

LUCIANA MESQUITA BELLEZA

**MODELO PARA SUPORTAR A ATUALIZAÇÃO E
CONSISTÊNCIA DE REQUISITOS
EM PROCESSOS DE MANUTENÇÃO
DE SOFTWARE**

Dissertação apresentada como requisito parcial a obtenção do grau de Mestre em Ciência da Computação, pelo Programa de Pós-Graduação em Ciência da Computação da Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: PROF. DR. RICARDO MELO BASTOS

PORTO ALEGRE
MARÇO de 2009

Dados Internacionais de Catalogação na Publicação (CIP)

B442m Belleza, Luciana Mesquita
Modelo para suportar a atualização e consistência de
requisitos em processos de manutenção de software / Luciana
Mesquita Belleza. – Porto Alegre, 2009.
122 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Ricardo Melo Bastos.

1. Informática. 2. Engenharia de Software. 3. Software –
Manutenção. I. Bastos, Ricardo Melo. II. Título.

CDD 005.1


**Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS**



Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "**Modelo para Suportar a Atualização e Consistência de Requisitos em Processos de Manutenção de Software**", apresentada por Luciana Mesquita Belleza, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 18/03/09 pela Comissão Examinadora:



Prof. Dr. Ricardo Melo Bastos - PPGCC/PUCRS
Orientador



Prof. Dr. Marcelo Blois Ribeiro - PPGCC/PUCRS



Prof. Dr. Marcelo Soares Pimenta - UFRGS

Homologada em 27/09/11, conforme Ata No. 19/11 pela Comissão Coordenadora.



Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS

Campus Central

Av. Ipiranga, 6681 - P32 - sala 507 - CEP: 90619-900
Fone: (51) 3320-3611 - Fax (51) 3320-3621
E-mail: ppgcc@puccrs.br

Dedico ao meu amado Alex da Silva Corrêa,
pela paciência e amor.

AGRADECIMENTOS

Em primeiro lugar, quero agradecer a DEUS por estar sempre comigo e me dar força para completar este desafio de fazer um mestrado tendo que trabalhar mais de 40 horas por semana. Muito obrigada por tudo!

Ao amor da minha vida, Alex, que sempre esteve do meu lado me incentivando, me ajudando a seguir em frente nesta jornada. Muito obrigada pela tua paciência e compreensão, quando nos finais de semana eu tive que ficar fazendo o trabalho.

Ao professor Ricardo Melo Bastos, um orientador sempre presente, compreensivo e muito competente. Um exemplo de professor a ser seguido.

Ao meu colega e amigo Alberto que me incentivou a fazer o mestrado, ajudou a definir o tema desta dissertação e me acompanhou durante todo o primeiro ano do curso. Alberto, sua ajuda foi essencial para que eu realizasse este trabalho.

A minha família por compreender que eu precisava estar ausente em alguns eventos importantes.

Rodrigo Noll e professor Marcelo Blois, muito obrigada pelas suas contribuições neste trabalho.

Ao Convênio Dell/PUCRS pelo apoio financeiro e profissional para realização deste trabalho.

E, por fim, agradeço a todos aqueles que me incentivaram a deixar a monotonia de lado e sair em busca de novos desafios! Obrigada a todos vocês!

“A parte mais difícil de construir um sistema de software é decidir precisamente o que construir” (Frederick Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering")

RESUMO

A manutenção e evolução do *software* demanda um custo muito alto das organizações. Um dos motivos para este alto custo é a falta de documentação. Os requisitos representam um dos principais meios de documentação do *software*. Geralmente os requisitos não são atualizados depois do término do desenvolvimento do *software*, ou seja, não são atualizados durante a fase de manutenção. O objetivo desta pesquisa é propor uma solução para o problema de manter os requisitos de aplicações de *software* atualizados e consistentes ao longo de processos de manutenção. A solução consiste em um modelo de gerência de requisitos que suporta a atualização e consistência dos requisitos ao longo de processo de manutenção. Este modelo é constituído por um Modelo Conceitual, representando os conceitos envolvidos no problema e como eles devem estar relacionados para que seja possível alcançar o objetivo, regras de consistência, e um mecanismo de versionamento dos requisitos. Os resultados são demonstrados através de exemplos, ilustrando os diversos cenários possíveis, utilizando um protótipo desenvolvido a partir do modelo proposto. A principal contribuição deste trabalho é um modelo que auxilie a manter os requisitos de *software* atualizados e consistentes ao longo de processos de manutenção, além de auxiliar na análise de impacto das requisições de mudança.

Palavras-chave: Engenharia de Requisitos, Gerência de Requisitos, Requisitos, Casos de Uso, Manutenção de *Software*.

ABSTRACT

The maintenance and software evolution demands a high cost to the organizations. One of the reasons for this high cost is the lack of documentation. The requirements represent an important way of documenting the software. Very commonly the requirements are not updated after the end of the software development project. The requirements are not updated during the maintenance phase. The purpose of this research is to propose a solution to the problem of keeping the software application requirements up to date and consistent during the software maintenance process. The solution is represented by a requirements management model that supports the updating and consistency of the requirements during the maintenance. This model is formed by a Conceptual Model, that represents the concepts involved in the problem and how they are related to each other in order that it is possible to reach the purpose, consistency rules, and a process to track the versions of the requirements. The results are presented through examples, illustrating various possible scenarios, using the prototype developed based on the proposed model. The main contribution of this research is a model that helps to maintain the software requirements up to date and consistent along the maintenance process, besides to help on the impact analysis of change requests.

Keywords: *Requirements Engineering, Requirements Management, Requirements, Use Cases, Software Maintenance.*

LISTA DE FIGURAS

Figura 1. Exemplo de casos de uso relacionados por pré e pós-condições.	27
Figura 2. Ciclo de Vida do Software - Adaptado de [BEN 2000].	28
Figura 3. Ciclo de vida de mudança.	31
Figura 4. Estrutura do documento ERD.	38
Figura 6. Remoção da Pós-condição.	55
Figura 7. Remoção de um caso de uso relacionado por Pós-condição.	55
Figura 8. Remoção de um caso de uso relacionado por Inclusão.	56
Figura 9. Remoção do passo de um caso de uso relacionado por Inclusão.	56
Figura 10. Remoção de um caso de uso relacionado por Extensão.	57
Figura 11 – Remoção de um caso de uso relacionado por Generalização.	58
Figura 12 – Versionamento de Requisitos.	59
Figura 13. Sistema de Vendas.	62
Figura 14. Sistema de Pagamento.	63
Figura 15. Sistema de Compra de Fornecedores.	64
Figura 16. Relações entre os sistemas.	65
Figura 23. Identificação da pré-condição do caso de uso "Aceitar Proposta do Fornecedor e Efetuar Compra" .	72
Figura 24. Casos de Uso Adicionados pela Requisição de Mudança.	72
Figura 25. Casos de Uso afetados pela Requisição de Mudança "Adicionar funcionalidades para requisição e fechamento de compra com fornecedor" .	71
Figura 26. Requisição de Mudança "Usuário não precisa estar autenticado no sistema para submissão de pedido" .	72
Figura 27. Requisição de Mudança "Usuário não precisa estar autenticado no sistema para submissão de pedido" .	72
Figura 28. Requisição de Mudança "Usuário não precisa estar autenticado no sistema para submissão de pedido" atendida.	73
Figura 29. Requisições de Mudança que removem casos de uso.	74
Figura 30. Relações por pré e pós-condições do caso de Uso "Submeter Pedido" .	75
Figura 31. Edição caso de Uso "Submeter Pedido" .	76
Figura 32. Mensagem de erro na remoção da pós-condição do caso de Uso "Submeter Pedido" .	76
Figura 33. Mensagem de erro na remoção do caso de Uso "Submeter Pedido" .	77
Figura 34. Histórico de mudança da aplicação "Sistema de Cotação de Pedidos com Fornecedores" .	78
Figura 35. Histórico do Caso de Uso "Submeter Pedido" .	79
Figura 36. Detalhes das versões do Caso de Uso "Submeter Pedido" .	80
Figura 37. Exemplo de inclusão de caso de uso de outra aplicação.	81
Figura 38. Requisição de Mudança "atendida" .	81

LISTA DE TABELAS

Tabela 1 – Conteúdo da especificação de casos de uso (adaptada de [AND 2001]).....	24
--	----

LISTA DE SIGLAS

CR – *Change Request*

ERD – *Enhancement Request Document*

LSI – *Latent Semantic Indexing*

MVC - *Model View Control*

PU - *Processo Unificado*

SCM – *Software Configuration Management*

SRS – *Software Requirements Specification*

UML – *Unified Modelling Language*

SUMÁRIO

1.	INTRODUÇÃO	14
1.1	Análise de ferramentas de gerência de requisitos existentes no mercado	15
1.2	Questão de Pesquisa	16
1.3	Objetivos.....	16
1.3.1	Objetivo Geral	16
1.3.2	Objetivos Específicos	16
1.4	Organização do Volume	17
2.	REFERENCIAL TEÓRICO	19
2.1	Engenharia de Requisitos	19
2.2	Requisitos	19
2.3	Gerência de Requisitos	21
2.4	Casos de Uso	22
2.4.1	Especificação de Casos de Uso.....	23
2.4.2	Requisitos Funcionais como Casos de Uso	25
2.4.3	Relacionamentos entre Casos de Uso.....	25
2.5	Manutenção de Software	27
2.5.1	Ciclo de Vida do Software	27
2.5.2	Ciclo de Vida de Mudança.....	29
2.6	Considerações do capítulo	31
3.	ESTUDOS RELACIONADOS.....	32
3.1	SOMÉ (2007)	32
3.2	LUCIA, FASANO, OLIVETO e TORTORA (2007)	33
3.3	NOLL e BLOIS (2007)	34
3.4	MOHAN, XU, CAO e RAMESH (2008).....	35
3.5	Considerações do Capítulo	35
4.	MODELO PROPOSTO.....	37
4.1	Descrição do Problema	37
4.1.1	Estudo de caso exploratório.....	37
4.1.2	Identificação dos principais requisitos para resolver o problema	40
4.2	Proposta	41
4.2.1	Detalhes do Modelo Conceitual	44
4.2.2	Principais características do Modelo Conceitual.....	51
4.2.3	Regras de consistência.....	54
4.2.4	Versionamento dos requisitos.....	58
4.3	Considerações finais	59
5.	AValiação DO MODELO PROPOSTO	61
5.1	Protótipo	61
5.2	Descrição dos sistemas utilizados para exemplificação	61
5.2.1	Sistema de Vendas	62

5.2.2	Sistema de Pagamento.....	63
5.2.3	Sistema de Compra de Fornecedores	63
5.2.4	Visão das Relações entre os Sistemas.....	64
5.3	Demonstração dos resultados	65
5.3.1	Identificação dos requisitos afetados por uma requisição de mudança	66
5.3.2	Manutenção dos requisitos atualizados e consistentes conforme as requisições de mudanças	76
5.3.3	Manter a rastreabilidade das mudanças dos requisitos de aplicação.....	78
5.3.4	Reuso de Casos de Uso	81
5.4	Considerações do capítulo	83
6.	CONSIDERAÇÕES FINAIS.....	84
6.1	Contribuições.....	85
6.2	Limitações do Estudo	85
6.3	Trabalhos Futuros	86
	REFERÊNCIAS BIBLIOGRÁFICAS	87
	APÊNDICE I - Documentação do Protótipo.....	92
	APÊNDICE II – Especificação dos Casos de uso dos sistemas de exemplo	113
	APÊNDICE III – Telas do Protótipo	120

1. INTRODUÇÃO

A Engenharia de Requisitos (ER), uma subárea da Engenharia de Software, estuda o processo de definição dos requisitos que o software deverá atender. O processo de definição de requisitos é uma interface entre os desejos e necessidades dos clientes e a posterior implementação desses requisitos em forma de *software*. A engenharia de requisitos é um processo que engloba todas as atividades que contribuem para a produção de documentos de requisitos e sua manutenção ao longo do tempo.

A gerência de requisitos em processos de manutenção é uma atividade que acompanha a evolução dos requisitos ao longo do ciclo de vida de uma aplicação de *software* ou produto. Os requisitos da aplicação devem ser gerenciados desde seu nascimento e enquanto estiverem presentes na aplicação. Frequentemente estas aplicações passam por processos de manutenção que alteram seus requisitos. Estas alterações devem ser devidamente registradas na documentação da aplicação para possibilitar a rastreabilidade e garantir que os requisitos atuais da aplicação estejam atualizados, consistentes e disponíveis.

Organizações de desenvolvimento de *software* estão se dando conta que seu principal artefato é sua própria experiência e o conhecimento ganho a partir dela. Elas estão dando passos em direção a estabelecer bases de conhecimento corporativo que tornam este artefato tangível. Os requisitos são pontos chave para esta base de conhecimento. Portanto existe a necessidade de um movimento do foco de gerência de requisitos ao longo do ciclo de vida de um projeto para o gerenciamento dos requisitos como uma contribuição contínua para o conhecimento geral da organização [FIN 2000]. [ANQ 2007] chama a atenção para a importância da disseminação do conhecimento das aplicações no processo de manutenção de *software*.

A gerência de requisitos em processos de manutenção contínua é uma atividade que acompanha a evolução dos requisitos ao longo do ciclo de vida de uma aplicação ou produto. A fim de compreender as dificuldades relacionadas com a gerência dos requisitos em processos de manutenção, realizou-se um estudo de caso exploratório, detalhado na seção 4.1.1 deste trabalho, junto a uma equipe de Engenheiros de Requisitos em uma fábrica de software que pertence a uma empresa multinacional.

Esta pesquisa visa encontrar uma solução para a deficiência identificada na literatura, e ilustrada no caso particular da empresa, no suporte à manutenção dos requisitos de aplicações consistentes e atualizados ao longo do Ciclo de Vida da Aplicação.

1.1 Análise de ferramentas de gerência de requisitos existentes no mercado

Em [BEL 2007], a autora realizou um estudo comparativo das ferramentas de gerência de requisitos existentes no mercado com o objetivo de analisar como as mesmas suportam o gerenciamento dos requisitos ao longo do ciclo de vida das aplicações. Segue uma breve descrição dos resultados deste estudo.

Primeiramente gerou-se uma lista dos requisitos que uma ferramenta de gerência de requisitos deve atender para resolver o problema foco deste trabalho. Considerando estes requisitos, fez-se uma análise de três ferramentas do mercado: *RequisitePro*, *Compuware Optimal Trace Enterprise* e *DOORS*. A conclusão do estudo mostrou que nenhuma das três ferramentas atendem os seguintes requisitos:

- Gerenciamento de mudanças: A ferramenta deve fornecer a possibilidade de manipulação de CRs (*Change Requests* – Requisições de Mudança) formais. A ferramenta deve armazenar a informação de quais os requisitos foram afetados pela requisição de mudança e de que forma estes requisitos foram afetados (adicionados, removidos ou alterados);
- Gerenciamento dos requisitos da aplicação (as ferramentas analisadas são orientadas a projeto): Habilidade de manter requisitos de aplicação através dos projetos de manutenção; Habilidade de ligar requisitos de aplicação com requisições de mudanças através de relações como Adicionar, Alterar, Remover; Habilidade de atualizar os requisitos de aplicação de forma automática quando a implementação da requisição de mudança é entregue ao cliente da aplicação; Habilidade de exportar um documento de requisitos da aplicação;
- Histórico de cada requisito da aplicação: apresenta informação sobre qual projeto ou requisição de mudança o requisito foi criado, modificado, removido; quais foram os autores das modificações; em que data as modificações ocorreram.

A partir desta análise, pode-se observar dois problemas críticos:

- As ferramentas não estão preparadas para suportar os requisitos das aplicações ao longo da sua evolução. Uma vez que elas são orientadas a projeto, os requisitos

existem apenas enquanto o projeto estiver ativo. Uma vez que o projeto termine, os requisitos também deixam de existir. Portanto, para manter os requisitos das aplicações, mesmo após o término do projeto, é necessário se efetuar uma cópia dos requisitos gerados ou atualizados pelo projeto. Este tipo de procedimento não garante a consistência e atualização dos requisitos. É necessário que se mantenha um repositório centralizados com os requisitos da aplicação;

- As ferramentas não suportam as requisições de mudanças: É crucial que se tenha o histórico dos requisitos da aplicação desde sua origem até o atual momento. Muitas vezes uma requisição de mudança afeta mais de um requisito. É importante que a ferramenta ligue a requisição de mudança com os requisitos que foram afetados e informe de que forma os mesmos foram afetados.

Baseado nesta análise pode-se afirmar que as ferramentas do mercado não estão preparadas para gerenciar os requisitos de uma aplicação ao longo de sua evolução.

1.2 Questão de Pesquisa

Este trabalho tem como principal objetivo propor uma solução para garantir a consistência dos artefatos de requisitos de aplicações. Na maioria das vezes o documento de requisitos de uma aplicação é concebido no início do projeto que visa desenvolvê-lo e após o fim do projeto, ou mesmo durante o projeto, o documento com os requisitos é esquecido e se torna obsoleto.

Neste sentido, a questão de pesquisa deste estudo é a seguinte: “Como suportar a consistência e atualização de requisitos afetados por requisições de mudança para manutenção de *software*?”

1.3 Objetivos

Uma vez definida a questão de pesquisa, definiu-se o objetivo geral e os objetivos específicos deste trabalho, os quais são apresentados a seguir.

1.3.1 Objetivo Geral

Propor um modelo que suporte a consistência e atualização das documentações de requisitos de aplicações.

1.3.2 Objetivos Específicos

- Embasamento teórico e contextualização do problema na literatura;
- Análise das dificuldades relacionadas com a gerência de requisitos de aplicações durante a fase de manutenção;
- Aprofundar o estudo teórico sobre as arquiteturas das ferramentas existentes no mercado;
- Especificar um modelo de arquitetura que atenda às necessidades identificadas;
- Desenvolvimento de uma instância do modelo de arquitetura especificado através de um protótipo de ferramenta de gerência de requisitos;
- Avaliar o uso do protótipo a partir de exemplos hipotéticos.

1.4 Organização do Volume

Este volume está organizado em sete capítulos. O capítulo 1 corresponde a esta introdução.

No capítulo 2 é apresentado o referencial teórico desta pesquisa, envolvendo os principais conceitos e áreas do estudo: engenharia de requisitos, requisitos, gerência de requisitos, casos de uso e manutenção de *software*.

O capítulo 3 apresenta os estudos relacionados com o tema de pesquisa deste trabalho. Foram selecionados artigos sobre estudos em áreas relacionadas ou complementares a pesquisa aqui apresentada. Durante este capítulo, são feitas considerações sobre estes trabalhos.

No capítulo 4 o problema de pesquisa é detalhado e o modelo de gerência de requisitos durante a manutenção do *software* é apresentado. No modelo proposto é apresentado o Modelo Conceitual com a representação dos principais conceitos envolvidos no problema e como estes se relacionam de forma a auxiliar no suporte a consistência e atualização dos requisitos. Neste capítulo também são apresentadas regras de consistência dos casos de uso e o mecanismo de versionamento dos mesmos.

O capítulo 5 descreve a avaliação do modelo proposto através do protótipo desenvolvido a partir do modelo. Para a demonstração dos resultados criou-se alguns exemplos hipotéticos de sistemas.

O capítulo 6 ilustra as considerações finais desta pesquisa. São descritas as contribuições deste estudo, bem como suas limitações e trabalhos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo descreve o referencial teórico deste trabalho apresentando os principais conceitos envolvidos com o tema da pesquisa: engenharia de requisitos, requisitos, gerência de requisitos, casos de uso e manutenção de *software*.

2.1 Engenharia de Requisitos

Engenharia de Requisitos (ER) é um conjunto de atividades voltadas a identificar e comunicar os objetivos de um sistema de *software*, e o contexto no qual o mesmo é usado. Portanto, ER atua como uma ponte entre as necessidades de usuários, clientes e outras partes do mundo real afetadas pelo *software* e as capacidades e oportunidades disponibilizadas pelas tecnologias de software [EAS 2004].

Thayer e Dorfman [THA 1990] definem engenharia de requisitos como a ciência e disciplina preocupada com a análise e documentação dos requisitos, incluindo análise das necessidades e análise e especificação dos requisitos. Além disso, a engenharia de requisitos fornece mecanismos apropriados para facilitar as atividades de análise, documentação e verificação.

Desde a década de 70, problemas com ER persistem como uma causa chave para a ineficiência e falhas de projetos de *software*. Melhorias no processo de ER têm potencial de reduzir custos e tempo de desenvolvimento, e aumentar a qualidade do *software* [SAD 2007].

A engenharia de requisitos (no contexto da engenharia de *software*) é um processo que engloba todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo.

Segundo [AUR 2003], a ER é ambas uma atividade organizacional e uma atividade de projeto. É uma atividade organizacional quando o enfoque é definir qual o conjunto de requisitos que são apropriados para o *software*, e quais requisitos serão entregues. É uma atividade de projeto quando este conjunto de requisitos é direcionado para a equipe que irá desenvolver o *software*. Este dualismo envolve um conjunto de decisões que tem que atender tanto as necessidades da organização, assim como as necessidades do projeto.

2.2 Requisitos

Segundo [DOR 1990], requisitos de *software* podem ter duas definições:

- Uma condição ou capacidade do sistema de *software* necessária para o usuário resolver um problema ou alcançar um objetivo;
- Uma condição ou capacidade que deve ser encontrada ou possuída por um sistema ou componente do sistema de *software* para satisfazer um contrato, padrão, especificação ou outro documento imposto formalmente.

Alguns autores definem um requisito como qualquer função ou característica necessária a um sistema - os comportamentos quantificáveis e verificáveis que um sistema deve ter, as restrições que deve atender ou outras propriedades que devem ser fornecidas, de forma a satisfazer os objetivos das organizações e resolver um conjunto de problemas. Outros, afirmam que os requisitos de um sistema definem o que o sistema deve fazer e as circunstâncias sobre as quais deve operar. Em outras palavras, os requisitos definem os serviços que o sistema deve fornecer e dispõem sobre as restrições à operação do mesmo.

Diversos autores [THA 1990][NUS 2000][KRU 2000][LEF 2000][SOM 2005] dividem os requisitos de software em dois tipos: funcionais (o que o software deve fazer) e não-funcionais (como o software se comporta em relação a alguns atributos observáveis).

- Requisitos funcionais devem definir as ações fundamentais que devem ser tomadas pelo *software* na aceitação e processamento das entradas, e no processamento e transformação das saídas [IEE 1998]. Requisitos funcionais devem ser descritos de forma completa e consistente, onde a completeza significa que todas as funções requeridas pelo usuário devem estar descritas, e consistência significa que os requisitos não devem ter definições contraditórias [SOM 2005]. Segundo [LEF 2000] requisitos funcionais podem ser representados por uma simples sentença ou na forma de casos de uso.
- Requisitos não-funcionais são requisitos que especificam as propriedades de um sistema de *software*, tais como restrições de ambiente e implementação, desempenho, dependência de plataformas, manutenibilidade, extensibilidade, e confiabilidade [JAC 1998]. Segundo [LEF 2000], requisitos não-funcionais são tipicamente documentados em linguagem natural. A falha em se cumprir um requisito não-funcional pode ser muito pior do que a falha em se cumprir um requisito funcional. Muitos requisitos não-funcionais dizem respeito ao sistema como um todo. Enquanto a falha em um

requisito funcional pode degradar parte do sistema, a falha em um requisito não-funcional pode tornar todo o sistema inútil [SOM 2003].

2.3 Gerência de Requisitos

Sommerville [SOM 2005] considera a atividade de Gerência de requisitos como uma das principais atividades do processo de engenharia de requisitos.

Atualmente tem-se convicção de que mudanças em requisitos ao longo do processo de desenvolvimento de *software* fazem parte do processo. A gerência de requisitos está associada ao processo de acompanhar a evolução dos requisitos ao longo do processo de desenvolvimento [SAY 2005].

De acordo com [KOT 1998], as principais responsabilidades da gerência de requisitos são:

- Gerenciar mudanças em requisitos já aprovados;
- Gerenciar os relacionamentos entre os requisitos;
- Gerenciar as dependências entre o documento de requisitos e outros documentos produzidos durante o processo de desenvolvimento do sistema.

Os requisitos evoluem devido a mudanças no ambiente do sistema e devido à evolução da compreensão dos *stakeholders* de suas reais necessidades. Novos requisitos surgem e requisitos existentes mudam em qualquer estágio do processo de desenvolvimento do sistema. Isto pode causar sérios problemas para os desenvolvedores do sistema. Para minimizar estas dificuldades é necessário controlar e documentar estas mudanças. O impacto das mudanças deve ser avaliado e, se as mudanças forem aceitas, o projeto e a implementação do sistema também devem ser modificados [ESP 2006].

Indispensável à tarefa de gerência de requisitos é a disponibilidade de facilidades de rastreamento [SAY 2005]. Um requisito é rastreável se é possível descobrir quem sugeriu o requisito (a fonte), por que o requisito existe (razões e motivações), que outros requisitos estão relacionados a ele (dependência entre requisitos) e como o requisito se relaciona com outras informações tais como arquitetura do sistema, implementação e documentação do usuário [SOM 1998].

Algumas pesquisas apontam que os principais motivos que levam a atrasos na entrega, aumento de custos, e ineficiência de um software são todos relacionados com práticas de gerência de requisitos [LEF 2000].

Existem diversos estudos relacionados à gerência de requisitos. Por exemplo, [DOU 2008] apresenta métodos para auxiliar os *stakeholders* não-técnicos envolvidos com a engenharia de requisitos a gerenciarem mudanças nos requisitos dentro de um domínio específico. Segundo o autor, uma parte importante de gerenciar requisitos que evoluem ao longo do tempo é manter a ordem temporal das mudanças e suportar a rastreabilidade das modificações.

2.4 Casos de Uso

Cenários podem ser úteis para facilitar a comunicação com os usuários. Os usuários geralmente acham mais fácil relacionar exemplos da vida real do que descrições abstratas. Os usuários podem compreender e criticar um cenário de como poderiam interagir com um sistema de *software*. Os engenheiros de requisitos podem utilizar as informações obtidas com essa discussão para formular os requisitos reais do sistema. A obtenção de requisitos com base em cenários pode ser realizada informalmente, e o engenheiro de requisitos trabalha com os *stakeholders* para identificar cenários e captar detalhes desses cenários [SOM 2003].

Os casos de uso são técnicas baseadas em cenários para a obtenção de requisitos utilizando uma abordagem estruturada. Eles são atualmente uma característica fundamental no uso da UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada) [JAC 1998]. Um caso de uso engloba um conjunto de cenários em que cada cenário é um traço isolado dentro do caso de uso [SOM 2003].

Casos de uso descrevem as interações entre o usuário e o sistema, focando no que o sistema faz para o usuário. O modelo de casos de uso descreve na totalidade o comportamento funcional do sistema [LEF 2000].

De acordo com a especificação UML [OMG 01], um caso de uso consiste na especificação de uma seqüência de ações, incluindo variações, que o sistema pode executar, interagindo com atores do sistema. Um caso de uso descreve uma porção do comportamento do sistema sem revelar a estrutura interna do sistema. Sendo assim casos de uso são úteis para a captura e documentação de requisitos externos. Eles também são ideais para a validação de requisitos através de protótipos. Vários processos de desenvolvimento de sistemas de *software* incluindo o Método Unificado de Desenvolvimento de *Software* (*Unified Software*

Development Process) recomendam a utilização de casos de uso para a documentação dos requisitos dos usuários [SOMé 2006].

2.4.1 Especificação de Casos de Uso

Um padrão de especificação de casos de uso é recomendado porque uma estrutura pré-definida auxilia aos engenheiros de requisitos a identificarem e incluírem importantes elementos em cada caso de uso [JAC 1998].

Existem diferentes padrões e guias para a especificação de requisitos na literatura. Exemplos podem ser encontrados em [JAC 1998], [COC 2001], [SCH 1998], [KRU 2000] e [LEF 2000]. O conteúdo encontrado com mais frequência é mostrado na Tabela 1.

Seguem abaixo as propriedades dos casos de uso que serão adotadas neste trabalho, bem como descrição detalhada de cada uma:

- **Objetivo:** Esta informação deve conter o motivo pelo qual o requisito é necessário para atender um objetivo de negócio [DUR 1999]. Apesar de não ser identificada com uma propriedade freqüente em *templates* de casos de uso, adotou-se esta propriedade como uma informação adicional que deve auxiliar os *stakeholders* a compreender o caso de uso;
- **Título:** Contém o nome do caso de uso. Cada caso de uso deve ter um nome único que sugira o seu objetivo;
- **Atores:** Consiste na lista de atores que interagem com o caso de uso. São entidades externas (usuários ou outros sistemas) que interagem com o caso de uso para atingir um determinado objetivo. Não se deve confundir ator com usuário, uma vez que um usuário pode realizar diferentes papéis no uso de um sistema, enquanto que um ator representa apenas um papel;
- **Pré-condições:** Listagem das condições que se devem ser atendidas antes de iniciar o caso de uso;
- **Pós-condições:** Descrevem o que deve ser verdadeiro quando da bem-sucedida conclusão do caso de uso – seja o cenário principal ou algum outro caminho alternativo. É uma garantia de sucesso do caso de uso que deve atender às necessidades de todos os *stakeholders*;
- **Importância:** Esta propriedade indica o quanto o requisito é importante para os

TABELA 1 . CONTEÚDO DA ESPECIFICAÇÃO DE CASOS DE USO (ADAPTADA DE [AND 2001]).

Propriedade	[JAC 1998]	[COC 2001]	[SCH 1998]	[KRU 2000]	[LEF 2000]
Título	X	X	X	X	X
Atores		X	X	X	X
Pré-condições	X	X	X		X
Fluxo básico de eventos	X	X	X	X	X
Pontos de extensão	X	X	X	X	X
Fluxos alternativos	X	X	X	X	X
Pós-condições	X	X	X	X	X

stakeholders. Para esta propriedade podem ser atribuídos valores numéricos ou algumas expressões como “vital”, “importante” ou “interessante se ter” [DUR 1999];

- Comentários: O propósito desta propriedade é permitir ao engenheiro de requisitos informar outras informações que não se enquadram nas outras propriedades;
- Frequência: Indica o número de vezes que se espera que o caso de uso execute. Apesar da frequência não ser um requisito, é uma informação importante para os desenvolvedores do *software*;
- Fluxo básico de eventos: Descreve o fluxo de eventos ou cenário principal que o caso de uso deve realizar para atingir o objetivo, ou seja, descreve o cenário típico de sucesso que satisfaz o objetivo dos *stakeholders*. Usualmente é descrito através de uma seqüência de eventos numerados;
- Fluxos alternativos: Representam fluxo de eventos ou cenários que são ramificações do fluxo básico de eventos (cenário principal);
- Requisitos Não-Funcionais: A especificação de caso de uso também pode conter a descrição de requisitos não-funcionais relacionados com o caso de uso (requisitos especiais). Entre esses requisitos podem-se ter requisitos de qualidade, tais como

desempenho, confiabilidade, usabilidade e restrições de projeto (frequentemente relativas a dispositivos de E / S) que foram impostas ou consideradas prováveis [LAR 2004].

2.4.2 Requisitos Funcionais como Casos de Uso

Cockburn [COC 2005] define que casos de uso representam requisitos que especificam o que o sistema de *software* deve fazer. Porém eles não são todos os requisitos, já que não especificam interfaces externas, formatos de dados, regras de negócio e fórmulas complexas. Ou seja, segundo o autor, os casos de uso constituem somente uma fração de todos os requisitos do sistema.

O trabalho de [DUR 1999] propõe uma solução para o problema da representação dos requisitos dos clientes de forma que possam ser entendidos tanto pelos engenheiros de software como também por profissionais não envolvidos com computação e usuários finais. Para tanto são apresentados padrões de representação dos requisitos funcionais e não funcionais. O padrão para requisitos funcionais apresentado está no formato de casos de uso. Ou seja, os autores consideram casos de uso como forma de descrever, de forma padronizada, os requisitos funcionais de um sistema.

Segundo Somé [SOMé 2006], casos de uso, que descrevem as possíveis interações envolvendo o sistema e seu ambiente, vêm cada vez mais sendo adotados como meio de elicitação e análise dos requisitos funcionais. Neste trabalho o autor apresenta um método suportado por uma ferramenta de engenharia de requisitos baseada em casos de uso. O método inclui a formalização dos casos de uso, uma forma restritiva de descrição dos casos de uso em linguagem natural, e a derivação de uma especificação executável, bem como um ambiente de simulação dos casos de uso.

