagem para reuso" dentro do metamodelo.

Já para a faceta "modelagem com o reuso", existe como conceito central o catálogo de produto, o qual possui todos os requisitos do produto relacionados para a especificação de um novo produto. Um requisito de produto pode ser algo específico ou ele pode ser reusado a partir do catálogo de requisitos reusáveis disponíveis no repositório. Neste caso, o requisito do produto mantém uma rastreabilidade com a fonte requisitos reusáveis, através da associação "reusado de", de forma que a especificação dos requisitos pode ser consultada a qualquer momento.

v) Método de Reuso de Requisitos para Linhas de Produto em Ambiente Distribuído

Este método descrito por (EBLING et al, 2009) foi desenvolvido com o intuito de reduzir as dificuldades encontradas na engenharia de requisitos distribuída através da integração do reuso de requisitos e linhas de produto.

A iteração inicial começa no passo definição inicial, o qual estabelece algumas definições para o reuso de requisitos usando linhas de produtos em desenvolvimento de software distribuído. O passo seguinte é a definição do domínio dos requisitos, onde os *assets* centrais da organização são criados, através da documentação de requisitos, ou da coleção de *features* e requisitos existentes. O terceiro passo é a definição dos requisitos de produto, onde os produtos da organização são criados através do reuso dos *assets* definidos na fase anterior.

O passo suporte de desenvolvimento de software distribuído ocorre de forma iterativa com os três apresentados acima, o qual auxilia a comunicação entre as equipes distribuídas, além de prover cursos, treinamentos, etc. Outro passo que ocorre de maneira semelhante é o gerenciamento de linhas de

produto, onde artefatos, processo de reuso, dicionário de linha de produto e base cultural são frequentemente gerenciados.

A partir da execução desses passos, os próximos iniciam-se no suporte de desenvolvimento de software distribuído, e são totalmente iterativos entre si.

2.3 Engenharia de Requisitos Orientada a Metas

A Engenharia de Requisitos Orientada a Metas (GORE – *Goal-Oriented Requirements Engineering*) está preocupada com o uso de metas para elicitar, elaborarar, estruturar, especificar, analisar, negociar, documentar e modificar requisitos (LAMSWEERDE, 2001). Algumas técnicas de GORE são encontradas na literatura, tais como: GBRAM (ANTÓN, 1996), Kaos (DARDENNE, LAMSWEERD, FICKAS, 1993), *Framework* NFR (CHUNG et al, 2000), V-Graph (YU, LEITE, MYLOPOULOS, 2004) e o i* (*Intentional STrategic Actor Relationships* – I-STAR) (YU, 1995), haja vista que a GORE apresenta as seguintes vantagens, dentre outras (LAMSWEERDE, 2001):

- O refinamento de metas oferece um mecanismo natural de estruturação dos documentos de requisitos complexos, visualização e maior facilidade de compreensão;
- As metas ajudam a detectar conflitos existentes entre requisitos; e
- As metas de alto-nível de um determinado sistema oferecem uma visão resumida a respeito do gerenciamento da evolução dos requisitos.

De acordo com (ANTÓN, 1997), metas são objetivos a serem alcançados pelo sistema. Assim, segundo (LAMSWEERDE, 2000) metas são descrições precisas de um objetivo cuja sua satisfação requer a cooperação de agentes (ou componentes ativos) no software e seu ambiente.

A utilização da GORE, no processo de ER, baseia-se em modelos de multivisão, os quais mostram de que forma as metas, operações, cenários e

objetivos se inter-relacionam com as propriedades de um determinado domínio em um sistema (SANTOS, 2008).

Uma meta funcional geralmente captura um conjunto de cenários desejado, sendo assim, estabelecida em um sentido claro. Já as metas de qualidade representam os RNFs de uma aplicação, e capturam características que não são representadas por metas funcionais, de forma que podem impor restrições para as operacionalizações (SANTOS, 2008).

Nesta dissertação, utiliza-se o *framework* NFR, que consiste em um modelo para representar e analisar relacionamentos positivos e negativos entre requisitos. Os conceitos do *framework* NFR são apresentados na Seção 2.3.1.

2.3.1 *Framework* NFR

O *framework* NFR (*Non-Functional Requirements Framework*) (CHUNG et al, 2000) baseia-se em metas para modelar e analisar RNFs, contribuindo para que os desenvolvedores tratem os RNFs de modo a expressá-los sistematicamente, e usá-los para realizar o processo de desenvolvimento de software racionalmente (COUTO, 2009).

Portanto, com o *framework* NFR, os RNFs são representados na forma de metas a serem atingidas. Uma meta pode interagir com outras metas e não precisam ser completamente satisfeitas, podendo haver uma satisfação parcial, negativa ou positiva (ANDREOPOULOS, 2004). Assim, o *framework* NFR pode modelar RNFs como segurança, desempenho, entre outros (CHUNG et al, 2000).

Considerando o fato de que a satisfação dos RNFs de um sistema é tão crucial quanto prover suas funcionalidades (CHUNG, NIXON, YU, 1994) e que abordar a estrutura de sistemas através de seus RFs é uma tarefa menos complexa (CHUNG et al., 2000), o *framework* NFR veio preencher as necessidades para a representação dos RNFs e apresentar uma metodologia

para encontrar estes requisitos, resolvendo possíveis conflitos e facilitando a rastreabilidade dos mesmos (CHUNG, 1991).

Com essas características, este *framework* auxilia os desenvolvedores a produzir soluções direcionadas a cada domínio, de acordo com o que foi aprendido em experiências anteriores, além de facilitar ao desenvolvedor a representação, organização, análise e utilização de conhecimentos específicos a respeito de RNFs.

Neste trabalho, considera-se que algumas softmetas de operacionalização (metas), modeladas pelo *framework* NFR, podem ser classificadas como requisitos funcionais, uma vez que geralmente apresentam ações a serem realizadas pelo sistema. Desta forma, entende-se que com a modelagem dos requisitos neste *framework*, tanto RNFs como RFs são abordados, podendo, então, no processo aqui definido, tanto um quanto o outro ser reusado.

Para permitir esse auxílio, o *framework* NFR provê alguns artefatos tais como softmetas, catálogos e grafos SIGs (*Softgoal Interdependence Graph*). As softmetas representam requisitos de qualidade; os catálogos contêm conhecimentos sobre os RNFs, sendo divididos em catálogo de tipos de RNFs, de operacionalização e de correlação; e os SIGs apresentam todas as softmetas relevantes ao domínio, bem como as relações existentes entre elas (CHUNG et al, 2000).

Assim, enquanto uma softmeta representa os RNFs da aplicação, o elemento operacionalização representa as ações que estão atreladas às softmetas, as quais geralmente representam os RFs modelados. Para este trabalho, utiliza-se o grafo SIG como exemplo para a representação dos requisitos a serem reusados.

Grafo SIG

Os grafos SIGs apresentam aos *stakeholders* vários conceitos do *framework* NFR, uma vez que é composto de softmetas, interdependências e tipos de

contribuição, os quais são utilizados no processo de reuso de requisitos aqui definido.

As softmetas podem ser do tipo: NFR, as quais representam os RNFs a serem satisfeitos na modelagem; operacionalização, que fornecem mecanismos para que os RNFs sejam atingidos; e argumento, que explicam por que um determinado relacionamento deve ser satisfeito. Essas três softmetas são representadas graficamente por nuvens, como apresenta a Figura 2.4.

Figura 2.4: Representação gráfica de uma softmeta (COUTO, 2009)

Para simplificar a referência das softmetas, no processo de reuso de requisitos aqui apresentado, a Softmeta de RNF é denominada **Softmeta**; a Softmeta de operacionalização denomina-se **Meta**; e a Softmeta de argumentação é tratada como ***Claim***. Dessa forma, a partir desse ponto, são utilizados os termos softmeta, meta e *claim*, como referência a estes elementos.

Em relação às dependências do *framework*, estas podem ser do tipo: decomposição, operacionalização ou argumentação.

A **decomposição** ocorre quando um elemento é refinado em outro do mesmo tipo, por exemplo, quando uma softmeta é refinada em outras softmetas. Este tipo de interdependência pode ser do tipo E (*And*) ou Ou (*Or*). A Figura 2.5 apresenta esse tipo de interdependência.

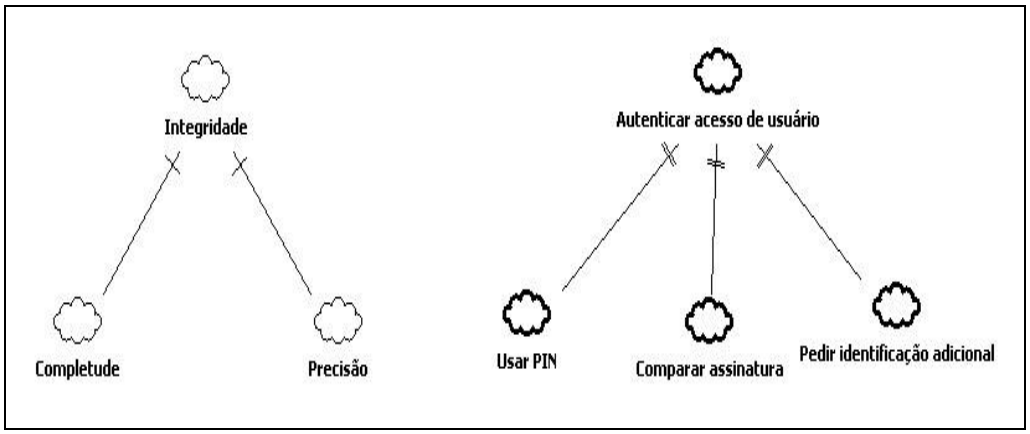

Figura 2.5: Decomposições do tipo E e do tipo OU, respectivamente

As decomposições do tipo E indicam que para que um determinado requisito seja atingido, todos os requisitos que estão ligados a ele, através desta decomposição, devem ser atendidos. Já a decomposição do tipo OU estabelece que nem todos os requisitos devem ser contemplados para que o requisito ao qual eles estão ligados seja alcançado.

Na **operacionalização**, as softmetas são refinadas em metas. Nesse caso, o *framework* NFR possui as notações *Make*, *Help*, *Break* e *Hurt*. A notação *Make* (++) modela uma contribuição suficiente/positiva; a notação *Help* (+) modela uma contribuição parcial/positiva; a notação *Break* (--) modela uma contribuição totalmente negativa; e a notação *Hurt* (-) modela uma contribuição parcial/negativa (CASTOR, 2004). A Figura 2.6 apresenta esses tipos de relacionamentos.

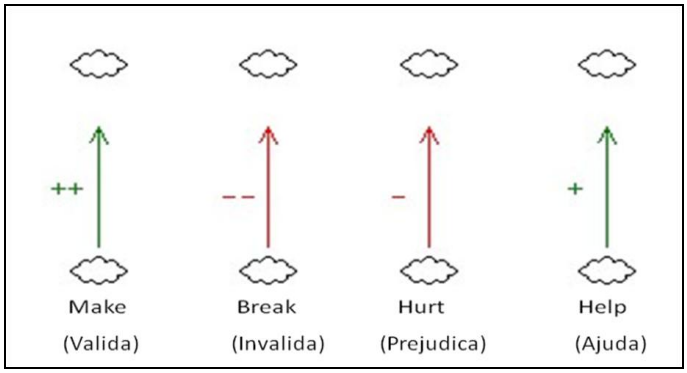

Figura 2.6: Representação das interdependências de operacionalização

Por último, a interdependência **argumentação** funciona como justificativa ou alerta do porque de uma determinada interdependência existir. A Figura 2.7 apresenta esse tipo de relacionamento.

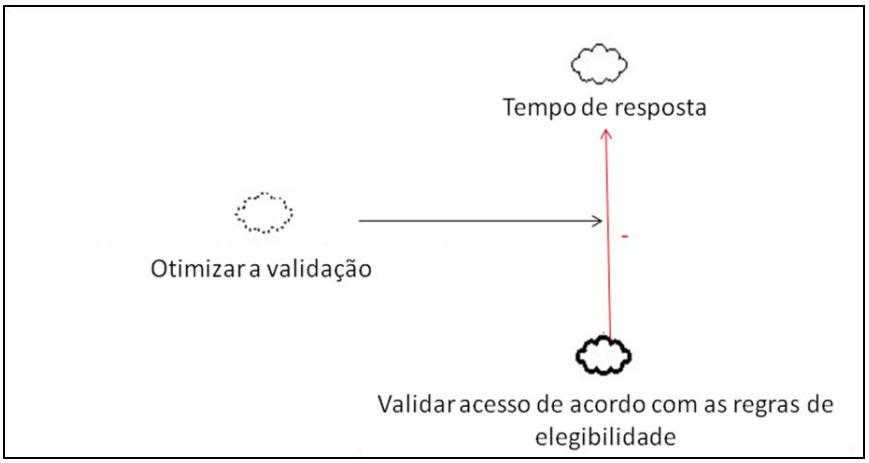

Figura 2.7: Interdependência do tipo argumentação

No exemplo da Figura 2.7, está representada a softmeta **tempo de resposta**, que relaciona-se operacionalmente com a meta **validar acesso de acordo com as regras de elegibilidade**, de forma que a segunda contribui de forma *hurt* com a primeira. O argumento **otimizar a validação** está relacionada na forma de argumento, com o relacionamento de operacionalização *hurt*.

O *framework* NFR foi escolhido para modelar os requisitos utilizados no processo de reuso, tendo em vista que seus conceitos são definidos de forma simples, em relação a outras técnicas de modelagem de requisitos orientadas a metas.

Dessa forma, tendo em vista a necessidade de se classificar os grafos SIG do *framework* no processo de reuso de requisitos aqui apresentado, a linguagem Q7, definida em (LEITE et al, 2005), foi escolhida. Esta consiste em uma linguagem de organização de qualidades, inserida também no contexto de orientação a metas. A mesma está apresentada na Seção 2.3.2 a seguir.

2.3.2 Q7: Linguagem de Organização de Qualidade

De acordo com Uddin, Mezbah-ul-Islam e Haque (2006), a classificação é uma maneira de organizar as informações em categorias, de acordo com suas semelhanças e diferenças. Ela pode ser usada como uma representação rica do que é conhecido e, portanto, ser bastante útil durante a comunicação, geração de novos ciclos de exploração, comparação e teorização, de forma que uma boa classificação reúne os seus conceitos em uma estrutura que auxilia os interessados (KWASNIK, 1999).

A Linguagem Q7consiste de sete questões representadas como 5W2H (*why*, *who*, *what*, *where*, *when*, *how* e *how much*). Esta linguagem foi desenvolvida a fim de assegurar tanto características de qualidade, quanto o relacionamento dessas características com descrições funcionais, dentro da abordagem de orientação a metas e de orientação a aspectos.

A orientação a aspectos é uma opção para que engenheiros de software organizem melhor todos os artefatos produzidos nas diversas fases de desenvolvimento, desde os documentos de requisitos (RASHID et al, 2002) até a implementação (KICZALES et al, 1997). Esta abordagem traz alguns conceitos tais como características transversais (*crosscutting concerns*), aspectos (*aspect*), ponto de junção (*join point*), conjunto de junção (*pointcut*), processo de combinação (*weaving*), comportamento transversal (*advice*) e declaração intertipo (*inter-type declaration*), de forma a representar de maneira mais clara como os requisitos são modelados e relacionados.

Embora essa abordagem trate com bastante veemência como os requisitos se relacionam, e como estes exercem conflitos entre si, a mesma não está incluída no escopo deste trabalho, mas nos trabalhos futuros é citado o tratamento de aspectos durante o reuso de requisitos. Para este trabalho, a linguagem Q7 é de suma importância, haja vista que é capaz de representar

conceitos tais como funções, pré-condições, contribuições, entre outros aspectos que contribuem para a classificação de requisitos a serem reusados.

Além disso, embora nessa dissertação o *framework* NFR seja apresentado como abordagem para modelagem dos requisitos e como a Q7 classifica as informações deste *framework*, esta linguagem pode ser associada a outros modelos de representação de requisitos, de forma que permita a realização do processo de reuso de requisitos que proposto em outras abordagens.

Para melhor entendimento, cada um dos parâmetros da linguagem Q7 está definido a seguir:

- *Why*: Para Leite et al (2005), a questão *why* é a central para a visão de qualidade, uma vez que RNFs foram propostos inicialmente para descrever atributos de qualidade para responder a perguntas como "Por que um determinado artefato necessita de um atributo de qualidade?". Logo, esta questão se refere à softmeta ou ao atributo de qualidade de deseja-se reusar.
- *Who*: esta questão caracteriza a meta principal do atributo de qualidade. Assim, o *who* representa o artefato associado com a softmeta ou a qualidade que deseja-se atingir.
- *What*: a questão *what* caracteriza a informação contextual de um dado *who*, que será a meta de um atributo de qualidade *why*. *Where*: este parâmetro é o endereço específico do requisito de qualidade de um artefato. No modelo de metas V-Graph, o *where* é o *pointcut (pointcuts* indicam os elementos afetados por uma determinada característica transversal, são mecanismos que encapsulam os *joinpoints. Joinpoints*, por sua vez, são locais bem definidos na estrutura ou fluxo de execução de programa onde comportamentos adicionais podem ser adicionados, ou seja, são afetados por aspectos (FILMAN et al., 2005)) em que os aspectos são apontados, isto é, os pontos em que as metas definidas pela questão

why são compostas. Para este trabalho, esta definição foi adaptada, uma vez que não se trata de orientação a aspectos.

- *When*: esta questão é usada para indicar uma pré-condição que necessita ser realizada antes que uma meta ou operacionalização (*how*) seja aplicada em certo *pointcut* (*where*). No *framework* NFR, o *when* indica os *claims*.
- *How*: a questão *how* endereça o refinamento de requisitos de qualidade em descrições funcionais. No *framework* NFR, esta questão indica as metas, ou seja, como os RNFs são operacionalizados.
- *How much*: essa questão indica os impactos da aplicação das metas (*how*) com o artefato. No *framework* NFR, o *how much* representa os relacionamentos de contribuição entre as metas e softmetas.

