UNIVERSIDADE CÂNDIDO MENDES - UCAM
MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

SUSANA BRUNORO COSTA DE OLIVEIRA

# CONTROLE AUTOMATIZADO DE VERSÕES DE ARTEFATOS DE REQUISITOS DE SOFTWARE

CAMPOS DOS GOYTACAZES
2006

UNIVERSIDADE CÂNDIDO MENDES - UCAM
MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Susana Brunoro Costa de Oliveira

# CONTROLE AUTOMATIZADO DE VERSÕES DE ARTEFATOS DE REQUISITOS DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Pesquisa Operacional e Inteligência Computacional da Universidade Cândido Mendes – Campos/RJ, para a obtenção do grau de MESTRE EM INTELIGÊNCIA COMPUTACIONAL E PESQUISA OPERACIONAL.

Orientadores: Prof. Astério Kiyoshi Tanaka, PhD
Prof. Dalessandro Soares Vianna, DSc

CAMPOS DOS GOYTACAZES, RJ
2006

SUSANA BRUNORO COSTA DE OLIVEIRA

CONTROLE AUTOMATIZADO DE VERSÕES DE ARTEFATOS DE REQUISITOS DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Pesquisa Operacional e Inteligência Computacional da Universidade Cândido Mendes – Campos/RJ, para a obtenção do grau de MESTRE EM INFORMÁTICA APLICADA.

Aprovada em 02 de junho de 2006

BANCA EXAMINADORA

_____

Prof. Astério Kiyoshi Tanaka, PhD – Orientador
Universidade Federal do Estado do Rio de Janeiro

_____

Prof. Dalessandro Soares Vianna, DSc - Orientador
Universidade Cândido Mendes

_____

Prof. Rogério Atem de Carvalho, DSc
Centro Federal de Educação Tecnológica de Campos

_____

Prof. Ulf Bergmann, DSc
Instituto Militar de Engenharia do Rio de Janeiro

CAMPOS DOS GOYTACAZES, RJ
2006

A Deus, meu refúgio e minha fortaleza;

Ao meu esposo Janio, meu amor e sustentáculo;

Aos meus filhos Lucas e Ana Luiza, meus amores em plenitude;

À minha mãe, meu exemplo de vida;

Ao meu pai e meu irmão Carlos Ney que me fazem tanta falta.

AGRADECIMENTOS

A Deus, acima de tudo, por todas as bênçãos derramadas.

Ao meu esposo Janio, por sua cumplicidade e compreensão em cada momento de angustia e desânimo e por acreditar em meu potencial, muito mais do que eu mesma.

Aos meus filhos, pelo carinho nos momentos de cansaço, pela compreensão em minha ausência e por tudo que aprendo com eles.

Ao professor Tanaka, grande orientador e amigo, por seus ensinamentos que foram tão ricos para mim e por seu exemplo como profissional, que certamente, guardarei para sempre.

Ao professor Dalessandro, que, mesmo tendo chegado depois, se mostrou presente e dedicado a esse trabalho.

A todos os professores do programa de mestrado, por seus ensinamentos.

Aos amigos de turma de mestrado, pelo companheirismo e troca de conhecimento.

Ao meu irmão José Augusto e minha cunhada Luciana, pela hospitalidade e carinho.

A todos os meus familiares pelo apoio e incentivo.

Ao Centro Universitário São Camilo Espírito Santo pelo apoio.

De tudo, ficaram três coisas:

A certeza de que estamos sempre começando...

A certeza de que é preciso continuar...

A certeza de que seremos interrompidos antes de terminar...

Portanto, devemos:

Fazer da interrupção um caminho novo...

Da queda um passo de dança...

Do medo, uma escada...

Do sonho, uma ponte...

Da procura, um encontro...

(Fernando Pessoa)

# RESUMO

## CONTROLE AUTOMATIZADO DE VERSÕES DE ARTEFATOS DE REQUISITOS DE SOFTWARE

A complexidade dos *softwares* atuais tem aumentado consideravelmente, elevando, assim, a quantidade e a variedade dos artefatos gerados durante todo processo de desenvolvimento. Paralelamente, vê-se um aumento na mesma proporção das ferramentas e meios de elaboração dos artefatos, o que potencializa as características inerentes em cada equipe de desenvolvimento. Equipes pequenas, altamente organizadas e integradas tendem a ter sua produtividade aumentada com a utilização da Internet, de ferramentas colaborativas e as diversas ferramentas CASE disponíveis. No entanto, controlar todos os artefatos gerados por uma equipe de médio ou grande porte, com baixa cultura de organização ou pouco integrada tende a ser um grande desafio e uma importante causa de insucesso nos processos de desenvolvimento de *software*. A necessidade de se controlar os artefatos do projeto começa a partir do momento que se inicia o projeto, ou seja, no levantamento preliminar dos requisitos. A partir desse momento, qualquer alteração de curso não documentada devidamente pode refletir em insucesso no final do projeto. Ter uma ferramenta que permita o controle dos artefatos gerados durante o processo de desenvolvimento é, portanto, de vital importância para as equipes de desenvolvimento. O presente trabalho analisa, com base em um estudo de caso, a importância do controle de mudanças ocorridas nos artefatos do projeto, desde a etapa de levantamento preliminar dos requisitos. Para aumentar a produtividade e a eficácia do projeto, o estudo é feito baseado na utilização de ferramentas disponíveis no mercado atualmente.



PALAVRAS-CHAVE: Ferramentas CASE, Engenharia de Software, Gerência de Requisitos, Gerência de Configuração e Mudança

ABSTRACT

AUTOMATIC CONTROL OF SOFTWARE REQUIREMENTS ARTIFACTS'S VERSION

The complexity of actual *softwares* has been increased considerabily, increasing, thus, the volume and the variety of the artifacts produced during the development process. Concomitantly, the same increase can be observed in tools and artifact elaboration methods, amplifying the inherent characteristics of each development team. A small, organized and integrated team tends to increase the produtivict when using internet, groupwares tools and available CASE tools. However, the control of all the artifacts generated by a medium or big development team, whitch doesn't have a great organization culture or is not very integrated tends to be a great challenge and an important reason of software development process failure. The necessity of project artifacts control starts from the moment that the project begins, that is, in the preliminary requirement's eliciting. From this moment on, any course change not duly registered can reflect in failure at the end of the project. To have a tool that allows the control of the artifacts generated during the development process is, thus, very important for the development teams. This work analyses, based on a case study, the importance of management changing requirements from the preliminary requirement's eliciting. To improve the productivity and the efficacy of the project, the analysis is based on available tools in the current market.

KEY WORDS: CASE Tools, Software Engineering, Requirement Management, Configuration and Change Management.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

# 1 INTRODUÇÃO

## 1.1 MOTIVAÇÃO

Segundo KRUCHTEN (2000), agora que todos os sistemas simples já foram desenvolvidos, gerenciar a complexidade de grandes sistemas tem se tornado o principal interesse das empresas desenvolvedoras de *software*. O dinamismo com que essas empresas têm que operar aliado à alta competitividade no setor tem feito com que os desenvolvedores de *software* busquem maior eficácia no processo de produção de seus produtos.

Com o intuito de aperfeiçoar o desenvolvimento de *software* e obter produtos com os níveis desejáveis de qualidade, dentro do cronograma e orçamento propostos, a última década assistiu a uma mudança de enfoque com relação à garantia da qualidade. Tem-se, então, uma nova abordagem na qual o foco principal das atenções não está mais nos produtos criados durante o desenvolvimento, mas, sim, no próprio processo produtivo, visto que este tem se mostrado um dos fatores determinantes para o alcance da qualidade do produto final (GOMES JUNIOR, 2000:9).

Apesar do termo **Engenharia de *Software*** ser largamente empregado e dos processos de desenvolvimento de *software* estarem alcançando uma maturidade cada vez maior, a produção de *software* ainda está muito distantes do ideal de um processo de engenharia, onde todos os envolvidos no processo, desde o engenheiro até o pedreiro são capazes de interpretar uma planta arquitetônica e executar suas tarefas com base neste artefato. Os desenvolvedores vivem o desafio de entender corretamente as necessidades de seus usuários, de produzir artefatos que, de fato, serão entendidos por todos os membros da equipe e, principalmente, de terem controle sobre a evolução dos artefatos produzidos.

Mas, o que é um artefato? Um artefato, segundo RUP (2004), é um pedaço de

informação que: 1) é produzido, modificado ou usado por um processo, 2) define uma área de responsabilidade, e 3) objetiva o controle de versões. Um artefato pode ser um modelo, um elemento de um modelo ou um documento, que por sua vez pode incluir outros documentos. Um pedaço físico de informação que é usado ou produzido por um processo de desenvolvimento de *software*. Exemplos de artefatos incluem modelos, arquivos fontes e arquivos binários executáveis.

## 1.2 OBJETIVO DA DISSERTAÇÃO

O objetivo desta dissertação é apresentar o resultado da avaliação de ferramentas, existentes atualmente no mercado, que permitem o desenvolvimento de artefatos de requisitos de software e o controle das mudanças sofridas no projeto, desde a etapa de levantamento preliminar dos requisitos. Para tanto, foi realizado um estudo sobre engenharia de *software*, com ênfase no processo de desenvolvimento de sistemas. Um enfoque maior foi dado ao Processo Unificado, em especial às atividades de gerência de requisitos e de configuração e mudança.

A abordagem proposta contempla desde a definição dos critérios para análise e a seleção das ferramentas para avaliação até a apresentação dos resultados obtidos. A conclusão final deste trabalho será uma importante ferramenta de decisão para gerentes de projeto de engenharia de *software*.

## 1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho é composto por sete capítulos divididos da seguinte forma:

Este primeiro capítulo apresenta a motivação, o objetivo e a organização do trabalho.

No segundo capítulo, é apresentado o resumo dos estudos realizados sobre processos de desenvolvimento de sistemas com ênfase na etapa de levantamento de requisitos. Também está presente um resumo dos estudos sobre padrões internacionais de qualidade de *software*.

O terceiro capítulo disserta sobre a atividade de Gerência de Requisitos, as ferramentas de controle de artefatos de requisitos analisadas, bem como os critérios utilizados para a avaliação.

No quarto capítulo, são apresentados os objetivos da atividade de Gerência de

Configuração dos processos de desenvolvimento de sistema, algumas ferramentas de controle de configuração de artefatos e os critérios utilizados para avaliá-las.

O quinto capítulo apresenta uma descrição sucinta do estudo de caso utilizado para a avaliação das diversas ferramentas estudadas.

No sexto capítulo, é apresentado o resultado da análise comparativa realizada.

O sétimo e último capítulo apresenta as conclusões do trabalho realizado.

# 2 FUNDAMENTAÇÃO TEÓRICA

No presente capítulo é apresentado o resumo dos estudos realizados sobre processos de desenvolvimento de sistemas. Inicialmente são apresentadas definições de processo de desenvolvimento de software e adiante são apresentados três modelos de processos bastante difundidos no meio corporativo: Modelo Cascata, Espiral e finalmente, o Processo Unificado.

## 2.1 PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE*

LEITE (2001) relata que, antes de mais nada, é preciso entender que ao falar-se de software e de sua engenharia fala-se ao mesmo tempo de produtos e de processos. De nada adianta centrar a atenção só no produto ou só no processo. É necessário que as duas visões caminhem juntas. Portanto, para lidar com qualidade é necessário ter claro que o processo de produção deve ter qualidade e que o produto deve ter qualidade.

Segundo PRESSMAN (1995:31), “Uma primeira definição de engenharia de *sofware* foi proposta por Fritz Bauer na primeira grande conferência dedicada ao assunto:

> ‘O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione efetivamente em máquinas reais’”

Várias outras definições surgiram, dentre elas, a de que um “processo é um conjunto de passos parcialmente ordenados, constituídos por atividades, métodos, práticas e transformações, usado para atingir uma meta. Esta meta geralmente está associada a um ou mais resultados concretos finais, que são os produtos de execução do processo” (PAULA FILHO, 2001:17).

MARTINS (1995) define engenharia de *software* como desenvolvimento e aplicação de ciência, matemática, técnicas, métodos, metodologias e ferramentas para o desenvolvimento e a manutenção econômica de *software* de qualidade, preditível e controlável, operando de modo econômico em máquinas e ambientes reais.

De modo geral, todas as definições apontam para a questão de viabilidade econômica, atendimento dos prazos previstos, confiabilidade e atendimento aos requisitos.

Com o advento da computação pessoal e da Internet, a área de tecnologia da informação teve um aumento exponencial, não somente pela quantidade, mas, principalmente, pela complexidade inserida nos *softwares*. Para atender à necessidade crescente dos profissionais de informática, vários processos têm sido propostos para o desenvolvimento de sistemas. Tais processos buscam normatizar a utilização de todo o conjunto de métodos e padrões necessários desde o primeiro contato dos usuários com o gerente do projeto até a fase final de implantação.

A seguir serão apresentados alguns dos padrões de processo de desenvolvimento de sistemas mais difundidos e utilizados atualmente.

## 2.2 MODELO CASCATA

Também conhecido com modelo *waterfall* ou ciclo clássico (PRESSMAN, 1995:32), é a mais antiga proposta de processo de desenvolvimento de sistemas em uso até hoje em muitas empresas. Como mostrado na figura 2-1, preconiza a execução das atividades de forma linear e seqüencial, ou seja, uma atividade terá início assim que sua antecessora estiver concluída. Embora seja de fácil entendimento e implantação, não contempla a realidade dos projetos, uma vez que somente há a interação dos usuários no levantamento de requisitos e após a conclusão do projeto, na implantação.

Outro problema é que, considerada a dificuldade dos usuários em geral de expressar suas reais necessidades e da equipe de desenvolvimento de compreender o que está sendo solicitado, esse paradigma sustenta um nível de risco elevado durante muito tempo no projeto.

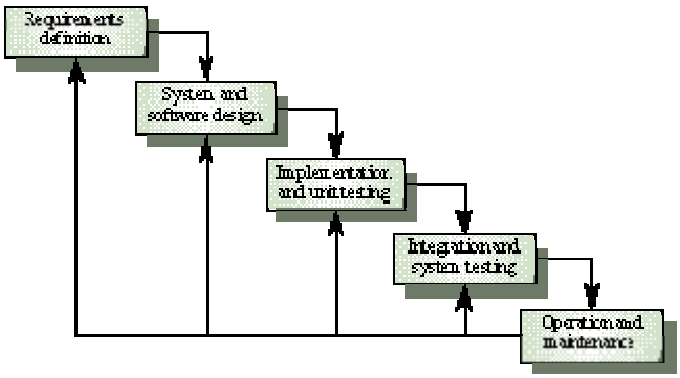



Figura 2-1 Modelo Cascata (PRESSMAN, 1995).

## 2.3 PROCESSO ESPIRAL

Baseado em prototipação e simulação fase a fase, cada uma formando um ciclo em espiral. (...) As fases correspondem às atividades de “determinar objetivos, alternativas e restrições; Avaliar alternativas, identificar e resolver riscos; desenvolver e verificar produtos de nível seguinte e planejar fases seguintes” (BOEHM, 1988). A figura 2-2 representa o desenvolvimento espiral.

O processo em espiral é dividido em iterações, ou ciclos, que iniciam no centro e caminha progressivamente para o exterior, à medida que o processo evolui. A cada iteração, é gerada uma versão mais completa do sistema, também chamada de prototipação evolutiva (UTAH, 2006), seus resultados são avaliados, o novo ciclo é planejado e seus riscos avaliados.

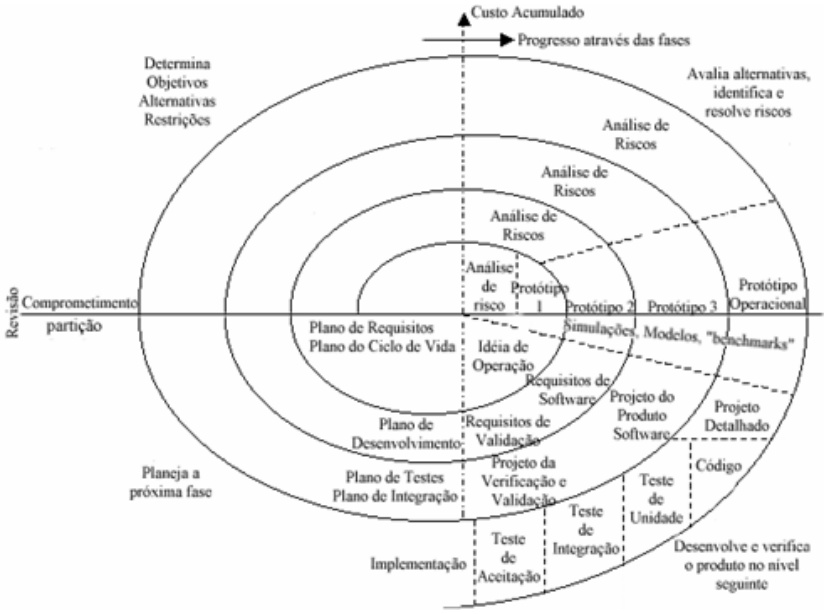


Figura 2-2 Processo Espiral (BOEHM, 1988).

## 2.4 PROCESSO UNIFICADO

### 2.4.1 Introdução

O Processo Unificado, assim como qualquer modelo de projeto de engenharia de *software,* objetiva, basicamente, definir quem deve estar fazendo o que, quando e como. Ao definir as competências, chamadas originalmente de *workers* e, mais recentemente, de *roles*, o processo unificado procura definir quem está fazendo parte de um determinado momento do processo de desenvolvimento. Cada atividade desenvolvida por uma competência responde como proceder para a produção de um artefato, que por sua vez é a tradução do que se fazer. As atividades, devidamente agrupadas em um fluxo de trabalho, compõem um *workflow*, que permite a definição de quando cada uma dessas atividades ocorrerá. A figura 2-3 apresenta os elementos básicos do Processo Unificado e o inter-relacionamento desses elementos.

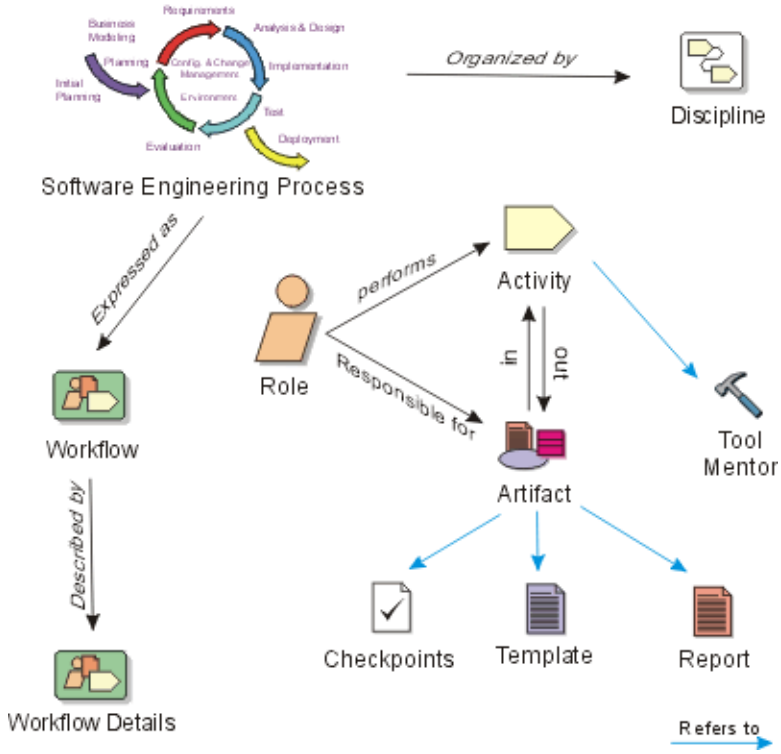



Figura 2-3 Elementos Básicos do Processo Unificado (RUP, 2004).

Um processo de Engenharia de *Software* (*Software Engineering Process*) é um conjunto de tarefas organizadas a fim de descrever como atingir uma meta e é

organizado em disciplinas que por sua vez podem ser entendidas como um conjunto de atividades inter-relacionadas. Um processo é expresso como um fluxo de trabalho (*workflow*), que apresenta quando cada um de seus detalhamentos (*workflow details*) ocorrerá.

Cada papel (*role*) executa uma ou mais atividades (*activity*), que é uma unidade de trabalho padrão e previamente definida. Além de receber uma orientação sobre o trabalho, é necessário que haja um mentor de ferramentas (*tool mentor*) que irá indicar qual a ferramenta ideal para realizar as atividades. É de responsabilidade de cada papel a elaboração de artefatos (*artifact*) que podem ser na forma de pontos de verificação (*check-points*), templates ou relatórios (*report*).

Como mostrado na figura 2-4, o Processo Unificado possui duas dimensões. O eixo horizontal, denominado Fases, mostra os aspectos do ciclo de vida do processo ao longo do tempo. Nesta dimensão estão retratados os aspectos dinâmicos do processo e expressa, para cada fase, suas iterações e marcos. O eixo vertical representa as Disciplinas, que agrupam as atividades de acordo com sua natureza. Nesse eixo são apresentados os aspectos estáticos do processo, como ele é descrito em termos de componentes, disciplinas, atividades, fluxos de trabalho, artefatos e papéis do processo (RUP, 2004).

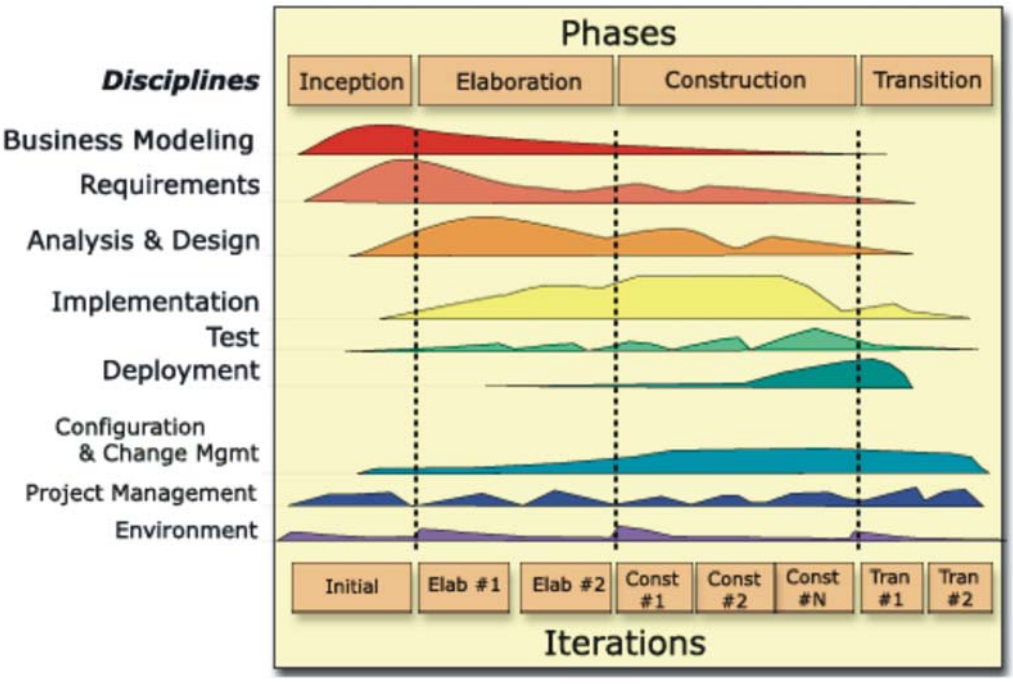



Figura 2-4 Estrutura Básica do Processo Unificado (RUP, 2004).

Ao contrário do Modelo Cascata, onde cada atividade começava quando a anterior estivesse concluída, neste modelo, cada disciplina acontece com maior ou menor ênfase, de acordo com a fase em que se encontra. Não é necessário que se termine uma atividade completamente para se começar a seguinte, elas podem acontecer em paralelo. Entretanto, algumas disciplinas requerem maior ou menor esforço de trabalho, de acordo com a fase em que se encontram. Esse paralelismo das atividades acontece em função do desenvolvimento iterativo e incremental descrito abaixo.

Todo o Processo Unificado norteia-se em tentar atacar as principais causas do insucesso no desenvolvimento de sistemas, através da utilização combinada de práticas consolidadas no mercado mundial que tenham apresentado resultados significativos e agregado valor ao produto final de um processo de desenvolvimento de sistemas. Essas práticas, denominadas 'As Melhores Práticas' estão divididas em seis grupos, a saber:

Desenvolver *Software* Iterativamente: uma alternativa ao desenvolvimento em cascata, baseia-se no modelo espiral de Barry Boehm, como mostrado na figura 2-5. Sua principal característica é de atacar os maiores riscos nas primeiras iterações, quando os custos são menores.


Figura 2-5 Desenvolvimento Iterativo (RUP, 2004).

Gerenciar Requisitos: em função de seu caráter dinâmico, gerenciar os requisitos de um sistema é um grande desafio que envolve diretamente o trabalho de toda a equipe de desenvolvimento. Ademais, identificar os requisitos reais do sistema é uma tarefa que se estende durante todo o processo, não somente pelo seu

dinamismo, mas também porque o entendimento por parte da equipe de desenvolvimento e a prioridade atribuída pelos futuros usuários do sistema também estão sujeitos à alteração. Neste contexto, gerenciar significa, portanto, elicitar, organizar e documentar os requisitos do sistema.

Usar Arquitetura Baseada em Componentes: um componente pode ser definido como uma parte encapsulada do sistema que atende a uma função bem definida dentro de um contexto. Com isso, as arquiteturas de componentes, baseadas em módulos independentes e substituíveis, auxiliam no gerenciamento da complexidade e induzem ao reuso. “A Arquitetura de *software* está preocupada não somente com estrutura e comportamento, mas também com usabilidades, funcionalidade, desempenho, confiabilidade, reusabilidade, compreensibilidade, economia e restrição tecnológica e comercial e preocupações estéticas” (KRUCHTEN, 2000: 10).

Modelar *Software* Visualmente: BOOCH (2000:6) define, de forma bastante clara, que um modelo é uma simplificação da realidade e ainda, que modelos são construídos para compreender melhor o sistema que está sendo desenvolvendo. Modelar permite visualizar e, principalmente, criticar o sistema antes que este comece a ser construído. Entretanto, para se tirar proveito da modelagem para a comunicação interna entre os membros da equipe é necessário, antes de qualquer coisa, que exista um padrão na linguagem de modelagem utilizada, caso contrário, a equipe se torna suscetível a inconsistências, ambigüidades e incoerências. Existem diversos padrões de linguagem de modelagem. Entretanto, a mais utilizada e recomendada pelo próprio Processo Unificado é a UML[1].

Verificação Continuada da Qualidade do *Software*: a figura 2-6 mostra que problemas em *software* podem se tornar de cem a mil vezes mais caro de encontrar e consertar no final do desenvolvimento do que no início. Por esse motivo, é de suma importância que se esteja verificando continuamente a qualidade do produto que está sendo desenvolvido em relação aos aspectos de desempenho, funcionalidade, segurança e usabilidade.

Gerenciar Configurações e Mudanças: um grande desafio no desenvolvimento

[1] Acrônimo para Unified Modeling Language.

de um sistema é quando se precisa integrar o trabalho de vários membros de uma equipe, controlar a interação de diferentes equipes e ainda administrar o trabalho realizado em locais fisicamente distantes e em diferentes plataformas. Gerenciar configurações e mudanças de um projeto se resume em basicamente três atividades essenciais: coordenar as atividades e os artefatos, coordenar iterações e versões e controlar as mudanças do *software*.