[DAR 2003], [DOU 2008] e [LAR 2007] também identificam casos de uso como forma de descrever os requisitos funcionais de um sistema.

2.4.3 Relacionamentos entre Casos de Uso

A UML [UML 2007] define os relacionamentos Inclusão (representado por «*include*»), Extensão (representado por «*extend*») e Generalização entre casos de uso.

O relacionamento de **Inclusão** define que um caso de uso contém o comportamento definido em outro caso de uso. O caso de uso que inclui pode depender apenas do resultado (valor) do caso de uso incluído. Este valor é obtido como um resultado da execução do caso

de uso incluído. A execução do caso de uso incluído não é opcional, ou seja, é sempre requerida pelo caso de uso que inclui. O relacionamento de inclusão permite a composição hierárquica e o reuso dos casos de uso.

O relacionamento de **Extensão** define que o comportamento de um caso de uso pode ser estendido pelo comportamento de outro caso de uso. O caso de uso estendido é definido independentemente do caso de uso que estende, ou seja, ele existe independente do caso de uso que o estende. Por outro lado, o caso de uso que estende pode definir comportamento que não necessariamente faça sentido por si só. Neste caso, o caso de uso que estende define um conjunto de comportamento modular que complementa a execução do caso de uso estendido quando determinada condição é encontrada.

O relacionamento **Generalização** é usado quando um caso de uso especializa outro caso de uso mais geral, ou seja, diz-se que o caso de uso é “um tipo de” outro caso de uso [COC01]. Segundo a UML [UML 2007] um relacionamento de generalização entre casos de uso implica que o caso de uso filho contém todos os atributos, seqüências de comportamento, e pontos de extensão definidos no caso de uso pai, e participa de todos os seus relacionamentos.

Além dos relacionamentos entre casos de uso definidos pela UML [UML 2007], será considerado neste trabalho o relacionamento entre casos de uso por suas pré-condições e pós-condições.

Pré-condições: são condições que devem ser verdadeiras quando a execução do caso de uso for invocada [UML 2007].

Pós-condições: são condições que serão verdadeiras quando o caso de uso completar a sua execução com sucesso, assumindo que suas pré-condições foram satisfeitas. Pós-condições são particularmente importantes quando o caso de uso executado leva o sistema para um determinado estado que caracterize uma pré-condição para outro caso de uso [COC 2005].

Somé [SOMé 2007] define que pré e pós-condições são especificações implícitas da seqüência de execução dos casos de uso. Pré-condições e pós-condições permitem especificar que casos de uso devem preceder um dado caso de uso e quais os casos de uso que devem suceder (executar após) o caso de uso. Mais detalhes sobre o trabalho [SOMé 2007] pode ser encontrado na seção 3.1 que descreve os estudos relacionados.

Larman [LAR 2007] descreve os Contratos de Operação como artefatos relacionados com a análise orientada a objeto. Os contratos de operação usam pré e pós-condições para descrever modificações detalhadas em objetos do Modelo de Domínio, como resultado de uma operação do sistema. Segundo o autor, as pós-condições descrevem modificações no estado dos objetos do Modelo de Domínio. Tais modificações no estado do Modelo do Domínio incluem instâncias criadas, associações formadas ou desfeitas e atributos modificados.

A Figura 1 mostra um exemplo de dois casos de uso que estão relacionados por pré e pós-condições. No exemplo, a pós-condição do caso de uso “Entrar no Sistema” é pré-condição do caso de uso “Submeter Pedido”.

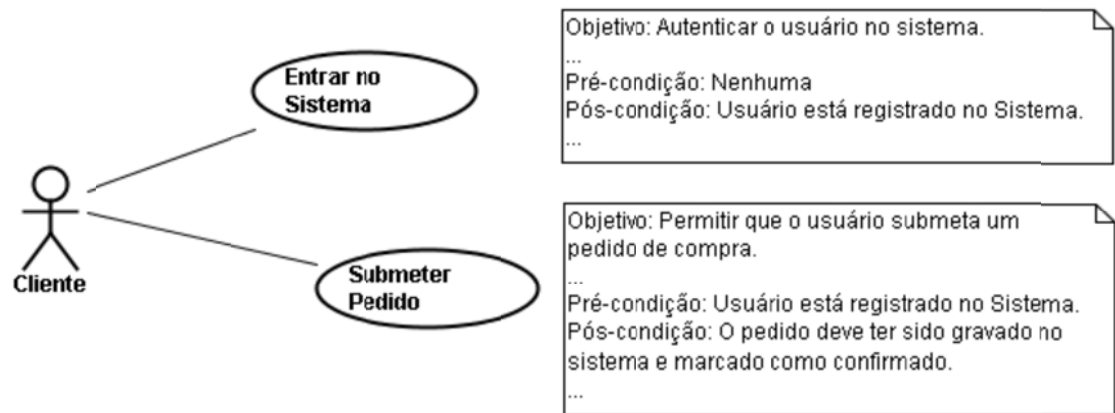


Figura 1. Exemplo de casos de uso relacionados por pré e pós-condições.

Mais exemplos de casos de uso relacionados por pré e pós-condições, conforme proposto neste trabalho, podem ser observados na seção 5.2.4.

2.5 Manutenção de Software

De acordo com [BEN 2000] a manutenção e evolução do software caracterizam-se por demandarem custo muito alto das organizações e também pela baixa velocidade de resposta (implementação de mudanças).

2.5.1 Ciclo de Vida do Software

[BEN 2000] apresenta um Modelo de fases do Ciclo de Vida do Software conforme mostra a Figura 2. A principal contribuição deste modelo é separar a fase de manutenção nos estágios Evolução, Serviços e Descontinuação.

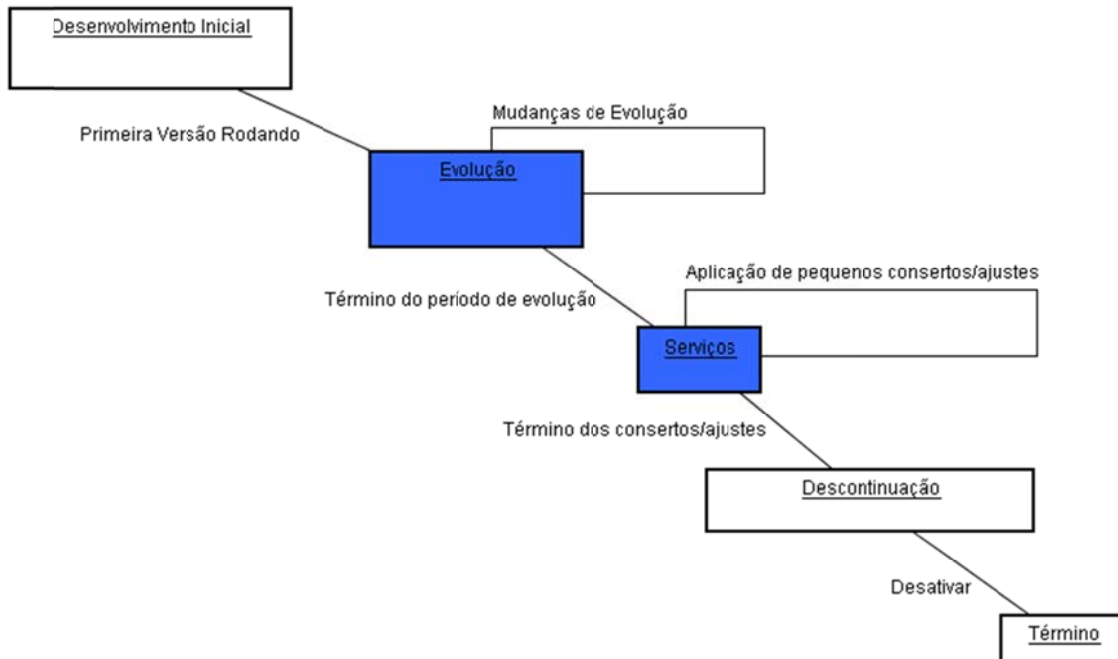


Figura 2. Ciclo de Vida do Software - Adaptado de [BEN 2000].

Desenvolvimento Inicial: Nesta fase a primeira versão do software é desenvolvida. Esta primeira versão pode não conter todas as funcionalidades, mas já tem uma arquitetura que persistirá pelo resto do ciclo de vida da aplicação. Um outro importante resultado do desenvolvimento inicial é o conhecimento da aplicação: conhecimento do domínio, dos requisitos de usuário, a função da aplicação no processo de negócio, as soluções e algoritmos utilizados, os formatos de dados, pontos fortes e fracos da arquitetura, ambiente de operação da aplicação, etc. Este conhecimento é um pré-requisito para a subsequente fase Evolução.

Evolução: Esta fase só inicia quando o desenvolvimento inicial foi bem sucedido. O objetivo desta fase é adaptar a aplicação para qualquer mudança nos requisitos do usuário e no ambiente operacional. A evolução também corrige as falhas da aplicação identificadas a partir do aprendizado do desenvolvedor e do usuário, no qual requisitos mais acurados são baseados na experiência passada com a aplicação. Em termos de negócio, a aplicação está evoluindo porque ela é bem sucedida no mercado: ela é lucrativa, existe forte demanda dos usuários sobre a mesma, a atmosfera de desenvolvimento é vibrante e positiva, e a organização suporta a aplicação. Resumindo, o retorno do investimento na aplicação é excelente. A arquitetura e o conhecimento da aplicação são responsáveis por tornar a evolução possível. Elas permitem que a equipe faça mudanças substanciais no software sem danificar a sua integridade. Uma

vez que um ou outro aspecto desaparece, o programa não é mais passível de evolução e entra no estágio de Serviço.

Serviço: Durante esta fase, somente pequenas mudanças táticas (concertos, ajustes) são possíveis. Para a organização, quando a aplicação entra neste estágio, é porque ela não é mais um produto essencial e o custo-benefício de se efetuar mudanças é baixo.

Descontinuação: Esta fase se dá quando nenhum serviço de manutenção (fase de Serviço) é necessário, mas a aplicação ainda está em produção. Os usuários precisam conviver com deficiências e limitações conhecidas.

Término: Na fase de término o software é desativado e os seus usuários são direcionados a outro software substituto.

As fases de Evolução e Serviço, que estão destacadas na Figura 2, são foco principal deste trabalho. Em ambas as fases a mudança da aplicação é a operação principal, com a única diferença que na fase de Evolução as mudanças tendem a ser mais complexas do que na de Serviço.

2.5.2 Ciclo de Vida de Mudança

Em [BEN 2000] também é apresentado o Ciclo de Vida de Mudança, conforme ilustrado na Figura 3, com as seguintes etapas:

- **Requisição de mudança:** Na maioria das vezes é originada dos usuários do sistema e pode ter a forma de um relato de um problema (*bug*) ou a requisição por uma funcionalidade nova. Normalmente é expressa em termos que representam conceitos do domínio da aplicação, por exemplo: “Adicione uma nova funcionalidade ao sistema de registro do estudante em cursos que impeça estudantes cujo registro esteja no estado retido não possam se registrar”.
- **Planejamento da mudança:** Esta atividade é compreendida por duas sub-atividades, são elas:
 - **Compreensão do software:** A compreensão do software é um pré-requisito para a mudança e tem sido tópico de diversas pesquisas. Relatos mostram que esta fase consome mais do que a metade dos recursos de manutenção [FJE 1982]. Uma substancial parte da compreensão do software é a localização dos conceitos do domínio da aplicação no código. Como no exemplo do registro do estudante em cursos, encontrar onde no código se encontra o “registro de estudante em cursos”;

- **Análise de impacto da mudança:** É a atividade pela qual os responsáveis pela aplicação avaliam a extensão da mudança, isto é, identificam os componentes que serão afetados pela mudança. A análise de impacto indica quanto custosa será a mudança a fim de indicar a viabilidade da mesma.
- **Implementação da mudança:** Consiste na implementação da mudança nos componentes de software identificados previamente. Pode acontecer de após modificar um determinado componente, este não se relacionar mais adequadamente com outros componentes, então estes outros componentes também devem ser alterados. Isso se chama propagação da mudança [YAU 1978] [RAJ 2000].
- **Verificação e validação da mudança:** Consiste na verificação se a requisição de mudança foi atendida como esperado.
- **Re-documentação:** Atualização da documentação do *software*. Se não existe documentação do software ou se ela é incompleta, o fim do ciclo de mudança é o momento de registrar a compreensão adquirida durante a mudança. Considerando o alto custo para compreensão do software, o registro do conhecimento adquirido é valioso. Mas na prática, ao término da mudança, os envolvidos na mudança do software tendem a voltar sua atenção a coisas novas em vez de atualizar a documentação. Por este motivo existem estudos, como apresentado em [RAJ 1999], que propõem a documentação incremental ao longo do processo de mudança. Segundo este método, depois de certo período de contínuas mudanças, uma documentação significativa tende a ser gerada.

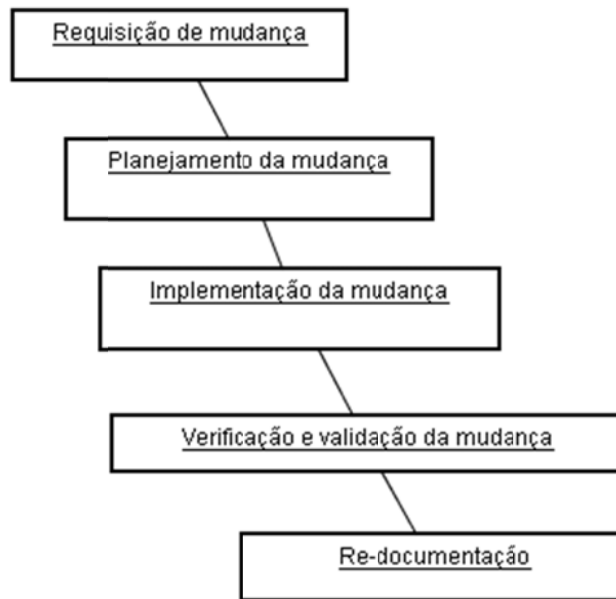


Figura 3. Ciclo de vida de mudança.

2.6 Considerações do capítulo

Este capítulo apresentou a fundamentação teórica dos principais conceitos envolvidos neste trabalho. São eles: engenharia de requisitos; requisitos; gerência de requisitos; casos de uso e manutenção de software.

Dos casos de uso, além da sua definição na literatura demonstrou-se que vários autores os consideram como uma forma de representar os requisitos funcionais. Os possíveis relacionamentos entre os casos de uso também foram apresentados.

Em relação à manutenção do *software* apresentou-se o ciclo de vida do *software*, dando ênfase para as fases de evolução e serviço, pois é quando acontecem as requisições de mudanças. O ciclo de vida da requisição de mudança também foi apresentado.

No próximo capítulo serão apresentados alguns trabalhos relacionados ao tema de pesquisa.

3. ESTUDOS RELACIONADOS

Após a pesquisa bibliográfica foram encontrados alguns trabalhos relacionados ao tema de pesquisa apresentado nesta dissertação. Este capítulo apresenta uma breve descrição dos mesmos, seguido das características relevantes para este trabalho.

3.1 SOMÉ (2007)

Somé propõe em [SOMé 2007] uma formalização da representação textual dos casos de uso. Segundo o autor, o fato dos casos de uso serem intuitivos, centralizados no usuário e textuais é uma das razões para o seu sucesso. Porém um certo nível de formalização é necessário para automatizar o desenvolvimento, incluindo a modelagem, verificação e validação, dos sistemas que se baseiam nos mesmos.

No nível sintático, um meta-modelo em UML e uma forma restringida de linguagem natural são definidas para a descrição dos casos de uso. A semântica de execução é proposta como um conjunto de Regras de Mapeamento (*Mapping Rules*) dos casos de uso bem-formados para as redes de Petri Básica.

O trabalho de Somé contribui com este trabalho de duas maneiras. A representação da descrição dos casos de uso através de um meta-modelo auxiliou na definição dos elementos que constituem o caso de uso bem como na sua representação no modelo proposto, conforme descrito na seção 2.4.3.

Outra contribuição deste trabalho é em relação ao relacionamento entre os casos de uso por pré e pós-condições. Somé cita o seguinte: “Pré-condições e pós-condições são especificações implícitas das condições de seqüência dos casos de uso. A pré-condição de um caso de uso é uma condição que precisa ser atendida até a execução do caso de uso, enquanto que a pós-condição é uma condição que é atendida no final da execução do caso de uso.”

Apesar de citar que as pré e pós-condições indicam a seqüência de execução dos casos de uso, Somé [SOMé 2007] não propõe o relacionamento direto por pré-condições e pós-condições entre os casos de uso, como é apresentado nesta dissertação. O autor propõe um relacionamento indireto que pode estar baseado nas pré-condições e pós-condições. Este relacionamento está representado pelas listas “*UseCaseEnabling*” (Casos de Uso Habilitados ou que procedem o caso de uso atual) e “*FollowList*” (Casos de Uso que Precedem o caso de uso atual).

3.2 LUCIA, FASANO, OLIVETO e TORTORA (2007)

Rastreabilidade dos artefatos de *software* é a habilidade de descrever e seguir a vida de um artefato (requisitos, código, testes, modelos, relatórios, planos, etc) desenvolvido durante o ciclo de vida do *software*, em ambas as direções (dos requisitos ao código e vice-versa) [GOT 1994].

A rastreabilidade pode fornecer importantes informações no desenvolvimento e evolução do *software* auxiliando na compreensão do *software*, na análise de impacto, e reuso de *softwares* existentes [PAL 2000].

A principal deficiência de sistemas de gerência de artefatos de *software* é a falta de geração automática ou semi-automática de ligações de rastreabilidade e manutenção destas ligações. Neste trabalho o autor propõe a integração de um sistema de gerência de artefato conhecido como ADAMS (*ADvanced Artefact Management System*), proposto em [LUC 2004], com uma ferramenta de recuperação de rastreabilidade baseada numa técnica de Recuperação de Informação (IR – *Information Retrieval*), denominada LSI (*Latent Semantic Indexing*).

As ligações de rastreabilidade armazenadas no ADAMS são úteis para análise de impacto e manutenção durante a evolução do software notificando os engenheiros que um artefato tem que ser modificado por consequência de mudanças aplicadas com artefatos que eles dependem. O sistema ADAMS auxilia na manutenção dos artefatos consistentes através das ligações de rastreabilidade entre os mesmos. Quando uma mudança afeta um artefato (por exemplo um caso de uso), existe a possibilidade de afetar os artefatos dependentes desse primeiro identificado. A representação da dependência entre os artefatos se dá através das ligações de rastreabilidade.

Seguem abaixo algumas conclusões identificadas pelo autor após a avaliação da ferramenta:

- O uso da ferramenta de recuperação da rastreabilidade reduz significativamente o tempo necessário para a identificação das ligações se comparado com o método manual;
- Métodos de Recuperação de Informação (IR – *Information Retrieval*) fornecem um suporte útil na identificação das ligações de rastreabilidade, mas não conseguem identificar todas as ligações existentes: A limitação da recuperação da rastreabilidade através de IR é que estes métodos não podem auxiliar na identificação de todos as

ligações corretas, forçando o engenheiro de *software* a analisar e descartar um alto número de falso positivos;

- O método incremental de recuperação da rastreabilidade foi identificado como melhor do que o método de identificação em uma única vez;
- Métodos baseados em IR pode auxiliar na identificação de problemas de qualidade da descrição textual de artefatos de software.

Conforme pode ser observado, a proposta do trabalho de [LUC 2007] é bastante similar à proposta desta dissertação, pois também propõe uma solução para suportar a evolução do *software*. A diferença principal é que a solução apresentada [LUC 2007] propõe auxiliar na evolução do *software* a partir da rastreabilidade entre todos os artefatos do *software*, tendo o suporte de uma ferramenta de gerência de artefatos e um método para geração automática das ligações de rastreabilidade. Enquanto que o trabalho que está sendo apresentado se propõe a suportar a evolução dos requisitos do *software* através de ligações de rastreabilidade entre as requisições de mudanças e os requisitos (funcionais, representados por casos de uso, e não funcionais), ligações entre os casos de uso (pelos relacionamentos propostos por [UML 2007] e [SOMé 2006]), e através do histórico dos requisitos ao longo do ciclo de vida do *software*.

Enfim, este trabalho é menos abrangente no sentido que não apresenta uma solução que abranja todos os artefatos do *software*, porém é mais específico e, por consequência, mais detalhista no suporte aos artefatos de requisitos.

3.3 NOLL e BLOIS (2007)

Este trabalho propõe a integração de ontologias no Processo Unificado (PU) a fim de fornecer rastreabilidade baseada em conceitos ao longo do ciclo de vida do software. Este método permite a integração de diferentes modelos do sistema de software incluindo negócio, requisitos, modelos de análise e de projeto. Para auxiliar os projetistas na criação da ontologia e ligação dos conceitos dos artefatos o autor disponibiliza uma ferramenta integrada com um modelador UML.

A principal diferença do trabalho de Noll [NOL 2007] em relação a este trabalho é que ele apresenta uma proposta de rastreabilidade dos artefatos do *software*, mas não apresenta uma proposta para manter a consistência e o controle de versão dos mesmos. O controle de versão é essencial para que seja possível se ter o histórico da evolução do *software*. Outra

diferença é que neste trabalho o autor propõe a rastreabilidade de todos os artefatos, enquanto que neste trabalho o enfoque é apenas artefatos de requisitos.

3.4 MOHAN, XU, CAO e RAMESH (2008)

Gerência de configuração do *software* ou *Software Configuration Management (SCM)* e rastreabilidade são duas práticas importantes no desenvolvimento do *software* que suporta a evolução do sistema e o controle de mudanças. SCM auxilia a gerenciar a evolução dos artefatos de *software* e suas documentações, enquanto que a rastreabilidade auxilia a gerenciar o conhecimento sobre o processo do desenvolvimento dos artefatos. Neste trabalho é apresentada uma solução que integra a rastreabilidade e SCM a fim de suportar o gerenciamento de mudanças durante o desenvolvimento e evolução dos artefatos de *software*. É apresentada uma ferramenta (chamada *Tracer*) que suporta a integração destas duas práticas. *Tracer* se caracteriza por uma ferramenta de rastreabilidade integrada com o *MS Visual SourceSafe*.

O trabalho de [MOH 2008] também apresenta uma proposta de solução de escopo mais abrangente do que esta dissertação, pois propõe uma solução que suporta a gerência de todos os artefatos do *software* ao longo de sua evolução. No entanto, o trabalho desta dissertação, por ter o foco apenas em requisitos, apresenta uma solução mais aprofundada para suportar a consistência e atualização dos requisitos do *software* ao longo de sua evolução. Em [MOH 2008] os casos de uso são armazenados como um único elemento, enquanto que neste trabalho os casos de uso são uma composição de elementos (Título, Pré-condições, Pós-condições, Objetivo, Atores, etc). A vantagem desta representação é que ela permite explorar o conteúdo dos casos de uso, como por exemplo, o relacionamento entre os casos de uso por suas pré-condições e pós-condições conforme descrito na seção 2.4.3 (Relacionamentos entre Casos de Uso). Outra diferença entre os trabalhos é que [MOH 2008] não propõe o reuso dos artefatos.

3.5 Considerações do Capítulo

Neste capítulo foi apresentada uma breve descrição do trabalho de Somé [SOMÉ 2007]. Embora o propósito do trabalho do autor não seja o mesmo deste trabalho, algumas de suas definições, como o meta-modelo de representação dos casos de uso e a identificação de pré e pós-condições como indicações da seqüência de execução dos casos de uso, contribuíram para o desenvolvimento da solução apresentada neste trabalho.

Neste capítulo também foram apresentados dois trabalhos, de [LUC 2007] e [MOH 2008], com propósito muito semelhante ao desta dissertação. Em suma, a principal diferença deste trabalho em relação a estes dois citados é que ele tem como foco a gerência dos requisitos, representados por casos de uso. Enquanto que os outros dois trabalhos focam na gerência de todos os artefatos do *software*, ou seja, são mais abrangentes.

Por ser mais específico, com o escopo limitado ao nível de requisitos, este trabalho se aprofunda mais no tópico e propõe uma solução para o suporte da consistência e atualização dos requisitos. Esta solução se difere pelos seguintes motivos: propõe um padrão de formação dos requisitos funcionais, representados por casos de uso; suporta a representação dos relacionamentos entre os casos de uso, não só os propostos na UML, mas também relacionamento dos casos de uso por suas pré e pós-condições; fornece a rastreabilidade entre a requisição de mudança e os requisitos afetados pela mesma; mantém o histórico de cada requisito do *software*; possibilita a associação de um mesmo requisito com mais de uma aplicação (reuso). Mais detalhes sobre a solução proposta são apresentados no próximo capítulo. Com estes recursos, este trabalho permite o suporte à atualização e consistência dos requisitos durante a manutenção.

O foco principal de [LUC 2007] é definir a rastreabilidade entre os artefatos do *software* de forma semi-automática através do método LSI. Mas esta rastreabilidade não garante que os requisitos estão definidos de forma consistente e que estão sendo atualizados a partir de cada requisição de mudança (evolução do *software*). O foco principal de [MOH 2008] é a integração entre as práticas de gerência de configuração e a rastreabilidade dos artefatos de *software*.

Ambos os trabalhos, de [LUC 2007] e [MOH 2008], não se preocupam em definir a forma como os requisitos estão representados, como eles se relacionam entre si, com as aplicações de *software*, e com as requisições de mudanças. Estas são características essenciais para suportar a consistência e a atualização dos requisitos ao longo da evolução do *software* e estão no escopo desta pesquisa.

O próximo capítulo apresenta um estudo de caso exploratório para melhor compreensão do problema alvo deste trabalho seguido da descrição do modelo proposto para resolvê-lo.

4. MODELO PROPOSTO

Este capítulo ilustra o problema que este trabalho se propõe a resolver ressaltando as principais dificuldades identificadas a partir de um estudo de caso exploratório realizado na fábrica de *software* citada anteriormente. A partir das dificuldades identificadas é apresentada a proposta de solução para o problema através do Modelo Conceitual do problema, regras de consistência e do versionamento dos requisitos.

4.1 *Descrição do Problema*

Nesta seção será apresentada a descrição de um estudo exploratório importante para a compreensão do problema e a identificação dos principais requisitos que o modelo de gerência de requisitos que será proposto deve atender.

4.1.1 *Estudo de caso exploratório*

Com o objetivo de identificar as principais dificuldades envolvidas na manutenção dos requisitos consistentes e atualizados ao longo de projetos de manutenção de *software* realizou-se uma análise de como uma equipe de 5 Analistas de Sistemas trabalham com os requisitos de mais de 25 aplicações que passam por constante processo de evolução na fábrica de *software*.

Para cada aplicação que o grupo tem a responsabilidade de manter existe um SRS (*Software Requirements Specification*) onde estão descritos todos os requisitos da mesma. A maior parte destes SRSs foram criados a partir de um processo de Engenharia Reversa. A Engenharia Reversa foi realizada com o objetivo principal de adquirir conhecimento da aplicação, considerando que a equipe em questão não tinha conhecimento prévio das mesmas até ser designada a mantê-las. Observou-se que os requisitos funcionais dos SRSs existentes estão organizados por funcionalidade, onde cada funcionalidade é descrita da seguinte forma:

- Introdução/Objetivo da funcionalidade: Contém a descrição do objetivo geral da funcionalidade;
- Sequência de seqüência de par estímulo-resposta: Contém a seqüência de entradas e respostas do sistema para alcançar os resultados desejados;
- Requisitos Associados: Contém os requisitos de entrada e saída; e requisitos não-funcionais associados.

No processo de manutenção das aplicações seguido pela equipe estão definidas todas as fases de um processo de desenvolvimento: fase de visão, planejamento, desenvolvimento, teste e estabilização.

Durante a fase de visão, a equipe recebe uma lista de requisições de mudanças dos usuários. Os engenheiros escrevem os requisitos para atender cada necessidade dos usuários (requisição de mudança). Estes requisitos são escritos em um documento customizado criado pela equipe (com base no documento SRS) denominado ERD (*Enhancement Request Document*), para conter os requisitos que atendem a cada necessidade. Por convenção da equipe, cada ERD está relacionado com apenas uma aplicação. A Figura 4 ilustra a estrutura do documento ERD.

1 Necessidade do negócio
2 Estado atual dos requisitos da aplicação
3 Requisitos para atender a necessidade
4 Estado dos requisitos da aplicação após as modificações

Figura 4. Estrutura do documento ERD.

Necessidade do negócio: Uma breve descrição da necessidade de negócio que a alteração da aplicação irá atender. Esta descrição é baseada na requisição de mudança submetida pelo usuário.

Estado atual dos requisitos da aplicação: Contém a descrição dos requisitos da funcionalidade(s) atingida(s) pela modificação. Esta descrição é uma cópia da descrição contida no SRS da aplicação. Cabe ressaltar que a identificação de qual funcionalidade(s) deve ser alterada para atender a necessidade do negócio é feita de forma manual, baseada no conhecimento prévio dos Engenheiros de Requisitos sobre a aplicação em questão. No caso da necessidade do negócio demandar a criação de uma funcionalidade nova, esta sessão deve indicar que a funcionalidade não existe na aplicação atualmente. Caso seja identificado que a necessidade de negócio implica em modificações em mais de uma aplicação, um ERD é criado para cada aplicação.

Requisitos para atender a necessidade: Nesta seção são identificados e devidamente descritos os requisitos que devem ser alterados, removidos ou adicionados à aplicação.

Estado dos requisitos da aplicação após as modificações: Esta seção deve conter a descrição dos requisitos da aplicação que serão alterados, removidos ou adicionados de tal forma que estes estejam prontos para serem transcritos para o SRS da aplicação.

A partir de entrevistas com membros da equipe de analistas, observou-se que a principal preocupação da equipe é de manter os SRSs consistentes contendo os requisitos das aplicações atualizados ao longo do processo de manutenção.

Tendo em vista esta preocupação apontada pela equipe, identificou-se a necessidade de entender melhor o problema, identificar as principais dificuldades e qual a dimensão das mesmas. Para isso partiu-se para uma análise dos documentos de requisitos (SRSs e ERDs) criados pela equipe. Seguem abaixo as perguntas que guiaram a análise. Estas perguntas foram elaboradas pela própria autora com o objetivo de identificar se os requisitos afetados por requisições de mudanças estavam sendo atualizados e de que forma eles eram identificados (análise de impacto) e atualizados no SRS das aplicações:

1. Qual o número de requisitos de aplicação afetados e por quais requisições de mudanças?
2. Estes requisitos já foram atualizados no SRS da aplicação?
3. De que forma os requisitos da aplicação foram alterados? Reescrita ou simples cópia dos requisitos descritos nas ERDs (seção 4)?
4. Quantos requisitos de aplicação foram adicionados, modificados e removidos?
5. Existe alguma indicação na ERD de quais e como os requisitos de aplicação que devem ser atualizados no SRS?

Estas perguntas foram respondidas a partir da análise de 12 ERDs feita pela pesquisadora sem intervenção dos analistas. No caso das ERDs analisadas, cada ERD está associada com uma única aplicação. Seis aplicações são afetadas por estas ERDs.

Identificou-se que as 12 ERDs analisadas alteram um total de 118 requisitos de aplicação. Dentre estes requisitos de aplicação alterados, nenhum foi atualizado no SRS das aplicações correspondentes. No entanto para todos os requisitos afetados, tem-se informação para determinar qual a requisição de mudança que os alterou e de que forma eles foram alterados. No momento em que foi feita esta análise, todas estas requisições de mudança já haviam sido implementadas e entregues ao cliente. De acordo com a própria equipe, após a entrega do produto aos clientes, os requisitos das aplicações afetadas devem ser atualizados nos SRSs de cada aplicação. A não atualização acarreta numa desatualização e inconsistência dos dados que pode ter conseqüências como: falha na análise de impacto, falha no

entendimento dos requisitos por parte da equipe de desenvolvedores e testadores, entre outros problemas.