Portanto, a linguagem de organização Q7 foi escolhida por proporcionar uma forma clara de como informações importantes de requisitos podem ser classificadas, proporcionando o auxílio ao reuso das mesmas. No entanto, para o escopo desta dissertação, foram realizadas alterações no significado de alguns de seus parâmetros com o intuito de atender completamente as necessidades do processo. Essas modificações são descritas na Seção 3.2 desta dissertação.

2.4 Considerações Finais do Capítulo

Este capítulo apresentou alguns conceitos relacionados ao estudo da ER e ao processo de reuso de requisitos. Vê-se a importância da ER para o desenvolvimento de software de sucesso, e as atividades realizadas para a sua realização.

Dessa forma, ficou clara a relevância do reuso de requisitos, seus benefícios e suas restrições, bem como a forma que alguns autores o desenvolvem. Por fim, foram apresentados o *framework* NFR e a linguagem Q7, os quais são utilizados no processo de reuso de requisitos definido nesta dissertação.

Capítulo 3

Cognitio: Um Processo para Reuso de Requisitos

Este Capítulo apresenta o processo de reuso de requisitos *Cognitio*. Embasado pela utilização de conceitos do *framework* NFR, da linguagem Q7 e de alguns trabalhos relacionados.

Com a aplicação deste processo, estima-se o aumento da qualidade de software, uma vez que os requisitos utilizados no processo, os quais são representados por grafos SIGs, são previamente validados e verificados, com o intuito de estabelecer o processo como apoio à elicitação de requisitos. Além disso, é provido:

i) **O desenvolvimento de software para o reuso**: tendo em vista que ao definir que a qualidade dos requisitos modelados deve ser prioridade neste processo, todo e qualquer grafo modelado e armazenado pode ser reusado; e

ii) **O desenvolvimento de software com o reuso**: uma vez que todo grafo pode ser recuperado, cabe ao engenheiro de requisitos e demais envolvidos, realizar validação dos requisitos, para saber se

os grafos disponíveis e recuperados satisfazem ao escopo do novo sistema.

Para compor o processo *Cognitio*, existem alguns sub-processos os quais têm o intuito de atingir um objetivo comum que é fornecer o reuso de requisitos, buscando a qualidade dos requisitos e o incentivo à prática do reuso. Para um melhor entendimento, a Figura 3.1 apresenta a abordagem na notação BPMN (*Business Process Modeling Notation*) (BPMN, 2009).

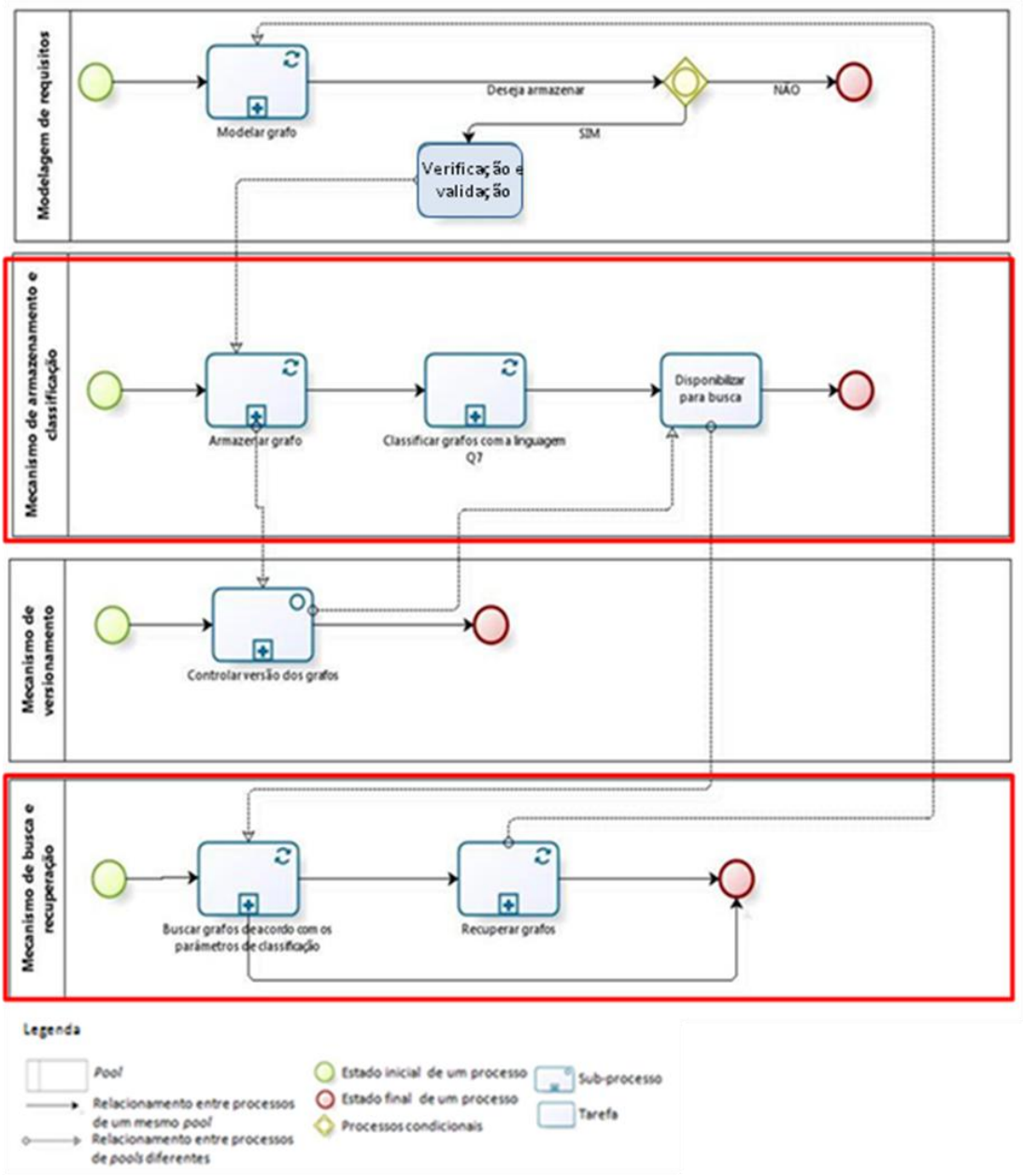

Figura 3.1: Representação do Processo de reuso de requisitos

O processo é dividido em quatro *pools* distintos, denominados **Modelagem de requisitos**, **Mecanismo de armazenamento e classificação, Mecanismo de busca e recuperação**, e **Mecanismo de versionamento**, de forma que cada *pool* é responsável pelo desenvolvimento de seus respectivos sub-processos.

Para melhor explicar o processo de reuso de requisitos, tem-se a premissa que ele está sendo realizado pela primeira vez. Assim, o sub-processo inicial é **Modelar grafo**, realizado pelos *stakeholders*, no *pool* **Modelagem de requisitos**. Durante a modelagem, devem ser realizadas também as atividades de verificação e validação dos requisitos.

Após a modelagem, o mecanismo **Armazenar grafos** pode ser iniciado, uma vez que o usuário desejar guardar os requisitos modelados, gerando um repositório de dados.

Quando um grafo é armazenado, ocorre o sub-processo **Classificar grafos com a linguagem Q7**. Dessa forma, esse mecanismo reconhece as informações importantes do modelo de requisitos, e os classificam de acordo com os parâmetros da linguagem Q7.

Paralelamente ao sub-processo de classificação, ocorre o sub-processo **Controlar versão dos grafos**. Após ocorrer o controle de versão, os grafos versionados e classificados são disponibilizados para a busca.

Neste momento, o sub-processo **Buscar grafos de acordo com os parâmetros de classificação** pode ser iniciado, a partir do **Mecanismo de busca e recuperação**. Ao buscar um grafo, o usuário tem as opções de recuperar ou não os grafos resultantes da busca. Caso ele não deseje recuperar o grafo, o sub-processo de busca é finalizado, caso contrário, o sub-processo **Recuperar grafos** é iniciado.

Ao recuperar um grafo, o usuário pode editá-lo ou não. Caso ele não deseje editar o grafo, o processo de recuperação é finalizado. Caso contrário, ele realiza as edições que achar necessárias. Verifica-se que a edição dos grafos não

é definida no *Cognitio*, cabendo a realização da mesma aos *stakeholders*. Com a edição dos grafos, o usuário realiza uma nova modelagem dos mesmos, reiniciando, portando, o processo de reuso de requisitos.

Para explicar cada sub-processo do *Cognitio*, são focados apenas os sub-processos apresentados nos *pools* **Mecanismo de armazenamento e classificação**, **Mecanismo de versionamento e Mecanismo de busca e recuperação**, haja vista que a forma como os *stakeholders* modelam os grafos no *pool* **Modelagem de requisitos** e as edições realizadas não são definidas pelo processo de requisitos, embora seja enfatizado que todo grafo deve ser validado e verificado, durante do processo de ER.

3.1 Armazenamento dos Grafos

Uma vez que o engenheiro de requisitos modela, verifica e valida um grafo SIG, este pode ser armazenado no repositório. A Figura 3.2 apresenta o sub-processo armazenar grafos.

Figura 3.2: Sub-processo Armazenar grafo no repositório

Ao passo que os dados são armazenados no repositório, o mecanismo de armazenamento deve reconhecer cada elemento que compõe o grafo, validá-lo e armazená-lo, de forma que os mecanismos de classificação e versionamento sejam ativados.

Após realizar o armazenamento das informações desejadas, são iniciados de forma paralela, os mecanismos de classificação e versionamento, os quais são apresentados nas Seções 3.2 e 3.3, respectivamente.

3.2 Classificação dos Grafos

A classificação dos grafos é iniciada logo que um grafo é inserido no repositório, através do mecanismo de classificação. Para tanto, o mecanismo reconhece os dados inseridos no repositório, de acordo com a notação do *framework* NFR. Através da classificação, procura-se melhorar o desempenho do processo de busca dos grafos, uma vez que as informações retornadas aos usuários estarão dentro de um parâmetro pré-estabelecido.

Além disso, com a classificação das informações do *framework* NFR, de acordo com as características de cada uma, é possibilitado o melhor entendimento dos usuários em relação ao grafo.

Para o *Cognitio*, os grafos SIGs são classificados de acordo com os conceitos especificados na linguagem de organização de qualidades Q7 (LEITE et al, 2005), apresentados na Seção 2.3.2, tendo em vista que esta linguagem define parâmetros importantes para a atividade de reuso de requisitos, e pode ser adaptada para abranger outros modelos de representação de requisitos.

O exemplo da Figura 3.3, que representa a softmeta Segurança e suas respectivas metas, serve como apoio para a explicação de cada parâmetro de classificação, apresentados da Seção 3.2.1 a 3.2.7.

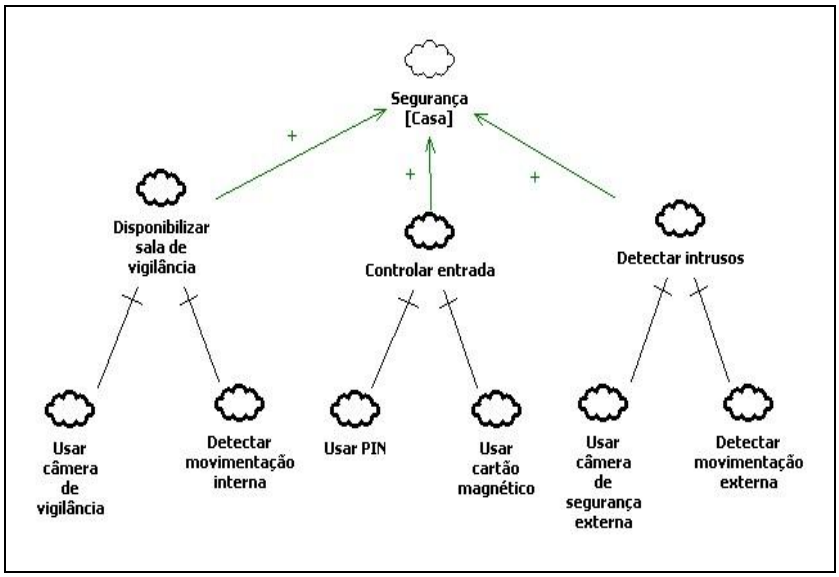

Figura 3.3: Modelagem da softmeta Segurança e suas operacionalizações no *framework* NFR

3.2.1 Parâmetro *Why*

De acordo com Leite et al (2005), este parâmetro consiste na visualização de uma softmeta pai, dentro do modelo NFR. Dessa forma, dado um requisito modelado, quando se aplica o *why* se busca saber por que este requisito foi modelado, ou por que ele deve ser provido.

Neste processo de reuso de requisitos, este parâmetro serve tanto para classificar softmetas quanto as metas pais de um determinado relacionamento. Desta forma, a partir do classificador *why*, é provido o reuso não apenas de softmetas, mas de todo e qualquer elemento que esteja modelado como pai de um relacionamento. Um elemento pai é aquele que, em um dado relacionamento, está recebendo a contribuição ou decomposição; por exemplo, na Figura 3.3, a softmeta Segurança é pai do relacionamento existente entre Segurança e Controlar entrada.

Para demonstrar a aplicação desse parâmetro, há o seguinte exemplo: se em uma modelagem existem as metas "O usuário deve usar PIN" e "O usuário deve usar cartão magnético", dessa forma, pode-se ter a seguinte pergunta do

tipo *why*: "Por que o usuário deve usar PIN e usar cartão magnético?". E como resposta: "Para prover a softmeta Segurança". A Figura 3.4 apresenta os elementos classificados com o parâmetro *why*, de acordo com o exemplo apresentado na Figura 3.3.

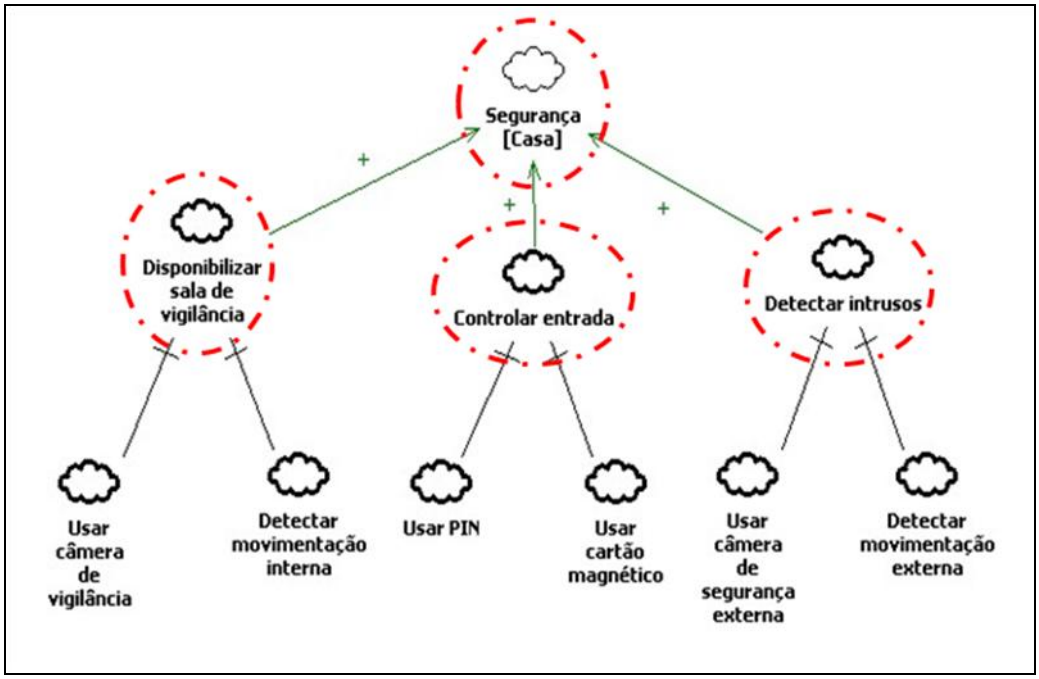

Figura 3.4: Elementos classificados pelo parâmetro *why*

Portanto, a principal contribuição de haver a classificação dos requisitos neste contexto é especificar a que elemento outro requisito está ligado, de forma que este elemento seja seu pai. Logo, todo e qualquer elemento, dentro de um grafo SIG, deve ser encontrado a partir de um elemento que esteja classificado através do parâmetro *why*.

Assim, no exemplo apresentado, os elementos pai são **segurança**, **disponibilizar sala de vigilância**, **controlar entrada** e **detectar intruso,** indicados pelo contorno vermelho e tracejado. Logo, um elemento que não possui filhos não é inserido no parâmetro *why*.

3.2.2 Parâmetro *Who*

O parâmetro *who*, no contexto desta dissertação, é utilizado para especificar o papel do ator que solicitou ou determinou cada requisito modelado. Embora não exista este parâmetro no *framework* NFR, este dado pode ser especificado, através da inserção desta informação, durante a modelagem dos requisitos.

Assim, para o processo de reuso, este parâmetro funciona como uma pré-rastreabilidade, no qual é definido o papel exercido por quem solicitou um determinado requisito dentro de um domínio, de forma que para cada requisito pode-se ter um *who* específico a ele vinculado.

Um exemplo de aplicação deste parâmetro é com a seguinte questão "Que papel exerce o *stakeholder* que solicitou o requisito Segurança?". Como exemplo de resposta, pode-se ter "O proprietário da residência" ou "O analista de segurança da empresa de seguro residencial".

Este parâmetro de classificação é importante para o processo de reuso uma vez que, ao passo que o engenheiro de requisitos conhece o papel de quem solicitou um dado requisito, fica possível, por exemplo, saber em que contexto ou quais necessidades um dado usuário possui, em relação àquele requisito.

3.2.3 Parâmetro *What*

De acordo com Leite et al (2005), o parâmetro *what* indica os tópicos dentro do *framework* NFR. Os tópicos são importantes para contextualização das metas e softmetas que o possuem. No exemplo da Figura 3.3, existe o tópico **[Casa]**, o qual está contextualizando a softmeta **Segurança**. Isto é, caso um usuário encontre este requisito no repositório, ficará ciente que o mesmo está inserido no contexto de segurança de residências.

3.2.4 Parâmetro *Where*

O parâmetro *where*, no escopo deste trabalho, especifica em que contexto um grafo está inserido. Assim, dado um requisito inserido no repositório, pertencente a um determinado grafo, o usuário pode facilmente entender qual a aplicação do mesmo.