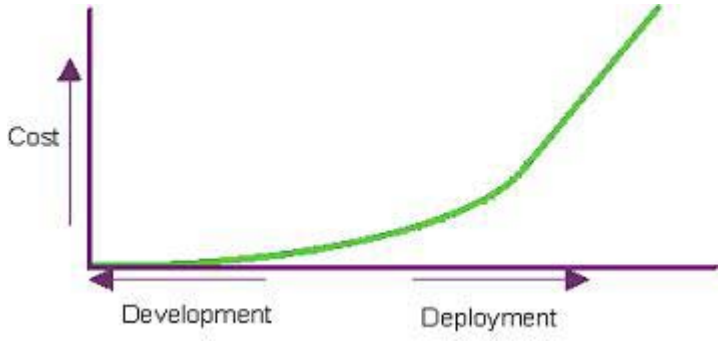



Figura 2-6 Custo de se tratar um problema (RUP, 2004).

## 2.4.2 Artefatos de um Processo

Sempre que se fala em artefato de desenvolvimento de sistema, pensa-se logo no programa executável, como se este fosse o único produto gerado que realmente interessa. Como de fato, um programa executável é de extrema importância e a razão da existência dos demais, mas não se pode ficar apenas nessa definição. Artefato é, na verdade, qualquer produto gerado por uma atividade realizada por um ou mais membros da equipe de desenvolvimento. Como as diversas atividades estão intimamente ligadas, um artefato gerado por uma atividade poderá servir, não apenas para ser entregue ao usuário final, como executáveis, manuais e etc., mas também como entrada para uma atividade subseqüente. Em alguns casos, o produto ou artefato gerado por uma atividade poderá ser o simples refinamento de outro. A figura 2-7 apresenta a inter-relação entre os principais artefatos do Processo Unificado.

A função de qualquer integrante da equipe de desenvolvimento é, portanto, gerar artefatos, em diferentes momentos, com diferentes características ou grau de importância. Os artefatos iniciais, produzidos na etapa de levantamento de requisitos servirão como base, ou *input* para todos os demais, até se chegar ao código executável. Todas as atividades subseqüentes dependem diretamente dos artefatos gerados nesse momento, daí, portanto, sua grande importância no projeto. Testar um

sistema consiste, acima de tudo, comparar o produto final com o que foi especificado inicialmente. Mas onde obter o que foi especificado inicialmente? Nos artefatos da disciplina de requisitos. Da mesma forma, seguem as demais disciplinas de análise e *design*, implementação e implantação.

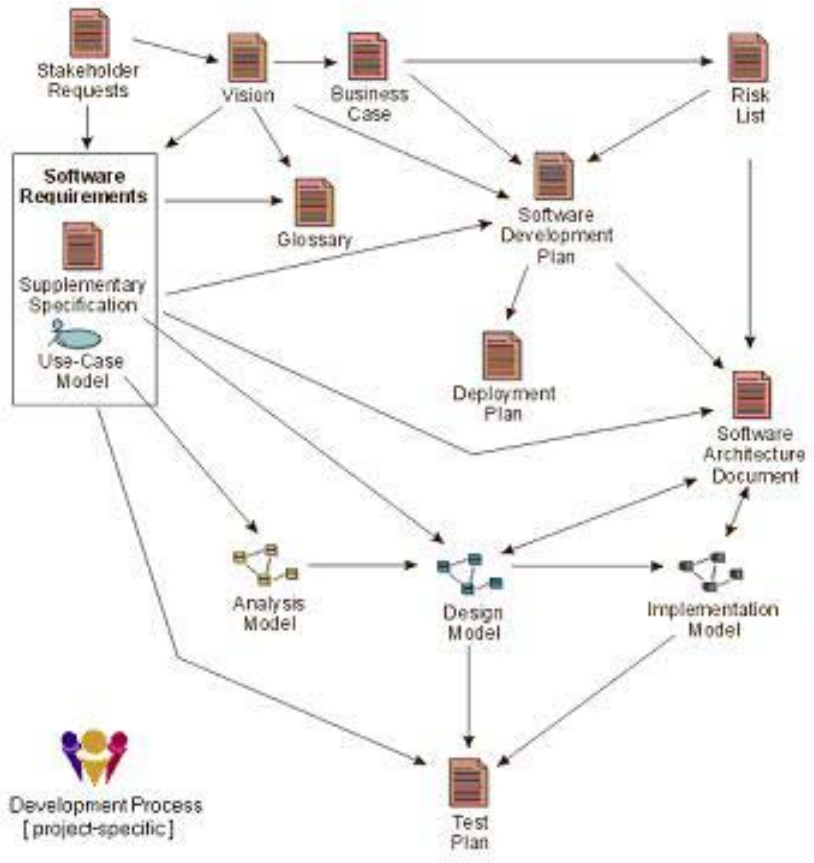



Figura 2-7 Fluxo de informação entre os principais artefatos do Processo Unificado (RUP,2004).

## 2.5 AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

Um Ambiente de Desenvolvimento de Software (ADS) é um sistema computacional que provê apoio para o desenvolvimento, reparo e melhorias em software e para o gerenciamento e controle destas atividades, contendo uma base de dados central e um conjunto de ferramentas de apoio. A base de dados central atua como um repositório para todas as informações relacionadas ao projeto ao longo do seu ciclo de vida, e as ferramentas oferecem apoio para as várias atividades técnicas e gerenciais passíveis de automação que devem ser realizadas no projeto (MOURA & ROCHA, 1992). Um ADS envolve o apoio a atividades individuais, ao trabalho em

grupo, ao gerenciamento do projeto, ao aumento da qualidade geral dos produtos e ao aumento da produtividade, permitindo ao engenheiro de software acompanhar o trabalho e medir a sua evolução, através de informações obtidas ao longo do seu desenvolvimento (TRAVASSOS, 1994).

As estruturas de organização de ADS propõem diferentes componentes para compor os ambientes e um conjunto de funcionalidades para prover apoio aos desenvolvedores de software de forma integrada (MACHADO, 2000).

## 2.6 QUALIDADE EM SOFTWARE

### 2.6.1 CMM

Em novembro de 1986, o *Software Engineering Institute* (SEI) iniciou o desenvolvimento de uma estrutura de maturidade de processo com o intuito de auxiliar empresas, em especial aquelas ligadas diretamente ao governo americano, a melhorar seus processos de desenvolvimento de *software*. Em setembro de 1987, o SEI disponibilizou uma breve descrição da estrutura de maturidade de processo. Após quarto anos de experiência, a estrutura de maturidade de processo evoluiu para um modelo completamente definido (SEI, 1993). Desde então, o CMM (*Capability Maturity Model*) do SEI tem se mostrado como o maior indicador de qualidade e maturidade no processo de desenvolvimento de software. Não obstante, é possível não deter o CMM e mesmo assim desenvolver um projeto com êxito, mas para tanto dependerá do talento individual dos membros da equipe, o que por si só é um grande risco para qualquer projeto. Gestão e planejamento são vitais para que haja ganho de previsibilidade. Daí a importância do foco nos processos e não nas características de uma dada equipe. Não se trata de descartar as práticas e a cultura de desenvolvimento consolidadas, mas sim de mapear a aderência ao *modus operandi* que o CMM pressupõe. A meta final, mais uma vez, é assegurar custos, prazos e qualidade, dentro do previsto. Havendo mudanças de percurso, os possíveis impactos têm de ser administrados.

O CMM é uma estrutura, ou conjunto de normas, que descreve os elementos de um efetivo processo de desenvolvimento de *software*. Seu objetivo é descrever passos evolucionários de um processo imaturo até atingir a maturidade e disciplina no desenvolvimento de sistemas. Por ele, são cobertas práticas de planejamento,

engenharia e gerenciamento de desenvolvimento de manutenção de *softwares*.

O CMM estabelece cinco níveis de maturidade, conforme ilustrado na figura 2.8. Cada nível de maturidade é composto por uma área chave de processo que por sua vez identifica um grupo de atividades relacionadas, que, quando executadas coletivamente, remetem a equipe ao alcance de metas importantes para o estabelecimento de maturidade no processo.

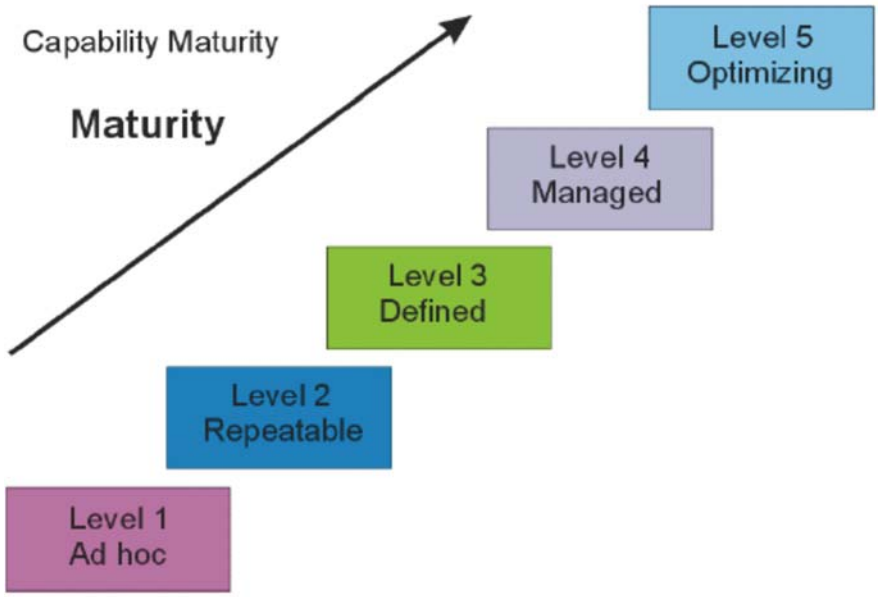



Figura 2-8 Os níveis do CMM (SEI, 1993).

- Primeiro Nível: Inicial

Encontram-se nesse nível empresas que não possuem um processo bem definido e estável de desenvolvimento e manutenção de *software*. O sucesso de um projeto depende exclusivamente do brilhantismo individual dos membros da equipe. Para que um processo seja repetido, é necessária a repetição da mesma equipe. Nesse nível, portanto, a capacidade está galgada nos indivíduos, não na empresa.

- Segundo Nível: Repetível

No segundo nível, políticas de gerenciamento de projeto e procedimentos para implementar tais políticas são estabelecidos. O planejamento e o gerenciamento de novos projetos são baseados na experiência em projetos similares. Um processo eficiente pode ser caracterizado como praticado, documentado, reforçado, treinado, medido e capaz de ser melhorado.

As áreas chaves de processo do segundo nível são gerência de requisitos, planejamento do projeto de *software*, supervisão e rastreamento do projeto, gerenciamento de subcontratação e gerência de configuração de *software*.

O objetivo da gerência de requisitos, segundo o CMM, é estabelecer um entendimento comum entre as partes evolvidas sobre o projeto em questão. Esse entendimento comum é base para o planejamento e gerenciamento do projeto. Controlar o relacionamento com o cliente depende de seguir um processo de controle de mudança efetivo. Portanto, atingir o segundo nível de maturidade, depende diretamente de atingir maturidade no gerenciamento dos requisitos e das mudanças ocorridas ao longo do processo de desenvolvimento.

- Terceiro Nível: Definido

Nesse nível, os processos de desenvolvimento e manutenção são bem documentados e integrados. Processos estabelecidos no terceiro nível são usados para apoiar gerentes e corpo técnico na realização de suas atividades de forma mais eficiente. As áreas chaves desse nível são foco da organização nos processos, definição clara dos Processos da organização, programa de treinamento e engenharia de produtos de software.

- Quarto Nível: Gerenciável

No quarto nível, a organização é capaz de estabelecer metas quantitativas para os seus produtos e processos. Medidas de qualidade e produtividade são coletadas em todos os projetos através de avaliação e análise contínuas de do desempenho, com isso, os projetos melhoram o seu controle sobre os produtos e processos, o que diminui a variância das medidas. Uma organização no nível quatro passa a ter uma gestão baseada em termos quantitativos.

- Quinto Nível: Otimizado

No ultimo nível de maturidade, a organização está engajada na melhoria contínua de seus processos. Pontos fracos e defeitos são identificados e ações preventivas sobre as causas são realizadas a fim de elevar continuamente a qualidade do processo. Mudanças mais significativas de processos ou de tecnologias são feitas a partir de análises de custo/benefício com base em dados quantitativos cuja coleta iniciou-se no nível anterior.

### 2.6.2 CMMI

O modelo CMM tornou-se o modelo de maturidade de processos mais conhecido, usado e respeitado. Baseado nesse sucesso, outros modelos foram criados, procurando cobrir outras áreas de interesse. Assim surgiram outros modelos (SEI, 2002): (1) *Software Acquisition CMM* (SA-CMM): usado para avaliar a maturidade de uma organização em seus processos de seleção, compra e instalação de software desenvolvido por terceiros; (2) *Systems Engineering CMM* (SE-CMM): avalia a maturidade da organização em seus processos de engenharia de sistemas, concebidos como algo maior que o software. Um sistema inclui o hardware, o software e quaisquer outros elementos que participam do produto completo; (3) *Integrated Product Development CMM* (IPD-CMM): ainda mais abrangente que o SE-CMM, inclui também outros processos necessários à produção e suporte ao produto, tais como suporte ao usuário, processos de fabricação etc.; (4) *People CMM* (P-CMM): avalia a maturidade da organização em seus processos de administração de recursos humanos no que se refere a *software*; recrutamento e seleção de desenvolvedores, treinamento e desenvolvimento, remuneração etc.

Com o intuito de integrar todos esses modelos, o SEI iniciou o projeto do CMMI (*CMM Integration*), com o objetivo de apoiar a melhoria de processos e produtos diminuindo a redundância e eliminando a inconsistência que surge ao se utilizar diversos modelos independentes. Uma outra preocupação do projeto foi a compatibilidade com a norma ISO 15504, de modo que avaliações em um modelo sejam reconhecidas como equivalentes aos do outro.

O CMMI oferece duas abordagens diferentes para a melhoria de processos. Essas duas abordagens são conhecidas como o “modelo contínuo” e o “modelo em estágios”.

O SW-CMM é um modelo em estágios. Existem cinco níveis de maturidade e a organização é avaliada como estando em apenas um deles. O modelo da norma ISO/IEC TR 15504 usa um modelo contínuo. Neste caso, cada área-chave de processo possui características relativas a mais de um nível. Assim, uma área-chave que, no modelo em estágios, pertence exclusivamente ao nível dois, no modelo contínuo pode ter características que a coloquem em outros níveis. No modelo contínuo, cada área chave é classificada separadamente, de modo que a organização

pode ter áreas no nível um, outras no nível dois, ainda outras no nível três e assim por diante.

Para dar liberdade a organizações com realidades e necessidades distintas, o CMMI admite as duas abordagens. Ambas contêm essencialmente as mesmas informações e a opção pelo modelo contínuo ou em estágios depende de cada organização. Cada modelo possui características que o tornam mais apropriado em uma situação ou outra.

O modelo em estágios oferece um caminho para melhoria de processos, indicando a ordem de implementação para cada área de processo, de acordo com os níveis de maturidade. Essa abordagem minimiza os riscos da melhoria de processos. A representação é indicada para organizações realmente comprometidas com a implantação do CMM em escala geral.

O modelo contínuo oferece uma abordagem mais flexível para a melhoria de processos, embora mais complexo de administrar. É indicado para organização que desejam dar prioridade à melhoria de uma área de processo ou conjunto de processos, de acordo com seus objetivos de negócio. Este modelo permite fácil comparação à ISO/IEC TR 15504, porque a organização das áreas de processo é derivada desta norma (BERGER, 2003).

# 3 A GERÊNCIA DE REQUISITOS

No presente capítulo é apresentado o resumo dos estudos realizados sobre a Gerência de Requisitos da Engenharia de *Software*. Inicialmente é apresentado um conceito de requisito de *software*, seguido de uma análise sobre as principais causas das mudanças nos requisitos durante o processo de desenvolvimento de sistemas. Adiante, são apresentadas uma análise da gerência de requisitos sob a perspectiva do Processo Unificado, uma especificação dos principais artefatos de requisitos e algumas ferramentas utilizadas para controle de requisitos.

## 3.1 INTRODUÇÃO

Segundo o Dicionário Aurélio da Língua Portuguesa, requisito é uma condição necessária para alcançar certo objetivo (HOLANDA, 1988). Certamente, o objetivo de todo desenvolvedor é a criação de um produto final que agregue qualidade ao ambiente onde será implantado, portanto é condição *si ne qua non* que todo *software* atenda aos requisitos estabelecidos pelas partes envolvidas com o sistema. No entanto, o amadurecimento da prática de engenharia de *software* tem mostrado, que os requisitos dos sistemas são elementos dinâmicos que mudam durante o processo de desenvolvimento dos sistemas. Controlar essas mudanças corresponde a um grande desafio para a equipe de gerência de requisitos. As principais preocupações da gerência de requisitos são: gerenciar mudanças nos requisitos acordados; gerenciar os relacionamentos entre os requisitos; gerenciar as dependências entre o documento de requisitos e outros documentos produzidos ao longo do processo (HAZAN & LEITE, 2003). A utilização de ferramentas apropriadas para o gerenciamento das mudanças nos requisitos é de grande importância, pois imprime maior qualidade ao processo de engenharia de *software*, o que, conseqüentemente, acarreta em qualidade no produto final gerado.

## 3.2 CONCEITO DE REQUISITOS

Um requisito é “uma condição ou uma capacidade que o sistema deve atender” (KRUCHTEN, 2000:157), pode ser funcional ou não funcional. Um requisito funcional é aquele que trata das ações que um sistema deve ser capaz de executar em favor do usuário final. Expressa o comportamento esperado do sistema, com base em suas entradas e saídas.

Requisitos não funcionais são aqueles que agregam qualidade ao sistema, e são tão necessários ao sistema quanto os requisitos funcionais. A supressão desses requisitos compromete a qualidade do sistema final. Para simplificar o entendimento, os requisitos costumam ser caracterizados como o modelo FURPS+ (GRADY, 1992), que significa:

F: *Funcionality* ou Funcionalidade: refere-se ao conjunto recursos e habilidades que o sistema deve prover;

U: *Usability* ou Usabilidade: refere-se aos fatores humanos, estéticos, facilidade e padronização de acesso, documentação e qualquer outro tipo de ajuda na utilização do sistema;

R: *Reliability* ou Confiabilidade: refere-se à freqüência, gravidade e previsibilidade das falhas no sistema, possibilidade de recuperação e precisão dos dados;

P: *Performance* ou Desempenho: quanto ao tempo de resposta, velocidade de acesso, consumo de recurso, disponibilidade do *software*, taxa de transferência, entre outros;

S: *Suportability* ou Suportabilidade: refere-se à capacidade do *software* de sofrer manutenções, extensões, adaptações e configurações. Compatibilidade e portabilidade também pertencem a esta categoria.

+: engloba os demais requisitos que não se encaixam nas classes acima, tais como restrição de projeto, atendimento a padrões de linguagem programação ou banco de dados e limitações de recursos, requisitos de interfaces com outros sistemas ou equipamentos.

## 3.3 PORQUE OS REQUISITOS MUDAM

Vários fatores conduzem às mudanças nos requisitos. Segundo BREITMAN (1998), evolução é “o processo de transformação sofrido por uma especificação ao longo da vida útil da aplicação que esta corresponde. Desta forma, uma especificação evolui durante as fases de desenvolvimento e manutenção da aplicação”. Mais adiante, completa com a afirmação de que o principal motivo das mudanças evolutivas é “o fato dos próprios usuários não conhecerem a totalidade dos requisitos do sistema no momento de sua elicitação”. Mudanças também decorrem da identificação de novas necessidades e do processo evolutivo de conhecimento do mundo real por parte da equipe desenvolvedora.

Este caráter evolutivo dos requisitos dos sistemas é “uma característica aceita pela maior parte dos desenvolvedores de sistemas e obriga a existência de um gerenciamento desta evolução, provendo mecanismos para manter a rastreabilidade, o controle de configuração e a eliminação de inconsistências e desatualizações” (BERGMANN, 2003:52).

## 3.4 A GERÊNCIA DE REQUISITOS SOB A PERSPECTIVA DO PROCESSO UNIFICADO

A atividade de gerenciar requisitos é “uma abordagem sistemática de encontrar, documentar e manter requisitos. Sem isso, mais que dois terços dos projetos acabam perdendo as necessidades dos usuários, se atrasam ou extrapolam o orçamento. Por que tanto? Em primeiro lugar, porque gerenciar requisitos significa gerenciar mudanças – e gerenciar mudanças é difícil. Acrescido do fato de mudança ser difusa. Vivemos em um mundo dinâmico: clientes mudam suas formas de pensar, concorrentes apresentam soluções melhores antes que entreguemos os nossos, o ambiente de negócio muda. Estar aberto a este ambiente de mudança é um bom negócio, mas pode trazer problemas. Mudança por si só não é ruim. O grande problema é a mudança sem controle – mudanças cujo impacto não é medido antes que elas aconteçam. Gerenciando requisitos, você está mais próximo de entregar em tempo uma solução que resolva os problemas reais de seus clientes.” (RATIONAL, 2004:3).

Como base para as demais etapas do desenvolvimento, qualquer descontrole

ou erro na gerência de requisitos compromete diretamente a qualidade final do sistema em desenvolvimento, afinal, o objetivo principal de qualquer sistema é atender os requisitos propostos pelos usuários. Se os requisitos não estão bem gerenciados, não há como garantir sua qualidade.

Mudanças nos requisitos são inevitáveis e não há como ignorá-las. Desde pequenos ajustes em algumas especificações até grandes mudanças de curso, é fatal que elas aconteçam em um processo de desenvolvimento de sistema.

Negligenciar a gerência dos requisitos significa negligenciar a qualidade do produto final. Para controlar os requisitos e suas mudanças, é necessário certo grau de formalismo em relação à forma de administrá-los. Portanto, gerenciar mudanças de requisitos deve ser uma atividade formalmente prevista no processo que consiste em gerenciar, integrar e, quando necessário, rejeitar uma solicitação de mudança. Uma forma simplificada de se controlar as mudanças nos requisitos de um sistema está apresentada na figura 3-1:

| Change Request | |
|---|---|
| **Change Request Item** | **Value** |
| Change Request ID | CR001 |
| Change Request Name | Safety Feature on Power On Button |
| Brief Description of Change | Add hold time to Power On button that requires user to hold button for xx seconds before system turns on |
| Requested by… | Safety Supervisor |

Figura 3-1 Controle de Mudança de requisitos (LEFFINGWELL, 2003: 11).

O Processo Unificado propõe o detalhamento do fluxo de trabalho de Gerenciar Requisitos Variáveis, cujo objetivo torneia entre avaliar as solicitações de mudança e definir seu impacto, estruturar o modelo de casos de uso e configurar os atributos e a rastreabilidade dos requisitos. A figura 3-2 apresenta o detalhamento do fluxo de trabalho Requisitos Variáveis onde estão previstas três atividades:

**Estruturar o Modelo de Caso de Uso**: de responsabilidade do analista de sistemas, objetiva reestruturar os casos de uso e atores afetados pelos requisitos.

**Gerenciar Dependências**: cujo objetivo é “usar os atributos e a rastreabilidade

dos requisitos do projeto para auxiliar no gerenciamento do escopo do projeto e gerenciar requisitos variáveis" (RATIONAL, 2004). Esta atividade, por sua vez, divide-se em três passos distintos: Designar Atributos: "os atributos mais importantes são benefício (por parte dos envolvidos), esforço para implementação, risco para o esforço de desenvolvimento, estabilidade (probabilidade de não sofrer mudanças) e impacto na arquitetura (é significativo do ponto de vista da arquitetura) de cada requisito (...) Os requisitos instáveis com alto risco, esforço ou benefício deverão ser sinalizados para uma análise mais detalhada. Os requisitos de poucos benefícios com alto esforço, risco ou instabilidade deverão ser sinalizados para possível remoção." (RATIONAL, 2004). Estabelecer e Verificar a Rastreabilidade: objetiva garantir a integridade entre os diversos artefatos inter-relacionados. Gerenciar Requisitos Variáveis: uma mudança não acontece necessariamente em um requisito, pode acontecer também em seus atributos ou em sua rastreabilidade e também devem ser gerenciados. Mudanças em requisitos, normalmente, possuem o efeito cascata, impactando em outros requisitos ou artefatos. Desta forma, devem-se gerenciar as mudanças do mais alto para o mais baixo nível. Em primeiro lugar, deve-se revisar o impacto no documento de visão, depois no modelo de caso de uso, modelo de design e, finalmente, no material de suporte ao usuário.

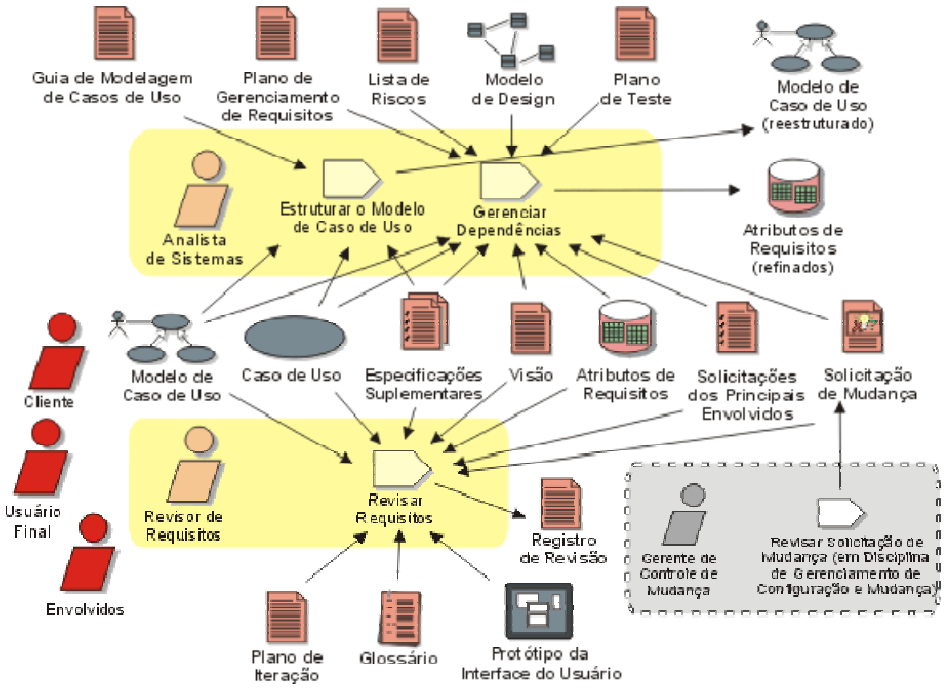



Figura 3-2 Detalhamento do fluxo de trabalho: Gerenciar requisitos Variáveis (RATIONAL: 2004).

**Revisar Requisitos**: atividade realizada pelo revisor de requisitos, consiste em verificar formalmente a conformidade do resultado das atividades anteriores com a expectativa do cliente. Essa verificação deve ser realizada em uma ou mais reuniões.