Não se sabe por que estes requisitos de aplicação não foram atualizados já que é uma preocupação apontada pela própria equipe. Uma hipótese de provável causa é o fato de não haver uma ferramenta conhecida pela equipe que auxilie no processo de atualização destes requisitos. A equipe utiliza o editor de texto Microsoft Word para escrita e atualização de requisitos.

Conforme relatado pela equipe e constatado na análise, por não existir ferramenta apropriada para atualização dos requisitos de aplicação ao longo do ciclo de vida da mesma (principalmente na fase de manutenção), a tarefa de manter os requisitos de aplicação atualizados torna-se bastante penosa e, por consequência, é deixada de lado na maioria dos casos.

4.1.2 Identificação dos principais requisitos para resolver o problema

O problema que motivou este trabalho está relacionado com a dificuldade de manter a documentação de requisitos de aplicação atualizados ao longo do seu ciclo de vida. De acordo com o caso prático descrito, os seguintes requisitos para a solução se destacam por descreverem as principais características que devem ser encontradas em um modelo de suporte à atualização e consistência dos requisitos para manutenção de *software*:

Requisito 1: Suporte à identificação dos requisitos afetados por uma requisição de mudança (análise de impacto). A identificação dos requisitos afetados de forma manual baseada no conhecimento dos *stakeholders*, considerando que uma requisição de mudança pode afetar mais de uma aplicação e que uma aplicação pode ter muitos requisitos (dependendo de sua complexidade) é uma atividade muito complexa e propensa a erros;

Requisito 2: Suporte ao gerenciamento da concorrência. Duas ou mais requisições de mudança podem afetar os mesmos requisitos de uma mesma aplicação em um mesmo período, sendo essencial o gerenciamento da concorrência;

Requisito 3: Os requisitos de aplicações devem ser atualizados e estarem consistentes para refletirem as alterações da requisição de mudança. Neste caso, além da atualização dos requisitos, deve-se ter cuidado para que estas atualizações só sejam consideradas como definitivas a partir do momento que a mudança é entregue aos usuários. Até este momento as

mudanças devem ser feitas de forma que possam ser descartadas, caso a requisição de mudança não seja atendida;

Requisito 4: Deve ser possível obter a rastreabilidade das mudanças dos requisitos de aplicação a fim de se ter informações tais como a partir de qual requisição de mudança um determinado requisito surgiu, foi removido ou alterado. Essa rastreabilidade entre as requisições de mudanças e os requisitos afetados fornece o histórico da evolução da aplicação. Este histórico pode auxiliar, por exemplo, os *stakeholders* na identificação das causas de problemas na aplicação, podendo agilizar a sua manutenção;

Requisito 5: Suporte ao *rollback* das alterações feitas por uma requisição de mudança. Uma requisição de mudança é implementada (os requisitos estão atualizados) e entregue aos usuários, porém, após certo período é verificado que a mudança deve ser desfeita (*rollback*).

Considerando os requisitos listados acima, as principais prioridades deste trabalho será encontrar uma solução para os seguintes requisitos: Identificação dos requisitos afetados (**requisito 1**), manutenção dos requisitos atualizados e consistentes conforme as requisições de mudanças (**requisito 3**) e manter a rastreabilidade das mudanças dos requisitos de aplicação (**requisito 4**).

O **requisito 2** que trata do suporte ao gerenciamento da concorrência não será abordado nesta dissertação por se tratar de um tópico de pesquisa complexo, que exige um estudo aprofundado do assunto, e não foi possível realizá-lo no tempo de desenvolvimento desta dissertação.

Estes problemas ficam em evidência para os analistas durante a fase de manutenção da aplicação, que é quando, na maioria das vezes, se despende muito tempo para identificar o impacto da mudança na aplicação e as modificações são feitas na aplicação sem se preocupar em atualizar a documentação dos requisitos.

4.2 Proposta

A partir do problema detalhado na seção anterior, desenvolveu-se um Modelo Conceitual como primeiro passo para se encontrar a sua solução. O modelo representa os conceitos envolvidos no problema. O modelo está demonstrado na Figura 5.

O Modelo Conceitual foi desenvolvido com base nos seguintes trabalhos:

- Cerri [CER 2007]: A composição do SRS apresentado na Figura 5 se baseia no Modelo Conceitual do SRS apresentado pela autora;

- Somé [SOMé 2007]: A definição dos elementos que compõem o Caso de Uso foi baseada no meta-modelo de casos de uso apresentado no trabalho do autor, com apenas duas exceções. São elas:
 - Relacionamento entre as pré e pós-condições: Diferente desta dissertação, Somé não propõe o relacionamento direto entre pré e pós-condições. Em vez disto ele propõe que o relacionamento é implícito e cria novas classes para representar estes relacionamentos: lista dos casos de uso que precedem o caso de uso atual (*FollowList*) e lista dos casos de uso que sucedem o caso de uso atual (*UseCaseEnabling*). Estas duas classes definidas por Somé não serão utilizadas pelo modelo apresentado neste trabalho.
 - Definição do relacionamento de generalização: O autor não aborda o relacionamento de generalização entre os casos de uso. Por este motivo neste trabalho foi adicionada a classe **SessaoGeneraliza** para permitir a representação do relacionamento.
 - Adição das classes **Participante**, **Sistema** e **Ator**: Somé não define estas classes no seu meta-modelo de casos de uso.
- [UML 2007]: A definição dos relacionamentos generalização, inclusão e extensão entre os casos de uso está baseada na especificação UML.

4.2.1 Detalhes do Modelo Conceitual

Esta seção apresenta a descrição das classes e dos relacionamentos entre as classes do Modelo Conceitual do problema apresentado. As principais classes que compõem o modelo são: a classe que representa o SRS (**SRS**); a classe que representa a Aplicação (**Aplicacao**); a classe que representa os Requisitos da Aplicação (**RequisitoAplicacao**); e a classe que representa as Requisições de Mudança (**RequisicaoMudanca**).

4.2.1.1 Classes que constituem o documento SRS

- **SRS:** A classe SRS representa o documento SRS da aplicação. Este documento mostra que um SRS é composto de quatro seções maiores: Introdução, Descrição Geral, Informações de Suporte e Requisitos de Aplicação, representadas respectivamente pelas classes: **IntroducaoSRS**, **DescricaoGeral**, **InformacoesSuporte** e **RequisitoAplicacao**). Esta definição da composição do SRS é baseada na representação do SRS apresentado em [CER 2007].
- **IntroducaoSRS:** A classe **IntroducaoSRS** representa a seção Introdução da SRS e contém informações que dão uma visão geral sobre o documento. Na SRS esta seção é composta de outras cinco subseções: Propósito, Escopo, Definições, Referências e Visão Geral representadas, respectivamente, pelas classes **Proposito**, **EscopoSRS**, **Definicoes**, **Referencias** e **VisaoGeral**.
- **Proposito:** A classe **Proposito** representa a subseção Propósito da SRS que indica seu objetivo geral. Indica, também, a audiência pretendida para a SRS.
- **EscopoSRS:** A classe **EscopoSRS** representa a subseção Escopo da SRS e descreve os objetivos específicos, focando no *software* que está sendo especificado.
- **Definicoes:** A classe **Definicoes** representa a subseção Definições, Acrônimos e Abreviaturas que contém as definições de todos os termos, siglas e abreviaturas necessárias para interpretar apropriadamente a SRS.
- **Referencias:** A classe **Referencias** representa a subseção Referências da SRS que identifica todos os documentos referenciados.
- **VisaoGeral:** A classe **VisaoGeral** representa a subseção Visão Geral da SRS que apresenta informações referentes à organização do documento.
- **DescricaoGeral:** A classe **DescricaoGeral** representa a Seção Descrição Geral da SRS, responsável por descrever os fatores gerais que afetam o produto. Esta seção

contém outras nove subseções: Funções do Produto, Características do Usuário, Suposições e Dependências, Restrições Gerais, Requisitos de Operação, Limites de Memória, Distribuição dos Requisitos, Requisitos de Adaptação de Local e Interfaces representados, respectivamente, pelas classes **FuncoesProduto**, **CaracteristicasUsuario**, **SuposicoesDependencias**, **RestricoesGerais**, **RequisitosOperacao**, **LimitesMemoria**, **DistribuicaoRequisitos**, **RequisitosAdaptacao** e **Interface**.

- **FuncoesProduto:** A classe **FuncoesProduto** representa a subseção Funções do Produto da SRS que fornece uma relação das funções do sistema, a fim de informar os principais objetivos. Esta classe é composta de outros dois elementos: *Stakeholders* e *Objetivo* representados, respectivamente, pelas classes **SRS_Stakeholder** e **Objetivo**.
- **SRS_Stakeholder:** Mantém a lista dos *stakeholders* (usuários, patrocinadores, etc) envolvidos com o *software*.
- **Objetivo:** A classe **Objetivo** é responsável por manter todos os objetivos necessários para o desenvolvimento completo do sistema.
- **CaracteristicasUsuario:** A classe **CaracteristicasUsuario** representa a subseção Características do Usuário da SRS que descreve as características gerais dos usuários do sistema.
- **SuposicoesDependencias:** A classe **SuposicoesDependencias** representa a subseção Suposições e Dependências da SRS que define os fatores que afetam os requisitos expressos na SRS como condições específicas de *hardware*.
- **RestricoesGerais:** A classe **RestricoesGerais** representa a subseção Restrições Gerais da SRS que fornece uma descrição geral de qualquer outro item que limite as opções dos desenvolvedores como normas reguladoras, limites de *hardware*, protocolos etc. Para isto o atributo *idRestricao* foi definido para manter a identificação da restrição e o atributo *descricao* para manter sua informação.
- **RequisitosOperacao:** A classe **RequisitosOperacao** representa a subseção Operação da SRS e descreve todas as operações normais e/ou especiais requisitadas pelo usuário, como rotinas de inicialização, processamento, backup's e restauração.

- **LimitesMemoria:** A classe **LimitesMemoria** representa a subseção Limites de Memória da SRS que especifica a memória (interna e externa) a ser, provavelmente, utilizada pelo *software*.
- **DistribuicaoRequisitos:** A classe **DistribuicaoRequisitos** representa a subseção Distribuição dos Requisitos na SRS que identifica os requisitos que podem ser adiados até versões futuras do sistema.
- **RequisitosAdaptacao:** A classe **RequisitosAdaptacao** representa a subseção Requisitos de Adaptação do Local da SRS que contém a especificação das situações em que o software deverá ser adaptado antes da instalação.
- **Interface:** A classe **Interface** representa as informações referentes às interfaces do sistema. Os tipos de interface são: Interfaces de Usuário, Interfaces do Hardware, Interfaces do Software e Interfaces de Comunicação.
- **InformacoesSuporte:** A classe **InformacoesSuporte** define a seção Informações do Suporte da SRS responsável por tornar a SRS mais fácil de ser utilizada. Esta seção é constituída por duas subseções: Tabela de Conteúdo e Índice e Apêndices representadas, respectivamente, pelas classes **Tabela** e **Apêndices**.
- **Tabela:** A classe **Tabela** representa a subseção Tabela de Conteúdo e Índice do SRS seguindo as práticas convencionais.
- **Apêndices:** A classe **Apêndices** representa a subseção Apêndices da SRS que os especifica, se necessário. Caso sejam identificados, deve ser informado se eles são ou não parte dos requisitos.

4.2.1.2 Classe que representa a Aplicação

- **Aplicacao:** Esta classe representa a aplicação de *software*. Nela devem estar definidos os dados da aplicação como a sua descrição, a data em que foi criada, por quem ela foi criada, qual o seu objetivo, qual o contexto no qual ela está inserida dentro da organização, número de usuários, tipos de usuários que utilizam a aplicação, entre outras informações.

4.2.1.3 Classes que constituem os Requisitos da Aplicação

- **RequisitoAplicacao:** Esta classe deve conter os requisitos da aplicação. Esta classe deve conter informações como o objetivo do requisito, a importância, comentários, dados da versão do requisito e seu estado, que indica se o requisito está ativo ou

inativo (caso tenha sido removido). Existem dois tipos de requisitos de aplicação: requisitos funcionais e requisitos não-funcionais, representados no modelo pelas classes **CasoUso** e **ReqNaoFuncionalGeral** respectivamente.

- **CasoUso:** Esta classe contém os requisitos funcionais da aplicação, que são representados por casos de uso. Esta classe deve conter informações como título do caso de uso, a frequência em que ele é executado, entre outras informações sobre o mesmo. Existem dois tipos de casos de uso no modelo, casos de uso normal e caso de uso de extensão, representados pelas classes **CasoUsoNormal** e **CasoUsoExtensao**. Os casos de uso “normais” podem conter pontos de extensão, representados no modelo pela classe **PontoExtensao**. Esta definição da composição dos casos de uso é baseada no trabalho de [SOMé 2007].
- **CasoUsoNormal:** Esta classe representa os casos de uso que não são extensão de outro(s) caso(s) de uso. Um caso de uso “normal” define o comportamento completo de um caso de uso. Sua execução resulta no atendimento de um objetivo ou em uma situação de erro.
- **CasoUsoExtensao:** Esta classe representa os casos de uso que estendem outro(s) caso(s) de uso. Um caso de uso estendido especifica um conjunto de comportamentos que são desvios com o objetivo de estender o comportamento definido por outros casos de uso. O caso de uso de extensão contém a descrição do caso de uso de extensão (representada pela classe **DescricaoCasoUsoExtensao**).
- **ReqNaoFuncionalGeral:** Esta classe representa os requisitos não-funcionais referentes ao sistema como um todo. Requisitos não-funcionais geral seriam, por exemplo, requisitos de desempenho que se apliquem a toda a aplicação.
- **PontoExtensao:** A classe **PontoExtensao** representa o ponto de extensão definido no caso de uso que é estendido por outro caso de uso. O ponto de extensão é um símbolo do caso de uso estendido que referencia um ponto particular do caso de uso que o estende. As interações definidas no caso de uso que o estende podem ser inseridas neste ponto de extensão. É através desta classe que está definido o **relacionamento de extensão** (*extend*) entre os casos de uso. Cada ponto de extensão está associado com uma ou mais partes (representadas pela classe **Parte**) do caso de uso de extensão. Estas partes são conjuntos de passos do caso de uso de extensão (representados pela classe **PassoCasoUso**).

- **ReqNaoFuncionalEspecifico:** Esta classe representa os requisitos não-funcionais referentes ao caso de uso ao qual estão associados. Requisitos não-funcionais específicos seriam, por exemplo, requisitos de desempenho que se apliquem apenas a funcionalidade descrita pelo caso de uso.
- **DescricaoCasoUso:** Esta classe representa a especificação do caso de uso. Ela é especializada por outras duas classes **DescricaoCasoUsoNormal** (especificação do caso de uso que não estende outros casos de uso) e **DescricaoCasoUsoExtensao** (especificação do caso de uso que estende outros casos de uso).
- **DescricaoCasoUsoNormal:** Classe que representa a especificação completa (título, pré-condições, pós-condições, passos do cenário principal, passos dos cenários alternativos, etc) do caso de uso que não estende nenhum outro caso de uso. Esta classe é composta pelas seguintes classes: **PreCondicao**, **PosCondicao**, **Alternativa** e **PassoCasoUso**.
- **PreCondicao:** Esta classe representa o conjunto de pré-condições associadas a um caso de uso.
- **PosCondicao:** Esta classe representa o conjunto de pós-condições associadas a um caso de uso.
- **Restricao:** Esta classe contém a descrição de uma restrição. A restrição é formada pelas informações **objeto** e **estado**. As pré-condições e pós-condições, representadas pelas classes **PreCondicao** e **PosCondicao** respectivamente, são tipos de restrições. A pós-condição “Usuário está registrado no Sistema” mostrada na Figura 1 mostra um exemplo de restrição. Neste caso “Usuário” é o objeto e “está registrado no Sistema” é o estado.
- **Alternativa:** A classe **Alternativa** representa os cenários alternativos do caso de uso. Uma alternativa especifica uma continuação possível do caso de uso após a execução de um passo. As alternativas são usadas para descrever exceções, situações de erro ou cursos de eventos menos comuns. As alternativas são constituídas por restrições (condições que devem ser verdadeiras para que a alternativa seja executada), passos da alternativa e pelo atraso, caso a alternativa tenha um tempo de espera (*delay*) associado. Estas informações são representadas no modelo pelas classes **Restricao**, **PassoCasoUso** e **Atraso** respectivamente.

- **PassoCasoUso:** Esta classe representa os passos do caso de uso. Um passo pode ser um bloco de repetição (representados pela classe **RepeteBloco**) ou um passo simples (representados pela classe **PassoSimples**).
- **RepeteBloco:** Esta classe define blocos de repetição. Um bloco de repetição define uma execução iterativa de uma seqüência de passos do caso de uso de acordo com uma condição e/ou um atraso. Blocos de repetição são descritos por palavras-chave de repetição tais como "enquanto" e "até que".
- **PassoSimples:** Esta classe representa todos os passos do caso de uso, exceto os que são blocos de repetição. Um passo simples pode ser dos seguintes tipos: passo que representa uma operação, passo de representa um desvio ou ramificação, passo que representa uma diretiva de inclusão de outro caso de uso e passo que representa uma diretiva de generalização de outro caso de uso. Os diferentes tipos de passos dos caso de uso estão representados pelas classes **PassoOperacao**, **Ramificacao**, **IncluiCasoUso** e **SessaoGeneraliza** respectivamente. Um passo simples pode estar associado com uma condição (representada pela classe **Restricao**), que deve ser verdadeira para que o passo seja executado, e/ou com um atraso (representada pela classe **Atraso**). A condição para o passo pode ser descrita através das palavras-chave "se ... então". Um passo simples pode estar associado com um ponto de extensão (classe **PontoExtensao**).
- **Atraso:** Representa o tempo que o sistema deve aguardar para a execução de um passo do caso de uso.
- **PassoOperacao:** Esta classe representa passos que são operações. Uma operação pode ser executada por um ator do ambiente, através de um gatilho, ou pelo próprio sistema, através de uma reação. Portanto uma operação pode ser um gatilho, representado pela classe **Gatilho** ou pode ser uma reação, representada pela classe **Reacao**. Um passo de operação pode estar associado com uma ou mais alternativas (representada pela classe **Alternativa**).
- **Gatilho:** O gatilho é uma operação disparada por um ator do sistema.
- **Reacao:** A reação é uma operação disparada pelo próprio sistema.

- **Ramificacao:** Esta classe representa passos que são ramificações. Um passo de ramificação inclui a referência para um passo *i*, de forma que o fluxo de eventos do caso de uso seja desviado para o passo *i*.
- **IncluiCasoUso:** Esta classe representa um passo do caso de uso que inclui um outro caso de uso. Este passo representa a realização de um **relacionamento de inclusão** entre um caso de uso e o caso de uso incluído referenciado no passo.
- **SessaoGeneraliza:** Esta classe representa um passo do caso de uso que generaliza um outro caso de uso. Esta classe permite a **relação de generalização** entre os casos de uso, ou seja, possibilita representar que um caso de uso generaliza outros casos de uso.
- **DescricaoCasoUsoExtensao:** Esta classe representa a descrição do caso de uso de extensão. A descrição do caso de uso de extensão é constituída por um conjunto de partes (representadas pela classe **Parte**).
- **Parte:** Representa uma parte do caso de uso de extensão. É nestas partes que se encontram as descrições dos passos do caso de uso (representados pela classe **PassoCasoUso**). Cada parte do caso de uso de extensão está relacionada com um ou mais pontos de extensão (representados pela classe **PontoExtensao**).
- **CasoUsoExtensao:** Representa o caso de uso de extensão. É constituído pela descrição do caso de uso de extensão (definido pela classe **DescricaoCasoUsoExtensao**).
- **Participante:** Esta classe representa o sistema ou o ator que interage com os casos de uso. Existem dois tipos de participantes: Sistema (representado pela classe **Sistema**) e Ator (representado pela classe **Ator**).
- **Sistema:** Esta classe representa o próprio sistema, que reage a operações disparadas pelo ator do caso de uso.
- **Ator:** Esta classe representa os atores do caso de uso.

4.2.1.4 Classe que representa as Requisições de Mudança

- **RequisicaoMudanca:** Esta classe representa as informações da Requisição de Mudança que devem alterar uma ou mais aplicações de *software*. Esta classe deve conter informações como: qual a origem da requisição de mudança, a data em que foi submetida, a sua descrição, o seu objetivo, a data em que as mudanças devem ser entregues aos clientes, a prioridade de implementação, o título, a pessoa que irá

verificar se a mudança foi atendida, o autor da requisição de mudança, entre outras informações.

4.2.2 Principais características do Modelo Conceitual

Nesta seção é apresentado como o Modelo Conceitual se propõe a atender aos requisitos identificados na seção 4.1.2.

- **Os requisitos da aplicação podem estar relacionados com mais de uma aplicação:** Isto é possível através do relacionamento entre as classes **Aplicacao** e **RequisitoAplicacao** que determina que uma aplicação pode estar associada com nenhum ou vários requisitos de aplicação, e que os requisitos de aplicação devem estar associados com pelo menos uma aplicação, podendo estar associados com mais de uma aplicação. A vantagem desta característica é que ela permite o **reuso dos requisitos**. Considerando o exemplo de um requisito funcional representado pelo caso de uso “Entrar no Sistema” demonstrado na Figura 1 da seção 2.4.3, este caso de uso pode estar associado com duas ou mais aplicações que possuam a mesma funcionalidade para registrar os seus usuários. Esta característica auxilia no atendimento dos **requisitos 1 e 3**. O reuso dos requisitos por mais de uma aplicação facilita a análise de impacto (**requisito 1**). Por exemplo, uma vez que se identifique que uma requisição de mudança afetou um requisito associado com mais de uma aplicação, já se sabe que estas aplicações foram afetadas pela mudança. Sem este recurso teria que se identificar todas as instâncias do requisito relacionados com cada aplicação. O reuso dos requisitos também auxilia no suporte a consistência e atualização dos requisitos (**requisito 3**), pois uma vez alterado o requisito, a modificação já refletirá para todas as aplicações associadas, sem a necessidade de replicação das modificações.
- **Ligação entre requisição de mudança e requisitos de aplicação:** Esta ligação se dá através dos relacionamentos “Adiciona”, “Altera” e “Remove” entre a classe **RequisicaoMudanca** e **RequisitoAplicacao**. O relacionamento “Adiciona” determina que uma requisição de mudança pode adicionar requisitos de aplicação. O relacionamento “Altera” determina que uma requisição de mudança pode alterar requisitos de aplicação. E o relacionamento “Remove” determina que uma requisição de mudança pode remover requisitos de aplicação. Através destas ligações é possível se obter a rastreabilidade das mudanças dos requisitos de aplicação a fim de se ter

informações tais como a partir de qual requisição de mudança um determinado requisito surgiu, foi removido ou alterado. Essa rastreabilidade, unida com um controle de versão dos requisitos, fornece o histórico da evolução da aplicação. Esta característica auxilia no atendimento dos **requisitos 1, 3, 4 e 5**. A rastreabilidade entre as requisições de mudanças e os requisitos de aplicação e o histórico dos requisitos de aplicações auxiliam na análise de impacto (**requisito 1**). Uma vez identificado que um determinado requisito foi afetado por uma requisição de mudança, o sistema fornece o histórico de mudanças que afetaram este requisito no passado, junto com os outros requisitos que também foram afetados pela mesma requisição de mudança. Se uma requisição de mudança do passado afetou os requisitos A e B, quando se identificar que uma nova requisição de mudança afeta o requisito A, existe a possibilidade dela também afetar o requisito B. Esta característica também auxilia na manutenção dos requisitos atualizados e consistentes (**requisito 3**) em relação às requisições de mudanças, já que permite identificar os requisitos de aplicação que foram afetados por uma requisição de mudança e de que forma eles foram afetados. Esta característica atende ao **requisito 4** (rastreabilidade das mudanças dos requisitos de aplicação). A rastreabilidade entre requisições de mudanças e requisitos de aplicação afetados é essencial para que seja possível o *rollback* das alterações feitas por uma requisição de mudança (**requisito 5**). Detalhes de como esta solução suporta o versionamento dos requisitos de aplicação podem ser encontrados na seção 4.2.4.

- **Representação dos casos de uso como um conjunto de elementos:** Conforme descrito na seção anterior, a classe **CasoUso** é constituída por um conjunto de classes, tais como, pré-condições (classe **PreCondicao**), pós-condições (classe **PosCondicao**), passos do cenário principal e dos cenários alternativos (classe **PassoCasoUso**), Atores (classe **Ator**), Ponto de Extensão (classe **PontoExtensao**), entre outros. Essa granularidade permite uma verificação de consistência mais eficaz, já que é possível por exemplo, relacionar a pré-condição do caso de uso “Submeter Pedido” com a pós-condição do caso de uso “Entrar no Sistema” como ilustrado na Figura 1 da seção 2.4.3. Esta característica auxilia no suporte ao **requisito 3**, que trata da manutenção dos requisitos atualizados e consistentes.
- **Representação da relação de inclusão entre os casos de uso:** Esta relação está representada pelo relacionamento entre as classes **IncluiCasoUso**, que é um tipo de passo (**IncluiCasoUso** é a especialização da classe **PassoSimples**), com a classe

CasoUsNormal. Ou seja, o caso de uso que inclui possui um passo cujo tipo é de Inclusão. O caso de uso incluído é representado pela classe **CasoUsNormal**. A representação do relacionamento de inclusão entre os casos de uso também auxilia na garantia da consistência (**requisito 3**). Por exemplo, se uma requisição de mudança afeta um caso de uso que inclui outro caso de uso, dependendo do tipo de mudança, o analista precisa verificar se a inclusão ainda é válida, ou se alguma mudança deve também ser feita no caso de uso incluído. Tendo o relacionamento devidamente mapeado, esta análise é facilitada. Esta característica também auxilia no suporte à análise de impacto (**requisito 1**). Mais detalhes podem ser verificados na seção 4.2.2 (Regras de consistência).

- **Representação da relação de extensão entre os casos de uso:** Esta relação está representada pelo relacionamento entre as classes **PontoExtensao** do caso de uso estendido e a classe **Parte**, que é o conjunto de passos do caso de uso que estende. A representação do relacionamento de extensão entre os casos de uso, pelos mesmos motivos que o relacionamento de inclusão, auxilia no suporte à consistência dos casos de uso (**requisito 3**). Esta característica também auxilia no suporte ao **requisito 1**. Mais detalhes podem ser verificados na seção 4.2.2 (Regras de consistência).
- **Representação da relação de generalização entre os casos de uso:** Esta relação está representada pelo relacionamento entre as classes **SessaoGeneraliza**, que é um tipo de passo (**SessaoGeneraliza** é a especialização de **PassoSimples**) com a classe **CasoUsNormal**. Neste caso, o caso de uso mais genérico possui um passo que é uma sessão de generalização. Esta sessão se liga com o caso de uso especializado, representado pela classe **CasoUsNormal**. Assim como o relacionamento de inclusão e extensão, o relacionamento de generalização também facilita o suporte à consistência das informações dos casos de uso (**requisito 3**). Esta característica também auxilia no suporte ao **requisito 1**. Mais detalhes podem ser verificados na seção 4.2.2 (Regras de consistência).
- **Representação da relação por pré e pós-condições entre os casos de uso:** Representada pela relação entre as classes **PreCondicao** e **PosCondicao**, esta relação possui a restrição “DescricaoCasoUsNormal distintos” que define que ela só é possível entre casos de uso distintos. Isto deve-se ao fato que a pré-condição de um caso de uso só pode estar relacionada com a pós-condição de outro caso de uso, e não com a pós-condição do próprio caso de uso. As classes **PreCondicao** e **PosCondicao**

herdam os atributos **Objeto** e **Estado** da classe **Restricao**. A descrição das pré e pós-condições do sistema proposto é definida da forma “Objeto” + “Estado”. Sendo que quando uma pré-condição de um caso de uso A é relacionada com a pós-condição de um caso de uso B, o sistema determina que esta pré-condição é formada pelo mesmo “Objeto” e “Estado” da pós-condição do caso de uso B, permitindo apenas que o analista complemente a definição do “Estado” com algum comentário ou definição que julgue necessário. Este relacionamento por pré e pós-condições auxilia no suporte à consistência (**requisito 3**) e na análise do impacto das requisições de mudanças (**requisito 1**). Detalhes sobre como esta característica auxilia no suporte à consistência podem ser obtidos na próxima seção. Em relação à análise de impacto, uma vez que uma requisição de mudança altere um caso de uso cuja pós-condição é pré-condição de outro caso de uso, este outro caso de uso também deve ser revisado, pois dependendo da mudança, também pode ser afetado.

4.2.3 Regras de consistência

A partir da definição do Modelo Conceitual, partiu-se para a definição das regras de consistência que devem ser respeitadas pelo sistema de gerência de requisitos que siga o modelo proposto neste trabalho. Estas regras de consistência apóiam a solução no atendimento do **requisito 3**, identificado na seção 4.1.2. O **requisito 3** determina que os requisitos de aplicações devem ser atualizados e estarem consistentes para refletirem as alterações das requisições de mudança.

1. O sistema de gerência de requisitos não deve permitir a remoção da pós-condição de um caso de uso que está associada com a pré-condição de outro caso de uso (visualizar Figura 6);
2. O sistema de gerência de requisitos não deve permitir a remoção de um caso de uso cuja pós-condição é pré-condição de outro caso de uso (visualizar Figura 7);

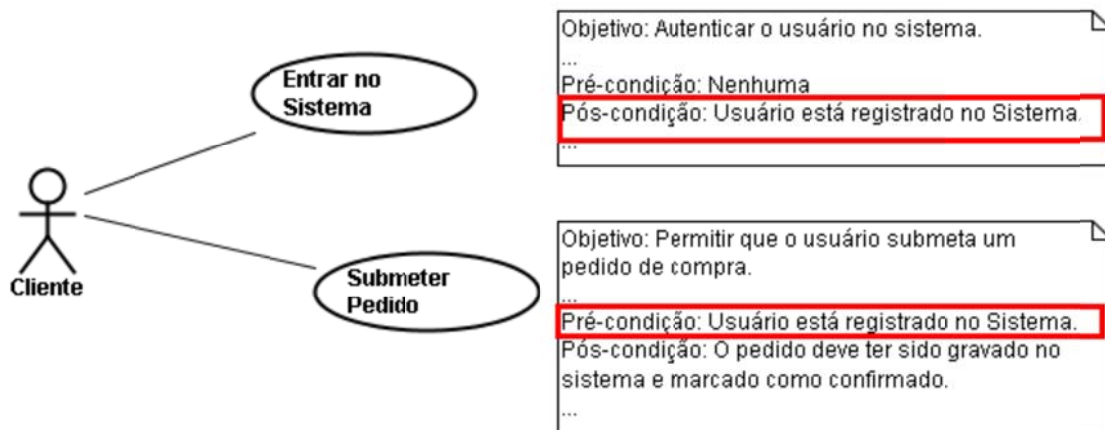


Figura 6. Remoção da Pós-condição.

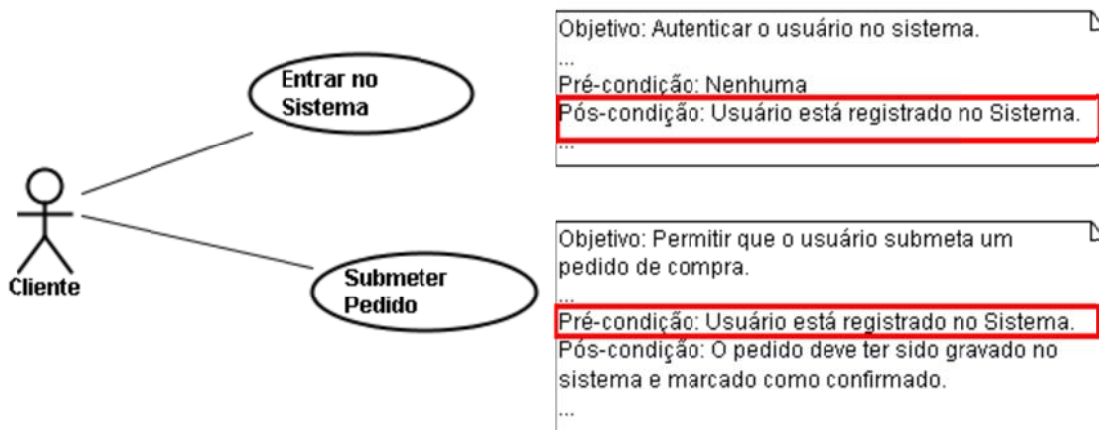


Figura 7. Remoção de um caso de uso relacionado por Pós-condição.