Desta forma, enquanto o parâmetro *what* indica o contexto de cada elemento do grafo, de forma isolada, o parâmetro *where* especifica o contexto geral de todo o grafo. O objetivo deste parâmetro é proporcionar aos usuários um melhor entendimento da aplicabilidade do grafo e de seus requisitos em relação ao domínio no qual se encontram.

3.2.5 Parâmetro *When*

O parâmetro *when*, de acordo com Leite et al (2005), indica o *claim*, dentro de um modelo NFR. Para o escopo desta dissertação, é utilizada esta mesma definição. No exemplo da Figura 3.3 não há este elemento modelado, mas o mesmo é apresentado no Capítulo 2, na Seção 2.3.1.

O interessante desse parâmetro é permitir a verificação, dentro de um grafo, de quais relacionamentos possuem pré-condições ou observações para a sua implementação. Assim, essas observações podem explicar aos *stakeholders*, por exemplo, por que um dado relacionamento deve existir naquela modelagem.

3.2.6 Parâmetro *How*

Este parâmetro de classificação indica o caminho inverso do parâmetro *why*. Diferentemente de como Leite et al (2005) definiram, a partir deste parâmetro são classificadas as metas filhas de um relacionamento. A relevância desta

classificação é de se conhecer as dependências existentes entre os requisitos modelados.

Com a aplicação deste parâmetro, pode-se haver a seguinte pergunta: "Como uma determinada meta ou softmeta pai é alcançada?" ou, de acordo com o exemplo da Figura 3.3, há "Como a meta **controlar entrada** é atingida?". E como exemplo de resposta: "A partir das metas filhas **Usar PIN** e **Usar cartão magnético**".

Este parâmetro é essencial para se verificar quais são os requisitos que estão diretamente ligados a uma softmeta ou meta pai, independente do tipo de relacionamento. Assim, esse atributo é determinado para que não apenas o requisito buscado no repositório seja apresentado ao usuário, mas toda a ramificação dos requisitos necessários para a definição de uma dada softmeta ou meta pai. A Figura 3.5 apresenta os elementos da Figura 3.3 classificados neste parâmetro.

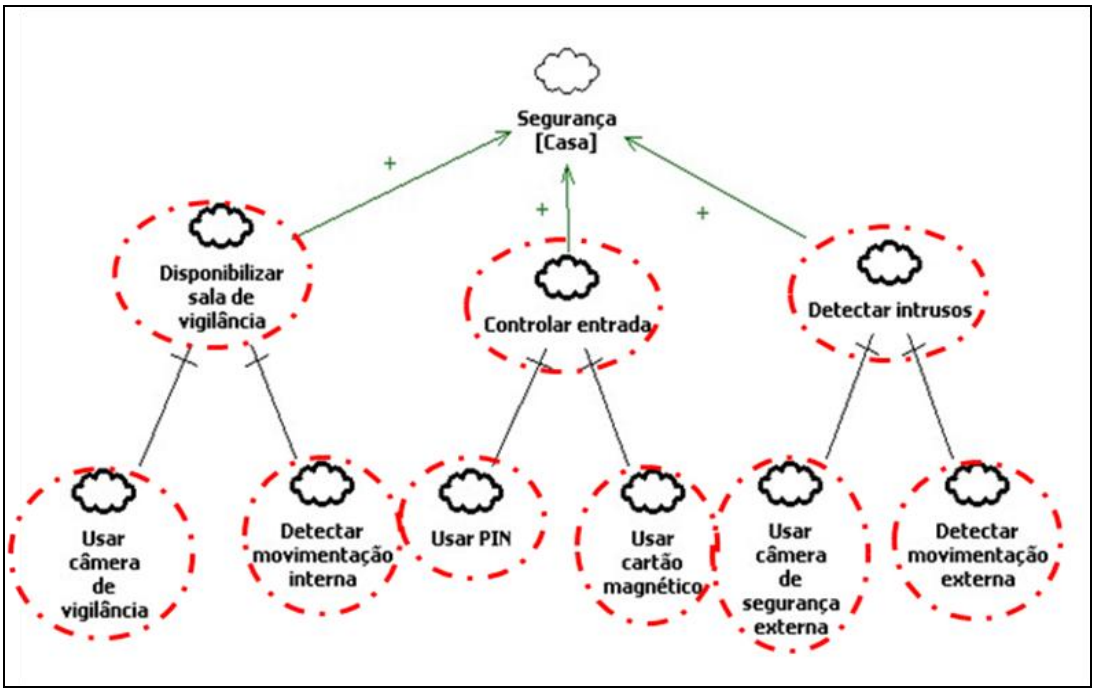

Figura 3.5: Elementos classificados pelo parâmetro *how*

Dessa forma, as metas classificadas como *how* são: **disponibilizar sala de vigilância**, **controlar entrada**, **detectar intrusos**, **usar câmera de vigilância**, **detectar movimentação interna**, **usar PIN**, **usar cartão magnético**, **usar câmera de segurança externa** e **detectar movimentação externa,** indicadas pelo contorno tracejado.

3.2.7 Parâmetro *How Much*

Assim como é especificado por Leite et al, 2005, o parâmetro *how much* é usado para indicar os impactos que algumas softmetas e/ou metas exercem sobre as outras, em relação às contribuições ou decomposições existentes entre elas.

Para esta dissertação, o *how much* indica as contribuições e as decomposições na modelagem. Dessa forma, quando uma decomposição do tipo "E", como acontece entre **usar PIN** e **usar cartão magnético**, para atingir a meta **controlar entrada**, isso quer dizer que, caso um usuário queira reusar o requisito **controlar entrada** dessa modelagem, sem alterações, ele terá que reusar também suas duas metas filhas. Caso exista na modelagem uma decomposição do tipo "OU", ao reusar uma meta ou softmeta, basta que pelo menos um dos seus elementos filhos seja reusado.

3.3 Atribuição e Controle de Versão dos Grafos

A fim de haver tanto o controle de versões quanto a diminuição de quantidade excessiva de informações repetidas no repositório de requisitos, é definido o sub-processo **Controlar versão dos grafos**.

Controle de versão é entendido como a atividade de gerenciar alterações em informações. Assim, ele consiste em um mecanismo capaz de armazenar tanto o conteúdo de arquivos (neste caso, grafos), quanto o histórico de alterações realizadas nesses ativos.

Portanto, essa atividade permite que cada *stakeholder* possa manipular a informação que lhe for conveniente, sem que o conteúdo original seja comprometido, de maneira que a alteração gerada seja armazenada com uma nova versão diferente da versão das informações já existentes. A Figura 3.6 apresenta o sub-processo **Controlar versão dos grafos**.

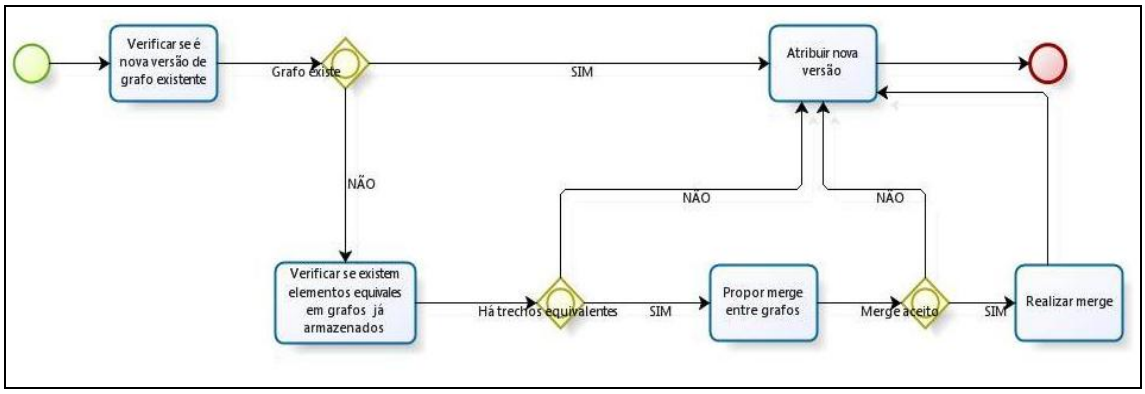

Figura 3.6: Sub-processo Controlar versão dos grafos

De acordo com a Figura 3.6, o processo de controle de versão é iniciado verificando-se se o grafo é referente a uma nova versão de grafos já contidos no repositório. Caso seja um grafo editado, isto é, uma nova versão de um grafo já existente no repositório, o mecanismo de versionamento realiza a revisão e logo após atribui nova versão ao grafo.

Caso o grafo seja novo, o mecanismo de versionamento visualiza se partes daquele grafo já existem no repositório. Caso existam, a realização do *merge* entre os grafos cujos elementos são coincidentes é proposta, cabendo ao usuário aceitar, ou não, realizar o merge. Após isso, uma nova versão é atribuída ao grafo.

Esses processos desempenhados pelo mecanismo de versionamento são de suma importância para o *Cognitio* por três motivos principais:

i) Evita a acumulação de um mesmo grafo em várias versões dentro do repositório, uma vez que apenas a última versão é guardada no mesmo;

ii) Proporciona o merge entre dois ou mais grafos que possuem elementos iguais;

iii) Aumenta o desempenho do repositório, uma vez que grafos repetidos ou com partes equivalentes deixam de existir no repositório, ou por meio do *merge* ou por terem sido sobrescritos, diminuindo o tempo gasto durante a atividade de busca e recuperação.

Após a realização do armazenamento, classificação e do controle e versão dos grafos, o sub-processo de busca dos grafos pode ser iniciado. A Seção 3.4 a seguir apresenta esse sub-processo.

3.4 Busca dos Grafos

A realização da busca dos grafos, no *Cognitio*, ocorre através dos parâmetros de classificação apresentados na Seção 3.2. O processo de busca é feito por meio do mecanismo de busca.

Para buscar um ativo no repositório de requisitos, o mecanismo de busca recebe os parâmetros inseridos pelo usuário, identifica os elementos que atendem aos parâmetros inseridos, organiza os resultados de acordo com a classificação da linguagem Q7 e os apresenta aos usuários, disponibilizando as opções de visualização e recuperação dos ativos. A Figura 3.7 apresenta esse processo.

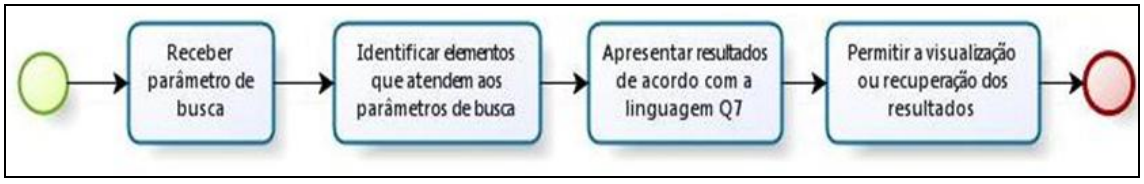

Figura 3. 7: Sub-processo Buscar grafos

Caso o usuário não encontre os grafos desejados, pode realizar novamente a inserção de novos parâmetros para a busca. Para melhor entendimento do processo de busca de grafos no repositório, observe o exemplo a seguir.

Imagine que um usuário deseja encontrar um requisito, realizando buscas através dos parâmetros *how* e *where*, com os termos **câmera** e **segurança**, respectivamente. Dessa forma, o sistema faz uma varredura apenas nos termos existentes no repositório que estão classificados nestes parâmetros. Logo, de acordo com o exemplo da Figura 3.3, que apresenta o grafo Segurança de casas, ao se realizar a busca descrita acima, como a palavra **câmera** está contida na sentença de um elemento filho das metas **disponibilizar sala de vigilância** e **detectar intrusos**, e como essas metas estão inseridas no contexto de **segurança**, ambas devem ser retornadas ao usuário, como requisitos encontrados.

Após realizar a busca, o usuário pode recuperar os requisitos encontrados. O processo de recuperação de grafos, o qual é apresentado na Seção 3.5, inicia-se com o mecanismo de recuperação.

3.5 Recuperação dos Grafos

Após ocorrer a busca dos grafos no repositório, a recuperação destes pode ser feita. Dessa forma, quando um grafo é encontrado e, de fato, o usuário pretende reutilizá-lo, o mecanismo de recuperação deve fornecer o requisito encontrado.

Sabe-se que quando uma meta ou softmeta é modelada em um grafo SIG, a mesma pode conter softmetas ou metas pais, ou softmetas ou metas filhas. Dessa forma, durante a recuperação dos ativos, os *stakeholders* podem recuperar apenas o requisito buscado, ou o requisito e seus respectivos filhos e pais, ou todo o grafo que contém o requisito localizado.

Portanto, ao escolher a opção desejada, o usuário realiza um *check-out*, isto é, a recuperação do ativo existente no repositório que foi escolhido, para o

seu ambiente local. Através da realização do *check-out*, o *stakeholder* recebe não só a sentença do requisito mas também as informações a ele atreladas de acordo com os parâmetros de classificação.

No contexto deste processo, verificou-se que outros ativos podem ser fornecidos de maneira a complementar as informações reusadas. Portanto, ao encontrar o ativo desejado, o usuário pode também:

- Receber um diagrama gráfico do grafo que contém o elemento selecionado para recuperação, mesmo que o usuário escolha recuperar apenas uma parte dele, visando a edição do grafo e o controle de versão do mesmo;
- Receber um relatório textual contendo informações em relação aos dados recuperados, o qual é organizado de acordo com os parâmetros da linguagem Q7;
- Receber o componente de código que implementa o requisito recuperado, caso este tenha sido registrado, independente da linguagem utilizada para tanto, de forma a servir como exemplo para auxiliar no entendimento para futura implementação do requisito.

  Logo, cada requisito deve possuir um identificador único. Assim, quando um componente é armazenado, o mesmo deve ser vinculado ao seu requisito através deste identificador. Caso o requisito seja atualizado, deve-se verificar se o componente de código continua vinculado à nova versão do requisito ou não.

Com essas informações, além do *stakeholder* obter uma noção clara das características daquele requisito, ele ainda pode visualizar graficamente como o mesmo foi estruturado, e como desenvolvedores o implementaram em sua aplicação.

Com a obtenção destas informações e com o seu posterior uso em outras aplicações, ocorre o desenvolvimento de software **com reuso**. A Figura 3.8 apresenta o processo de recuperação.

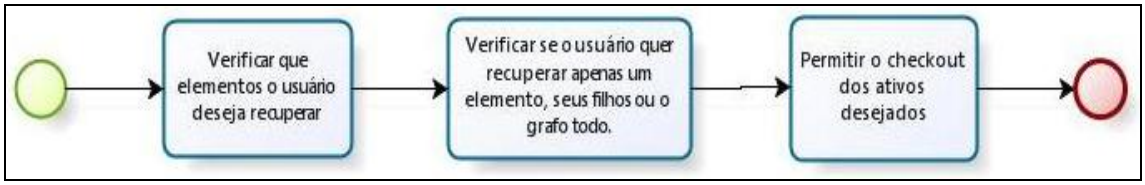

Figura 3.8: Sub-processo de recuperação de grafos

No entanto, ao recuperar um determinado grafo, o *stakeholder* pode perceber que para a sua aplicação aquela especificação não é interessante ou suficiente. Assim, ele pode manipulá-lo da maneira que lhe convier, localmente em seu ambiente de trabalho, realizando edições no mesmo, tais como: adição ou remoção de elementos no grafo, união entre grafos, entre outros. Com a edição dos grafos, o usuário retorna ao primeiro passo do processo de reuso (modelar grafos) reiniciando-o.

3.6 Considerações Finais do Capítulo

Este Capítulo apresentou o processo *Cognitio* proposto nesta dissertação. O mesmo deve ocorrer durante a ER na atividade de elicitação de requisitos, de forma que o repositório represente uma fonte de informação, no qual os usuários encontram os requisitos necessários para a sua aplicação. Além disso, os usuários podem contribuir para que novos requisitos formem o repositório, por meio da inserção dos mesmos.

Assim, através da busca e recuperação dos requisitos, o usuário está desenvolvendo software com o reuso, e através da modificação de um determinado requisito e inserção de novos requisitos no repositório, o usuário está desenvolvendo software para o reuso.

Considerando, ainda, que o desenvolvimento de um mecanismo automatizado é indispensável para o processo de reuso, o que assegura maior confiabilidade e menores custos de tempo para a realização das fases deste

processo, o Capítulo 4 apresenta a ferramenta *Cognitio Tool*, o qual apresenta estratégias para servirem de apoio ao processo de reuso de requisitos aqui definido.

Capítulo 4

Ferramenta *Cognitio Tool*

Tendo em vista a necessidade de se automatizar o processo de reuso de requisitos, a fim de estimular a prática desta atividade, tornando-a acessível, desenvolveu-se a ferramenta *Cognitio Tool*. A mesma consiste em apresentar estratégias para a realização das atividades do processo de reuso de requisitos.

Este Capítulo apresenta como estas atividades são realizadas de forma automatizada a partir da ferramenta, embora alguns módulos não estejam implementados, tais como, o módulo de controle de versão e de recuperação.

Para a realização de suas atribuições, a *Cognitio Tool* foi desenvolvida na linguagem PHP (*Hypertext Preprocessor*) e com o sistema de gerenciamento de banco de dados (SGBD) MySQL. Nas Seções a seguir, são apresentadas a arquitetura, as funcionalidades e as ferramentas utilizadas para a realização das atividades da ferramenta.

4.1 Arquitetura da Ferramenta *Cognitio Tool*

Tendo em vista a busca pelo incentivo à prática do reuso de requisitos, a arquitetura da *Cognitio Tool* foi desenvolvida com foco na sua abrangência a um maior número de usuários. Assim, a mesma foi especificada de forma que suas funcionalidades possam ser executadas em um navegador de Internet, disponibilizando-as na rede e permitindo que diferentes interessados as utilizem.

Dessa, a arquitetura da ferramenta *Cognitio Tool* está estruturada como apresenta a Figura 4.1.