## 3.5 ARTEFATOS DE REQUISITOS

A prática das atividades de engenharia de requisitos, em um processo de desenvolvimento de sistemas, resulta na construção de artefatos que objetivam especificar o problema atual e trilhar caminhos que levam à construção da solução para tal problema. Os artefatos produzidos durante o levantamento de requisitos estão fortemente inter-relacionados a praticamente todos os artefatos gerados nas demais disciplinas de engenharia de *software* e por esse motivo, são altamente complexos. A tabela 3-1 apresenta a relação dos artefatos de requisitos sugeridos pelo processo unificado.

| Artefato | Definição |
|---|---|
| Plano de Gerenciamento de Requisitos (Anexo 01) | Desenvolvido pelo analista de sistemas durante a fase de concepção do sistema, esse artefato deve especificar as informações e controlar os mecanismos que serão coletados e usados para avaliar, documentar e controlar mudanças nos requisitos do produto.<br>A finalidade do Plano de Gerenciamento de Requisitos é descrever como o projeto irá configurar documentos de requisitos, tipos de requisitos e seus atributos e rastreabilidade. Deve ser atualizado em cada marco principal do desenvolvimento. |
| Solicitações das Partes Envolvidas (Anexo 02) | A finalidade desse artefato, controlado pelo analista de sistemas, é capturar todas as solicitações feitas em relação ao projeto e o modo como elas foram abordadas. Neste caso, não só o analista de sistemas, mas também todos os envolvidos no sistema são responsáveis pelo seu desenvolvimento, pois seu conteúdo é fruto das necessidades e expectativas de cada um. Devido a sua extensão e a sua diversidade de fonte, é fundamental que o controle deste artefato seja feito através de |

| | |
|---|---|
| | uma ferramenta, já que não basta saber quais são os requisitos, mas vários outros atributos, como sua origem, prioridade, taxa de esforço, causa, risco, implicabilidade, benefício e estabilidade. |
| Glossário (Anexo 03) | Fundamental para dirimir qualquer dualidade das definições de termos críticos necessários para o sistema, esse artefato é desenvolvido pelo analista de sistemas e usado por toda a equipe técnica durante o processo de desenvolvimento do sistema. |
| Documento de Visão ou de Requisitos do Produto (Anexo 04) | Criado no início da fase de concepção, pode ser usado como base contratual entre as partes envolvidas. Seu objetivo é capturar os requisitos em seu mais alto nível e possíveis restrições de projeto.<br>É desenvolvido pelo analista de sistemas e lido pelos gerentes, patrocinadores e equipe de desenvolvimento em geral.<br>Seu conteúdo pode ser alterado, à medida que evolui o processo de desenvolvimento do sistema e, conseqüentemente, o entendimento das partes envolvidas. |
| Especificações Suplementares (Anexo 05) | As Especificações Suplementares são geridas pelo analista de sistemas, criadas na fase de Concepção do sistema e refinadas nas fases subseqüentes de elaboração e construção.<br>Servem para capturar os requisitos de sistema que não são contemplados pelos casos de uso. Esses requisitos podem ser:<br>Requisitos legais e de regulamentação e padrões de aplicativo.<br>Atributos de qualidade do sistema a ser criado, incluindo requisitos de usabilidade, confiabilidade, desempenho e suportabilidade.<br>Outros requisitos, como sistemas operacionais e ambientes, requisitos de compatibilidade e restrições de *design*. |
| Atributos dos Requisitos | Este artefato contém um repositório de textos, dependências, atributos e rastreabilidade de todos os requisitos de projeto.<br>Deve ser gerido pelo analista de sistemas e acessado por toda a equipe de desenvolvimento. |

| | |
|---|---|
| Especificação dos Requisitos de *Software* (Anexo 06) | Um pacote que se concentra em coletar e organizar todos os requisitos que envolvem o projeto, tais como os casos de uso, modelo de casos de uso e Especificações Suplementares. |
| Modelo de Caso de Uso | Representa o ambiente e os requisitos funcionais necessários ao sistema. Seu principal objetivo é ser usado como fonte de informação para a equipe de desenvolvimento na disciplina de análise e design e na disciplina de teste. |
| Classes de Fronteira | Objetiva representar a inter-relação entre o sistema e seu mundo externo e intercambiar as informações entre estes. Servirá como base para a criação do protótipo do sistema. |
| Especificação dos Casos de Uso (Anexo 07) | Descrição textual e lógica dos casos de uso e sua interação com os atores. |

Tabela 3-1 - Artefatos de Requisitos.

## 3.6 FERRAMENTAS PARA CONTROLE DE REQUISITOS

### 3.6.1 Repositório de Requisitos

Esse grupo de ferramentas de gerência de requisitos deve prover um repositório central de informações acerca das necessidades apresentadas pelos usuários e por todas as partes envolvidas com o sistema a ser desenvolvido (DAVIS & LEFFINGWELL, 1996:2). Esse repositório deve envolver os requisitos, seus atributos, inter-relacionamentos existentes e qualquer outra informação gerencial pertinente ao ambiente. Tal repositório não é importante apenas para o sucesso do projeto, ele passa a fazer parte do acervo da empresa, uma vez que os requisitos podem ser evoluídos, adaptados e reusados, acarretando em minoração dos custos e prazos de projetos seguintes. A figura 3-3 apresenta uma macro-visão de seu funcionamento.

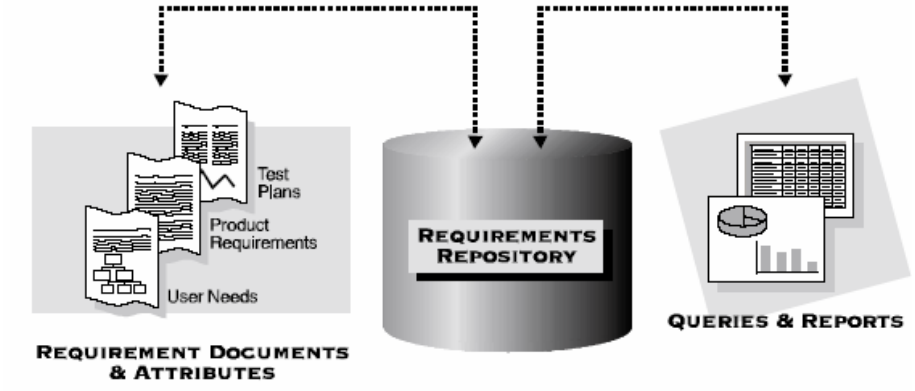



Figura 3-3 Repositório de Requisitos (DAVIS & LEFFINGWELL, 1996).

Fazem parte desse grupo, entre outras, as ferramentas RequisitePro[2], Caliber-RM[3], Doors[4], RDT[5], SLATE[6] e REM. Desse grupo de ferramentas, esperam-se algumas características básicas a fim de resolver problemas específicos, conforme apresentado na tabela 3-2.

| Características | Problema a resolver |
|---|---|
| Predisposição para detalhamento textual dos requisitos através de integração dinâmica com *softwares* editores de textos ou através de armazenamento de texto explicativo (BATISTA, 2003). | Requisitos armazenados em bancos de dados carecem de contexto e são difíceis para os usuários entenderem. Requisitos armazena-dos somente em editores de texto são difíceis de organizar, priorizar e rastrear. |
| Integração com outras ferramentas de desenvolvimento de sistema (BREITMAN, 1998). | Equipes de desenvolvimento e teste precisam ter acesso ao estado mais atual dos requisitos. Contudo, duplicar os requisitos em ferramentas de desenvolvimento e teste é ineficiente e propenso a falhas. |
| Acesso distribuído dos requisitos (LOPES, 2003). | Equipes geograficamente distribuídas podem não estar providas com as |

[2] RATIONAL RequisitePro: Rational Software Corporation, 2003. Conjunto de Programas. (RATIONAL, 2004).
[3] BORLAND Caliber-RM: Borland Software Corporation, 2005. Conjunto de Programas. (BORLAND, 2005).
[4] TELELOGIC Doors: Telelogic North America Inc., 2005. Conjunto de Programas. (TELELOGIC, 2005).
[5] IGATECH RDT: Igatech Systems Pty Ltd, 2004. Conjunto de Programas. (IGATECH, 2005).
[6] UGS Slate: UGS Corporation, 2005. Conjunto de Programas. (UGS, 2005).

| | |
|---|---|
| | últimas informações dos requisitos e quando pressionados pelo tempo, podem prosseguir o seu trabalho com informações obsoletas. |
| Rastreabilidade dos requisitos (JARKE, 1998). | Funcionalidades esperadas e não implementadas causam frustrações aos usuários finais. |
| Impacto da mudança de um requisito (DAVIS, 1996). | Requisitos mudam e é difícil de se entender o impacto que uma mudança causa no sistema e em outros requisitos. |
| Permitir utilizar atributos e *queries* em requisitos (DAVIS, 1996). | Sem uma documentação de risco, prioridade e dificuldade associada com cada requisito, pode-se não conseguir definir objetivamente quais requisitos devem ser implementados primeiro. |
| Auditar as mudanças dos requisitos (TORO, 2000). | Muitas vezes é impossível dizer quem criou ou modificou um requisito e por que. |
| Repositório central para todos os elementos dos requisitos (TORO, 2000). | Requisitos são atualizados sem nenhum histórico de revisão, documentos de requisitos são compartilhados entre os membros da equipe e estes ficam sem ter certeza de qual é a mais atual, ou ainda, membros diferentes da equipe podem estar trabalhando com diferentes versões dos requisitos. |
| Acesso controlado às informações dos requisitos (LOPES, 2003). | Requisitos são modificados por membros da equipe não autorizados para fazê-lo, resultando em requisitos que não espelham efetivamente as necessidades dos usuários. |
| Estrutura customizavel de requisitos (KRUCHTEN, 2000). | Requisitos normalmente são numerosos e não organizados logicamente. |

| | |
|---|---|
| Oferta de *templates* de projeto (KRUCHTEN, 2000). | Retrabalho de definição de projetos semelhantes despendem tempo desnecessariamente. |
| Exportação dos conteúdos para um formato não proprietário (BREITMAN, 1998) | Os requisitos de um sistema precisam ser divulgados a todos os membros da equipe de desenvolvimento. Para isso, é importante que possam ser apresentados em formato não proprietário e independente da ferramenta de controle de requisitos. |

Tabela 3-2 Características esperadas de Ferramentas de Controle de Requisitos

### 3.6.2 Outras Ferramentas para Controle de Requisitos

Um outro grupo de igual relevância para a gerência de requisitos é composto pelas ferramentas CASE[7], em especial aquelas que geram diagramas de casos de uso. Atualmente, existem no mercado diversas ferramentas gratuitas e proprietárias que geram esse tipo de artefato. Dentre elas, podem-se citar Rational Rose[8] e Jude-Take[9]. A tabela 3-3 lista as características esperadas deste grupo de ferramentas e uma breve descrição dessas características:

| Características | Descrição |
|---|---|
| Aderência a padrões (FOWLER & SCOTT, 1997). | A ferramenta deve estar de acordo com as especificações básicas dos padrões ao qual ela se propõe apoiar, neste caso, a UML. |
| Inclusão, exclusão, movimentação, agrupamento e redimensionamento de objetos (PRESSMAN, 1995). | A ferramenta permite que os elementos, após serem criados, possam ser modificados. |
| Integração com outras Ferramentas (RUP, 2004). | Diagramas de casos de uso necessitam de especificações textuais para complementá-los. É necessário que as ferramentas |

[7] Acrônimo para Computer-Aided Software Engineering
[8] RATIONAL Rose: Rational Software Corporation, 2003. Conjunto de Programas (RATIONAL, 2004).
[9] JUDE-TAKE: Eiwa System Management, versão 1.3, 2004. Conjunto de Programas (JUDE, 2005).

| | |
|---|---|
| | permitam essa integração. |
| Portabilidade dos artefatos (ISO, 1991). | É possível exportar objetos criados pela ferramenta para outros formatos ou importar objetos criados em outros formatos para a ferramenta. |
| Organização gráfica (ISO, 1991). | Facilidade de manipulação e visualização gráfica dos elementos dos diagramas. |
| Diagramas da UML (BOOCH, 2000). | Embora nem todos os diagramas da UML sejam usados na disciplina de Requisitos, é importante que a ferramenta os suporte a fim integrar todas as atividades em uma única ferramenta. |

Tabela 3-3 Características Esperadas de Ferramentas CASE para UML

Requisitos, muitas vezes devem ser representados de forma textual, planilha de cálculos ou imagens. Esses tipos de informação também devem ser armazenados e controlados junto com os demais artefatos. Embora esse conjunto de ferramentas não possua uma utilização específica para documentação de sistemas, é um poderoso aliado para a representação dos requisitos na maioria dos sistemas. Por isso, algumas ferramentas têm como objetivo gerar *templates* utilizáveis em editores de texto, como o *Microsoft Word*[10]. Uma dessas ferramentas é o *Rational SoDA*[11], um gerador de relatórios que suporta a customização tanto de relatórios formais quanto informais. O objetivo principal desse tipo de ferramenta é automatizar a produção de documentação de *software*, reduzindo o esforço requerido para essa atividade.

[10] MICROSOFT Word: Microsoft Corporation, 2003. Conjunto de Programas (MICROSOFT, 2005).
[11] RATIONAL SoDA: Rational Software Corporation, 2003. Conjunto de Programas (RATIONAL, 2004).

# 4 GERÊNCIA DE CONFIGURAÇÃO E MUDANÇA

No presente capítulo é apresentado o resumo dos estudos realizados sobre a Gerência de Configuração e Mudança da Engenharia de *Software*. Inicialmente é apresentado um conceito de gerência de configuração e mudança, com ênfase nas atividades de gerência de configuração. Adiante, são apresentadas uma análise da gerência de configuração e mudança sob a perspectiva do Processo Unificado, uma especificação dos principais artefatos de configuração e mudança e algumas ferramentas utilizadas para controle de configuração.

## 4.1 INTRODUÇÃO

"A Primeira Lei da Engenharia de Sistemas declara: Não importa onde você se encontre no ciclo de vida do sistema, o sistema se modificará, e o desejo de mudá-lo persistirá em todo o ciclo de vida dele" (PRESSMAN, 1995: 917).

O *software* durante o seu ciclo de vida passa por diversos estágios e representações (OLIVEIRA, 2001:26). Seu desenvolvimento é um processo dinâmico que sofre mudanças de curso até a sua conclusão. Para que não haja inconsistência nos itens de informação importantes para o projeto, a criação e as alterações desses itens devem ser acompanhadas e controladas pelo gerente de projeto. O processo de gerência de configuração é um processo de ciclo de vida do *software* que permite que esse controle seja realizado (ROCHA *et al*, 2001: 59).

As atividades de Gerência de Configuração são desenvolvidas para "(1) identificar a mudança, (2) controlar a mudança; (3) garantir que a mudança esteja sendo adequadamente implementada; e (4) relatar a mudança a outras pessoas que possam ter interesse nela" (PRESSMAN, 1995: 916).

Em um primeiro momento, gerência de configuração pode ser confundida com manutenção de *software*, porém, uma diferença básica existe entre essas duas atividades: manutenção de *software* só acontece após a sua entrega, quando já está em produção. Por outro lado, a gerência de configuração é uma atividade que inicia concomitantemente com o início do projeto do *software* e perdura até que seja tirado de operação.

## 4.2 CONCEITO DE GERÊNCIA DE CONFIGURAÇÃO E MUDANÇA

Todos os elementos do *software* estão sujeitos a alterações, portanto é importante que estejam sob controle de alterações. Cada elemento do *software* sujeito a esse controle de configuração é denominado **item de configuração de *software***. O conjunto de itens de configuração do *software* compõe uma **configuração do *software***.

> Para que cada item de configuração possa ser efetivamente gerenciado, é necessário o estabelecimento de pontos bem definidos dentro do processo de desenvolvimento de software: as linhas de referências (*baselines*). Esses pontos podem ocorrer ao final de cada uma das fases do processo de desenvolvimento de software ou de algum outro modo definido pela gerência. Nos pontos estabelecidos pelas linhas de referência, os itens de configuração devem ser identificados, analisados, corrigidos, aprovados e armazenados em um local sob controle de acesso denominado repositório de itens de configuração. Esses itens só poderão ser alterados após uma solicitação de alteração formalmente aprovada pelo gerente de configuração. Essa é uma forma de controlar a situação de cada um dos itens de configuração, evitando assim inconsistências (ROCHA et al, 2001:60).

Cada vez que se deseja trabalhar com um item de configuração é necessária uma operação de *check-out*, ou seja, a transferência do item de configuração para a área de trabalho do desenvolvedor. Quando é concluída a alteração nesse elemento, realiza-se uma operação de *ckeck-in*, ou seja, o armazenamento do objeto alterado para o servidor de configuração.

## 4.3 A GERÊNCIA DE CONFIGURAÇÃO E MUDANÇA SOB A PERSPECTIVA DO PROCESSO UNIFICADO

Com o objetivo de manter a integridade do sistema, essa disciplina se propõe a controlar as mudanças feitas nos artefatos do projeto. Seu fluxo de atividades está apresentado na figura 4-1:

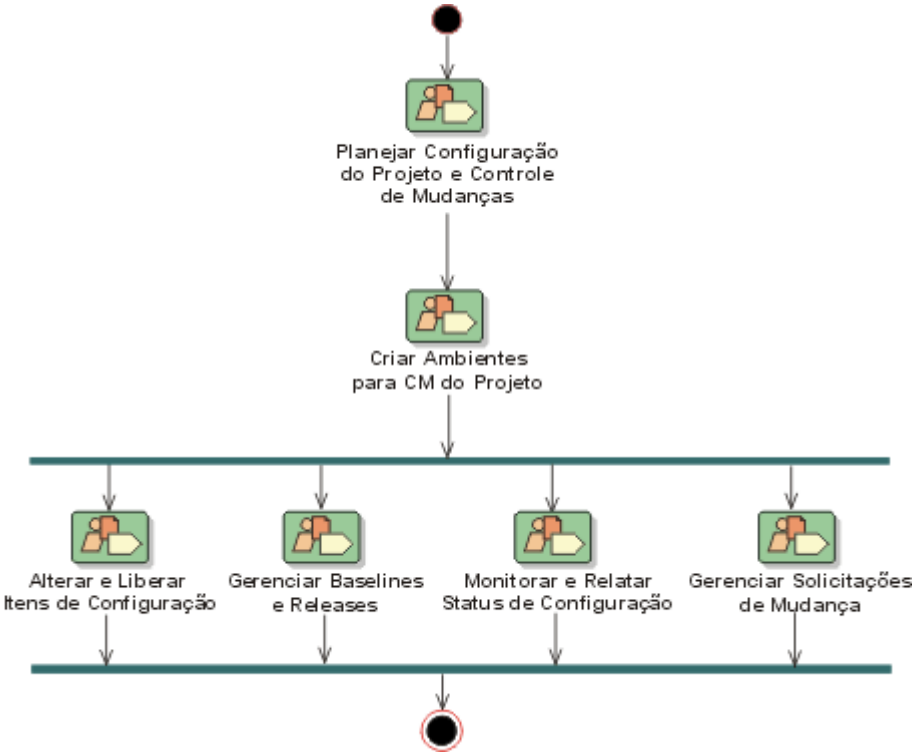



Figura 4-1 Fluxo de Trabalho da Disciplina de Gerência de Configuração e Mudança (RUP, 2004).

Com a finalidade de controlar todos os artefatos produzidos pela equipe de desenvolvimento de sistema, a importância desta disciplina é tão maior quanto maior for o tamanho do projeto e a complexidade do sistema desenvolvido. Mesmo que haja apenas um desenvolvedor envolvido no projeto, esta disciplina não deve ser desprezada ou deixada em segundo plano. Sistemas são dinâmicos e suscetíveis a mudanças, que por sua vez impactam ora positiva ora negativamente no projeto, dependendo do rigor em seus controles. Em um sistema composto por uma equipe muito grande, heterogênea e geograficamente distribuída, o impacto das mudanças tende a ser muito mais crítico e de difícil controle. Além das características intrínsecas do sistema, deve-se ainda considerar o dinamismo conseqüente do processo iterativo e incremental que incita ainda mais a existência de mudanças no projeto. A disciplina de gerência de configuração e mudança pode ser representada na figura 4-2, proposta pelo modelo CMM do SEI.

A face da Gerência de Configuração está relacionada à estrutura do produto e lida diretamente com a edição dos artefatos, suas versões e interdependências. Lida, ainda, com a edição de ambientes de trabalhos dos desenvolvedores a fim de evitar inconsistência nos trabalhos. À medida que um artefato evolui, várias versões passam

a existir e é necessário que se identifiquem os artefatos, suas versões e seu histórico de mudança. Como os artefatos não são produtos isolados dentro do ambiente do projeto, a mudança em um artefato pode implicar em mudança em outros. Entretanto, de modo geral, os membros da equipe costumam ter uma visão do seu mini-mundo com os artefatos por eles produzidos ou utilizados. É, portanto, de extrema importância a existência de alguém na equipe preocupado em assegurar que todos os componentes foram devidamente criados ou modificados e testados.

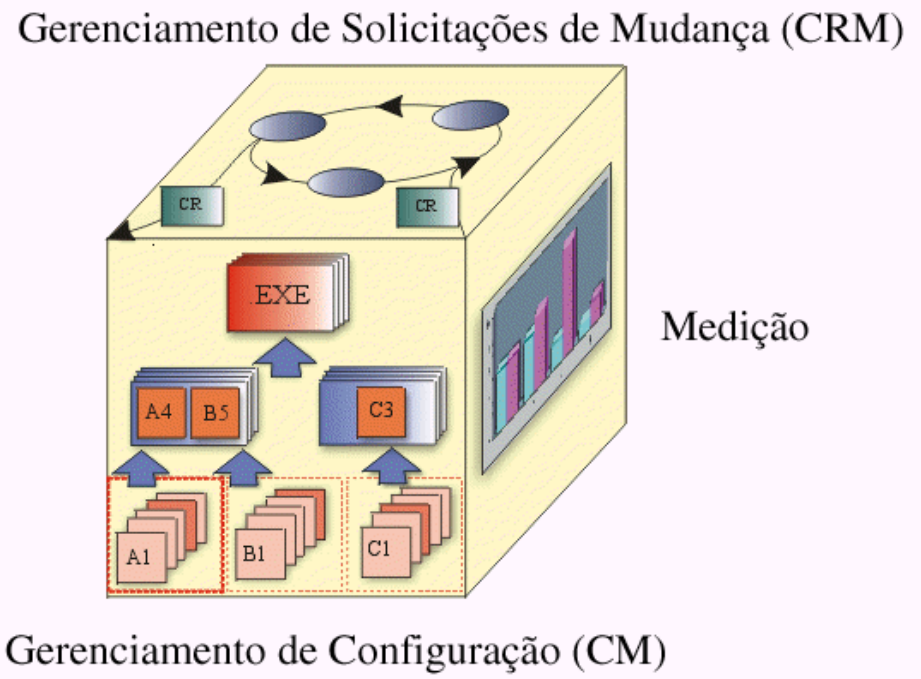



Figura 4-2 Cubo das Atividades da disciplina de Gerência de Configuração e Mudança (RUP, 2004).

A segunda face, de Gerência de Solicitações e Mudanças, está relacionada com a estrutura do processo, ou seja, a captura e manutenção de solicitações de mudança geradas interna ou externamente ao ambiente de desenvolvimento. Neste âmbito, é analisado o impacto potencial da mudança e o caminho a se tomar até que esta esteja completa. Esta atividade deve levar em conta o resultado da análise de impacto, assim como as prioridades relativas para determinar qual mudança será implementada, quando e em qual *release*.

Um requisito de mudança é um documento de proposta de mudança em um ou mais artefatos e pode ter várias razões de existir: reparar um defeito, aumentar a qualidade do produto, acrescentar um requisito ou documentar a mudança de uma

iteração para a próxima. Um requisito de mudança pode ser representado por uma máquina de estado que possui estados como novo, registrado, aprovado, especificado e completo. As informações sobre as mudanças, tais como a razão para a mudança, a motivação, os artefatos afetados e o impacto do projeto, da arquitetura, do custo e dos prazos, também devem ser mantidas, mesmo quando a solicitação não é implementada, incluindo, neste caso, a informação sobre o motivo da não implementação.

A terceira e ultima face é responsável pelo controle de status e medidas das mudanças que estão sendo realizadas. Faz parte do seu escopo avaliar o status, progresso, tendência e qualidade do produto. Cabe ainda controlar o que foi feito e o que ainda falta ser feito e, conseqüentemente, o custo da mudança. Nesta atividade, as diversas requisições são acumuladas em um banco de dados e representam o volume de trabalho a ser realizado. Deste banco de dados, podem-se extrair estatísticas de controle com base em cada um de seus atributos.

## 4.4 ARTEFATOS DE GERÊNCIA DE CONFIGURAÇÃO E MUDANÇAS

Segundo RUP (2004), a disciplina de Gerência de Configuração e Mudança produz alguns artefatos importantes para o controle do sistema, apresentados na tabela 4-1.

| Artefato | Definição |
|---|---|
| Plano de Gerenciamento de Configuração | Artefato de responsabilidade do gerente de configuração, elaborado nas atividades de Escrever Plano de Gerenciamento de Configuração, estabelecer processo de controle de mudança e estabelecer políticas de gerenciamento de configuração. Sua finalidade é descrever todas as atividades do Gerenciamento de Controle de Configuração e Mudança que serão executadas durante o ciclo de vida do produto ou do projeto. Ele detalha o cronograma de atividades, as responsabilidades atribuídas e os recursos necessários, como equipes, ferramentas e computadores. |
| Registro da Auditoria de | Produzido na atividade de Realizar Auditoria de Configuração, tem a finalidade de registrar se o desempenho do *software* |

| | |
|---|---|
| Configuração | desenvolvido está em conformidade com seus requisitos e se os artefatos necessários estão fisicamente presentes. A elaboração deste artefato fica sob a responsabilidade do gerente de configuração. |
| Repositório do Projeto | Fonte de todas as versões de diretórios e arquivos do projeto, armazena também todos os dados e metadados derivados que estejam associados a esses diretórios e arquivos. Embora seja de responsabilidade do gerente de configuração e mudança, sua base de dados é atualizada por, praticamente, todos os membros da equipe, já que seu conteúdo deve refletir o caráter dinâmico do sistema. |
| Solicitação de Mudança | De responsabilidade do gerente de controle de mudança, este artefato visa padronizar as mudanças nos artefatos de desenvolvimento. As Solicitações de Mudança são usadas para documentar e controlar defeitos, solicitações de melhorias e qualquer outro tipo de solicitação de mudança no produto. A vantagem das solicitações de mudança é que elas fornecem um registro das decisões e, devido a seu processo de avaliação, garantem que os impactos das mudanças sejam entendidos no projeto (RUP, 2004). |
| Espaços de Trabalho | Os Espaços de Trabalho permitem acesso a artefatos e recursos necessários para o desenvolvimento e a montagem dos produtos liberados. Há dois tipos de espaços de trabalho:<br>O espaço de trabalho de desenvolvimento é uma área de desenvolvimento privada, em que cada membro da equipe pode efetuar mudanças em artefatos sem que elas se tornem imediatamente visíveis para as outras pessoas.<br>O espaço de trabalho de integração é compartilhado e pode ser acessado por todos os membros da equipe do projeto. O produto como um todo é criado no espaço de trabalho de integração e também o utiliza como sua *baseline* (RUP, 2004) |

Tabela 4-1 Artefatos de Gerência de Configuração e Mudança

## 4.5 FERRAMENTAS PARA GERÊNCIA DE CONFIGURAÇÃO

Existe, atualmente, no mercado, uma grande variedade de ferramentas para gerência de configuração, principalmente, no que diz respeito a controle de versão de softwares. Muitas delas, inclusive, integradas às IDEs[12] das linguagens. Não é objetivo do presente trabalho a avaliação dessas ferramentas sob a ótica da construção de códigos, a avaliação é feita sob o ponto de vista de gerência de requisitos e como essas ferramentas podem agregar qualidade a essa atividade no processo de desenvolvimento.