3. Quando na remoção de um caso de uso, o sistema de gerência de requisitos deve verificar se o caso de uso sendo removido inclui outros casos de uso. Para cada caso de uso incluído, o sistema deve remover também o caso de uso incluído, caso o mesmo não possua relação com ator ou com nenhum outro caso de uso (por inclusão, extensão, generalização ou é pré-condição de outro caso de uso). Veja na Figura 8 que se o caso de uso "Criar Credíário para Cliente" for removido, o sistema deve também remover o caso de uso "Verificar Cadastro do Cliente no SPC" porque o mesmo não tem relação com ator e nem com nenhum outro caso de uso.

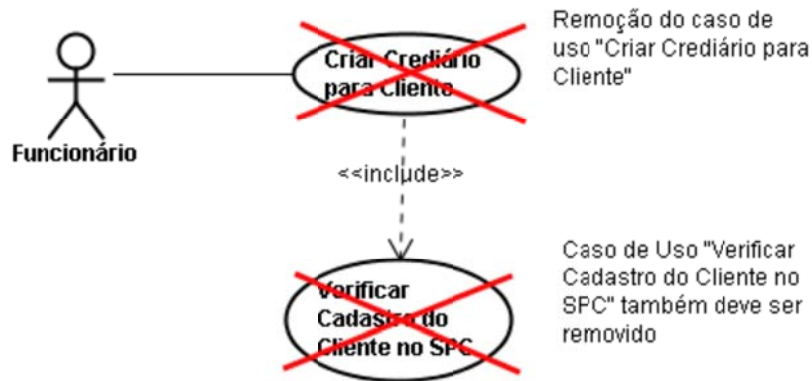


Figura 8. Remoção de um caso de uso relacionado por Inclusão.

4. Quando na remoção de um passo de um caso de uso que inclui outro caso de uso. O sistema deve remover também o caso de uso incluído, caso o mesmo não possua relação com ator ou com nenhum outro caso de uso. Veja na Figura 9 que se o passo que do caso de uso "Criar Credário para Cliente" que inclui o caso de uso "Verificar Cadastro do Cliente no SPC" for removido, o sistema deve também remover o caso de uso "Verificar Cadastro do Cliente no SPC".

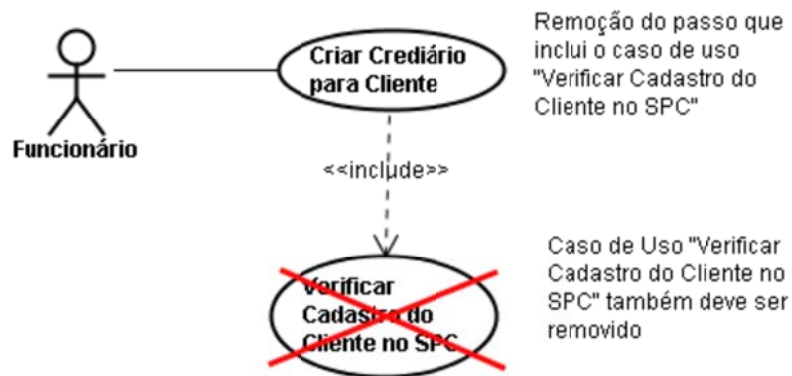


Figura 9. Remoção do passo de um caso de uso relacionado por Inclusão.

5. Quando na remoção de um caso de uso, o sistema de gerência de requisitos deve verificar se o caso de uso sendo removido é estendido por outros casos de uso. Para cada caso de uso que estende o caso de uso sendo removido, o sistema deve removê-lo, caso o mesmo não possua relação com ator ou com nenhum outro caso de uso. Um exemplo é demonstrado na Figura 10.

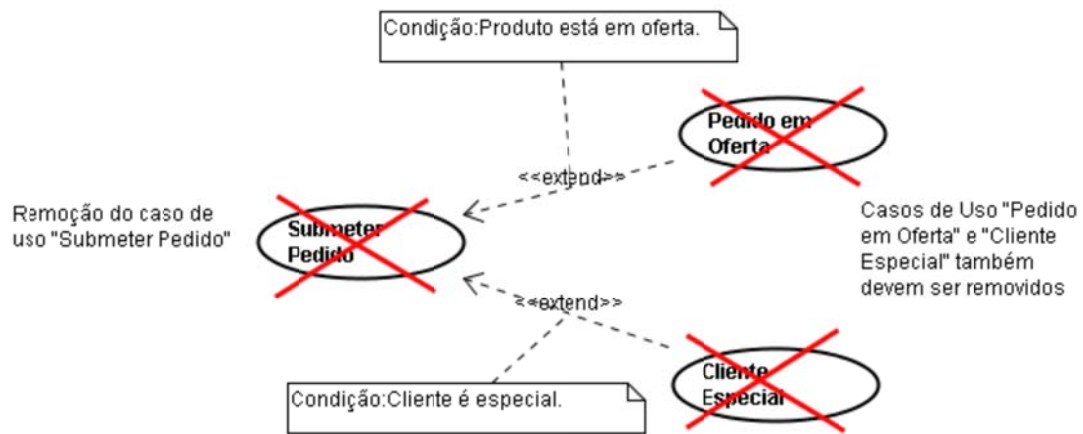


Figura 10. Remoção de um caso de uso relacionado por Extensão.

6. Quando na remoção de um passo de um caso de uso que representa a extensão por outro caso de uso. O sistema deve remover também o caso de uso que estende, caso o mesmo não possua relação com ator ou com nenhum outro caso de uso;
7. Quando na remoção de um caso de uso, o sistema de gerência de requisitos deve verificar se o caso de uso sendo removido generaliza outros casos de uso. Para cada caso de uso especializado pelo caso de uso sendo removido, o sistema deve removê-lo, caso o mesmo não possua relação com ator ou com nenhum outro caso de uso. A Figura 11 ilustra um exemplo desta situação.
8. Quando na remoção de um passo de um caso de uso que generaliza outros casos de uso. Para cada caso de uso especializado pelo caso de uso sendo removido, o sistema deve removê-lo, caso o mesmo não possua relação com ator ou com nenhum outro caso de uso;

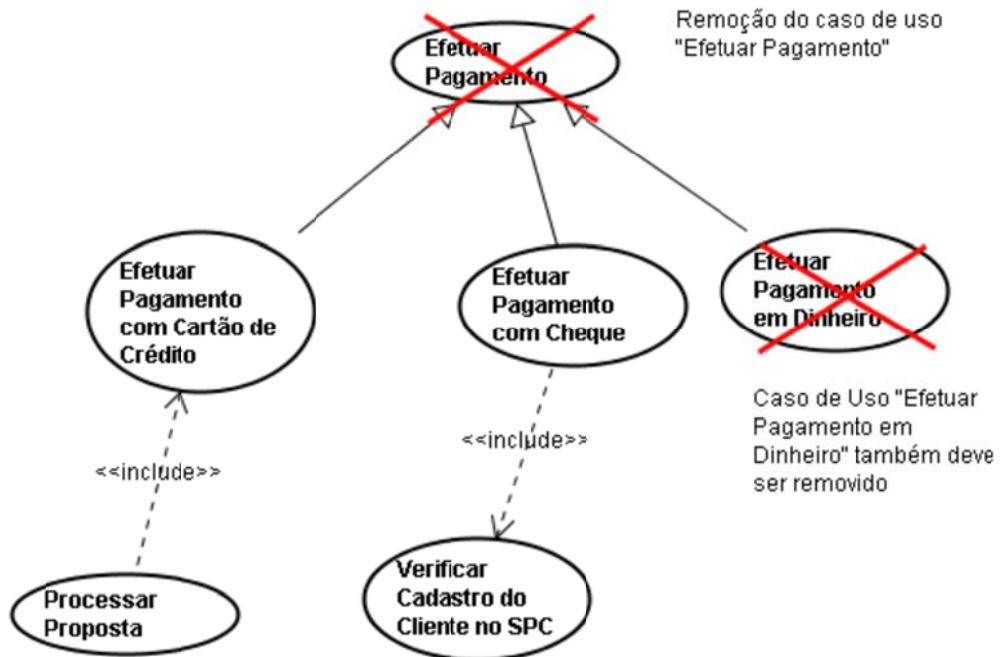


Figura 11. Remoção de um caso de uso relacionado por Generalização.

4.2.4 Versionamento dos requisitos

Além da representação dos casos de uso conforme demonstrado no Modelo Conceitual da Figura 5 e das regras de consistência apresentadas na seção anterior, outro artifício utilizado nesta solução para o suporte à consistência e atualização dos requisitos (**requisito 3**) e o histórico da evolução da aplicação (**requisito 4**), é o versionamento dos requisitos.

Para o versionamento dos requisitos foi utilizado o conceito de Revisão [COL 2004]. A cada alteração de um requisito de aplicação, o sistema de gerência de requisitos deve criar uma nova revisão. A revisão é um número que representa o estado do requisito depois de uma transação. Sendo que a transação pode ser de criação do requisito, alteração ou remoção do mesmo.

Se um elemento de um caso de uso (requisito funcional), como por exemplo, um passo, é alterado, uma nova revisão do caso de uso é gerada. Esta revisão está associada com todos os elementos que compõem o caso de uso no momento da transação.

Quando um requisito é criado ele está automaticamente associado com a revisão de número 1. Se ele sofrer qualquer tipo de alteração, o sistema automaticamente criará a revisão de número 2, e assim consecutivamente. Cada requisito contém sua própria numeração de revisão, ou seja, a numeração não é única entre os requisitos.

Quando uma requisição de mudança cria ou altera um requisito uma nova revisão do requisito é gerada, e a requisição de mudança é associada a esta nova revisão, conforme mostra a Figura 12. O objeto *RequisitoAplicacao1* foi criado pela *RequisicaoMudanca1* e a revisão de número 1 foi criada. Quando o mesmo requisito *RequisitoAplicacao1* foi alterado pela *RequisicaoMudanca2*, a revisão de número 2 foi criada. O estado do objeto *RequisitoAplicacao1* associado com a revisão de número 1 é mantido para histórico. Isto permite que o analista saiba exatamente qual o estado do requisito após as modificações de uma determinada requisição de mudança. O sistema deve permitir que o analista possa visualizar o estado do requisito em cada revisão.



Figura 12. Versionamento de Requisitos.

Este recurso de versionamento viabiliza o armazenamento do histórico da evolução da aplicação, conforme descrito no **requisito 4** da seção 4.1.2. Mais detalhes em relação ao recurso de versionamento serão fornecidos nos exemplos que demonstram os resultados no próximo capítulo, seção 5.3.3.

4.3 Considerações finais

O **requisito 1**, que consiste na identificação dos requisitos afetados por uma requisição de mudança, foi atendido parcialmente. A solução proposta auxilia na identificação dos requisitos, considerando que mostra ao analista todos os requisitos das aplicações e como eles estão relacionados, inclusive propondo novos relacionamentos: por pré e pós-condições e pelo histórico de mudanças. Porém a identificação continua sendo manual. Uma vez identificado que uma requisição de mudança afeta um caso de uso específico, o modelo proposto pode auxiliar na identificação dos outros casos de uso afetados a partir do histórico de mudanças e também das dependências entre os casos de uso.

O **requisito 3**, que corresponde à manutenção dos requisitos atualizados e consistentes conforme as requisições de mudanças também foi parcialmente atendida pela solução. Parcialmente porque apesar de suportar a atualização e consistência dos requisitos, este modelo não considera o estado da requisição de mudança para só efetivar as alterações a partir do momento em que a requisição de mudança seja entregue ao usuário.

O **requisito 4**, que consiste em manter a rastreabilidade das mudanças dos requisitos de aplicação foi totalmente atendido por esta solução através do Modelo Conceitual e também do recurso de versionamento.

O **requisito 5**, que consiste em suportar o *rollback* das alterações feitas por uma requisição de mudança foi atendido parcialmente pelo relacionamento entre a requisição de mudança e os requisitos de aplicação. Com esta rastreabilidade é possível recuperar o estado anterior a uma requisição de mudança.

O **requisito 2**, que trata do gerenciamento da concorrência dos requisitos de aplicação serão tratadas como trabalho futuro.

Estas considerações em relação ao escopo atendido por esta solução serão ilustradas no próximo capítulo.

5. AVALIAÇÃO DO MODELO PROPOSTO

Este capítulo visa apresentar a avaliação do modelo proposto. Para viabilizar esta avaliação foi desenvolvido um protótipo que será descrito na seção 5.1.

Após o desenvolvimento do protótipo, fez-se um levantamento de todos os cenários relevantes ao contexto deste trabalho, considerando todas as peculiaridades desta solução. Considerando os cenários, selecionou-se três sistemas hipotéticos para utilizá-los nas demonstrações. Estes exemplos estão descritos na seção 5.2. As demonstrações de como o sistema se comporta nos diversos cenários estão apresentadas na seção 5.3.

5.1 Protótipo

Com o objetivo de auxiliar a avaliação do modelo proposto neste trabalho, um protótipo de *software* foi desenvolvido. A implementação iniciou com a escolha da arquitetura e das linguagens para a implementação do protótipo. Foi definido que a persistência seria feita em um banco de dados, com a entrada de dados feita através de uma interface gráfica. Assim, optou-se pela arquitetura em três camadas, no modelo MVC (*Model View Control*). Microsoft C# .Net foi a linguagem de implementação utilizada para as camadas de dados, controle e interface. Já para a camada de persistência, que dá suporte a camada de dados, foi utilizado o banco de dados Microsoft Access. Estas escolhas foram feitas por sua grande utilização tanto no meio acadêmico como no meio industrial, por experiências de sucesso alcançadas anteriormente, pela disponibilidade das ferramentas e, também, pela facilidade de integração entre o banco de dados e a linguagem definida.

Os documentos do protótipo criados (Modelo de Casos de Uso, Especificação dos Casos de Uso e Diagrama de Classes) podem ser encontrados no Apêndice I.

5.2 Descrição dos sistemas utilizados para exemplificação

Com o propósito de avaliação da solução apresentada neste trabalho através do protótipo, definiu-se um exemplo hipotético envolvendo três sistemas. Segue a descrição do que se tratam estes sistemas para melhor compreensão dos resultados que serão apresentados na posterior (seção 5.3).

5.2.1 Sistema de Vendas

Este sistema tem como objetivo possibilitar a venda online de produtos de uma empresa. O sistema realiza o atendimento ao cliente, permitindo que o mesmo selecione os produtos e solicite comprá-los. Para solicitar a compra dos produtos, o cliente deve informar o código de cada produto. Quando informado o código, o sistema verifica se o produto não está em oferta e caso esteja, o sistema calcula o desconto no preço. Caso o cliente seja um cliente especial da loja o sistema calcula um desconto em cima do preço total da compra. O controle do pagamento dos produtos é feito através da interface com outro sistema chamado “Sistema de Pagamento”.

O sistema permite que o cliente acompanhe o andamento do seu pedido e que cancele o mesmo caso queira desistir da compra.

O sistema recebe os produtos de seus fornecedores e entrega os produtos aos seus clientes através de uma transportadora. A cotação de preços dos produtos e compra dos produtos dos fornecedores é feita por outro sistema “Sistema de Compra de Fornecedores”. A Figura 13 abaixo ilustra o modelo de casos de uso deste sistema. A especificação de cada caso de uso pode ser encontrada no Apêndice II deste trabalho.

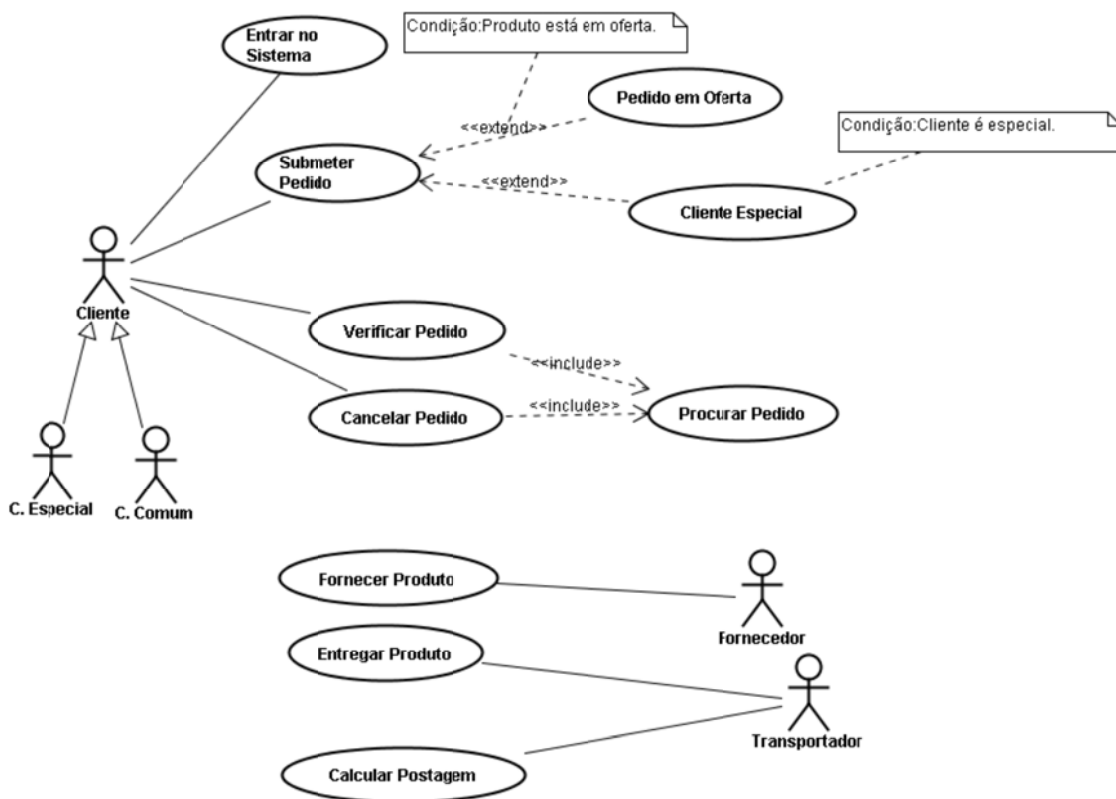


Figura 13. Sistema de Vendas.

5.2.2 Sistema de Pagamento

O sistema de pagamento tem como propósito permitir que o cliente efetue o pagamento para a empresa. O cliente pode efetuar o pagamento de três formas:

- Com cartão de crédito: Neste caso o funcionário deve informar os dados do cartão de crédito do cliente. O sistema verifica o crédito do cliente e em caso positivo efetua o pagamento;
- Com cheque: Neste caso o cliente deve informar os dados do cheque. O sistema verifica o crédito do cliente junto ao SPC e em caso positivo efetua o pagamento;
- Em dinheiro: O sistema solicita que o funcionário informe a quantia recebida em dinheiro. O sistema efetua o pagamento.

A Figura 14 ilustra o modelo de casos de uso deste sistema. A especificação de cada caso de uso pode ser encontrada no Apêndice II deste trabalho.

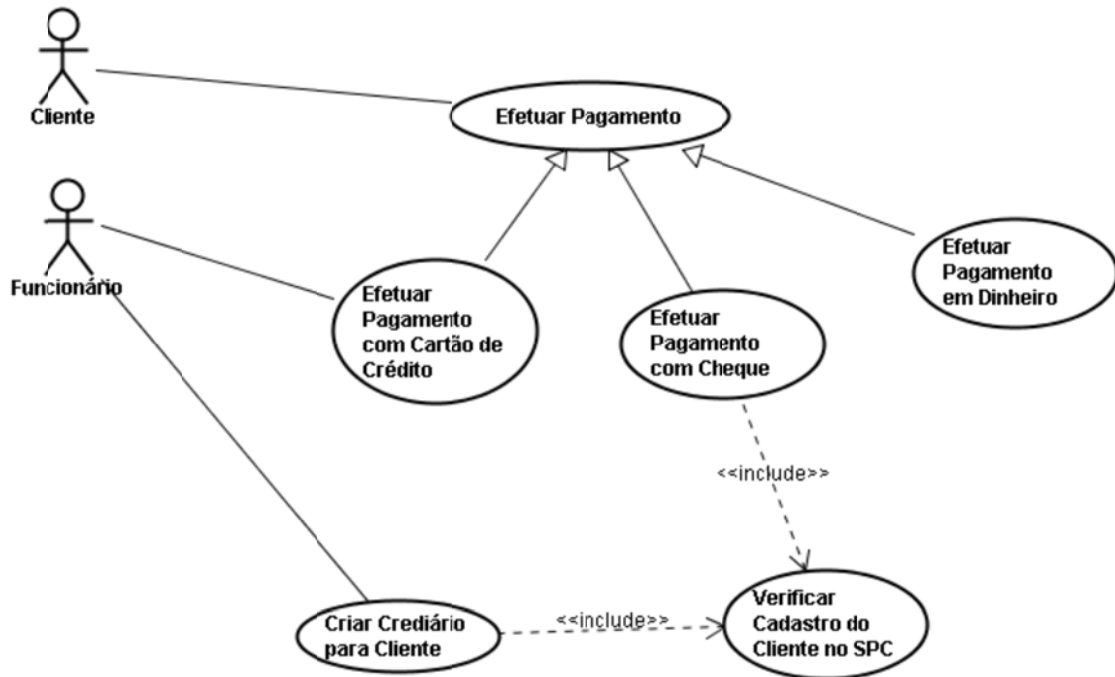


Figura 14. Sistema de Pagamento.

5.2.3 Sistema de Compra de Fornecedores

Este é um sistema online cujo objetivo é permitir que o funcionário da empresa submeta requisições de compra aos seus fornecedores a fim de atender aos pedidos dos seus

clientes. O sistema permite que os fornecedores informem suas cotações de preços de produtos e então submetam propostas que atendam as requisições de compra enviadas pela empresa. O funcionário da empresa pode então aceitar a proposta enviada pelo fornecedor e enfim efetuar a compra do mesmo. A Figura 15 ilustra o modelo de casos de uso deste sistema. A especificação de cada caso de uso pode ser encontrada no Apêndice II deste trabalho.

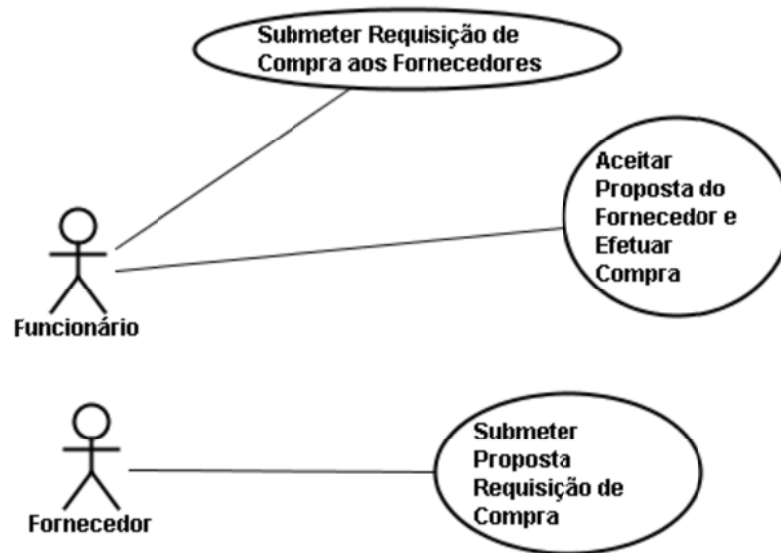


Figura 15. Sistema de Compra de Fornecedores.

5.2.4 Visão das Relações entre os Sistemas

Os três sistemas apresentados nas seções anteriores estão interligados. O Sistema de Vendas necessita do Sistema de Pagamento para efetuar o pagamento de suas vendas, assim como necessita do Sistema de Compra de Fornecedores para efetuar a compra de produtos dos seus fornecedores. Tanto o Sistema de Pagamento como o Sistema de Compra de Fornecedores não serão acionados caso não tenha acontecido um pedido de compra do cliente através do Sistema de Vendas. Estas ligações entre os sistemas estão ilustradas na Figura 16.

Pode-se perceber que além das relações descritas pela UML, também estão identificados os relacionamentos por pré e pós-condições conforme apresentado na seção 2.4.3. Para efeito de ilustração, os relacionamentos por pré e pós estão identificados de forma gráfica. Porém é importante ressaltar que não está sendo proposta uma extensão da UML. A representação gráfica destes relacionamentos por pré e pós-condições é apenas para auxiliar na compreensão dos exemplos.

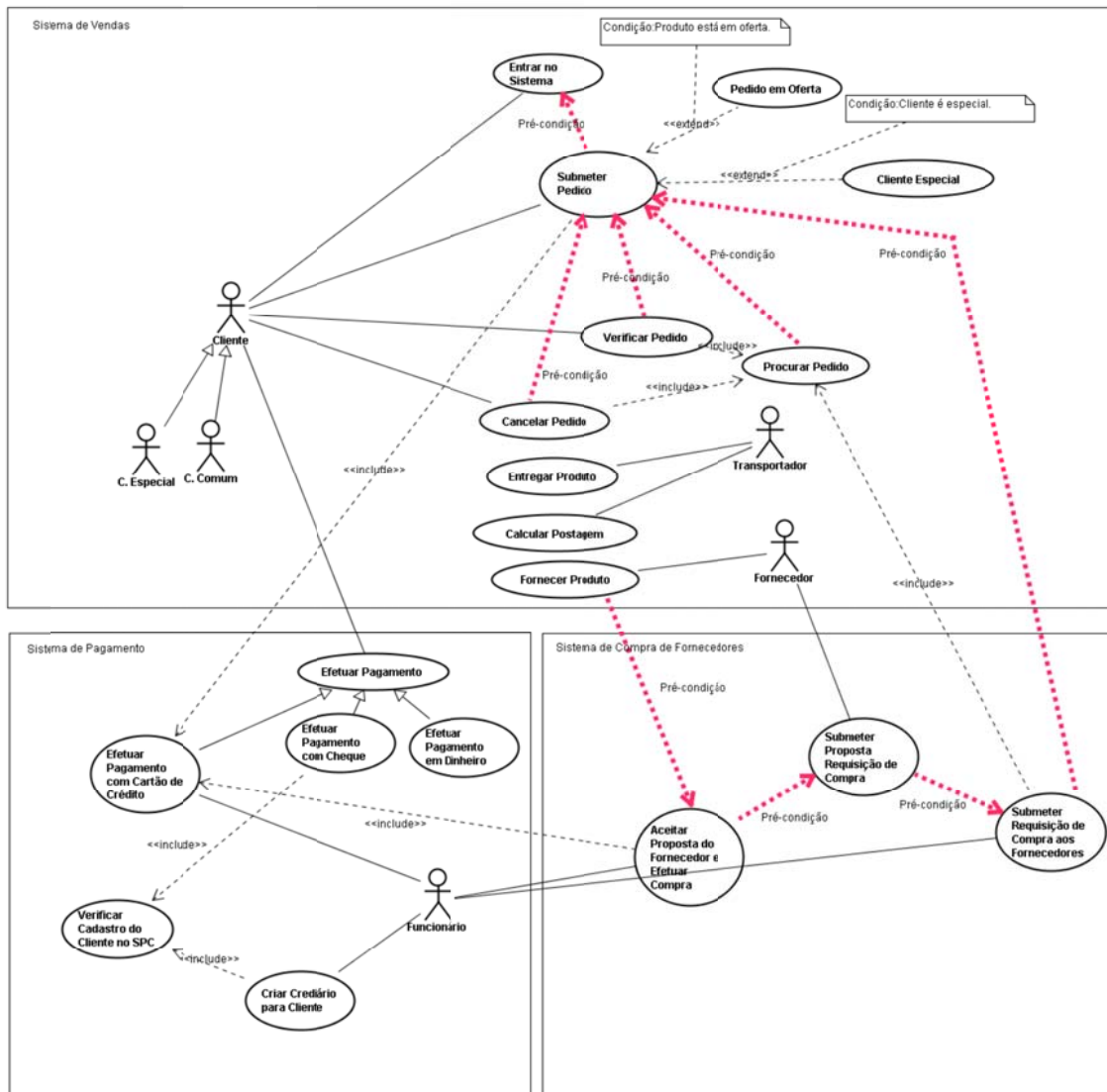


Figura 16. Relações entre os sistemas.

5.3 Demonstração dos resultados

Conforme descrito na seção 4.1.2 do capítulo anterior, dos cinco requisitos listados, a solução apresentada neste trabalho se propõe a resolver quatro: requisitos **1, 3, 4 e 5**. Seguem abaixo demonstrações a partir de cenários, utilizando os sistemas da seção 5.2 para exemplificação, de como o sistema atende estes requisitos. O sistema de gerência de requisitos está representado pelo protótipo desenvolvido.

5.3.1 Identificação dos requisitos afetados por uma requisição de mudança

Nesta seção serão mostrados cenários ilustrando como o sistema atende o **requisito 1**, através do cadastro de requisições de mudança e da identificação dos requisitos de aplicação que são afetados pelas mesmas.

A Figura 17 mostra a tela do protótipo onde o analista deve preencher os dados da requisição de mudança. Após preencher os dados da requisição de mudança e salvá-la, o analista deve identificar os requisitos de aplicação que a requisição de mudança modifica. Para esta identificação, o sistema fornece as opções para identificar os casos de uso afetados (através do botão “Identificar Casos de Uso Afetados”) e identificar os requisitos não-funcionais afetados (através do botão “Identificar Requisitos Não Funcionais Geral Afetados”). Como esta requisição de mudança se trata da adição de novas funcionalidades, neste exemplo o analista deve selecionar a opção “Identificar Casos de Uso Afetados”.

A imagem mostra uma janela de software intitulada "Adicionar Requisição de Mudança". O formulário contém os seguintes campos e botões:

- Origem da Mudança: Requisição de Mudança dos usuários do sistema.
- Título: Adicionar funcionalidades para requisição e fechamento de compra com fornecedor.
- Descrição: (campo vazio)
- Projeto: Manutenção Sistema de Cotação de Pedidos com Fornecedores
- Verificador: Luciana_Belleza
- Botões: Salvar, Identificar Casos de Uso Afetados, Identificar Requisitos Não Funcionais Geral Afetados.
- Mensagem de status: Requisição de Mudança salva com sucesso.

Figura 17. Cadastro da Requisição de Mudança.

Quando o analista seleciona a opção de identificação dos casos de uso afetados o sistema mostra uma tela com as seguintes informações:

Árvore “**Aplicação x Casos de Uso**”: O primeiro nível de elementos desta árvore contém lista de todas as aplicações cadastradas nos sistema. Conforme pode ser observado na Figura 18, as aplicações descritas na seção 5.2 (Sistema de Vendas, Sistema de Pagamento e Sistema de Cotação de Pedidos com Fornecedores) estão cadastradas no sistema. No segundo nível da árvore estão os casos de uso pertencentes a cada aplicação. Para cada caso de uso listado, o sistema gera três sub-árvores com os casos de uso incluídos (sub-árvore “Inclui”),

que estendem (sub-árvore “Extendido Por”) ou que são generalizados (sub-árvore “Generaliza”) pelo caso de uso. No caso da aplicação “Sistema de Cotação de Pedidos com Fornecedores”, apenas o caso de uso “Submeter Requisição de Compra aos Fornecedores” está cadastrado. Pode-se observar que o caso de uso “Submeter Requisição de Compra aos Fornecedores” possui uma sub-árvore “Inclui”, que mostra o caso de uso incluído: “Procurar Pedido” (Figura 20). Os outros casos desta aplicação serão adicionados pela requisição de mudança cadastrada na Figura 17. As Figuras 21 e 22 mostram os casos de uso sendo adicionados. As Figuras 18 e 19 mostram exemplos de sub-árvores “Extendido Por” e “Generaliza” (estas sub-árvores só são mostradas quando o caso de uso possui os respectivos relacionamentos).