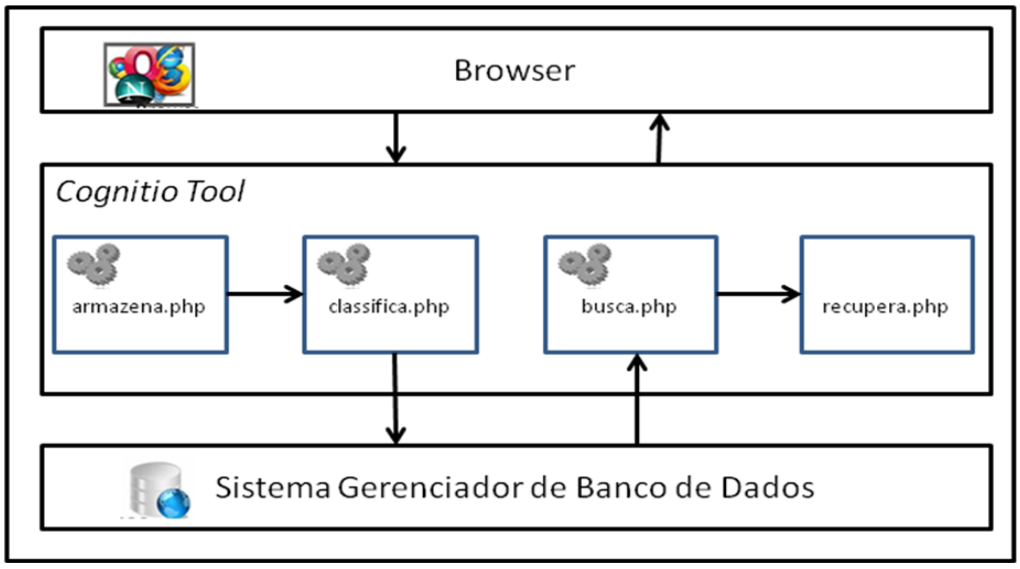

Figura 4.1: Arquitetura da ferramenta *Cognitio Tool*

Dessa forma, a arquitetura da ferramenta conta com a interface com o usuário, a qual é realizada através da comunicação entre o *browser* de acesso escolhido pelo usuário e os módulos de entrada e saída. O módulo de entrada é formado pelos módulos armazena.php e classifica.php. O repositório de requisitos contém todos os requisitos inseridos nos módulos de entrada, de forma que os mesmos são gerenciados através de um SGBD. Por último, há o módulo de saída da ferramenta, o qual é composto pelos módulos busca.php e recupera.php. Na Figura 4.1, os módulos representados por uma imagem de

uma engrenagem são os módulos já implementados, enquanto que os que não possuem a imagem de uma engrenagem, ainda não estão contemplados de forma prática na *Cognitio Tool*.

Além dos módulos apresentados na Figura 4.1, a *Cognitio Tool* possui ainda o módulo com definição de um controle de acesso (ainda não implementado) e um glossário.

Para melhor entendimento da utilização da ferramenta *Cognitio Tool*, o grafo Segurança de contas foi utilizado para representar como as atividades do processo *Cognitio* são realizadas na mesma. Este grafo apresenta a modelagem das softmetas segurança, usabilidade e performance de um sistema de contas de usuário (CHUNG et al, 2000).

A escolha desta modelagem deu-se pelo fato da mesma apresentar requisitos não funcionais tais como segurança, performance e usabilidade, os quais são bem definidos na literatura. Desta forma, tanto essas três softmetas principais, quanto todas as interdependências dessas softmetas foram modeladas no *framework* NFR, com auxílio da ferramenta StarUML como ilustrada a Figura 4.2.

Este exemplo possui todos os atributos do *framework* NFR, facilitando a apresentação do uso da ferramenta *Cognitio Tool*. Por exemplo, veja na Figura 4.2, a softmeta **performance**, a meta **usar indexação**, o *claim* **otimizar a validação**, a decomposição entre **integridade**, **plenitude** e **precisão**, e a contribuição entre **usabilidade** e **pedir ID adicional**.

Logo, todos os parâmetros de classificação da linguagem Q7 são apresentados nesse exemplo, como se observa nas Seções a seguir, durante a apresentação dos módulos da *Cognitio Tool*.

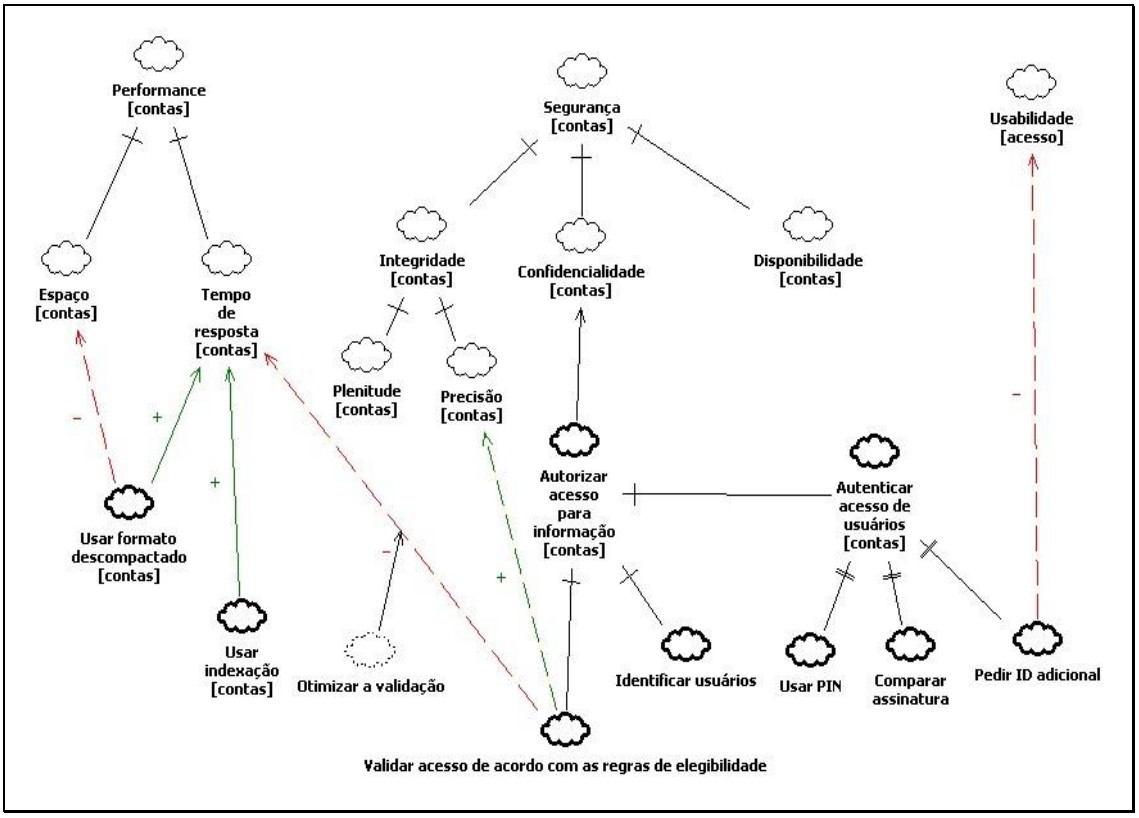

Figura 4.2: Modelagem do grafo Segurança de contas (CHUNG et al, 2000)

4.1.1 Módulo de Armazenamento

O módulo de armazenamento consiste no oferecimento de uma interface amigável ao usuário, de forma que seja facilitada a inserção dos grafos SIG no repositório. Este módulo denomina-se **armazena.php** e seu acesso é feito através do navegador de Internet. A interface do módulo é mostrada na Figura 4.3.

Embora seja interessante a forma de disponibilização dos grafos armazenados, isto é, se pública ou privada, a ferramenta *Cognitio Tool* ainda não provê esse mecanismo, de forma que todos os grafos inseridos no repositório são disponibilizados para todo e qualquer usuário, sem haver qualquer tipo de restrição.

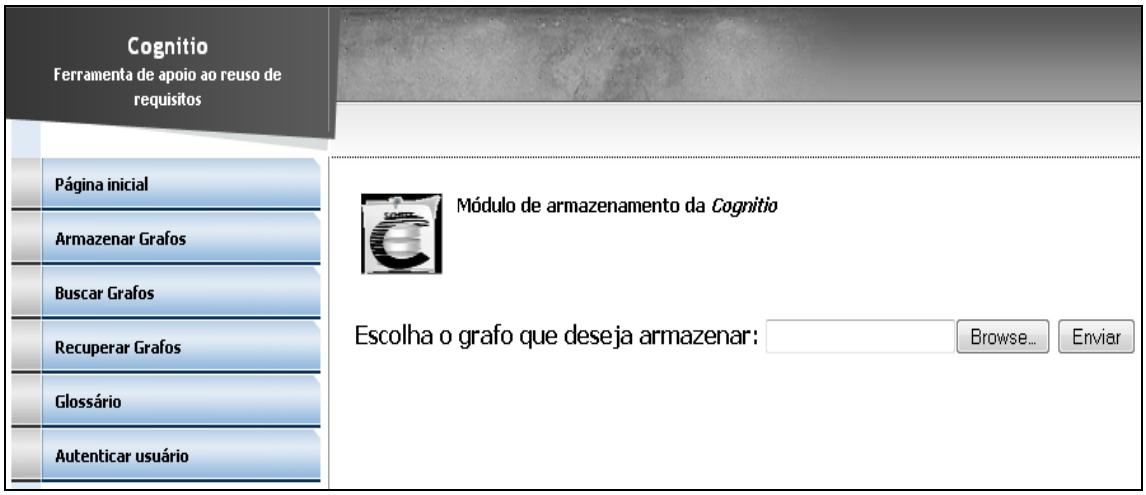

Figura 4. 3: Módulo de armazenamento de grafos

Caso estivesse implementada, escolhendo a forma de armazenamento e disponibilização dos grafos, o *stakeholder* teria a liberdade de determinar se seus requisitos seriam acessados de forma pública ou privada.

Dessa forma, se os *stakeholders* disponibilizassem os grafos gerados para diversos usuários, de forma que qualquer um que tenha acesso ao repositório pode reusá-lo, o estado do grafo seria definido como público.

Por outro lado, podem haver casos em que os *stakeholders* não disponibilizassem seus grafos para todos os usuários que têm acesso ao repositório de requisitos. Neste caso, o estado de armazenamento do grafo seria definido como privado.

Com a modelagem do grafo Segurança de contas através da ferramenta StarUML, é feita a exportação de um arquivo no formato XML, o qual é usado como entrada para a ferramenta *Cognitio Tool*.

O Quadro 4.1 exibe alguns trechos do XML da modelagem do grafo Segurança de contas. Para melhor entendimento, os atributos do XML que são reconhecidos pela *Cognitio Tool* estão realçados na cor cinza, haja vista que os demais não trazem informações importantes para o repositório. No lado direito do Quadro 4.1 é apresentado o que cada *tag* representa, informando em que parâmetro de classificação do Q7 o elemento modelado se insere.

```
<UML:Package xmi.id='UMLPackage.31' name='Sistema de
 segurança de acesso'  visibility='public' isSpecification='false'
 namespace='UMLPackage.30'  isRoot='false' isLeaf='false'
 isAbstract='false' isActive='false' >

 <UML:Class xmi.id='UMLClass.32' name='Autorizar acesso
  para  informações' visibility='public'  isSpecification='false'
  namespace='UMLPackage.31'
  clientDependency='UMLDependency.13 UMLDependency.25'
  supplierDependency='UMLDependency.3  UMLDependency.4
  UMLDependency.5' isRoot='false'  isLeaf='false'
  isAbstract='false' isActive='false' >

  <UML:Classifier.feature>
    <UML:Attribute xmi.id='UMLAttribute.33' name='Conta'
     visibility='public' isSpecification='false'
     ownerScope='instance' changeability='changeable'
     targetScope='instance'  type='' owner='UMLClass.32' />
  </UML:Classifier.feature>

  <UML:ModelElement.templateParameter>
    <UML:TemplateParameter
     xmi.id='UMLTemplateParameter.34'
     name='Gerente de segurança'
     modelElement='UMLClass.32' modelElement2='X.62' />
  </UML:ModelElement.templateParameter>

 </UML:Class>

      ...

  <UML:Stereotype xmi.id='X.77' name='AND'
    extendElement='UMLDependency.3 UMLDependency.4
    UMLDependency.5 UMLDependency.6 UMLDependency.7
    UMLDependency.8 UMLDependency.9
    UMLDependency.14 UMLDependency.15
    UMLDependency.16' />

      ...

     <UML:Class xmi.id='UMLClass.5' name='Otimizar a
      validação' visibility='public'  isSpecification='false'
      namespace='UMLModel.2'
      clientDependency='UMLDependency.7 isRoot='false'
      isLeaf='false'  isAbstract='false'  isActive='false' />

     …

</UML:Package>
```

Declaração do domínio no qual o grafo está inserido. Refere-se ao parâmetro *where.*

Declaração de uma meta ou softmeta

Parâmetro *how*

Parâmetro *why*

Declaração do contexto de cada elemento do grafo. Refere-se ao parâmetro *what.*

Declaração do papel do usuário que solicitou o requisito. Refere-se ao parâmetro *who*

Declaração dos relacionamentos do tipo E existentes. Refere-se ao parâmetro *how-much*

Declaração dos *claims* existentes. Refere-se ao parâmetro *when*.

Quadro 4.1: Trecho do arquivo XML do Segurança de contas

Como mostra o Quadro 4.1, cada grafo é iniciado pela *tag* **UML:Package**. A *tag* **UML:Class** pode se referir a uma meta, softmeta ou *claim*. A *tag* **UML:Classifier.feature** indica a declaração dos tópicos de uma meta ou softmeta. A *tag* **UML:ModelElement.TemplateParameter** indica o papel de quem solicitou um dado requisito. Por último, a *tag* **UML:Stereotype** apresenta os relacionamentos existentes na modelagem, agrupando-os de acordo com o tipo.

Em relação aos atributos do XML, o **xmi.id** representa o identificador gerado automaticamente para cada elemento modelado. O atributo **name** indica o rótulo ou sentença atribuída a cada elemento do grafo. O atributo **namespace** refere-se à *tag* **UML:Package** à qual o elemento modelado está atrelado. Esses atributos são utilizados para que a definição dos relacionamentos entre os elementos do grafo, bem como os elementos existentes no grafo sejam reconhecidas pela ferramenta *Cognitio Tool*.

Os atributos **clientDependency** e **supplierDependency** referem-se, respectivamente, aos filhos e aos pais de determinados relacionamentos. Os atributos **owner** e **modelElement** indicam a que metas, softmeta ou *claim*, um dado elemento está vinculado, de forma que **owner** está para o parâmetro *what*, assim como o **modelElement** está para o *how*. Por último, o atributo **extendElement** indica os relacionamentos que estão associados a determinado **name** da *tag* **UML:Stereotype**.

Assim, ferramenta *Cognitio Tool* realiza o reconhecimento das *tags* e atributos do arquivo XML, através do módulo de classificação, apresentado a seguir.

4.1.2 Módulo de Classificação

O módulo de classificação dos grafos, o qual se denomina **classifica.php,** consiste em realizar a classificação dos elementos do grafo, de acordo com os parâmetros da linguagem Q7 que foram apresentados no Capítulo 3.

Tabela 4. 1: Classificação do grafo Segurança de contas

Why	**Who**	**What**	**Where**	**When**	**How**	**How Much**
Performance	Gerente de performance	Contas	Segurança de contas de acesso		Espaço	And
Performance	Gerente de performance	Contas	Segurança de contas de acesso		Tempo de resposta	And
Segurança	Gerente de segurança	Contas	Segurança de contas de acesso		Integridade	And
Segurança	Gerente de segurança	Contas	Segurança de contas de acesso		Confidencialidade	And
Segurança	Gerente de segurança	Contas	Segurança de contas de acesso		Disponibilidade	And
Usabilidade	Usuário do sistema	Acesso	Segurança de contas de acesso		Pedir ID adicional	C-Hurt
Espaço	Gerente de performance	Contas	Segurança de contas de acesso		Usar formato descompactado	Hurt
Tempo de resposta	Gerente de performance	Contas	Segurança de contas de acesso		Usar formato descompactado	Help
Tempo de resposta	Gerente de performance	Contas	Segurança de contas de acesso		Usar indexação	Help
Tempo de resposta	Gerente de performance	Contas	Segurança de contas de acesso	Otimizar validação	Validar acesso de acordo com as regras de elegibilidade	C-Hurt
Integridade	Analista de segurança	Contas	Segurança de contas de acesso		Plenitude	And
Integridade	Analista de segurança	Contas	Segurança de contas de acesso		Precisão	And
Confidenciali dade	Analista de segurança	Contas	Segurança de contas de acesso		Autorizar acesso para informação	Satisficing
Precisão	Analista de segurança	Contas	Segurança de contas de acesso		Validar acesso de acordo com as regras de elegibilidade	C-Help
Autorizar acesso para informação	Analista de segurança	Contas	Segurança de contas de acesso		Validar acesso de acordo com as regras de elegibilidade	And
Autorizar acesso para informação	Analista de segurança	Contas	Segurança de contas de acesso		Identificar usuários	And
Autorizar acesso para informação	Analista de segurança	Contas	Segurança de contas de acesso		Autenticar acesso de usuários	And
Autenticar acesso de usuários	Analista de segurança	Contas	Segurança de contas de acesso		Usar PIN	Or
Autenticar acesso de usuários	Analista de segurança	Contas	Segurança de contas de acesso		Comparar assinatura	Or
Autenticar acesso de usuários	Analista de segurança	Contas	Segurança de contas de acesso		Pedir ID adicional	Or

Dessa forma, esse módulo reconhece as *tags* e atributos do arquivo XML e, de acordo com a modelagem de cada elemento, forma as tabelas com os dados classificados. A classificação do grafo Segurança de contas é apresentada na Tabela 4.1 acima.

4.1.3 Módulo de Busca

Uma vez que o repositório de requisitos já possui grafos armazenados, a tarefa de busca pode ser realizada por parte dos *stakeholders* através do mecanismo de busca que é ativado no módulo **busca.php**.

Dessa forma, a busca é realizada através do acesso à tabela Grafo, a qual é mostrada na Tabela 4.1, de acordo com os parâmetros de busca inseridos pelos usuários. Para tanto, o módulo oferece uma interface amigável, onde no lado esquerdo da tela o usuário observa quais são os grafos existentes no repositório, acompanhados de seus domínios, e no lado direito são apresentados os campos para inserção dos termos a serem buscados, como ilustra a Figura 4.4.