Uma forma simples de classificar as ferramentas de gerência de configuração foi proposta por (DART, 1996) levando-se em consideração suas características funcionais. A figura 4-3 ilustra essa classificação. Como se pode observar, na verdade, não se trata exatamente de uma classificação, mas de uma evolução funcional das ferramentas.


Figura 4-3 Tipos de Ferramentas de GCS (OLIVEIRA, 2001).

Ferramentas de controle de versões: Nessa categoria estão as ferramentas que implementam controle de versões dos arquivos que são criados durante o desenvolvimento. Os objetivos dessas ferramentas são automatizar o rastreamento de arquivos, prevenir conflitos entre desenvolvedores, recuperar versões prévias, permitir o desenvolvimento paralelo, auditar desenvolvimento sobre quem, quando e o quê, estabelecer agregações de arquivos: configurações-base e distribuições e reduzir espaço de armazenamento.

Ferramentas orientadas ao desenvolvedor: Nessa categoria encontram-se ferramentas que, além de suportarem controle de versões, oferecem suporte ao

[12] Acrônimo para *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado

trabalho em equipe, facilitando principalmente desenvolvimento concorrente, isto é, desenvolvedores trabalhando ao mesmo tempo sobre o mesmo conjunto de arquivos. Têm como característica principal integrarem-se no ambiente de desenvolvimento.

Ferramentas orientadas a processo: Ferramentas que suportam controle de versões e parte das funcionalidades das ferramentas orientadas ao desenvolvedor, mas cujo ponto forte é a automatização do gerenciamento do ciclo de vida dos objetos envolvidos no desenvolvimento. De maneira geral, as ferramentas desta categoria também fornecem uma abordagem integrada à gerência de mudanças e ao rastreamento de correção de defeitos.

Existem diversas ferramentas no mercado que suportam a atividade de gerência de configuração, dentre elas, podem-se citar as ferramentas *open source* Subversion[13], CVS[14], CVSNT[15], Aegis[16], Arch[17] e as proprietárias ClearCase[18], StarTeam[19], Perforce[20] e BitKeeper[21].

Desse grupo de ferramentas, esperam-se algumas características básicas a fim de suprir necessidades específicas, conforme apresentado na tabela 4-2.

| Característica | Descrição |
|---|---|
| Suporte a configurações (DIAS, 2006). | Identificação de distintas versões de arquivos que compõem uma distribuição e recuperação da mesma. Devem ser analisados os aspectos de rastreabilidade que tornam possível o estabelecimento de relações entre distintos itens de configuração e seus históricos de modificações. |
| Controle de | Atribuir permissões individuais para os membros da |

[13] Subversion versão 1.2.3: Tigris.org, 2005. Conjunto de programas (TIGRIS, 2004).
[14] CVS-Concurrent Versions System versão 1.12.13: cvshome.org. Conjunto de programas. (CVSHOME, 2005)
[15] CVSNT versão 2.5.02.2115: march-hare.com/cvsnt. Conjunto de programas (CVSNT, 2005).
[16] Aegis versão 4.20: sourceforge.net. Conjunto de programas (SOURCEFORGE, 2005).
[17] GNU Arch versão 2.0: gnuarch.org. Conjunto de programas (ARCH, 2005).
[18] Rational ClearCase versão 2003.06.15: Rational Software Corporation, 2003. Conjunto de progamas (RATIONAL, 2004).
[19] BORLAND Staream versão 5.4: Borland Software Corporation, 2005. Conjunto de programas (BORLAND, 2005).
[20] Perforce versão 2005.1: Perforce Software Inc., 2005. Conjunto de programas (PERFORCE, 2005).
[21] BitKeeper versão bk-3.2.6: BitMover Inc., 2005. Conjunto de programas (BITKEEPER, 2005)

| | |
|---|---|
| permissões (MILLIGAN, 2003). | equipe de desenvolvimento aos diversos itens de configuração. Os níveis de permissão podem ser de gravação, leitura ou sem acesso. |
| Controle de arquivos em diversos formatos (DAVIS, 1996). | As ferramentas não podem se restringir a controle de arquivos em formato ASCII, mas devem prover controle também em formatos Unicode e binário |
| Suporte a desenvolvimento cooperativo (LOPES, 2003). | Deve suportar projetos com diversos *sites* de acesso remoto, realidade que se faz presente no trabalho de equipes que se distribuem em diferentes áreas geográficas, em que cada membro realiza uma atividade, e a integração do trabalho do grupo é feita continuamente. |
| Controle das mudanças (DIAS, 2006). | Prover informações sobre quais itens estão em alteração por algum membro da equipe. Para cada alteração deve ser permitido associar informações sobre data, hora e responsável pela alteração, bem como permitir que se descreva textualmente um resumo sobre a alteração realizada. |
| Portabilidade (DIAS, 2006). | Deve suportar a integração de itens de configuração oriundos de clientes com ambientes operacionais distintos |

Tabela 4-2: Características esperadas das ferramentas de Gerência de Configuração

# 5 ESTUDO DE CASO

A avaliação das ferramentas foi feita tendo como base o desenvolvimento de um sistema de controle de vendas em check-out de lojas de conveniências, farmácias, padarias, mini-market, etc, denominado Sistema Paragon. O objetivo geral do sistema é o controle das vendas com emissão de cupom fiscal (ECF) e transferência eletrônica de fundos (TEF). A alta concorrência no mercado e a heterogeneidade do público usuário são as principais causas da instabilidade nos requisitos e por isso tornam o estudo de caso adequado para o presente trabalho.

Este capítulo apresenta os artefatos gerados na primeira iteração do projeto do Sistema Paragon e estão aqui apresentados com o objetivo de elucidar as características do estudo de caso que serve de objeto de controle pelas ferramentas estudadas e apresentadas no capítulo 6. A escolha desse sistema foi baseada no fato de ser um sistema sujeito a constantes alterações em seus requisitos oriundas da grande concorrência do mercado, necessidade de adaptação às especificidades dos diferentes ramos de comércio e ao elevado número de alterações na legislação que rege o setor.

## 5.1 DOCUMENTO DE VISÃO

### 5.1.1 Introdução

O presente documento tem como objetivo apresentar uma visão geral do sistema **Paragon**, apresentando os problemas que se propõe a resolver, as partes envolvidas e seus requisitos básicos.

## 5.1.2 Circunstâncias

### 5.1.2.1 Apresentação do Problema

| | |
|---|---|
| O problema de | Atender as exigências da legislação atual sobre emissão de cupom fiscal através de equipamentos emissores de cupom fiscal (ECF) e de Transferência Eletrônica de Fundos (TEF) entre estabelecimentos comerciais e administradoras de cartões de crédito em contrapartida à dificuldade técnica de utilização de sistemas informatizados nas empresas do setor. |
| Afeta | Os proprietários, administradores, funcionários em geral e clientes de estabelecimentos comerciais varejistas de qualquer porte. |
| Cujo impacto é | A necessidade imediata de utilização de um sistema homologado pela receita a fim de evitar sanções legais. |
| Uma solução seria: | O desenvolvimento de um sistema que seja ao mesmo tempo simples de instalar e usar, flexível para as diferentes realidades das empresas e que atenda à legislação vigente. |

Tabela 5-1 Apresentação do Problema

## 5.1.3 Descrição das Partes Envolvidas

### 5.1.3.1 Resumo das Partes Envolvidas

| Nome | Responsabilidades |
|---|---|
| Gerente de vendas | Responsável por gerenciar as atividades dos vendedores e caixas das lojas. Normalmente é quem decide sobre a autorização de vendas fora dos padrões normais. |
| Vendedor | Responsável por atender o cliente, registrar as vendas e negociar a forma de pagamento. |
| Crediarista | Responsável por cadastrar clientes novos e manter atualizado o cadastro de clientes antigos. Avalia a ficha do cliente para liberação de crédito. |

| Nome | Responsabilidades |
|---|---|
| Caixa | Responsável pelo recebimento das vendas e controle do caixa. Emite o cupom fiscal e concretiza as transações financeiras com as administradoras de cartão de crédito. |
| Estoquista | Responsável pela reposição dos produtos no estoque e atualização dos preços dos produtos. |
| Comprador | Responsável pela administração dos pedidos e compras dos produtos. |
| Financista | Responsável pelo controle financeiro da loja: recebimento de duplicatas e cartões de créditos. Pagamentos diversos e controle de caixa e banco. |

Tabela 5-2 Resumo das Partes Envolvidas.

#### 5.1.3.2 Ambiente do Usuário

Considerando o propósito do sistema de atender diversos tipos de estabelecimentos comerciais, o ambiente do usuário, volumes e freqüência das transações tendem a ser bastante variados, o que leva a uma faixa de valores bem ampla, conforme descrito abaixo:

Número de pessoas envolvidas: desde uma única pessoa trabalhando em um equipamento *desktop* até ambientes com 20 (vinte) ou mais pessoas trabalhando simultaneamente.

O ciclo completo de uma tarefa de atendimento de uma venda pode demorar algumas horas, mas o processo de fechamento da venda não pode levar mais do que alguns poucos minutos, ou segundo. As atividades devem ser descomplicadas e rápidas.

O volume de vendas diárias depende do estabelecimento comercial, podendo chegar a centenas de vendas realizadas diariamente.

O ambiente, embora diferencie de porte, tende a seguir um mesmo padrão, funcionando em uma rede local de pequeno porte. Em alguns casos, o escritório central fica distante e depende de acesso remoto.

O ideal é que o sistema não esteja preso a nenhuma plataforma, a fim de atender o maior número possível de usuários. Deve funcionar em ambiente Windows

e é desejável que funcione também em plataforma Linux.

As administradoras de cartões de crédito e os fabricantes de impressoras fiscais disponibilizam bibliotecas de acesso a seus produtos. O sistema deve acessar as funcionalidades dessas bibliotecas.

### 5.1.4 Visão Geral do Produto

#### 5.1.4.1 Perspectiva do Produto

O sistema deve registrar todas as vendas realizadas no estabelecimento.

Deve emitir o cupom fiscal durante a venda.

Deve dar baixa no estoque à medida que é vendido um produto e somente nessa condição.

Deve permitir cancelamento do ultimo cupom fiscal emitido.

Deve permitir o cancelamento de um produto qualquer de uma venda enquanto esta estiver sendo realizada.

Deve comunicar diretamente com administradoras de cartões de crédito, via TEF (Transferência Eletrônica de Fundos).

Deve permitir emissão de cupons não fiscais vinculados à venda para as vendas com cartão de crédito.

Deve permitir o cadastro de clientes e associar, a cada venda, pelo menos um cliente.

Deve permitir o cadastro de vendedores e permitir associar até dois vendedores a cada venda.

Deve permitir o cadastro de formas de pagamentos e associar quantas forem necessárias a cada venda.

Deve ser interligado ao sistema de controle de estoque para dar baixa automática nos produtos vendidos.

Deve ser interligado ao sistema de controle financeiro para lançamento de contas a receber.

Deve permitir operações de sangria e suprimento de caixa registradas na impressora fiscal.

Deve gerar informações para a fiscalização no formato exigido (Sintegra).

#### 5.1.4.2 Suposições e Dependências

Os fatores externos abaixo relacionados afetam diretamente o sistema e devem, portanto, ser observados:

O sistema deve funcionar independente da existência de um sistema de controle financeiro integrado a ele.

É necessário, para seu funcionamento, a existência de um sistema básico de controle de estoque, do qual o sistema **Paragon** buscará os dados dos produtos vendidos.

Apesar dos esforços dos fabricantes de impressoras fiscais em padronizar a interface dos seus produtos, o Sistema **Paragon** deve levar em conta a realidade atual, onde cada fabricante possui uma interface proprietária, e estar preparado para as principais impressoras do mercado.

### 5.1.5 Outros Requisitos

O Sistema **Paragon** funcionará em equipamentos *desktop* ou em ambientes cliente-servidor. Será usado um Sistema Gerenciador de Banco de Dados gratuito, evitando custo adicional com licença de uso.

Por exigência da legislação, o sistema deve ser totalmente tolerante a falhas, tais como:

Caso uma operação de venda seja interrompida no meio, o sistema deverá cancelá-la inteiramente ou retornar exatamente de onde parou;

Não poderá ser feita uma venda sem que a impressora fiscal esteja conectada e ligada;

O sistema deve bloquear a troca de impressora durante uma venda;

Devido às suas características comerciais, o sistema deverá ser de fácil instalação, possuir material de apoio robusto e de fácil entendimento;

Mensagens de erros e advertências devem ser claras e precisas para que o usuário leigo consiga entende-las.

O Sistema **Paragon** deverá ser distribuído comercialmente através de pacotes.

A operação de venda deve ter um tempo de resposta baixo, uma vez que afeta diretamente a satisfação dos clientes das empresas usuárias.

## 5.2 GLOSSÁRIO

### 5.2.1 Introdução

Este documento é usado para definir a terminologia específica do domínio do problema, explicando termos que podem ser desconhecidos do leitor das descrições de caso de uso ou de outros documentos do projeto. Também estão relacionados os acrônimos presentes no projeto

#### 5.2.1.1 Finalidade

Freqüentemente, este documento poderá ser usado como um dicionário de dados informal, capturando definições de dados para que as descrições de caso de uso e outros documentos do projeto possam concentrar-se no que o sistema deve fazer com as informações.

#### 5.2.1.2 Escopo

O presente glossário engloba termos usados no projeto do Sistema **Paragon.**

### 5.2.2 Definições

Os termos definidos aqui são a essência do documento e estão descritos em ordem alfabética.

Dados de Cliente Pessoa Física: Dados dos clientes pessoa física cadastrados no sistema. Podem ser agrupados da seguinte forma:

Identificação: nome, identidade, carteira de trabalho, cpf, data de nascimento, sexo e estado civil.

Endereço: Endereço, bairro, cidade, estado, cep, e-mail e home page.

Telefone: residencial, comercial, ramal e celular.

Profissional: profissão, renda, local de trabalho, data de início.

Crediário: data do cadastro, data da ultima consulta ao SPC ou SERASA, limite de crédito, identificador se pertence a lista negra.

Referências: filiação, nome do cônjuge.

Somente o nome é obrigatório. Os demais dados são opcionais.

Dados de Cliente Pessoa Jurídica: Dados dos clientes pessoa jurídica cadastrados no sistema. Podem ser agrupados da seguinte forma:

Identificação: Razão Social, Nome fantasia, CNPJ, Inscrição Estadual.

Endereço: Endereço, bairro, cidade, estado, cep, e-mail e home page.

Contatos: Nome, telefone, ramal, celular e e-mail.

Somente a razão social é obrigatória. Os demais dados são opcionais.

Dados do Vendedor: Nome, setor, percentual de comissão, data de admissão, data de demissão.

ECF: Emissor de Cupom Fiscal. Equipamento ligado à porta serial do computador pelo qual deve ser impresso qualquer comprovante de venda. É lacrado pelo FISCO e sua utilização é obrigatória em alguns tipos de estabelecimentos comerciais.

Leitura X: Relatório emitido pelas ECFs com o resumo das vendas no dia.

Redução Z: Relatório similar à leitura X, porém, após a sua emissão, o caixa fica fechado até o próximo dia. Nenhum cupom fiscal é emitido no dia após sua emissão.

Sangria: Retirada de numerários do caixa.

SINTEGRA: Sistema Integrado de informações sobre Operações Interestaduais com Mercadorias e Serviços. Sistema nacional que visa o controle informatizado das operações de entradas e saídas interestaduais realizadas pelos contribuintes do ICMS.

Suprimento: Abastecimento de numerários no caixa.

TEF: Transferência Eletrônica de Fundos. Operação de pagamento realizado através de administradoras de cartão de crédito, débito ou de vale refeição.

## 5.3 DIAGRAMA DE CASO DE USO

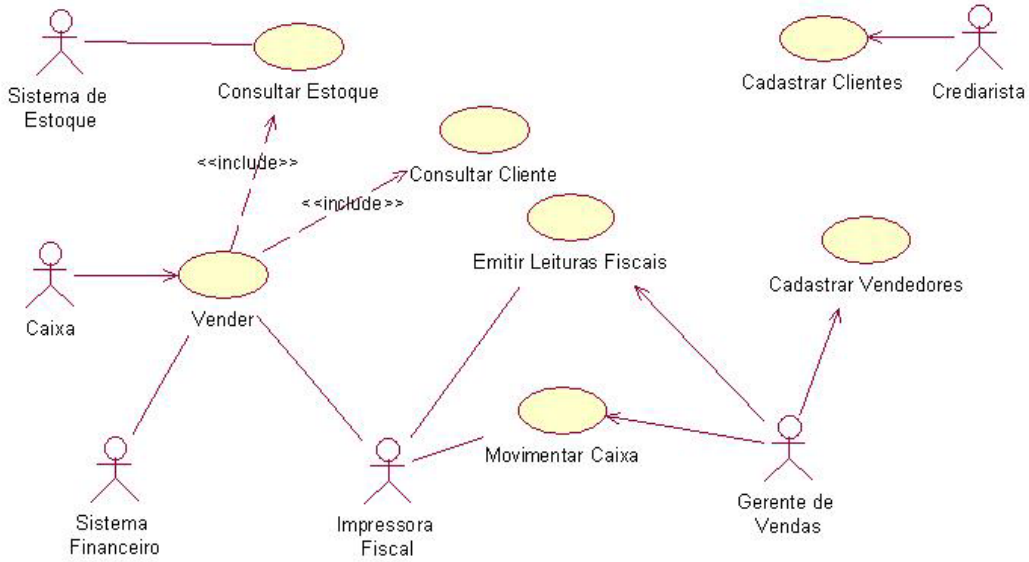



Figura 5-1 Diagrama de Caso de Uso.

## 5.4 BREVE DESCRIÇÃO DOS CASOS DE USO

Cadastrar Clientes: Cadastrar os clientes pessoa física ou jurídica. Nem todas as lojas fazem controle dos clientes que compram, portanto não utilizam esta opção.

Cadastrar Vendedores: Cadastrar os dados básicos dos vendedores. Nem todas as lojas fazem controle dos vendedores responsáveis pelas vendas, portanto não utilizam esta opção.

Consultar Clientes: Permite a consulta aos dados cadastrais dos clientes pessoa física ou jurídica.

Consultar estoque: Permite a consulta ao estoque de produtos.

Vender: Este caso de uso descreve os passos necessários para concluir uma venda no sistema.

Emitir Leituras Fiscais: Emite a leitura X, redução Z e leitura da memória fiscal da impressora fiscal.

Movimentar Caixa: Realiza operações de sangria e suprimento de caixa.

## 5.5 ESPECIFICAÇÕES SUPLEMENTARES

**Introdução**

As Especificações Suplementares capturam os requisitos de sistema que não são capturados imediatamente nos casos de uso do modelo de casos de uso. Entre os requisitos estão incluídos os requisitos legais e de regulamentação e padrões de aplicativo; atributos de qualidade do sistema a ser criado, incluindo requisitos de usabilidade, confiabilidade, desempenho e suportabilidade e outros requisitos, como sistemas operacionais e ambientes, requisitos de compatibilidade e restrições de design.

**Finalidade**

O presente documento tem como finalidade suprir de informações os seguintes papeis:

1. Analistas de Sistemas: criam e mantêm as Especificações Suplementares, usadas como meio de comunicação entre eles, o cliente e os desenvolvedores;
2. Desenvolvedores: usam as Especificações Suplementares como referência quando definem responsabilidades, operações e atributos nas classes e quando ajustam as classes ao ambiente de implementação;
3. Implementadores: usam as Especificações Suplementares para entrada durante a implementação de classes;
4. Gerentes de Projeto: usam as Especificações Suplementares para entrada durante o planejamento de iterações;
**5.** Testadores: usam as Especificações Suplementares para verificar a compatibilidade do sistema.

**Funcionalidade**

Os requisitos funcionais são capturados através das especificações dos casos de uso, entretanto, alguns requisitos comuns a diversos casos de uso são descritos nessa seção a fim de evitar duplicidade.

**Geração automática de Códigos:** Todas as funcionalidades do sistema que tratam de cadastramento de alguma informação deverão gerar códigos seqüenciais automáticos.

**Opções de Consulta:** Todas as funcionalidades que necessitam que o ator

informe o código de algum dado (vendedor, forma de pagamento, produto, etc.) deve permitir que o ator possa consultar esses códigos com base em algum outro atributo significativo (nome, descrição, etc.)

**Usabilidade:** O sistema deverá exigir pouco conhecimento de informática do usuário, principalmente dos caixas. As mensagens de erros e advertências devem ser claras e auto-explicativas. Com isso, o tempo de treinamento deve ser breve e a autonomia operacional, alta. As interfaces devem ser “limpas” e apresentar apenas informações condizentes com o contexto. É desejável que seja seguido o padrão GUI da *Microsoft*.

**Disponibilidade:** O sistema deve estar disponível durante todo o tempo de funcionamento da loja. Deve ser prevista a sua utilização em estabelecimentos que funcionam 24 horas por dia, 7 dias por semana.

**Tolerância a Falhas:** O sistema deve suportar situações de falhas de operação, tais como: Aviso de fim da bobina de papel da impressora fiscal; Não permitir troca da impressora fiscal durante a operação de vendas; Não operar sem uma impressora fiscal conectada; Recuperação ou cancelamento de uma venda interrompida.

**Volume:** Considerando o propósito do sistema de atender diversos tipos de estabelecimentos comerciais, o ambiente do usuário, volumes e freqüência das transações tendem a ser bastante variados, o que leva a uma faixa de valores bem ampla, conforme descrito abaixo:

Número de pessoas envolvidas: desde uma única pessoa trabalhando em um equipamento *desktop* até ambientes com 20 (vinte) ou mais pessoas trabalhando simultaneamente.

O volume de vendas diárias depende do estabelecimento comercial, podendo chegar a centenas de vendas realizadas diariamente.

**Reuso:** Funcionalidades relacionadas à impressora fiscal serão reusadas em outros sistemas.

**Utilitários de Manutenção:** O sistema deverá possuir funcionalidades básicas de cópia de segurança e restauração dos dados, criação de usuários, senhas e atribuição de permissões de acesso.

**Restrições de Design**

**Comunicação com Impressora Fiscal e Administradoras de Cartões de Créditos:** O Sistema deverá ser desenvolvido considerando as especificações de interface com as impressoras fiscais dos principais fabricantes do mercado e com todas as administradoras de cartões de crédito do país.

**Design Orientado a Objetos:** Devido à necessidade de reuso, o sistema deverá ser totalmente orientado a objeto.

**Banco de Dados:** O sistema deverá permitir a comunicação com Sistemas Gerenciadores de Banco de Dados (SGBD) de diferentes fabricantes, não podendo, portanto, estar restrito a um banco de dados único.

**Linguagem de Programação:** Devido à necessidade de funcionamento em diferentes plataformas, o sistema deverá ser desenvolvido em linguagem Java.

**Requisitos de Sistema de Ajuda e de Documentação de Usuário On-line:** O sistema deverá ser acompanhado de um manual de instalação, manual de usuário e possuir *help* de contexto.

**Componentes Externos**

Os componentes responsáveis pela comunicação com as administradoras de cartão de crédito e impressoras fiscais são disponibilizados gratuitamente pelos fabricantes.

# 6 AVALIAÇÃO COMPARATIVA

No presente capítulo, serão apresentadas as ferramentas avaliadas e o resultado da avaliação realizada individualmente e integrando o controle de requisitos ao controle de configuração.

## 6.1 FERRAMENTAS DE CONTROLE DE REQUISITOS

### 6.1.1 Rational RequisitePro

Ferramenta desenvolvida pela Rational Corporation, atualmente integrada à IBM, o Rational RequisitePro trabalha integrado com o editor de texto Microsoft Word. Com isso, é possível descrever os requisitos no editor de textos sem que haja uma ruptura entre os requisitos controlados e os documentos de levantamento de dados e especificação do sistema.

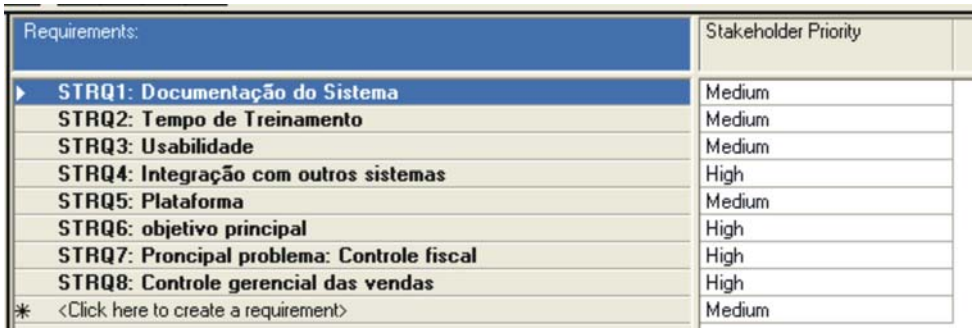
Ao iniciar, o *software* permite a criação de novos projetos a partir de *templates* predefinidos e a criação de novos *templates*, conforme a necessidade da equipe de desenvolvimento. Os dados podem ser armazenados em bancos de dados MS-Access, Oracle ou SQL-Server. Para o presente trabalho, foi utilizado o banco de dados MS-Access com o Use-case Template. Com a configuração utilizada, o RequisitePro apresenta quatro tipos básicos de requisitos:

FEAT: Representam as características necessárias ou desejáveis do sistema. Podem ser cadastrados diretamente no banco de dados ou extraídos do documento de visão (Anexo 04). A figura 6-1 mostra os requisitos extraídos do documento de visão do sistema com seus respectivos atributos:

| Requirements: | Priority | Status | Difficulty | Stability | Risk | Planned Ite |
|---|---|---|---|---|---|---|
| FEAT1: Registrar as Vendas | High | Proposed | Low | High | Schedule - Low | 4 |
| FEAT2: Emitir Cupom | High | Proposed | Medium | High | Schedule - Low, | 1 |
| FEAT3: Baixa do Estoque | Medium | Proposed | Low | High | Schedule - Low | 4 |
| FEAT4: Cancelamento de Cupom | Medium | Proposed | Low | High | Technology - Lo | 3 |
| FEAT5: Cancelamento de Produto no Cupom | Medium | Proposed | High | High | Technology - Me | 2 |
| FEAT6: Transferência Eletrônica de Fundos | Medium | Proposed | High | Medium | Technology - Hi | 1 |
| FEAT7: Emitir Cupom não fiscal vinculado | Medium | Proposed | Medium | Medium | Technology - Hi | 1 |
| FEAT8: Cadastrar Clientes | Medium | Proposed | Medium | High | Schedule - Medi | 3 |
| FEAT9: Cadastrar Vendedores | Medium | Proposed | Medium | High | Schedule - Low, | 4 |
| FEAT10: Cadastrar Formas de Pagamento | Medium | Proposed | Medium | High | Schedule - Low, | 4 |
| FEAT11: Interligar ao Sistema de Estoque | Medium | Proposed | Medium | Medium | Schedule - Low | 4 |
| FEAT12: Interligar ao Sistema Financeiro | Medium | Proposed | High | Medium | Technology - Me | 2 |
| FEAT13: Movimentação de Caixa | Medium | Proposed | Low | Medium | Technology - Lo | 3 |
| FEAT14: Gerar Sintegra | Medium | Proposed | High | Medium | Technology - Hi | 1 |
| FEAT15: Independencia de Controle Financeiro | Medium | Proposed | Medium | High | Technology - Lo | 3 |
| FEAT16: Atender aos diversos padrões de... | Medium | Proposed | High | Medium | Technology - Hi | 1 |
| FEAT17: Integridade das vendas interrompidas | Medium | Proposed | High | High | Technology - Hi | 1 |
| FEAT18: Venda com impressora desligada | Medium | Proposed | Low | High | Technology - Me | 2 |
| FEAT19: Troca de Impressora durante a venda | Medium | Proposed | High | High | Technology - Me | 2 |
| FEAT20: Facilidade de Instalação | Medium | Proposed | High | Medium | Technology - Me | 2 |
| FEAT21: Clareza das mensagens de erro e... | Medium | Proposed | Medium | Medium | Technology - Lo | 3 |
| FEAT22: Empacotamento do produto | Medium | Proposed | High | Medium | Technology - Me | 2 |
| FEAT23: Tempo de resposta | Medium | Proposed | Medium | Medium | Technology - Me | 2 |
| <Click here to create a requirement> | Medium | Proposed | Medium | Medium | | |

Figura 6-1 Requisitos extraídos do documento de visão para o RequisitePro.