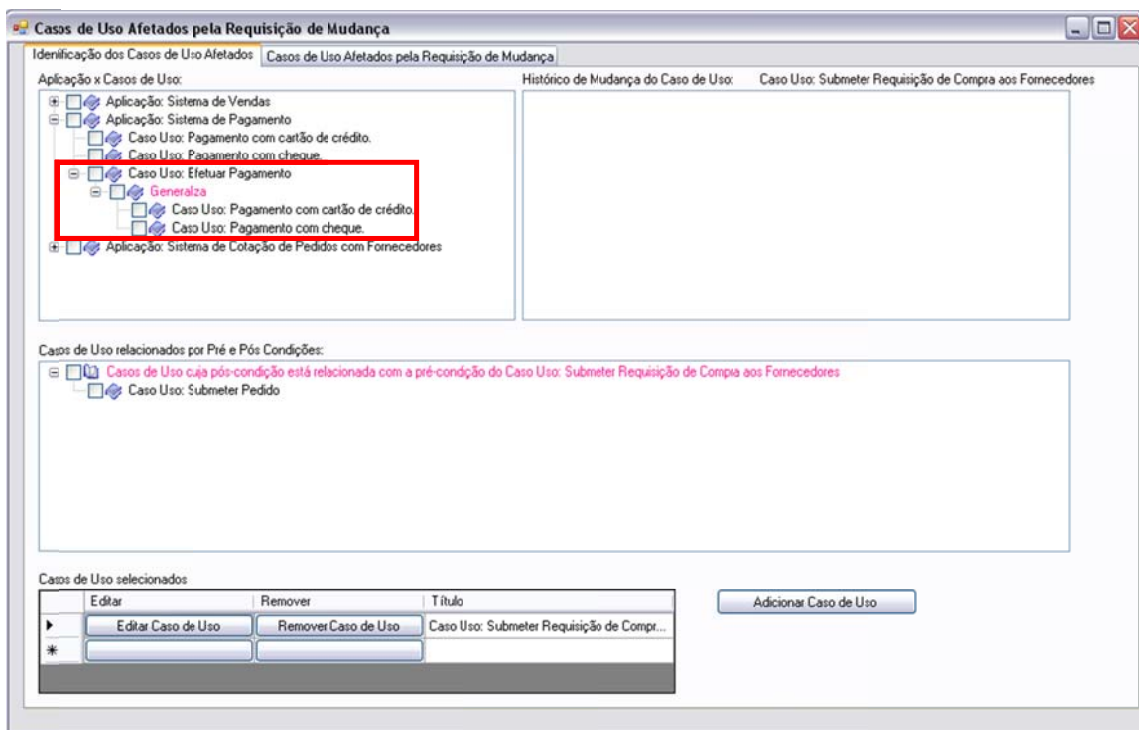


Figura 18. Representação do relacionamento de Generalização.

Árvore “**Histórico de Mudança do Caso de Uso**”: O objetivo desta árvore é mostrar ao analista o histórico de modificação de uma aplicação ou de um caso de uso selecionado. Então quando uma aplicação ou um caso de uso da árvore “**Aplicação x Casos de Uso**” é selecionado, o seu histórico de mudanças é mostrado na árvore “**Histórico de Mudança do Caso de Uso**”. O primeiro nível desta árvore contém a lista das requisições de mudanças que afetaram a aplicação ou o caso de uso. O segundo nível contém a lista dos casos de uso (exceto o caso de uso selecionado) afetados por cada requisição de mudança. Para cada caso

de uso selecionado na árvore “**Histórico de Mudança do Caso de Uso**”, o sistema mostra abaixo do caso de uso uma sub-árvore com o seu histórico de mudança.

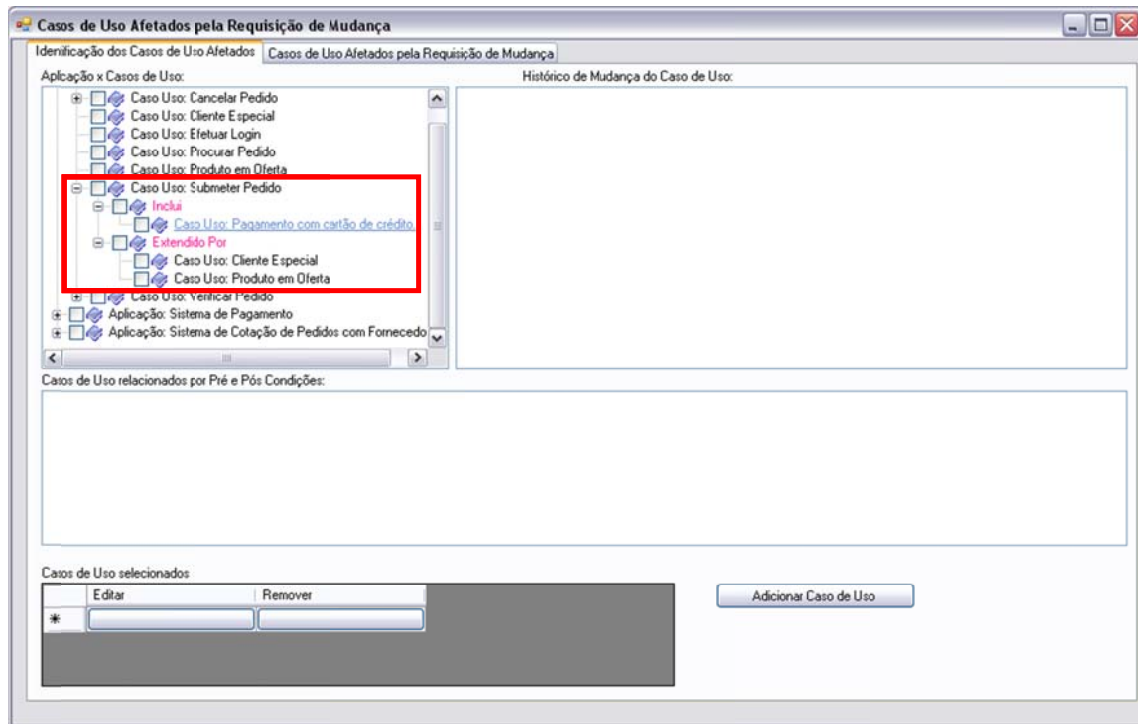


Figura 19. Representação do relacionamento de Extensão.

Árvore “**Casos de Uso relacionados por Pré e Pós-Condições**”: O objetivo desta árvore é mostrar ao analista os casos de uso relacionados por pré e pós-condições com um caso de uso selecionado. Quando um caso de uso da árvore “**Aplicação x Casos de Uso**” é selecionado, o sistema mostra duas listas com as seguintes informações: os casos de uso cuja pós-condição está relacionada com a pré-condição do caso de uso selecionado (casos de uso que precedem o caso de uso selecionado); e os casos de uso cuja pré-condição está relacionada com a pós-condição do caso de uso selecionado (casos de uso que precedem o caso de uso selecionado).

Funcionalidades de **Adicionar**, **Editar** e **Remover** um caso de uso: Quando o analista identifica que um caso de uso deve ser editado ou removido pela requisição de mudança, ele deve selecionar o caso de uso da árvore “**Aplicação x Casos de Uso**” ou da árvore “**Histórico de Mudança do Caso de Uso**”. Cada caso de uso selecionado é adicionado à lista “**Casos de Uso selecionados**”. Nesta lista o analista pode optar por editar ou remover um caso de uso. Quando o analista identifica que para atender a requisição de mudança um novo caso de uso deve ser adicionado, ele deve selecionar o botão “**Adicionar Caso de Uso**”.

Aba “Casos de Uso Afetados pela Requisição de Mudança”: Esta aba mostra a lista dos casos de uso que foram adicionados, alterados ou removidos pela requisição de mudança. Um exemplo desta funcionalidade pode ser visto na Figura 25.

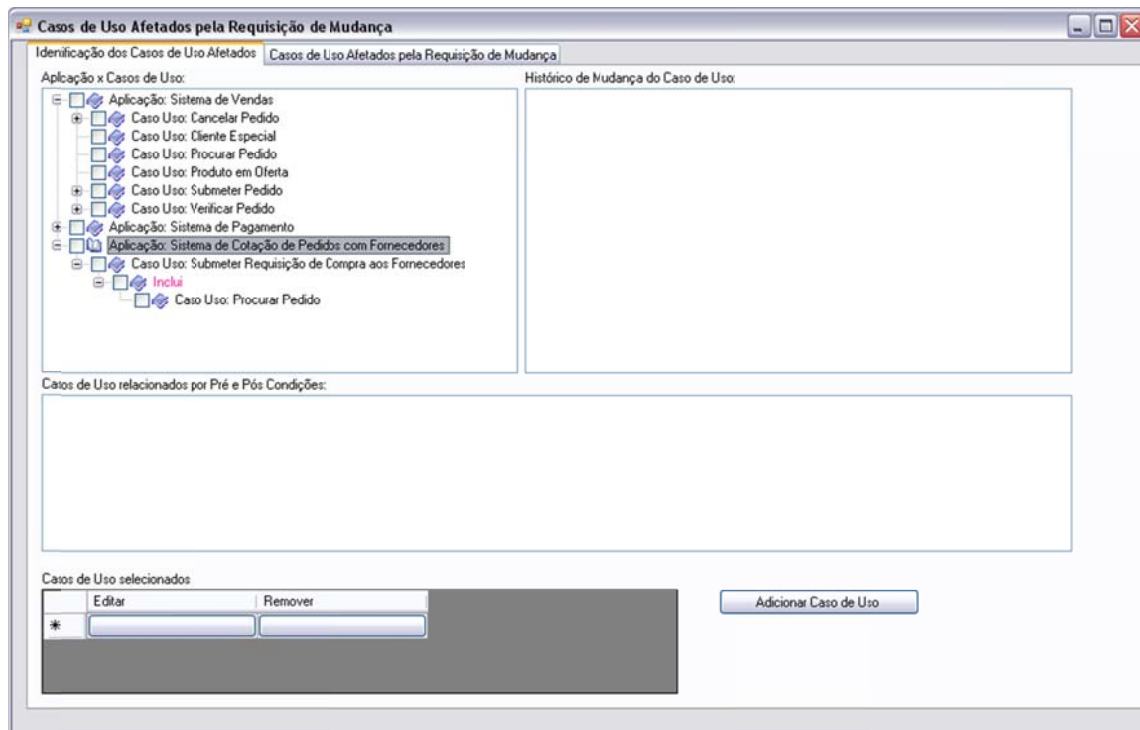


Figura 20. Tela de Casos de Uso Afetados pela Requisição de Mudança.

Cenário 1: Voltando ao exemplo da Figura 17, na qual foi realizado o cadastro da requisição de mudança "Adicionar funcionalidades para requisição e fechamento de compra com fornecedor". Após selecionada a opção de identificar os casos de uso afetados mostrado na Figura 20, o próximo passo é adicionar os casos de uso "Submeter Proposta Requisição de Compra" e "Aceitar Proposta do Fornecedor e Efetuar Compra" para atender à requisição. As Figuras 21 e 22 mostram a adição dos casos de uso "Submeter Proposta Requisição de Compra" e "Aceitar Proposta do Fornecedor e Efetuar Compra". A Figura 23 mostra a identificação da pré-condição do caso de uso "Aceitar Proposta do Fornecedor e Efetuar Compra" que é vinculada com a pós-condição do caso de uso "Submeter Proposta Requisição de Compra".

Aplicação: Sistema de Cotação de Pedidos com Fornecedores

Objetivo: Fornecedor submete proposta para atender requisição de compra

Título: Submeter Proposta Requisição de Compra

Atores: Fornecedor

Pré-condições

Objeto	Estado
Requisição de compra	foi submetida aos fornecedores.

Pós-Condições

Objeto	Estado
Proposta	foi submetida ao funcionário.

Importancia:

Comentarios:

Frequencia: 1

Cenário Principal: Adicionar Cenário Principal

Cenários Alternativos: Adicionar Cenários Alternativos

Requisitos Não Funcionais: Adicionar Requisitos Não Funcionais

Salvar Caso de Uso

Figura 21. Adição do caso de uso "Submeter Proposta Requisição de Compra".

Após adicionados os casos de uso "Submeter Proposta Requisição de Compra" e "Aceitar Proposta do Fornecedor e Efetuar Compra", o sistema atualiza a tela "Casos de Uso Afetados pela Requisição de Mudança" como mostrado na Figura 24. Os casos de uso foram adicionados à árvore "**Aplicação x Casos de Uso**". O caso de uso "Aceitar Proposta do Fornecedor e Efetuar Compra" inclui o caso de uso "Pagamento com cartão de crédito" da aplicação Sistema de Pagamento. Quando o caso de uso "Aceitar Proposta do Fornecedor e Efetuar Compra" é selecionado, o sistema mostra a relação por pré-condição que o mesmo tem com o caso de uso "Submeter Proposta Requisição de Compra". Neste caso, a pré-condição ("Proposta foi submetida ao funcionário") do caso de uso "Aceitar Proposta do Fornecedor e Efetuar Compra" está relacionada com a pós-condição ("Proposta foi submetida ao funcionário") do caso de uso "Submeter Proposta Requisição de Compra".

Outra observação que deve ser feita na Figura 24 é que o sistema está mostrando o histórico de mudanças do caso de uso selecionado "Aceitar Proposta do Fornecedor e Efetuar Compra". A única requisição de mudança que afetou este caso de uso foi a requisição de mudança "Adicionar funcionalidades para requisição e fechamento de compra com fornecedor". Abaixo da requisição de mudança o sistema mostra os outros casos de uso

afetados pela mesma. Neste caso, a requisição de mudança também afetou o caso de uso “Submeter Proposta Requisição de Compra”.

Adicionar Caso de Uso

Aplicação: Sistema de Cotação de Pedidos com Fornecedores

Objetivo: Funcionário aceita proposta do fornecedor e efetua a compra.

Título: Processar Proposta

Atores: Funcionário

Pré-condições: Identificar Pré-Condições

Objeto	Estado
Proposta	foi submetida ao funcionário.

Pós-Condições: Identificar Pós-Condições

Objeto	Estado
Compra	é efetuada

Importancia:

Comentarios:

Frequencia: 1

Cenário Principal: Adicionar Cenário Principal

Cenários Alternativos: Adicionar Cenários Alternativos

Requisitos Não Funcionais: Adicionar Requisitos Não Funcionais

Salvar Caso de Uso

Figura 22. Adição do caso de uso "Aceitar Proposta do Fornecedor e Efetuar Compra".

Cenário 2: Outro cenário para demonstrar como a proposta suporta a identificação dos requisitos afetados por uma requisição de mudança é ilustrado nas Figuras 26, 27 e 28. Digamos que o analista receba uma requisição de mudança descrevendo que o usuário do Sistema de Vendas não precisa mais estar autenticado no sistema para submeter pedidos de compra conforme mostra a Figura 26. O sistema informa, na Figura 27, que a pré-condição do caso de uso “Submeter Pedido” está relacionada com a pós-condição do caso de uso “Efetuar Login”.

Neste caso, para atender esta requisição de mudança deve-se remover a pré-condição do caso de uso “Submeter Pedido” que está vinculada com a pós-condição do caso de uso “Efetuar Login”. A Figura 28 mostra como ficam as relações por pré e pós-condições do caso de uso “Submeter Pedido” após as modificações para atender a requisição de mudança.

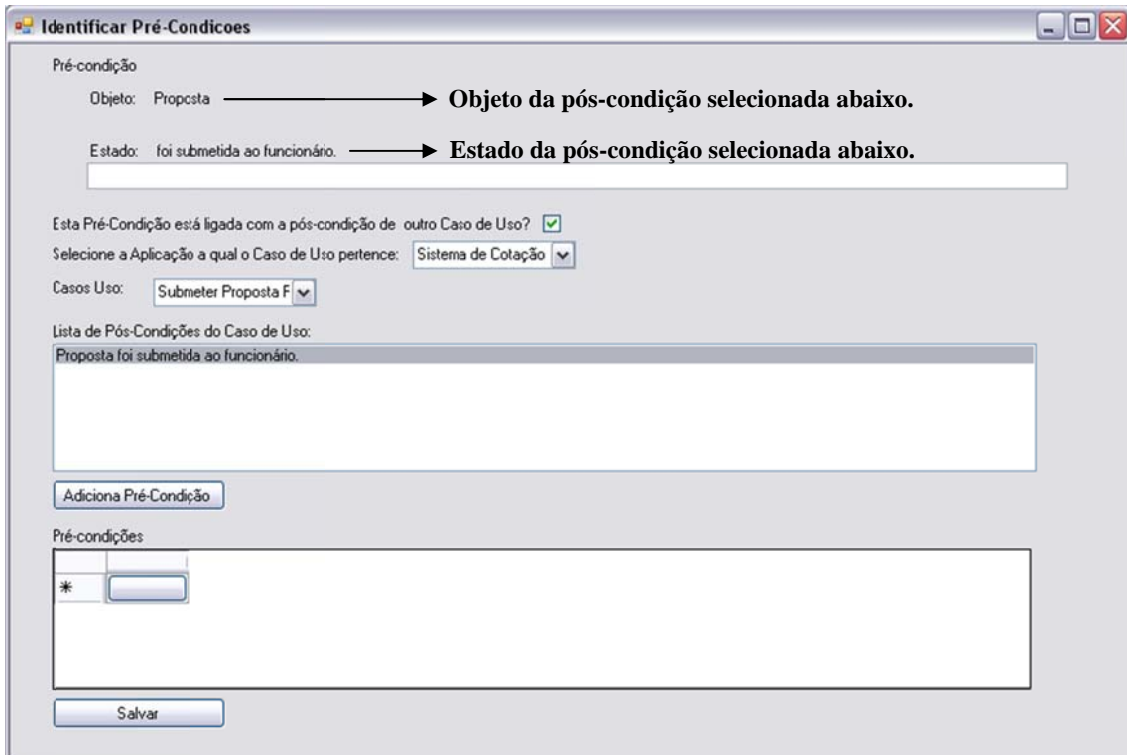


Figura 23. Identificação da pré-condição do caso de uso "Aceitar Proposta do Fornecedor e Efetuar Compra".

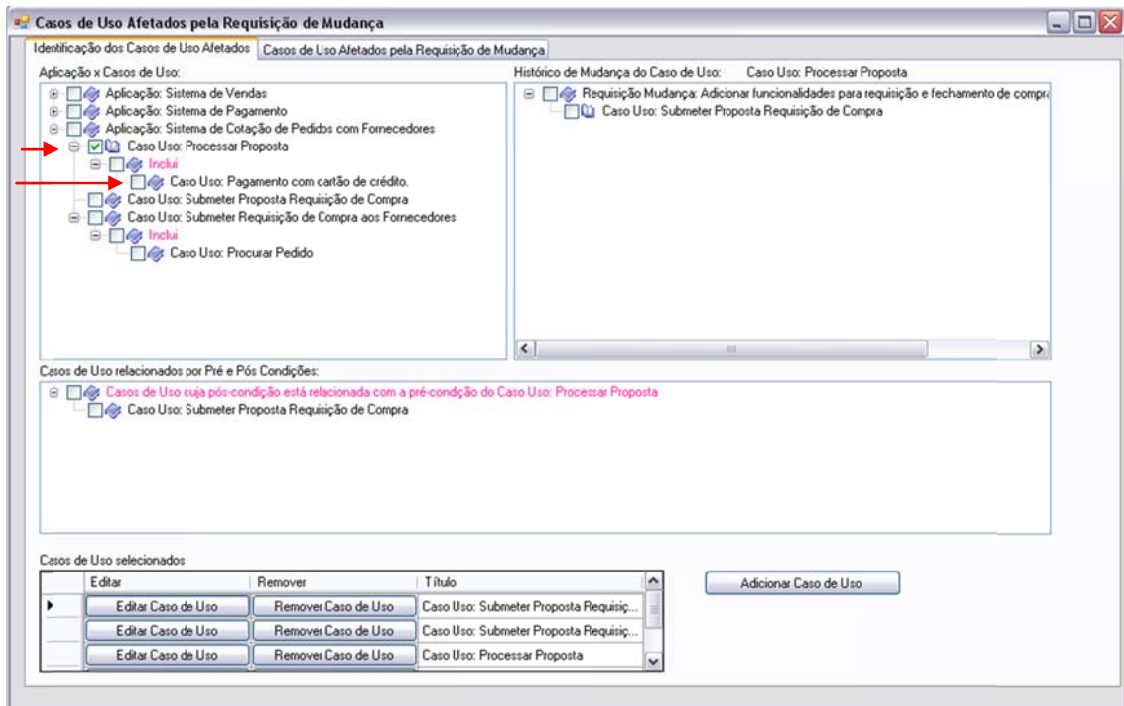


Figura 24. Casos de Uso Adicionados pela Requisição de Mudança.

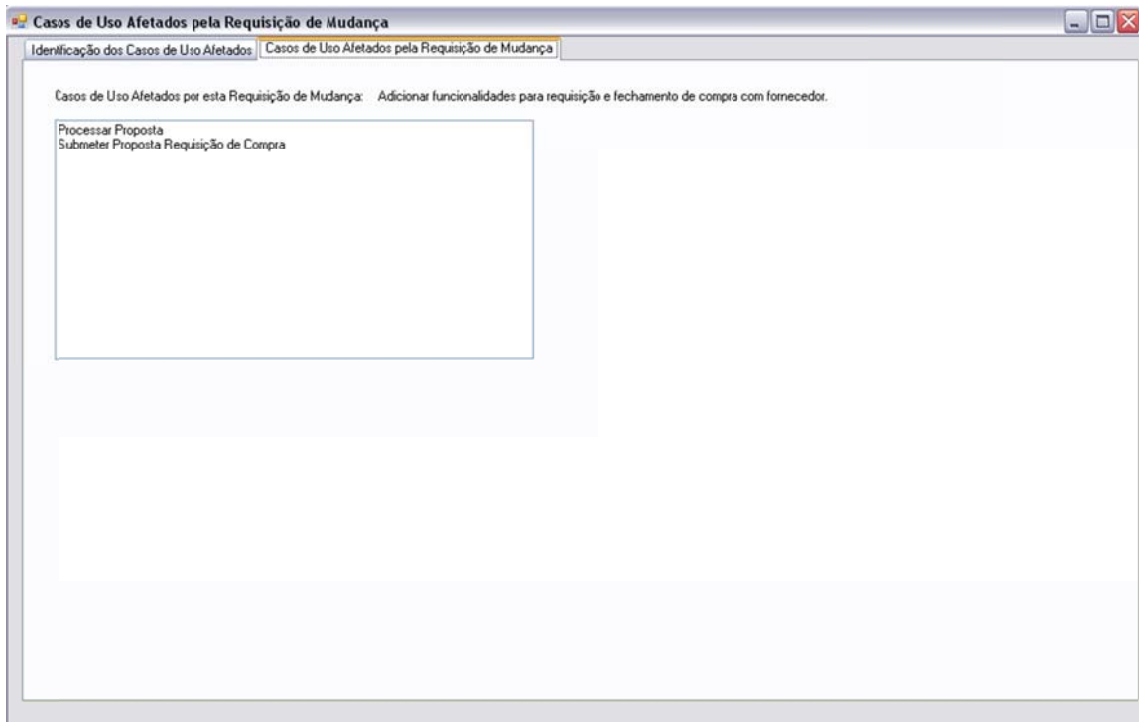


Figura 25. Casos de Uso afetados pela Requisição de Mudança "Adicionar funcionalidades para requisição e fechamento de compra com fornecedor".

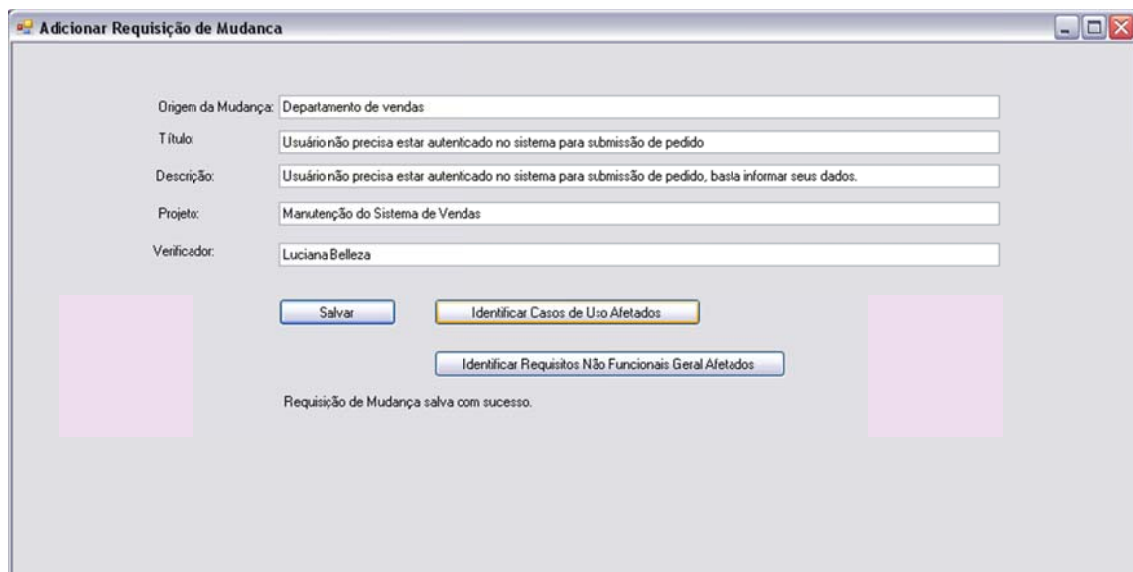


Figura 26. Requisição de Mudança "Usuário não precisa estar autenticado no sistema para submissão de pedido".

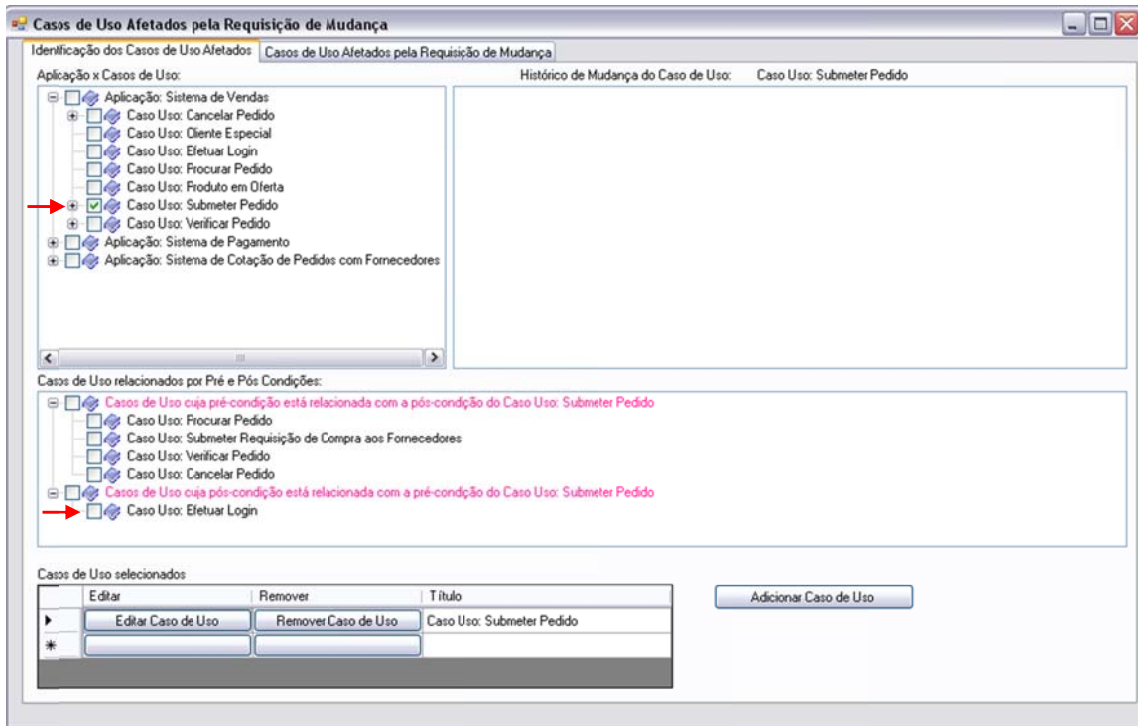


Figura 27. Requisição de Mudança "Usuário não precisa estar autenticado no sistema para submissão de pedido".

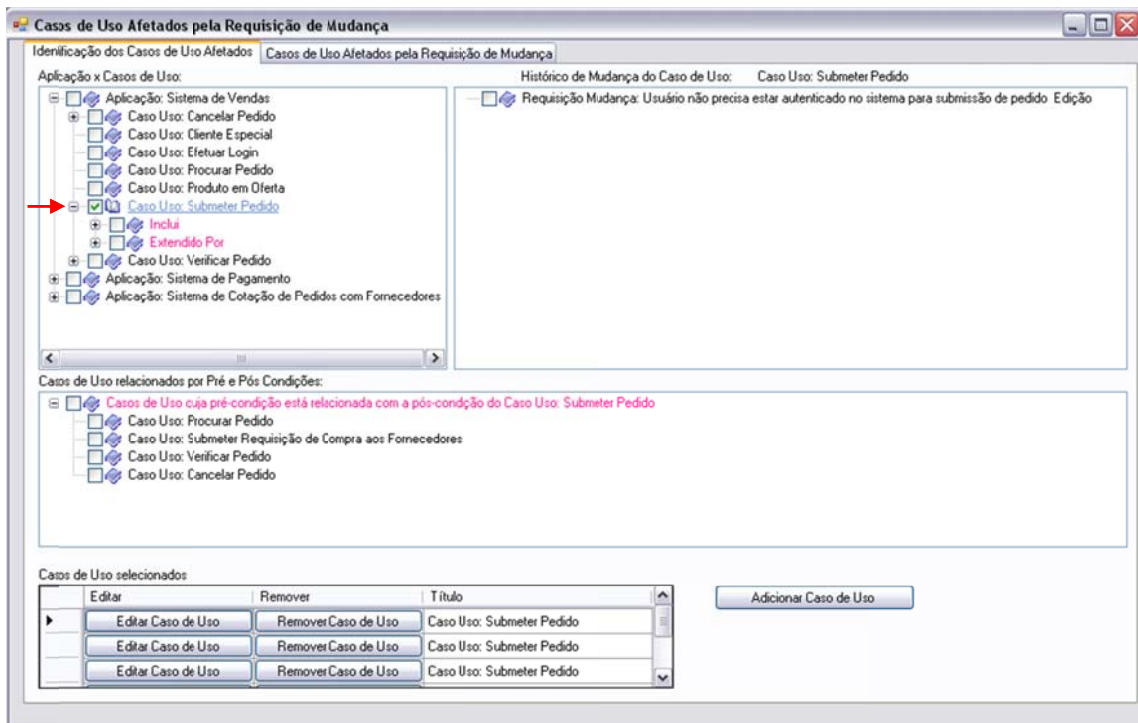


Figura 28. Requisição de Mudança "Usuário não precisa estar autenticado no sistema para submissão de pedido" atendida.

Cenário 3: Outro cenário a ser demonstrado é quando o analista recebe uma requisição de mudança que tem como consequência a remoção de casos de uso.

No exemplo da Figura 29 o analista recebeu duas requisições de mudança. A primeira descrevia que o sistema de pagamento não devia mais permitir o pagamento com cheque. Para esta requisição ser atendida, o caso de uso “Efetuar Pagamento com Cheque” foi removido. A segunda requisição de mudança recebida descrevia que o sistema não devia mais permitir a criação de crediários. Para esta requisição ser atendida, o caso de uso “Criar Crediário para Cliente” foi removido. Respeitando a **regra de consistência 3** descrita na seção 4.2.2, o sistema também removeu o caso de uso “Verificar Cadastro do Cliente no SPC”. Deve-se observar que o sistema não remove os casos de uso fisicamente. Os casos de uso removidos continuam disponíveis para consulta.