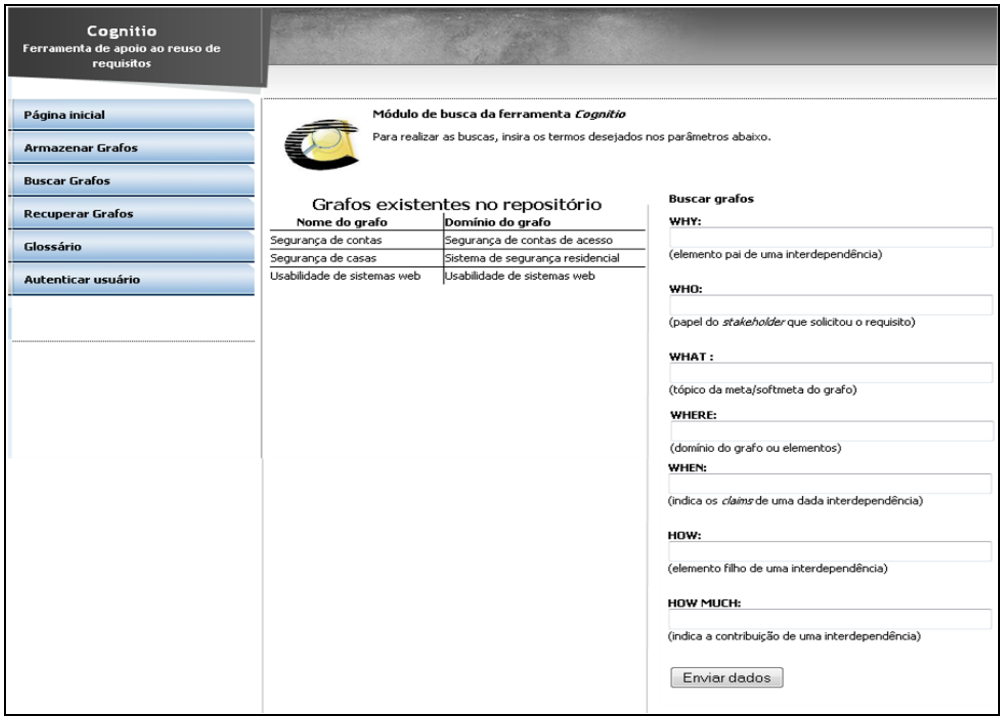

Figura 4.4: Módulo de busca da ferramenta *Cognitio Tool*

A apresentação dos grafos já inseridos, bem como do domínio em que cada um se insere é importante para que os usuários tenham noção do que podem buscar e do que podem encontrar através das buscas. Através da busca, o usuário obtém as informações pertinentes aos termos e parâmetros utilizados

para tanto. Por exemplo, caso ele busque pelo termo **Autenticar usuários**, inserindo-o no parâmetro *why* ele tem o resultado mostrado na Figura 4.5.

Figura 4.5: Resultado da busca pelo termo autenticar usuários a partir do parâmetro why

No entanto, se o mesmo termo for buscado através do parâmetro *how* o resultado retornado é diferente, como apresenta a Figura 4.6.

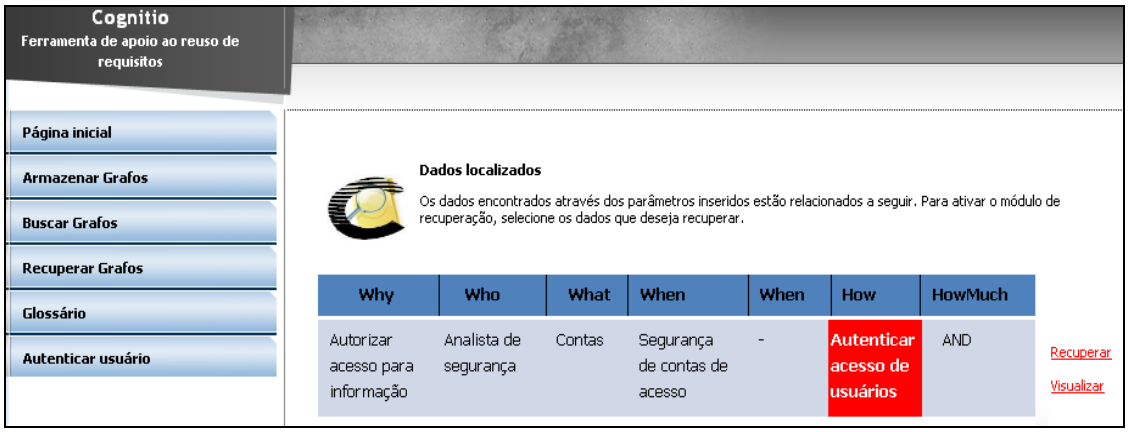

Figura 4.6: Busca pelo termo autenticar usuários a partir do parâmetro how

Quando a busca é realizada, o módulo busca.php apresenta uma nova tela com os dados encontrados, de forma que o usuário pode escolher se deseja recuperar ou visualizar o grafo que contém as informações encontradas. Caso ele escolha visualizar, uma nova tela é apresentada, contendo o grafo localizado, como ilustra a Figura 4.7.

Essa visualização é interessante, pois permite uma visualização gráfica das informações encontradas, auxiliando o entendimento do usuário, em relação aos relacionamentos e demais elementos existentes no grafo.

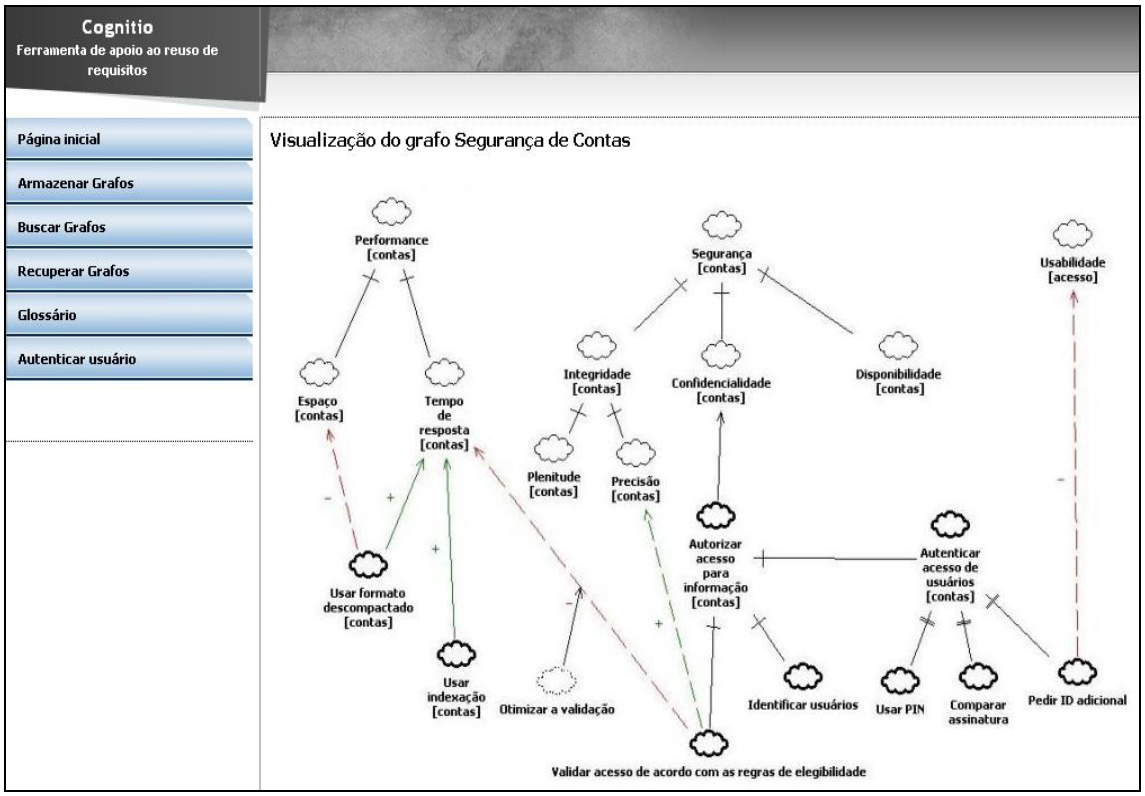

Figura 4.7: Visualização do grafo buscado

Por outro lado, ao escolher a opção recuperar, uma nova tela é apresentada ao usuário, já pertencente ao módulo de recuperação. Este módulo é apresentado na Seção 4.1.4 a seguir.

4.1.4 Módulo de Recuperação

O módulo de recuperação, denominado **recupera.php**, oferece aos usuários a possibilidade de realizar o *check-out* dos dados que desejar, de forma que possam reusá-los da forma que acharem conveniente. Este módulo ainda não está implementado na ferramenta, embora a Figura 4.8 apresente a interface do mesmo.

Uma vez que os usuários escolhem que dados desejam recuperar, a ferramenta deveria prover o *check-out* dos dados nos formatos XML e UML, além dos dados de forma textual e exemplos de implementação dos grafos recuperados.

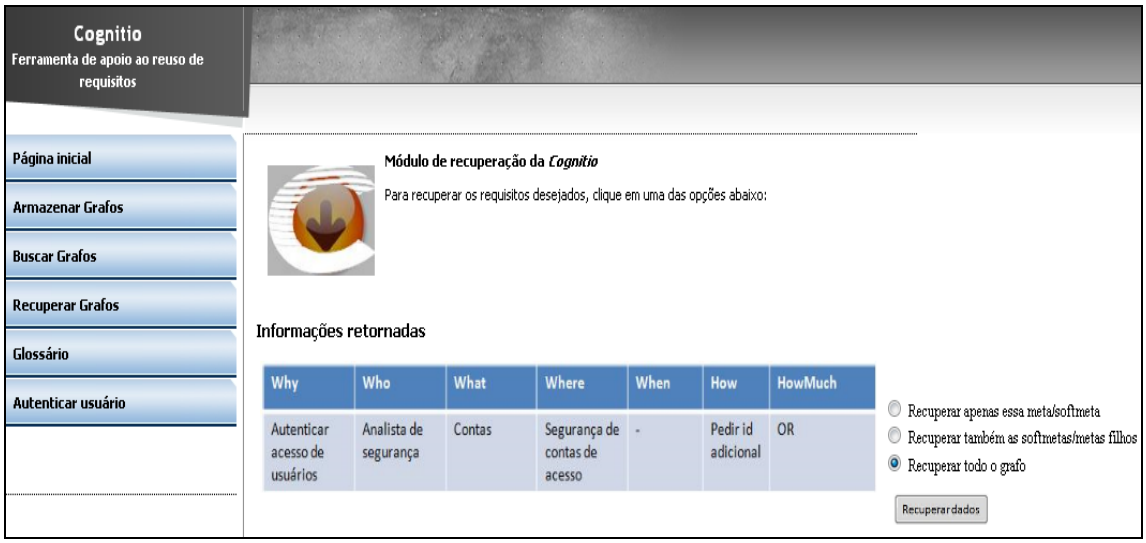

Figura 4.8: Interface do módulo de recuperação

Portanto, este mecanismo é de grande valia, uma vez que quando os *stakeholders* têm acesso a informações de como certo requisito é implementado ou comporta-se no âmbito de um domínio, torna claro se, de fato, aquele requisito é interessante para sua aplicação e se não o for, pode servir de guia para especificação de novos requisitos.

4.1.5 Módulo de Glossário

A ferramenta *Cognitio Tool* possui o módulo de glossário, o qual é denominado glossario.php, para que conceitos do *framework* NFR, da linguagem Q7, e de outros elementos que compõem o processo de reuso de requisitos fiquem claros. Assim, o entendimento e a utilização do processo e da ferramenta são facilitados. A interface deste módulo é exibida na Figura 4.9.

Figura 4.9: Interface do módulo glossário

4.1.6 Módulo de Controle de Acesso

Este módulo consiste em prover ao processo de requisitos o controle de acesso de usuários. O controle ocorre através da inserção do *login* e senha do usuário, de forma que este módulo verifica se aquele usuário já existe e se a sua senha está correta. Caso essas condições sejam atendidas, o módulo checa qual é o tipo de usuário e ativa suas respectivas permissões.

Embora este módulo ainda não esteja implementado, para a utilização da ferramenta *Cognitio Tool* foram definidos dois tipos de usuários de acordo com o seu papel: Engenheiro de requisitos e Pesquisador.

i) **Engenheiro de requisitos**: o engenheiro de requisitos realiza todas as tarefas determinadas no *Cognitio*. Dessa forma, ele tanto participa do processo para prover o desenvolvimento de software com reuso, isto é, realizando reuso de requisitos para o desenvolvimento de novas aplicações, tanto do desenvolvimento de software para o reuso, uma vez que cabe a ele preparar os

requisitos visando à qualidade dos mesmos, tendo em vista o seu posterior uso.

ii) **Pesquisador**: o usuário pesquisador realiza apenas as atividades de busca e recuperação dos requisitos. Logo, este participa apenas do processo de desenvolvimento de software com reuso, uma vez que é entendido que este grupo é composto por usuários dos mais variados níveis de conhecimento em relação à ER, de forma que estes nem sempre têm domínio o suficiente para preparar requisitos para o seu posterior reuso.

A Figura 4.10 apresenta a interface do módulo de controle de acesso da ferramenta *Cognitio Tool*.

Figura 4.10: Interface do módulo de controle de acesso

4.1.7 Módulo de Repositório de Requisitos

Quando um grafo é inserido, o SGBD cria quatro tabelas, as quais são denominadas **Arquivo**, **ElementosGrafo**, **Interdependências** e **Grafo**.

Na tabela **Arquivo**, existe a chave primária denominada IdArquivo a qual serve de referência para as outras tabelas. Cada grafo é vinculado a um

único arquivo, de forma que caso nomes ou ids de grafos se repitam, deve haver a comparação com o id do arquivo, para que conflitos sejam reparados. Há o atributo NomeDoArquivo, o qual corresponde ao nome do grafo armazenado, e o atributo DataDeCriação, o qual foi definido a fim de informar ao usuário quando um determinado grafo foi inserido no repositório.

A tabela **ElementosGrafo** tem como chaves primárias o IdElementosGrafo, a qual é vinculada aos elementos classificados pelo parâmetro *why*. Como chave estrangeira há o IdArquivo, uma vez que cada elemento deve estar relacionado a apenas um arquivo. Como atributos desta tabela existem os parâmetros *who*, *what*, domínio, *claim* e versão, além do *why* que já funciona como chave primária.

A tabela **Interdependências** é composta pela chave primária IdInterdependencia, que é único para cada interdependência ocorrida no grafo. Além disso, nessa tabela há os atributos *why*, *how* e *howMuch*.

A tabela **Grafo** possui como chaves estrangeiras o IdArquivo, IdElementosGrafo e o IdInterdependencia. Essa tabela é a principal do repositório, uma vez que consiste na junção de todos os dados referentes ao grafo armazenado.

A Figura 4.11 apresenta o modelo entidade-relacionamento (MER) lógico das tabelas, de forma que: um arquivo só possui um grafo e cada grafo está inserido em um único arquivo; um arquivo possui vários elementos de um grafo, mas cada elemento só está armazenado em um arquivo; um grafo possui vários elementos, mas cada elemento está atrelado a apenas um grafo; e um grafo possui várias interdependências, mas cada interdependência armazenada só pertence a um grafo.

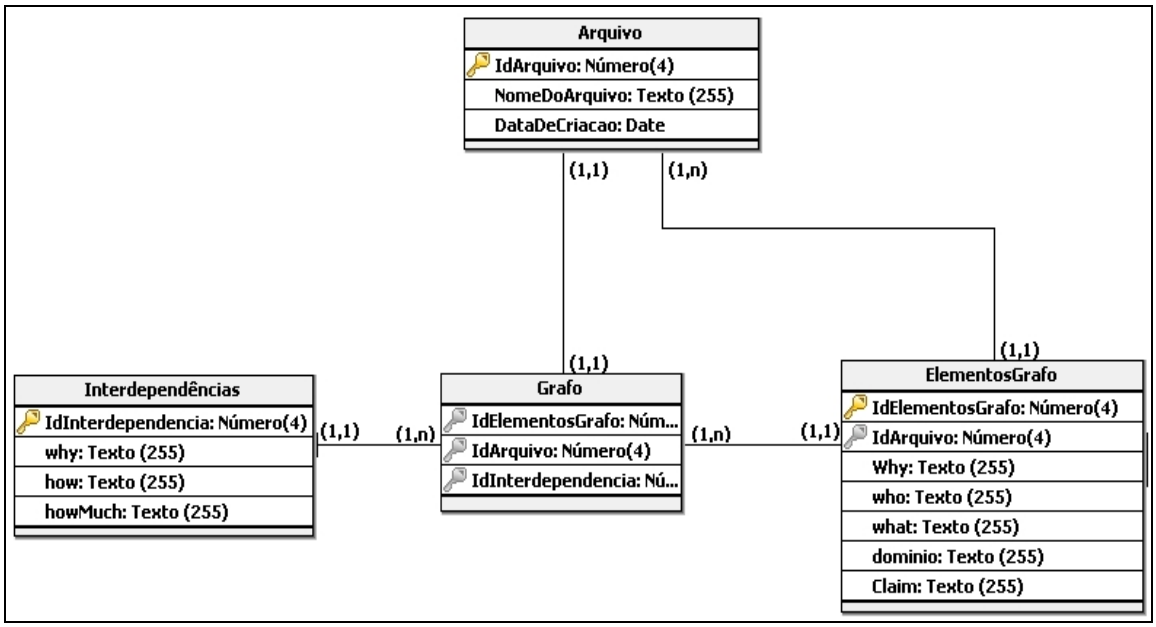

Figura 4.11: MER lógico das tabelas do repositório de requisitos

Ao enviar o arquivo XML contendo o grafo desejado, os módulos de classificação e versionamento também são ativados, uma vez que estas atividades ocorrem automaticamente, sem a intervenção do usuário. A seguir são apresentados cada um destes módulos.