STRQ: Representam as requisições dos principais envolvidos do sistema. Podem ser cadastrados diretamente no banco de dados ou extraídos do documento Solicitação das Partes Envolvidas (Anexo 02). A figura 6-2 mostra os requisitos extraídos do documento Solicitações das Partes Envolvidas do sistema com seus respectivos atributos:



| Requirements: | Stakeholder Priority |
|---|---|
| STRQ1: Documentação do Sistema | Medium |
| STRQ2: Tempo de Treinamento | Medium |
| STRQ3: Usabilidade | Medium |
| STRQ4: Integração com outros sistemas | High |
| STRQ5: Plataforma | Medium |
| STRQ6: objetivo principal | High |
| STRQ7: Proncipal problema: Controle fiscal | High |
| STRQ8: Controle gerencial das vendas | High |
| <Click here to create a requirement> | Medium |

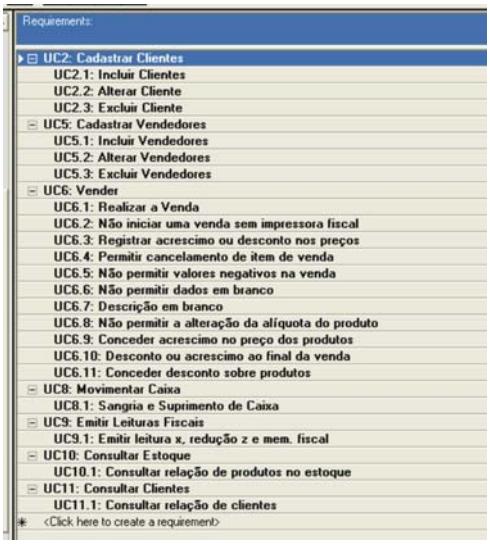
Figura 6-2 Requisitos extraídos do documento Solicitação das Partes Envolvidas.

SUPL: Representam as requisições suplementares do sistema. Podem ser cadastrados diretamente no banco de dados ou extraídos do documento Especificações Suplementares (Anexo 05). A figura 6-3 mostra os requisitos extraídos do documento Especificações Suplementares do sistema com seus respectivos atributos:

| Requirements: | Priority | Status | Difficulty | Stability | Risk |
|---|---|---|---|---|---|
| SUPL1: Geração automática de códigos | Medium | Proposed | Medium | High | |
| SUPL2: Opções de Consulta | High | Proposed | Medium | High | |
| SUPL3: Usabilidade | High | Proposed | Medium | High | |
| SUPL4: Disponibilidade | High | Proposed | High | High | |
| SUPL5: Tolerância a Falhas | High | Proposed | High | High | |
| SUPL6: Desempenho: Volume | Medium | Proposed | High | High | |
| SUPL7: Suportabilidade: Reuso | Low | Proposed | High | High | |
| SUPL8: Suportabilidade: Utilitários de Manutenção | Low | Proposed | Low | High | |
| SUPL9: Comunicação com Impressora Fiscal e... | High | Proposed | Low | High | |
| SUPL10: Design Orientado a Objetos | High | Proposed | Medium | High | |
| SUPL11: Banco de Dados | Medium | Proposed | Medium | High | |
| SUPL12: Linguagem de Programação | Medium | Proposed | Medium | High | |
| SUPL13: Documentação | Low | Proposed | Medium | High | |
| SUPL14: omponentes Externos | High | Proposed | High | High | |
| <Click here to create a requirement> | Medium | Proposed | Medium | Medium | |

Figura 6-3 Requisitos extraídos do documento Especificações Suplementares.

UC: Representam as especificações funcionais presentes nos casos de uso. Podem ser cadastradas diretamente no banco de dados ou extraídos dos documentos de especificações dos Casos de Uso (Anexo 07). A figura 6-4 mostra os requisitos extraídos dos documentos Especificações dos Casos de Uso:



| Requirements: |
|---|
| UC2: Cadastrar Clientes |
| UC2.1: Incluir Clientes |
| UC2.2: Alterar Cliente |
| UC2.3: Excluir Cliente |
| UC5: Cadastrar Vendedores |
| UC5.1: Incluir Vendedores |
| UC5.2: Alterar Vendedores |
| UC5.3: Excluir Vendedores |
| UC6: Vender |
| UC6.1: Realizar a Venda |
| UC6.2: Não iniciar uma venda sem impressora fiscal |
| UC6.3: Registrar acrescimo ou desconto nos preços |
| UC6.4: Permitir cancelamento de item de venda |
| UC6.5: Não permitir valores negativos na venda |
| UC6.6: Não permitir dados em branco |
| UC6.7: Descrição em branco |
| UC6.8: Não permitir a alteração da alíquota do produto |
| UC6.9: Conceder acrescimo no preço dos produtos |
| UC6.10: Desconto ou acrescimo ao final da venda |
| UC6.11: Conceder desconto sobre produtos |
| UC8: Movimentar Caixa |
| UC8.1: Sangria e Suprimento de Caixa |
| UC9: Emitir Leituras Fiscais |
| UC9.1: Emitir leitura x, redução z e mem. fiscal |
| UC10: Consultar Estoque |
| UC10.1: Consultar relação de produtos no estoque |
| UC11: Consultar Clientes |
| UC11.1: Consultar relação de clientes |
| <Click here to create a requirement> |

Figura 6-4 Requisitos extraídos dos documentos Especificações dos Casos de Uso.

A correlação entre os diversos requisitos levantados é feita através de planilhas, onde são estabelecidas as relações *Trace To* e *Trace From* de dependência entre os requisitos. É possível rastrear os casos de uso (UC), as especificações suplementares (SUPL) e as solicitações das partes envolvidas (STRQ) com as funcionalidades (FEAT). Na analise de impacto, é possível estabelecer o impacto causado nas características do sistema pelas requisições dos usuários, as especificações suplementares e os casos de uso impactados pelas mudanças nas

características do sistema. Na pasta de Análise de cobertura, o *software* permite o gerenciamentos das características não rastreadas por solicitações das partes envolvidas, requisitos suplementares e casos de uso não rastreados por características.

### 6.1.2 REM – Requirements Engineering Methodology

Ferramenta desenvolvida na Universidade de Sevilha como parte integrante da tese de doutorado *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información* defendida por Amador Duran Toro. A ferramenta trabalha com banco de dados MS-Access e gera toda a documentação em um arquivo XML[22] que, associado ao padrão XSL[23], geram páginas HTML[24] que podem ser consideradas como a documentação final dos requisitos (TORO, 2000:368).

Os três elementos chaves tratados pela ferramenta são as especificações dos requisitos dos clientes (denominados requisitos-C ou CRS), especificações dos requisitos dos desenvolvedores (denominados requisitos-D ou DRS) e especificação dos conflitos (CFS). De acordo com o autor, essas especificações não têm necessariamente que coincidir com os documentos físicos, são simplesmente um instrumento para a organização abstrata das informações.

Também são consideradas partes integrantes do projeto as informações correspondentes aos participantes e empresas às quais estão associados, podendo ser clientes ou desenvolvedores.

A figura 6-5 apresenta a tela principal da ferramenta com os dados do projeto Paragon. A árvore de pastas que se encontra à direita é definida pelo próprio usuário, conforme suas necessidades. A janela principal apresenta as especificações do sistema em formato HTML. A barra de ferramentas, em detalhe na figura 6-6, mostra os diferentes tipos de elementos que podem conter a especificação de um sistema na ferramenta, a saber: nova sessão/apêndice; novo parágrafo/item do glossário; novo arquivo gráfico; nova organização; novo *stakeholder;* nova reunião;

[22] Acrônimo para Extensible Markup Language
[23] Acrônimo para Extended Style Language
[24] Acrônimo para Hipertext Markup Language

novo objetivo; novo ator; novo requisito de informação; novo requisito de restrição; novo caso de uso; novo requisito funcional; novo requisito não funcional; nova matriz de rastreamento; novo tipo de objeto; novo tipo de valor; nova associação; nova operação de sistema; novo conflito; novo defeito; novo requisito de mudança.

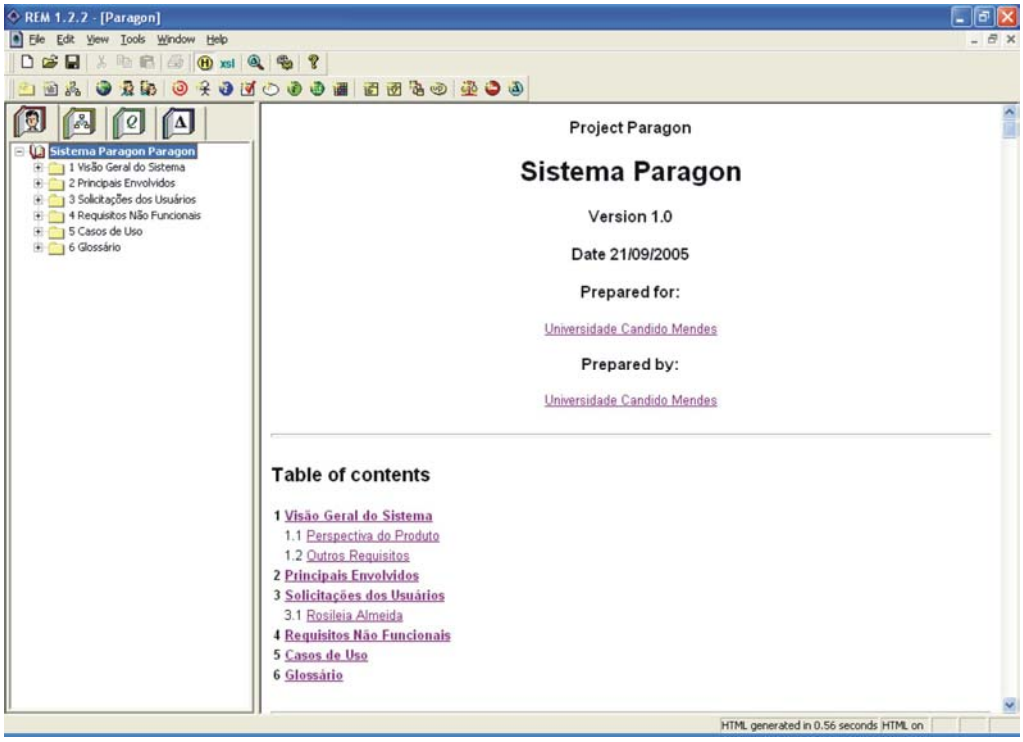



Figura 6-5 Tela principal da ferramenta REM.

Figura 6-6 Barra de Ferramentas da ferramenta REM.

Para cada elemento, é possível se estabelecer a sua rastreabilidade, utilizando-se o conceito de *Trace To* e *Trace From* apresentados na ferramenta RequisitePro. Essa informação é imprescindível para se avaliar o impacto de mudanças em um requisito, conforme pode ser observado na figura 6-7, através dessa ferramenta, pode-se avaliar claramente quais os objetos afetados pela mudança em algum outro objeto.

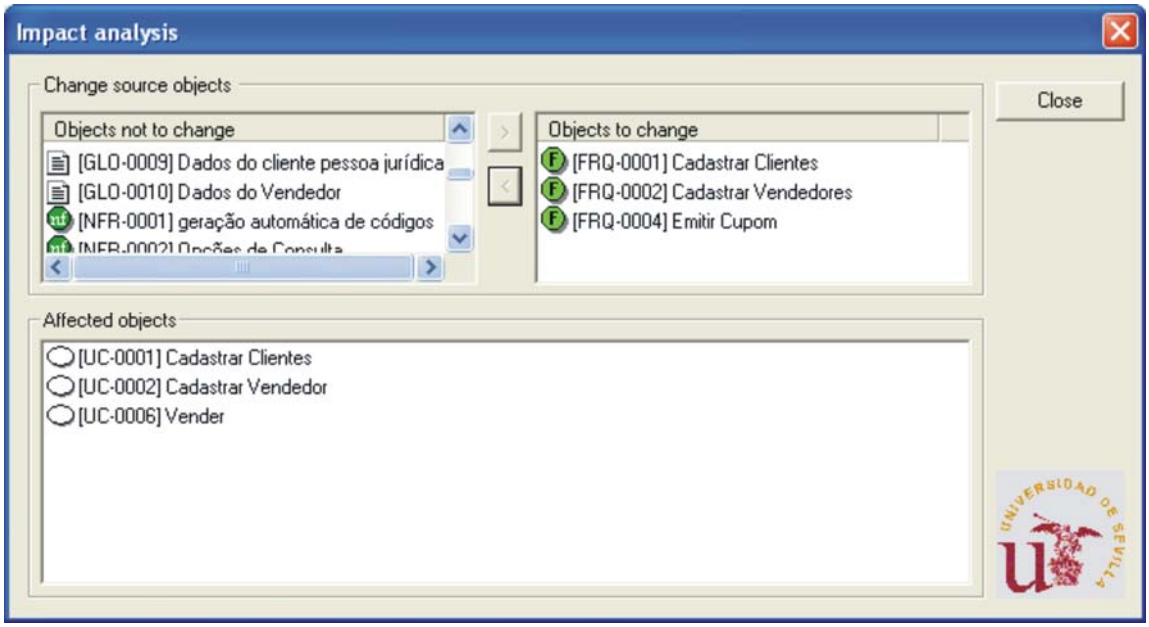



Figura 6-7 Análise de Impacto na ferramenta REM.

## 6.2 AVALIAÇÃO INDIVIDUAL DAS FERRAMENTAS DE CONTROLE DE REQUISITOS

Muito embora não seja objetivo desse trabalho a avaliação individual desse grupo de ferramentas, faz-se necessário avaliar suas características específicas já que essas influenciarão diretamente na qualidade do processo de controle das mudanças dos requisitos. As ferramentas analisadas foram avaliadas com base nos critérios definidos na tabela 3-2 de características esperadas de ferramentas de controle de requisitos no item 3.6.1 do presente trabalho.

Predisposição para detalhamento textual dos requisitos através de integração dinâmica com *softwares* editores de textos ou através de armazenamento de texto explicativo: As duas ferramentas avaliadas atendem essa condição de diferentes formas. Enquanto o RequisitePro permite integração com textos de documentos criados no MS-Word, o REM permite que se criem textos explicativos associados a cada elemento de requisito. Sem dúvida, a solução de tratar o requisito diretamente no documento texto enriquece o conjunto de artefatos e facilita a consistência de seus conteúdos, o que é importante quando se trata de projetos maiores ou mais complexos. Entretanto a solução adotada pela ferramenta REM não descumpre esse pré-requisito e atende satisfatoriamente a projetos de menor complexidade.

**Integração com outras ferramentas de desenvolvimento de sistema:**

Apenas o RequisitePro prevê integração com outras ferramentas tais como o RationalRose, Rational ClearCase e MS-Word. O REM, nessa questão deixa a desejar, já que não atende de nenhuma forma esse requisito.

**Acesso distribuído dos requisitos**: Também nesse quesito, de fundamental importância para projetos com equipes geograficamente distribuída, apenas o RequisitePro provê uma solução.

**Rastreabilidade dos requisitos:** Ambas as ferramentas prevêem esse tipo de controle, nos dois casos é possível associar um requisito a um ou mais casos de uso e verificar quais requisitos não estão sendo tratados em nenhum caso de uso.

**Impacto da mudança de um requisito**: Também é prevista por ambas as ferramentas a análise de impacto que pode acarretar da mudança de algum requisito.

**Permitir utilizar atributos e *queries* em requisitos**: As duas ferramentas permitem a atribuição de propriedades aos requisitos. O requisitePro apresenta ainda a vantagem de permitir criar novos atributos além dos pré-definidos.

**Auditar as mudanças dos requisitos**: Ambas as ferramentas atendem esse quesito, com isso é possível ter controle sobre quem é o responsável pela criação e alteração dos requisitos bem como saber a data da criação e/ou modificação. No REM, é possível ainda se criar um histórico individual para cada mudança.

**Repositório central para todos os elementos dos requisitos**: Ambas as ferramentas possuem repositório para os requisitos através do banco de dados MS-Access. O RequisitePro permite ainda que os requisitos sejam persistidos em banco de dados Oracle ou SQL-Server. As figuras 6.8 e 6.9 apresentam os modelos de dados das ferramentas RequisitePro e REM, respectivamente:

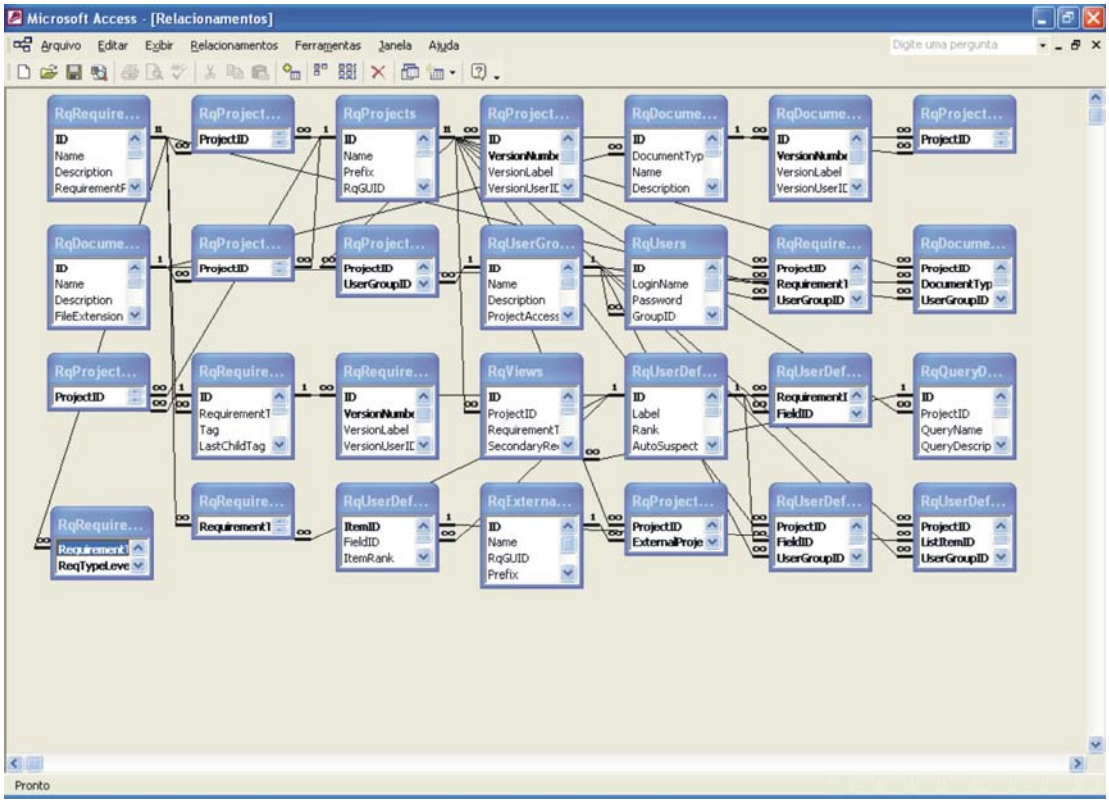



Figura 6-8 Modelo de Dados do RequisitePro.

**Acesso controlado às informações dos requisitos**: Também nesse quesito, de fundamental importância para projetos com equipes maiores, apenas o RequisitePro provê uma solução.

**Estrutura customizavel de requisitos**: Nesse caso, ambas as ferramentas atendem de forma satisfatória, pode-se criar novas pastas para agrupar os requisitos e assim, criar uma estrutura customizada que atenda às características específicas do projeto ou da equipe.

**Oferta de *templates* de projeto**: Esse quesito é de menor importância para equipes mais experientes, no entanto para equipes insipientes com desenvolvimento de sistema ou com a utilização de ferramentas de controle de requisitos, é bastante útil. Ao utilizar o RequisitePro, foi adotado o modelo de projeto baseado em caso de uso, dessa forma, ao iniciar o novo projeto, a ferramenta já apresentou os principais pacotes, modelos de documentos e objetos em geral que deveriam ser utilizados, guiando, assim, o trabalho de documentação. No REM, por outro lado, houve a necessidade de um trabalho inicial de se estabelecer a estrutura da documentação para que depois se pudesse partir para a documentação em si.

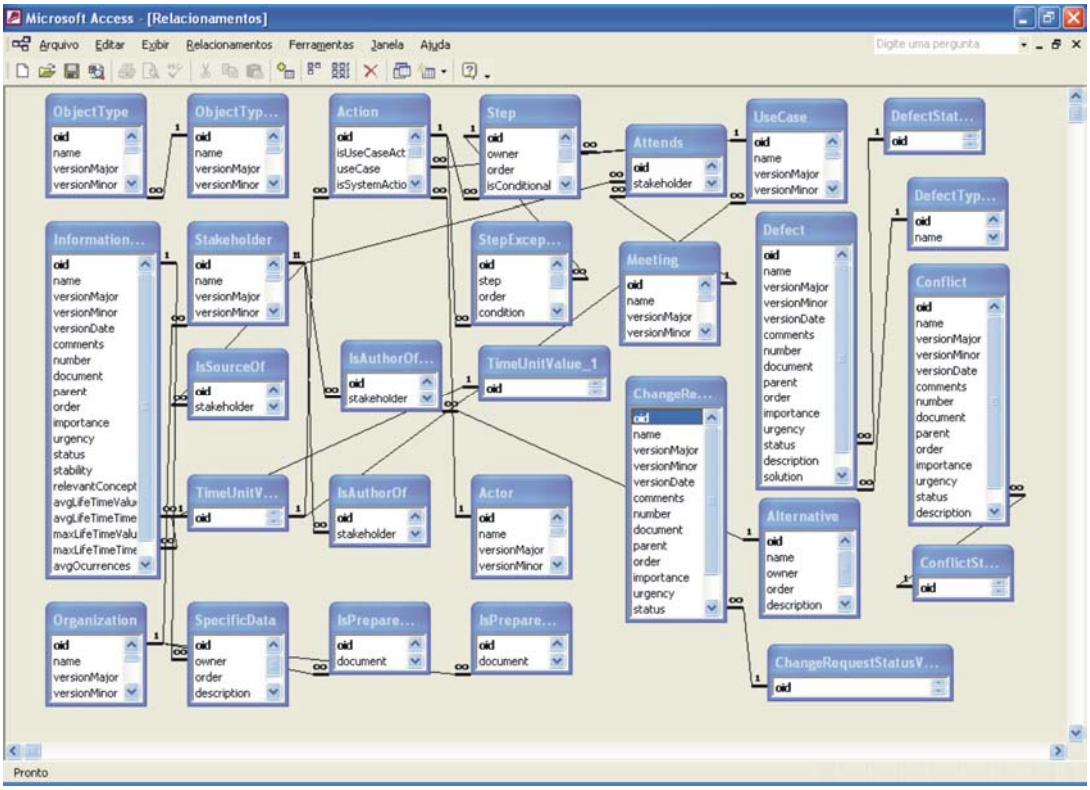



Figura 6-9 Modelo de Dados do REM.

**Exportação dos conteúdos para um formato não proprietário**: Um dos pontos fortes do REM é a geração automática de toda a documentação em HTML. Dessa forma, fica muito mais fácil disponibilizar para os demais membros da equipe a visualização dos requisitos do sistema. O RequisitePro permite a exportação individual das diversas visões (views) do projeto para formato MS-Word ou CSV[25], o que não permite gerar uma documentação completa dos requisitos. Outra solução provida pela ferramenta é o *RequisiteWeb*, que permite que usuários acessem o conteúdo do RequisitePro via *internet*.

## 6.3 FERRAMENTAS DE GERÊNCIA DE CONFIGURAÇÃO

### 6.3.1 Rational ClearCase

Ferramenta desenvolvida pela Rational Corporation, atualmente integrada à IBM, o Rational ClearCase se propõe a apoiar a atividade de SCM – *Software Configuration Management*, ou gerência de configuração de sistemas para ambiente

[25] Acrônimo para Comma-Separated Value

corporativos via rede local ou internet. Para atender pequenos projetos, existe uma versão denominada ClearCase LT. Essa versão contém um subconjunto de ferramentas e é destinada a usuários que não necessitam da escalabilidade, flexibilidade e robustez disponíveis na versão completa. Para o presente trabalho, foi utilizada a versão completa do produto.

Para melhor entender o funcionamento do ClearCase, faz-se necessário o entendimento de alguns termos por ele utilizado:

*File Element*: Um arquivo que contém códigos fonte, documentos HTML, documentos ou qualquer outro tipo de arquivo a ser armazenado.

*Directory Element*: contém *file elements* e outros *directory elements*. Segue o conceito tradicional de diretórios de arquivos.

VOB ou *versioned object base*: Repositório de *file elements* e *directories elements* sob controle de versão. A figura 6-10 apresenta a tela inicial do assistente de criação de VOB no Windows.

No ClearCase, é possível a criação de uma VOB pública. Nesse caso, seus elementos podem ser acessados por toda a comunidade desenvolvedora; VOBs privadas somente podem ser acessadas por seus criadores. A figura 6-11 apresenta o resumo da criação de uma VOB pública.

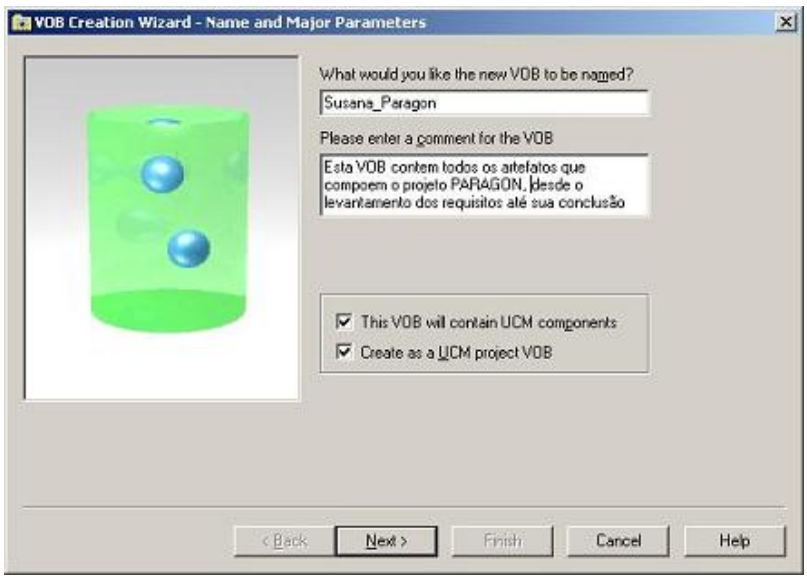



Figura 6-10 Assistente de Criação de VOB no ClearCase.