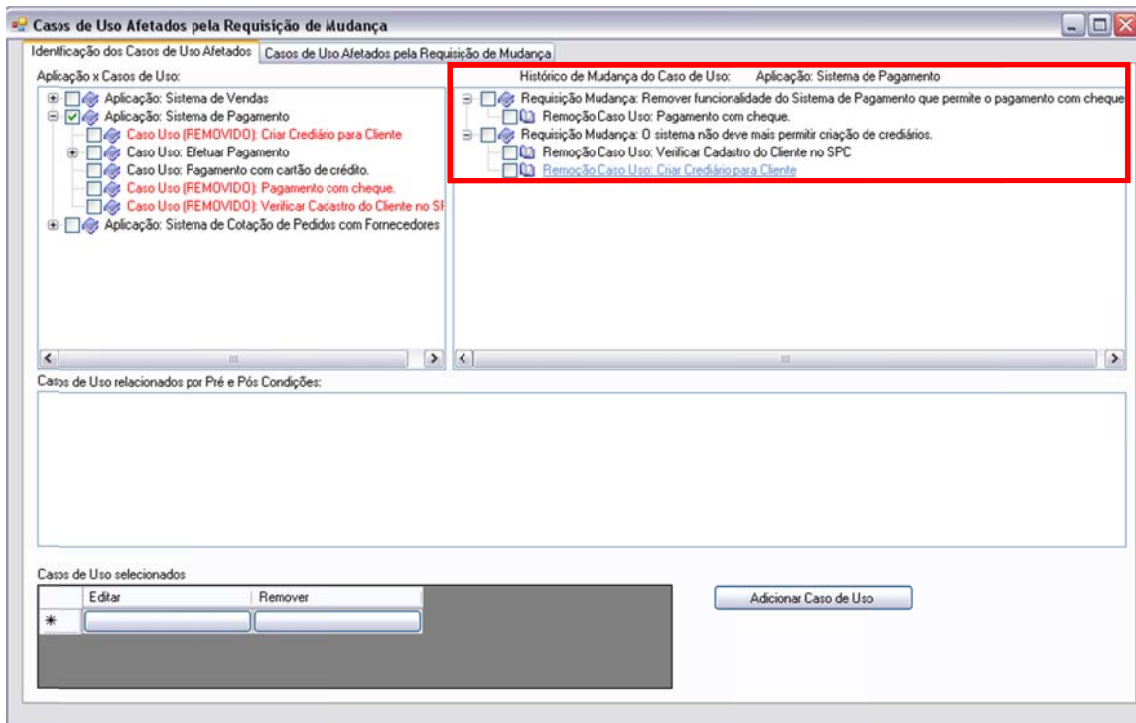


Figura 29. Requisições de Mudança que removem casos de uso.

Cenário 4: Segue um outro cenário para ilustrar como a proposta auxilia na análise de impacto. Supondo-se que chegue uma requisição de mudança com a seguinte descrição “Funcionário deve garantir que a proposta do fornecedor contém o preço de cada produto com o preço à vista e com o preço a prazo.” Considerando que esta requisição trata-se de uma mudança no recebimento da proposta do fornecedor, o analista deve identificar que o caso de uso “Aceitar Proposta do Fornecedor e Efetuar Compra” pode sofrer uma alteração. Sendo que o caso de uso “Aceitar Proposta do Fornecedor e Efetuar Compra” está relacionado com o caso de uso “Submeter Proposta Requisição de Compra” (ver Figura 24), existe grande

probabilidade da alteração também afetar este caso de uso. Ambos os casos de uso manipulam o mesmo objeto “proposta”, que é alvo da requisição de mudança recebida.

5.3.2 Manutenção dos requisitos atualizados e consistentes conforme as requisições de mudanças

Como já foi demonstrado nos exemplos ilustrados na seção anterior como o sistema mantém os requisitos atualizados de acordo com as requisições de mudanças, nesta seção se dará enfoque a como o sistema mantém a consistência destes requisitos (**requisito 3**). Este trabalho se propõe a suportar a consistência respeitando as regras apresentadas na seção 4.2.2.

Para exemplificar a **regra de consistência 1**, que descreve que o sistema não deve permitir a remoção da pós-condição de um caso de uso que está associada com a pré-condição de outro caso de uso, seguem as Figuras 30, 31 e 32.

Conforme mostra a Figura 30, a pós-condição do caso de uso “Submeter Pedido” é pré-condição dos casos de uso “Procurar Pedido”, “Submeter Requisição de Compra aos Fornecedores”, “Verificar Pedido” e “Cancelar Pedido”. Se o analista receber uma requisição de mudança e identificar que para atendê-la deve remover a pós-condição do caso de uso “Submeter Pedido”, o sistema mostrará uma mensagem de erro informando que a operação não é possível conforme mostra a Figura 32.

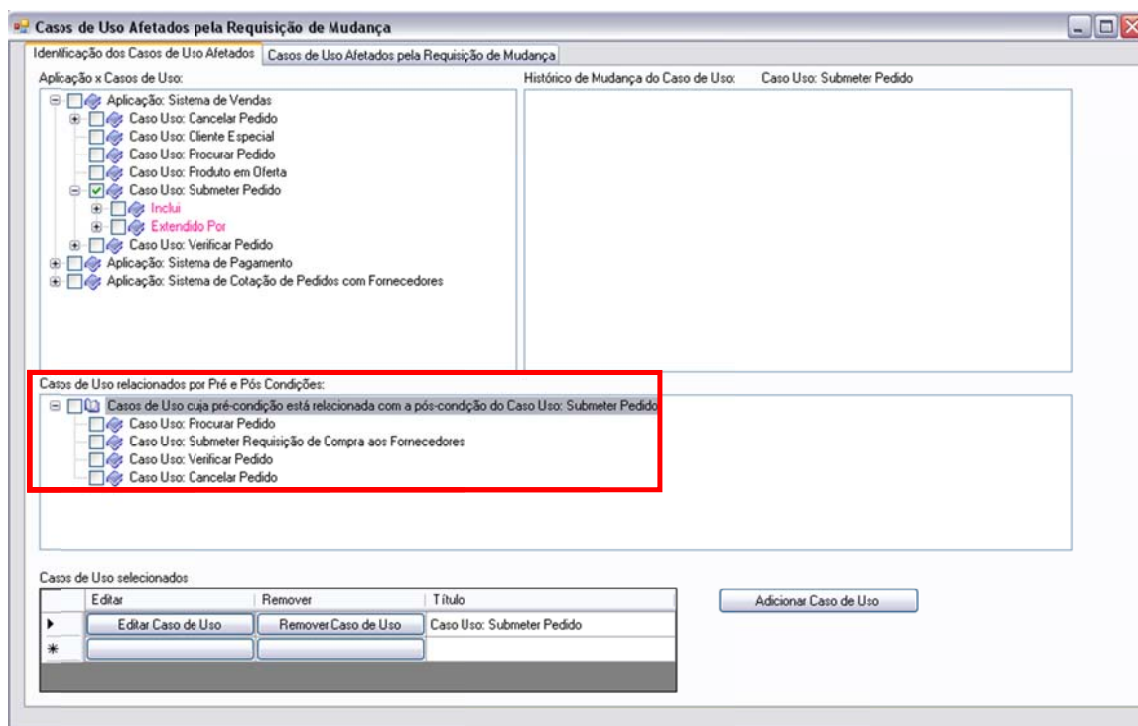


Figura 30. Relações por pré e pós-condições do caso de Uso “Submeter Pedido”.

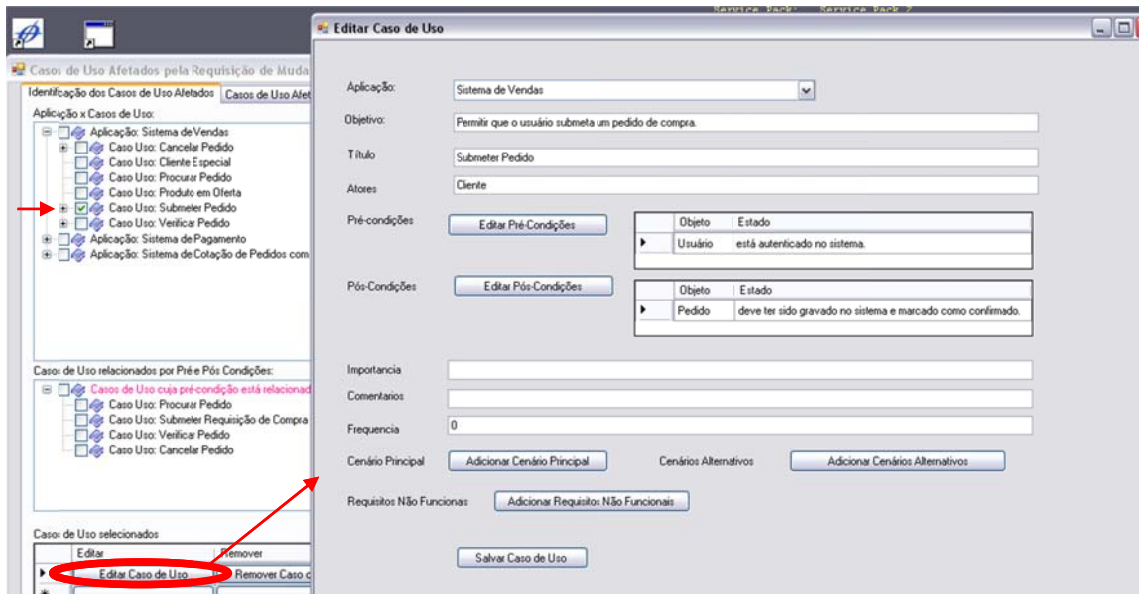


Figura 31. Edição caso de Uso “Submeter Pedido”.

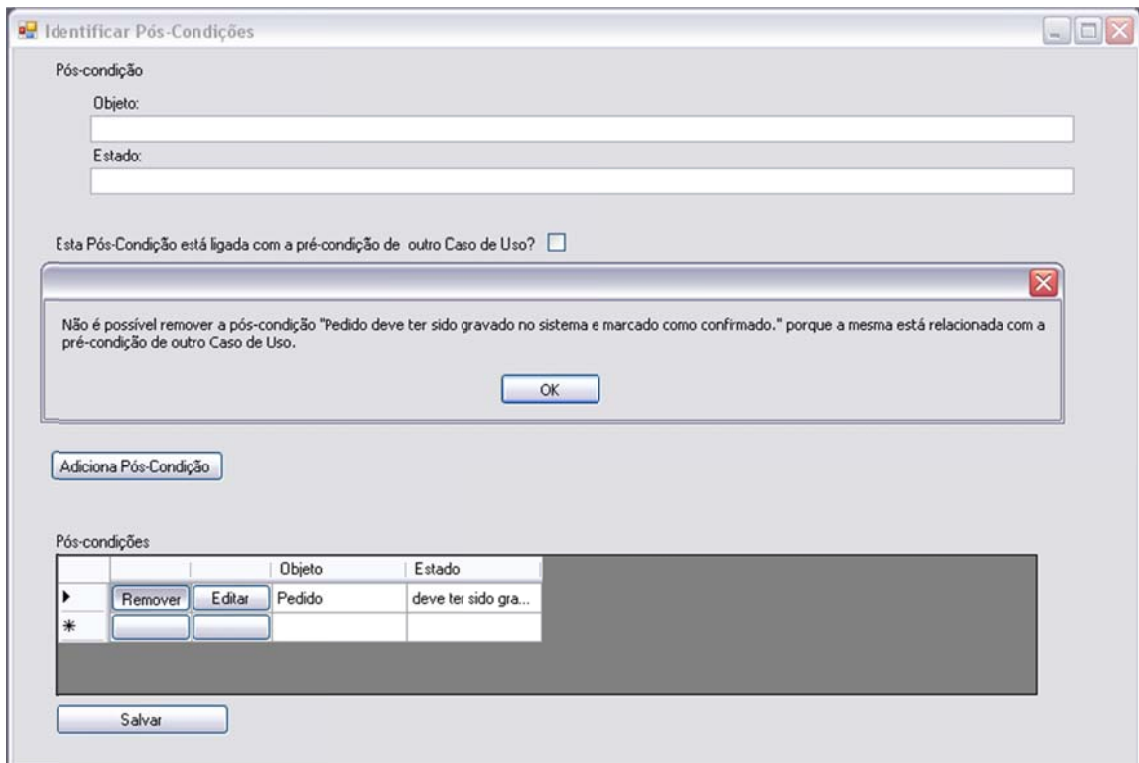


Figura 32. Mensagem de erro na remoção da pós-condição do caso de Uso “Submeter Pedido”.

A **regra de consistência 2**, que descreve que o sistema de gerência de requisitos não deve permitir a remoção de um caso de uso cuja pós-condição é pré-condição de outro caso de uso é demonstrada na Figura 33.

A **regra de consistência 3**, que descreve que o sistema, na remoção de um caso de uso deve remover também o caso de uso incluído, caso o mesmo não possua relação com ator ou com nenhum outro caso de uso, já foi demonstrada no exemplo da Figura 29 (cenário 3 da seção anterior). A **regra de consistência 4** é muito semelhante à regra 3 com a exceção de que a remoção é de um passo do caso de uso que inclui, e não do caso de uso em si. As demais regras (**regras de consistência 5, 6, 7 e 8**) não serão demonstradas por serem muito semelhantes às regras 3 e 4, com a diferença de tratarem dos outros tipos de relacionamentos da UML: Extensão e Generalização.

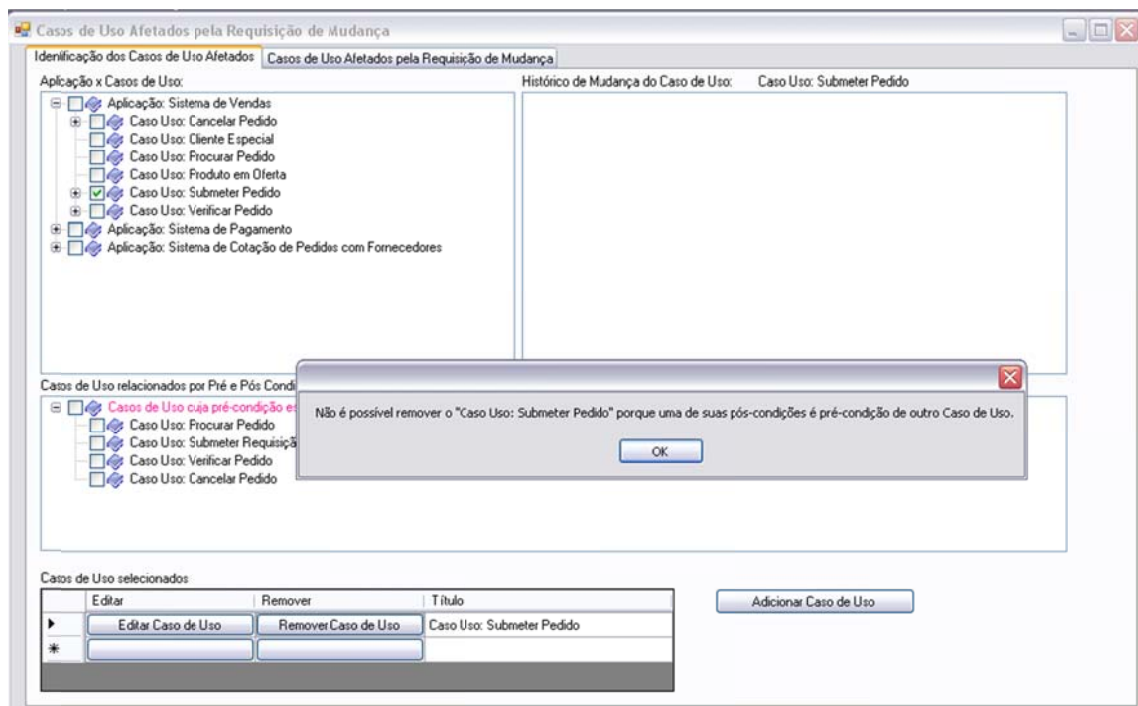


Figura 33. Mensagem de erro na remoção do caso de Uso “Submeter Pedido”.

5.3.3 Manter a rastreabilidade das mudanças dos requisitos de aplicação

O sistema permite que o analista visualize o histórico de mudanças de uma aplicação. Este histórico contém informações como por quais requisições de mudança a aplicação foi afetada, mostrando, para cada requisição de mudança, quais os requisitos modificados e de que forma eles foram modificados. Nesta seção será demonstrado como a solução suporta o **requisito 4**.

Na seção 5.3.1 foi demonstrada a adição dos casos de uso "Submeter Proposta Requisição de Compra" e "Aceitar Proposta do Fornecedor e Efetuar Compra" para atender à requisição de mudança "Adicionar funcionalidades para requisição e fechamento de compra com fornecedor" (cenário 1). Na Figura 34 pode-se observar na árvore **“Histórico de Mudança do Caso de Uso”**, o histórico da aplicação “Sistema de Cotação de Pedidos com Fornecedores” após as modificações pela requisição de mudança. O primeiro nível da árvore mostra as requisições de mudanças que afetaram requisitos deste sistema. No segundo nível, é mostrada a lista dos casos de uso afetados. Antes do nome do caso de uso é mostrada a descrição do tipo de modificação sofrida: Remoção, Alteração ou Adição. Outro exemplo de histórico da aplicação foi mostrado na Figura 29, com a diferença que ao invés de adição de casos de uso, neste exemplo foi realizada a remoção de casos de uso do “Sistema de Pagamento”.

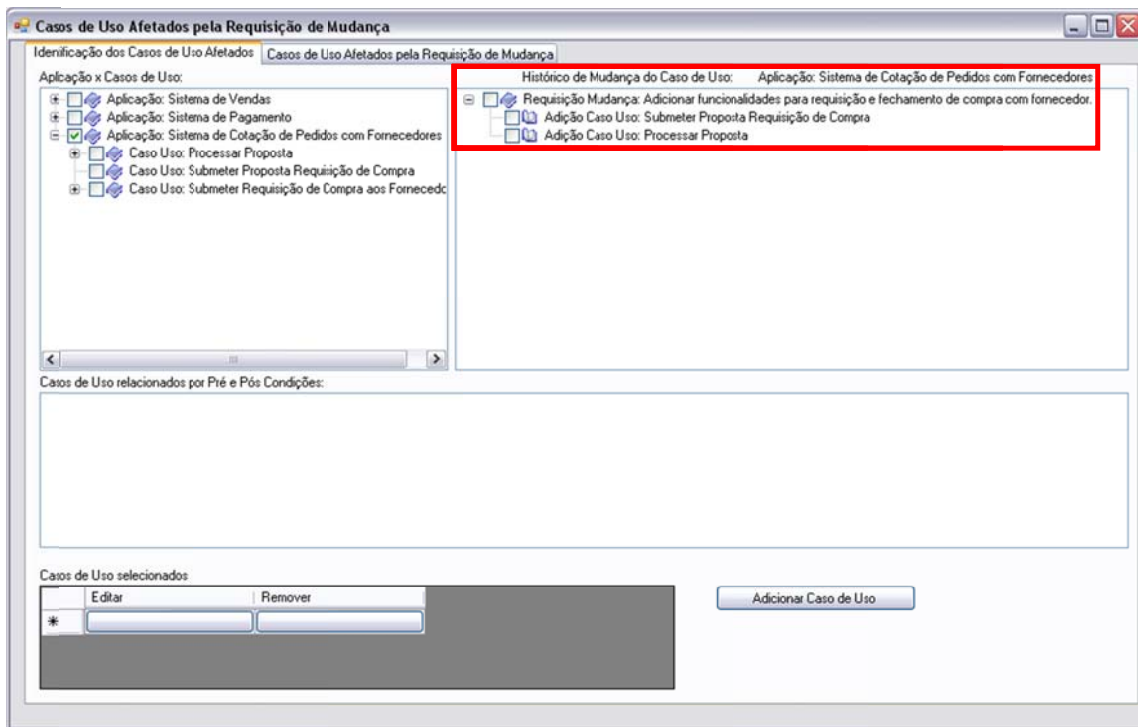


Figura 34. Histórico de mudança da aplicação “Sistema de Cotação de Pedidos com Fornecedores”.

Outra funcionalidade do sistema de gerência de requisitos proposto neste trabalho é o histórico dos requisitos. O analista pode visualizar as alterações do requisito desde sua criação até sua remoção.

Outro exemplo da seção 5.3.1, mostrado nas Figuras 26, 27 e 28, foi a alteração do caso de uso "Submeter Pedido" para atender a requisição de mudança "Usuário não precisa

estar autenticado no sistema para submissão de pedido". Neste caso, após a alteração do caso de uso "Submeter Pedido" o sistema gerou uma nova versão do mesmo. Este histórico do caso de uso está ilustrado na Figura 35. Quando o analista selecionar o botão "Ver Versão" ele pode visualizar os detalhes do caso de uso associado com a versão da linha em que o botão se encontra, conforme mostrado na Figura 36. Nesta figura pode-se observar que o caso de uso na sua primeira versão tinha a pré-condição "Usuário está autenticado no sistema". Enquanto que na sua segunda versão esta pré-condição foi removida.

Neste modelo proposto, cada versão do caso de uso está associada com uma requisição de mudança, exceto se o caso de uso não foi criado na fase de manutenção da aplicação. Para cada requisição de mudança que afeta requisitos, uma nova versão de cada requisito afetado será gerada e a rastreabilidade entre a requisição de mudança e a nova versão do requisito é mantida.

Ver	Versao	Título	Frequencia	DescricaoVersao	Atores
▶ Ver Versão	1	Submeter Pedido	0	Criação do Caso ...	Cliente
Ver Versão	2	Submeter Pedido	0	Edição	Cliente
* Ver					

Figura 35. Histórico do Caso de Uso "Submeter Pedido".

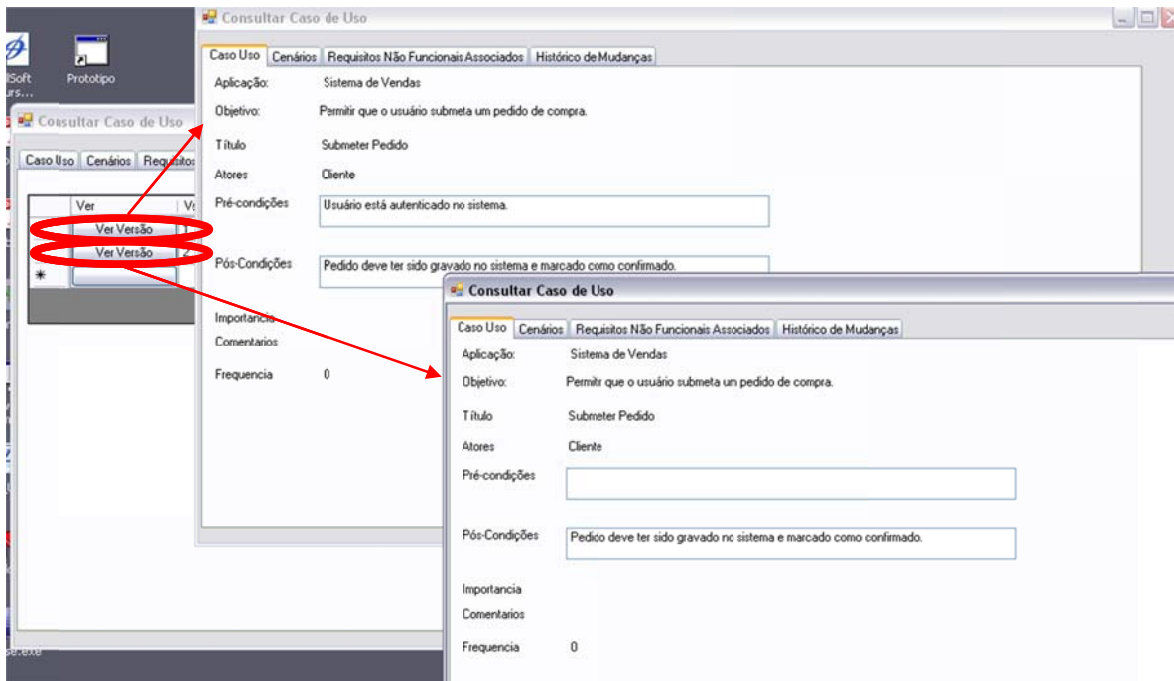


Figura 36. Detalhes das versões do Caso de Uso “Submeter Pedido”.

5.3.4 Reuso de Casos de Uso

O modelo proposto também provê a possibilidade de reuso dos casos de uso entre as diversas aplicações que auxilia no atendimento dos **requisitos 1 e 3**. No Modelo Conceitual apresentado na seção 4.2.1, um caso de uso pode estar associado com uma ou mais aplicações. Nesta seção será apresentado um exemplo de como o sistema suporta este reuso.

Na Figura 37 o caso de uso “Submeter Requisição de Compra aos Fornecedores” da aplicação “Sistema de Compra de Fornecedores” inclui o caso de uso “Procurar Pedido” da aplicação “Sistema de Vendas”. Da mesma forma, um caso de uso de uma aplicação pode ser estendido ou generalizado por um caso de uso de outra aplicação.

A Figura 38 mostra outro exemplo de reuso, porém através do relacionamento por pré e pós-condições. O caso de uso “Submeter Pedido” da aplicação “Sistema de Vendas” está relacionado por sua pós-condição com o caso de uso “Submeter Requisição de Compra aos Forecedores” da aplicação “Sistema de Compra de Fornecedores”.

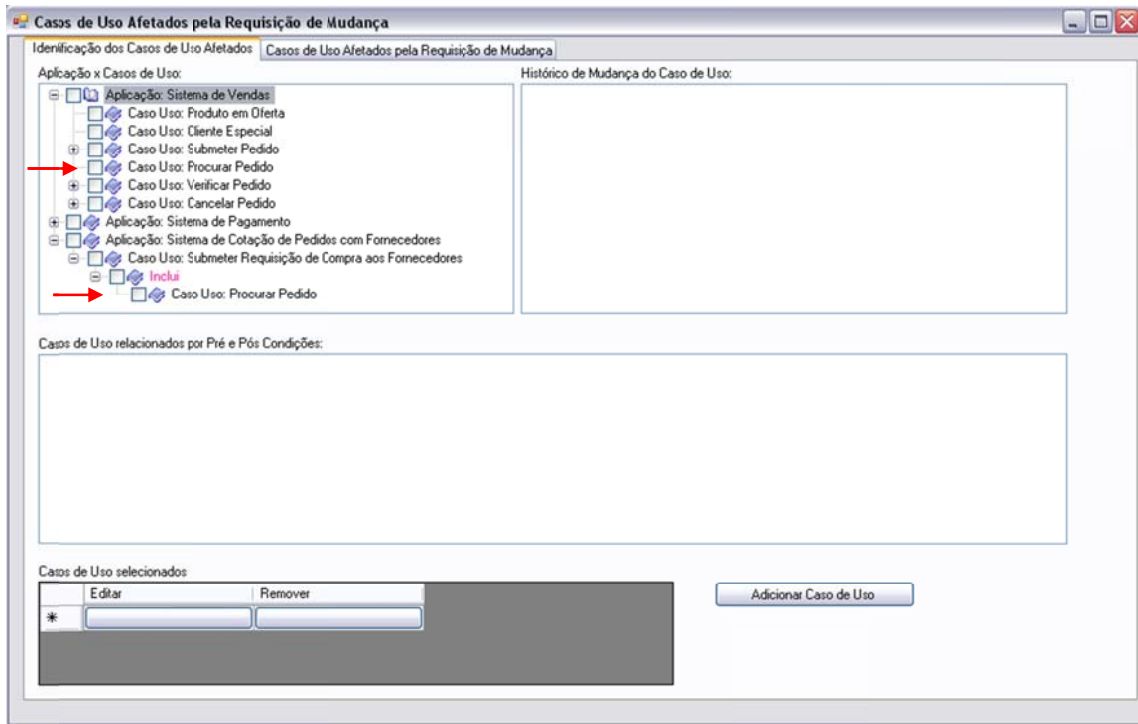


Figura 37. Exemplo de inclusão de caso de uso de outra aplicação.

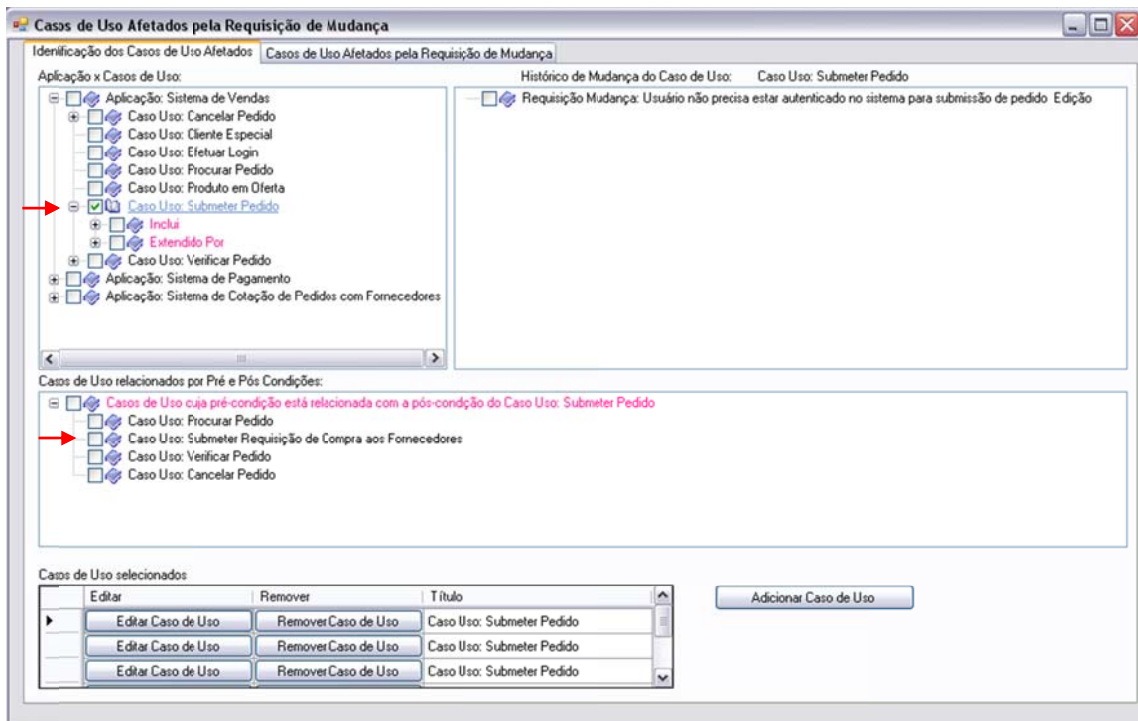


Figura 38. Requisição de Mudança “atendida”.

5.4 Considerações do capítulo

Neste capítulo demonstrou-se como o protótipo, instância do modelo de gerência de requisitos proposto neste trabalho, suporta a análise do impacto das requisições de mudança sobre requisitos de diversas aplicações, a atualização, e a manutenção da consistência destes requisitos.

Os exemplos demonstraram como a solução atende o **requisito 1**, que trata da identificação dos requisitos afetados por uma requisição de mudança (análise de impacto), o **requisito 3**, que consiste na manutenção dos requisitos atualizados e consistentes conforme as requisições de mudanças, e o **requisito 4**, que se refere a como manter a rastreabilidade das mudanças dos requisitos de aplicação, definidos na seção seção 4.1.2 deste trabalho.

Embora não tenham sido apresentados todos os exemplos possíveis da aplicação do modelo proposto neste trabalho, os apresentados contribuíram facilitando a visualização da proposta e comprovando seu funcionamento diante dos objetivos inicialmente propostos. Mais ilustrações de telas do protótipo podem ser observadas no Apêndice III.

O próximo capítulo apresenta as considerações finais obtidas com o desenvolvimento desta dissertação.

6. CONSIDERAÇÕES FINAIS

A engenharia de requisitos é uma das atividades críticas e mais importantes do processo de desenvolvimento de software, visto que é a partir dela que o sistema final será desenvolvido. Atualmente, é tida como um dos principais desafios na engenharia de software e como a principal razão de muitas das falhas ocorridas nos sistemas.

A manutenção e evolução do *software* caracterizam-se por demandarem custo muito alto das organizações. Se na fase de evolução do *software* os requisitos não forem eficientemente gerenciados, muito tempo pode ser demandado para a manutenção e, no pior caso, informações sobre funcionalidades importantes do sistema podem ser perdidas.

Neste contexto, o tema abordado nesta dissertação busca auxiliar a atividade de gerência dos requisitos durante a evolução do sistema de *software* através de um modelo de sistema de gerência de requisitos que suporte a atualização e a consistência dos requisitos de *software* afetados por requisições de mudança.

O sistema de gerência de requisitos proposto, instanciado através do protótipo apresentado no Capítulo 5, fornece as seguintes principais funcionalidades: manter a rastreabilidade entre requisições de mudanças e os requisitos afetados pelas mesmas; relacionar requisitos entre si pelos relacionamentos definidos na UML (“inclui”, “estende” e “generaliza”) e por pré e pós-condições; o registro do histórico dos requisitos desde a sua criação até o seu término; e o reuso entre requisitos de diferentes aplicações.