4.2 Ferramentas Utilizadas

Para a execução dos mecanismos da *Cognitio Tool*, são necessárias algumas ferramentas que auxiliam tanto o desenvolvimento dos grafos quanto o armazenamento e manipulação dos grafos no MySQL. Essas ferramentas estão apresentadas nas Seções 4.3.1 e 4.3.2 a seguir.

4.2.1 Ferramenta StarUML

Tendo em vista a existência de ferramentas que auxiliam o desenvolvimento de grafos do *framework* NFR, e como o processo de reuso de requisitos provê que para o armazenamento dos requisitos tem-se como entrada os grafos SIG, a

ferramenta StarUML (STARUML, 2009) em sua versão 5.0, e com o *plugin* para criação de grafos SIG, foi utilizada como base para a modelagem dos requisitos no NFR.

A escolha da StarUML deu-se por esta ser uma ferramenta multi-plataforma, *open-source*, gratuita e que possui um *plugin* que desenvolve modelagem de requisitos de acordo com os conceitos do *framework* NFR. Além disso, a StarUML gera arquivos no formato XML e UML[1]. Esses arquivos servem, respectivamente, como entrada e saída da *Cognitio Tool*.

Assim, no módulo armazena.php, apresentado na Seção 4.2.2, é feito o reconhecimento das *tags* e dos atributos que compõem o arquivo gerado na StarUML, abrangendo apenas aqueles que interessam ao escopo deste trabalho. No entanto, nem todos os parâmetros de classificação definidos são cobertos pelos grafos SIG.

Dessa forma, para que a *Cognitio Tool* reconheça todos os parâmetros do grafo SIG de forma automática, embora o *framework* NFR não contemple dados como, por exemplo, papel do *stakeholder* que solicitou um determinado requisito, foram definidas as seguintes regras para a modelagem dos mesmos na ferramenta StarUML, como apresenta a Tabela 4.2.

Portanto, a sentença de cada elemento do grafo é referida na StarUML como *name*; o tipo, isto é, se o elemento é uma meta, softmeta, interdependência e qual o tipo de interdependência é definido na StarUML através do parâmetro *stereotype*; o parâmetro *why* é definido através do atributo *supplier*; o *how*, através do *client*; o parâmetro *who* é referido a partir do atributo *template*; o *where* através do atributo *package*; e o *when* por meio do atributo *claimsoftgoal*.

Tabela 4.2: Parâmetros de classificação de acordo com o os arquivos gerados pela ferramenta StarUML

[1] Neste caso, UML não indica o padrão de modelos Unified Modeling Language, mas a extensão utilizada pela ferramenta StarUML para gerar seus arquivos. Por exemplo, um arquivo no "formato" UML seria "teste.uml"

Parâmetro do elemento	Parâmetro do StarUML
Nome do elemento (Exemplo: nome da meta, softmeta, *claim*...)	*Name*
Tipo (se meta, softmeta, interdependência, inclusive *How much*)	*Stereotype*
Why	*Supplier*
How	*Client*
What	*Attribute*
Who	*Template*
Where	*Package*
When	*ClaimSoftgoal*

Assim, esta ferramenta é de grande importância para o *Cognitio*, haja vista é através dela que os grafos são construídos e editados, evitando possíveis erros durante o armazenamento. Caso o processo de reuso aqui definido abranja outras técnicas de modelagem de requisitos, tais como i*, casos de usos, entre outros, deve-se observar quais ferramentas provêem o desenvolvimento destes modelos, para que a inserção dos requisitos que a eles pertencem seja realizada também de forma automática.

4.2.2 Ferramenta Xampp

Para manter os dados armazenados no repositório de requisitos, e para que os mecanismos de armazenamento, classificação, busca e recuperação, que foram desenvolvidos na linguagem PHP, possam ser executados, utilizou-se a ferramenta Xampp (XAMPP, 2009), em sua versão 1.7.3.

Assim como a StarUML, a ferramenta Xampp é gratuita e multi-pltaforma. A mesma possui uma interface intuitiva e proporciona a funcionalidade de um SGBD. Além disso, a Xampp é considerada um dos mais

conhecidos e utilizados pacotes AMP (Apache, MySQL e PHP), cujas funcionalidades são usadas na *Cognitio Tool*.

4.3 Considerações Finais do Capítulo

Este Capítulo apresentou as características da ferramenta *Cognitio Tool*, a qual serve de apoio para a realização das atividades do processo *Cognitio*. Através da *Cognitio Tool* é possível que haja a automatização do processo de reuso de requisitos, o que é de extrema importância uma vez que incentiva a prática do reuso, promovendo a diminuição do retrabalho durante o desenvolvimento de software.

Embora nem todos os módulos estejam funcionando plenamente na ferramenta, foram definidas as funcionalidades inerentes a cada um, visando a expansão da *Cognitio Tool*. Além disso, estabeleceu-se que a mesma pudesse ser utilizada na *web* tendo em vista a abrangência da *Cognitio Tool* a um maior número de usuários.

Dessa forma, a fim de se validar tanto *Cognitio* quanto a ferramenta *Cognitio Tool*, o Capítulo 5 a seguir apresenta um estudo de caso mostrando como estes contribuem para o reuso de requisitos.

Capítulo 5

Estudo de Caso

Este estudo de caso foi desenvolvido com o intuito de validar o *Cognitio* como processo de reuso de requisitos. Para tanto, foram criados alguns cenários a fim de representar requisitos inerentes aos exemplos apresentados nas Seções a seguir. Logo, os requisitos foram modelados de acordo com os artefatos do *framework* NFR, através da ferramenta StarUML.

A seguir, na Seção 5.1 é apresentada uma breve descrição acerca de sistemas móveis. A Seção 5.2 apresenta o sistema modelado e usado neste estudo de caso. Na Seção 5.3 é apresentada a utilização do *Cognitio* para o reuso dos requisitos apresentados. Na Seção 5.4 são abordadas as contribuições e resultados obtidos a partir do estudo de caso. Na Seção 5.5 são apresentadas as considerações finais do Capítulo.

5.1 Sistemas Móveis

A tecnologia móvel forma a base da principal revolução tecnológica do século XXI, permitindo ao indivíduo comunicar-se a qualquer momento e em qualquer lugar.

Os pequenos dispositivos móveis, como celulares, Palms, Laptops, e a comunicação sem fio formam um dos principais meios para que as pessoas se mantenham constantemente informadas, onde quer que estejam.

Alguns tipos de aplicações procuradas são para o acesso a informações e notícias, leitura de correio eletrônico, transações bancárias e entretenimento, além do o uso da Internet por estes dispositivos, caracterizando o conceito de Internet Móvel (ou Internet sem fio).

Logo, tendo em vista o crescente número de usuários destes dispositivos, aumentam também as exigências para os mesmos. Desta forma, podem-se elencar como principais requisitos não funcionais desta tecnologia a segurança, usabilidade, disponibilidade e performance. Além disso, características como eficiência de energia e capacidade de armazenamento, são fundamentais para o funcionamento destes equipamentos.

A seguir, é apresentada a modelagem no *framework* NFR de alguns requisitos inerentes a um dispositivo móvel celular, abrangendo requisitos funcionais e não funcionais.

5.2 Sistema Móvel Celular

O sistema utilizado neste estudo de caso trata-se de um exemplo simples de sistema celular, o qual possui sete requisitos funcionais e quatro não funcionais, listados a seguir.

Requisitos Não Funcionais

RNF1: O sistema deve apresentar segurança. Para tanto, o mesmo deve integrar controle de acesso com opções de *firewall* ou antivírus.

RNF2: O sistema deve apresentar usabilidade aos usuários. Para tanto, o mesmo deve apresentar opções de configuração personalizada do sistema aos usuários.

RNF3: O sistema deve apresentar eficiência de energia. Para tanto, ele deve possuir opções de controle de energia por parte o usuário

RNF4: O sistema deve apresentar capacidade de armazenamento. Para tanto, o mesmo deve permitir a compactação dos arquivos inseridos pelo usuário.

Requisitos Funcionais

RF1: O sistema deve exigir ao usuário que o mesmo informe PIN ao ligá-lo;

RF2: O sistema deve permitir que o usuário informe se deseja acionar uma forma adicional de identificação;

RF3: O sistema deve integrar tecnologias como *firewall* e softwares antivírus;

RF4: O sistema deve permitir que o usuário modifique a configuração, de acordo com suas necessidades, possibilitando uma melhor interface;

RF5: O sistema deve possibilitar a compactação e descompactação de arquivos;

RF6: O sistema deve possibilitar o recebimento, gravação e envio de mídias;

RF7: O sistema deve monitorar a perda de energia de sua bateria, sugerindo recarga da mesma.

A partir destes requisitos funcionais e não funcionais, foi desenvolvida a modelagem no *framework* NFR de um sistema móvel celular. Alguns dos requisitos funcionais apresentados foram detalhados e subdivididos, para melhor representação. A Figura 5.1 apresenta o modelo criado.

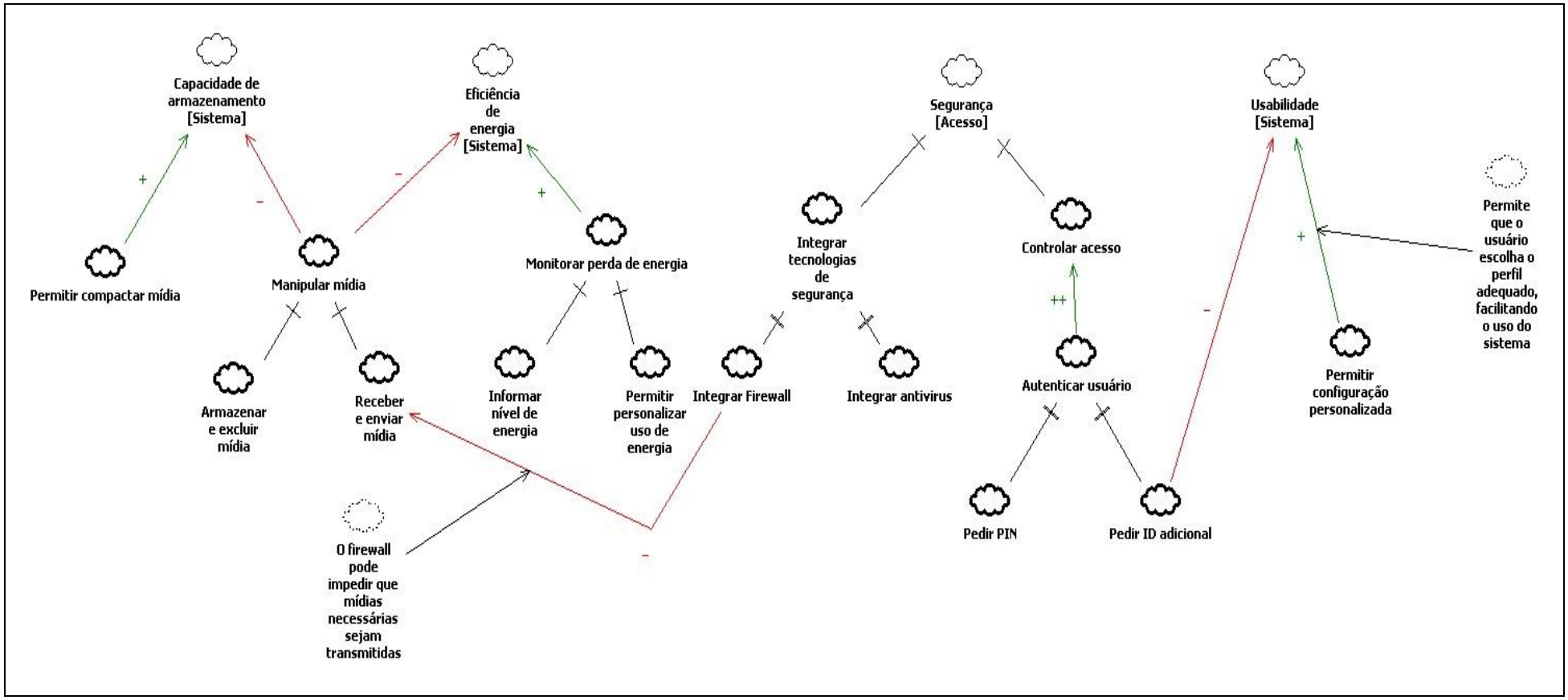

Figura 5.1: Sig de sistema móvel celular

Na Figura 5.1 pode-se observar que todos os RNFs acima listados são representados por softmetas, e os sete RFs foram subdivididos em quinze metas, a fim de serem melhor especificados. Como exemplo, visualiza-se que o monitoramento de perda de energia auxilia a eficiência desta no sistema, enquanto que a manipulação de mídias prejudica a mesma.

Há ainda duas *claims*, sendo que uma justifica o porque do *firewall* prejudicar o envio e recebimento de mídias, e a outra indica o como a configuração personalizada do sistema possibilita melhor usabilidade.

5.3 Aplicação do Processo *Cognitio* no Estudo de Caso

Para apresentar a aplicação do *Cognitio*, foram criados alguns cenários, representando a necessidade de reuso de requisitos para um sistema móvel.

Dessa forma, para melhor embasamento da aplicação do reuso, a Tabela 5.1 apresenta a classificação, de acordo com a linguagem Q7, da modelagem dos requisitos apresentada na Figura 5.1.

Como foi visto no Capítulo 3, as principais contribuições da classificação de cada elemento do *framework* NFR são apresentar ao usuário o comportamento de cada elemento modelado, isto é, se trata-se de meta, softmeta, *claim*, domínio, pré-rastreabilidade, entre outros.

Logo, observa-se o seguinte: a softmeta **capacidade de armazenamento** possui dois elementos filhos, os quais são **permitir compactar mídia** e **manipular mídia**. Verifica-se, no entanto, que os relacionamentos entre estes são distintos: o primeiro é do tipo *Help*, isto é, a compactação de mídia auxilia na capacidade de armazenamento; e a segunda do tipo *Hurt*, o que indica que a manipulação de mídia, a qual tem como filhos **armazenar e excluir mídia** e **receber e enviar mídia**, prejudica a capacidade de armazenamento, tendo em vista que através do armazenamento e recebimento de mídia, o espaço disponível no sistema móvel pode ser comprometido.

Tabela 5.1: Classificação dos requisitos

Why	**Who**	**What**	**Where**	**When**	**How**	**How Much**
Capacidade de armazenamento	Cliente	Sistema	Armazenamento de mídia	-	Permitir compactar mídia	Help
Capacidade de armazenamento	Cliente	Sistema	Armazenamento de mídia	-	Manipular mídia	Hurt
Manipular mídia	Cliente	Sistema	Manipulação de mídia	-	Armazenar e excluir mídia	AND
Manipular mídia	Cliente	Sistema	Manipulação de mídia	-	Receber e enviar mídia	AND
Eficiência de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Manipular mídia	Hurt
Eficiência de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Monitorar perda de energia	Help
Monitorar perda de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Informar nível de energia	AND
Monitorar perda de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Permitir personalizar uso de energia	AND
Segurança	Analista de segurança	Acesso	Segurança de acesso a sistemas móveis	-	Integrar tecnologias de segurança	AND
Segurança	Analista de segurança	Acesso	Segurança de acesso a sistemas móveis	-	Controlar acesso	AND
Integrar tecnologias de segurança	Analista de segurança	Acesso	Tecnologias de segurança	-	Integrar Firewall	OR
Integrar tecnologias de segurança	Analista de segurança	Acesso	Tecnologias de segurança	-	Integrar Antivirus	OR
Receber e enviar mídia	Analista de segurança	Sistema	Tecnologias de segurança	O firewall pode impedir que mídias necessárias sejam transmitidas	Integrar Firewall	Hurt
Controlar acesso	Analista de segurança	Acesso	Tecnologias de segurança	-	Autenticar usuário	Make
Autenticar usuário	Analista de segurança	Acesso	Controles de acesso	-	Pedir PIN	OR
Autenticar usuário	Analista de segurança	Acesso	Controles de acesso	-	Pedir ID adicional	OR
Usabilidade	Gerente de usabilidade	Sistema	Interface Humana-Máquina	-	Pedir ID adicional	Hurt
Usabilidade	Gerente de usabilidade	Sistema	Interface Humana-Máquina	Permite que o usuário escolha o perfil adequado, facilitando o uso do sistema	Permitir configuração personalizada	Help

A seguir, estão apresentados três cenários para melhor entendimento em relação à utilização do processo *Cognitio.* Esses cenários representam situações nas quais o modelo ilustrado na Figura 5.1 é reusado.

Cenário 1

Um engenheiro de requisitos está desenvolvendo a elicitação de requisitos para um sistema móvel. Ele sabe que estes tipos de dispositivos, possuem capacidade de armazenamento limitada, no entanto, não sabe quais requisitos devem existir para garantir alcançar esse requisito não-funcional. Dessa forma, ele deseja conhecer todos os requisitos, independente do tipo de relacionamento entre eles, ou seja, tanto os que auxiliam quanto os que prejudicam a capacidade de armazenamento.

Logo, o engenheiro de requisitos utilizou o processo *Cognitio* em conjunto com a ferramenta *Cognitio Tool*, e realizou as seguintes buscas.

i) Como ele deseja encontrar requisitos em relação à capacidade de armazenamento, ele inseriu o termo "armazenamento" no parâmetro *why* e o termo "sistema móvel" no parâmetro *what*. No entanto, não foi localizado nenhum requisito que atendesse a estes dois parâmetros concomitantemente.

Tendo em vista o insucesso durante a busca, o engenheiro de requisitos resolveu restringir a consulta a apenas um parâmetro, realizando a seguinte inserção:

ii) O engenheiro de requisitos inseriu apenas o termo "armazenamento" no parâmetro *why*. Através desta consulta, foi retornado a ele o resultado apresentado na Tabela 5.2.

Logo, foi retornado ao engenheiro de requisitos o requisito capacidade de armazenamento, o qual possui dois elementos filhos: Permitir compactar mídia e Manipular mídia, representados na coluna do parâmetro How. Com isto, ele reconheceu, através do parâmetro How much, que um elemento filho

auxilia enquanto que o outro contribui de forma negativa para o aumento da capacidade de armazenamento. Dessa forma, ele decidiu recuperá-los, para realizar posterior validação.