Figura 6-11 Resumo da criação de uma VOB no ClearCase.

*Version*: Versão ou revisão de um elemento cada versão de cada elemento é armazenado individualmente, sem perder a versão anterior.

Operações de *check-out* e *check-in*: com a operação de *check-out*, é criada uma cópia editável de um elemento em uma visão. Quando é feita a operação de *check-in*, uma nova versão do elemento é incorporada à VOB.

*View*: Visão. Provê acesso a uma versão específica de um ou mais elementos em uma VOB.

### 6.3.2 CVSNT e TortoiseCVS

O CVS é uma substituição do sistema RCS (*Revision Control System*) criado em 1982 por Walter F. Tichy. Conforme relatado em CVSNT (2005), o CVS teve início a partir de um conjunto de *shell scripts* escritos por Dick Grune e publicado no *newsgroup* comp.sources.unix, em dezembro de 1986. Embora a versão atual do CVS não apresente mais nenhum código desses *shell scripts*, muitos dos algoritmos de resolução de conflitos procedem deles. Em abril de 1989, Brian Berliner projetou e codificou o CVS. Em dezembro de 1999 Tony Hoyle converteu o CVS baseado em Unix para rodar sob a plataforma Windows NT. Esse, mais tarde, se tornou o CVSNT, cujo principal objetivo era resolver problemas de adaptação do CVS com o ambiente Windows, principalmente em relação a nome de arquivos. Atualmente o CVSNT é um projeto independente que conta com sua própria comunidade de desenvolvimento e suporte.

O CVSNT é uma ferramenta de código aberto, livre e licenciado sob a Licença Pública Geral[26] GNU e mantém basicamente as mesmas funcionalidades do CVS no que diz respeito ao controle de versões e gerência de configuração. A principal diferença atualmente está nas ferramentas adicionais decorrente de diferenças ideológicas das duas equipes de desenvolvimento.

Tanto o CVS quanto o CVSNT possuem enfoque para o lado servidor da aplicação, sendo necessária a instalação de uma ferramenta cliente no ambiente do usuário, que interaja com o servidor. Dente as ferramentas cliente para CVSNT, destacam-se o TortoiseCVS[27] e WinCVS[28].

TortoiseCVS é um cliente *front-end* cujo objetivo é tornar a utilização do CVSNT mais fácil. TortoiseCVS trabalha diretamente através do Windows Explorer[29], o que torna intuitivo o trabalho daqueles que são familiarizados com esse padrão de interface gráfica .

Para entender o funcionamento do conjunto CVSNT/ TortoiseCVS, é importante o entendimento de alguns termos utilizados:

*Repository*: Um repositório onde são armazenados arquivos e diretórios que estão sob controle de versão. A figura 6-12 apresenta a relação dos repositórios em um servidor CVSNT. Um mesmo servidor pode possuir vários repositórios independentes ou pode-se trabalhar com o conceito de módulos, descrito abaixo.

*Send Box*: Os arquivos sob controle de versão devem ser copiados para o computador do clientes para que sejam editados. Essa cópia local dos arquivos é denominada *send box*. Os arquivos armazenados em uma *send box* podem ser manipulados normalmente pelos seus respectivos programas de acesso. O que indica que uma determinada pasta do Windows é uma *send box* é a alteração na aparência dos ícones dos seus arquivos, conforme mostrado na figura 6-14.

*Module*: Permite agrupar arquivos e diretórios afins dentro de um repositório.

[26] General Public License (GPL) (GNU, 2005).
[27] ToutoiseCVS versão 1.8.22: *software* livre distribuído pelo TortoiseCVS.org sob licença GNU. Conjunto de programas (TORTOISECVS, 2005).
[28] WinCVS versão 2.0.2.4: *software* livre distribuído pelo WinCVS.org sob licença GNU. Conjunto de programas (WINCVS, 2005).
[29] MicrosoftWindows Explorer. Microsoft Corporation, 2003. Parte integrante do Conjunto de programas Microsoft Windows (MICROSOFT, 2005).

Sua utilização não é estritamente necessária, mas são convenientes para a organização dos projetos. A figura 6-13 apresenta o resumo da criação de um módulo em um repositório CVSNT.

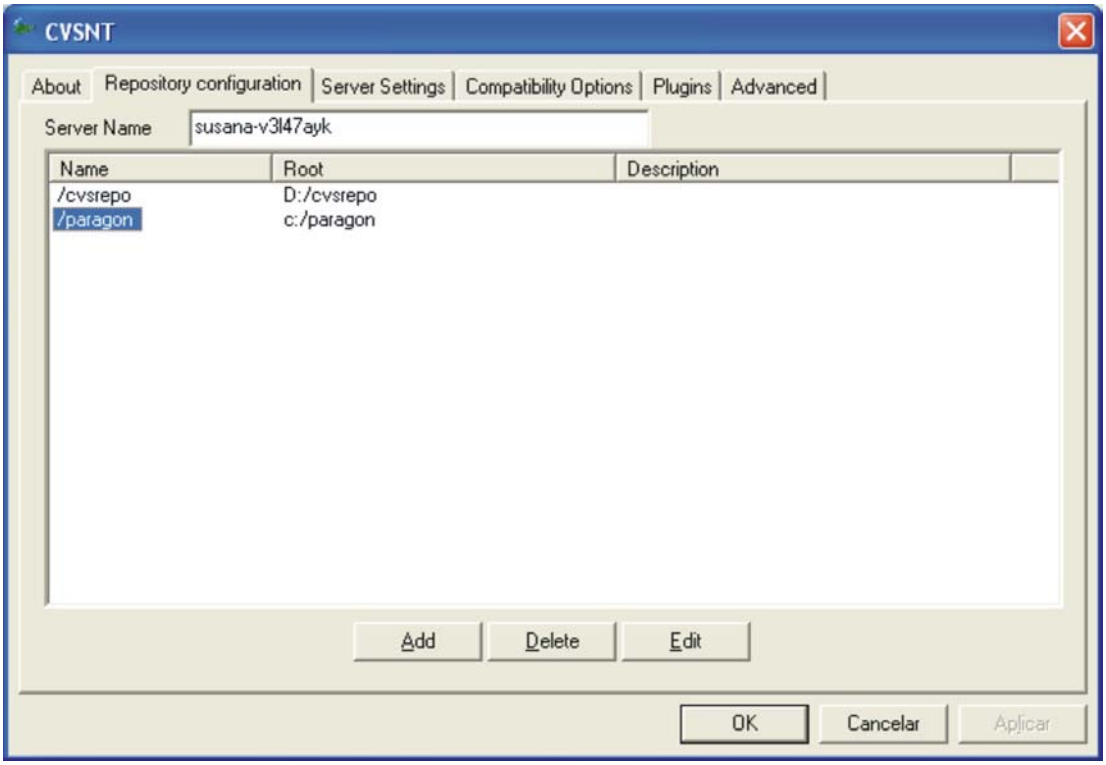



Figura 6-12 Relação de Repositórios do CVSNT.

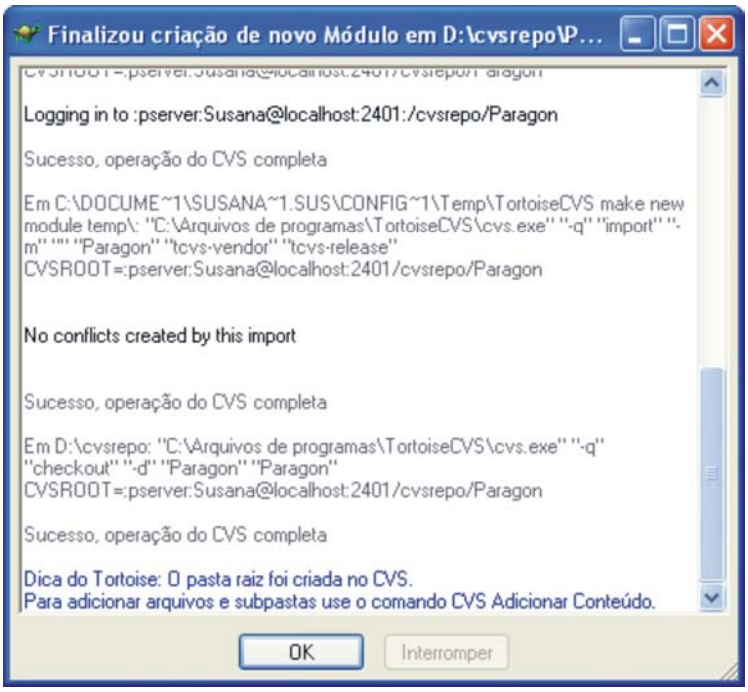



Figura 6-13 Resumo da criação de um módulo no CVSNT.

*Checkout*: Operação para transferência de arquivos de um *repository* para um *send box*.

*Update*: Operação para incorporar atualizações realizadas por outros desenvolvedores em uma área de trabalho local.

*Commit*: Transfere os arquivos modificados localmente para o repositório central. Antes de realizar essa operação, deve-se, entretanto, realizar a operação de *Update* para se resguardar de problemas de conflitos que possam ocorrer quando há mais de um desenvolvedor atualizando o mesmo arquivo em *send boxes* distintas.

*Tag/Label*: Em um dado momento do desenvolvimento faz-se necessário atribuir um rótulo a um conjunto de arquivos, a fim de se estabelecer um identificador de versão. A essa operação no CVSNT, dá-se o nome de *Tagging*.

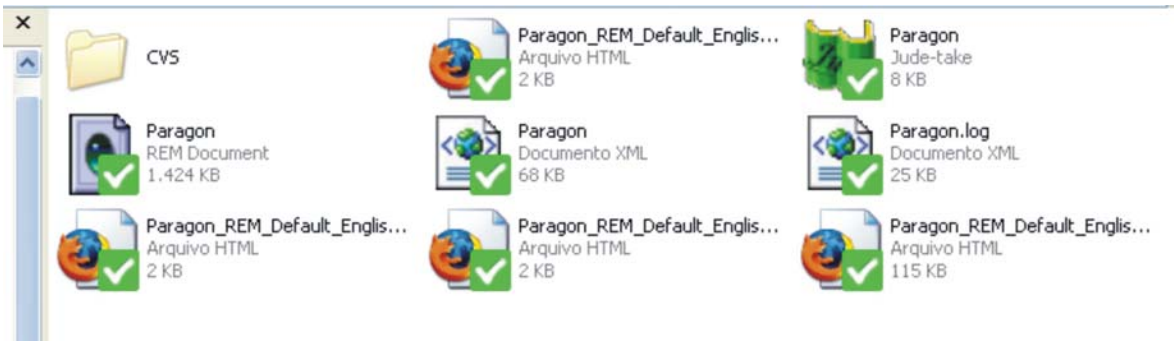



Figura 6-14 Arquivos em uma *send box.*

## 6.4 AVALIAÇÃO INDIVIDUAL DAS FERRAMENTAS DE CONTROLE DE CONFIGURAÇÃO

Muito embora não seja objetivo desse trabalho a avaliação individual desse grupo de ferramentas, faz-se necessário avaliar suas características específicas já que essas influenciarão diretamente na qualidade do processo de controle das mudanças dos requisitos. As ferramentas analisadas foram avaliadas com base nos critérios definidos na tabela 4-2 de características esperadas de ferramentas de gerência de configuração no item 4.5 do presente trabalho.

**Suporte a configurações**: Ambas as ferramentas analisadas implementam essa funcionalidade que é fundamental para que se possa caracterizar uma ferramenta como sendo de controle de configuração. Enquanto o ClearCase

utiliza o conceito de *View*, o CVSNT utiliza o conceito de *Tag/Label*, ambos definidos anteriormente.

**Controle de permissões**: Ambas as ferramentas possuem controle de acesso e permissões através de usuários e grupos, aos quais são atribuídas permissões de gravação, leitura ou acesso.

**Controle de arquivos em diversos formatos**: Ambas as ferramentas suportam o controle de versões de arquivos binários sem comprometer a integridade de seus conteúdos. Nas duas análises, foram armazenados arquivos de ferramentas CASE para geração de diagramas UML e banco de dados com os requisitos do projeto.

**Suporte a desenvolvimento remoto**: Nos testes realizados foram avaliados apenas os ambientes de rede local. Ambas as ferramentas trabalham com o conceito de cliente/servidor e comportam-se adequadamente nesse ambiente. Apesar de descrita nas duas ferramentas, não foi testada a utilização em ambientes remotos.

**Controle das mudanças**: As descrições sobre as atualizações efetuadas são implementadas de formas similares nas duas ferramentas: sempre que se faz a atualização do repositório central com os arquivos alterados. Da mesma forma, ao se retirar um arquivo para atualização (operação de *check-out*), ficam registrados no servidor os dados de identificação dessa operação.

**Portabilidade**: Todos os testes foram realizados em ambiente Windows XP Professional, entretanto, na documentação de ambas as ferramentas é informada a sua compatibilidade com diversos ambientes operacionais, além da possibilidade de se trabalhar com ambientes de clientes diferentes do servidor e diferentes entre si.

## 6.5 AVALIAÇÃO CONJUNTA DAS FERRAMENTAS DE CONTROLE DE REQUISITOS ASSOCIADAS ÀS FERRAMENTAS DE CONTROLE DE CONFIGURAÇÃO

Para a avaliação integrada, foram utilizados dois conjuntos distintos: o primeiro conjunto é composto pelas ferramentas Rational RequisitePro para controle de requisitos e ClearCase para controle de versões alem dessas duas ferramentas, foi

ainda utilizado o RationalRose para geração do modelo de Caso de Uso. O segundo conjunto avaliado é composto pelas ferramentas REM para controle de requisitos, CVSNT para servidor e TortoiseCVS para cliente de controle de versões, foi ainda utilizada a ferramenta Jude-Take para geração do modelo de Casos de Uso.

A escolha do primeiro grupo tem por objetivo trabalhar sobre um conjunto de ferramentas com participação expressiva no mercado mundial de ferramentas CASE. No segundo grupo, a escolha recaiu sobre ferramentas livres, que têm recebido uma atenção especial do mercado corporativo e principalmente do meio acadêmico. A idéia inicial era que esse segundo conjunto fosse utilizado em ambiente operacional Linux para melhor caracterizar o conjunto, entretanto, a ferramenta REM não funciona sobre esse ambiente. Acreditamos que, com a escolha desses dois conjuntos, o presente trabalho possa contribuir com o maior número possível de usuários.

### 6.5.1 Resultados Esperados da Integração entre as Ferramentas

Espera-se, com a presente avaliação, observar o comportamento das ferramentas de gerência de configuração quanto ao tratamento e controle dos arquivos gerados pela equipe de controle de requisitos e sua disponibilização para os demais desenvolvedores de forma padronizada junto ao restante dos artefatos de desenvolvimento.

### 6.5.2 Integração entre Rational RequisitePro e Rational ClearCase

A integração entre as duas ferramentas se dá de forma quase automática. Como mostra a figura 6-15, o próprio RequisitePro já prevê que seus dados sejam armazenados sob controle do ClearCase.

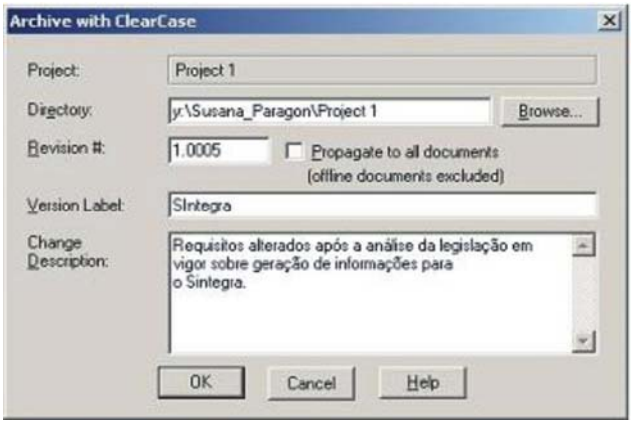



Figura 6-15 Integração entre as ferramentas RequisitePro e ClearCase.

Inicialmente, deve ser informado que os requisitos serão controlados pelo ClearCase e a localização da VOB na qual serão armazenados os elementos do projeto. Feito isso, a cada vez que se deseja criar uma nova versão dos elementos, deve-se solicitar dentro do próprio RequisitePro que se arquivem os elementos com o ClearCase. Nesse caso, é solicitado que o desenvolvedor informe um rótulo e uma descrição para a versão criada. Esses dados farão parte do histórico de revisões dos elementos dentro do ClearCase, como mostra a figura 6-16.

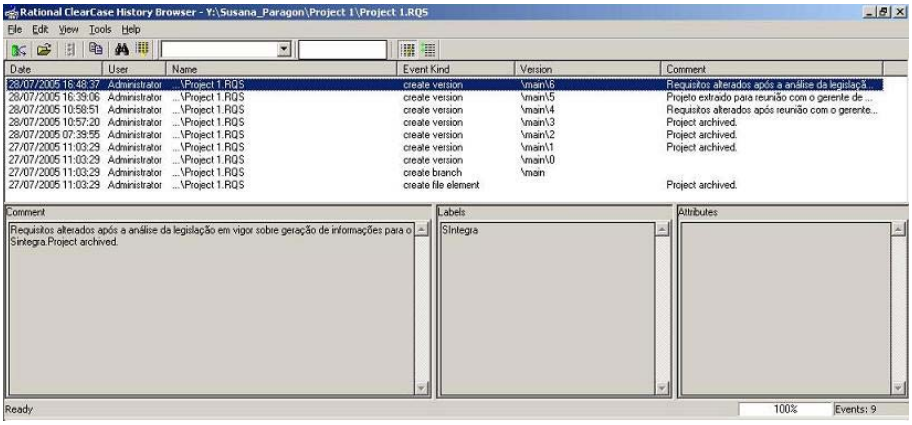



Figura 6-16 Histórico das versões de um arquivo do RequisitePro no ClearCase.

Como pode ser observado, o rótulo e a descrição da alteração acompanham o elemento no ClearCase. Também é criada a *stream* do elemento, conforme apresentado na figura 6-17:

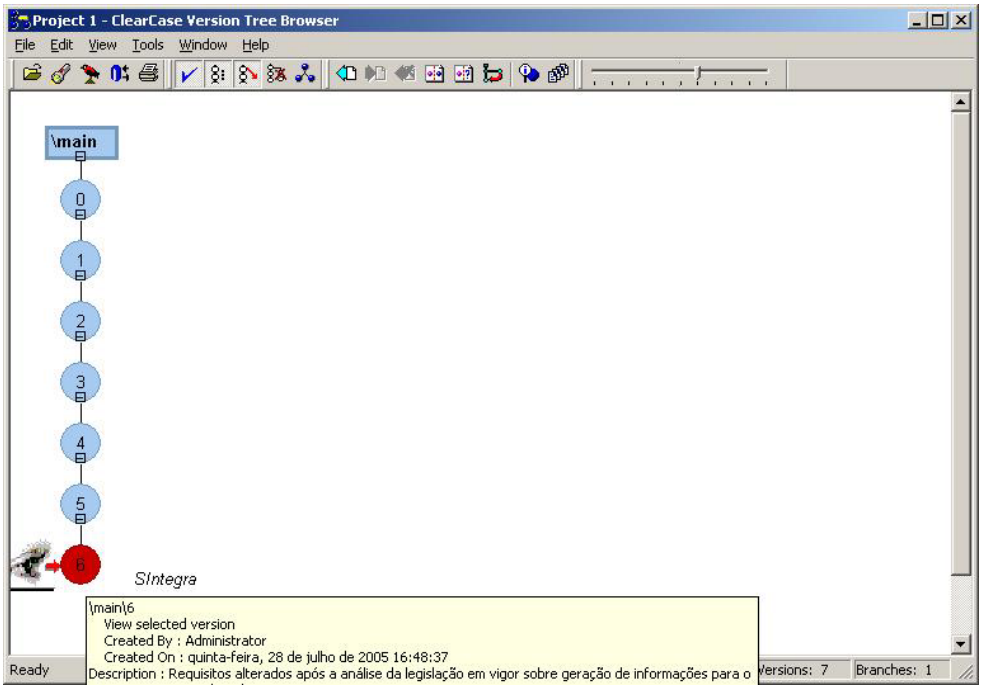



Figura 6-17 Stream de um elemento no ClearCase.

### 6.5.3 Integração entre CVSNT e REM

Esse caso representa a maioria dos casos atuais do mercado de ferramentas CASE, já que, apesar de existirem ferramentas poderosíssimas para diversos fins, a maioria trabalha isoladamente, sem se preocupar com a integração umas com as outras. O fato de não haver integração direta entre as ferramentas não comprometeu a integridade dos dados por elas manipulados.

Inicialmente é necessário configurar o CVSNT com a informação de quais arquivos devem ser tratados como binários, apesar da ferramenta possuir uma relação pré-definida desses tipos e ainda possuir a capacidade de converter os arquivos binários de seus formatos nativos em um formato no qual possa armazená-los em seus repositórios. Essa configuração é sugerida pelo próprio CVSNT a fim de garantir a integridade. Para que o CVSNT identificasse os arquivos gerados pelas ferramentas REM e Jude-Take como binários as linhas de comando abaixo foram inseridas no arquivo `cvswrappers`:

*.rem -kb

*.jude -kb

O parâmetro -kb garante que não haverá conversão na forma de armazenamento dos arquivos binários.

Inicialmente, deve ser criado um novo repositório ou módulo dentro de um repositório existente para o armazenamento das versões dos arquivos. Para o presente trabalho foi criado um novo repositório denominado `/paragon` e nele, um módulo denominado `par_teste`. Feito isso, os arquivos já criados podem ser copiados para a pasta que contem o repositório e quando se deseja extrair algum arquivo, basta selecionar o local de destino e escolher a opção “CVS obter módulo” do TortoiseCVS através do menu de contexto do Windows Explorer, conforme ilustrado na figura 6-18.

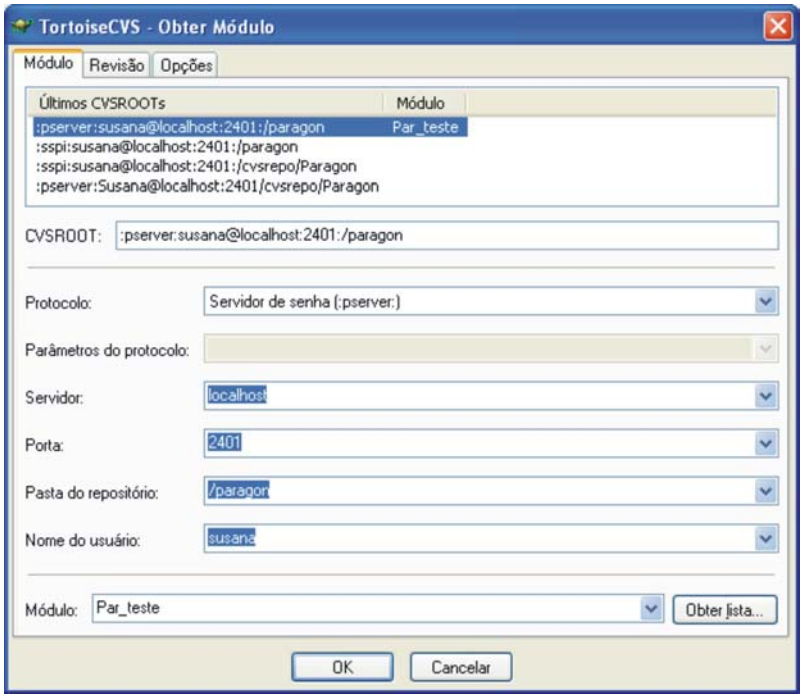



Figura 6-18 Obtenção de um módulo para edição.

A cada operação de *checkout* e *commit*, o CVSNT registra as operações e através do TortoiseCVS é possível observar seu histórico. A figura 6-19 apresenta o histórico de alterações do banco de requisitos através do TortoiseCVS.


Figura 6-19 Histórico de alterações no CVSNT.

# 7 CONCLUSÃO

## 7.1 CONSIDERAÇÕES FINAIS

A definição e a utilização de um processo consistente de desenvolvimento é uma condição *si ne qua non* para o alcance de índices cada vez mais altos de qualidade para os produtos desenvolvidos, característica essa que se tem mostrado cada vez mais necessária para a sobrevivência dos desenvolvedores de *software* no mercado atual. No entanto, não basta somente definir e utilizar um processo, é preciso que este esteja amparado por ferramentas que verdadeiramente dêem suporte ao trabalho dos desenvolvedores, respeitando a particularidade de cada ambiente e perfil de cada equipe desenvolvedora. É de mister importância, também, que essas ferramentas estejam interligadas e dessa forma, amparem-se mutuamente no objetivo de majorar a qualidade do processo, refletindo fidedignamente, através das suas funcionalidades as atividades dos desenvolvedores.

Para que uma ferramenta CASE repercuta a prática humana é necessário que ela esteja amparada para atender àquela que seja talvez a mais certa de todas as características de um desenvolvimento: a mudança nos requisitos ao longo do desenvolvimento.

Baseado nessa hipótese, o presente trabalho apresentou uma análise sobre a importância do controle das alterações dos requisitos apoiado por ferramentas. A partir desta abordagem teórica, foi desenvolvido um estudo de caso para apoiar os procedimentos de verificação e controle das mudanças sofridas pelos seus requisitos.

O controle dos requisitos foi feito por dois conjuntos distintos de ferramentas: o primeiro formado por ferramentas proprietárias e o segundo por ferramentas livres. Cada conjunto foi composto por uma ferramenta de controle de requisitos, uma de

diagramas UML e uma terceira de controle de versões.

Durante o decorrer deste trabalho, foram observados alguns pontos que devem ser mencionados. A mudança ocorrida nos requisitos no sistema é prevista por todos os processos atuais de engenharia de software e refletem diretamente no produto final do processo, entretanto poucas são as equipes que possuem um controle efetivo sobre seus efeitos. Quanto mais interligadas estiverem as ferramentas CASE, mais interligados estarão os artefatos por elas produzidos, por isso a sugestão feita pelo presente trabalho é que o mesmo ambiente que controla as mudanças nos códigos fontes sirvam para o controle das alterações nos requisitos. Isso aumenta a integração e a familiaridade dos desenvolvedores com o ambiente.

## 7.2 CONTRIBUIÇÕES DA DISSERTAÇÃO

A crescente preocupação das empresas em produzir *softwares* de qualidade reconhecida tem elevado a demanda de se estabelecer padrões de eficiência nos seus processos de desenvolvimento. Nesse contexto, o objetivo deste trabalho foi definir um processo sistemático de controle das mudanças nos artefatos de requisitos de *software* durante o seu processo de desenvolvimento. As contribuições desta abordagem são, portanto:

- A proposta de um modelo de organização dos artefatos através de ferramentas CASE de controle de requisitos associadas a ferramentas de gerência de configuração e mudança de software;
- Um levantamento das ferramentas existentes no mercado mundial para a realização dessa atividade de controle das mudanças dos requisitos de *software*;
- O estabelecimento de critérios relevantes para a avaliação de ferramentas CASE de controle de requisitos, modelagem UML e gerência de configuração e mudança;
- A análise comparativa da utilização de conjuntos diferentes de ferramentas para apoiar a atividade de controle de mudanças dos requisitos de *software*;
- A compilação de modelos de documentos para a criação dos artefatos de gerência de requisitos do processo de desenvolvimento de *software*;

Finalmente, espera-se que o presente trabalho tenha atingido seu principal objetivo, que é o de proporcionar aos desenvolvedores em geral maior qualidade às suas atividades para alcançar o grande desafio da engenharia de software: desenvolver produtos com qualidade, no prazo e orçamento previstos.