Com estas funcionalidades, descritas no capítulo 4 e demonstradas no Capítulo 5, pode-se afirmar que o objetivo geral desta pesquisa foi atingido.

O referencial teórico, apresentado no Capítulo 2, surgiu devido à pesquisa bibliográfica realizada sobre o tema, onde os principais pontos foram considerados e apresentados. O estudo de caso exploratório realizado na empresa auxiliou na compreensão do problema e na identificação das principais dificuldades envolvidas. A instanciação do modelo proposto através do protótipo e o detalhamento do seu comportamento perante os diferentes cenários levantados através dos sistemas de exemplos, apresentados no Capítulo 5, permitiram avaliá-lo para a confirmação do alcance dos objetivos desta pesquisa.

6.1 Contribuições

Uma das principais contribuições desta pesquisa é a possibilidade de manter os requisitos de aplicações de software atualizados ao longo da sua evolução. Os sistemas de gerência de requisitos existentes atualmente no mercado são orientados a projeto. Uma vez terminado o projeto, os requisitos tendem a serem esquecidos e não mais atualizados. Com o modelo de gerência apresentado é possível se obter o histórico de cada requisito de aplicação desde sua criação até o seu término (desativação do ambiente de produção do *software*).

Além de auxiliar na manutenção dos requisitos atualizados, uma outra importante contribuição é o suporte à consistência dos requisitos funcionais representados por casos de uso. No geral os casos de uso são definidos pelos analistas de *software* como texto livre, sem muitas regras de formação. No modelo apresentado neste trabalho o caso de uso é representado por sub-elementos (ator, pré-condições, pós-condições, passos da sequência básica, etc) que auxiliam o analista na descrição dos casos de uso. O modelo também suporta regras de consistência nos relacionamentos entre os casos de uso: inclusão, extensão e generalização, definidos pela UML, e por pré e pós-condições proposto neste trabalho.

Este modelo também traz benefícios para análise de impacto das requisições de mudanças através dos recursos que mostram o histórico dos requisitos e como os requisitos das aplicações estão relacionados. O relacionamento por pré e pós-condições auxilia na identificação dos requisitos associados com um mesmo objeto, além de representar a ordem de seqüência de execução dos casos de uso.

6.2 Limitações do Estudo

Este trabalho atendeu ao objetivo ao qual se propôs, porém existem algumas limitações a serem trabalhadas:

- Análise de impacto automatizada: Apesar de auxiliar na análise de impacto, o modelo proposto só auxilia na identificação dos requisitos afetados se o analista identificar pelo menos um primeiro requisito. A identificação de requisitos afetados pelas requisições de mudanças de forma automática seria um recurso que traria muitos benefícios;
- Controle de concorrência: O modelo proposto não suporta o controle de concorrência descrito pelo **requisito 2** da seção 4.1.2;

- Suporte ao *rollback* das alterações feitas por uma requisição de mudança: Apesar do modelo proposto suportar o *rollback*, não foi desenvolvido o mecanismo de *rollback* das alterações dos requisitos feitas para atender uma requisição de mudança.

6.3 Trabalhos Futuros

Além de atender às limitações apontadas na seção anterior, a partir deste trabalho identificou-se algumas oportunidades de desenvolvimento de trabalhos futuros.

Por exemplo, o reuso dos casos de uso pode ser mais explorado de tal forma que o sistema verifique, quando um analista está alterando um caso de uso utilizado por mais de uma aplicação, até que ponto modificações são aceitáveis para ainda se caracterizar o caso de uso reutilizado. Dependendo das alterações pode ser necessária a criação de outra instância do caso de uso, separando o caso de uso alterado do caso de uso comum as outras aplicações.

Em relação ao suporte à consistência mais avanços podem ser realizados no que diz respeito à definição de regras de formação e consistência na definição dos casos de uso. Com mais regras, evita-se que casos de uso sejam descritos como texto livre, o que facilita o suporte à atualização e consistência dos mesmos pelas ferramentas. Um exemplo de nova regra poderia ser quando o analista remover a pré-condição de um caso de uso, o sistema verificaria que pelo menos um passo deste caso de uso deve ser alterado.

Outra possibilidade de trabalho futuro é a geração do Modelo de Domínio dos sistemas cadastrados a partir dos conceitos descritos nos casos de uso. Além do Modelo de Domínio é desejado que o sistema avance para suportar a rastreabilidade com os outros artefatos do *software* como Diagrama de Classes, Diagrama de Seqüência, etc.

REFERÊNCIAS BIBLIOGRÁFICAS

[AND 2001] Anda, B.; Sjøberg, D.; Jørgensen, M. **Quality and Understandability of Use Case Models**. In: ECOOP 2001 - Object-Oriented Programming, 15th European Conference, p. 18-22, v. 2072, 2001, Budapest, Hungary. Proceedings...Springer Berlin, Heidelberg. 2001. 27p.

[ANQ 2007] Anquetil, N.; Oliveira, K. M.; Sousa, K. D.; Dias, M. G. B. **Software maintenance seen as a knowledge management issue**. Information and Software Technology, v. 49, p. 515-529, 2007. Disponível em: <<http://www.sciencedirect.com>>. Acesso em: 20 nov. 2008.

[AUR 2003] Aurum, A.; Wohlin, C. **The fundamental nature of requirements engineering activities as a decision-making process**. Information and Software Technology, v. 45, p. 945-954, 2003. Disponível em: <<http://www.sciencedirect.com>>. Acesso em: 11 ago. 2008.

[BEL 2007] Belleza, L. **Análise de ferramentas de gerência de requisitos com foco em manutenção contínua**. 2007. 59 f. Trabalho Individual II (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação da Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2007.

[BEN 2000] Bennett, K. H.; Rajlich V. T. **Software Maintenance and Evolution: a Roadmap**. In: International Conference on Software Engineering, 2000, Limerick, Ireland. Proceedings... 2000.

[CER 2007] Cerri, E. **Um modelo de rastreabilidade entre o documento de especificação de requisitos e o modelo de casos de uso do sistema**. 2007. 190 f. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação da Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2007.

[COC 2005] Cockburn, A. **Escrevendo Casos de Uso Eficazes**. Porto Alegre: Bookman Companhia ED, 2005. 254p.

[COL 2004] Collins-Sussman, B.; Fitzpatrick, B. W.; Pilato, C. M. **Version Control with Subversion**. Califórnia: O'Reilly Media, 2004. 320p.

[DAR 2003] Kulak, D; Guiney, E. **Use Cases: Requirements in Context**. 2. ed. New York: Addison-Wesley, 2003. 272p.

[DOR 1990] Dorfman, M.; Thayer, R. **Standards, Guidelines and Examples of System and Software Requirements**. Portland: IEEE Computer Society Press, 1990. 620p.

[DOU 2008] Douglas, C. W. **Visual Language Representation for Use Case Evolution and Traceability**. 2008. 147 p. Dissertação (requisito parcial para Doutorado em Filosofia) - The Department of Computer Science, Louisiana State University, Louisiana, 2008.

[DUR 1999] Durán A.; Bernárdez, B.; Ruiz, A.; Toro M. **A Requirements Elicitation Approach Based in Templates and Patterns**. In: II (Ibero-American) Workshop on Requirements Engineering, 1999, Buenos Aires. Argentina, 1999.

[EAS 2004] Easterbrook, S. **What is Requirements Engineering?**. 2004. Disponível em: <<http://www.cs.toronto.edu/~sme/papers/2004/FoRE-chapter01-v7.pdf>> Acesso em: 20 maio. 2007.

[ESP 2006] Espindola, R. **Uma Arquitetura de Informação para Gerência de Requisitos em Desenvolvimento Distribuído de Software**. 2006. 125p. Dissertação de Mestrado. FACIN – PPGCC, PUCRS, Porto Alegre, 2006.

[FJE 1982] Fjeldstad, R. K.; Hamlen, W. T. **Application Program Maintenance Study: Report to Our Respondents**. G. Parikh, N. Zvegintzov eds. Tutorial on Software Maintenance, IEEE Computer Society Press, pp. 13 – 30. 1982, Los Alamitos, CA.

[FIN 2000] Finkelstein, A.; Emmerich, W. **The Future of Requirements Management Tools**. Information Systems in Public Administration and Law. G. Quirchmayr, R. Wagner and M. Wimmer. Eds.: Oesterreichische Computer Gesellschaft. Áustria, 2000.

[GOT 1994] Gotel, O.; Finkelstein, A. **An analysis of the requirements traceability problem**. In: 1st International Conference on Requirements Engineering, 101., 1994, Colorado, CO. Proceedings... Califórnia: IEEE Computer Society Press, 1994.

[IEE 1998] IEEE Std 830-1998. **IEEE Recommended Practice for Software Requirements Specifications**. New York: Institute of Electrical and Electronic Engineers, 1998. 38p.

[JAC 1998] Jacobson, I.; Grady, B.; Rumbaugh, J. **The Unified Software Development Process**. New York: Addison-Wesley, 1998. 463p.

[KOT 1998] Kotonya, G.; Sommerville, I. **Requirements Engineering: process and techniques**. New York: John Wiley & Sons Ltd, 1998. 282p.

[KRU 2000] Kruchten, P. **The Rational Unified Process: An Introduction**. Upper Saddle River, 2. ed. New Jersey: Addison-Wesley, 2000. 320p.

[LAR 2007] Larman, C. **Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado**. trad. Rosana T. Vaccare Braga, Paulo Cesar Masiero, Rosângela Penteadó e Fernão Germano. 3. ed. Porto Alegre : Bookman, 2007. 695p.

[LEF 2000] Leffingwell, D.; Widrig, D. **Managing Software Requirements: A Unified Approach**. 1. ed. England: Addison-Wesley, 2000. 544p.

[LUC 2004] De Lucia, A.; Fasano, F.; Francese, R.; Tortora, G. **ADAMS: An artifact-based process support system**. In: 16th International Conference on Software Engineering and Knowledge Engineering, 2004, Canada. Proceedings... F. Maurer and G. Ruhe, Eds. p. 31–36.

[LUC 2007] De Lucia, A.; Fasano, F.; Oliveto, R.; Tortora, G. **Recovering Traceability Links in Software Artifact Management Systems using Information Retrieval Methods**. Itália: University of Salerno. 2007.

[MOH 2008] Mohan, K.; Xu, P.; Cao, L.; Ramesh, B. **Improving change management in software development: Integrating traceability and software configuration management**. Decision Support Systems. v. 45, p. 922–936, 2008. Disponível em: <<http://www.sciencedirect.com>>. Acesso em: 10 nov. 2008.

[NOL 2007] Noll, R. P.; Ribeiro; M. B. **Ontological Traceability over the Unified Process**. In 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 2007, Tucson, Arizona. Proceedings... 2007.

[NUS 2000] Nuseibeh, B.; Easterbrook, S. **Requirements Engineering: A Roadmap**. Limerick: ACM, Future of Software Engineering, 2000. p. 37-45.

[PAL 2000] Palmer, J. D.. **Traceability In Software Requirements Engineering**. Los Alamitos: 2. ed. R. H. Thayer and M. Dorfman. Eds. IEEE Computer Society Press, 2000. p. 412-422.

[FJE 1982] Fjeldstad, R. K.; Hamlen, W. T. **Application Program Maintenance Study: Report to Our Respondents**. G. Parikh, N. Zvegintzov eds. Tutorial on Software Maintenance, IEEE Computer Society Press, pp. 13 – 30. 1982, Los Alamitos, CA.

[RAJ 1999] Rajlich V.; Varadajan S. **Using the Web for Software Annotations**. Software Engineering and Knowledge Engineering, Singapore, v. 9, p. 55-72. 1999.

[RAJ 2000] Rajlich V. **Modeling Software Evolution by Evolving Interoperation Graphs**. Annals of Software Engineering, J. C. Baltzer AG, Science Publishers, Red Bank, NJ, v. 9. 2000.

[SAD 2007] Sadraei, E.; Aurum, A.; Beydoun, G.; Paech, B. **A field study of the requirements engineering practice in Australian software industry**. Requirements Engineering. Springer-Verlag, New York, 18p. 2007.

[SAY 2005] Sayão, M; Leite, J. C. S. P. **Rastreabilidade de Requisitos**. Revista de Informática Teórica e Aplicada – RITA, Porto Alegre, v. XII, num. 1., 2005.

[SOM 1998] Sommerville, I. **Software Engineering**. Massachusetts: Addison-Wesley, Reading, 1998.

[SOM 2003] Sommerville, I. **Engenharia de Software**. 6. ed. São Paulo: Addison Wesley, 2003. 606p.

[SOM 2005] Sommerville, I. **Integrated Requirements Engineering: A Tutorial**. IEEE Software, v. 22 (1), p. 16-23, Janeiro/Fevereiro. 2005.

[SOMé 2006] Somé, S. S. **Supporting Use Case based Requirements Engineering**. Information and Software Technology, v. 48, p. 43–58. 2006.

[SOMé 2007] Somé, S. S. **Petri Nets Based Formalization of Textual Use Cases**. School of Information Technology and Engineering, University of Ottawa, Ontario, Canada. 2007. Disponível em <http://www.site.uottawa.ca/eng/school/publications/techrep/2007/TR-2007-11.pdf>. Acesso em: 26 abr. 2008.

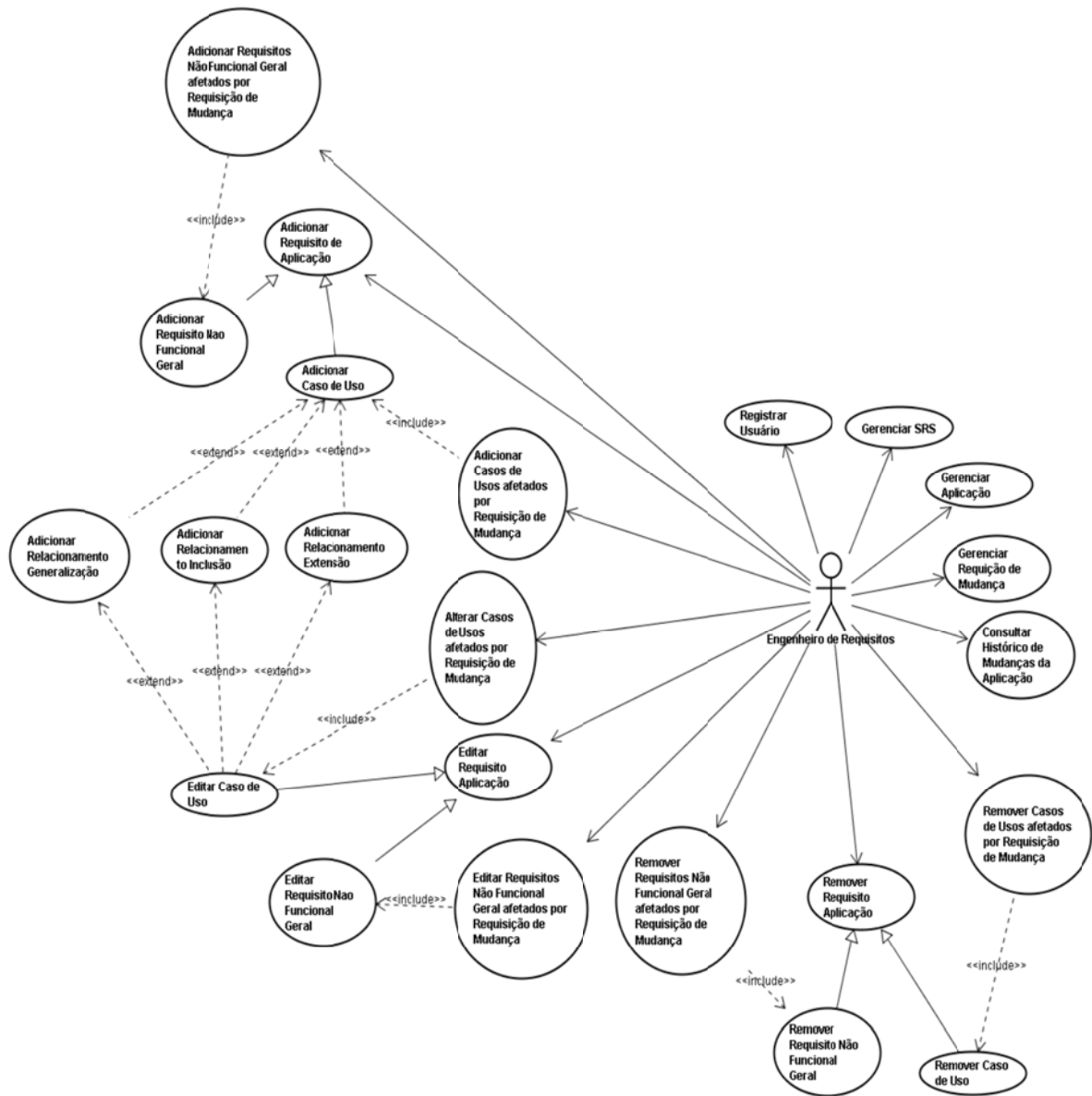
[THA 1990] Thayer, R.; Dorfman, M. **System and Software Requirements Engineering**. IEEE Computer Society Press Tutorial. Los Alamitos, 718p. 1990.

[UML 2007] UML. **OMG Unified Modeling Language (OMG UML): Superstructure**. Version 2.1.2. Object Management Group, 738p. 2007. Disponível em <http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF>. Acesso em: 20 mar. 2008.

[YAU 1978] Yau, S. S.; Collofello J. S.; MacGregor T. **Ripple effect analysis of software maintenance.** In: Compsac, IEEE Computer Society Press, Los Alamitos, CA. Proceedings... p. 60-65. 1978.

APÊNDICE I - DOCUMENTAÇÃO DO PROTÓTIPO

1. Modelo de Casos de Uso do protótipo



2. Especificação dos casos de uso do protótipo

Caso de Uso “Registrar Usuário”

Objetivo: Permitir que o usuário se identifique no sistema.

Atores: Engenheiro de Requisitos

Pré-condição:

Pós-condição de Sucesso: Usuário é registrado no Sistema.

Passos:

- 1- O Usuário acessa o Sistema.
- 2- O Sistema requisita que o usuário informe o nome do usuário.
- 3- O usuário informa seu nome.
- 4- O Sistema registra o Usuário no Sistema.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso “Gerenciar Aplicação”

Objetivo: Permitir que o usuário possa adicionar, consultar, editar e remover aplicações do sistema.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Aplicação foi adicionada, editada ou removida.

Passos:

- 1- O usuário acessa a funcionalidade de cadastro de aplicação.
- 2- O sistema disponibiliza as funções: **Adiciona**, **Edita**, **Remove** ou **Consulta**.
- 3- O usuário solicita a **adição** de uma Aplicação.
- 4- O sistema requisita as seguintes informações para cadastro da nova aplicação:
Nome, Descrição.
- 5- O usuário informa as informações e solicita salvá-las.
- 6- O sistema salva as informações.
- 7- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da adição (usuário registrado) e Data e Hora da adição.

Alternativas:

Alternativa 1: Edição

- 3- O Usuário solicita a **edição** de uma Aplicação.
- 4- O Sistema mostra a lista de aplicações existentes.
- 5- O Usuário escolhe uma aplicação e altera o Nome ou a Descrição.
- 6- O Sistema salva as informações.
- 7- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da edição (usuário registrado) e Data e Hora da edição.

Alternativa 2: Remoção

- 3- O Usuário solicita a **remoção** de uma Aplicação.
- 4- O Sistema mostra a lista de aplicações existentes.
- 5- O Usuário escolhe uma aplicação e solicita sua remoção.
- 6- O Sistema verifica que não existe SRS e requisitos de aplicação associados à aplicação e remove a mesma.
- 7- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da remoção (usuário registrado) e Data e Hora da remoção.

Alternativa 2.1: Remoção: Aplicação contém SRS e requisitos associados

- 6- O Sistema verifica que existe SRS e requisitos de aplicação associados à aplicação.
- 7- A operação é cancelada.

Alternativa 3: Consulta

- 3- O Usuário solicita a **consulta** de uma Aplicação.
- 4- O Sistema mostra a lista de aplicações existentes.
- 5- O Usuário escolhe uma aplicação.
- 6- O Sistema mostra as informações da Aplicação.

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso “Gerenciar SRS”

Objetivo: Permitir que o usuário possa adicionar, editar, remover ou consultar SRS do sistema.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso:

Passos:

- 1- O usuário acessa a funcionalidade de cadastro de SRS.
- 2- O sistema disponibiliza as funções: **Adiciona, Edita, Remove** ou **Consulta**.
- 3- O usuário solicita a **adição** de um SRS.
- 4- O sistema requisita as seguintes informações para cadastro do SRS: Nome da Aplicação ao qual o SRS está associado, Informações de Suporte, Introdução do SRS e Descrição Geral.
- 5- O usuário informa os dados e solicita salvá-los.
- 6- O sistema salva as informações do novo SRS.
- 7- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da adição (usuário registrado) e Data e Hora da adição.

Alternativas:

Alternativa 1: Edição

- 3- O usuário solicita a **edição** de um SRS.
- 4- O sistema mostra a lista de SRS existentes.
- 5- O usuário escolhe um SRS e altera as Informações de Suporte, Introdução do SRS ou Descrição Geral.
- 6- O sistema salva as informações.
- 7- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da edição (usuário registrado) e Data e Hora da edição.

Alternativa 2: Remoção

- 3- O usuário solicita a **remoção** de um SRS.
- 4- O sistema mostra a lista dos SRSs existentes.
- 5- O usuário escolhe um SRS e solicita sua remoção.
- 6- O Sistema verifica que não existem requisitos de aplicação associados ao SRS e efetua a sua remoção.
- 7- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da remoção (usuário registrado) e Data e Hora da remoção.

Alternativa 2.1: Remoção: SRS contém requisitos associados

- 6- O Sistema verifica que existem requisitos de aplicação associados ao SRS.
- 7- A operação é cancelada.

Alternativa 3: Consulta

- 3- O Usuário solicita a **consulta** de um SRS.
- 4- O Sistema mostra a lista de SRSs existentes.
- 5- O Usuário escolhe um SRS.
- 6- O Sistema mostra as informações do SRS.

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Adicionar Requisito de Aplicação

Objetivo: Permitir que o usuário possa adicionar Requisitos de Aplicação no sistema.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso:

Passos:

- 1- O Usuário solicita a **adição** de um Requisito de Aplicação.
- 2- O Sistema requisita que o usuário informe com qual ou quais as aplicações que o novo requisito deve ser associado.
- 3- O Usuário informa uma aplicação.
- 4- O Sistema mostra os tipos de Requisitos de Aplicação: Não Funcional Geral ou Caso de Uso.
 1. Se o usuário escolher Não Funcional Geral, ver subseção **Adicionar Requisito Não Funcional Geral**.
 2. Se o usuário escolher Caso de Uso, ver subseção **Adicionar Caso de Uso**.
- 5- O sistema salva o requisito.
- 6- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da adição (usuário registrado) e Data e Hora da adição.

Subseção Adicionar Requisito Não Funcional Geral

- 1- O Usuário informa a descrição do requisito não funcional geral.

Subseção Adicionar Caso de Uso

- 1- O usuário informa o Objetivo, Título, Atores, Pré-condição e Pós-condições.
- 2- O usuário informa o passo do cenário principal ou cenário alternativo.
- 3- O sistema repete o passo 2 até que todos os passos sejam informados.

- 4- O usuário informa os requisitos não funcionais específicos associados com o caso de uso.

Alternativas:

Alternativa 1: Casos de Uso que precedem o caso de uso sendo adicionado

- 2- O usuário identifica que existem casos de uso que devem ser executados antes (precedem) do caso de uso sendo adicionado.
- 3- O sistema mostra uma lista dos casos de uso das aplicações associadas com o caso de uso sendo adicionado.
- 4- O usuário pode selecionar um ou mais casos de uso.
- 5- O sistema salva a relação de precedência entre os casos de uso e volta ao passo 2 da **Subseção Adicionar Caso de Uso**.

Alternativa 2: Adicionar Relacionamento Extensão

- 2- O Usuário informa um passo que representa um relacionamento extensão com outro caso de uso.
- 3- O Sistema executa o caso de uso **Adicionar Relacionamento Extensão**.
- 4- O Sistema volta ao passo 2 da **Subseção Adicionar Caso de Uso**.

Alternativa 3: Adicionar Relacionamento Inclusão

- 2- O Usuário informa um passo que representa um relacionamento inclusão com outro caso de uso.
- 3- O Sistema executa o caso de uso **Adicionar Relacionamento Inclusão**.
- 4- O Sistema volta ao passo 2 da **Subseção Adicionar Caso de Uso**.

Alternativa 4: Adicionar Relacionamento Generalização

- 2- O Usuário informa um passo que representa um relacionamento de generalização com outro caso de uso.
- 3- O Sistema executa o caso de uso **Adicionar Relacionamento Generalização**.
- 4- O Sistema volta ao passo 2 da **Subseção Adicionar Caso de Uso**.

Alternativa 5: Casos de Uso que procedem o caso de uso sendo adicionado

- 2- O sistema mostra uma lista dos casos de uso das aplicações associadas com o caso de uso sendo adicionado.
- 3- O usuário pode selecionar um ou mais casos de uso.

- 4- O sistema salva a relação de procedência entre os casos de uso e volta ao passo 2 da **Subseção Adicionar Caso de Uso**.

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Editar Requisito de Aplicação

Objetivo: Permitir que o usuário possa editar Requisitos de Aplicação do sistema.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso:

Passos:

- 1- O usuário seleciona um Requisito de Aplicação para edição.
 1. Se o requisito sendo editado for do tipo Não Funcional Geral, ver subseção **Editar Requisito Não Funcional Geral**.
 2. Se o requisito sendo editado for do tipo Caso de Uso, ver subseção **Editar Caso de Uso**.
- 2- O sistema salva o requisito.
- 3- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da edição (usuário registrado) e Data e Hora da edição.

Subseção Editar Requisito Não Funcional Geral

- 1- O Usuário informa a descrição do requisito não funcional.

Subseção Editar Caso de Uso

- 1- O usuário informa o Objetivo, Título, Atores, Pré-condição e Pós-condições.
- 2- O usuário informa os passos do cenário principal e cenários alternativos.
- 3- O usuário informa os requisitos não funcionais específicos associados com o caso de uso.

Alternativas:

Alternativa 1: Casos de Uso que precedem o caso de uso

- 2- O usuário identifica que existem casos de uso que devem ser executados antes (precedem) do caso de uso sendo editado.
- 3- O sistema mostra uma lista dos casos de uso das aplicações associadas com o caso de uso sendo editado.

- 4- O usuário pode selecionar um ou mais casos de uso.
- 5- O sistema salva a relação de precedência entre os casos de uso e volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 2: Adicionar Relacionamento Extensão

- 2- O Usuário informa um passo que representa um relacionamento *extensão* com outro caso de uso.
- 3- O Sistema executa o caso de uso **Adicionar Relacionamento Extensão**.
- 4- O Sistema volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 3: Remover Relacionamento Extensão

- 2- O Usuário solicita remover um passo que representa um relacionamento *extensão* com outro caso de uso já cadastrado no sistema, ou seja, existe outro caso de uso que estende o caso de uso sendo editado.
- 3- O Sistema remove o relacionamento *extensão* entre os casos de uso.
- 4- O Sistema volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 4: Editar Relacionamento Extensão

- 2- O Usuário solicita alterar um passo que representa um relacionamento *extensão* com outro caso de uso já cadastrado no sistema, ou seja, existe outro caso de uso que estende o caso de uso sendo editado.
- 3- O Usuário altera a descrição do Ponto de Extensão.
- 4- O Sistema salva a alteração do relacionamento *extensão* entre os casos de uso.
- 5- O Sistema volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 5: Adicionar Relacionamento Inclusão

- 2- O Usuário informa um passo que representa um relacionamento *inclusão* com outro caso de uso.
- 3- O Sistema executa o caso de uso **Adicionar Relacionamento Inclusão**.
- 4- O Sistema volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 6: Remove Relacionamento Inclusão

- 2- O Usuário remove um passo que representa um relacionamento *inclusão* com outro caso de uso já cadastrado no sistema, ou seja, existe outro caso de uso que é incluído pelo caso de uso sendo editado.
- 3- O Sistema remove o relacionamento *inclusão* entre os casos de uso.
- 4- O Sistema volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 7: Adicionar Relacionamento Generalização

- 2- O Usuário informa um passo que representa um relacionamento de *generalização* com outro caso de uso.
- 3- O Sistema executa o caso de uso **Adicionar Relacionamento Generalização**.
- 4- O Sistema volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 8: Remover Relacionamento Generalização

- 2- O Usuário remove um passo que representa um relacionamento de *generalização* com outro caso de uso já cadastrado no sistema, ou seja, existe outro caso de uso que é a especialização (caso de uso filho) do caso de uso sendo adicionado.
- 3- O Sistema registra a remoção do relacionamento *generalização* entre os casos de uso.
- 4- O Sistema volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 9: Casos de Uso que procedem o caso de uso sendo adicionado

- 2- O Usuário informa um passo que representa um habilitador de outros casos de uso, ou seja, outros casos de uso são habilitados pelo caso de uso sendo adicionado.
- 3- O sistema mostra uma lista dos casos de uso das aplicações associadas com o caso de uso sendo editado.
- 4- O usuário pode selecionar um ou mais casos de uso.
- 5- O sistema salva a relação de procedência entre os casos de uso e volta ao passo 2 da **Subseção Editar Caso de Uso**.

Alternativa 10: Casos de Uso que procedem ao caso de uso sendo adicionado

- 2- O Usuário remove um passo que representa um habilitador de outros casos de uso, ou seja, outros casos de uso são habilitados pelo caso de uso sendo adicionado.
- 3- O Sistema registra a remoção do relacionamento de procedência entre os casos de uso e volta ao passo 2 da **Subseção Editar Caso de Uso**.

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Remover Requisito de Aplicação

Objetivo: Permitir que o usuário possa remover Requisitos de Aplicação do sistema.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso:

Passos:

- 1- O usuário seleciona um Requisito de Aplicação para remoção.
- 2- O Sistema mostra a lista de Aplicações que o requisito está associado e confirma se o usuário deseja remover o requisito.
- 3- O usuário confirma a remoção.
 1. Se o requisito selecionado for do tipo Não Funcional Geral, executa passo 4.
 2. Se o requisito selecionado for do tipo Caso de Uso, ver subseção **Remover Caso de Uso**.
- 4- O sistema remove o requisito.
- 5- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da remoção (usuário registrado) e Data e Hora da remoção.

Subseção Remover Caso de Uso

- 1- O sistema verifica que o caso de uso sendo removido inclui outros casos de usos que não tem relacionamento com nenhum outro caso de uso, nem relacionamento com atores.
- 2- O sistema remove o caso de uso e os casos de uso incluídos.

Pontos de Extensão:

Requisitos Não Funcionais Associados:

- 1- Na remoção, o sistema não deve remover o requisito de aplicação fisicamente do Sistema. A remoção deve ser apenas lógica.

Caso de Uso Gerenciar Requisições de Mudanças

Objetivo: Permitir que o usuário possa adicionar, editar, remover ou consultar requisições de mudança do sistema.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Requisição de mudança é cadastrada e os requisitos de aplicação afetados pela mesma estão identificados no sistema.

Passos:

- 1- O usuário acessa a funcionalidade de cadastro de Requisições de Mudanças.
- 2- O sistema disponibiliza as funções: Adiciona, Altera ou Remove.
- 3- O usuário solicita a **adição** de uma Requisição de Mudança.
- 4- O sistema requisita as seguintes informações: Origem da Mudança, Descrição, Tipo de Mudança, Data de Entrega Planejada, Prioridade de Implementação, Implementador, Gerador, Prioridade do Gerador, Nome do Projeto, Título da Requisição de Mudança e o Verificador.
- 5- O usuário informa os dados e solicita salvá-los.
- 6- O Sistema salva a requisição de mudança e informações como Data de Submissão, Data e Hora da Atualização, Nome do Autor e Número da Versão da modificação.