Tabela 5.2: Resultado da consulta "armazenamento" pelo parâmetro why

Why	**Who**	**What**	**Where**	**When**	**How**	**How Much**
Capacidade de armazenamento	Cliente	Sistema	Armazenamento de mídia	-	Permitir compactar mídia	Help
Capacidade de armazenamento	Cliente	Sistema	Armazenamento de mídia	-	Manipular mídia	Hurt

Cenário 2

Um aluno da disciplina de redes de computadores está desenvolvendo uma simulação para apresentar o funcionamento de uma rede de sensores sem fio. Logo, ele sabe que esta tecnologia possui um limitado tempo de vida, mas não sabe que requisitos podem auxiliar e prejudicar a necessidade pelo baixo consumo de energia de um sensor.

Tendo em vista essas dúvidas, o aluno decidiu, através do processo *Cognitio*, realizar buscas para conhecer estes requisitos, e reusá-los. Para tanto, ele realizou as seguintes buscas:

i) Como ele não tem idéia de quais requisitos estão atrelados à eficiência de energia, em sua primeira consulta ele inseriu os termos “energia” no parâmetro *why* e “móvel” no parâmetro *where*. Como resultado a essa consulta, ele recebeu os seguintes dados apresentados na Tabela 5.3.

Vendo os resultados, o aluno percebeu que a manipulação de mídia contribui de forma negativa para o tempo de vida útil do dispositivo, como apresenta a primeira linha da coluna *How much*, e lembrou-se que ainda não havia definido os requisitos para a manipulação de mídia, uma vez que, de fato, ocorre entre os sensores sem fio.

Tabela 5.3: Resultado da busca dos termos energia e móvel nos parâmetros *why* e *where*, respectivamente

Why	**Who**	**What**	**Where**	**When**	**How**	**How much**
Eficiência de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Manipular mídia	Hurt
Eficiência de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Monitorar perda de energia	Help
Monitorar perda de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Informar nível de energia	AND
Monitorar perda de energia	Gerente de disponibilidade	Sistema	Disponibilidade de componente móvel	-	Permitir personalizar uso de energia	AND

Logo, o aluno decidiu verificar se estes requisitos também estão armazenados no repositório, realizando as seguintes consultas: o termo "manipular mídia" no parâmetro *why*, e o termo "mídia" no parâmetro *where*. Através desta consulta, o seguinte resultado apresentado na Tabela 5.4 foi retornado a ele:

Tabela 5.4: Resultado da busca pelos termos "manipular mídia" e "mídia" nos parâmetros *why* e *where*, respectivamente.

Why	**Who**	**What**	**Where**	**When**	**How**	**How much**
Manipular mídia	Cliente	Sistema	Manipulação de mídia	-	Armazenar e excluir mídia	AND
Manipular mídia	Cliente	Sistema	Manipulação de mídia	-	Receber e enviar mídia	AND

Tendo em vista que a manipulação de mídia envolve o recebimento e envio de mídias, o aluno percebeu que através desses requisitos, pode haver queda na segurança do sensor, podendo o mesmo ser atacado por vírus, ou por *hackers*. Dessa forma, ele buscou na ferramenta algo relacionado com antivírus e sistemas de *firewall*. Para tanto, realizou a seguinte busca: inseriu o termo "antivírus" no parâmetro *why* e "segurança" no parâmetro *where*. No entanto, nenhum resultado foi retornado.

Porém, o aluno continuou a busca e inseriu o termo “antivírus” no parâmetro *how* e continuou com o termo “segurança” no parâmetro *where*. Para essa consulta, foi retornado o resultado apresentado na Tabela 5.5.

Tabela 5.5: Resultado da busca pelos termos "antivírus" e "segurança" nos parâmetros *how* e *where*, respectivamente

Why	**Who**	**What**	**Where**	**When**	**How**	**How much**
Integrar tecnologias de segurança	Analista de segurança	Acesso	Tecnologias de segurança	-	Integrar Antivirus	OR

Logo, ele percebeu que o antivírus está inserido, de fato, no contexto de segurança e é uma decomposição, do tipo Ou, como apresenta a coluna *How much*, para se alcançar o requisito integrar tecnologias de segurança. Dessa forma, o aluno decidiu realizar novas buscas para saber que outras tecnologias de segurança estão modeladas, através da inserção dos seguintes termos: “analista de segurança” no parâmetro *who*, “segurança” nos parâmetros *why* e *where*; e o termo “Or” no parâmetro *How much*. Como resultados, o aluno obteve os dados listados na Tabela 5.6.

Tabela 5.6: Resultado da consulta pelos termos "analista de segurança" no parâmetro *who*, "segurança" nos parâmetros *why* e *where*, e "Or" no parâmetro *how much*

Why	**Who**	**What**	**Where**	**When**	**How**	**How much**
Integrar tecnologias de segurança	Analista de segurança	Acesso	Tecnologias de segurança	-	Integrar Firewall	OR
Integrar tecnologias de segurança	Analista de segurança	Acesso	Tecnologias de segurança	-	Integrar Antivirus	OR

Perceba que repetiu-se a informação cujo parâmetro *how* é “integrar antivírus” e o aluno encontrou uma outra tecnologia para prover a segurança

nos sensores, que é a meta "integrar *firewall*". Logo, ao recuperar estes requisitos, o aluno percebeu que a ação de antivírus e de *firewall*, embora protejam o seu sistema, prejudicam a eficiência de energia.

Desta forma, como o foco de sua simulação é demonstrar a eficiência de energia e a manipulação de mídia em sensores sem fio, e não a segurança nestes dispositivos, ele não reusou os requisitos atrelados ao RNF segurança.

A Figura ilustra os requisitos que foram recuperados e reusados pelo aluno.

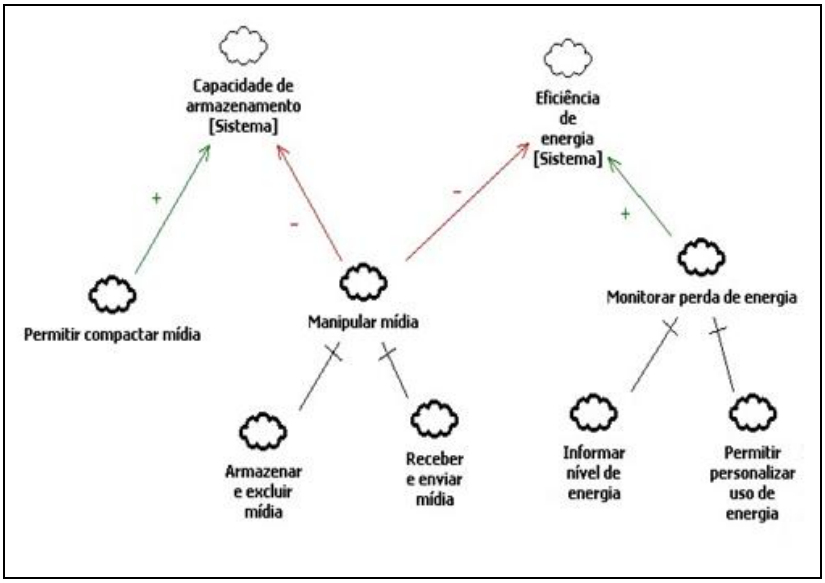

Figura 5.2: Requisitos reusados no Cenário 2

Cenário 3

Um engenheiro de requisitos desenvolveu a elicitação de requisitos para um laptop visando a sua usabilidade. Ao utilizar a ferramenta *Cognitio Tool*, consultando os requisitos já armazenados em relação à usabilidade, ele verificou que, em se tratando de dispositivos móveis, só existiam os dados apresentados na Tabela 5.7.

Tabela 5.7: Resultado da consulta pelo termo usabilidade no parâmetro *why*

Why	**Who**	**What**	**Where**	**When**	**How**	**How much**
Usabilidade	Gerente de usabilidade	Sistema	Interface Humana-Máquina	-	Pedir ID adicional	Hurt
Usabilidade	Gerente de usabilidade	Sistema	Interface Humana-Máquina	Permite que o usuário escolha o perfil adequado, facilitando o uso do sistema	Permitir configuração personalizada	Help

Logo, como o engenheiro já havia especificado os requisitos do seu projeto, ele pode através do módulo glossario.php, conhecer como se dá o processo *Cognitio*, qual a notação usada, e quais os passos para o reuso de requisitos. Dessa forma, ele decidiu contribuir com o reuso alimentando o repositório de requisitos com novos requisitos sobre usabilidade. A Figura 5.3 apresenta a modelagem gerada, a partir dos requisitos elicitados pelo engenheiro de requisitos.

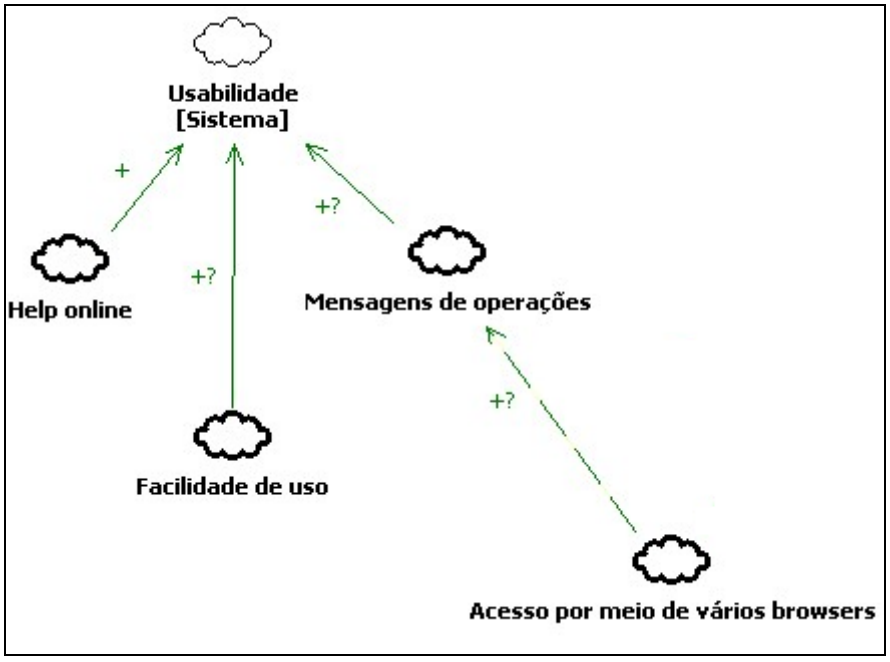

Figura 5.3: Modelagem do requisito Usabilidade

Assim, com a modelagem desses requisitos e seu posterior armazenamento, o engenheiro de requisitos contribui para o desenvolvimento

de software para o reuso. Enquanto que nas situações descritas nos Cenários 1 e 2, tanto o engenheiro de requisitos quanto o aluno desenvolveram software com o reuso.

Dessa forma, vê-se através dos cenários acima descritos que, dependendo dos termos e dos parâmetros utilizados durante a busca, os resultados podem ser distintos. Além disso, verificou-se como o processo *Cognitio* auxilia no desenvolvimento de software com e para o reuso.

5.4 Considerações Finais do Capítulo

Neste Capítulo foi apresentado um estudo de caso a fim de se validar o processo definido nesta dissertação. Verificou-se, através do exemplo modelado, como ocorreu a classificação dos mesmos, e como a modelagem e classificação auxiliam o posterior reuso dos requisitos.

Por meio dos três cenários que foram apresentados, viu-se que o *Cognitio*, de fato, auxilia no reuso. O Capítulo 6 a seguir apresenta com mais detalhes as contribuições, vantagens e desvantagens do *Cognitio*, comparando-o a outros processos de reuso de requisitos.

Capítulo 6

Trabalhos relacionados

Neste Capítulo são apresentados trabalhos que estão relacionados ao processo desenvolvido, a fim de se realizar um estudo comparativo entre os mesmos. Desta forma, na Seção 6.1 é apresentado um estudo comparativo entre os trabalhos escolhidos e o processo desenvolvido. Na Seção 6.2 são abordadas as considerações finais do Capítulo.

6.1. Processos de Reuso de Requisitos

De acordo com Davenport (1993), processo consiste em um conjunto de atividades estruturadas e mensuráveis para produzir uma saída específica para um determinado cliente ou mercado. Já para Hammer e Stanton apud Usirono (2003), processo é um grupo de tarefas inter-relacionadas e que geram valor para o cliente.

No âmbito da Engenharia de Software, o termo "processo de software" pode ser definido como um processo composto de um conjunto de atividades capazes de conduzir a organização envolvida com a construção de produtos de

software com qualidade e custos previsíveis, de forma eficiente, gerenciada e com a possibilidade de melhoria constante (SANT'ANNA, 2000).

Na literatura existem alguns processos de reuso de requisitos, alguns dos quais são abordados neste Capítulo com o intuito de verificar as vantagens e desvantagens do *Cognitio* em relação aos demais processos. A Tabela 6.1 apresenta os trabalhos relacionados.

Tabela 6.1: Trabalhos relacionados

Referência	Propriedades-chaves
(Kang,1990)	Desenvolvimento de linhas de produto
(Toval et al, 2002)	Segurança de sistemas
(López; Laguna; García, 2002)	Reuso de requisitos através de utilização de repositórios de armazenamento
(Moros; Chicote; Toval, 2008)	Variabilidade de requisitos
(Ebling; Audy; Prikladnick, 2009)	Linhas de produto

6.1.1 Vantagens do Processo *Cognitio*

Verificou-se, ao longo deste trabalho, os interesses e soluções determinadas para o desenvolvimento tanto do *Cognitio* quanto da *Cognitio Tool*, trazendo algumas vantagens em relação a processos já consolidados existentes na literatura.

Dessa forma, o *Cognitio* estabelece um mecanismo próprio de classificação dos requisitos, de forma a permitir que vários conceitos tais como contexto, pré-rastreabilidade, pós-rastreabilidade, rastreabilidade entre requisitos, entre outros, possam ser verificados e reutilizados pelos *stakeholders*, de forma a permitir maior compreensão e, dessa forma, possibilitando um reuso de maior qualidade.

No entanto, estes aspectos não são considerados, por exemplo, nos processos apresentados em (KANG, 1990), (LÓPEZ, LAGUNA, GARCÍA, 2002), (EBLING, AUDY, PRIKLADNICK, 2009).

Além desses aspectos, o *Cognitio* considera tanto o reuso de RFs quanto de RNFs, de forma que trabalhos como (TOVAL et al, 2002), (LÓPEZ, LAGUNA, GARCÍA, 2002), (MOROS, CHICOTE, TOVAL, 2008) não explicitam que tipo de requisitos reusam, deixando apenas clara a reutilização de RFs.

Além disso, o processo *Cognitio* visa a necessidade de se realizar as etapas inerentes à ER durante a modelagem dos requisitos, de forma que os mesmos passem por etapas como verificação e validação, buscando melhorias durante o desenvolvimento de software para o reuso, tendo como foco a elicitação de requisitos.

Em relação à modelagem dos requisitos, sabe-se que o processo *Cognitio* utiliza o *framework* NFR. Em Kang (1990) é utilizado o Modelo Entidade-Relacionamento; Toval et al (2002) utiliza catálogos de requisitos; e López, Laguna e García (2002) utilizam um modelo conceitual. Já os trabalhos (MOROS, CHICOTE, TOVAL, 2008) e (EBLING, AUDY, PRIKLADNICK, 2009) não especificam qual o modelo usado para organizar os requisitos.

Por fim, nesta dissertação definiu-se estratégias de utilização da *Cognitio Tool*, embora a mesma não esteja totalmente implementada, mas apresenta módulos que servem de apoio à realização das etapas do *Cognitio*. Dessa forma, considera-se este ponto como vantagem, tendo em vista que trabalhos como (TOVAL et al, 2002) não apresentam ferramentas de suporte.

A Tabela 6.2 apresenta um comparativo com algumas características presentes nos processos de reuso de requisitos.

Tabela 6. 2- Comparativo entre os trabalhos relacionados e o *Cognitio*

Características	***Cognitio***	**Kang**	**Toval**	**López, Laguna e García**	**Moros, Chicote e Toval**	**Ebling, Audy e Prikladnick**
Foco	Elicitação de requisitos	Análise de contexto	Elicitação de requisitos		Modelagem de Variabilidade	ER distribuída e linhas de produtos
Pré-rastreabilidade	Sim	Sim	-	-	Sim	Sim
Pós-rastreabilidade	Sim, através de artefatos como gráficos, componentes, entre outros	-	-	Sim	Sim	-
Rastreabilidade entre requisitos	Contempla	-	Sim	Sim	Sim	Sim
Classificação dos ativos	Através da linguagem Q7	-	Os ativos são classificados de acordo com os perfis e domínios	-	De acordo com o tipo de cada requisito	-
Requisitos contemplados	Tanto RF quanto RNF	Tanto RF quanto RNF	Apenas RF	Apenas RF	Apenas RF	Tanto RF quanto RNF
Modelo de organização dos requisitos	Framework NFR	Modelo Entidade Relacionamento	Através de catálogos de requisitos	Modelo conceitual	-	-
Ferramenta de apoio	*Cognitio Tool*	-	SirenTool	R²	REMM-Studio	-

6.1.2 Desvantagens do Processo *Cognitio*

São elencadas como desvantagens do *Cognitio*, alguns pontos que não foram supridos nesta dissertação. Por exemplo, não foi contemplada a variabilidade dos requisitos, embora saiba-se que este é um ponto de grande valia para o reuso de requisitos, o qual é abordado em (MOROS, CHICOTE, TOVAL, 2008).

Além disso, o *Cognitio* abrange apenas grafos SIGs como modelos de requisitos, restringindo a sua utilização, uma vez que nem todos os usuários conhecem esse modelo, diferentemente de (KANG, 1990) e (LÓPEZ, LAGUNA, GARCÍA, 2002) que utilizam documentos comuns de requisitos para o provimento do reuso.

Outra desvantagem é o fato de tanto o *Cognitio* quanto a *Cognitio Tool* não reconhecerem semanticamente os documentos ou as representações dos requisitos, diferentemente de (LÓPEZ, LAGUNA, GARCÍA, 2002) os quais

reconhecem o léxico de cada parte dos requisitos, diminuindo ou evitando erros em suas especificações. Desta forma, no processo aqui definido cabe ainda ao usuário verificar e validar se cada requisito está especificado corretamente, com o intuito de diminuir riscos posteriores.