## 7.3 PERSPECTIVAS FUTURAS

Como perspectiva imediata decorrente do presente trabalho, vislumbra-se o desenvolvimento de uma ferramenta de código livre que integre a função de repositório de requisitos com outra ferramenta de código livre de gerência de configuração e mudança, como o CVS ou CVSNT. Assim, espera-se que as equipes de desenvolvimento de *software* possam se beneficiar da utilização de uma ferramenta CASE apropriada desde o levantamento preliminar dos requisitos.

O resultado dos estudos desenvolvidos no presente trabalho apontam também para a necessidade de controle das mudanças ocorridas nos artefatos gerados, não somente nas atividades de gerência de requisitos, mas de todas as atividades do processo de desenvolvimento de software. Tal constatação impele para que se continue a investigação detalhada das demais atividades bem como a existência e compatibilidade de ferramentas CASE disponíveis no mercado para apoiar cada uma dessas atividades. Em especial, pretende-se estender o presente estudo para as atividades de análise e projeto de sistemas.

# 8 REFERÊNCIAS


ARCH. Disponível em http://www.gnu.org/software/gnu-arch/. Acessado em 09/10/2005.

BATISTA, Edinelson Aparecido; CARVALHO, Ariadne M.B.R. *Uma Taxonomia Facetada para Técnicas de Levantamento de Requisitos*. Anais do WER03 - Workshop em Engenharia de Requisitos, Piracicaba-SP, 2003.

BERGER, Patrícia Machado. *Instanciação de Processos de Software em Ambientes Configurados na Estação TABA*. Tese de Doutorado. Disponível em http://ramses.cos.ufrj.br/taba/. Acessado em 18/06/2004. Rio de Janeiro: COPPE-UFRJ, 2003.

BERGMANN, Ulf. *Evolução de Cenários Através de um Mecanismo de Rastreamento Baseado em Transformações*. Tese de Doutorado. Disponível em http://www-di.inf.puc-rio.br/~julio/bnncap3.pdf. Acessado em 19/05/05, Rio de janeiro: PUC-RJ, 2003.

BITKEEPER. Disponível em http://www.bitkeeper.com/. Acessado em 09/10/2005.

BREITMAN, Karin Koogan; LEITE J.C.S.P. *Suporte Automatizado à Gerência da Evolução de Cenários*. Tese de Doutorado. Disponível em http://www.inf.puc-rio.br/~wer98/. Acessado em 09/12/2004. Rio de Janeiro: PUC-RJ, 1998.

BOEHM, Barry. *A Spiral Model of Software Development and Enhancement*. Computer, vol. 21, no. 5, pp. 61-72. Disponível em http://portal.acm.org/citation.cfm?id=45797.45801. Acessado em 19/08/2005. Los Alamitos, CA, USA, 1988.

BOOCH, Grandy. *UML, Guia do Usuário*. Rio de Janeiro: Editora Campus, 2000.

BORLAND. Disponível em http://www.borland.com. Acessado em 09/10/2005.

CVSHOME. Disponível em http://www.cvshome.org. Acessado em 09/10/2005.

CVSNT. Disponível em http://www.march-hare.com/cvsnt. Acessado em 09/10/2005.

DAVIS, Alan M, Dean A. Leffingwell. Using Requirements Management to Speed

Delivery of Higher Quality Application. Disponível em http://www-306.ibm.com/ software/rational/ info/literature/reqanalysis.jsp. Acessado em 20/08/2004, 1996.

DIAS, André Felipe. *GC do Ponto de Vista das Ferramentas de Apoio*. Disponível em http://www.pronus.eng.br/artigos_tutoriais/gerencia_configuracao/gerencia_configuracao.php?pagNum=3. Acessado em 20/04/2006.

FOWLER, M., SCOTT, K., UML *Distilled - Applying the Standard Object Modeling Language*, Addison Wesley Object Technology Series, 1997

GNU. *The GNU General Public License*. Disponível em http://www.gnu.org/licenses/ licenses.html#GPL. Acessado em 24/10/2005.

GOMES JUNIOR, Augusto Gonçalves, *Avaliação de Processos de Software baseado em Medições*. Tese de Doutorado. Rio de Janeiro: COPPE/UFRJ, 2000.

GRADY, Robert. Practical Software Metrics for Project Management and Process Improvement. Prentice-Hall, 1992.

HAZAN, Claudia e LEITE, Julio César Sampaio do Prado. *Indicadores para a Gerência de Requisitos*. Disponível em http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER03/ claudia_hazan.pdf. Acessado em 14/05/2005. Rio de Janeiro: PUC-RJ, 2003.

HOLANDA, Aurélio Buarque de Holanda. *Dicionário da língua portuguesa*. Rio de Janeiro,: Nova Fronteira, 1988.

IGATECH. Disponível em http://www.igatech.com. Acessado em 09/12/2005.

ISO International Organization for Standardization. Information Technology - Software Product Evaluation- Quality Characteristics and Guidelines for their use, ISO/IEC 9126. [S.l.], 1991.

JARKE, M. *Requirements tracing*. Communications of the ACM, 41(12):32--36, 1998

JUDE. Disponível em http://www.componentsource.com/. Acessado em 09/12/2005.

KRUCHTEN, Philippe. The Rational Unified Process: an Introduction. Boston, EUA, 2000.

LEFFINGWELL, Dean. *Features, Use Cases, Requirements, Oh My!*. Disponível em http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/featucreqom.pdf. Acessado em 20/08/2004. 2003.

LEITE, Julio César Sampaio de Prado. “Gerenciando a Qualidade de Software com Base em Requistos”. In: Rocha, Maldonado, Weber. *Qualidade de Software: Teoria e Prática*. São Paulo: Prentice-Hall, 2001.

LOPES, Leandro T.; Majdenbaum, Azriel; Audy ,Jorge Luis N. *Uma Proposta para Processo de Requisitos em Ambientes de Desenvolvimento Distribuído de Software*. Anais do WER03 - Workshop em Engenharia de Requisitos, Piracicaba-SP, Brasil, 2003.

MACHADO, Luiz Felipe Dionísio Cavalcanti. *Modelo para Definição de Processos de Software na Estação TABA*. Tese de Doutorado. Disponível em http://ramses.cos.ufrj.br/taba/. Acessado em 18/06/2004. Rio de Janeiro: COPPE/UFRJ, 2000.

MARTINS, Vidal. *Engenharia de Programas - Uma Visão Geral*. Paraná, 1995. Disponível em http://www.pr.gov.br/celepar/celepar/batebyte/edicoes/1995/bb39/engenhar.htm. Acesso em: 08/11/2004.

MICROSOFT. Disponível em http://www.microsoft.com. Acessado em 09/12/2005.

MILLIGAN, Tom. Better Software Configuration Management Means Better Business: The Seven Keys to Improving Business Value. Disponível em http://www-306.ibm.com/software/rational/info/ literature/. Acessado em 20/08/2004, 2003.

MOURA, L.M.V., ROCHA, A.R.C., 1992, Ambientes de Desenvolvimento de Software, Relatório Técnico COPPE/UFRJ ES-271/92, Rio de Janeiro, RJ, Brasil. Disponível em http://www.cos.ufrj.br. Acessado em 08/11/2004.

OLIVEIRA, Angelina A.A.C.P. *Gerência de Configuração de Software*. Instituto Nacional de Tecnologia da Informação – ITI, órgão do Ministério da Ciência e Tecnologia, Campinas, SP: 2001.

PAULA FILHO, Wilson de Pádua, *Engenharia de Software Fundamentos, Métodos e Padrões*: Rio de Janeiro: LTC, 2001.

PERFORCE. Disponível em http://www.perforce.com/. Acessado em 09/10/2005.

PRESSMAN, Roger S., Engenharia de Software: São Paulo: Makron Books, 1995.

RATIONAL *Use Case management with Rational Rose and Rational RequisitePro*. Disponível em http://www-306.ibm.com/software/rational/info/literature/reqanalysis.jsp. Acessado em 20/08/2004.

ROCHA, Ana Regina Cavalcanti da, Maldonado, José Carlos, Weber, Kival Chaves. *Qualidade de Software – Teoria e Prática*. São Paulo: Prentice Hall, 2001.

RUP, Rational Unified Process, versão 2003.06.00.65: Rational Corporation, 2004. Conjunto de Programas. 1 CD-ROM.

SEI. *Software Engineering Institute Technical Report CMU/SEI-93-TR-024*, 1993. Disponível em http://www.sei.cmu.edu. Acessado em 20/08/2005.

SEI. Capability Maturity Model Integration, Version 1.1 Continuous Representation, 2002. Disponível em http://www.sei.cmu.edu. Acessado em 20/08/2005.

SOURCEFORGE. Disponível em http://www.sourceforge.net. Acessado em 09/10/2005.

TELELOGIC. Disponível em http://www.telelogic.com/corp/index.cfm. Acessado em 09/12/2005.

TIGRIS. Disponível em http://argouml.tigris.org. Acessado em 09/12/2004.

TORTOISECVS. Disponível em http://www.tortoisecvs.org/. Acessado em 24/10/2005.

TRAVASSOS, G.H. *O Modelo de Integração de Ferramentas da Estação TABA*, Tese de Doutorado. Rio de Janeiro: COPPE/UFRJ, 1994. Disponível em http://ramses.cos.ufrj.br/. Acessado em 20/08/2005.

TORO, Amador Duran. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información*. Tese de Doutorado. Disponível em http://www.lsi.us.es/~amador/. Acessado em 16/09/2005. Sevilha, Espanha, 2000.

UGS. Disponível em http://www.ugs.com/index.shtml. Acessado em 09/12/2005.

UTAH, University of. *Rapid Prototyping Home Page*. Disponível em http://www.cc.utah.edu/~asn8200/rapid.html. Acessado em 03/02/2006.

WINCVS. Disponível em http://www.wincvs.org/. Acessado em 09/10/2005.

ANEXO 01 – PLANO DE GERENCIAMENTO DE REQUISITOS DO RUP

# <Nome do Projeto>
# Plano de Gerenciamento de Requisitos

**Versão <1.0>**

## 1.Introdução

[A introdução do **Plano de Gerenciamento de Requisitos** fornece uma visão geral de todo o documento. Ela contém a finalidade, o escopo, as definições, os acrônimos, as abreviações, as referências e a visão geral deste **Plano de Gerenciamento de Requisitos**.]

### 1.1 Finalidade

[Especifique a finalidade deste Plano de Gerenciamento de Requisitos.]

### 1.2 Escopo

[Uma breve descrição do escopo deste **Plano de Gerenciamento de Requisitos**.]

### 1.3 Definições, Acrônimos e Abreviações

[Esta subseção fornece as definições de todos os termos, acrônimos e abreviações necessárias à adequada interpretação do **Plano de Gerenciamento de Requisitos**. Essas informações podem ser fornecidas mediante referência ao Glossário do projeto.]

### 1.4 Referências

[Esta subseção deve fornecer uma lista completa de todos os documentos mencionados em qualquer outra parte do **Plano de Gerenciamento de Requisitos**. Cada documento é identificado por título, número do relatório (se aplicável), data e organização de publicação. Especifique as fontes a partir das quais as referências podem ser obtidas. Essas informações podem ser fornecidas por um anexo ou outro documento.

### 1.5 Visão Geral

[Esta subseção descreve o que o restante do **Plano de Gerenciamento de Requisitos** contém e explica como o documento está organizado.]

# 2. Gerenciamento de Requisitos

## 2.1 Organização, Responsabilidades e Interfaces

[Descreva quem será responsável por executar as diversas atividades descritas nos fluxos de trabalho de requisitos.]

## 2.2 Ferramentas, Ambiente e Infra-estrutura

[Descreva o ambiente computacional e as ferramentas de software a serem usadas na execução das funções de Gerenciamento de Requisitos durante todo o ciclo de vida do produto ou do projeto.

Descreva as ferramentas e os procedimentos usados para controlar a versão dos itens de Requisitos gerados durante todo o ciclo de vida do produto ou do projeto.]

# 3. O Programa de Gerenciamento de Requisitos

## 3.1 Identificação dos Requisitos

[Descreva os itens de rastreabilidade e defina como eles devem ser nomeados, marcados e numerados. (Um item de rastreabilidade é qualquer elemento de projeto que necessite ser explicitamente rastreado a partir de outro item textual ou de modelo, a fim de manter-se um controle das dependências entre eles. Em relação ao Rational RequisitePro, essa definição pode ser reformulada da seguinte maneira: qualquer elemento de projeto representado no RequisitePro por uma instância de um tipo de requisito do RequisitePro.)]

[Paca cada tipo de artefato ou documento de requisitos do projeto, liste os itens de rastreabilidade contidos nele e explique brevemente para que ele é usado. Também poderá ser conveniente listar o papel responsável.]

| Artefato Tipo de Documento) | Item de Rastreabilidade | Descrição |
|---|---|---|
| Solicitações dos Principais Envolvidos (STR) | Solicitação do Envolvido (STRQ) | As principais solicitações, incluindo Solicitações de Mudança, dos envolvidos.<br><br>[Se você usar uma ferramenta de Gerenciamento de Solicitações de Mudança, |

| | | |
|---|---|---|
| | | como o Rational ClearQuest, as solicitações dos principais envolvidos serão freqüentemente armazenadas nessa ferramenta e não serão duplicadas na ferramenta de gerenciamento de requisitos.] |
| Visão (VIS) | Necessidade dos Envolvidos (NEED) | A principal necessidade dos envolvidos ou dos usuários |
| Visão (VIS) | Recurso (FEAT) | Condições ou recursos desse release do sistema |
| Modelo de Casos de Uso | Caso de Uso (UC) | Os casos de uso desse release, documentados no Rational Rose |
| Especificação Suplementar (SS) | Requisito Suplementar (SUPP) | Os requisitos não-funcionais que não são capturados no modelo de casos de uso |

## 3.2 Rastreabilidade

[Visão geral de rastreabilidade; por exemplo, um gráfico de rastreabilidade.]

### 3.2.1 Critérios de <item de rastreabilidade>

[Para cada item de rastreabilidade identificado, liste todas as regras ou diretrizes adicionais que se aplicam aos vínculos de rastreabilidade. Descreva todas as restrições aplicáveis como, por exemplo, "cada recurso aprovado deve estar relacionado a um ou mais Casos de Uso ou a um ou mais Requisitos Suplementares."]

## 3.3 Atributos

### 3.3.1 Atributos de <item de rastreabilidade>

[Para cada item de rastreabilidade identificado, liste os atributos que você usará e explique brevemente o que significam. Por exemplo, poderão ser especificados os seguintes atributos para um item de rastreabilidade de "recurso".]

Status

[Definido pela equipe de gerenciamento do projeto após a negociação e a revisão. Controla o andamento durante a definição da baseline do projeto.]

| | |
|---|---|
| Proposto | [Usado para descrever recursos que estão sendo discutidos, mas que ainda não foram revisados e aceitos pelo "canal oficial" como, por exemplo, um grupo de trabalho formado por representantes da equipe do projeto, do gerenciamento do produto e da comunidade de usuários ou de clientes.] |
| Aprovado | [Recursos que são considerados úteis e viáveis, e que foram aprovados para implementação pelo canal oficial.] |
| Rejeitado | [Rejeitado pelo canal oficial.] |
| Incorporado | [Recursos incorporados à baseline do produto em um momento específico no tempo.] |

Benefício

[Definido pelo departamento de marketing, pelo gerente do produto ou pelo analista de negócios. Todos os requisitos diferem entre si. A classificação dos requisitos por seu benefício relativo para o usuário final inicia um diálogo com os clientes, analistas e membros da equipe de desenvolvimento. Usado no gerenciamento do escopo e na determinação da prioridade de desenvolvimento.]

| | |
|---|---|
| Crítico | [Recursos essenciais. A não-implementação implica que o sistema não atenderá às necessidades do cliente. Todos os recursos críticos deverão ser implementados no release ou a programação será retardada.] |
| Importante | [Recursos importantes para a eficácia e a eficiência do sistema da maior parte dos aplicativos. A funcionalidade não poderá ser fornecida facilmente de outra maneira. Caso um recurso importante não seja incluído, a satisfação do cliente ou do usuário, ou até a receita, poderão ser afetadas, mas isso não retardará o release.] |
| Útil | [Os recursos que são úteis em aplicativos menos típicos ou para os quais podem se obter soluções razoavelmente eficientes serão usados com menor freqüência. Não se pode esperar nenhum impacto significativo na receita ou na satisfação do cliente se esse tipo de recurso não for incluído em um release.] |

Esforço

[Definido pela equipe de desenvolvimento. Como algumas funcionalidades necessitam de mais tempo e de mais recursos do que outras, estimar o número de semanas de participação de cada pessoa ou equipe, as linhas de código necessárias ou os pontos de função, por exemplo, é a melhor maneira de avaliar a complexidade e definir expectativas do que poderá ou não ser feito em um determinado período de tempo. Usado no gerenciamento do escopo e na determinação da prioridade de desenvolvimento.]

Risco

[Definido pela equipe de desenvolvimento e baseado na probabilidade de ocorrerem eventos indesejáveis no projeto como, por exemplo, custos excessivos, atrasos na programação ou até cancelamentos. A maior parte dos gerentes de projeto considera que a categorização dos riscos em altos, médios e baixos é suficiente, embora sejam possíveis gradações ainda mais específicas. Freqüentemente os riscos poderão ser avaliados indiretamente medindo-se o grau de incerteza (intervalo) da programação estimada das equipes dos projetos.]

Estabilidade

[Este atributo é definido pelo analista e pela equipe de desenvolvimento. Baseia-se na probabilidade de o recurso sofrer mudanças ou na probabilidade de a equipe vir a compreender o recurso de uma forma diferente. É usado para ajudar a estabelecer prioridades de desenvolvimento e determinar os itens para os quais uma averiguação adicional é a próxima ação apropriada.]

Release-alvo

[Registra a versão planejada do produto em que o recurso aparecerá pela primeira vez. Este campo poderá ser usado para alocar recursos de um documento **Visão** em um release de baseline específico. Quando for usado em conjunto com o campo de status, sua equipe poderá propor, registrar e discutir vários recursos do release sem que tenham que ser necessariamente desenvolvidos. Somente serão implementados os recursos cujo Status estiver definido como Incorporado e cujo Release-alvo estiver definido. Quando ocorrer o gerenciamento de escopo, o Número da Versão do Release-alvo poderá ser aumentado de modo que o item permaneça no documento **Visão**, mas seja programado para um release posterior.]

Atribuído A

[Em muitos projetos, os recursos serão atribuídos a "equipes de recursos" responsáveis por averiguar e escrever os requisitos do software, e também por sua implementação. Esta lista suspensa simples ajudará a todos da equipe do projeto a compreenderem melhor suas responsabilidades.]

Motivo

[Este campo de texto é usado para rastrear a origem do recurso solicitado. Os

requisitos existem devido a razões específicas. Este campo registra uma explicação ou uma referência a uma explicação. Por exemplo, a referência poderá ser ao número de uma linha e de uma página de uma especificação de requisitos do produto ou a um minúsculo marcador em um vídeo de uma entrevista com um importante cliente.]

## 3.4 Relatórios e Medidas

[Descreva o conteúdo, o formato e a finalidade dos relatórios/medidas solicitados.]

## 3.5 Gerenciamento de Mudança de Requisitos

### 3.5.1 Processamento e Aprovação de Solicitações de Mudança

[Descreva o processo através do qual os problemas e mudanças são enviados, revisados e organizados. Nessa descrição, deverá estar incluído o processo para negociar mudanças de requisitos com os clientes e quaisquer processos, atividades e restrições contratuais].

### 3.5.2 Comitê de Controle de Mudança (CCB)

[Descreva a participação e os procedimentos para processar solicitações e aprovações de mudança a serem seguidos pelo CCB.]

### 3.5.3 *Baselines* do Projeto

[As baselines funcionam como um padrão oficial no qual os trabalhos subseqüentes são baseados. Somente mudanças autorizadas podem ser efetuadas nas baselines.

Descreva em que pontos do ciclo de vida do projeto ou produto as baselines devem ser estabelecidas. As baselines mais comuns devem ser definidas ao final de cada uma das fases de Iniciação, Elaboração, Construção e Transição. Elas também podem ser geradas no final de iterações ocorridas dentro das várias fases ou com freqüência ainda maior.

Descreva quem autoriza uma baseline e o que ela contém.]

## 3.6 Fluxos de Trabalho e Atividades

[Descreva os fluxos de trabalho e as atividades que se aplicam ao gerenciamento de requisitos.

Descreva as atividades de revisão, incluindo seus objetivos, responsabilidades, andamento e procedimentos].

# 4. Marcos

[Identifique os marcos internos e do cliente relacionados ao esforço de Gerenciamento de Requisitos. Esta seção deve incluir detalhes sobre quando o Plano de Gerenciamento de Requisitos propriamente dito deverá ser atualizado.]

# 5. Treinamento e Recursos

[Descreva o treinamento, o pessoal e as ferramentas de software necessários para implementar as atividades de Gerenciamento de Requisitos especificadas.]

ANEXO 02 – SOLICITAÇÃO DOS PRINCIPAIS ENVOLVIDOS DO RUP

# <Nome do Projeto>
# Solicitações dos Principais Envolvidos
# Versão <1.0>

# Solicitações dos Principais Envolvidos

## 1. Introdução

[A introdução das Solicitações dos Principais Envolvidos deve fornecer uma visão geral de todo o documento. Ela deve incluir a finalidade, o escopo, as definições, os acrônimos, as abreviações, as referências e a visão geral desta coleta de Solicitações dos Principais Envolvidos.]

[Script de Entrevista Sem Contexto: Existem grandes oportunidades em nossa indústria para aprimorar os esforços de desenvolvimento de aplicativos. Compreender as necessidades dos usuários ou dos envolvidos antes de iniciar o desenvolvimento é crucial para aprimorar esse processo. Há muitas técnicas disponíveis para averiguar as solicitações dos usuários ou dos envolvidos. Uma técnica simples e de baixo custo que poderá ser adequadamente usada em praticamente todas as situações é a Entrevista Genérica. A Entrevista Genérica poderá ajudar o desenvolvedor ou o analista a compreender os objetivos e os problemas dos usuários ou dos envolvidos. Com esse discernimento, os desenvolvedores poderão criar aplicativos que se ajustem às necessidades reais dos usuários ou dos envolvidos e que aumentem sua satisfação.]

[A Entrevista Genérica neste template apresenta perguntas criadas para buscar uma compreensão dos problemas e do ambiente dos usuários ou dos envolvidos. Essas perguntas exploram os requisitos de funcionalidade, usabilidade, confiabilidade, desempenho e suportabilidade do aplicativo. Com o uso da Entrevista Genérica, o desenvolvedor ou o analista obterá um conhecimento do problema que está sendo resolvido, assim como uma compreensão das percepções dos usuários ou dos envolvidos acerca das características das soluções eficazes.]

### 1.1 Finalidade

[Especifique a finalidade desta coleta de Solicitações dos Principais Envolvidos.]

[Diretrizes de Uso: Se a Entrevista Genérica não estiver adequada às suas necessidades, você poderá modificá-la sempre que desejar. Com um pouco de preparação e uma entrevista bem estruturada, qualquer desenvolvedor ou analista poderá entrevistar de maneira eficiente. Abaixo são relacionadas algumas dicas:

- Faça uma pesquisa prévia da formação dos envolvidos ou dos

usuários e da empresa.

- Revise as perguntas antes da entrevista.

- Consulte o formato durante a entrevista para assegurar que as perguntas certas estão sendo feitas.

- Resuma os dois ou três problemas principais no final da entrevista. Repita o que você aprendeu para confirmar sua compreensão.

Não deixe que o script limite demais as suas iniciativas. Depois de ter sido estabelecida uma harmonia, geralmente a entrevista tomará vida própria, e o envolvido ou o usuário poderá falar bastante sobre as dificuldades que está enfrentando. Não os interrompa. Anote essas respostas o mais rápido possível. Faça perguntas após obter essas informações. Após essa troca de informações chegar a um fim lógico, prossiga com as outras perguntas contidas na lista. Boa sorte e boa entrevista!]

## 1.2 Escopo

[Uma breve descrição do escopo desta coleta de Solicitações dos Principais Envolvidos; a que Projeto(s) ela está associada e tudo o mais que seja afetado ou influenciado por este documento.]

## 1.3 Definições, Acrônimos e Abreviações

[Esta subseção deve fornecer as definições de todos os termos, acrônimos e abreviações necessárias à adequada interpretação das Solicitações dos Principais Envolvidos. Essas informações podem ser fornecidas mediante referência ao Glossário do projeto.]

## 1.4 Referências

[Esta subseção deve fornecer uma lista completa de todos os documentos mencionados em qualquer outra parte das Solicitações dos Principais Envolvidos. Cada documento deverá ser identificado por título, número do relatório (se aplicável), data e organização de publicação. Especifique as fontes a partir das quais as referências podem ser obtidas. Essas informações podem ser fornecidas por um anexo ou outro documento.]

## 1.5 Visão Geral

[Esta subseção descreve o que o restante das Solicitações dos Principais Envolvidos contém e explica como o documento está organizado.]

## 2. Estabelecimento do Perfil dos Envolvidos ou dos Usuários

- Nome: Empresa / Indústria:
- Cargo:
- Quais são suas principais responsabilidades?
- Que produtos você produz? Para quem?
- Como o sucesso é medido?
- Que problemas interferem no seu sucesso?
- Que tendências, se houver, facilitam ou dificultam o seu trabalho?

## 3. Avaliação do Problema

- Para que problemas de <tipo de aplicativo> você necessita de boas soluções?
- Quais são elas? [Dica: Procure sempre perguntar "Algo mais?]

Pergunte em relação a cada problema:

- Por que este problema existe?
- Como é possível solucioná-lo agora?
- Como você gostaria de solucioná-lo?

## 4. Compreensão do Ambiente do Usuário

- Quem são os usuários?
- Qual é a sua formação educacional?
- Quais são seus conhecimentos de computador?
- Os usuários estão familiarizados com esse tipo de aplicativo?
- Que plataformas estão sendo usadas? Quais são seus planos para plataformas futuras?
- Quais são os aplicativos adicionais que você utiliza com os quais precisamos estabelecer uma interface?
- Quais são suas expectativas em relação à usabilidade do produto?
- Quais são suas expectativas em relação ao tempo de treinamento?
- Que tipos de documentação impressa e on-line são necessários?

## 5. Recapitulação para fins de Entendimento

- Você me disse [liste com suas próprias palavras os problemas descritos pelo envolvido]:
- Isso representa os problemas que você está enfrentando em sua solução existente?
- Que outros problemas, se houver, você está enfrentando?

## 6. Opiniões do Analista referentes ao Problema do Envolvido (validar ou invalidar suposições)

- *[Se não tiverem sido abordados]* Que problemas, se houver, estão associados a:

[Liste todas as necessidades ou problemas adicionais que você acha que

dizem respeito ao envolvido ou ao usuário]

- Pergunte em relação a cada problema sugerido:
- Trata-se de um problema real?
- Quais são as razões que explicam este problema?
- Como é possível solucioná-lo no momento?
- Como você gostaria de solucioná-lo?
- Como você classificaria esses problemas em comparação com outros que você mencionou?

## 7. Avaliação de Sua Solução (se aplicável)

- E se você pudesse... [resuma os principais aspectos da solução proposta por você]
- Como você classificaria a importância desses fatores?

## 8. Avaliação da Oportunidade

- Quem necessita desse aplicativo em sua organização?
- Quantos dentre esses tipos de usuários utilizariam o aplicativo?
- Como você avaliaria uma solução eficaz?