Alternativas:

Alterntativa 1: Edição

- 3- O usuário solicita a **edição** de uma Requisição de Mudança.
- 4- O sistema permite que o usuário modifique as seguintes informações: Origem da Mudança, Descrição, Tipo de Mudança, Data de Entrega Planejada, Prioridade de Implementação, Implementador, Prioridade do Gerador, Nome do Projeto, Título da Requisição de Mudança e o Verificador.
- 5- O usuário informa os dados e solicita salvá-los.
- 6- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da edição (usuário registrado) e Data e Hora da atualização.

Alterntativa 2: Remoção

- 3- O usuário solicita a **remoção** de uma Requisição de Mudança.
- 4- O sistema verifica se não há requisitos de aplicação associados à Requisição de Mudança e remove a mesma.
- 5- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da remoção (usuário registrado) e Data e Hora da atualização.

Alterntativa 3: Remoção com Requisitos Relacionados

- 3- O usuário solicita a **remoção** de uma Requisição de Mudança.
- 4- O sistema verifica se há requisitos de aplicação associados à Requisição de Mudança e mostra uma mensagem ao usuário indicando que a remoção da Requisição de Mudança implicará na perda dos relacionamentos entre Requisição de Mudança e requisitos de aplicação.
- 5- O usuário confirma a remoção.
- 6- O Sistema remove a requisição de mudança e os relacionamentos com os requisitos de aplicação.
- 7- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da remoção (usuário registrado) e Data e Hora da atualização.

Alterntativa 4: Consulta

- 3- O usuário solicita a **consulta** de uma Requisição de Mudança.
- 4- O sistema mostra as informações da Requisição de Mudança tais como:
 - a. Origem da Mudança, Descrição, Tipo de Mudança, Data de Entrega Planejada, Prioridade de Implementação, Implementador, Gerador, Prioridade do Gerador, Nome do Projeto, Título da Requisição de Mudança e o Verificador.
 - b. Lista de Requisitos de Aplicação afetados pela mesma.

Pontos de Extensão:

Requisitos Não Funcionais Associados:

- 1- Na remoção, o sistema não deve remover a Requisição de Mudança e os relacionamentos com requisitos de aplicação fisicamente do Sistema. A remoção deve ser apenas lógica.

Caso de Uso Adicionar Casos de Usos afetados por Requisição de Mudança

Objetivo: Permitir que o usuário possa adicionar casos de usos identificados através de uma requisição de mudança.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Os requisitos de aplicação adicionados por uma Requisição de Mudança estão identificados no sistema.

Passos:

- 1- O Usuário seleciona uma Requisição de Mudança.
- 2- O usuário identifica que deve adicionar um caso de uso para atender à requisição de mudança.
- 3- O Sistema executa o caso de uso **Adicionar Caso de Uso**.
- 4- O Sistema relaciona o caso de uso adicionado com a requisição de mudança, indicando que a requisição de mudança adiciona o caso de uso.
- 5- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da modificação (usuário registrado) e Data e Hora da modificação.
- 6- O Sistema repete os passos de 2 a 5 até identificar todos os requisitos de aplicação **adicionados** pela requisição de mudança.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Alterar Casos de Usos afetados por Requisição de Mudança

Objetivo: Permitir que o usuário possa alterar casos de usos identificados através de uma requisição de mudança.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Os requisitos de aplicação alterados por uma Requisição de Mudança estão identificados no sistema.

Passos:

- 1- O Usuário seleciona uma Requisição de Mudança.
- 2- O usuário identifica que deve editar um caso de uso para atender à requisição de mudança.

- 3- O Sistema mostra a lista de casos de uso existentes agrupados por aplicação e com seus relacionamentos inclusão, extensão e generalização devidamente representados.
- 4- Para cada caso de uso, o sistema mostra a lista de casos de uso que precedem e os casos de uso que procedem o mesmo.
- 5- O usuário seleciona um ou mais casos de uso.
- 6- O sistema apresenta o histórico de alterações de cada caso de uso selecionado, como segue:
 - a. Lista de requisições de mudança que foram implementadas no passado e que afetaram o caso de uso;
 - b. Para cada requisição de mudança listada, o sistema deve mostrar todos os casos de uso de aplicações afetados pela requisição de mudança.
- 7- O usuário seleciona um caso de uso afetado pelas requisições de mudanças anteriores.
- 8- O usuário repete o passo 5 a 7 até selecionar todos os casos de uso alterados pela requisição de mudança.
- 9- O Sistema executa o caso de uso **Editar Caso de Uso** para cada caso de uso selecionado.
- 10- O Sistema relaciona os casos de uso alterados com a requisição de mudança, indicando que a requisição de mudança altera os casos de uso identificados.
- 11- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da modificação (usuário registrado) e Data e Hora da modificação.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Remove Casos de Usos afetados por Requisição de Mudança

Objetivo: Permitir que o usuário possa remover casos de usos identificados através de uma requisição de mudança.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Os requisitos de aplicação removidos por uma Requisição de Mudança estão identificados no sistema.

Passos:

- 1- O Usuário seleciona uma Requisição de Mudança.
- 2- O usuário identifica que deve remover um caso de uso para atender à requisição de mudança.
- 3- O Sistema mostra a lista de casos de uso existentes agrupados por aplicação e com seus relacionamentos inclusão, extensão e generalização devidamente representados.
- 4- Para cada caso de uso, o sistema mostra a lista de casos de uso que precedem e os casos de uso que procedem o mesmo.
- 5- O usuário seleciona um caso de uso para ser removido.
- 6- O sistema apresenta o histórico de alterações deste caso de uso como segue:
 - a. Lista de requisições de mudança que foram implementadas no passado e que afetaram o caso de uso;
 - b. Para cada requisição de mudança listada, o sistema deve mostrar todos os casos de uso de aplicações afetados pela requisição de mudança com informação se o caso de uso foi adicionado, alterado ou removido.
- 7- O usuário seleciona um caso de uso afetado pelas requisições de mudanças anteriores que também deve ser removido.
- 8- O usuário repete o passo 5 a 7 até selecionar todos os casos de uso que devem ser removidos pela requisição de mudança.
- 9- O Sistema executa o caso de uso **Remover Caso de Uso** para cada caso de uso selecionado.
- 10- O Sistema relaciona os casos de uso removido com a requisição de mudança, indicando que a requisição de mudança remove os casos de uso.
- 11- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da modificação (usuário registrado) e Data e Hora da modificação.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Adicionar Requisitos Não Funcional Geral afetados por Requisição de Mudança

Objetivo: Permitir que o usuário possa adicionar requisitos não funcional geral afetados por uma requisição de mudança.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Os requisitos de aplicação não funcional geral adicionados por uma Requisição de Mudança estão identificados no sistema.

Passos:

- 1- O Usuário seleciona uma Requisição de Mudança.
- 2- O Usuário identifica que deve adicionar um requisito não funcional geral para atender à requisição de mudança.
- 3- O Sistema executa o caso de uso **Adicionar Requisito Não Funcional Geral**.
- 4- O Sistema relaciona o requisito não funcional geral adicionado com a requisição de mudança, indicando que a requisição de mudança adiciona o requisito.
- 5- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da modificação (usuário registrado) e Data e Hora da modificação.
- 6- O Sistema repete os passos de 2 a 5 até identificar todos os requisitos não funcional geral **adicionados** pela requisição de mudança.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Editar Requisitos Não Funcional Geral afetados por Requisição de Mudança

Objetivo: Permitir que o usuário possa alterar requisitos não funcional geral afetados por uma requisição de mudança.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Os requisitos de aplicação não funcional geral alterados por uma Requisição de Mudança estão identificados no sistema.

Passos:

- 1- O Usuário identifica que deve alterar um requisito não funcional geral para atender à requisição de mudança.
- 2- O Sistema mostra a lista de requisitos não funcional geral existentes agrupados por aplicação.
- 3- O Usuário seleciona um requisito para edição.
- 4- O sistema apresenta o histórico de alterações deste requisito não funcional geral como segue:
 - a. Lista de requisições de mudança que foram implementadas no passado e que afetaram o requisito não funcional geral;
 - b. Para cada requisição de mudança listada, o sistema deve mostrar todos os requisitos não funcional geral de aplicações afetados pela requisição de mudança.
- 5- O usuário seleciona um requisito não funcional geral afetado pelas requisições de mudanças anteriores.
- 6- O usuário repete o passo 4 a 6 até selecionar todos os requisitos alterados pela requisição de mudança.
- 7- O Sistema executa o caso de uso **Editar Requisito Não Funcional Geral**.
- 8- O Sistema relaciona o requisito não funcional geral adicionado com a requisição de mudança, indicando que a requisição de mudança adiciona o requisito.
- 9- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da modificação (usuário registrado) e Data e Hora da modificação.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Remover Requisitos Não Funcional Geral afetados por Requisição de Mudança

Objetivo: Permitir que o usuário possa remover requisitos não funcional geral afetados por uma requisição de mudança.

Atores: Engenheiro de Requisitos

Pré-condição: Usuário está registrado no Sistema.

Pós-condição de Sucesso: Os requisitos de aplicação não funcional geral removidos por uma Requisição de Mudança estão identificados no sistema.

Passos:

- 1- O Usuário identifica que deve remover um requisito não funcional geral para atender à requisição de mudança.
- 2- O Sistema mostra a lista de requisitos não funcional geral existentes agrupados por aplicação.
- 3- O Usuário seleciona um requisito para remoção.
- 4- O sistema apresenta o histórico de alterações deste requisito não funcional geral como segue:
 - a. Lista de requisições de mudança que foram implementadas no passado e que afetaram o requisito não funcional geral;
 - b. Para cada requisição de mudança listada, o sistema deve mostrar todos os requisitos não funcional geral de aplicações afetados pela requisição de mudança.
- 5- O usuário seleciona um requisito não funcional geral afetado pelas requisições de mudanças anteriores.
- 6- O usuário repete o passo 4 a 6 até selecionar todos os requisitos removidos pela requisição de mudança.
- 7- O Sistema executa o caso de uso **Remover Requisito Não Funcional Geral**.
- 8- O Sistema relaciona os requisitos não funcional geral removidos com a requisição de mudança, indicando que a requisição de mudança remove os requisitos.
- 9- O Sistema registra dados do histórico da mudança efetuada: Número da Versão, Autor da modificação (usuário registrado) e Data e Hora da modificação.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Adicionar Relacionamento Extensão

Objetivo: Permitir que o usuário possa relacionar um caso de uso com outro caso de uso que estende o mesmo.

Atores: Engenheiro de Requisitos

Pré-condição:

Pós-condição de Sucesso: O caso de uso é relacionado com o caso de uso que o estende.

Passos:

- 1- O Usuário adiciona um passo ao caso de uso que representa um relacionamento *extensão* com outro caso de uso já cadastrado no sistema, ou seja, existe outro caso de uso que estende o caso de uso.
- 2- O Sistema mostra a lista de casos de uso da aplicação, ou aplicações, que o caso de uso sendo adicionado/editado é associado.
- 3- O Usuário seleciona um caso de uso da lista.
- 4- O Usuário informa a descrição do Ponto de Extensão.
- 5- O Sistema registra o relacionamento *extensão* entre os casos de uso.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Adicionar Relacionamento Inclusão

Objetivo: Permitir que o usuário possa relacionar um caso de uso com outro caso de uso que é incluído pelo mesmo.

Atores: Engenheiro de Requisitos

Pré-condição:

Pós-condição de Sucesso: O caso de uso é relacionado com o caso de uso incluído.

Passos:

- 1- O Usuário informa um passo que representa um relacionamento *inclusão* com outro caso de uso já cadastrado no sistema, ou seja, existe outro caso de uso que é incluído pelo caso de uso.
- 2- O Sistema mostra a lista de casos de uso da aplicação, ou aplicações, que o caso de uso sendo adicionado/editado é associado.
- 3- O Usuário seleciona um caso de uso da lista.
- 4- O Sistema registra o relacionamento *inclusão* entre os casos de uso.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Adicionar Relacionamento Generalização

Objetivo: Permitir que o usuário possa relacionar um caso de uso com outro caso de uso que é sua especialização.

Atores: Engenheiro de Requisitos

Pré-condição:

Pós-condição de Sucesso: O caso de uso é relacionado com o caso de uso especializado.

Passos:

- 1- O Usuário informa um passo que representa um relacionamento de *generalização* com outro caso de uso já cadastrado no sistema, ou seja, existe outro caso de uso que é a especialização (caso de uso filho) do caso de uso.
- 2- O Sistema mostra a lista de casos de uso da aplicação, ou aplicações, que o caso de uso sendo adicionado/editado é associado.
- 3- O Usuário seleciona um caso de uso da lista.
- 4- O Sistema registra o relacionamento *generalização* entre os casos de uso.

Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

Caso de Uso Consultar Histórico de Mudanças da Aplicação

Objetivo: Permitir que o usuário possa consultar o histórico de mudanças de uma aplicação.

Atores: Engenheiro de Requisitos

Pré-condição:

Pós-condição de Sucesso:

Passos:

- 1- O Usuário solicita consultar o histórico de mudanças de uma aplicação.
- 2- O Sistema mostra a lista de aplicações existentes.
- 3- O Usuário seleciona uma aplicação.
- 4- O Sistema mostra a lista de requisições de mudanças que afetaram aplicação, os requisitos que foram afetados por cada mudança, de que forma foram afetados (adicionados, alterados ou removidos), a data e hora da modificação, e o autor.

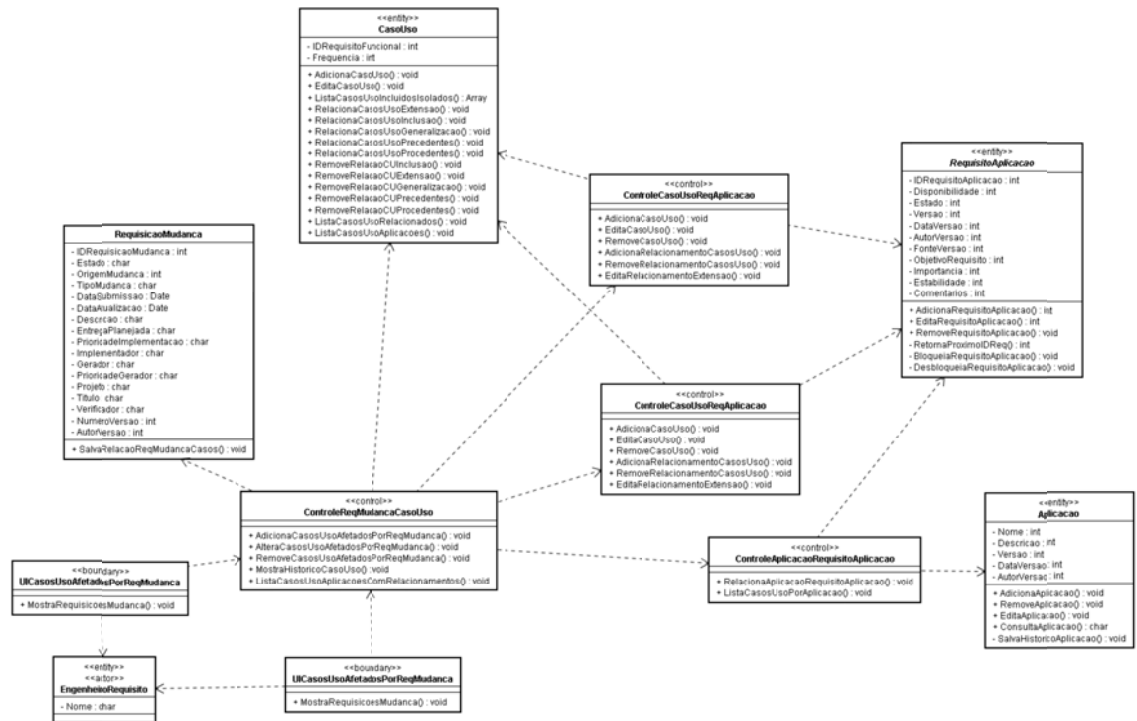
Alternativas:

Pontos de Extensão:

Requisitos Não Funcionais Associados:

3. Diagrama de Classes

O diagrama de classes abaixo mostra um sub-conjunto das classes do protótipo que representam a funcionalidade que altera casos de uso afetados pelas requisições de mudanças.



APÊNDICE II – ESPECIFICAÇÃO DOS CASOS DE USO DOS SISTEMAS DE EXEMPLO

1. Sistema de Vendas

Caso de Uso Entrar no Sistema

Objetivo: Autenticar o usuário no sistema.

Atores: Cliente

Pré-condição:

Pós-condição: Usuário está autenticado no sistema.

Fluxo de Eventos (caminho básico):

1. O caso de uso começa quando o cliente entra no sistema.
2. O sistema requisita que o usuário informe o login e a senha.
3. O usuário informa os dados.
4. O sistema autentica o usuário e redireciona para a página principal do sistema.

Alternativas:

Alternativa 1:

4. O sistema informa ao usuário que o login ou a senha são inválidos.

Caso de Uso Submeter Pedido

Objetivo: Permitir que o usuário submeta um pedido de compra.

Atores: Cliente

Pré-condição: Usuário está autenticado no sistema.

Pós-condição: O pedido deve ter sido gravado no sistema e marcado como confirmado.

Fluxo de Eventos (caminho básico):

1. O caso de uso começa quando o cliente seleciona "submeter pedido".
2. O cliente fornece seu nome e endereço.
3. Se o cliente fornece apenas o CEP, o sistema coloca automaticamente a cidade e o estado.
4. O cliente fornece os códigos do produto.
5. O sistema devolve as descrições e o preço de cada produto.
6. O sistema calculará os valores totais para cada produto fornecido.
7. O sistema verifica que o pagamento foi efetuado, marca o pedido como "pendente".
8. Inclui caso de uso "Efetuar Pagamento com cartão de crédito".
9. Quando o pagamento é confirmado, o pedido é marcado como "confirmado" e o número de pedido (NP) é dado ao cliente.

Alternativas:

Alternativa 1:

4. Enquanto o cliente quiser pedir itens faça:
 1. O cliente fornece código do produto.
 2. O sistema fornece a descrição e preço do produto.
 3. O sistema atualiza o valor total.
5. O sistema verifica que o pagamento foi efetuado, marca o pedido como "pendente". Inclui caso de uso "Efetuar Pagamento com cartão de crédito".

6. Quando o pagamento é confirmado, o pedido é marcado como "confirmado" e o número de pedido (NP) é dado ao cliente.

Requisitos Não Funcionais:

A qualquer momento antes de submeter, o cliente pode selecionar cancelar. O pedido não é gravado e o caso de uso termina.

No passo 7, se alguma informação estiver incorreta, o sistema pede ao cliente para corrigir a informação.

Caso de Uso Cliente Especial

Objetivo: Calcular valor total das compras do cliente especial baseado no desconto.

Estende

1. Submeter Pedido, antes do passo 6

Fluxo de Eventos Primário (caminho básico):

1. O sistema procura o valor do desconto para o cliente.
2. O sistema mostra o desconto ao cliente.
3. O sistema atualiza o total, subtraindo o valor do desconto.

Pré-condição: Cliente ser Cliente Especial

Pós-condição: Valor do desconto total calculado e subtraído do total de compras

Caso de Uso Produto em Oferta

Objetivo: Fornecer valor de produto em oferta.

Estende

1. Submeter Pedido, antes do passo 5

Fluxo de Eventos Primário (caminho básico):

1. O sistema procura o valor do desconto para o produto
2. O sistema mostra o desconto do produto ao usuário
3. O sistema calcula o valor do desconto
4. O sistema atualiza o total, subtraindo o valor do desconto

Pré-condição: Produto estar em oferta

Pós-condição: Valor do desconto calculado.

Caso de Uso Verificar Pedido

Objetivo: Verificar os dados da situação do pedido.

Atores: Cliente, Funcionário

Fluxo de Eventos Primário (caminho básico):

1. O caso de uso começa quando o cliente seleciona "Meu pedido".
2. Usa Procurar Pedido
3. O Sistema mostra os dados da situação do pedido e o caso de uso termina.

Fluxo de Secundário (caminho alternativo):

3. O Sistema informa que o pedido não está cadastrado e solicita que o usuário verifique se os dados do pedido estão corretos.
4. O usuário corrige os dados.
5. Usa Procurar Pedido
6. O Sistema mostra os dados da situação do pedido e o caso de uso termina.

Pré-condição: O usuário ter feito o pedido

Pós-condição: A situação do pedido não ter sido alterada.

Caso de Uso Cancelar Pedido

Objetivo: Cancelar o pedido.

Atores: Cliente, Funcionário

Fluxo de Eventos Primário (caminho básico):

1. O caso de uso começa quando o cliente seleciona "Cancelar Pedido".
2. Usa Procurar Pedido
3. O Sistema mostra os dados da situação do pedido.
4. O Usuário requisita cancelar o pedido.
5. O Sistema cancela o pedido.

Pré-condição: O usuário ter feito o pedido

Pós-condição: A situação do pedido é marcada como cancelada.

Requisitos não funcionais:

O registro do pedido permanece para o histórico de pedidos.

Caso de Uso Procurar Pedido

Objetivo: Procurar por pedido.

Atores

Fluxo de Eventos Primário (caminho básico):

1. O cliente pode fornecer o número do pedido (NP), a identificação ou o nome do cliente
2. O cliente ativa "Busca"
3. Se o cliente tiver fornecido o NP
 1. O sistema devolve o pedido e o caso de uso termina
4. Se o cliente tiver fornecido a identificação ou o nome do cliente
 1. O sistema mostra a lista com todos os pedidos do cliente
 2. O cliente seleciona o produto
 3. O sistema devolve o pedido e o caso de uso termina

Fluxo de Secundário (caminho alternativo):

2. O sistema não encontra o pedido.
3. O sistema solicita que o usuário verifique se os dados do pedido estão corretos.
4. O caso de uso é encerrado.

Pré-condição: O usuário ter feito o pedido

Pós-condição: A situação do pedido não ter sido alterada.

Caso de Uso Fornecer produto

Objetivo: Entregar os produtos comprados.

Atores: Fornecedor

Pré-condição: A compra de produtos do fornecedor já foi efetuada.

Fluxo de Eventos (caminho básico):

1. O caso de uso começa quando o fornecedor pega a lista de produtos comprados.
2. O Fornecedor entrega os produtos de acordo com as especificações.
3. O Sistema atualiza a quantidade disponível de produtos.
4. Quando o sistema realizar a atualização deve ser emitido uma confirmação de recebimento para o fornecedor.

Pós-condição: Lista de produtos deve estar atualizada.

2. Sistema de Compra de Fornecedores

Caso de Uso Submeter Requisição de Compra aos Fornecedores

Objetivo: Funcionário submete requisição de compra aos fornecedores para atender ao pedido do cliente.

Atores: Funcionário

Pré-condição: Pedido do cliente foi gerado.

Pós-condição: Requisição de compra foi submetida aos fornecedores.

Fluxo de Eventos (caminho básico):

1. O caso de uso começa quando o funcionário seleciona um pedido do cliente. Inclui Procurar Pedido
2. O sistema mostra a lista de fornecedores.
3. O usuário seleciona os fornecedores para os quais deseja submeter a requisição de compra.
4. O sistema submete a requisição para os fornecedores proverem suas cotações.

Caso de Uso Aceitar Proposta do Fornecedor e Efetuar Compra

Objetivo: Funcionário aceita proposta do fornecedor e efetua a compra.

Atores: Funcionário

Pré-condição: Proposta do fornecedor foi submetida. Relação com Pós-condição do caso de uso “Submeter Proposta para Requisição de Compra”

Pós-condição: Compra é efetuada.

Fluxo de Eventos (caminho básico):

1. O caso de uso começa quando o funcionário recebe e aceita uma proposta do fornecedor.
2. O funcionário solicita ao sistema para efetuar a compra. Inclui caso de uso “Efetuar Pagamento com cartão de crédito”.
3. Quando o pagamento é confirmado o sistema confirma a compra.

Caso de Uso Submeter Proposta Requisição de Compra

Objetivo: Fornecedor submete proposta para atender requisição de compra.

Atores: Fornecedor

Pré-condições: Requisição de compra foi submetida pelo funcionário.

Pós-condições: Proposta foi submetida ao funcionário.

Fluxo de Eventos (caminho básico):

1. O caso de uso começa quando o fornecedor recebe uma requisição de compra do funcionário da empresa.
2. O fornecedor informa uma proposta para a requisição de compra.
3. O sistema submete a proposta ao funcionário.

Alternativas:

Alternativa 1:

2. O fornecedor não tem proposta para atender à requisição de compra.
3. O caso de uso termina e nenhuma proposta é submetida.

3. Sistema de Pagamento

Caso de Uso Efetuar Pagamento

Objetivo: Efetuar o pagamento na forma de cartão de crédito ou cheque.

Atores: Funcionário; Cliente

Pré-condições: Compra em andamento.

Pós-condições: Pagamento é efetuado.

Fluxo de Eventos (caminho básico):

1. Sistema apresenta as formas de pagamento aceitas.
2. Sistema solicita uma forma de pagamento.
3. Usuário seleciona uma forma de pagamento: “Pagamento com cartão de crédito” ou “Pagamento com cheque” ou “Pagamento em dinheiro”.

Alternativas:

Alternativa 1: Pagamento com cartão de crédito

1. Usuário seleciona "Pagamento com Cartão de crédito" no passo 3.
2. Sistema solicita os dados do cartão de crédito.
3. Usuário entra com os dados do cartão de crédito.
4. Sistema apresenta mensagem que informa que o Banco autorizou o pagamento.
5. Usuário confirma o pagamento.
6. Sistema imprime comprovante de pagamento.
7. Caso de uso termina.

Alternativa 2: Pagamento com cheque

1. Usuário seleciona “Pagamento com Cheque” passo 3
2. Sistema solicita os dados do cheque.
3. Usuário entra com os dados do cheque.
4. Incluir “Verificar Cadastro do Cliente no SPC”
5. Sistema apresenta mensagem que cliente não possui restrições.
6. Usuário confirma o pagamento.
7. Sistema imprime comprovante de pagamento.
8. Caso de uso termina.

Alternativa 3: Pagamento em dinheiro

1. Usuário seleciona “Pagamento em Dinheiro” passo 3
2. Funcionário informa a quantia.
3. Sistema imprime comprovante de pagamento.
4. Caso de uso termina.

Verificar Cadastro do Cliente no SPC

Objetivo: Verificar se o cliente está no SPC.

Atores:

Pré-condições:

Pós-condições:

Fluxo de Eventos (caminho básico):

1. Sistema verifica se cliente possui cadastro no SPC.
2. Sistema informa que cliente não possui cadastro.

Alternativas:

Alternativa 1:

2. Sistema informa que cliente possui cadastro no SPC e mostra o valor da dívida.

Criar Crediário para Cliente

Objetivo: Criar crediário para cliente

Atores: Funcionário

Pré-condições:

Pós-condições: Crediário criado.

Fluxo de Eventos (caminho básico):

1. Usuário solicita criação de crediário para o cliente.
2. Incluir “Verificar Cadastro do Cliente no SPC”.

APÊNDICE III – TELAS DO PROTÓTIPO

1. Tela para adição do Caso de Uso “Submeter Pedido”:

Adicionar Caso de Uso

Aplicação: Sistema de Vendas

Objetivo: Permitir que o usuário submeta um pedido de compra.

Título: Submeter Pedido

Atores: Cliente

Pré-condições: Identificar Pré-Condições

Objeto	Estado
Usuário	deve estar autenticado do sistema.

Pós-Condições: Identificar Pós-Condições

Objeto	Estado
Pedido	deve ter sido gravado no sistema e marcado como confirmado.

Importancia: Alta

Comentarios:

Frequencia:

Cenário Principal: Adicionar Cenário Principal

Cenários Alternativos: Adicionar Cenários Alternativos

Requisitos Não Funcionais: Adicionar Requisitos Não Funcionais

Salvar Caso de Uso

2. Telas para definição do cenário principal do Caso de Uso “Submeter Pedido”:

Cenário Principal

Tipo de Passo: Extensão

Número do Passo: 5

Descrição do Passo: O sistema devolve as descrições e o preço de cada produto.

Lista de Casos de Uso. Seleccione um Caso de Uso que Estende o Caso de Uso corrente.

Aplicação: Sistema de Vendas

Ponto de Extensão: Produto em Oferta

Aplicação: "Sistema de Vendas" Título: "Produto em Oferta"

Aplicação: "Sistema de Vendas" Título: "Cliente Especial"

Salvar Passo

Salvar Cenário Principal

Passo do Cenário Principal

	Ordem	Descrição	Incluído:	Estendem	Generaliza
▶	1	O caso de uso começa quando o cliente seleciona "submeter pedido".			
	2	O cliente fornece seu nome e endereço.			
	3	Se o cliente fornece apenas o CEP, o sistema coloca automaticamente ...			
	4	O cliente fornece os códigos do produto.			
*					

Cenário Principal

Tipo de Passo
Extensão

Número do Passo 6

Descrição do Passo O sistema calculará os valores totais para cada produto fornecido.

Lista de Casos de Uso. Selecione um Caso de Uso que Estende o Caso de Uso corrente.

Aplicação: Sistema de Vendas

Aplicação: "Sistema de Vendas" Título: "Produto em Oferta"
Aplicação: "Sistema de Vendas" Título: "Cliente Especial"

Ponto de Extensão: Cliente Especial

Salvar Passo Salvar Cenário Principal

Passo do Cenário Principal

		Ordem	Descrição	Incluído:	Estendem	Gener.
▶	Remover Editar	1	O caso de uso começa quando o cliente seleciona "submeter pedido".			
	Remover Editar	2	O cliente fornece seu nome e endereço.			
	Remover Editar	3	Se o cliente fornece apenas o CEP, o sistema coloca automaticamente ...			
	Remover Editar	4	O cliente fornece os códigos do produto.			
	Remover Editar	5	O sistema devolve as descrições e o preço de cada produto.		Produto em Oferta	

Cenário Principal

Tipo de Passo
Normal

Número do Passo

Descrição do Passo

Salvar Passo Salvar Cenário Principal

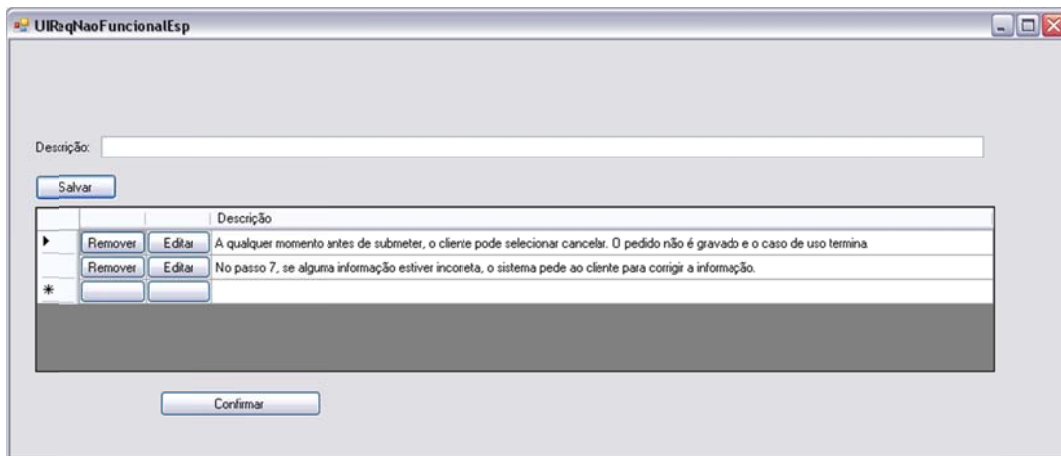
Passo do Cenário Principal

		Ordem	Descrição	Incluído:	Estendem	Gener.
▶	Remover Editar	1	O caso de uso começa quando o cliente seleciona "submeter pedido".			
	Remover Editar	2	O cliente fornece seu nome e endereço.			
	Remover Editar	3	Se o cliente fornece apenas o CEP, o sistema coloca automaticamente ...			
	Remover Editar	4	O cliente fornece os códigos do produto.			
	Remover Editar	5	O sistema devolve as descrições e o preço de cada produto.		Produto em Oferta	

3. Tela para definição do cenário alternativo do Caso de Uso "Submeter Pedido":



4. Tela para definição dos requisitos não-funcionais do Caso de Uso “Submeter Pedido”:



5. Tela que mostra os casos de uso que não estão relacionados com outros casos de uso por suas pré e pós-condições.