6.1.3 Semelhanças e Diferenças entre os Trabalhos Relacionados

Uma vez que os trabalhos relacionados e esta dissertação tenham interesses semelhantes, isto é, prover o reuso de requisitos, algumas semelhanças são encontradas entre eles.

Dessa forma tanto o *Cognitio* quanto o método FODA (KANG, 1990) preocupam-se com a análise de contexto de cada requisito, embora o método FODA preocupe-se tão somente em reusar artefatos pertencentes a um mesmo contexto, diferentemente do *Cognitio*.

Em relação ao mecanismo automatizado, tanto esta dissertação quanto o trabalho apresentado em (LÓPEZ, LAGUNA, GARCÍA, 2002) e (TOVAL et al, 2002) trazem os conceitos de repositório como forma de armazenamento e gerenciamento dos requisitos. Para tanto, todos estes trabalhos apresentam ferramentas de apoio. No entanto, embora todos os requisitos sejam armazenados, apenas o *Cognitio* evidencia a classificação dos ativos como forma de organização do repositório de requisitos.

6.2 Considerações Finais do Capítulo

Este Capítulo apresentou cinco trabalhos relacionados ao *Cognitio*, os quais nos serviram de base para o provimento tanto do processo quanto da ferramenta *Cognitio Tool*, bem como para analisar suas limitações buscando cobri-las no processo aqui definido.

Capítulo 7

Conclusões

Esta dissertação está centrada na área de Engenharia de Requisitos (ER), através da especificação de um processo de reuso de requisitos funcionais e não-funcionais. A principal motivação para o desenvolvimento deste trabalho foi o fato da ER ser por muitas vezes negligenciada durante o processo de desenvolvimento de software. Com isto, ocorrem grandes custos aos desenvolvedores, tendo em vista as falhas na especificação do sistema.

Dessa forma, com a aplicação do processo de reuso definido, é disponibilizado aos engenheiros de requisitos e demais *stakeholders* uma fonte de informações para elicitação de requisitos, uma vez que o usuário pode utilizar o processo e a ferramenta *Cognitio Tool* como repositório de informação, para busca e recuperação de requisitos.

Tendo isso em vista, o processo *Cognitio* aliado à ferramenta *Cognitio Tool* aqui apresentada, estimula o desenvolvimento de software para o reuso e com o reuso, além de estabelecer passos necessários para que os requisitos que compõem o repositório gerado apresentem a qualidade esperada, uma vez que

todo e qualquer grafo ali armazenado deve ter sido anteriormente validado e verificado.

O *framework* NFR foi usado para modelar os requisitos a serem armazenados, tendo em vista que o mesmo abrange os conceitos de RF e RNF, dando ênfase a este último, além de apresentar os conflitos e as contribuições entre tais requisitos.

A linguagem Q7 de organização de qualidades foi escolhida, uma vez que através dessa é proporcionado um melhor entendimento e organização das informações inerentes aos requisitos dentro do *Cognitio*. Além disso, com Q7 pode-se abranger o *Cognitio* a outras abordagens de modelagem de requisitos, havendo a necessidade de adaptar os parâmetros do Q7 para cada tipo de modelo.

7.1 Contribuições

O trabalho desenvolvido tem como contribuição a definição de um processo de reuso de requisitos bem como o desenvolvimento de um repositório, o qual permite que requisitos sejam armazenados e recuperados, apresentando-se, portanto, como fonte de conhecimento, que pode auxiliar durante a elicitação de requisitos.

Dessa forma, através do processo são estabelecidas atividades necessárias para que o reuso de requisitos ocorra, as quais são centradas na busca pela qualidade e confiabilidade dos ativos a serem reusados. Dessa forma, alguns pontos considerados como contribuição desta dissertação, são:

- O desenvolvimento de um processo de reuso de requisitos que provê o desenvolvimento de software com reuso e para o reuso;
- A adaptação da linguagem Q7 não só para o escopo de reuso de RNFs, mas também para os RFs.

- O desenvolvimento dos conceitos funcionais da ferramenta *Cognitio Tool* como sistema que automatiza o processo de reuso;

Embora a *Cognitio Tool* ainda não contemple todas as fases do processo definido, visualiza-se no estudo de caso deste trabalho que o processo de reuso de requisitos é viável e que contribui para o desenvolvimento de sistemas a partir dos requisitos ali contidos.

7.2 Limitações

Uma vez que o *framework* NFR foi escolhido como guia para a modelagem dos requisitos que compõem o *Cognitio*, alguns passos deste foram desenvolvidos adaptando-se aos conceitos do NFR.

Dessa forma, embora seja possível ampliá-lo para a utilização de outros métodos de modelagem, até o momento ele contempla apenas o armazenamento de requisitos modelados nos grafos SIG do *framework* NFR.

Além disso, tendo em vista o pequeno número de grafos utilizados no estudo de caso desta dissertação, aliado ao pequeno porte dos mesmos, não pode-se verificar neste trabalho questões quantitativas como, por exemplo, o desempenho do repositório em função do número de dados contidos no mesmo.

Outra limitação é que o *Cognitio* não consiste em verificar se os dados armazenados são de qualidade, isto é, se eles passaram pelas fases de elicitação, validação e verificação do processo de Engenharia de Requisitos. Logo, esta tarefa fica incumbida aos *stakeholders*, o que não se pode garantir que as mesmas são cumpridas, embora o *Cognitio* exija que as mesmas devam ser realizadas antes da ocorrência do armazenamento dos grafos no repositório.

7.3 Trabalhos Futuros

Tendo em vista que o reuso de requisitos é reconhecido como uma atividade vital para a diminuição dos custos durante a especificação de uma aplicação e para o aumento da qualidade do sistema desenvolvido, estão elencados alguns trabalhos futuros, buscando-se a utilização plena do *Cognitio*.

Desta forma, em relação à modelagem dos requisitos propõe-se abranger outros métodos de modelagem, tais como casos de uso, ou i*, de forma que o repositório seja diversificado, e atenda cada vez mais usuários com necessidades e conhecimentos distintos.

Em relação à pós-rastreabilidade existente no *Cognitio*, propõe-se a verificação de outras formas de recuperação das informações contidas no repositório, isto é, que diagramas são indispensáveis, ou que outros dados são pertinentes para o auxílio do reuso de requisitos. Para isso, deve ser observado como será feito o rastreamento entre elas e os requisitos recuperados.

Propõe-se, ainda, que o *Cognitio* abranja o conceito de Aspectos em seu escopo, tendo em vista a importância destes para o tratamento de requisitos.

Em relação à *Cognitio Tool* propõe-se que futuramente a ferramenta dê suporte ao versionamento e à edição dos grafos, uma vez que se torna uma forma de facilitar a manipulação dos modelos, de maneira mais intuitiva e rápida, embora a ferramenta StarUML já proveja esta última atividade.

Referências Bibliográficas

ALMEIDA, K. M. **Garantia da qualidade de software no desenvolvimento baseado em reuso: uma abordagem no contexto do Ministério da Defesa e seus comandos subordinados**. Instituto Militar de Engenharia – IME. Rio de Janeiro, RJ. 2004.

ANDREOPOULOS, B. **Achieving Software Quality Using the NFR Framework: Maintainability and Performance**. In Proceedings of the 3rd International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'04), Special Session on Object-oriented software engineering, Cairo, Egito. 2004.

ANTÓN, A. **Goal Based Requirements Analisys. Second International**. Conference on Requirements Engineering (ICRE'96). Pags. 136-144. 1996.

BELLIZONA, R.; FUGINI, M. G.; PERCINI, B. **An environment for specification reuse**. Technical Report POLIMIUDUNIV. 92.E.2.9E.8.4, Ithaca. 1992.

BOEHM, B. **Characteristics of Software Quality**. North Holland Press. 1978.

BPMN. **Business Process Modeling Notation**. Disponível em: http://www.bpmn.org/. Acesso em: dezembro de 2009.

BROOKS, F. P. **No Silver Bullet. Essence and accidentes of software engineering**. IEEE Computer, pp 10-19. 1987.

CASTOR, A. O. P. **Rastreamento de Requisitos no processo de desenvolvimento de software orientado a agentes**. Universidade Federal de Pernambuco. Recife-PE. 2004.

CHAVES, F. C. **Especificação e documentação de requisitos: um modelo aplicável à análise da informação utilizando "casos de uso".** Universidade Estadual de Campinas. Campinas-SP. 2005.

CHUNG, L. **Representation and utilization of non-functional requirements for information system design**. In: ANDERSEN, R., JR., J. A. B., e SØLVBERG, A., editores, CAiSE, volume 498 of Lecture Notes in Computer Science, págs. 5–30. Springer, 1991.

CHUNG, L.; NIXON, B. A.; YU, E. **Using quality requeriments to systematically develop quality software**. In: Fourth International Conference on Software Quality. McLean, VA, U.S.A. 1994.

CHUNG, L.; NIXON, B.; YU, E.; MYLOPOULOS, J. **Non-Functional Requirements in Software Engineering**. Kluwer Academic Publishers. Boston/Dordrecht/London. 2000.

COUTO, A. A. **Um processo de validação de requisitos não-funcionais baseado no NFR-Framework**. Universidade Metodista de Piracicaba. Piracicaba/SP. 2009.

CYBULSKI, J. L. **Patterns in software requirements reuse**. Technical report, Department of Information Systems. University of Melbourne. 1998.

CYSNEIROS, L. M. **Non-Functional Requeirements: From Elicitation to Conceptual Models**. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro-RJ. 2001.

_____. **Evaluating the effectiveness of using catalogues to elicit non-functional requirements**. In: Proc. Of 10th Workshop in Requirements Engineering. PP. 107-115. 2007.

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

DAVENPORT, T. H. **Process Innovation: Reengineering Work through Information Technology**, Harvard Business School Press, Boston, 1993.

DAVIS, A. **Software Requirements: Objects Functions and States.** Prentice Hall. 1993.

DAY, L. E. **Requirements re-use in the IT business process web implementation product family**. In: International Workshop on Requirements Reuse in System Family Engineering. Carlos III University of Madrid. Madrid-Spain. 2004.

EBLING, T.; AUDY, J. L. N.; PRIKLADNICKI, R. **Towards a requirements reuse method using Product Line in distributed environments**. Proceedings of the XII Workshop on Requirements Engineering (WER), Chile. 2009.

FILMAN, R. E. et al. Aspect**-Oriented Software Development**. Addison-Wesley. 2005.

FOOK, K. D.; ABDELOUAHAB, Z. **Reutilização de Objetivos na Engenharia de Requisitos**. In: WER - IV Workshop Engineering Requirements, Buenos Aires. 2001.

HOMOD, S.; RINE, D. **Building requirements repository using requirements transformation techniques to support requirements reuse**. World Multi-Conference on Systemics, Cybernetics and Informatics, Volume 2. 1999.

KANG, K.; COHEN, S.; HESS, J.; NOVAK, W.; PETERSON, S. **Feature-Oriented Domain Analysis (FODA) Feasibility Study**. Technical Report CMU/SEI-90-TR-21. 1990.

KELLER, S.E. et al. **Specifying Software Quality Requirements with Metrics**. in Tutorial System and Software Requirements Engineering IEEE Computer Society Press. pp:145-163. 1990.

KICZALES, G. et al. **Aspect-Oriented Programming**. In: PROC. OF THE 11th EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, Vol. 1241 de LNCS, Springer-Verlag, 1997. p. 220-242.

KOSCIANSKI, A.; SOARES, S. M. **Qualidade de Software**. Editora Novatec. São Paulo, Brasil. 2006.

KOTONYA, G.; SOMMERVILLE, I. **Requirements engineering. Processes and techniques**. England: John Wiley & Sons Ltda. pág. 282. 1998.

KWASNIK, B. H. **The role of classification in knowledge representation and discovery**. Library Trends 48(1): p22-47. 1999.

LAM, W.; McDERMID, J. A.; VICKERS, A. J. **Ten Steps Towards Systematic Requirements Reuse**. Requirements Engineering Journall, 102-113. 1997.

LAMSWEERDE, A. **Goal-Oriented Requirements Engineering: A Guided Tour**. Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering; Toronto. 249-263. 2001.

LEITE, J. C. S. P. **Engenharia de requisitos**. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro-RJ. pág. 63. 1994.

_____. **Requisitos: a ponte entre a organização e o software**. Palestra da Rio Info - Seminário Internacional de Engenharia de Software. 2006.

LEITE, J. C. S. P.; DOORN, J. H. **Perspectives on software requirements**. Massachussetts, Kluwer Academic Publishers. 2004.

LEITE, J. C. S. P.; YU, Y.; LIU, Y., YU E. S. K.; MYLOPOULOS, J. **Quality-Based Software Reuse.** CAiSE-05, LNCS 3520, pp.535-550, Springer-Verlag. 2005.

LOPEZ, O.; LAGUNA, M. A.; GARCIA, F. J. **Metamodeling for Requirements Reuse**. Anais do WER02 - Workshop em Engenharia de Requisitos, Valencia, Espanha. 2002.

MACEDO, N.A.M.; LEITE, J.C.S.P. **Elicit@99 um Prototipo de Ferramenta para a Elicitacao de Requisitos**. In Proceedings of WER'1999. p. 45-55. 1999.

MONZÓN, A.; DUEÑAS, J. C. **Experience-based Approach to Requirements Reuse in Product Families with DOORS**. IWREQFAM, Technical Report, Universidad Politécnica de Madrid. 2004.

MOROS, B.; CHICOTE, C. V.; TOVAL, A. **Metamodeling variability to enable requirements reuse**. Workshop on Exploring Modeling Methods for Systems Analysis and Design – EMMSAD'08. Montpellier-France. 2008.

MOSTOW, J. **Towards Better Models of Design Process**. AI Magazine, Vol. 6 No. 1 1989, pp:44-57 Spring 85.

NUSEIBEH, B.; EASTERBROOK, S. **Requirements engineering: a roadmap**. In ICSE '00: Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA. ACM Press. pág. 35.46. 2000.

RASHID, A. et al. **Early aspects: a model for aspect-oriented requirements engineering**. In: PROC. OF IEEE JOINT CONFERENCE ON REQUIREMENTS ENGINEERING, Germany, p. 199-202. 2002.

RAMOS, R. A. **Treinamento prático em UML.** Universo dos Livros Editora LTDA. 1999.

RAMOS, R. A.; ARAÚJO, J.; MOREIRA, A.; CASTRO, J.; ALENCAR, F.; PENTEADO, R. **Um Padrão para Requisitos Duplicados**. 6th Latin American Conference on Pattern Languages of Programming, Recife/PE, Brasil, 2007.

ROBINSON, W. N.; PAWLOWSKI, S. D.; VOLKOV, V. **Requirements Interaction Management**. ACM Computing Surveys, Vol. 35, No. 2. p. 132-190. 2003.

ROLLAND, C.; PRAKASH, N. **From conceptual modelling to requirements engineering.** Technical Report Series 99-11, CREWS. 1999.

ROMAN, G. **A Taxonomy of Current Issues in Requirements Engineering**. IEEE Computer Vol. 18, No. 4 Apr. pp:14-23. 1985.

SANT'ANNA, N. **Um ambiente integrado para apoio ao desenvolvimento e gestão de projetos de *software* para sistemas de controle de satélites**. São José dos Campos: INPE (INPE - 8306 - TDI/765). 2000.

SANTOS, E. B. **Uma proposta de métricas para avaliar modelos i***. Centro de Informática. Universidade Federal de Pernambuco. Recife/PE. 2008.

SANTOS, M.; CASTRO, J. B.; LENCASTRE, M.; FONSECA, D. **O uso do Framework NFR no Projeto de Banco de Dados Distribuído**. In: III Workshop de Engenharia de Requisitos, 2000, Rio de Janeiro. III Workshop de Engenharia de Requisitos, p. 209-229. 2000.

SILVA, L. F. **Uma Estratégia Orientada a Aspectos para Modelagem de Requisitos**. Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro-RJ. 2006.

SOMMERVILLE, I.; SAWYER, P. **Requirements engineering. A good practice guide**. Englad: John & Sons Ltda. pág. 391. 1997.

SOMMERVILLE, I.; SAWYER, P.; VILLER, S. **Viewpoint for requirements elicitation: a pratical approach**. ICRE'98 Third Internactional Conference on Requirements Engineering. Colorado Springs, USA. USA: IEEE CSP, Los Alamitos, CA. Proceedings. pág. 74-81. 1998.

SOMMERVILLE, I. **Software Engineering**, Ed. 6th, Addison- Wesley, 2000.

STARUML. **Ferramenta StarUML.** Disponível em http://staruml.sourceforge.net/en/. Acesso em: novembro de 2009.

TOVAL, A.; NICOLÁS, J.; MOROS, B.; GARCÍA, F. **Requirements Reuse for Improving Information Systems Security: A Practicioner's Approach**. Requirements Engineering Journal. Springer, 6(4): 205-219. 2002.

UDDIN, M. N.; MEZBAH-UL-ISLAM, M.; HAQUE, K. M. G. **Information description and discovery method using classification structures in Web**. Malaysian Journal of Library and Information Science. 2006.

USIRONO, C. H. **Tecnologia Workflow*:* O impacto de sua utilização nos processos de negócio**. Universidade de São Paulo. São Paulo-SP. 2003.

VILLEGAS, O. L.; LAGUNA, M. A. **Requirements Reuse for Software Development**. RE 01 Doctoral Workshop. in 5th - IEEE International Symposium on Requirements Engineering. Toronto, Canada, 27–31. 2001.

XAMPP. **Ferramenta Xampp.** Disponível em: http://www.apachefriends.org/en/xampp.html. Acesso em: setembro de 2009.

YU, E. **Modelling Strategic Relationships for Process Reengineering**. Graduate Department of Computer Science, University of Toronto, Toronto, Canada, 1995.

YU, Y.; LEITE, J. C. S. P.; MYLOPOULOS, J. **From goals to aspects: discovering aspects from requirements goal models**. Proceedings of Requirements Engineering Conference, 12th IEEE International (RE'04). Japan. pp. 38-47. 2004.