## 9. Avaliação da Confiabilidade, do Desempenho e das Necessidades de Suporte

Quis são as suas expectativas em relação à confiabilidade?

Quis são as suas expectativas em relação ao desempenho?

- Você dará suporte ao produto ou isso será feito por outras pessoas?
- Você tem necessidades especiais de suporte? E o acesso a serviços e à manutenção?
- Quais são os requisitos de segurança?
- Quais são os requisitos de instalação e de configuração?
- Quais são os requisitos especiais de licenciamento?
- Como o *software* será distribuído?
- Quais são os requisitos de rotulação e de empacotamento?

Outros Requisitos

- Se houver, quais são os requisitos ou padrões reguladores ou ambientais que deverão ser suportados?
- Você se recorda de mais algum requisito que devamos conhecer?

## 10. Conclusão

- Há outras perguntas que eu poderia fazer a você?
- Se eu precisar fazer perguntas de acompanhamento, posso lhe telefonar?
- Você gostaria de participar de uma revisão de requisitos?

## 11. Resumo do Analista

[Resuma a seguir os três ou quatro problemas de maior prioridade para o usuário/envolvido]

ANEXO 03 – GLOSSÁRIO DO RUP

# Glossário

## 1. Introdução

[A introdução do Glossário fornece uma visão geral de todo o documento. Apresente todas as informações que poderão ser necessárias para que o leitor compreenda o documento nesta seção. Este documento é usado para definir a terminologia específica do domínio do problema, explicando termos que podem ser desconhecidos do leitor das descrições de caso de uso ou de outros documentos do projeto. Freqüentemente, este documento poderá ser usado como um dicionário de dados informal, capturando definições de dados para que as descrições de caso de uso e outros documentos do projeto possam concentrar-se no que o sistema deve fazer com as informações. Este documento deverá ser salvo em um arquivo denominado Glossário.]

### 1.1. Finalidade

[Especifique a finalidade deste Glossário.]

### 1.2. Escopo

[Uma breve descrição do escopo deste Glossário; a que Projeto(s) ele está associado e tudo o mais que seja afetado ou influenciado por este documento.]

### 1.3. Referências

[Esta subseção fornece uma lista completa de todos os documentos mencionados em qualquer outra parte do Glossário. Identifique cada documento por título, número do relatório (se aplicável), data e organização de publicação. Especifique as fontes a partir das quais as referências podem ser obtidas. Essas informações podem ser fornecidas por um anexo ou outro documento.]

### 1.4. Visão Geral

[Esta subseção descreve o que o restante do Glossário contém e explica como o documento está organizado.]

## 2. Definições

[Os termos definidos aqui são a essência do documento. Eles poderão ser definidos na ordem desejada, mas geralmente a ordem alfabética propicia maior acessibilidade.]

## 2.1. <aTerm>

[A definição de <aTerm> é apresentada aqui. Você deverá apresentar quantas informações forem necessárias para que o leitor compreenda o conceito.]

## 2.2. <anotherTerm>

A definição de <anotherTerm> é apresentada aqui. Você deverá apresentar quantas informações forem necessárias para que o leitor compreenda o conceito

## 2.3. <aGroupofTerms>

[Às vezes, é útil organizar os termos em grupos para aprimorar a legibilidade. Por exemplo, se o domínio do problema contiver termos relacionados a contabilidade e a construção de prédios (como seria o caso se estivéssemos desenvolvendo um sistema para gerenciar projetos de construção), apresentar os termos dos dois subdomínios diferentes poderá gerar confusão para o leitor. Para resolver o problema, utilizamos agrupamentos de termos. Ao apresentar os agrupamentos de termos, forneça uma breve descrição que ajude o leitor a compreender o que <aGroupOfTerms> representa. Os termos apresentados no grupo deverão ser organizados alfabeticamente para possibilitar um fácil acesso.]

### 2.3.1. <aGroupTerm>

[A definição de <aGroupTerm> é apresentada aqui. Apresente quantas informações forem necessárias para que o leitor compreenda o conceito.]

### 2.3.2. <anotherGroupTerm>

[A definição de <anotherGroupTerm> é apresentada aqui. Apresente quantas informações forem necessárias para que o leitor compreenda o conceito.]

## 2.4. <aSecondGroupofTerms>

### 2.4.1. <yetAnotherGroupTerm>

[A definição do termo é apresentada aqui. Apresente quantas informações forem necessárias para que o leitor compreenda o conceito.]

### 2.4.2. <andAnotherGroupTerm>

[A definição do termo é apresentada aqui. Apresente quantas informações forem necessárias para que o leitor compreenda o conceito.]

# 3. Estereótipos de UML

[Esta seção contém ou faz referência a especificações de estereótipos de Linguagem de Modelagem Unificada (UML) e suas implicações semânticas - uma descrição textual do significado e da importância dos estereótipos e de quaisquer limitações de seu uso - para estereótipos já conhecidos ou que se mostraram importantes para o sistema que está sendo modelado. O uso desses estereótipos pode ser simplesmente recomendado ou até mesmo obrigatório; por exemplo, quando o uso desses estereótipos for exigido por um padrão imposto ou quando se considerar que o uso facilitará em muito o entendimento. Esta seção poderá ficar em branco se não for considerado necessário nenhum estereótipo adicional além dos predefinidos pela UML e pelo Rational Unified Process.]

ANEXO 04 –DOCUMENTO DE VISÃO DO RUP

# <Nome do Projeto >
# Visão

## 1. Introdução

## 2. Circunstâncias

### 2.1 Apresentação do Problema

[Apresenta um resumo dos problemas a serem resolvidos pelo sistema. Pode-se usar o seguinte formato:]

| | |
|---|---|
| O problema de | [descreva o problema] |
| Afeta | [as partes afetadas pelo problema] |
| Cujo impacto é | [qual é o impacto do problema?] |
| Uma solução seria: | [liste alguns benefícios de uma solução satisfatória] |

## 3. Descrição das Partes Envolvidas

### 3.1 Resumo das Partes Envolvidas

| Nome | Descrição | Responsabilidades |
|---|---|---|
| [Nome da parte envolvida (função).] | [Breve descrição.] | [Resume suas responsabilidades chaves levando em consideração o sistema a ser desenvolvido] |

### 3.2 Ambiente do Usuário

[Detalhe o ambiente de trabalho dos usuários. Algumas sugestões:

Número de pessoas envolvidas em toda a tarefa? Tem-se mudado?

Quanto dura um ciclo da tarefa? Tempo gasto em cada atividade? Tem-se mudado?

Alguma restrição quanto ao ambiente: móvel, externo, etc?

Qual plataforma é usada hoje em da? Plataformas futuras?

Outras aplicações são usadas? Sua aplicação precisa se integrar com as demais ?

# 4. Visão Geral do Produto

## 4.1 Perspectiva do Produto

[Esta sessão do documento de visão coloca o produto em perspective com outros produtos e ambientes dos usuários.Se o produto é independente escreva aqui.

## 4.2 Suposições e Dependências

[Liste cada fator que afeta as características apresentadas no documento de visão.Liste fatores que, se alterados afetam o documento de visão.

# 5. Outros Requisitos

[Em alto nível, liste padrões aplicáveis, equipamentos ou plataformas necessárias;performance requerida e ambiente requerido.

Defina os limites de qualidade para desempenho, robustez, tolerância a falha, usabilidadee características similares que não são capturadas no conjunto de requisitos.

Anote restrições de design,restrições externas ou outras dependências.

Defina qualquer requisito específico de documentação, incluindo manual de usuário, ajuda on-line, instalação e criação de pacote.

Defina a prioridade desses outros requisitos de produtos. Inclua, se necessário, atributos tais como estabilidade, benefícios, esforço e riscos.]

# <Nome do Projeto>
# Especificação Suplementar
# Versão <1.0>

## 1. Introdução

[A introdução da Especificação Suplementar deve fornecer uma visão geral de todo o documento. Ela deve incluir a finalidade, o escopo, as definições, os acrônimos, as abreviações, as referências e a visão geral desta Especificação Suplementar.]

A Especificação Suplementar captura os requisitos de sistema que não são capturados imediatamente nos casos de uso do modelo de casos de uso. Entre os requisitos estão incluídos:

- Requisitos legais e reguladores, incluindo padrões de aplicativo.

Atributos de qualidade do sistema a ser criado, incluindo requisitos de usabilidade, confiabilidade, desempenho e suportabilidade.

Outros requisitos, como sistemas operacionais e ambientes, requisitos de compatibilidade e restrições de design.]

### 1.1 Finalidade

[Especifique a finalidade desta Especificação Suplementar.]

### 1.2 Escopo

[Uma breve descrição do escopo desta Especificação Suplementar; a que Projeto(s) ela está associada e tudo o mais que seja afetado ou influenciado por este documento.]

### 1.3 Definições, Acrônimos e Abreviações

[Esta subseção deve fornecer as definições de todos os termos, acrônimos e abreviações necessárias à adequada interpretação da Especificação Suplementar. Essas informações podem ser fornecidas mediante referência ao Glossário do projeto.]

### 1.4 Referências

[Esta subseção deve fornecer uma lista completa de todos os documentos

mencionados em qualquer outra parte da Especificação Suplementar. Cada documento deve ser identificado por título, número do relatório (se aplicável), data e organização de publicação. Especifique as fontes a partir das quais as referências podem ser obtidas. Essas informações podem ser fornecidas por um anexo ou outro documento.]

### 1.5 Visão Geral

[Esta subseção deve descrever o que o restante da Especificação Suplementar contém e deve explicar como o documento está organizado.]

## 2. Funcionalidade

[Esta seção descreve os requisitos funcionais do sistema que são expressos no estilo de linguagem natural. Para muitos aplicativos, isso poderá constituir o volume do Pacote SRS e deve-se refletir muito para organizar esta seção. Normalmente, ela é organizada por recurso, mas métodos de organização alternativos como, por exemplo, organização por usuário ou organização por subsistema, também podem ser apropriados. Entre os requisitos funcionais podem estar incluídos conjuntos de recursos, capacidades e segurança.

Quando as ferramentas de desenvolvimento de aplicativos, como ferramentas de requisitos, ferramentas de modelagem, entre outras, forem utilizadas para capturar a funcionalidade, esta seção fará referência à disponibilidade desses dados, indicando o local e o nome da ferramenta usada para capturar os dados.]

### 2.1 <Requisito Funcional Um>

[A descrição do requisito deve ser feita aqui.]

## 3. Usabilidade

[Esta seção deve incluir todos os requisitos que afetam a usabilidade. Estes são alguns exemplos:

- especifique o tempo de treinamento necessário para que usuários normais e usuários com conhecimentos avançados se tornem produtivos em operações específicas
- especifique períodos de tempo mensuráveis para tarefas típicas ou
- especifique requisitos que estejam em conformidade com os padrões comuns de usabilidade como, por exemplo, os padrões CUA da IBM ou os padrões GUI da Microsoft]

### 3.1 <Requisito de Usabilidade Um>

A descrição do requisito.

## 4. Confiabilidade

[Os requisitos de confiabilidade do sistema devem ser especificados aqui. Abaixo, algumas sugestões:

- Disponibilidade - especifique a porcentagem de tempo disponível ( xx.xx%), as horas de uso, o acesso à manutenção, as operações de modo degradado etc.

- Tempo Médio entre Falhas (MTBF) - normalmente especificado em horas, mas também poderá ser especificado em termos de dias, meses ou anos.

- Tempo Médio para Reparo (MTTR) - quanto tempo o sistema poderá ficar sem funcionar após uma falha?

- Exatidão - especifique a precisão (resolução) e exatidão (através de algum padrão conhecido) necessárias na saída dos sistemas.

- Taxa máxima de erros ou defeitos - geralmente expressa em termos de erros/KLOC (thousands of lines of code, milhares de linhas de código) ou de erros/ponto de função.

- Taxa de erros ou defeitos - categorizados em termos de erros pouco importantes, importantes e críticos: o(s) requisito(s) devem definir o que se entende por um erro "crítico" (ex: perda total de dados ou total incapacidade de usar determinadas partes da funcionalidade do sistema).]

### 4.1 <Requisito de Confiabilidade Um>

[A descrição do requisito deve ser feita aqui.]

## 5. Desempenho

[As características de desempenho do sistema devem ser descritas nesta seção. Inclua tempos de resposta específicos. Quando aplicável, faça referência, por nome, aos Casos de Uso relacionados.

- Tempo de resposta de uma transação (médio, máximo)

- Taxa de transferência (ex: transações por segundo)

- Capacidade (ex: o número de clientes ou de transações que podem ser acomodados pelo sistema)

- Modos de degradação (o modo aceitável de operação quando o sistema tiver sido degradado de alguma maneira)

- Utilização de recursos: memória, disco, comunicações etc.]

### 5.1 <Requisito de Desempenho Um>

[A descrição do requisito deve ser feita aqui.]

## 6. Suportabilidade

[Esta seção indica todos os requisitos que aprimorarão a suportabilidade ou manutenibilidade do sistema que está sendo criado, incluindo padrões de codificação, convenções de nomeação, bibliotecas de classes, acesso à manutenção e utilitários de manutenção.]

### 6.1 <Requisito de Suportabilidade Um>

[A descrição do requisito deve ser feita aqui.]

## 7. Restrições de Design

[Esta seção deve indicar todas as restrições de design referentes ao sistema que está sendo criado. As restrições de design representam decisões de design que foram impostas e devem ser obedecidas. Entre os exemplos desse tipo de restrição estão linguagens de software, requisitos de processo de software, uso prescrito de ferramentas de desenvolvimento, restrições de design e de arquitetura, componentes comprados, bibliotecas de classes etc.]

### 7.1 <Restrição de Design Um>

[A descrição do requisito deve ser feita aqui.]

## 8. Requisitos de Sistema de Ajuda e de Documentação de Usuário On-line

[Descreva os requisitos, se houver, de documentação de usuário on-line, sistemas de ajuda, observações sobre ajuda etc.]

## 9. Componentes Comprados

[Esta seção descreve todos os documentos comprados a serem usados no sistema, quaisquer restrições de utilização ou de licenciamento aplicáveis e

quaisquer padrões associados de compatibilidade/interoperabilidade ou de interface.]

# 10. Interfaces

[Esta seção define as interfaces que devem ser suportadas pelo aplicativo. Ela deve conter especificidades, protocolos, portas e endereços lógicos adequados, entre outros, para que o software possa ser desenvolvido e verificado em relação aos requisitos de interface.]

## 10.1 Interfaces de Usuário

[Descreva as interfaces de usuário que deverão ser implementadas pelo software.]

## 10.2 Interfaces de Hardware

[Esta seção define todas as interfaces de hardware que devem ser suportadas pelo software, incluindo a estrutura lógica, os endereços físicos, o comportamento esperado etc. ]

## 10.3 Interfaces de *Software*

[Esta seção descreve as interfaces de software com outros componentes do sistema de software. Poderão ser componentes comprados, componentes reutilizados de outro aplicativo ou componentes que estão sendo desenvolvidos para subsistemas fora do escopo desta SRS, mas com os quais esse aplicativo de software deve interagir.]

## 10.4 Interfaces de Comunicações

[Descreva todas as interfaces de comunicações com outros sistemas ou dispositivos como, por exemplo, redes locais, dispositivos seriais remotos etc.]

# 11. Requisitos de Licenciamento

[Esta seção define todos os requisitos de imposição de licenciamento ou outros requisitos de restrição de utilização que devem ser exibidos pelo software.]

# 12. Observações Legais, de Direitos Autorais etc

[Esta seção descreve todos os avisos legais necessários, garantias, observações sobre direitos autorais, observações sobre patentes, logomarcas, marcas comerciais ou problemas de conformidade com logotipos referentes ao

software.]

# 13. Padrões Aplicáveis

[Esta seção descreve, por meio de referências, todos os padrões aplicáveis e as seções específicas desses padrões que se aplicam ao sistema que está sendo descrito. Entre esses padrões estão incluídos, por exemplo, padrões reguladores, de qualidade e legais, padrões de indústria referentes à usabilidade, interoperabilidade, internacionalização, compatibilidade com o sistema operacional etc.]

# <Nome do Projeto>
# Especificação de Requisitos de *Software*
# Para <Subsistema ou Recurso>
# Versão <1.0>

## 1. Introdução

[A introdução da Especificação de Requisitos de Software (SRS) deve fornecer uma visão geral de todo o documento. Ela deve incluir a finalidade, o escopo, as definições, os acrônimos, as abreviações, as referências e a visão geral da Especificação de Requisitos de Software.]

[Observação: A Especificação de Requisitos de Software captura todos os requisitos de software do sistema ou de uma parte do sistema. A seguir, há um esquema de uma típica Especificação de Requisitos de Software para um projeto utilizando a modelagem de casos de uso. Esse artefato consiste em um pacote contendo casos de uso do modelo de casos de uso, Especificações Suplementares aplicáveis e outras informações de suporte. Para ter acesso a um template de uma Especificação de Requisitos de Software que não utilize a modelagem de casos de uso e que capture todos os requisitos em um único documento, com as seções aplicáveis inseridas a partir das Especificações Suplementares (que não seriam mais necessárias), consulte o arquivo rup_srs.dot.]

[É possível organizar a Especificação de Requisitos de Software de várias maneiras diferentes. Consulte o padrão [IEEE830-1998] para obter explicações mais detalhadas, assim como outras opções de organização para uma Especificação de Requisitos de Software.]

### 1.1 Finalidade

[Especifique a finalidade desta Especificação de Requisitos de Software. A Especificação de Requisitos de Software deve descrever totalmente o comportamento externo do aplicativo ou do subsistema identificado. Ela também deverá descrever requisitos não funcionais, restrições de design e outros fatores necessários para fornecer uma visão completa e abrangente dos requisitos do software.]

### 1.2 Escopo

[Uma breve descrição do aplicativo de software a que se aplica a Especificação de Requisitos de Software; o recurso ou outro agrupamento de subsistemas; a que modelo(s) de caso de uso a Especificação de Requisitos está associada; e tudo o mais que seja afetado ou influenciado por este documento.]

## 1.3 Definições, Acrônimos e Abreviações

[Esta subseção deve fornecer as definições de todos os termos, acrônimos e abreviações necessárias à adequada interpretação da Especificação de Requisitos de Software. Essas informações podem ser fornecidas mediante referência ao Glossário do projeto.]

## 1.4 Referências

[Esta subseção deve fornecer uma lista completa de todos os documentos mencionados em qualquer outra parte da Especificação de Requisitos de Software. Cada documento deverá ser identificado por título, número do relatório (se aplicável), data e organização de publicação. Especifique as fontes a partir das quais as referências podem ser obtidas. Essas informações podem ser fornecidas por um anexo ou outro documento.]

## 1.5 Visão Geral

[Esta subseção deve descrever o que o restante da Especificação de Requisitos de Software contém e explica como o documento está organizado.]

# 2. Descrição Geral

[Esta seção da Especificação de Requisitos de Software deve descrever os fatores gerais que afetam o produto e seus requisitos. Ela não deve definir requisitos específicos. Em vez disso, deve fornecer uma base para esses requisitos, que serão definidos detalhadamente na Seção 3, e facilitar sua compreensão. Inclua itens como perspectiva e funções do produto, características do usuário, restrições, suposições e dependências, e subconjuntos de requisitos.]

## 2.1 Relatório Sintético de Modelo de Casos de Uso

[Se a modelagem de caso de uso for utilizada, esta seção conterá uma visão geral do modelo de casos de uso ou do subconjunto do modelo de casos de uso aplicável a esse subsistema ou recurso. Estará incluída uma lista de nomes e breves descrições de todos os atores e casos de uso, juntamente com os diagramas e relacionamentos aplicáveis. Consulte o Relatório Sintético de Modelo de Casos de Uso, que poderá ser usado como um anexo nesse ponto.]

## 2.2 Suposições e Dependências

[Esta seção descreve a possibilidade de execução de quaisquer recursos técnicos importantes, a disponibilidade dos componentes ou dos subsistemas, ou outras suposições relacionadas ao projeto em que poderá se basear a viabilidade do software descrita por esta Especificação de Requisitos de Software.]

# 3. Requisitos Específicos

[Esta seção da Especificação de Requisitos de Software deve conter todos os requisitos de software em um nível de detalhamento suficiente para possibilitar que os designers projetem um sistema que satisfaça esses requisitos e que os testadores verifiquem se o sistema satisfaz esses requisitos. Quando for utilizada a modelagem de casos de uso, esses requisitos serão capturados nos casos de uso e nas especificações suplementares aplicáveis. Se a modelagem de casos de uso não for utilizada, o esquema das especificações suplementares poderá ser inserido diretamente nesta seção.]

## 3.1 Relatórios de Caso de Uso

[Na modelagem de casos de uso, freqüentemente os casos de uso definem a maior parte dos requisitos funcionais do sistema, juntamente com alguns requisitos não funcionais. Para cada caso de uso do modelo de casos de uso acima, ou de seu subconjunto, faça referência ao relatório de caso de uso ou inclua-o nesta seção. Certifique-se de que cada requisito esteja claramente nomeado.]

## 3.2 Requisitos Suplementares

[As Especificações Suplementares capturam os requisitos que não estão incluídos nos casos de uso. Os requisitos específicos das Especificações Suplementares, que se aplicam ao subsistema ou ao recurso, devem ser incluídos aqui e especificados de acordo com o nível de detalhamento necessário para descrever esse subsistema ou recurso. Esses requisitos poderão ser capturados diretamente no documento ou mencionados como Especificações Suplementares individuais, que poderão ser usadas como um anexo nesse ponto. Certifique-se de que cada requisito esteja nomeado claramente.]

# 4. Informações de Suporte

[As informações de suporte facilitam o uso da Especificação de Requisitos de Software. Elas incluem:

- Índice Analítico
- Índice
- Apêndices

Poderão estar incluídos roteiros de caso de uso ou protótipos de interface do usuário. Quando forem incluídos apêndices, a Especificação de Requisitos de Software deverá especificar explicitamente se eles deverão ou não ser considerados parte integrante dos requisitos.]

ANEXO 07 – ESPECIFICAÇÃO DE CASO DE USO DO RUP

# <Nome do Projeto>
# Especificação de Caso de Uso: <Nome do Caso de Uso>
### Versão <1.0>

[O template a seguir é fornecido para uma Especificação de Caso de Uso, que contém as propriedades de texto do caso de uso. Este documento é usado com uma ferramenta de gerenciamento de requisitos, como o Rational RequisitePro, para especificar e marcar os requisitos contidos nas propriedades do caso de uso.

Os diagramas do caso de uso podem ser desenvolvidos em uma ferramenta de modelagem visual, como o Rational Rose. Um relatório de caso de uso (com todas as propriedades) pode ser gerado com o Rational SoDA. Para obter mais informações, consulte os mentores de ferramentas do Rational Unified Process.]

## 1.1 Breve Descrição

[A descrição relata brevemente a finalidade do caso de uso. Para tanto, será suficiente um único parágrafo.]

# 2. Fluxo de Eventos

## 2.1 Fluxo Básico

[Este caso de uso é iniciado quando o ator faz algo. Um ator sempre inicia os casos de uso. O caso de uso descreve o que o ator faz e o que o sistema faz em resposta. Ele deve ser elaborado como um diálogo entre o ator e o sistema.

O caso de uso descreve o que acontece dentro do sistema, mas não o porquê nem como. Se forem trocadas informações, seja específico no que diz respeito ao conteúdo que é passado e retornado. Por exemplo, não é muito esclarecedor dizer que o ator fornece informações do cliente. É melhor dizer que ele fornece o nome e o endereço do cliente. Freqüentemente é útil fazer uso de um Glossário de Termos para manter a complexidade do caso de uso sob controle — poderá ser conveniente definir termos como, por exemplo, informações do cliente nesse glossário, a fim de evitar que o caso de uso fique repleto de detalhes.

As alternativas simples poderão ser apresentadas no texto do caso de uso. Se forem necessárias apenas algumas frases para descrever o que acontece quando há uma alternativa, faça essa descrição diretamente na seção Fluxo de Eventos. Se o fluxo alternativo for mais complexo, use uma seção separada para descrevê-lo. Por exemplo, uma subseção Fluxo Alternativo explica como descrever alternativas mais complexas.

Às vezes, uma figura vale por mil palavras, embora não haja nada que possa substituir uma redação clara e organizada. Se for mais esclarecedor, sinta-se à vontade para colar representações gráficas de interfaces do usuário, fluxos de processo ou outras imagens no caso de uso. Se um fluxograma for útil para apresentar um processo complexo de decisões, utilize-o sem nenhuma dúvida! Semelhantemente no que diz respeito a comportamentos dependentes de estado, um diagrama de transição de estado geralmente esclarece o comportamento de um sistema muito mais do que páginas e páginas de texto. Use o meio de apresentação certo para o problema, mas procure evitar o uso de terminologia, notações ou imagens que o público possa deixar de compreender. Lembre-se de que sua finalidade é esclarecer e não obscurecer.]

## 2.2 Fluxos Alternativos

### 2.2.1 < Primeiro Fluxo Alternativo >

[As alternativas mais complexas são descritas em uma seção separada, a que é feita referência na subseção Fluxo Básico da seção Fluxo de Eventos. Pense nas subseções Fluxo Alternativo como comportamentos alternativos — cada fluxo alternativo representa um comportamento alternativo geralmente devido a exceções que ocorrem no fluxo principal. O tamanho desses fluxos poderá ser tão extenso quanto o necessário para descrever os eventos associados ao comportamento alternativo. Quando um fluxo alternativo termina, os eventos do fluxo principal de eventos são retomados a menos que seja especificado de outra maneira.]

#### 2.2.1.1 < Um Subfluxo Alternativo >

[Os subfluxos alternativos, por sua vez, poderão ser divididos em subseções para aprimorar a clareza.]

### 2.2.2 < Segundo Fluxo Alternativo >

[Poderá haver, e muito provavelmente haverá, uma série de fluxos alternativos em um caso de uso. Mantenha cada fluxo alternativo separado para aprimorar a clareza. O uso de fluxos alternativos melhora a legibilidade do caso de uso e também evita que os casos de uso sejam decompostos em hierarquias de casos de uso. Lembre-se de que os casos de uso são apenas descrições textuais e que sua finalidade principal é documentar o comportamento de um sistema de maneira clara, concisa e compreensível.]

# 3. Requisitos Especiais

[Normalmente um requisito especial é um requisito não funcional que é específico de um caso de uso, mas que não é especificado, de maneira fácil ou natural, no texto do fluxo de eventos do caso de uso. Entre os exemplos de requisitos especiais estão incluídos requisitos legais e reguladores, padrões de aplicativo e atributos de qualidade do sistema a ser criado incluindo requisitos de usabilidade, confiabilidade, desempenho ou suportabilidade. Além disso, outros

requisitos — como sistemas operacionais e ambientes, requisitos de compatibilidade e restrições de design — deverão ser capturados nesta seção.]

### 3.1 < Primeiro Requisito Especial >

## 4. Condições Prévias

[Uma condição prévia de um caso de uso é o estado do sistema que deve estar presente antes de um caso de uso ser realizado.]

### 4.1 < Condição Prévia Um >

## 5. Condições Posteriores

[Uma condição posterior de um caso de uso é uma lista dos possíveis estados em que o sistema poderá se encontrar imediatamente depois do término de um caso de uso.]

### 5.1 < Condição Posterior Um >

## 6. Pontos de Extensão

[Pontos de extensão do caso de uso.]

### 6.1 <Nome do Ponto de Extensão>

[Definição da localização do ponto de extensão no fluxo de eventos.]